AN ABSTRACT OF THE DISSERTATION OF

Dylan B. Keon for the degree of Doctor of Philosophy in Geography presented on November 16, 2012.
Title: Automated Web-based Analysis and Visualization of Spatiotemporal Data.

Abstract approved: ____
Dawn J. Wright

Most data are associated with a place, and many are also associated with a moment in time, a time interval, or another linked temporal component. Spatiotemporal data (i.e., data with elements of both space and time) can be used to assess movement or change over time in a particular location, an approach that is useful across many disciplines. However, spatiotemporal data structures can be quite complex, and the datasets very large. Although GIS software programs are capable of processing and analyzing spatial information, most contain no (or minimal) features for handling temporal information and have limited capability to deal with large, complex multidimensional spatiotemporal data. A related problem is how to best represent spatiotemporal data to support efficient processing, analysis, and visualization.

In the era of "big data," efficient methods for analyzing and visualizing large quantities of spatiotemporal data have become increasingly necessary. Automated processing approaches, when made scalable and generalizable, can result in much greater efficiency in spatiotemporal data analysis. The growing popularity of web services and server-side processing methods can be leveraged to create systems for processing spatiotemporal data on the server, with delivery of output products to the client. In many cases, the client can be a standard web browser, providing a common platform from which users can interact with complex server-side processing systems to produce specific output data and visualizations. The rise of complex JavaScript libraries for creating interactive client-side tools has enabled the development of rich internet applications (RIA) that provide interactive data exploration capabilities and an enhanced user experience within the web browser.

Three projects involving time-series tsunami simulation data, potential human response in a tsunami evacuation scenario, and large sets of modeled time-series climate grids were conducted to explore automated web-based analysis, processing, and visualization of spatiotemporal data. Methods were developed for efficient handling of spatiotemporal data on the server side, as well as for interactive animation and visualization tools on the client side. The common web browser, particularly when combined with specialized server side code and client side RIA libraries, was found to be an effective platform for analysis and visualization tools that quickly interact with complex spatiotemporal data. Although specialized methods were developed to for each project, in most cases those methods can be generalized to other disciplines or computational domains where similar problem sets exist.

Automated Web-based Analysis and Visualization of Spatiotemporal Data

by
Dylan B. Keon

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented November 16, 2012
Commencement June 2013

Doctor of Philosophy dissertation of Dylan B. Keon presented on November 16, 2012.

APPROVED:

Major Professor, representing Geography

Dean of the College of Earth, Ocean, and Atmospheric Sciences

---

Dean of the Graduate School

Dylan B. Keon, Author

ACKNOWLEDGEMENTS

The author expresses sincere appreciation for the guidance provided by his advisor Dr. Dawn J. Wright and his committee members, the encouragement given by his colleagues at NACSE and the PRISM Climate Group, and the heartfelt support provided by friends and family. Most of all, the author expresses deep gratitude to Karen, Julia, and Anna, who endured his absence over many long nights and weekends and provided unwavering love and support during the process.

CONTRIBUTION OF AUTHORS

Cherri Pancake and Harry Yeh procured funding for the projects described in Chapters 2 and 3, and provided guidance in their implementation. Cherri Pancake, Harry Yeh, and Dawn Wright assisted with editing of Chapters 2 and 3. Chris Daly assisted with interface design guidance and editing of Chapter 4. Ben Steinberg contributed to the development of the TCP and grid export code described in Chapter 2, and the development and implementation of the server-side simulation code described in Chapter 3. Adam Ryan contributed to the development and implementation of the server-side grid processing code described in Chapter 4.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF TABLES

LIST OF TABLES (Continued)

LIST OF APPENDICES

LIST OF ACRONYMS

1D	One Dimensional
2D	Two Dimensional
3D	Three Dimensional
ADCIRC	A [Parallel] Advanced Circulation Model for Oceanic, Coastal and Estuarine Waters
AJAX	Asynchronous Javascript and XML
API	Application Programming Interface
ARSC	Arctic Region Supercomputing Center
ASCII	American Standard Code for Information Interchange
BIL	Band Interleaved by Line
CGI	Common Gateway Interface
COMCOT	Cornell Multi-grid Coupled Tsunami Model
ComMIT	Community Model Interface for Tsunami
CONUS	Conterminous United States
CPU	Central Processing Unit
CSV	Comma Separated Value
DB	Database
DEM	Digital Elevation Model
DoD	Department of Defense
DOM	Document Object Model
EC2	Elastic Compute Cloud
EHdr	ESRI .hdr Labeled
ENSO	El Niño-Southern Oscillation
ENVI	Environment for Visualization of Imagery
EPSG	European Petroleum Survey Group
ESRI	Environmental Systems Research Institute
ETOPO	Earth Topography Digital Dataset
GB	Gigabyte
GDAL	Geospatial Data Abstraction Layer
GEBCO	General Bathymetric Chart of the Oceans
GeoJSON	Geographic JavaScript Object Notation
GEOS	Geometry Engine – Open Source
GHz	Gigahertz
GIS	Geographic Information Systems
GISci	Geographic Information Science

LIST OF ACRONYMS (Continued)

GiST	Generalized Search Tree
GML	Geographic Markup Language
GMT	Generic Mapping Tools
GPS	Global Positioning System
GPU	Graphics Processing Unit
GRASS	Geographic Resources Analysis Support System
HPCMP	High Performance Computing Modernization Program
HTML	Hypertext Markup Language
HTML5	Hypertext Markup Language – fifth revision
HTTP	Hypertext Transfer Protocol
HUC	Hydrologic Unit Code
JSON	JavaScript Object Notation
KB	Kilobyte
KML	Keyhole Markup Language
LiDAR	Light Detection and Ranging
LSW	Linear Shallow Water
MB	Megabyte
MOST	Method of Splitting Tsunami
NACSE	Northwest Alliance for Computational Science & Engineering
NAS	Network Attached Storage
NED	National Elevation Dataset
NetCDF	Network Common Data Form
NFS	Network File System
NGDC	National Geospatial Data Committee
NLSW	Nonlinear Shallow Water
NOAA	National Oceanographic and Atmospheric Administration
NumPy	Numerical Tools for Python
NWS	National Weather Service
OGC	Open Geospatial Consortium
OGR	OGR Simple Features Library
OLAP	Online Analytical Processing
OSU	Oregon State University
PCA	Principal Components Analysis
PHP	PHP: Hypertext Preprocessor
PLSS	Public Lands Survey System

LIST OF ACRONYMS (Continued)

PPT	Precipitation
PRISM	Parameter-elevation Regressions on Independent Slopes Model
PTVA	Papathoma Tsunami Vulnerability Assessment
RAID	Redundant Array of Inexpensive Disks
RDBMS	Relational Database Management System
REST	Representational State Transfer
RHEL	Red Hat Enterprise Linux
RIA	Rich Internet Applications
ROMS	Regional Ocean Modeling System
RPM	Revolutions per Minute
SAS	Serial Attached SCSI
SciPy	Scientific Tools for Python
SCSI	Small Computer System Interface
SQL	Structured Query Language
SVG	Scalable Vector Graphics
TB	Terabyte
TCP	Tsunami Computational Portal
TIFF	Tagged Image File Format
TMAX	Maximum Temperature
TMEAN	Mean Temperature
TMIN	Minimum Temperature
TUNAMI-N2	Tohoku University’s Numerical Analysis Model for Investigation of Near field tsunamis
UAF	University of Alaska, Fairbanks
UCGIS	University Consortium for Geographic Information Science
UNIX	Uniplexed Information and Computing System
USGS	United States Geological Survey
WCS	Web Coverage Service
WebGL	Web-based Graphics Library
WFS	Web Feature Service
WGS84	World Geodetic Survey 1984
WMS	Web Mapping Service
WSGI	Web Server Gateway Interface
XML	Extensible Markup Language

This dissertation is dedicated to my daughters Julia and Anna,
and to the memory of my grandfather, Dr. Lester E. Eyer

Chapter 1: Introduction

Nearly all data possess a spatial component. The proliferation and accessibility of easy-to-use mapping tools such as Google Maps (2012) and Google Earth (2012), combined with (often automatic) spatial application features (e.g., geotagging, location-based services, smartphone GPS mapping capabilities), has thrust technology that leverages the spatial component into a prominent position of reliance and everyday usage. Perhaps less prominent, but equally present and important, is the element of time. Many data possess a temporal component (or have the *potential* for a linked temporal component) in addition to a spatial component, such as the time at which a photograph was taken, the recorded time of a temperature observation, or the time step represented by a video frame in a dynamic simulation. However, the temporal component is generally less well-supported than the spatial component in GIS and mapping software.

Geographic information systems (GISs) include well-established methods for displaying, manipulating, and analyzing data that contain a spatial component. Basic usage and manipulation of spatial datasets in GIS is relatively easy to understand. However, because most GIS software packages do not have the capability to analyze and visualize data that represent dynamic phenomena (or have limited capabilities for doing so), they are not able to effectively manage spatial data that also contain a temporal component (i.e., spatiotemporal data). A GIS that can evaluate spatial data but not temporal data can only display information about one slice in time for the area of interest and, therefore, provides limited information about dynamic events or processes within the area of interest.

Spatiotemporal datasets are often rich in content by nature. They can describe the movement of an individual or a group of individuals over a period of weeks (e.g., Wen *et al.* 2012), the change in forest composition over hundreds of years (e.g., McLachlan *et al.* 2005), or, indeed, virtually any feature across any time interval. Analysis and visualization software must be able to effectively represent and display potentially complex spatiotemporal datasets in a manner that allows users to easily interpret them, explore them, and identify subsets of information for further analysis. Geovisualization tools (e.g., Buckley *et al.* 2005, Kraak 2008) are well-suited for this approach, but the three-dimensional (3D) nature of spatiotemporal datasets can make them difficult to understand by non-technical users working with them in 3D geovisualization applications (Zhong *et al.* 2012). Adding to their complexity, spatiotemporal datasets are often very large in size, requiring significant storage space, processing capabilities, and effective database storage schemes.

1.1 Spatiotemporal Data Representation and Software Tools

The representation of spatiotemporal data is an important consideration – not only for how the data are communicated, but also how they are stored digitally or, in other words, "the binary structure in a computer or electronic storage medium that corresponds with an object, measurement, or phenomenon in the world" (Yuan *et al.*

2005). In the late 1980s, a considerable amount of research examined the representation of spatiotemporal data. Langran and Chrisman (1988) were the first to describe a detailed framework for incorporating temporal data in a GIS. In the following years a number of additional models were proposed and implemented, beginning with file-based approaches that became known as the "snapshot model" (e.g., Armstrong 1988). In the snapshot model, temporal information is attached to spatial information, such that the entire geographic representation is duplicated at each time slice, with changes in the phenomena at that geographic area also represented. Beller *et al.* (1991) proposed the Temporal Map Sets (TMS) model, an extension of the snapshot model where each geographic cell is time-stamped and is considered to be either in or out of the event. Later, Langran (1992) published an influential work that recognized the upward trajectory of GIS, its limitations in terms of analyzing dynamically-changing events, and fully explored methods for the inclusion of temporal data in GIS.

The representation of geographic data significantly affects all three levels of GIS and spatial analysis: Data modeling, formalization, and visualization (Yuan *et al.* 2005), and has been examined, described, and summarized in many forms (e.g., Peuquet 1984, Yuan 2001, Goodchild *et al.* 2007). Although numerous studies have examined the representation of temporal data for use in GIS and spatiotemporal analysis (Langran and Chrisman 1988, Langran 1992, Peuquet 1994, Yuan 1999, Peuquet 2001, Peuquet 2002, Miller and Bridwell 2008), questions remain regarding the most efficient representations of spatiotemporal data to support dynamic, real-time operations, especially in the era of “big data” (Wang and Lu 2009, Brown *et al.* 2011, Goth 2012, Lohr 2012). Due to the potentially significant effects that different forms of data representation can have on data analysis and visualization, the representation of spatiotemporal data was identified as a priority research area by the University Consortium for Geographic Information Science (UCGIS) (Yuan *et al.* 2005), and remains an active research topic today (e.g., Tøssebro and Nygård 2011, Li and Kraak 2012).

Object-oriented approaches to database modeling offer some advantages over the snapshot model. Namely, they avoid duplication of data and provide added functionality such as the inheritance of properties from one object to another. Worboys (1992) took the time-stamping approach used in the snapshot and TMS models and applied it to spatial objects, describing “spatiotemporal atoms” as subunits of spatiotemporal objects. Peuquet and Duan (1995) used an object-oriented approach in their event-based spatiotemporal data model (ESTDM), which also uses a time-stamping approach but only stores the changes related to an object from one time slice to the next, minimizing redundancy in the data. Goodchild *et al.* (1999) and Goodchild *et al.* (2007) adopted the concept described by Worboys (1992) of reducing all geographic information to an atomic unit, which can help generalize geographic representation.

The implementation of temporal capabilities in relational database management systems (RDBMSs) has been described in many forms (e.g., Snodgrass 2000). Certain software products such as TimeDB (2012) attempt direct implementations of temporal data storage and management. Although many RDBMS products such as Oracle and PostgreSQL have powerful spatial extensions available (Oracle Spatial and PostGIS, respectively), none have similar temporal extensions available. Temporal data representation and analysis has only recently become available in GIS software from major vendors such as Esri's ArcGIS (2012).

1.2 Advancements in Web-based Technology

Exploration and analysis of multidimensional spatiotemporal data typically requires specialized locally-installed (a.k.a. "desktop") GIS software. However, advancements in server-side mapping and analysis tools, their integration with web servers, and a range of complex client-side software libraries make the standard web browser a compelling candidate as a common user interface for working with spatiotemporal data. Innovative combinations of web-based software tools and custom code facilitate the development of rich internet applications (RIA), which can include many of the user interface features commonly found in locally-installed software applications.

Although the client-side programming language JavaScript has been available since its development in the Netscape web browser in 1995, the past several years have witnessed a revolution in the usage and advancement of client-side methods that leverage JavaScript in unique ways. Core to this advancement is the concept of asynchronous communication between a client (i.e., a web browser) and server, instantiated as the Asynchronous JavaScript and XML (AJAX) approach and so-named by Garrett (2005). Simply put, the AJAX approach enables content on a web page to be updated without having to reload the entire page. Google's Gmail and Maps web applications (deployed in 2004 and 2005, respectively) represented the first wide-scale usage of the AJAX approach, and popularized the technique to the extent that most major websites now use it in one form or another. AJAX approaches are central to the development of RIA, which use AJAX directly, as well as in supporting tools such as jQuery (2012), a library that simplifies and centralizes client-side operations while providing cross-browser support. In the context of web-based mapping applications, OpenLayers (2012) provides a powerful example of a client-side mapping toolkit that supports the development of both simple and complex web-mapping applications, all of which use the AJAX approach.

Web-based server-side software has also advanced in the past decade, particularly in the area of open source mapping tools. MapServer (2012), a server-side open source web mapping toolkit that was initially developed in 1994 and released under an open source software license in 1999, has since undergone significant changes and is still under active development. GeoServer (2012), written in Java, is an example of a server-side web mapping toolkit that includes full implementation of the Open

Geospatial Consortium's (OGC) web mapping service (WMS), web feature service (WFS), and web coverage service (WCS) standards. Although server-side web programming languages such as PHP have also undergone significant development in the past decade, the advent of web application frameworks is perhaps a more noteworthy event, particularly in the case of lightweight Python frameworks based on the web server gateway interface (WSGI), such as Pyramid (2012). These frameworks enable the fast development of targeted web-based applications that can leverage other server-side Python processing code and deliver the results via the hypertext transfer protocol (HTTP).

1.3 Automated Data Processing

To support real-time operations on spatiotemporal data in a web-based context, not only must powerful server-side processing capabilities be used, but innovative methods must be developed to automate and, ideally, generalize the related processes. Automated processing of this type can require a complex set of linked tools, services, databases, files, and custom code, and necessitates the efficient handling of large quantities of multidimensional data.

As the size of spatiotemporal datasets and databases has grown, researchers have devised automated methods to efficiently process and analyze them. Although the storage and processing difficulties associated with "big data" seem increasingly prevalent today, spatial data collections have presented formidable computational challenges since the early days of GIS. Several discussions about automated processing occurred in the 1980s. Openshaw (1987) advocated for the development of fully automated geographical analysis systems, and Openshaw *et al.* (1987) put theory into practice, building an automated geographical analysis machine (GAM) on a Mark I computer system to perform repeated, automated point pattern analyses. At the time, this was a fairly unique example of a dedicated machine developed for a specific, automated spatial processing task (Goodchild *et al.* 1992). Addressing a common spatial data processing issue, Brassel and Weibel (1988) reviewed the possibilities of automated map generalization using currently available computer technology.

In recent years, the automated processing of spatiotemporal data and automated generation of geovisualizations have received an increasing amount of attention in the research community. Sharma *et al.* (2012) devised an automated processing chain to convert data from the Geographic Markup Language (GML) format to shapefiles for use in a custom query and visualization tool. Lakshmanan (2012) described the development of automated algorithms for the analysis of spatial grids. Misra *et al.* (2011) describe the rapid development of software tools for automated analysis of spatial data using open source software components. Today, advanced server-side technology such as GIS-based server software, related application programming interfaces (APIs), and custom code, enables the development of powerful systems that can quickly process spatiotemporal data to produce output and deliver it to the user for visualization, exploration, and further analysis.

The automation of geovisualizations refers to the automatic creation of output (e.g., a movie file or interactive interface) from spatiotemporal data. This is often done by converting multiple snapshots of data into an animated sequence, with each snapshot representing a single slice in time for a given area. If the visualization methods are extended (i.e., beyond the creation of a simple movie file), the output can become interactive, providing controls that can be manipulated by the user. Relatively little work has been done to automate the generation of geovisualizations that not only utilize user input but also provide interactive controls in the generated interface, although automation and user interactivity have both been identified as important research priorities (e.g., Buckley *et al.* 2005).

1.4 Data Exploration and Visualization

GIS can help users identify patterns in data that are not otherwise readily apparent. Similarly, scientific visualization often reveals patterns in data that lead to further understanding, by displaying data in a form that had not been viewed before (MacEachren *et al.* 1999). The term geovisualization refers to visualization techniques that are used with spatial or spatiotemporal data. Spatiotemporal data are particularly well-suited to geovisualization techniques that can display change in an area over time, a situation that often occurs as the result of a larger process. One example is the visualization of a simulated dataset representing a tsunami's propagation across the ocean and inundation of a community on the U.S. Pacific coast.

Geovisualizations are powerful tools that can be used for identifying patterns within complex multivariate datasets (Buckley *et al.* 2005, Kraak 2008), and can even be used as a method of data mining and knowledge discovery (Gahegan *et al.* 2001). Traditional methods of visualization typically involve the creation of an animated representation of an area or object changing across multiple slices of time. The animation is often rendered as a simple movie file. Many commercial software packages (e.g., ArcScene) offer 3D capabilities and generation of fly-through animations, but few (if any) have the capability of automating the dynamic generation of visualizations given a set of user-defined input parameters. The resulting animations are typically non-interactive, meaning that layers cannot be activated or deactivated, and other features cannot be controlled. However, recent developments in commercial software such as Esri CityEngine (2012) and Microsoft Layerscape (2012) show promise for enabling these capabilities within a web browser. Recently, Schultz and Bailey (2012) described an interesting approach for processing spatiotemporal data and generating interactive OpenGL-based visualizations of the datasets as extruded volumes.

An even smaller subset of software tools allow the display of time-series animations to be overlaid with spatial layers and manipulated in that context. One example is Google Earth which enables the control and animation of time-series data via time-stamped KML-defined layers, manipulated by a time slider mechanism. However, the user is still constrained by the fact that (1) Google Earth is a locally-installed application (although it is a free product and can be installed on all common operating systems, it still requires more steps than simply using a web browser) and (2) it is not possible to query the underlying layers at any location, as the user could do in a true GIS or web-based mapping tool designed for that purpose. In recent versions, ArcGIS Desktop software includes tools for enabling time-based operations on spatiotemporal datasets that include temporal attributes assigned using the snapshot approach (ArcGIS Resource Center 2012). The ArcGIS graphical interface utilizes a time slider similar approach similar to that available in Google Earth, but includes additional functionality.

Interactive geovisualizations and related tools (e.g., interactive charts) that enable spatiotemporal data exploration can effectively enhance the user experience. For example, placing output data values in a static list or table is helpful, particularly if a data export facility is provided so the user can easily obtain the data for further analysis on their own computer, but placing data values in a dynamic table linked to an interactive chart, overlaying them on an interactive mapping tool that highlights temporal relationships, and generating and plotting meaningful statistical chart aids based on the data values provides a richer user experience. If the data contain both spatial and temporal components, it is worthwhile to invest the effort to make that complexity available to, and interpretable by, the user. As Tufte (1983, p. 30) noted:

> Time-series displays are at their best for big data sets with real variability. Why waste the power of data graphics on simple linear changes, which can usually be better summarized in one or two numbers? Instead, graphics should be reserved for the richer, more complex, more difficult statistical material.

Although written nearly 30 years ago, with innumerable technical advances in GIS, computer graphics, and visualization realized in the intervening period, Tufte's statement remains just as applicable today.

1.5 Spatiotemporal Data Used in This Study

Spatiotemporal datasets exist in many forms and at multiple scales. The approach taken in this study was to work with a set of large spatiotemporal datasets (on the order of terabytes and millions of observations) to provide sufficiently complex test cases and challenges for data acquisition, processing, storage, representation, analysis, visualization, and dynamic value extraction. Data representing modeled spatiotemporal phenomena (i.e., tsunami inundation modeling, tsunami evacuation simulation, and climate modeling across the conterminous U.S.) were deemed appropriate for this purpose, and are described briefly in this section.

1.5.1 Tsunami Inundation Modeling Input Data

Tsunami modeling involves defining a complex set of input parameters and preparing the necessary gridded bathymetry and topography data to be used as inputs for analysis. Tsunami simulations are performed using any of a number of modeling codes that calculate the event initiation and propagation of the resulting tsunami wave across the open ocean at each simulated time step, including the inundation of the tsunami flow on land. One means of producing tsunami simulation time-series output data is via the Tsunami Computational Portal (TCP 2012), a web-based research portal that aggregates multiple tsunami simulation model codes and provides a common parameterization interface for defining simulation runs. A large number of gridded input bathymetry datasets are available for selection in the TCP. The gridded datasets vary in resolution and extent, and have specific requirements for nesting (i.e., substituting a high-resolution dataset for a near-shore area within a low-resolution bathymetry grid), which complicate server-side data preparation and processing operations.

1.5.2 Tsunami Evacuation Simulation Data

Tsunami simulation modeling runs produce binary output data products representing gridded time-series wave height for each time step, maximum wave height across all time steps, and, depending upon the selected model, matching U and V velocity vectors. All of these data products are used as inputs in a tsunami simulation modeling framework to provide per-pixel water depth values at each time step. The simulation framework uses these data to calculate potential casualties during an evacuation scenario, and produces an additional spatiotemporal dataset representing the movement and status of the evacuating population at each time step. This dataset drives a set of web-based tools that allow exploration of the simulation over time.

1.5.3 PRISM Modeled Climate Data

The PRISM Climate Group at Oregon State University produces climate grids using weather station data and the Parameter-elevation Regressions on Independent Slopes Model (PRISM) approach to model precipitation, maximum temperature, minimum temperature, and mean temperature across the conterminous U.S. on daily, monthly, and annual time scales (Daly *et al.* 1994, Daly *et al.* 2002). Data describing weather stations, observation times, and other information are stored in a database, but the modeled time-series data are all stored on the filesystem as spatial grids in the binary interleaved by line (BIL) format. Currently over 49,000 PRISM climate grids are stored on the filesystem, representing the set of time-series climate data used in this study.

1.6 Research Goals

The overarching goal of this dissertation was to develop methods for storing, accessing, analyzing, and processing spatiotemporal data in support of real-time web-based operations. The projects undertaken in this dissertation represent real-world research challenges and provide good test cases for attaining that goal. The specific research goals of this dissertation include:

- Automate the processing of spatiotemporal data based on user input to generate output data products and visualizations in support of data analysis and exploration.

- For each project, determine the best spatiotemporal data representations for fast access and processing.
- Create functional web-based tools that run within a common user interface (i.e., a standard web browser).
- Build all web-based tools and related utilities using open source server-side and client-side software, in addition to custom code.
- Animate time-series output data in a web-based geovisualization tool, while allowing simultaneous interaction with the data (map operations, querying, etc.).

The *tools* developed in this dissertation represent solutions designed for specific research problems. However, the *methods* are typically generalized, such that they could be applied to any discipline using similarly-structured spatiotemporal datasets. Chapter 2 describes methods for data processing and storage for a unique web-based application, while Chapters 3 and 4 describe the methods used to develop rich internet applications for real-time interaction with large spatiotemporal datasets. Chapter 5 presents the conclusion of the dissertation and suggested paths forward. The purpose of Chapters 2-4 is described briefly below.

1.6.1 Spatial Data Handling in the Tsunami Computational Portal (Chapter 2)

The TCP enables tsunami researchers to configure simulation runs across three different model codes using a common parameterization scheme. All input settings are configured in a web interface that supports dynamic, map-based selection of the gridded bathymetry and topography data used as inputs for the model codes. The available bathymetry and topography datasets include global data such as GEBCO

(2010) and ETOPO1 (Amante and Eakins 2009), in addition to fine-scale grids located near shore. The gridded datasets are all stored in a PostgreSQL (2012) database, spatially enabled via PostGIS (2012). Assignment of the input datasets is a dynamic process that involves ensuring alignment of grid points and dynamic, real-time snapping of selected boundaries to existing datasets. Tsunami simulation runs produce modeled time-series output data used to generate animations or as inputs for further analysis. This chapter will be submitted for publication in the journal *Transactions in GIS*.

1.6.2 Web-based Tsunami Scenario Simulation Framework (Chapter 3)

Modeled tsunami inundation data (from the TCP or other simulation tools) are used as inputs for an evacuation simulation framework, for which the input parameters are defined by the user via a web interface (i.e., the user controls the parameters governing the evacuation scenario simulation). The framework simulates predicted human movement and uses the time-series inundation data together with a casualty model to identify each simulated individual's casualty status at each time step of the simulation. The time-series simulation output data is written to a spatial database and used to dynamically populate a web-based evacuation simulation mapping tool, in which the user can animate both the time-series inundation data and the time-series evacuation scenario, plus query any of the underlying data at any point without interrupting the animation. Most web-based GIS implementations allow attributes of the underlying spatial layers to be queried; however, few web-based interfaces allow time-series animations to be overlaid with spatial layers and, at the same time, enable querying of the time-series data in a manner similar to the spatial layers. This chapter will be submitted for publication in the *International Journal of Geographical Information Science*.

1.6.3 Web-based Climate Grid Statistics (Chapter 4)

The PRISM climate grids represent a large (about 4.5 TB) set of time-series data, with each grid approximately 85 MB in size. Loading the grids over the network for analysis of large spatial or temporal scales can take a prohibitively long amount of time, and analysis of the grid sets can require significant processing power. Leveraging and extending a Python and open source GIS server-side grid processing framework allows the time-series data to be dynamically extracted from selected grid sets over user-defined spatial and temporal windows. The extracted data can be analyzed using a number of statistical techniques, with the output data displayed on a map and plotted on a chart, depending upon the type of analysis selected. All operations are defined by the user via an interactive web-based interface that handles the definition of input temporal and spatial scales as well as visualization and exploration of the output data, which can be exported for further analysis. This chapter will be submitted for publication in the journal *Computers & Geosciences*.

1.7 References

Amante, C. and B.W. Eakins. 2009. ETOPO1 1 arc-minute global relief model: Procedures, data sources and analysis. NOAA Technical Memorandum NESDIS NGDC-24, 19 pp, March.

ArcGIS. 2012. Esri ArcGIS – Mapping & analysis for understanding our world, http://www.esri.com/software/arcgis (last accessed 20 September 2012).

ArcGIS Resource Center. 2012. How time is supported in spatial data, http://resources.arcgis.com/en/help/main/10.1/index.html#//005z00000004000000 (last accessed 27 September 2012).

Armstrong, M.P. 1988. Temporality in spatial databases. Proceedings: GIS/LIS '88 2: 880-889.

Beller, A., T. Giblin, K.V. Le, S. Litz, T. Kittel, and D. Schimel. 1991. A temporal GIS prototype for global change research. Proceedings: GIS/LIS '91 2: 752-765.

Brassel, K.E. and R. Weibel. 1988. A review and conceptual framework of automated map generalization. *International Journal of Geographical Information Science* 2(3): 229-244.

Brown, B., M. Chui, and J. Manyika. 2011. Are you ready for the era of 'big data?' McKinsey Quarterly, October, https://www.mckinseyquarterly.com/Are_you_ready_for_the_era_of_big_data_2864 (last accessed 20 September 2012).

Buckley, A.R., M. Gahegan, and K. Clarke. 2005. Geographic visualization. Pages 313-333 *in* R. B. McMaster and E. L. Usery (eds.), A research agenda for geographic information science. CRC Press, Boca Raton, FL.

Daly, C., R.P. Neilson, and D.L. Phillips. 1994. A statistical-topographic model for mapping climatological precipitation over mountainous terrain. *Journal of Applied Meteorology* 33: 140-158.

Daly, C., W.P. Gibson, G.H. Taylor, G.L. Johnson, and P. Pasteris. 2002. A knowledge-based approach to the statistical mapping of climate. *Climate Research* 22: 99-113.

Esri CityEngine. 2012. Esri CityEngine: Smart 3D city models, http://www.esri.com/software/cityengine (last accessed 12 October 2012).

Gahegan, M., M. Wachowicz, M. Harrower, and T-M. Rhyne. 2001. The integration of geographic visualization with knowledge discovery in databases and geocomputation. *Cartography and Geographic Information Science* 28(1): 29-44.

Garrett, J.J. 2005. Ajax: A new approach to web applications, http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications (last accessed 28 September 2012).

GEBCO. 2010. General bathymetric chart of the oceans: The GEBCO_08 grid, http://www.gebco.net/data_and_products/gridded_bathymetry_data/documents/gebco_08.pdf (last accessed 11 February 2012).

GeoServer. 2012. GeoServer – a Java-based software server that allows users to view and edit geospatial data, http://www.geoserver.org (last accessed 28 September 2012).

Goodchild, M.F., R.P. Haining, S. Wise, *et al.* 1992. Integrating GIS and spatial analysis – Problems and possibilities. *International Journal of Geographic Information Science* 6(5): 407-423.

Goodchild, M.F., M.J. Egenhofer, K.K. Kemp, D.M. Mark, and E. Sheppard. 1999. Introduction to the Varenius project. *International Journal of Geographical Information Science* 13(8): 731-745.

Goodchild, M., M. Yuan, and T. Cova. 2007. Towards a general theory of geographic representation in GIS. *International Journal of Geographic Information Science* 21(3): 239-260.

Google Earth. 2012. Google Earth, http://earth.google.com (last accessed 20 September 2012).

Google Maps. 2012. Google Maps, http://maps.google.com (last accessed 20 September 2012).

Goth, G. 2012. The science of better science. *Communications of the Association for Computing Machinery* 55(2): 13-15.

jQuery. 2012. jQuery JavaScript library, http://www.jquery.com (last accessed 13 September 2012).

Kraak, M.-J. 2008. Geovisualization and time – New opportunities for the space-time cube. Pages 293-318 *in* M. Dodge, M. McDerby, and M. Turner (eds.), Geographic visualization: Concepts, tools, and applications. Wiley, West Sussex, England.

Lakshmanan, V. 2012. Automated analysis of spatial grids: Motivation and challenges. *Geotechnologies and the Environment* 6: 1-18.

Langran, G. and N.R. Chrisman. 1988. A framework for temporal geographic information. *Cartographica* 25(3): 1-14.

Langran, G. 1992. Time in geographic information systems. Taylor & Francis, London.

Li, X. and M.-J. Kraak. 2012. Explore multivariable spatio-temporal data with the time wave: Case study on meteorological data. Pages 79-92 *in* Advances in spatial data handling and GIS. Lecture notes in geoinformation and cartography, part 3. Springer, Berlin.

Lohr, S. 2012. The age of big data. The New York Times, February 11, http://www.nytimes.com/2012/02/12/sunday-review/big-datas-impact-in-the-world.html (last accessed 20 September 2012).

MacEachren, A., M. Wachowicz, R. Edsall, D. Haug, and R. Masters. 1999. Constructing knowledge from multivariate spatiotemporal data: Integrating geographical visualization with knowledge discovery in database methods. *International Journal of Geographical Information Science* 13(4): 311-334.

MapServer. 2012. MapServer – Open source web mapping, http://www.mapserver.org (last accessed 20 September 2012).

McLachlan, J., D. Foster, S. Clayden, and S. Barry. 2005. Long-term forest and landscape dynamics. Pages 125-141 *in* D.R. Foster and J.D. Aber (eds.), Forests in time: The environmental consequences of 1,000 years of change in New England. Yale University Press, New Haven, CT.

Microsoft LayerScape. 2012. Microsoft LayerScape, http://research.microsoft.com/en-us/projects/layerscape (last accessed 12 October 2012).

Miller, H. and S.A. Bridwell. 2008. A field-based theory for time geography. *Annals of the Association of American Geographers* 99(1): 49-75.

Misra, I., S.M. Moorthi, R.K. Gambhir, and R. Ramakrishnan. 2011. Evolutionary rapid development using open source framework for geospatial data processing. *Trends in Information Management* 7(1): 31-40.

OpenLayers. 2012. OpenLayers: Free maps for the web, http://www.openlayers.org (last accessed 20 September 2012).

Openshaw, S. 1987. An automated geographical information system. *Environment and Planning A* 19: 431-436.

Openshaw, S., M. Charlton, C. Wymer, and A. Craft. 1987. A Mark 1 geographical analysis machine for the automated analysis of point data sets. *International Journal of Geographical Information Systems* 1(4): 335-358.

Peuquet, D.J. 1984. A conceptual framework and comparison of spatial data models. *Cartographica* 21(4): 66-113.

Peuquet, D.J. 1994. It's about time: A conceptual framework for the representation of temporal dynamics in geographic information systems. *Annals of the Association of American Geographers* 84(3): 441-461.

Peuquet, D.J. and N. Duan. 1995. An event-based spatiotemporal data model (ESTDM) for temporal analysis of geographical data. *International Journal of Geographical Information Systems* 9(1): 7-24.

Peuquet, D.J. 2001. Making space for time: Issues in space-time data representation. *GeoInformatica* 5(1): 11-32.

Peuquet, D.J. 2002. Representations of space and time. Guilford Press, New York.

PostGIS. 2012. PostGIS – A spatial extension to PostgreSQL, http://postgis.refractions.net (last accessed 20 September 2012).

PostgreSQL. 2012. The PostgreSQL relational database management system, http://www.postgresql.org (last accessed 20 September 2012).

Pyramid. 2012. Pyramid open source web application framework, http://www.pylonsproject.org/projects/pyramid/about (last accessed 18 August 2012).

Schultz, N. and M. Bailey. 2012. Using extruded volumes to visualize time-series datasets. Pages 127-148 *in* J. Dill, R. Earnshaw, D. Kasik, J. Vince, and P.C. Wong (eds.), Expanding the frontiers of visual analytics and visualization. Springer, London.

Sharma, S, U.S. Tim, and S. Gadia. 2012. AutoConViz: Automating the conversion and visualization of spatio-temporal query results in GIS. *Geo-spatial Information Science* (in press). Available online first at http://dx.doi.org/10.1080/10095020.2012.714099 (last accessed 27 September 2012).

Snodgrass, R.T. 2000. Developing time-oriented database applications in SQL. Morgan Kauffman, San Francisco.

TimeDB. 2012. TimeDB bitemporal relational DBMS, http://www.timeconsult.com/Software/Software.html (last accessed 20 September 2012).

TCP. 2012. Tsunami Computational Portal, http://tsunamiportal.nacse.org (last accessed 30 August 2012).

Tøssebro, E. and M. Nygård. 2011. Representing topological relationships for spatiotemporal objects. *Geoinformatica* 15: 633-661.

Tufte, E.L. 1983. The visual display of quantitative information. Graphics Press, Cheshire, CT.

Wang, S. and Y. Liu. 2009. TeraGrid GIScience gateway: Bridging cyberinfrastructure and GIScience. *International Journal of Geographical Information Science* 23(5): 631–656.

Wen, T.-H., M.-H. Lin, and C.-T. Fang. 2012. Population movement and vector-borne disease transmission: Differentiating spatial-temporal diffusion patterns of commuting and noncommuting dengue cases. *Annals of the Association of American Geographers* 102(5): 1026-1037.

Worboys, M.F. 1992. A model for spatio-temporal information. Proceedings: The 5th International Symposium on Spatial Data Handling 2: 602-611.

Yuan, M. 1999. Use of a three-domain representation to enhance GIS support for complex spatiotemporal queries. *Transactions in GIS* 3(2): 137-159.

Yuan, M. 2001. Representing complex geographic phenomena in GIS. *Cartography and Geographic Information Science* 28(2): 83-96.

Yuan, M., D.M. Mark, M.J. Egenhofer, and D.J. Peuquet. 2005. Extensions to geographic representations. Pages 129-156 *in* R.B. McMaster and E.L. Usery (eds.), A research agenda for geographic information science. CRC Press, Boca Raton, FL.

Zhong, C., T. Wang, W. Zeng, and S.M. Arisona. 2012. Spatiotemporal visualization: A survey and outlook. Pages 299-317 *in* S.M. Arisona, G. Aschwanden, J. Halatsch, and P. Wonka (eds.), Communications in computer and information science, vol 242: Digital urban modeling and simulation. Springer, New York.

Chapter 2: Spatial data configuration, storage, and analysis for web-based tsunami computational modeling and visualization

Authors: Dylan B. Keon, Cherri M. Pancake, Harry H. Yeh, Dawn J. Wright

2.1 Abstract

The relative isolation of useful computational models and the difficulty of comparing alternative approaches are common problems shared by many research disciplines. Models typically have unique input/output formats and parameterization schemes, and the codes may require specific hardware/software environments. This makes running similar jobs against different models in order to compare results difficult. Storage, representation, and efficient retrieval of massive data sets for analysis present additional challenges.

The Tsunami Computational Portal (TCP) is a cyberinfrastructure service that brings together computational models in a common environment to simplify comparison. Its web-based interface facilitates executing the same inputs and controls against multiple models, or modifying parameters incrementally for a particular model. Researchers retrieve and visualize the results using tools available in the interface. The web interface is driven by a large PostgreSQL database (spatially-enabled by PostGIS) that contains global and local bathymetry and topography datasets (grids). A web-based mapping tool allows users to select grids of interest, nesting fine-resolution near-shore grids within coarse open-ocean grids, and automatically snapping them to points in the database to satisfy alignment requirements of the model codes. Custom utilities automatically handle the export of spatial objects from the database in the binary grid format required for processing on large computational clusters, and manage communication among related components.

Post-processing code automatically generates images and video of each model run from the output data.

The TCP has made it possible to compare tsunami models and observe the effects of alternative numerical methods in a common interface. A similar approach would benefit any domain where alternative models need to be exercised broadly and compared in order to increase scientific understanding of time-series phenomena. Managing the spatial data required for model run parameterization and simulation analysis presents data configuration and storage challenges, particularly with regard to efficient real-time (i.e., via web interface) spatial queries and data export. Through effective data storage, indexing, and spatial query techniques the TCP gives researchers fast access to grids needed for tsunami modeling, while contributing a useful, shared, web-based resource to the tsunami modeling community.

2.2 Introduction

Tsunami awareness has increased greatly in the past decade, in large part due to the major tsunami events caused by the 2004 Indian Ocean earthquake and the 2011 Tōhoku earthquake near Japan, both of which resulted in widespread damage and loss of human life in impacted coastal areas (Titov *et al.* 2005, Geist *et al.* 2006, Dunbar *et al*. 2011, Gupta 2011). The 2011 Tōhoku tsunami also caused damage to areas along the U.S. West Coast vulnerable to tsunami inundation, and resulted in one death in California (Dunbar *et al*. 2011, Allan *et al.* 2012). Japanese tsunami marine debris is still adrift in the Pacific Ocean and is actively being tracked (Showstack 2011, NOAA 2012). Debris has already reached the U.S. West Coast (e.g., Murphy 2012), with more predicted to land in 2013 (NOAA 2012), leading to continued interest in tsunamis and their aftereffects from both researchers and the general public. Along the U.S. West Coast, many tsunami-related educational and outreach efforts have taken place at State and Federal levels (e.g., Bernard 2005), as well as at the local level where tsunami hazard mitigation planning, inundation zone mapping, and citizen education continue to be areas of active research (e.g., Johnston *et al.* 2004, Tang *et al.* 2008). Significant work remains, however, in accurately predicting tsunami behavior within specific geographic areas.

Post-tsunami surveys can provide useful data for interpreting tsunami behavior, but field observations alone give only a partial understanding of a tsunami event (Lynett and Liu 2011). Computational modeling of known and hypothetical tsunami events can improve understanding of tsunami behavior. Tsunami modeling consists of three phases: (1) generation of the tsunami event (typically by underwater earthquake but possibly by landslide or other means), (2) propagation of the tsunami wave across the ocean, and (3) inundation and run-up of the tsunami wave on shore, where substantial tsunami energy can be released (Yeh 1991, Bernard *et al*. 2006, Gisler 2008). High-quality spatial data (bathymetry and coastal topography grids) at multiple scales are required to effectively study tsunami wave generation, propagation, and inundation. Typically, less detail (i.e, grid resolution) is needed in mid-ocean domains, with an increasing amount of detail required as the modeled tsunami wave approaches shore. This presents challenges from both computational (i.e., switching resolutions dynamically during simulation) and data-availability (i.e., fine-resolution data must be aligned exactly with coarse-resolution data) perspectives.

There is also growing recognition of the need for high quality data describing both recent tsunami and paleotsunami events, to aid researchers in duplicating event parameters for tsunami modeling purposes (Goff *et al.* 2011, Goff *et al.* 2012). Databases describing such events have been established (e.g., Goff *et al*. 2010) and are available online (e.g., Dunbar and McCullough 2011). However, while verified event data may be available online and good quality global spatial data (combined bathymetry and topography) are available at a coarse scale, the access and availability of tsunami numerical modeling codes and computational systems on which they can be run efficiently remain difficult to obtain. Furthermore, expert skill is still required to interpolate or otherwise align grids from different sources and at different scales by matching fine-scale data points with coarse-scale underlying data. Processing a single simulation scenario across different model codes for comparative studies has been impractical due to differences in model parameterization schemes and idiosyncratic input requirements.

Computational portals can provide an effective means of assembling geographically distributed resources and services in a single interface (Huang *et al*. 2006). The Tsunami Computational Portal (TCP 2012) is a cyberinfrastructure service that brings together multiple computational tsunami models in a common environment. The purpose of the TCP is to provide a useful, centralized, web-based service to researchers engaged in developing and comparing numerical models of tsunami generation, propagation, and inundation, and to facilitate comparison and further exploration of simulation output. Goals of the TCP service include:

- Provide an intuitive web-based portal with access to multiple models.
- Create a consistent user interface and model parameterization scheme.
- Enable parameter suite studies across sets of similar scenarios.
- Facilitate identical runs across different computational models.
- Give users the ability to view job status and retrieve results.
- Provide visualization tools that work directly with TCP output data.

The web-based portal interface allows tsunami researchers, as well as model developers, to study and compare numerical models of tsunami generation, propagation, and inundation. The TCP facilitates the execution of identical inputs and controls across multiple models, as well as so-called parameter sweep studies where parameters are modified incrementally while executing the same model multiple times. Researchers can retrieve and visualize results using tools available in the portal interface.

The TCP was developed as a collaborative effort involving Oregon State University (OSU), the Northwest Alliance for Computational Science and Engineering (NACSE) at OSU, the Arctic Region Supercomputing Center (ARSC) at University of Alaska Fairbanks (UAF), and tsunami researchers from institutions worldwide. Development of the TCP was funded by the U.S. National Oceanic and Atmospheric Administration (NOAA) and the U.S. Department of Defense, through its High Performance Computing Modernization Program (HPCMP).

A critical component of the TCP is the inclusion of, and rapid access to, high-quality bathymetry and coastal topography grids. This presents multiple challenges related to spatial data configuration, storage, and analysis. Tsunami numerical modeling codes are often tasked with computing trans-oceanic simulations with high-resolution near-shore datasets included for greater detail in select coastal areas. Large computational problems like these can require tens of millions of gridded data points to represent the domain of interest at multiple resolutions, exceeding the processing capabilities of standalone servers (i.e., requiring parallel processing on large computational systems). Time-critical simulations that require fast model run configuration, including satisfying spatial data alignment issues across multiple resolutions and preparing data for model input, also require fast processing. Finally, post-processing of output time-series datasets can be resource intensive.

2.3 Tsunami Modeling

Although the increased focus on tsunami science has drawn attention to tsunami modeling techniques and numerical codes used for tsunami simulation, the relative isolation of useful computational models and the difficulty of comparing alternative approaches remain common problems shared not only by the tsunami research community, but also by many other scientific research disciplines. Model source code cannot always be shared and, even when it is, other research groups may find it difficult to install and run an unfamiliar model. Each model has unique input/output formats and parameterization schemes, and the model codes themselves must often be run with exactly the hardware (e.g., parallel clusters running the Solaris vs. Linux operating system) or software (e.g., Fortran-based vs. C-based) environments used by the original developers of the model codes. The result is that a significant amount of work may be required simply to run a similar job configuration on two different model codes in order to compare results. Furthermore, researchers may have limited access to “big iron” – the supercomputing resources needed to efficiently compute large tsunami simulation processing jobs.

Several established hydrodynamic numerical modeling codes exist for modeling tsunami propagation and inundation (Table 2.1).

Table 2.1: Selected set of established tsunami numerical modeling codes. Numerical model names in **bold** text are available in the portal.

Numerical Model	Reference	Model Description
ADCIRC (A [Parallel] Advanced Circulation Model for Oceanic, Coastal and Estuarine Waters)	Blain and Kelly 2001; ADCIRC 2012	Utilizes equations discretized in space using the finite element method and in time using the finite difference method. Can be run using either the spherical or Cartesian coordinate system.
COMCOT (Cornell Multi-grid Coupled Tsunami Model)	Liu *et al.* 1995; Liu *et al.* 1998	A package of computational programs for solving the linear shallow water (LSW) equations in both spherical and Cartesian coordinate systems, and the nonlinear shallow water (NLSW) equations in the Cartesian coordinate system.
MOST (Method of Splitting Tsunami)	Titov and Gonzales 1997; NOAA Center for Tsunami Research 2012	A suite of numerical codes for simulating event generation, wave propagation, and inundation. Uses a finite difference method to divide its computational domain.
Tsunami CLAW	George and LeVeque 2006	An adaptive-mesh model that uses a finite-volume approach for solving the NLSW equations but allows for zooming in on coastal regions to better capture the effects of local bathymetry variation.
TUNAMI-N2 (Tohoku University's Numerical Analysis Model for Investigation of Near field tsunamis)	Imamura *et al.* 2006	Uses a finite distance model to apply linear theory in deep ocean areas, and shallow-water theory in shallow areas and runup on land with constant grids.
UAF Tsunami Model	Kowalik and Murty 1993; Suleimani 2004	A finite difference model that solves the NLSW equations for transoceanic tsunami propagation in the spherical coordinate system, and in the Cartesian coordinate system for runup calculation.

Effective tsunami inundation modeling depends upon three important requirements (National Research Council 2011):

1. An understanding of the phenomenon that generated the tsunami event (often via earthquake and associated shifting of the seafloor), including information that can be used to parameterize a computational model.
2. Accurate and precise gridded bathymetry and coastal topography data representing the ocean/land interface (preferably with higher-resolution data representing near-shore areas).
3. Hydrodynamic computational model code to simulate wave propagation and inundation, as well as the computing hardware required to effectively process the simulation.

The TCP addresses all three of these requirements:

1. Researchers using the TCP provide parameters that describe the event generation, to simulate either a past tsunami-generating event using known seafloor displacement information or a hypothetical event.
2. The TCP includes common global-scale bathymetry and coastal topography data (GEBCO and ETOPO), as well as fine-resolution bathymetry and coastal topography data provided by researchers to represent specific near-shore areas at various locations around the world. All grids added to the system are made available to any researcher using the portal.
3. Parameters and controls from the three established numerical tsunami modeling codes were mapped to create a common parameterization scheme: COMCOT, the UAF Tsunami Model, and Tsunami CLAW.

The three model codes available in the TCP were retrofitted to the shared parameterization scheme and ported to run on ARSC supercomputers. All of the codes can process multiple nested grids (i.e., a coarse-resolution grid can contain multiple, finer-resolution grids representing near-shore geographic locations or other regions where more detailed modeling is required).

2.4 Portal Infrastructure

The Tsunami Computational Portal system comprises multiple components distributed among various systems at NACSE and ARSC (Figure 2.1). Spatial data configuration, storage, and analysis operations are central to the system and are driven primarily by the relational database management system. The portal database, website, and custom-built data processing tools are all driven by open source software.

The portal website is a step-by-step guided interface written in a mix of PHP and JavaScript that allows the user to configure a model run and provides them with intelligent choices and defaults based upon their previous selections. Extensive error checking code validates every parameter value and provides the user with

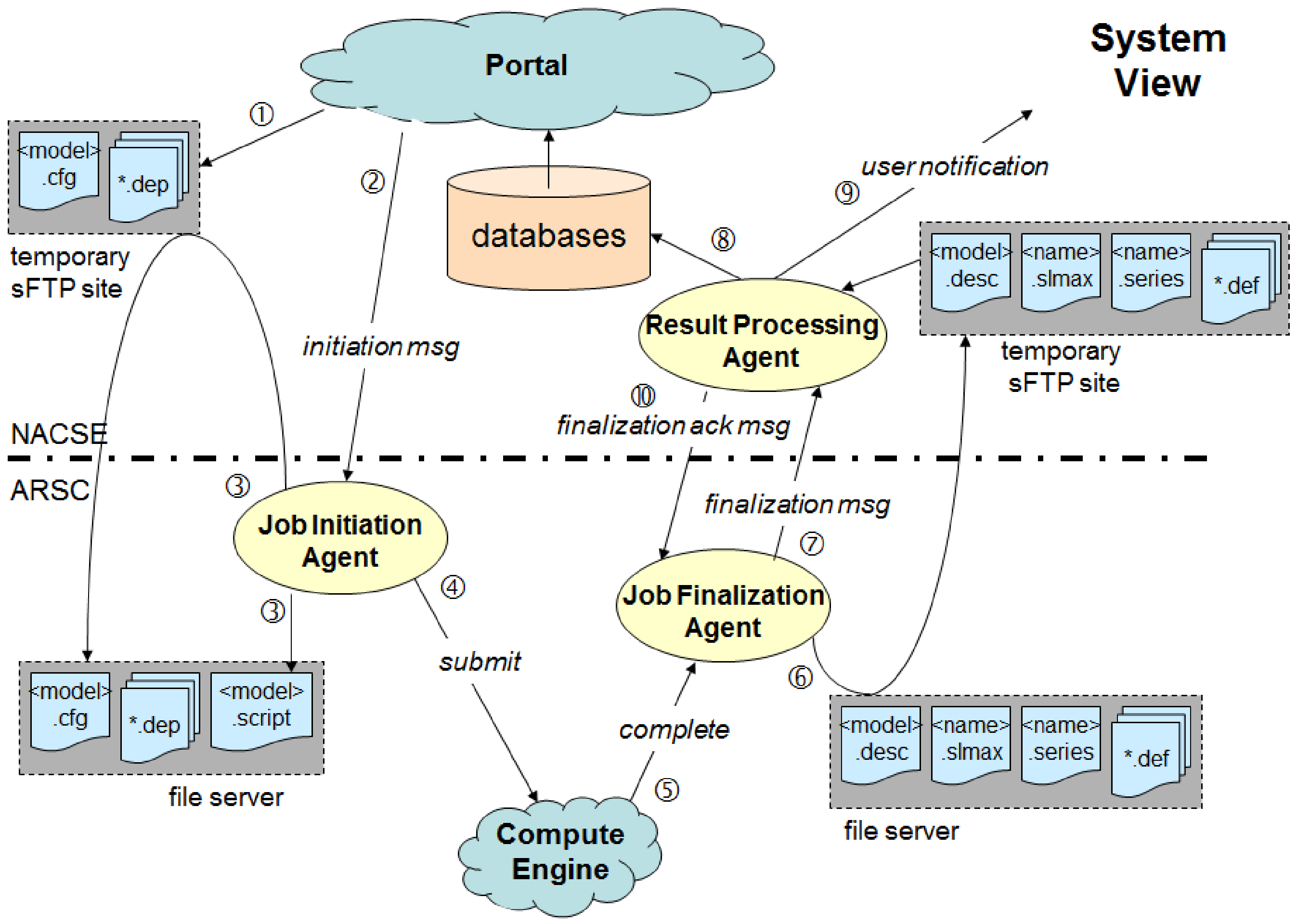


Figure 2.1: Tsunami Computational Portal system-level view architecture. Aside from job initiation and finalization messages sent by NACSE servers, all communication (including data transfers) is initiated by ARSC. This is done for security reasons (ARSC is a secure DoD site).

information about the expected size of result files based on their current settings. Each user can retrieve results, configure new jobs, and easily replicate or modify previous settings to create suites of jobs for comparison across different model codes.

The numerical models available in the portal require individual input configurations in different formats. A unified configuration scheme was developed that utilizes nearly all of the same input parameters across all models, facilitating the use of shared components in the system. Developing a common parameterization scheme across model codes was a challenging task undertaken primarily by ARSC staff. Custom utilities perform automatic communication and transfer of data between NACSE and ARSC, and enable the automatic export of spatial objects in the binary format required for processing on ARSC supercomputers. The export utility was developed as a database function that dynamically extracts the necessary slices of data from each requested grid as text and packs them into an ordered binary format that can be directly pulled into ARSC processing jobs.

Grids made available in the portal are stored and managed in the open source PostgreSQL (2012) relational database management system that is spatially enabled via the open source PostGIS (2012) module. The PostgreSQL database server instance runs on a dedicated Dell PowerEdge R710 server containing four quad-core 2.53 GHz Intel Xeon processors and 32 GB memory, with data stored across eight

320GB 10,000 RPM serial-attached SCSI disks in a RAID 5 configuration (via a hardware RAID controller). The server runs the Red Hat Enterprise Linux 6 operating system.

The database structure (Figure 2.2) was designed to store metadata that describes the bathymetry and coastal topography grids used as model inputs, the grids themselves, and detailed information about each configured model run.

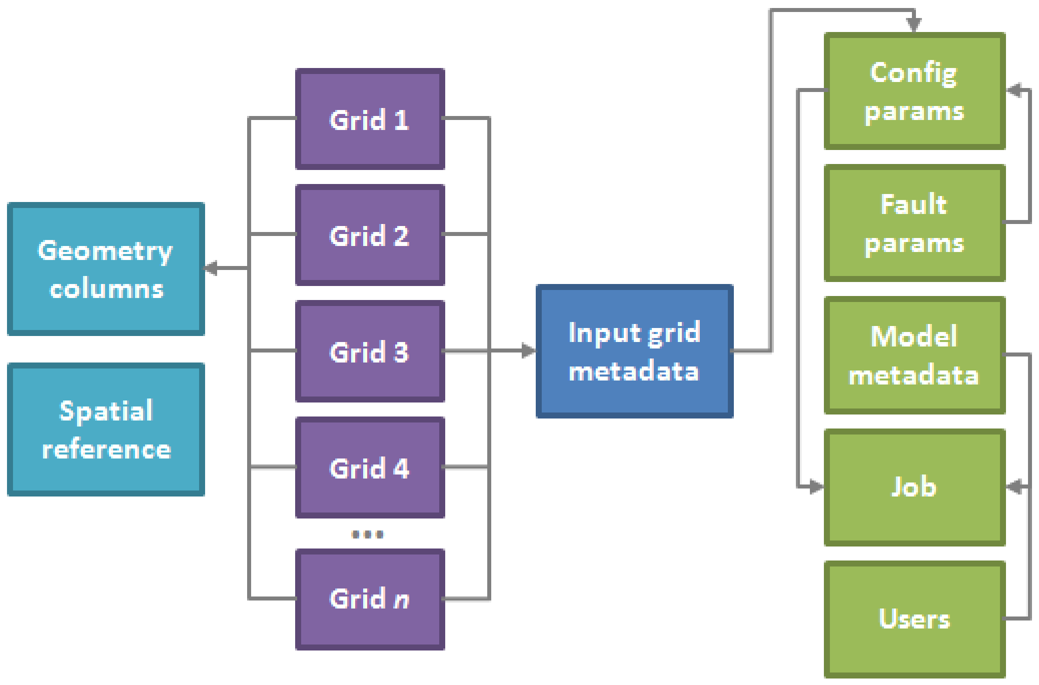


Figure 2.2: Generalized database structure diagram (key tables displayed). The database is designed to hold an unlimited number of gridded spatial datasets and their associated metadata, as well as information describing model runs configured by portal users.

Storing this detailed information allows the user to reconstruct a model run, which is particularly useful in comparative studies where the user may want to change a single parameter value while keeping the rest constant. Grids are stored as individual tables (one table per grid), while a management table stores information about each grid (e.g., the grid point spacing value, number of rows and columns, maximum extent in each direction, and calculated values describing the grid domain). A separate PostGIS management table (geometry_columns) stores information about each grid's coordinate system.

Information about completed jobs is added automatically to the database, and the output data files are automatically made available within the portal interface. Users receive notification via email when the results are ready, and can download the files from their account in the portal. Automatically-triggered post-processing code at NACSE generates images and video of each model run to provide a quick visualization of model output.

2.4.1 Database Configuration

PostgreSQL was selected as the production relational database management system for the following reasons:

- Highly regarded reputation as an enterprise-class open source relational database management system (e.g., Garbin and Fisher 2010).

- Spatial query capabilities via PostGIS that conform to the OpenGIS Simple Features Specification for SQL (Open Geospatial Consortium Inc. 2010).
- Support for unlimited rows and indexes per table, as well as a very large (32 TB) maximum table size (PostgreSQL 2012).
- Proven in-house ability to run efficiently on Linux-based servers in a production computing environment.

To enable the full set of spatial operations supported by PostGIS, the PROJ.4, GEOS, and LibXML2 open source packages must be compiled and PostGIS must be configured to use them (Table 2.2). Of particular importance, GEOS supports key spatial operations within PostGIS, as well as operators for controlling which indexes are utilized in queries (Zhang and Yi 2010). The PROJ.4 package is necessary for performing coordinate conversions within the database.

Table 2.2: Software packages required for PostgreSQL database installation and support of spatial features and operations. All software packages listed in this table are released as open source products.

Software Package	**Purpose**
PostgreSQL (postgresql.org)	Relational database management system.
PostGIS (postgis.refractions.net)	Spatial database extension for PostgreSQL (adds support for spatial objects and operations).
GEOS (geos.osgeo.org)	C++ port of the Java Topology Suite. Provides OpenGIS Simple Features operations to PostgreSQL/PostGIS.
PROJ.4 (proj.osgeo.org)	Cartographic projections library that allows the database engine to perform coordinate system transformations on spatial objects.
LibXML2 (xmlsoft.org)	XML C parser and toolkit for building tag-based structured data and documents.

The PostgreSQL database server can be tuned via the many configuration parameters in the `postgresql.conf` file that affect the database system. Four particular settings have the greatest effect on database read/write performance (Table 2.3), and have been modified on the TCP production database server to match the server's hardware configuration.

Table 2.3: Key parameter settings in the PostgreSQL `postgresql.conf` configuration file. The "Modified Setting" column displays values in use on the TCP production database server.

Parameter	**Default Setting**	**Modified Setting**	**Rationale**
shared_buffers	32 MB	8192 MB	A value that is 25% of the memory on a dedicated server gives the best balance of memory allocation and performance.
effective_ cache_size	128 MB	16384 MB	A value that is 50% of the memory on a dedicated server gives the best balance of memory allocation and performance.
checkpoint_ segments	3	32	Best set higher than default (somewhere between 16 and 64) so that the system is not continually writing checkpoints.
wal_buffers	64 kB	512 kB	Set higher than the default value (but below 1MB total) to improve performance in write-heavy operations.

2.5 Spatial Data Handling

Tsunami propagation across the ocean is strongly influenced by bathymetry and coastal topography (Lynett 2011, National Research Council 2011). Many tsunami inundation models require bathymetry data that cover the entire ocean basin as a coarse-resolution continuous base layer, with additional higher-resolution bathymetry and integrated topography data included for more detailed modeling of wave propagation and inundation along coastal areas. In the tsunami computational modeling domain, these bathymetry and coastal topography datasets are typically represented as gridded point data rather than raster-based data products.

Data points located on a regularly-spaced grid are necessary for numerical models, giving them a framework for calculating tsunami movement from one point to the next across the modeled domain. A continuous coarse-resolution grid (i.e., 30 arc-second, or ~900 m) covering the globe exists and can be used for this purpose. Researchers examining particular near-shore areas often generate finer-scale (i.e., < 100 m) bathymetry and coastal topography grids from survey data, or obtain finer-scale grids from sources such as the NOAA/NGDC coastal relief models (www.ngdc.noaa.gov/mgg/coastal), which are 1 arc-second (~30 m) or finer resolution.

2.5.1 Input Grid Specifications

Grids stored in the spatial database consist of bathymetry and coastal topography datasets representing global and regional extents. Grids added to the database are often acquired as large space-delimited ASCII text files in "xyz" format (i.e., containing longitude, latitude, and depth [elevation] values), although grids are occasionally acquired in netCDF format. The open source geospatial processing tools GMT (2012), PROJ.4 (2012), and the OGR Simple Features Library (OGR

2012) are used in various combinations for data preparation and pre-processing tasks such as grid point alignment or reordering operations, or coordinate system conversion.

Because most gridded data to be inserted in the database are typically obtained in xyz (or zxy, etc.) format, a Perl (2012) script was developed to process the gridded data (Appendix A). Following any pre-processing work, files in space-delimited xyz format are parsed into a series of structured query language (SQL) insert statements via the processing script, and inserted into the database as spatial objects via PostGIS functions. After insertion of each grid, B-tree (generalized binary search tree) indexes are built for the Cartesian x and y columns and a GiST (Generalized Search Tree)-based spatial index (specifically, R-tree-over-GiST as implemented by PostGIS) is built for the column that stores the geometry point objects. A multicolumn index (combining the indexes on the x and y columns) is also built for very large datasets (e.g., GEBCO_08) to speed queries in instances when any WHERE clauses applied to the individual x and y indexes then need to be ANDed or ORed across both columns (i.e., to query a range of points between minimum and maximum values of x and y).

After a grid is added to the spatial database, an operation is performed to update the internal statistics the database maintains for each table. Finally, custom database functions written in PostgreSQL's PL/pgSQL procedural language automatically calculate the maximum allowable time step for each grid, and the time required to cross the maximum domain of each grid (Appendix B). These values then feed directly into the TCP interface to provide guidance for users and for the system to verify that input grids are prepared to the specifications required by the model codes at ARSC.

The TCP spatial database contains the two global relief grids GEBCO_08 (GEBCO 2010) and ETOPO1 (Amante and Eakins 2009), in addition to a number of fine-resolution near-shore grids (Table 2.4). GEBCO_08 is a 30 arc-second global grid developed via the collaborative effort of a number of international oceanographic and hydrographic governmental agencies. ETOPO1 is a 1 arc-minute global relief model built from multiple global and regional datasets by the U.S. NOAA National Geophysical Data Center. Earlier versions of both global grids (GEBCO 1 arc-minute and ETOPO2 2 arc-minute) were initially added to the TCP spatial database and used in the portal prior to the availability of the newer versions. Although both earlier versions have been deprecated, and ETOPO2 may actually be misregistered in latitude and longitude (Marks and Smith 2006), the grids remains available in the TCP spatial database for researchers who may need to use them for comparative studies or to reproduce past results.

Table 2.4: Selected grids available in the TCP spatial database. The parameters max_allowable_time_step and time_to_cross_max_domain are not calculated for global grids (e.g., etopo1), as they only apply to grids of less than global extent. The "etopor" grid is a "rolled" version of the global etopo2 grid with longitude configured 0 to 360 (vs. -180 to 180), which allows researchers to run transoceanic simulations across the Pacific Ocean. The gebco30 grid is the full global GEBCO_08 30 arc-second dataset, the largest global grid stored in the TCP spatial database at nearly a billion rows of data. Latitude, longitude, and spacing values were rounded to three decimal places in this table.

Grid Name	**Min_x (longitude)**	**Min_y (latitude)**	**Max_x (longitude)**	**Max_y (latitude)**	**Spacing (arc sec)**	**Rows**	**Columns**	**Point count (dbase rows)**
cook_inlet_24s	-155.997	55.003	-147.003	61.997	24	1350	1050	1,417,500
crm_or_wa_coast_12s	-126	44	-123	47	12	901	901	811,801
crm_or_wa_coast_3s	-126	44	-123	47	3	3601	3601	12,967,901
etopo1	-180	-90	180	90	60	21601	10801	233,312,401
etopo2	-180	-89.967	179.967	90	120	10800	5400	58,320,000
etopor	0	-89.967	359.967	90	120	10800	5400	58,320,000
gebco	-180	-90	179.983	90	60	21600	10801	233,301,600
gebco30	-179.996	-89.996	179.996	89.996	30	43200	21600	933,120,000
gulf_of_alaska_2m	-168.983	52.017	-140.017	61.983	120	870	300	261,000
homer_1s	-151.558	59.584	-151.367	59.667	0.889	777	339	263,403
homer_3s	-151.751	59.534	-150.907	59.793	2.667	1140	351	400,140
homer_8s	-152.166	59.254	-150.908	59.792	8	567	243	137,781
kingscove5s	-163.3	53.2	-161.701	55.399	5	1152	1584	1,824,768
seaside_3s	-126	45	-123.5	47	3	3001	2401	7,205,401
seldovia_1s	-151.782	59.392	-151.684	59.47	0.889	399	336	134,064
seldovia_3s	-151.882	59.391	-151.476	59.533	2.667	549	192	105,408
seward_1s	-149.467	60.059	-149.308	60.158	0.889	639	402	135,000
seward_3s	-149.5	59.876	-149.251	60.166	2.667	336	393	132,441
seward_8s	-149.999	59.501	-149.001	60.165	8	450	300	137,781

2.5.2 Grid Alignment

Grids stored in the spatial database are grid/node registered, meaning that grid cells are centered on lines of latitude and longitude rather than the grid cell edges being aligned to lines of latitude and longitude (Figure 2.3). The TCP system enforces strict grid spacing alignment requirements to ensure that points in fine-resolution grids are exactly co-located with matching points (i.e., identical lat/lon coordinates) in coarse-resolution grids when the grids are nested together. The implemented tsunami numerical codes support grid spacing ratios of 5:1 and 3:1. For instance, five fine-resolution (e.g., 12 arc-second or ~360 m) gridded point spaces fit within the space between a pair of coarse-resolution (e.g., 60 arc-second or ~1.8 km) gridded points. Figure 2.3 illustrates these grid spacing requirements and provides a grid cell-level example of nesting a fine-resolution grid in a coarse-resolution grid. This grid spacing regularity enables the computational model codes to seamlessly switch from processing coarse-resolution data to fine-resolution data on an expected alignment and spacing ratio within a single model run. Fine-resolution grids require two rows and columns of “padding” points around their perimeter. This requirement satisfies the alignment system built into the grid verification code and matches the fine-resolution grids originally provided to the TCP system by UAF researchers.

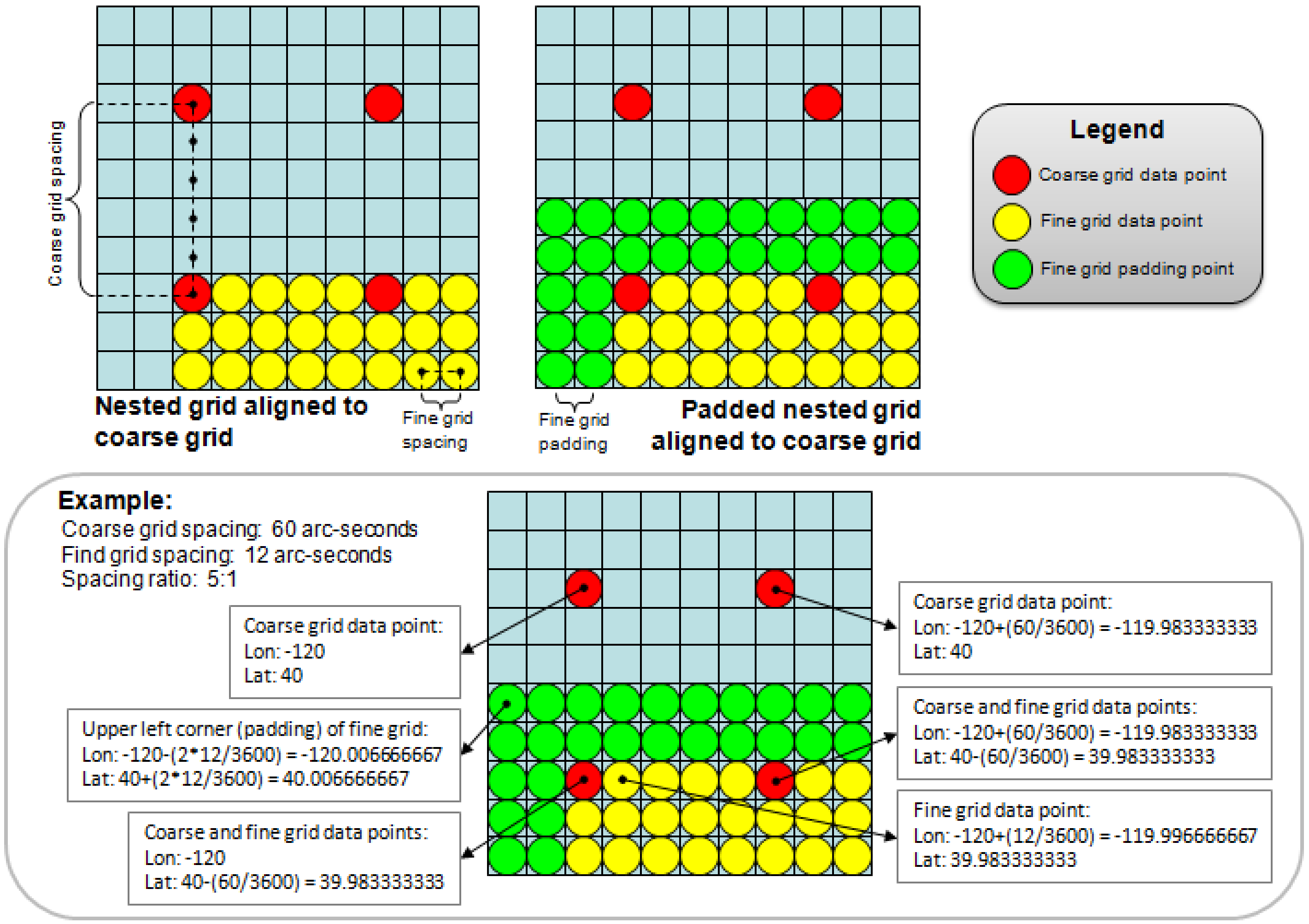

Coarse grid spacing
Nested grid aligned to coarse grid
Fine grid spacing
Fine grid padding
Padded nested grid aligned to coarse grid
Legend
Coarse grid data point
Fine grid data point
Fine grid padding point
Example:
Coarse grid spacing: 60 arc-seconds
Find grid spacing: 12 arc-seconds
Spacing ratio: 5:1
Coarse grid data point: Lon: -120 Lat: 40
Upper left corner (padding) of fine grid: Lon: -120-(2*12/3600) = -120.006666667 Lat: 40+(2*12/3600) = 40.006666667
Coarse and fine grid data points: Lon: -120 Lat: 40-(60/3600) = 39.983333333
Coarse grid data point: Lon: -120+(60/3600) = -119.983333333 Lat: 40
Coarse and fine grid data points: Lon: -120+(60/3600) = -119.983333333 Lat: 40-(60/3600) = 39.983333333
Fine grid data point: Lon: -120+(12/3600) = -119.996666667 Lat: 39.983333333

Figure 2.3: Illustration of spacing requirements for gridded data used in the TCP. Fine-resolution grids nested in coarse-resolution grids must align exactly with the coarse grid points and must have the correct coarse-to-fine spacing ratio.

Up to four fine-resolution subgrids can be nested in a master grid, with an extra subgrid nested within each of the first-level subgrids (Figure 2.4); however, subgrid borders cannot cross or touch one other, and each model restricts how deeply nesting can occur. To simplify this selection process for portal users configuring model runs, the TCP handles these restrictions automatically – a validation system written in mix of PHP and JavaScript (examples in Appendix C) ensures that nested subgrids are aligned properly within parent grids and match the acceptable 5:1 or 3:1 grid spacing ratios. A custom map-based grid selection tool (Figure 2.5) allows users to select input grids dynamically, and user-selected extents of nested subgrids are automatically snapped to matching points within their parent grids according to the requirements of the selected model code. The mapping tool displays the grid selection and nesting arrangement graphically, with the defined extent outlined and the grid name and resolution labeled within each extent (Figure 2.6). Actions such as snapping subgrid points to parent grid points and checking parameter settings against the selected spatial extent all take place in real time while the user is configuring a model run. Storing the grids in a spatial database makes these real-time checking and verification operations possible – the PHP-based website code can quickly connect to the PostgreSQL database, run a query, process the result, and provide the necessary feedback to the user.

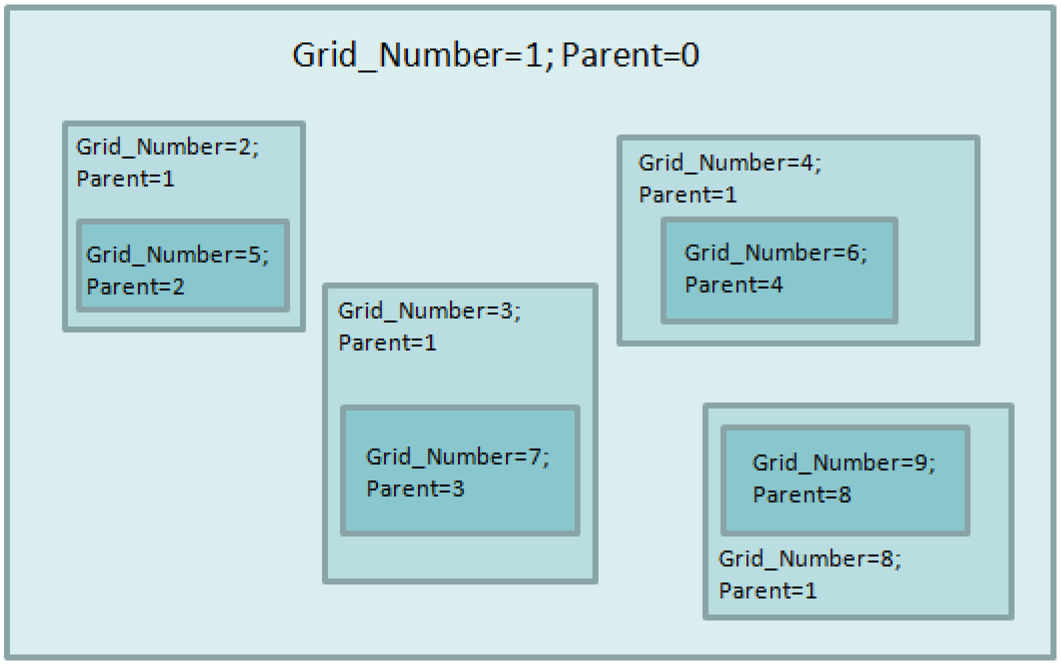
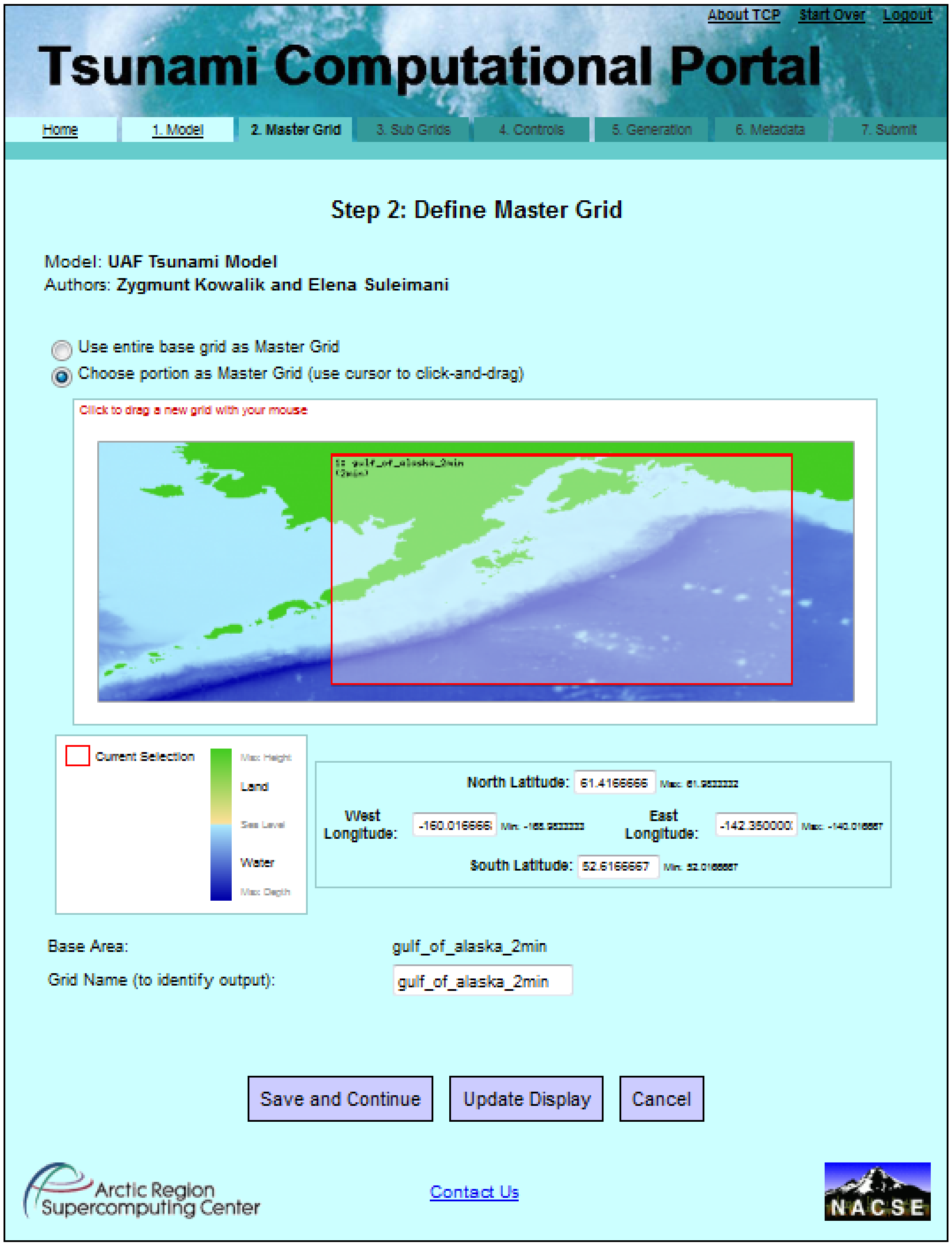


Figure 2.4. Representation of an acceptable grid nesting arrangement. This is the maximum allowed number of grids and level of nesting for a single model run. The largest region (Grid_Number=1) represents the master grid, containing four subgrids (numbers 2, 3, 4, 8), each of which also contain a subgrid (numbers 5, 6, 7, 9).



Step 2: Define Master Grid

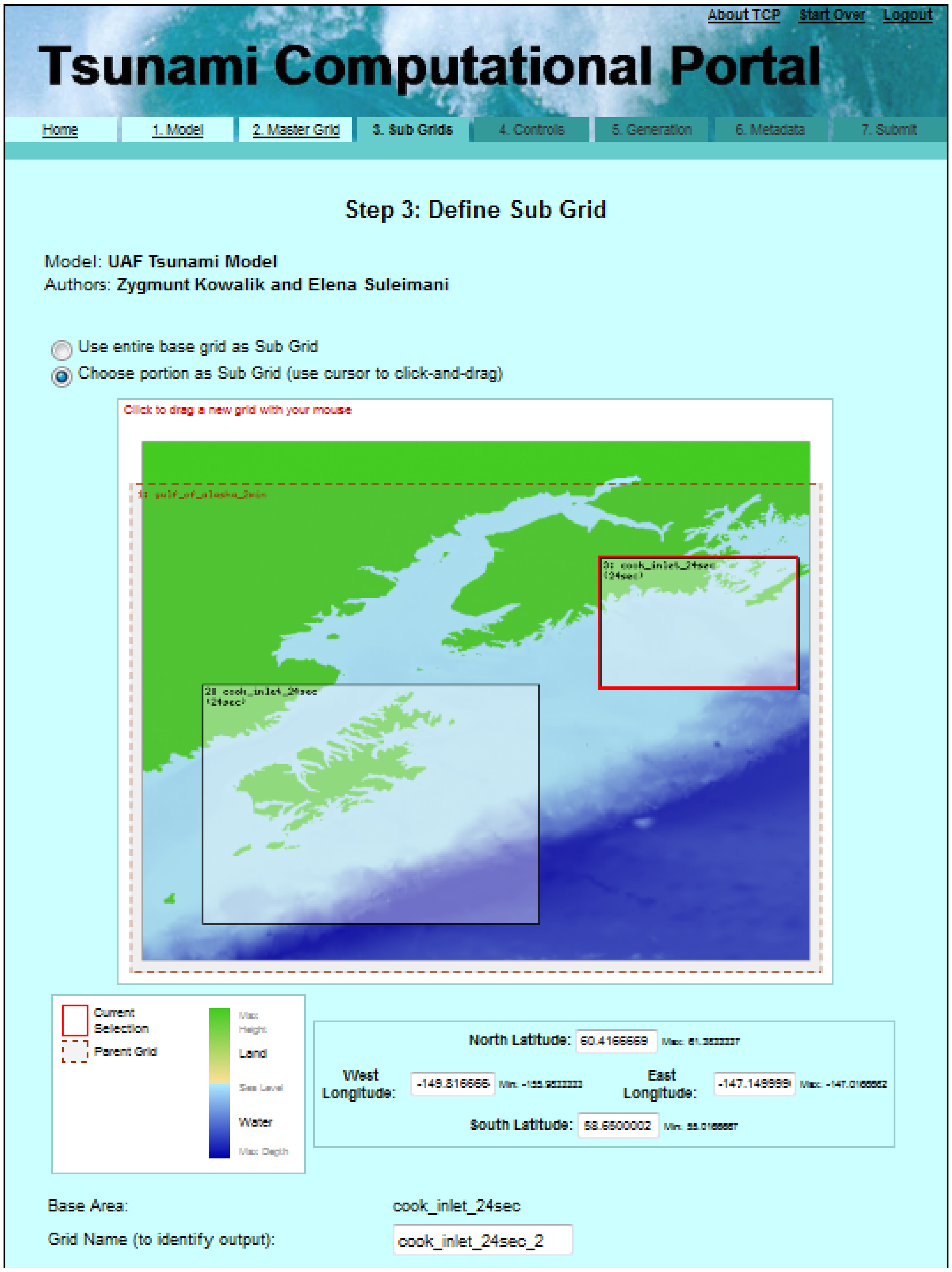
Model: **UAF Tsunami Model**
Authors: **Zygmunt Kowalik and Elena Suleimani**

( ) Use entire base grid as Master Grid
(●) Choose portion as Master Grid (use cursor to click-and-drag)

North Latitude: 61.4166666 Max: 61.9833333
West Longitude: -160.016666 Min: -165.9833333
East Longitude: -142.350000 Max: -140.0166667
South Latitude: 52.6166667 Min: 52.0166667

Base Area: gulf_of_alaska_2min
Grid Name (to identify output): gulf_of_alaska_2min

Figure 2.5: Step 2 of the TCP interface, displaying the use of the custom mapping tool to define the master grid extent for a COMCOT model run. If exact coordinates are known, they can be entered and the selection box will snap to the defined extent.



About TCP Start Over Logout

Tsunami Computational Portal

Home | 1. Model | 2. Master Grid | 3. Sub Grids | 4. Controls | 5. Generation | 6. Metadata | 7. Submit

Step 3: Define Sub Grid

Model: **UAF Tsunami Model**
Authors: **Zygmunt Kowalik and Elena Suleimani**

Use entire base grid as Sub Grid [✓ Choose portion as Sub Grid (use cursor to click-and-drag)]

Click to drag a new grid with your mouse

🟥 Current Selection
▨ Parent Grid

North Latitude: 60.4166669 Max: 61.3333337

West Longitude: -149.816666 Min: -155.9333333

East Longitude: -147.149999 Max: -147.0166662

South Latitude: 58.6500002 Min: 55.0166667

Base Area:	cook_inlet_24sec
Grid Name (to identify output):	cook_inlet_24sec_2

Figure 2.6: Step 3 of the TCP interface, displaying the map-based subgrid selection mechanism. Subgrids must be placed within the master grid (dashed line boundary), and may not overlap or touch the boundaries of any other defined subgrids.

2.5.3 Grid Preparation for a Model Run

Once a model run is parameterized and the user has submitted the job for processing, the selected master grid and subgrids are prepared and packaged along with the metadata and parameters that define the model run. These operations take place on NACSE servers. To be compatible with the model codes running on ARSC supercomputers, all gridded datasets submitted for tsunami simulation processing must be converted to a binary file format that matches the following specification:

- Each data element must be:
  - Four bytes in size.
  - Type float or real.
  - Big-endian (MSB) byte order.
- The file must not have a header.
- The file must contain *n* data elements, where *n* is the product of the number of rows and columns.

Data to be configured as input grids for model runs are extracted from the database and packaged into the necessary binary format via a database function written in the PostgreSQL C application programmer's interface (API). The PostgreSQL C API is implemented as the libpq C library, which provides a set of functions that allow C programs to submit queries to the PostgreSQL server instance and receive the results. Performing these operations via compiled C code can be much faster than running the equivalent query using an interpreted programming language (e.g., Perl) or the PL/pgSQL procedural language. Writing the grid export functions in C facilitates the export of gridded point data directly from the database to the required binary file format (Appendix D). The clipped grids are packaged along with metadata and model run configuration parameters, and made available to ARSC for processing.

2.6 Results

The TCP system is actively used by researchers interested in modeling tsunami propagation and inundation across multiple model codes. A range of input bathymetry grids are available by default, and additional grids can be added to the system. Powerful computer hardware at NACSE, together with effective indexing of database columns, provides fast access to the input datasets for region selection and preparation. Fast supercomputers at ARSC, where configured simulation jobs are executed, guarantee quick turnaround of complex modeling scenarios.

2.6.1 Portal Usage

Over 100 TCP user accounts have been approved and established. Over 800 jobs have been configured and run by portal users since the portal was introduced. Output data have been described in technical papers and conference presentations (e.g., Barkan *et al*. 2008, Keon *et al*. 2009), and have also been used in various quantitative analyses. To investigate possible effects of a large earthquake-generated tsunami on coastal Europe, Barkan *et al.* (2009) used the portal to run multiple simulations of a large tsunami event based upon known historic information about the 1755 Lisbon earthquake-generated tsunami. Barkan and ten Brink (2010) later used the portal to model multiple simulations of the tsunami generated by the 1867 Virgin Island earthquake.

2.6.2 Input Grid Availability

Of the 30 bathymetry and coastal topography grids available in the spatial database, four are global datasets acquired and processed by NACSE personnel and the remaining 26 are of sub-global extent (and typically finer-resolution than the global grids), and were provided by researchers who used them in model runs and made them available to all portal users. The portal interface at NACSE runs on servers protected by redundant network access and backup power capabilities, which guarantees high availability. Configuring the system to store and serve input grids via a fast spatial database gives portal users effective real-time web-based access to these large datasets, as well as the built-in verification of grid spacing, alignment, and overlap protection that the database calculates dynamically using stored metadata, user input, and fast spatial queries.

2.6.3 Output Grids and Data Products

Several data output products are delivered upon completion of a model run (Table 2.5). All output data products are file-based (i.e., none are inserted into the database), but are linked to database metadata records via unique identifiers. Of particular interest for further modeling or visualization are the time series and sea level max files, each of which can be converted to an image (or a set of images or video in the case of the series file).

Table 2.5: Output data products produced by numerical codes at ARSC for each model run. These sets of files are transferred from ARSC to NACSE for post-processing and then made available to the user who submitted the job.

Ouput Data Product	**Format**	**Description**
Time series	Binary	Contains sea level height at each grid cell location for each time step. One file is returned for each input grid.
Sea level max	Binary	Contains the single highest sea level height recorded at each grid cell location across all time steps.
Model run statistics	ASCII	System-level statistics related to the model run (host, model code & version, job ID, elapsed time, etc.).
U velocity vectors	Binary	Optional output. Data describing the U velocity component for every grid cell at each time step.
V velocity vectors	Binary	Optional output. Data describing the V velocity component for every grid cell at each time step.
Input configuration	ASCII	Copy of the input configuration defining the successfully completed model run.

When the data output products are successfully transferred from ARSC to NACSE, automatically-triggered processes at NACSE unpack the files to the correct location on the filesystem and update the database to indicate the availability of the output data products. Post-processing code renders a complete set of images for the time series data (one image per time step) and automatically assembles them into a video file (Figure 2.7).

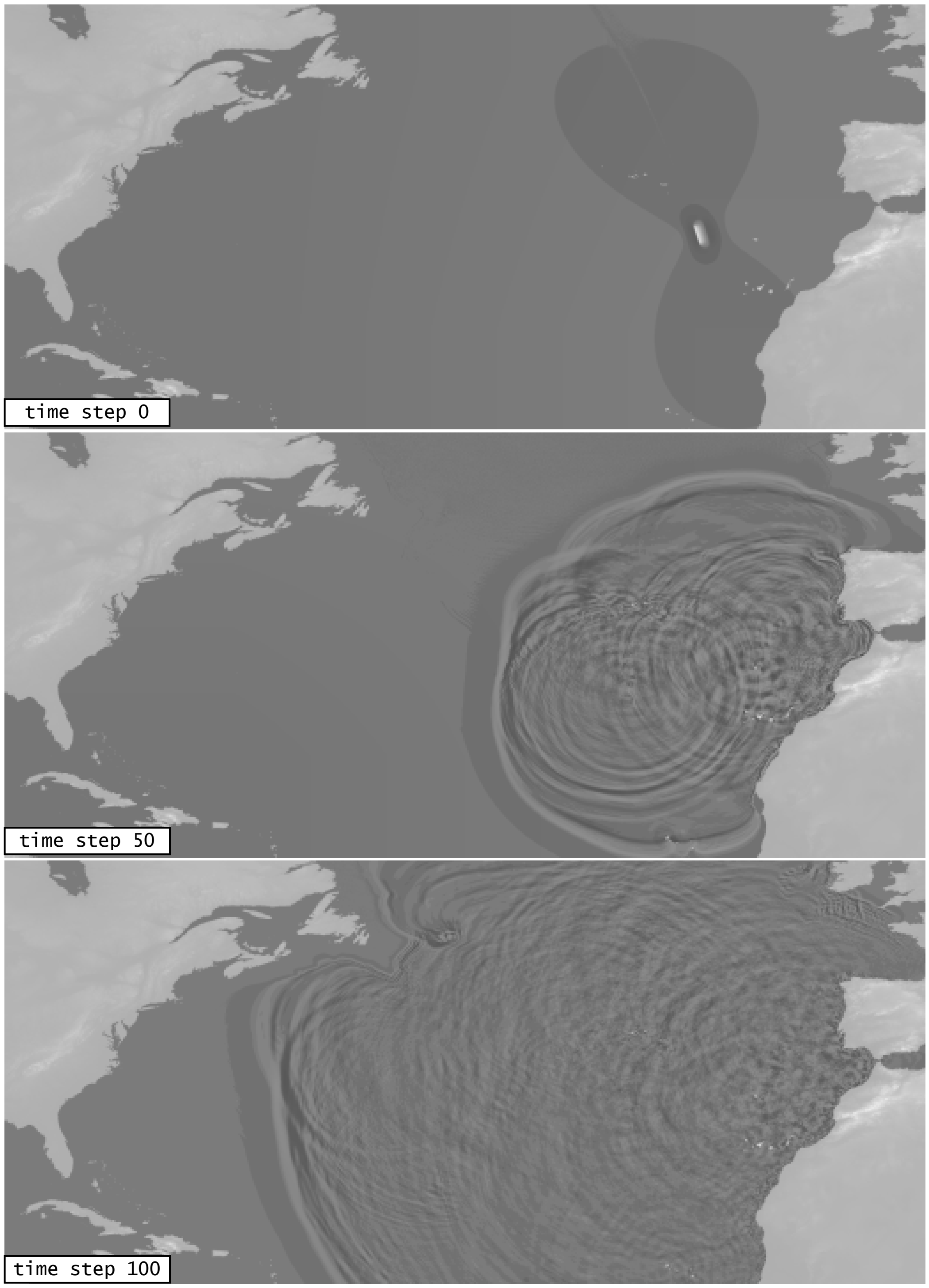


Figure 2.7: Sequence of three selected images representing tsunami propagation in a North Atlantic Ocean model run. The images were generated via post-processing of the time series data output, with one image generated for each output time step.

2.6.4 Spatial Database Performance

The grid management system developed for this project, in which the spatial database is a key component central to web-based model parameterization and preparation of clipped input grids, relies upon particular database optimizations (e.g., spatial indexes) to function at the speed necessary to support portal operations. Spatial indexing schemes, such as the R-tree-over-GiST spatial index that PostGIS functions utilize, facilitate fast querying across spatial features stored in the database (Nguyen 2009, Keon 2010). Spatial indexes are essential for enabling efficient queries against very large datasets; without them, spatial queries would resort to sequential scans of the data, which would run extremely slow in comparison.

By generating spatial indexes on all grids stored in the database, the performance of queries necessary for (1) real-time portal responsiveness and (2) gridded data export for preparation of model input grids was greatly increased. The sample comparison below demonstrates the speed of a real-world query against the GEBCO_08 grid that utilizes the spatial index vs. the identical query with the spatial index disabled (pre-pending an underscore on the PostGIS function name ST_Within disables usage of the spatial index, which simulates the absence of a spatial index on the column). In this comparison, all gridded points within a bounding box of two degrees longitude by one degree latitude were extracted:

QUERY #1 (spatial index enabled)

```
SELECT id, x, y, ST_x(geom) AS lon, ST_y(geom) AS lat
FROM gebco30
WHERE ST_Within(geom, GeometryFromText('POLYGON((-125 44,
-125 45, -123 45, -123 44, -125 44))', 4326));

28800 rows returned
query time: 342 ms
```

QUERY #2 (spatial index disabled)

```
SELECT id, x, y, ST_x(geom) AS lon, ST_y(geom) AS lat
FROM gebco30
WHERE _ST_Within(geom, GeometryFromText('POLYGON((-125 44, -125 45,
-123 45, -123 44, -125 44))', 4326));

28800 rows returned
query time: 489180 ms
```

The sample query run with the spatial index disabled results in a query execution time many orders of magnitude slower (489.18 seconds vs. 0.34 seconds) than with the spatial index enabled. Similarly, the creation of indexes on the Cartesian x and y columns of every grid table facilitates fast querying of data based on the indexed values. This is particularly beneficial for exporting gridded data as binary files to deliver to ARSC for processing, since the data are queried based on sequential x,y point position prior to conversion to binary (Appendix D).

For very large tables (e.g., GEBCO_08), a combined (i.e., “multicolumn”) index was generated based on the existing x and y column indexes, which provides much faster query response than relying solely upon the individual column indexes. In the comparison below, a random set of grid points was selected by setting a WHERE clause on the Cartesian x and y grid point location values, first on a copy of the GEBCO_08 grid for which indexes were built on x, y, and xy, and second on a copy of the same grid for which indexes were built only on x and y.

QUERY #3 (with x, y, and xy spatial indexes)

```
SELECT id, x, y, ST_x(geom) AS lon, ST_y(geom) AS lat
FROM gebco30
WHERE x < 19048 and x > 11429 and y < 98 and y > 21;

578968 rows returned
query time: 3370 ms
```

QUERY #4 (with x, y spatial indexes only)

```
SELECT id, x, y, ST_x(geom) AS lon, ST_y(geom) AS lat
FROM gebco30
WHERE x < 19048 and x > 11429 and y < 98 and y > 21;

578968 rows returned
query time: 7551 ms
```

The difference in query speed is not nearly as dramatic as in the previous spatial query example, but use of the multicolumn index decreased query execution time by more than half of the time (3.37 s vs. 7.55 s) required using single-column indexes. This is a significant performance gain that is particularly effective when querying large sets of points, or when concurrent queries place a high load on the database server. Gains in overall database performance were also obtained by modifying `postgresql.conf` parameters as described in Table 2.3. These configuration modifications are particularly helpful for write-heavy operations such as initial grid loading.

2.7 Discussion

The specialized handling required for large spatial datasets used in the TCP presents a challenging set of constraints that must be met in near-real-time scenarios. Successfully meeting those constraints and generating the inputs required by the model codes at ARSC requires tight integration among system components, as well as the development of specialized code to (1) prepare and package spatial data and metadata for simulation modeling jobs, and (2) automatically process the spatiotemporal output data into useful products.

2.7.1 System Integration

The TCP's integration of the input bathymetry grids, spatial database, and web portal system at NACSE, together with the supercomputing resources at ARSC, represents a unique implementation of a distributed scientific portal framework. A number of important factors contribute to system operations; one of the key factors is the common model parameterization scheme, which enables job configuration within a single interface (the web portal) and facilitates job processing at ARSC. Leveraging the computational resources at ARSC allows the fast computation of complex modeling scenarios that would otherwise be difficult (or impossible) to compute in a timely fashion.

Gains in multi-core processor speed may make local processing of model codes more feasible, but large tsunami modeling jobs will still require the parallel processing advantage of a large computational cluster or supercomputer. Running parallel computational tsunami modeling operations on a graphics processing unit (GPU) presents interesting possibilities for relatively low-cost, yet fast, computation across large domains. This approach was tested using the set of linear shallow-water equations by Schmidt *et al*. (2010) who, using a single GPU, found an increase in processing speed by a factor of eight over the processing speed required by a single CPU.

2.7.2 Portal Design and Spatial Data Handling

The methods described for spatial data configuration, storage, and analysis in support of tsunami computational modeling enable fast access to large quantities of spatial data both in a web-based context and for export and packaging of grids in preparation for processing. Open source database and spatial data processing software packages constitute an effective platform for importing, exporting, querying, and manipulating grids used as inputs for tsunami numerical modeling. APIs provided by some of the products give portal developers the ability to design and implement custom solutions for particular applications. GIS-based mapping and visualization aids help the user to define the extent of interest and grid nesting arrangement for a model run. As others have noted (e.g., Merati *et al*. 2010), these capabilities are extremely useful in hazard assessment and modeling tools.

Access to data and a common parameterization interface facilitates the comparison of both hypothetical and actual tsunami events across multiple models. An important benefit of the TCP system is the ability to parameterize a single simulation and run it across multiple models, all within the same interface. The TCP represents a useful implementation of a scientific web-based portal that gives users access to high-performance computing resources. The system was written specifically for this project – no framework was used in its development, although such frameworks have since been developed (e.g., Van Hemert *et al*. 2011).

Maintaining bathymetry and coastal topography grids in a relational database gives researchers easy access to the data and associated metadata via a web-based interface, and enables fast spatial queries for real-time alignment and validation checks performed by the portal interface. Spatial indexing is essential for enabling fast spatial queries in the database. The use of multicolumn indexes in particularly large grid tables also decreases query time, reducing the total time required to configure a new model run. A disk space penalty is incurred when an index is built on a grid table. For instance, the table containing the GEBCO_08 grid (including spatial objects) occupies approximately 59 GB of disk space on the production system, while the four column indexes (x, y, xy, and spatial) created for that table occupy approximately 112 GB. However, the efficiency in query speed gained by effective index usage far outweighs the associated disk space penalty.

2.7.3 Automation and Output Data Products

A key feature of the TCP system is automation – in particular, the automated processing of spatial data as inputs for analysis and of output time-series data for interpretation and visualization. While the spatial database greatly facilitates the exploration and selection of grids via the web-based interface, and the PostgreSQL C API enables fast data extraction queries via external code, the spatial database itself may not be the fastest solution for extracting and packaging data to prepare inputs for analysis. Since the initial development of the TCP system, advancements in open source GIS toolkits and scripting languages such as Python have provided improved compatibility with native formats such as netCDF, a common format used for gridded bathymetry data. It should be possible today to develop a grid extraction tool written in Python that uses the GDAL Python API to dynamically clip and extract netCDF files (i.e., read from and write to netCDF directly, which GDAL supports). However, any potential speed improvement may be insignificant in the context of the portal system, which processes jobs in a queued manner depending upon resource availability at ARSC.

Although tsunami model output data are often used to produce visualizations or quantitative comparisons of varying event conditions, the output products can also be used as inputs for visualization of simulated inundation of the modeled time-series phenomenon, or utilized in other types of simulation modeling such as human evacuation behavior during tsunami events. To support this type of simulation modeling, the model codes must calculate and return U and V velocity vectors along with the other output data. Two of the model codes currently in the TCP system support the computation of these output products as an optional feature. Other non-web-based visualization tools can also utilize the output data products as inputs. For instance, Janik *et al.* (2007) developed a Java-based software tool for direct exploration and visualization of TCP output data products. The tool can be triggered from the TCP interface via Java Web Start to begin exploration of a user's output data. Similar tools have been developed by other groups: Vance *et al*. (2007) developed a Java-based tool that utilizes the Visualization Toolkit and ESRI's ArcObjects to support exploration and visualization of tsunami simulation output data. Merati *et al*. (2010) developed Java-based tools for exploration and visualization of simulated NOAA tsunami inundation data.

2.8 Conclusion

The TCP is a unique web-based system in which three tsunami simulation model codes can be configured using a common parameterization scheme, multiple global and local bathymetry data are available for selection, and complex simulation jobs are processed quickly on ARSC supercomputers. The TCP's integration among web, database, and supercomputing systems, combined with its support of disparate computational models in a common environment, represent a model platform that can be emulated in other disciplines where a collection of computational models in a comparative environment is required.

Cyberinfrastructure is characterized by transformative computational approaches leading to technological innovations that can be used by broader communities (Atkins *et al*. 2003). The specialized resources and abilities of multiple research groups and domain experts were leveraged to develop the TCP system and provide access to multiple tsunami computational models within a common interface. This cyberinfrastructure approach has greatly expanded the audience for the models and has made it possible for the first time to perform in-depth comparisons of different models and observe the effects of alternative algorithms step by step. Model runs with iterative changes can be conducted simply by modifying configuration parameters in the web portal and submitting the jobs for processing, and the results can be compared by distributed teams of researchers.

The tsunami research community has benefitted from these capabilities. An equally notable contribution is the design of the TCP. The portal system architecture is not limited to the tsunami modeling domain – it is extensible to other computational domains such as weather modeling or ecological modeling, in which alternative modeling approaches need to be exercised broadly and compared to increase scientific understanding. Improvements to the TCP could include the porting of new modeling codes, as well as the addition of more high-resolution datasets providing bathymetry and coastal topography coverage of near-shore areas vulnerable to tsunami inundation, and more advanced visualization tools.

2.9 Acknowledgments

Development of the Tsunami Computational Portal was supported by funding from NOAA/NWS and DoD/HPCMP. Rob Steiner and Ben Steinberg contributed code to the portal and database systems at NACSE. Tom Logan ported the tsunami simulation model codes to the ARSC platform and developed model parameter and constraint information. Elena Suleimani provided several grids for the Alaska region, information about the UAF Tsunami Model code, and feedback on TCP system functionality. Philip Liu provided information about the COMCOT model code and assistance in porting it to the ARSC platform. David George and Randy LeVeque provided information about the Tsunami CLAW model code and assistance with its parameterization. Craig Stephenson provided assistance with TCP operations at ARSC.

2.10 References

ADCIRC. 2012. A (Parallel) Advanced Circulation Model for Oceanic, Coastal and Estuarine Waters, http://www.adcirc.org (last accessed 10 February 2012).

Allan, J.C., P.D. Komar, P. Ruggiero, and R. Witter. 2012. The March 2011 Tōhoku tsunami and its impacts along the U.S. West Coast. *Journal of Coastal Research* 28(5): 1142-1153.

Amante, C. and B.W. Eakins. 2009. ETOPO1 1 arc-minute global relief model: Procedures, data sources and analysis. NOAA Technical Memorandum NESDIS NGDC-24, 19 pp, March.

Atkins, D.E., K.K. Droegemeier, S. Feldman, H. Garcia-Molina, M.L. Klein, D.G. Messerschmitt, P. Messina, J.P. Ostriker, and M.H. Wright. 2003. Revolutionizing science and engineering through cyberinfrastructure: Report of the National Science Foundation blue-ribbon advisory panel on cyberinfrastructure, http://www.nsf.gov/cise/sci/reports/atkins.pdf (last accessed 24 February 2012).

Barkan, R., U.S. ten Brink, and J. Lin. 2008. Trans-Atlantic tsunamis: Simulations of the 1755 Lisbon and of hypothetical Puerto Rico trench earthquake tsunamis. American Geophysical Union (AGU) Fall Meeting, San Francisco, CA, December 15-19.

Barkan, R., U.S. ten Brink, and J. Lin. 2009. Far field simulations of the 1755 Lisbon earthquake: Implications for tsunami hazard to the U.S. East Coast and the Caribbean. *Marine Geology* 264(1-2): 109-122.

Barkan, R. and U. ten Brink. 2010. Tsunami simulations of the 1867 Virgin Island earthquake: Constraints on epicenter location and fault parameters. *Bulletin of the Seismological Society of America* 100(3): 995-1009.

Bernard, E.N. 2005. The U.S. National Tsunami Hazard Mitigation Program: A successful state-federal partnership. *Natural Hazards* 35(1): 5-24.

Bernard, E.N., H.O. Mofjeld, V. Titov, C.E. Synolakis, and F.I. González. 2006. Tsunami: Scientific frontiers, mitigation, forecasting and policy implications. *Philosophical Transactions of the Royal Society* 364: 1989-2007.

Blain, C.A. and K. Kelly. 2001. Software design description for the Advanced Circulation Model (ADCIRC). NRL Memorandum Report, NRL/MR/7320-01-8271, Naval Research Laboratory, Department of the Navy, 12/2001.

Dunbar, P. and H. McCullough. 2011. Global tsunami deposits database. *Natural Hazards* 63(1): 267-278.

Dunbar, P., H. McCullough, and G. Mungov. 2011. 2011 Tohoku earthquake and tsunami data available from the National Oceanic and Atmospheric Administration/National Geophysical Data Center. *Geomatics, Natural Hazards and Risk* 2(4): 305-323.

Garbin, D.A. and J.L. Fisher. 2010. Open source for enterprise geographic information systems. *IT Professional* 12(6): 38-45.

GEBCO. 2010. General bathymetric chart of the oceans: The GEBCO_08 grid, http://www.gebco.net/data_and_products/gridded_bathymetry_data/documents/gebco_08.pdf (last accessed 11 February 2012).

Geist, E.L., V.V. Titov, and C.E. Synolakis. 2006. Tsunami: Wave of change. *Scientific American* 294(1): 56-63.

George, D.L. and R.J. LeVeque. 2006. Finite volume methods and adaptive refinement for global tsunami propagation and local inundation. *Science of Tsunami Hazards* 24(5): 319-328.

Gisler, G.R. 2008. Tsunami simulations. *Annual Review of Fluid Mechanics* 40: 71-90.

GMT. 2012. Generic Mapping Tools, http://gmt.soest.hawaii.edu (last accessed 24 January 2012).

Goff, J.R., S.L. Nichol, and D. Kennedy. 2010. Development of a palaeotsunami database for New Zealand. *Natural Hazards* 54(2): 193-208.

Goff, J., C. Chagué-Goff, D. Dominey-Howes, B. McAdoo, S. Cronin, M. Bonté-Grapetin, S. Nichol, M. Horrocks, M. Cisternas, G. Lamarche, B. Pelletier, B. Jaffe, and W. Dudley. 2011. Palaeotsunamis in the Pacific Islands. *Earth-Science Reviews* 107: 141-146.

Goff, J.R., C. Chagué-Goff, and J.P. Terry. 2012. The value of a Pacific-wide tsunami database to risk reduction: Putting theory into practice. *Geological Society, London, Special Publications* 361: 209-220.

Gupta, A.K. 2011. Recent Tohoku tsunami and earthquake: A natural consequence of the movement of the Pacific plate along the trench, east of the Japanese island arc. *Science and Culture* 77(5-6): 175-185.

Huang, W., C.-L. Huang, and C.-H. Wu. 2006. The development of a computational grid portal. In Proceedings of the sixth IEEE international symposium on cluster computing and the grid, Singapore, May 16-19.

Imamura, F., A.C. Yalciner, and G. Ozyurt. 2006. Tsunami modeling manual (TUNAMI model), http://www.tsunami.civil.tohoku.ac.jp/hokusai3/J/projects/manual-ver-3.1.pdf (last accessed 10 February 2012).

Janik, C., M. Bailey, D. Keon, C. Pancake, and H. Yeh. 2007. Web-based tsunami visualization. 9th International Conference on Fluid Control, Measurements, and Visualization (FLUCOME), Tallahassee, FL, September 16-19.

Johnston, D., D. Paton, G.L. Crawford, K. Ronan, B. Houghton, and P. Bürgelt. 2004. Measuring tsunami preparedness in coastal Washington, United States. *Natural Hazards* 35(1): 173-184.

Keon, D., C. Pancake, H. Yeh, and T. Logan. 2009. The Tsunami Computational Portal: Distributed infrastructure for executing and comparing multiple computational models. Association of American Geographers (AAG) Annual Meeting, Cyberinfrastructure Specialty Group, Las Vegas, NV, March 22-27.

Keon, D. 2010. Database management systems. Pages 671-675 *in* B. Warf (ed.), Encyclopedia of geography. SAGE Publications, Thousand Oaks, CA.

Kowalik, Z. and T.S. Murty. 1993. Numerical simulation of two-dimensional tsunami runup. *Marine Geodesy* 16: 87-100.

Liu, P.L.-F., Y-S. Cho, M.J. Briggs, C.E. Synolakis, and U. Kanoglu. 1995. Run-up of solitary waves on a circular island. *Journal of Fluid Mechanics* 302: 259-285.

Liu, P.L.-F., S.-B. Woo, and Y.-S. Cho. 1998. Computer programs for tsunami propagation and inundation, http://ceeserver.cee.cornell.edu/pll-group/doc/comcot_technical_manual.pdf (last accessed 14 February 2012).

Lynett, P.J. 2011. Tsunami inundation, modeling of. Pages 1008-1021 *in* R.A. Meyers (ed.), Extreme environmental events: Complexity in forecasting and early warning. Springer, New York.

Lynett, P.J. and P.L.F. Liu. 2011. Numerical simulation of complex tsunami behavior. *Computing in Science & Engineering* 13(4): 50-57.

Marks, K. and W. Smith. 2006. An evaluation of publicly available global bathymetry grids. *Marine Geophysical Research* 27(1): 19-34.

Merati, N., C. Chamberlin, C. Moore, V. Titov, and T.C. Vance. 2010. Integration of tsunami analysis tools into a GIS workspace – Research, modeling, and hazard mitigation efforts within NOAA's Center for Tsunami Research. Pages 273-294 *in* P.S. Showalter and Y. Lu (eds.), Geospatial techniques in urban hazard and disaster analysis 2. Springer, Netherlands.

Murphy, K. 2012. Tsunami debris: Huge dock washes up on Oregon coast. Los Angeles Times, 6 Jun 2012, http://articles.latimes.com/2012/jun/06/nation/la-na-nn-huge-dock-tsunami-20120606 (last accessed 20 July 2012).

National Research Council. 2011. Tsunami warning and preparedness: An assessment of the U.S. tsunami program and the nation's preparedness efforts. The National Academies Press, Washington, D.C. 284 pp.

Nguyen, T.T. 2009. Indexing PostGIS databases and spatial query performance evaluations. *International Journal of Geoinformatics* 5(3): 1-9.

NOAA. 2012. Japan tsunami marine debris. NOAA Marine Debris Program, http://marinedebris.noaa.gov/info/pdf/japanfaq.pdf (last accessed 10 February 2012).

NOAA Center for Tsunami Research. 2012. NOAA Center for Tsunami Research: Tsunami Modeling and Research, http://nctr.pmel.noaa.gov/model.html (last accessed 14 February 2012).

OGR. 2012. OGR Simple Features Library, http://www.gdal.org/ogr (last accessed 24 January 2012).

Open Geospatial Consortium, Inc. 2010. OpenGIS® implementation standard for geographic information - simple feature access - part 2: SQL option. Reference number OGC 06-104r4, http://portal.opengeospatial.org/files/?artifact_id=25354 (last accessed 8 February 2012).

Perl. 2012. Perl: Practical Extraction and Report Language, http://www.perl.org (last accessed 13 February 2012).

PostGIS. 2012. PostGIS – A spatial extension to PostgreSQL, http://postgis.refractions.net (last accessed 21 January 2012).

PostgreSQL. 2012. PostgreSQL relational database management system, http://www.postgresql.org (last accessed 21 January 2012).

PROJ.4. 2012. PROJ.4 Cartographic Projections Library, http://proj.osgeo.org (last accessed 21 January 2012).

Schmidt, J., C. Piret, N. Zhang, B.J. Kadlec, D.A. Yuen, Y. Liu, G.B. Wright, and E.O.D. Sevre. 2010. Modeling of tsunami waves and atmospheric swirling flows with graphics processing unit (GPU) and radial bias functions (RBF). *Concurrency and Computation: Practice and Experience* 22: 1813-1835.

Showstack, R. 2011. Oceanographer tracks marine debris from the Japan tsunami and other incidents. *Eos Transactions American Geophysical Union* 92(37): 306.

Suleimani, E. 2004. UAF Tsunami Code, http://tsunamiportal.nacse.org/documentation/UAF_TsunamiCode.pdf (last accessed 14 February 2012).

Tang, Z., M.K. Lindell, C.S. Prater, and S.D. Brody. 2008. Measuring tsunami planning capacity on U.S. Pacific Coast. *Natural Hazards Review* 9(2): 91-100.

TCP. 2012. Tsunami Computational Portal, http://tsunamiportal.nacse.org (last accessed 15 September 2012).

Titov, V. and F.I. González. 1997. Implementation and testing of the Method of Splitting Tsunami (MOST) model. NOAA Tech. Memo. ERL PMEL-112 (PB98-122773), NOAA/Pacific Marine Environmental Laboratory, Seattle, WA, 11 pp.

Titov, V., A.B. Rabinovich, H.O. Mofjeld, R.E. Thomson, and F.I. González. 2005. The global reach of the 26 December 2004 Sumatra tsunami. *Science* 309: 2045-2048.

Van Hemert, J., J. Koetsier, L. Torterolo, I. Porro, M. Melato, and R. Barbera. 2011. Generating web-based user interfaces for computational science. *Concurrency and Computation: Practice and Experience* 23: 256-268.

Vance, T.C., N. Merati, S.M. Mesick, C.W. Moore, and D.J. Wright. 2007. GeoModeler: Tightly linking spatially-explicit models and data with a GIS for analysis and geovisualization. *In* Proceedings of the 15th ACM International Symposium on Advances in Geographic Information Systems (ACM GIS 2007), Seattle, WA, 7-9 November.

Yeh, H.H. 1991. Tsunami bore runup. *Natural Hazards* 4(2): 209-220.

Zhang, L. and J. Yi. 2010. Management methods of spatial data based on PostGIS. Pages 410-413 *in* Proceedings of the Second Pacific-Asia Conference on Circuits, Communications and System (PACCS), August 1-2.

Chapter 3: Web-based simulation modeling and visualization of tsunami inundation and potential human response

Authors: Dylan B. Keon, Benjamin M. Steinberg, Harry H. Yeh, Cherri M. Pancake, Dawn J. Wright

3.1 Abstract

Modeling spatiotemporal phenomena can provide insight into the potential behavior of simulated objects during hypothetical events. Simulation frameworks are one means of modeling such scenarios, and become more flexible when developed in a generalized fashion that facilitates the automated generation of output based on variable input parameters. By wrapping a simulation framework within a web-based system, users can not only assign input parameters of their choosing, but also run a simulation and explore output data in a dynamic, animated, map-based context.

The framework described here utilizes tsunami simulation data and user input to generate a combined web-based tsunami visualization and simulation model of human response to tsunami inundation. Input parameters that define the human population and response are provided by the user and guide the automated development of a simulation model scenario of spatiotemporal human response to a hypothetical tsunami inundation event. Simulated human movement is calculated per time step on the processing server using casualty model algorithms informed by behavioral research and variables such as water depth and road networks, while a mix of server-side and client-side code renders the mapping interface and supports user interaction within the web browser. Interactive controls included in the web-based simulation viewer allow the user to manipulate the map display and query the underlying data either manually by time step or interactively while the animation is underway.

Vector-based simulation data stored as files or spatial objects in a relational database (PostgreSQL with PostGIS) are overlaid on raster data and accessed on-the-fly to render the animation, which is displayed in a web interface developed using PHP and open source mapping tools such as OpenLayers, MapServer, and GDAL. The server-side simulation framework is written in a mix of custom C++ and Python code.

In a web-based system, an effective user experience is defined to a large extent by the responsiveness of the web-based interface. To that end, performance enhancements on both the server- and client-side were addressed, including web and database server configuration, database indexing, and spatiotemporal data representation.

3.2 Introduction

Property damage and potential loss of life due to tsunami inundation are relevant concerns in coastal communities. The 2004 Indian Ocean tsunami and the 2011 Tōhoku tsunami both caused extreme damage to coastal communities and raised awareness of tsunami danger around the world (Titov *et al*. 2005, Geist *et al*. 2006, Dunbar *et al*. 2011, Gupta 2011). Simulating potential human response to a modeled tsunami inundation event can lead to a better understanding of how a population might move in such a scenario. The development and implementation of an automated simulation framework system to process multiple inputs and produce simulation output for use in visualization represents a significant research challenge, one that is addressed in this study.

3.2.1 Tsunami Evacuation

Although U.S. states along the Pacific coast have tsunami awareness programs and evacuation maps in place (e.g., Oregon.gov 2012), tsunami hazard management plans in many coastal counties are lacking in implementation (Tang *et al*. 2008, Lindell and Prater 2010). Several different types of tsunami modeling and human evacuation scenarios have been conducted for areas along the U.S. Pacific Northwest coast (e.g., Geist and Parsons 2006, Dominey-Howes *et al*. 2010, Priest *et al*. 2010, Schneider 2011, Karon and Yeh 2011). Simulating potential human response during a tsunami evacuation scenario can help researchers and local planners understand risks to the community under varying population composition and tsunami inundation conditions. Designing a unified interface that allows users to conduct simulation scenarios, examine the output data, and view the results is a challenge, and work remains to effectively simulate human movement during tsunami evacuation scenarios and visualize output in a user-friendly fashion. Automating the production of simulation output based on user input is an added research challenge. This study examines the town of Seaside, Oregon as a test case for modeling tsunami inundation and human movement during a tsunami evacuation scenario generated by a custom simulation framework controlled via a web-based interface.

3.2.2 Simulation Modeling

To model dynamic phenomena, one must consider change in objects and their attributes across both space and time. Complexities arise from spatial and temporal representation of the objects and their attributes, analysis and characterization of interactions among objects, effective visualization, and usability of simulated output. Fast processing of potentially large datasets presents an additional challenge in applications for which real-time responsiveness is required. In particular, modeling potential human movement across space and time is a challenging task that is important to many research fields (Yu and Shaw 2008). Representation of spatiotemporal data in relational databases is a longstanding research challenge (e.g., Yuan *et al*. 2005, Le 2012), and many approaches have been devised for analyzing spatiotemporal data describing human movement (summarized in Andrienko *et al*. 2011). Through the use of specific computational techniques, relational database engines, and effective visualization methods, such analyses can provide both understanding and prediction of potential human movement (Andrienko *et al*. 2007, Yu 2007).

Computational simulation frameworks can provide a useful means of addressing the challenges inherent to analysis of spatiotemporal data and movement prediction. Properly designed, a simulation framework can efficiently process large amounts of spatiotemporal data, predict potential human movement, and help the user interpret output to provide better understanding of the modeled event. Simulation frameworks are typically designed to be run in a “closed” system (i.e., in specialized software running on the researcher’s computer or on a localized computer network). A researcher might define a set of input parameters, run a simulation, and interpret the results as raw data in a spreadsheet, or perhaps in a visualized form in their specialized software package. Simulation frameworks become more flexible when developed in a generalized manner that facilitates the automated generation of simulation output based on variable, user-defined input parameters. Enabling input/output access to a simulation framework from within a web-based interface gives researchers the ability to access and run simulations without requiring any specialized software on their computer apart from a standard web browser. In this manner, a user can guide the input parameterization of the simulation, as well as the visualization of simulated output, all from within a common user interface. User interactivity is an important feature common to effective web-based visualizations. Giving the user the ability to interactively guide the work of the system can improve a simulation by incorporating the user’s expert knowledge (Andrienko *et al*. 2007).

3.2.3 Spatiotemporal Data Representation

Effective representation and display of time-series data in a spatial context is a challenging problem that has been explored in many forms (Miller 1991, Langran 1992, Yuan 1996, Peuquet 2002, Kapler and Wright 2005, Yu 2006, Yu and Shaw 2007, Miller and Bridwell 2008). The earliest approach, developed by Hägerstrand (1970) and often used today in 3D spatiotemporal geovisualizations, is the concept of a space-time prism with space as the horizontal dimension and time as the vertical dimension (Figure 3.1).

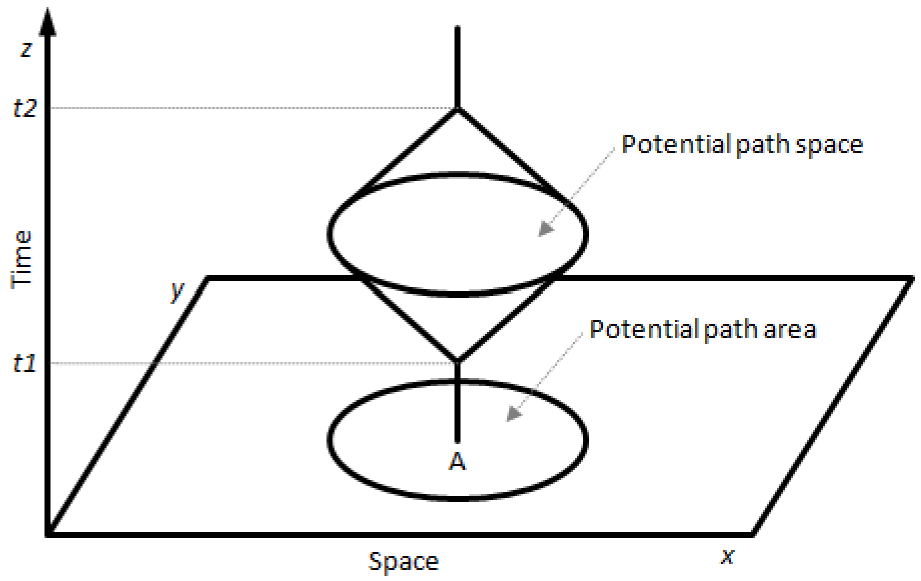


Figure 3.1: Conceptual diagram representing the space-time prism (after Miller 1991, Raubal *et al*. 2007).

Although the space-time prism is effective for visualizing relationships in space and time, it is less effective at helping the user understand the actual spatial context of temporal phenomena. Conversely, although static cartographic maps are very effective at displaying spatial context and can convey some spatiotemporal relationships to the viewer when temporal snapshot data are overlaid, static maps are inherently limited in effective representation of changing temporal information. Well-designed dynamic or animated maps can better express spatiotemporal relationships than static cartographic maps. Although map-based animation of movement over time is sometimes characterized as having limited effectiveness (e.g., Andrienko *et al*. 2011), map-based approaches remain one of the best methods of data visualization and exploration, particularly when developed as dynamic, interactive systems that integrate spatiotemporal data. Such data are often represented as an individual's movement over time, in the form of space-time paths (Hägerstrand 1970, Kraak 2003, Shaw *et al*. 2008) or geospatial lifelines (Mark 1998, Hornsby and Egenhofer 2002). Figure 3.2 displays a generalized diagram of a space-time path.

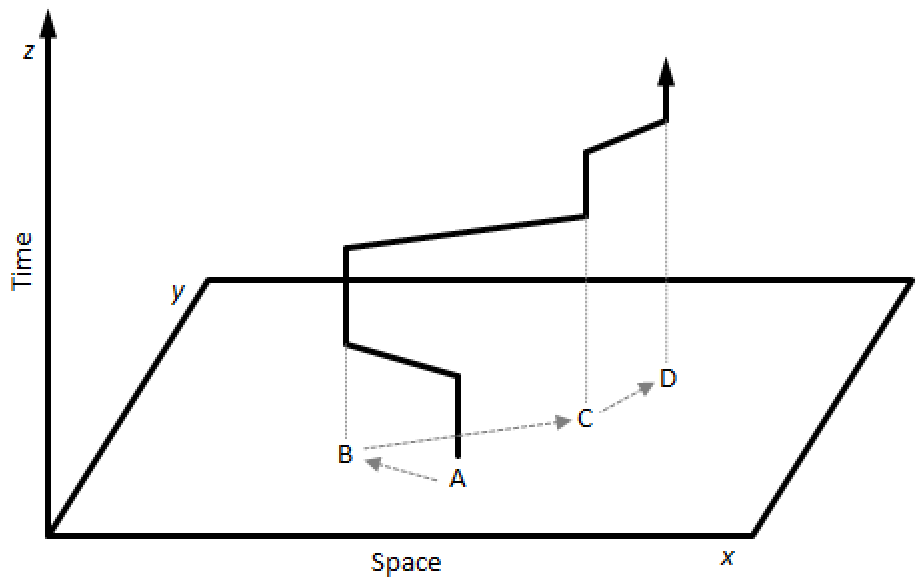


Figure 3.2: Conceptual diagram representing an individual's space-time path (after Miller 1991).

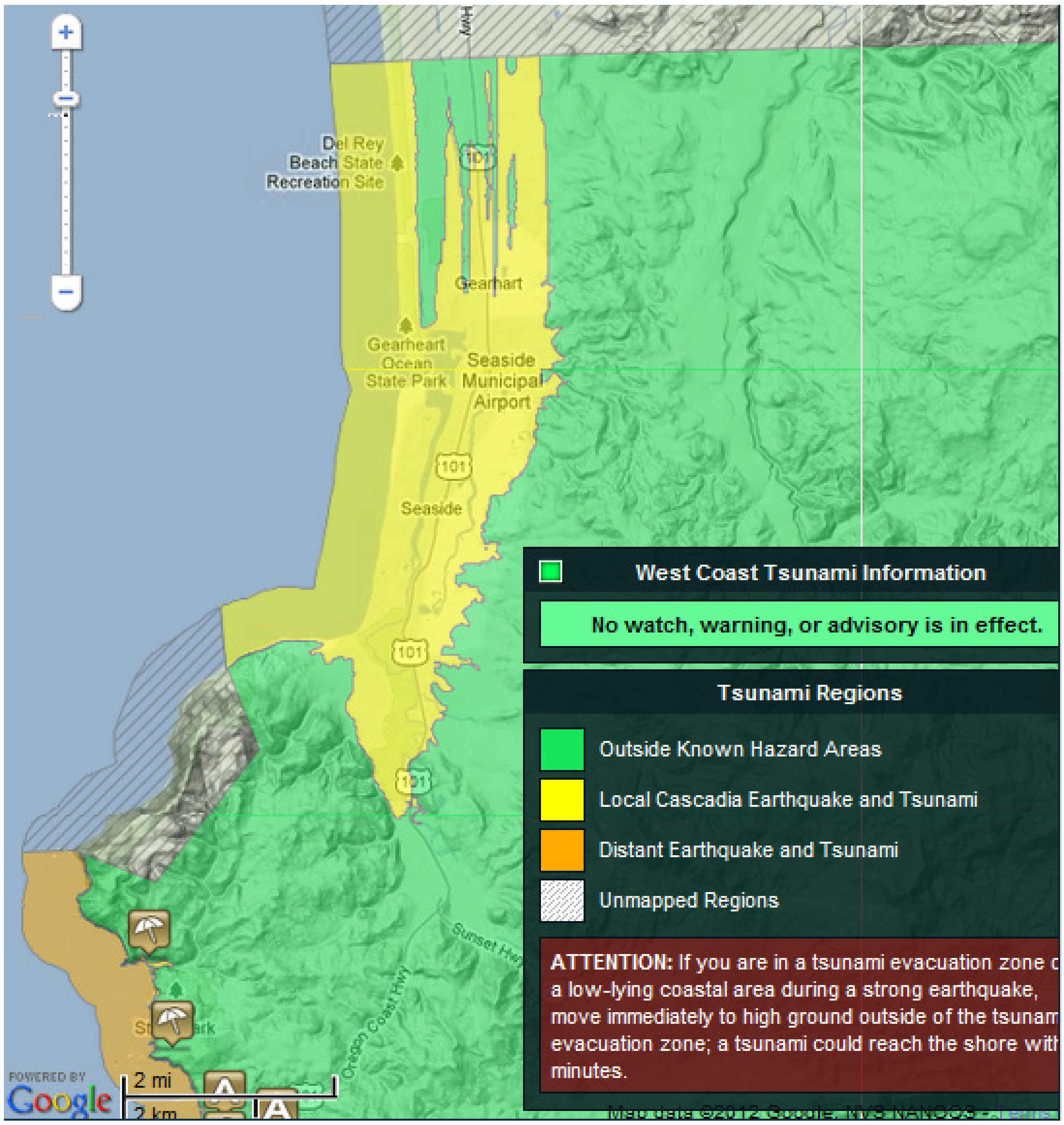
3.2.4 Animation of Spatiotemporal Data

Animated representations of spatiotemporal phenomena are often produced as static video files or simple Adobe Flash-based animations. Interactivity can be programmed into advanced Flash-based animations via frameworks such as Flex (Adobe 2012), for example, and useful tools and toolkits can be constructed in this manner (e.g., Van Ho *et al*. 2011). However, the user is often limited to a set of fixed choices for manipulating the display of maps or content in the animation. Moreover, controls for querying or retrieving information may not function while an animation is in progress, and Flash itself is susceptible to security vulnerabilities. Although non-interactive data visualizations may be useful, they can limit the level of information and understanding the user gains from the experience. User interaction with the underlying data is typically limited to querying static layers in a non-animated system or running pre-defined queries in a system that may support animation, and support for in-depth exploration of spatiotemporal datasets within web-based applications remains limited. Advances in web-based technology and both client-side and server-side processing enable the development of sophisticated query and visualization tools that can run within a web browser.

3.2.5 Evacuation Modeling

Time-series phenomena such as modeled tsunami inundation events provide good test cases for the development of interactive web-based visualizations that are both data-rich and visually interesting. Tsunami simulation modeling and visualization of results are important methods for helping researchers understand the potential behavior of a tsunami flow. It is also important to understand how people may respond to a tsunami inundation event or disaster warning. Bernard *et al*. (2006) noted: "Little research has been done on the human response to tsunamis. These studies would represent a new frontier in modelling the response of humans to this threat." Similarly, Lindell and Prater (2010) called for research integrating computer-based analysis tools for evacuation simulation and planning in Oregon and Washington.

In this study, a prototype system was designed that utilizes modeled tsunami simulation data representing water depth and location in both space and time as an input into a custom-built simulation framework. The Seaside, Oregon region, which is at risk of tsunami inundation from a local Cascadia earthquake and tsunami (Figure 3.3), was used as a test case in the prototype development. Many coastal communities in the U.S. Pacific Northwest have nearby topographic features that rise significantly above sea level, providing locations for individuals to ascend to safety in the event of a tsunami. The community of Seaside remains considerably flat for a significant distance inland, meaning an incoming tsunami flow could potentially travel quite far, making evacuation in the event of a tsunami more difficult than in other coastal communities.


Del Rey Beach State Recreation Site
101
Gearhart
Gearheart Ocean State Park
Seaside Municipal Airport
101
Seaside
101
101
Sunset Hwy
Oregon Coast Hwy
POWERED BY Google
2 mi
2 km
Map data ©2012 Google, NWS NANOOS
West Coast Tsunami Information
No watch, warning, or advisory is in effect.
Tsunami Regions
🟩 Outside Known Hazard Areas
🟨 Local Cascadia Earthquake and Tsunami
🟧 Distant Earthquake and Tsunami
▨ Unmapped Regions

ATTENTION: If you are in a tsunami evacuation zone o a low-lying coastal area during a strong earthquake, move immediately to high ground outside of the tsunam evacuation zone; a tsunami could reach the shore with minutes.

Figure 3.3: Tsunami risk area in the Seaside, Oregon region (NANOOS 2012).

Pre-existing high-resolution tsunami inundation data for the Seaside region were used during prototype development, but the system can also utilize modeled tsunami data created as output products from the Tsunami Computational Portal (TCP), a web-based research portal that provides generalized access to multiple tsunami computational codes and runs model simulations on supercomputers. The modeled data are combined with data describing a hypothetical population, structures, roads, and other features to simulate potential human response to a tsunami inundation event.

3.2.6 Study Goals

The prototype system was developed with six primary goals in mind:

1. Develop a framework for simulating potential human response to a modeled tsunami inundation event.
2. Automate the production of simulation output and geovisualizations that incorporate the output data products.
3. Create a web-based system to interface with the simulation framework, so that all framework features are accessible and configurable via a web browser.
4. Enable user interactivity features to help drive simulation generation, and to enhance usability of simulation output.
5. Develop a web-based mapping and visualization tool capable of animating time-series data in a spatial context, while allowing simultaneous use of map navigation and query functions to examine the data driving the animation.
6. Attempt to build and integrate the simulation framework, web-based system, and related processing code using open source software tools.

A number of techniques were employed to create the simulation framework, including a range of computational, database, and mapping and visualization technologies. The simulation framework models not only potential human movement but also the interaction of moving objects (wave inundation and human movement) over time.

3.3 Related Work

Simulation frameworks, tsunami modeling, visualization, modeling of human movement, and other topics addressed in this study are active areas of research across multiple disciplines. In this section, related research is discussed in the context of the main themes explored by this study.

3.3.1 Simulation Frameworks

Advances in computing technology have made it possible to simulate large, complex, dynamic systems using relatively inexpensive hardware resources. By linking custom software tools with existing software packages, a powerful framework for processing inputs and generating simulation outputs can be produced. Simulation frameworks have been developed for modeling behavior in many disciplines; they are commonly developed in the computational sciences for research projects such as simulating wireless sensor network performance in adaptive environments (Niazi and Hussain 2011) or modeling cloud computing configuration and resource provisioning (Calheiros *et al*. 2010). Simulation frameworks have also been implemented in the biological sciences (e.g., Dalquen *et al*. 2012), the agricultural sciences (e.g., McCarthy *et al*. 2010), and other research areas. The simulation framework developed for this study uses a uniquely multidisciplinary approach in its design and implementation, incorporating elements from geography, tsunami simulation, hydrodynamics, structural engineering, and human factors analysis.

3.3.2 Tsunami Modeling and Visualization

Output data from tsunami modeling serve as an important input data source for the simulation framework developed for this study. Tsunami modeling involves the use of complex computational codes to simulate tsunami wave generation, propagation across the ocean, and inundation on shore. A number of authors have described the research behind tsunami modeling and the related algorithms and computational processing requirements (e.g., Geist *et al*. 2006, Synolakis and Bernard 2006, Gisler 2008, Lynett 2011, Lynett and Liu 2011). Tsunami computational codes that model tsunami propagation across the ocean often process multiple nested grids at increasingly higher spatial resolutions as they progress toward near-shore areas. These computational tasks can require significant processing power (i.e., supercomputers or computational clusters) to reach completion in a reasonable amount of time.

Several tsunami computational models have been developed and are in use today. One of the most commonly used models is NOAA's Method of Splitting Tsunami

(MOST), a suite of numerical codes for modeling tsunami generation, propagation, and inundation (Titov and González 1997). The TUNAMI-N2 (Tohoku University's Numerical Analysis Model for Investigation of Near field tsunamis) model is also commonly used for developing tsunami simulations (Imamura *et al*. 2006). The Tsunami Computational Portal (Keon *et al*. 2009, TCP 2012) incorporates three tsunami computational model codes (COMCOT, UAF Tsunami, and Tsunami CLAW) that share a common parameterization scheme and are made available to researchers via a web-based portal interface. The TCP's output data products can be used as inputs by the simulation framework developed for this study.

In addition to tsunami computational models, NOAA researchers have developed a simulation framework that integrates modeled tsunami simulation output with data analysis and visualization capabilities in a GIS-based system (Vance *et al*. 2007). Their Java-based software application uses Esri's ArcEngine and ArcObjects products, as well as the Visualization Toolkit 3D API, and links directly to the Regional Ocean Modeling System (ROMS) and MOST models to visualize and analyze tsunami runup data. NOAA has also developed Python- and ArcGIS-based tools to process and visualize modeled tsunami propagation and inundation data (Merati *et al*. 2007). More recently, NOAA programmers developed the Community Model Interface for Tsunami (ComMIT) tool, a Java-based application that allows a user to control input parameters for the MOST model and access the model output files for visualization and analysis, and the Tsunami GIS application, a custom

ArcGIS-based tool that facilitates the integration of gridded tsunami inundation output data with vector-based socioeconomic and infrastructure data (Merati *et al*. 2010). U.S. Geological Survey personnel developed a GIS-based approach for modeling sea level rise (including potential surge effects from tsunamis) at global, regional, and local scales via statistical summary data representing elevation, land cover, and population, and produced non-interactive animations of their simulation output (Usery *et al*. 2010).

3.3.3 Modeling Human Movement

Human movement across space and time can be represented using time geography concepts such as those described by Hägerstrand (1970) and Miller (1991). By implementing Hägerstrand's space-time prism using a network-based approach, an individual's potential path can be calculated and represented visually in a two-dimensional GIS environment (Miller 1991, Miller and Wu 2000) or in a three-dimensional GIS (Yu 2007, Yu and Shaw 2008). With spatiotemporal data describing human movement, that individual's space-time path can be visualized, queried, and summarized (Mountain 2005). The context in which a person's movement occurs can affect the spatial and temporal relationships that influence that person's decision-making process. Andrienko *et al*. (2011a) developed a conceptual framework and general taxonomy of techniques for analyzing movement data, and Andrienko *et al*. (2011b) analyzed movement data in their spatiotemporal context in order to describe relationships among objects as events.

In simulated evacuation scenarios, human movement is often modeled along existing road networks leading to evacuation points (e.g., tsunami shelters on high ground in coastal communities). The shortest-path method is commonly used to model movement along a route from start to end point (Katada *et al*. 2006, Chen and Zhan 2006), although dynamic routing and agent-based modeling approaches are increasingly used (Liu *et al*. 2008, Goto *et al*. 2012, Mas *et al*. 2012, Nagarajan *et al*. 2012). Network flow models (e.g., Cova and Johnson 2003) can be used to develop optimal routing plans for human movement in traffic-based scenarios.

3.3.4 Tsunami Evacuation Simulation

Simulating an evacuation simulation in response to a disaster such as a tsunami can be treated as a specific case of modeling human movement. Katada (2003) researched tsunami inundation events and human behavior in disaster situations and designed a simulation system to predict human movement during hypothetical evacuation scenarios. His work was influenced by Oikawa and Katada (1999), who studied the effects of flood experience and human behavior on evacuation scenarios, and Katada *et al*. (2000), who developed an early, related version of a scenario simulator for dissemination of disaster warnings. As part of the work that followed Katada (2003), a tsunami scenario simulator was developed, which took into account human factors engineering and other simulation variables such as warning transmission type and timing (Katada *et al*. 2004, Wolman 2005). Later, Katada *et al*. (2006) developed a GIS-based version of the simulation framework and used it to produce several evacuation scenarios for the Owase, Japan area. In a related effort, the researchers developed a web-based interface to present the Owase simulation output, incorporating non-interactive video files that portray four individual, pre-configured simulations of human response to a tsunami event (OWASE 2012). These simulations were used by Goto *et al*. (2010) for disaster reduction education and research. Uno and Kashiyama (2008) developed a disaster evacuation simulation system based on a multi-agent model, using GIS data and routing algorithms to model human movement. Goto *et al*. (2012) also developed a multi-agent simulation model. Using a modeling approach similar to Katada *et al*. (2006), Karon and Yeh (2011) conducted a simulation of human response to a tsunami inundation event in the Cannon Beach, Oregon area. Their simulation framework was implemented as a system that generated animated output as video files representing various scenarios (Tappister 2012). All of these simulation systems were developed as locally-installed software products and produce video-based output that cannot be queried for further information. Moreover, they are limited in the use of tsunami inundation modeling data as an input to inform the simulation of water depth at each time step.

Similar to Katada's warning dissemination research, Nagarajan *et al*. (2012) developed an agent-based model of multiple agents (households) in a hypothetical community to simulate the influence of human behavior on message transmission. Clerveaux *et al*. (2008) used the Katada *et al*. (2006) tsunami evacuation scenario tool for simulating a tsunami inundation event in the Turks and Caicos Islands, focusing on warning message dissemination challenges in multi-language communities. Drawing upon community input, Kobayashi *et al*. (2007) used a unique disaster simulation system with a tangible user interface to support collaborative planning of disaster warning and damage prevention. Liu *et al*. (2007) worked with community members to help understand tsunami evacuation behavior and used their input, in part, to help design a multi-agent evacuation simulation system. Mas *et al*. (2011) examined and modeled the evacuation decision process based on human risk perception in tsunami scenarios and, using modeled tsunami wave output data from the TUNAMI-N2 model combined with an agent-based modeling approach, Mas *et al*. (2012) simulated a potential evacuation scenario for the Arahama Town area in Japan, which was inundated during the destructive 2011 Tōhoku tsunami.

3.3.5 Casualty and Loss Estimation

The modeling of human movement in response to a tsunami inundation event must incorporate some method of determining the conditions under which a person would be overtaken by the tsunami flow. Many variables must be considered when estimating casualties in this manner. Koshimura *et al*. (2006) developed a tsunami casualty estimation method based on a simple model of hydrodynamic forces as they affect the human body, tested the method using tsunami simulation data for the Seattle, WA area, and produced a tsunami casualty index to show the casualty potential at a location. While they did consider a person's weight in their casualty estimation equation, they did not consider other anthropometric variables. Yeh (2010) developed a tsunami casualty estimation method that takes into account gender and age factors. Using mean anthropometric data values from Kroemer *et al*. (1997) including height, weight, and eight body measurements (foot width, shoulder breadth, etc.), Yeh developed equations to estimate casualties while considering two different failure modes (force balance and moment balance). Both Koshimura *et al*. and Yeh made the assumption that when a person is swept off their feet by the tsunami flow, they are considered a casualty. Yeh's tsunami casualty estimation method is implemented in the simulation framework developed for this study.

Several similar efforts have developed methods for estimating loss of life due to flooding. Jonkman *et al*. (2008) developed estimation methods for different types of floods in different regions, and related the loss of life to flood characteristics and possibilities for evacuation. They also conducted a comprehensive review of extant methods for casualty estimation due to flooding, including human instability in flowing water from tsunami inundation. Usery *et al*. (2010) considered tsunami surge effects at multiple scales and the resulting impact on population. A number of studies have developed computer-based flood evacuation models and simulations, or examined the use of integrated geographic information systems for flood risk assessment and management (e.g., Zerger and Wealands 2004, Simonovic and

Ahmad 2005, Maantay *et al*. 2010). Remotely sensed data have also been used for estimating coastal vulnerability (Koshimura and Takashima 2005).

In addition to the human impact of tsunamis, it is also important to predict the potential impact of tsunamis on the built environment. GIS-based approaches are often used for modeling the vulnerability of buildings to tsunami inundation (Wood and Good 2004, Tarbotton *et al*. 2012), and Leone *et al*. (2011) developed a spatial model of tsunami risk assessment. The Papathoma Tsunami Vulnerability Assessment (PTVA) model establishes an assessment equation based on building design, condition, and surroundings (Papathoma and Dominey-Howes 2003). Revised twice, the model has been used to conduct building assessments of several study areas around the world (Papathoma *et al*. 2003, Dall'Osso *et al*. 2009, Dall'Osso *et al*. 2010, Dominey-Howes *et al*. 2010). Other numerical modeling approaches to vulnerability assessment have been developed (e.g., Wood and Good 2004, Omira *et al*. 2010), and Eckert *et al*. (2012) used digital surface models derived from remotely sensed data to assess tsunami risk to structures in Egypt. Increasingly, "fragility functions" are used to mathematically estimate the probability of house/structural damage due to tsunami inundation (Koshimura *et al*. 2009a, Koshimura *et al*. 2009b, Reese *et al*. 2011, Suppasri *et al*. 2012).

3.4 Simulation Framework

The simulation framework developed for this study includes the software and custom code that processes input data, runs the simulation model, generates output data, and runs the web-based results interface. To simulate potential human movement in response to a tsunami event, the simulation framework must:

- Read large gridded tsunami inundation input datasets.
- Generate an initial population distribution for the community of interest, place people in known residential and commercial structures, and write the data to a spatial database.
- Route the population along an established road network toward identified evacuation locations.
- Compute the intersection of tsunami inundation data with population locations at each time step (i.e., simulated moments in time, separated by a defined interval) and, using human factors data and a casualty determination algorithm, compute whether casualties occur and write the spatiotemporal data to the database.
- Generate visualizations of the output in an interactive map-based context that allows data exploration.

3.4.1 Framework Architecture

The simulation framework is composed of the hardware (processing and database servers), software (custom server-side and client-side code, APIs, relational database management system), and input data (tsunami inundation, GIS framework) that run the simulation model and produce outputs used in queries and visualizations. Figure 3.4 provides a diagrammatic description of the framework architecture.

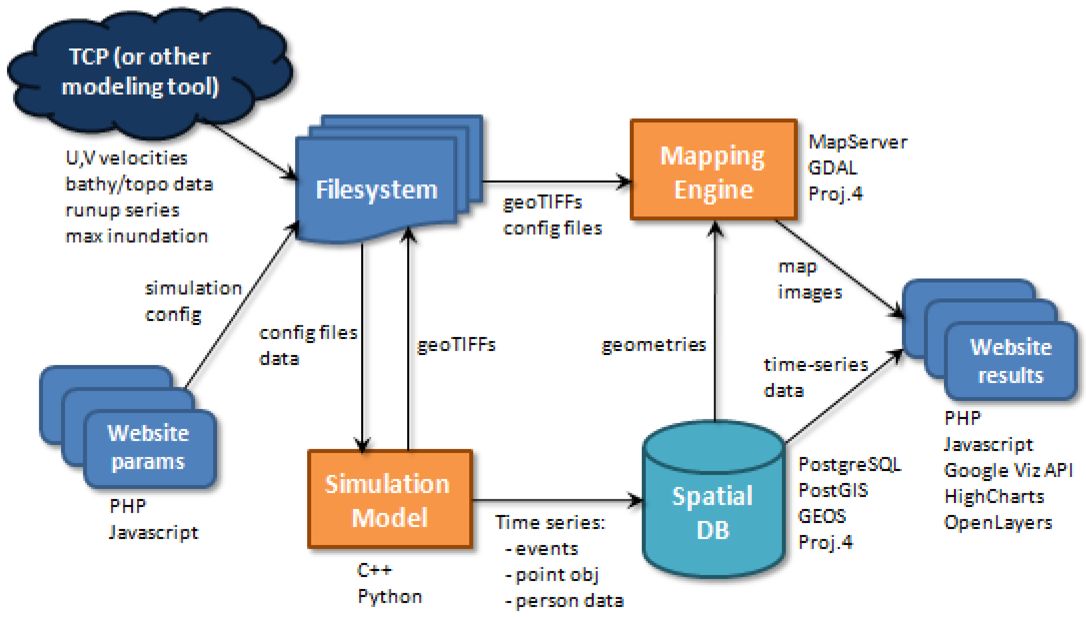


Figure 3.4: Diagram of the simulation framework system architecture. Software packages used are listed beside each diagram element, while objects passed from one element to another are listed beside the arrows. Tsunami inundation input data are generated via a separate tsunami modeling framework (the TCP or another modeling tool).

One goal of this study was to attempt to build the entire simulation framework using open source software tools, in addition to custom code. The open source software packages used by the simulation framework are listed in Table 3.1, along with the purpose for which each package was used. Custom code for the study was written in the programming languages C++, Python, PHP, JavaScript, and HTML, as well as the pl/pgSQL procedural database language.

Table 3.1: Open source software packages used by the simulation framework, and the purpose for which they were used.

Software Package	**Purpose**
PostgreSQL *(postgresql.org)*	Storage and management of input and output data used by the simulation framework.
PostGIS *(postgis.refractions.net)*	Support for spatial objects and operations within PostgreSQL, enable spatial queries.
GEOS *(geos.osgeo.org)*	OpenGIS Simple Features operations used within PostgreSQL/PostGIS.
PROJ.4 *(proj.osgeo.org)*	Coordinate system transformations on spatial objects stored in the database.
GDAL *(gdal.org)*	Process water depth data into TIFFs, and process other raster datasets.
OGR *(gdal.org/ogr)*	Process vector datasets, convert vector output from PostgreSQL/PostGIS to GeoJSON format.
MapServer *(mapserver.org)*	Prepares map layers on the server for inclusion in the OpenLayers web mapping and visualization interface.
OpenLayers *(openlayers.org)*	Enables the web mapping and visualization interface including base maps, vector data, and animated results.
Highcharts *(highcharts.com)*	Client-side charting – enables interactive charting capabilities via JavaScript.
Python *(python.org)*	Scripting language used for server-side tasks in the simulation framework.
PHP *(php.net)*	Server-side web programming language that drives the simulation results web interface.

3.4.2 Spatial Database

Spatiotemporal data describing human movement are represented in the database by time-stamping at the record level – each simulated person is represented as a database object that is assigned a set of time stamps, locations, and status data by the simulation framework. This approach provides greater relational flexibility than the traditional "snapshot" method of storing a separate GIS layer to represent each time step, yet, with proper indexing, also allows for extremely fast queries. Figure 3.5 displays a generalized diagram of the database schema used to store data that describe the initial population distribution, as well as the time-series data that describe the state of the simulated objects at each time step of the modeled tsunami event. All frequently-used columns in the database tables are indexed for fast query performance. GiST (Generalized Search Tree) indexes are created on all columns storing spatial geometries, and can greatly increase the performance of spatial queries.

The database used by the simulation framework is stored and managed in an instance of the open source PostgreSQL (2012) relational database management system with the open source PostGIS (2012) module included to support the storage, manipulation, and querying of spatial data as database objects. PostgreSQL/PostGIS allow the execution of advanced spatial queries and can be accessed directly via web programming languages such as PHP and web mapping tools such as MapServer (Lime 2008, MapServer 2012). The PostgreSQL database server instance runs on a dedicated Dell PowerEdge R710 server containing four quad-core 2.53 GHz Intel Xeon processors and 32 GB memory, with data stored across eight 320GB 10,000 RPM serial-attached SCSI disks in a RAID 5 configuration (via a hardware RAID controller). The server runs the Red Hat Enterprise Linux 6 operating system.

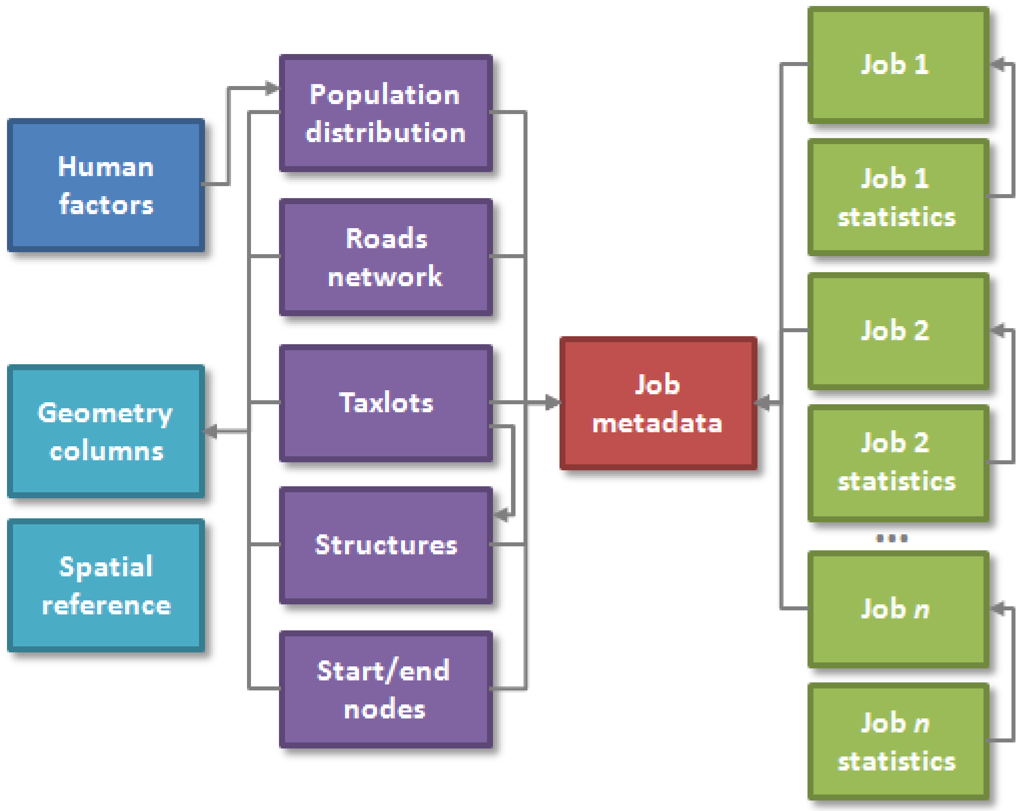
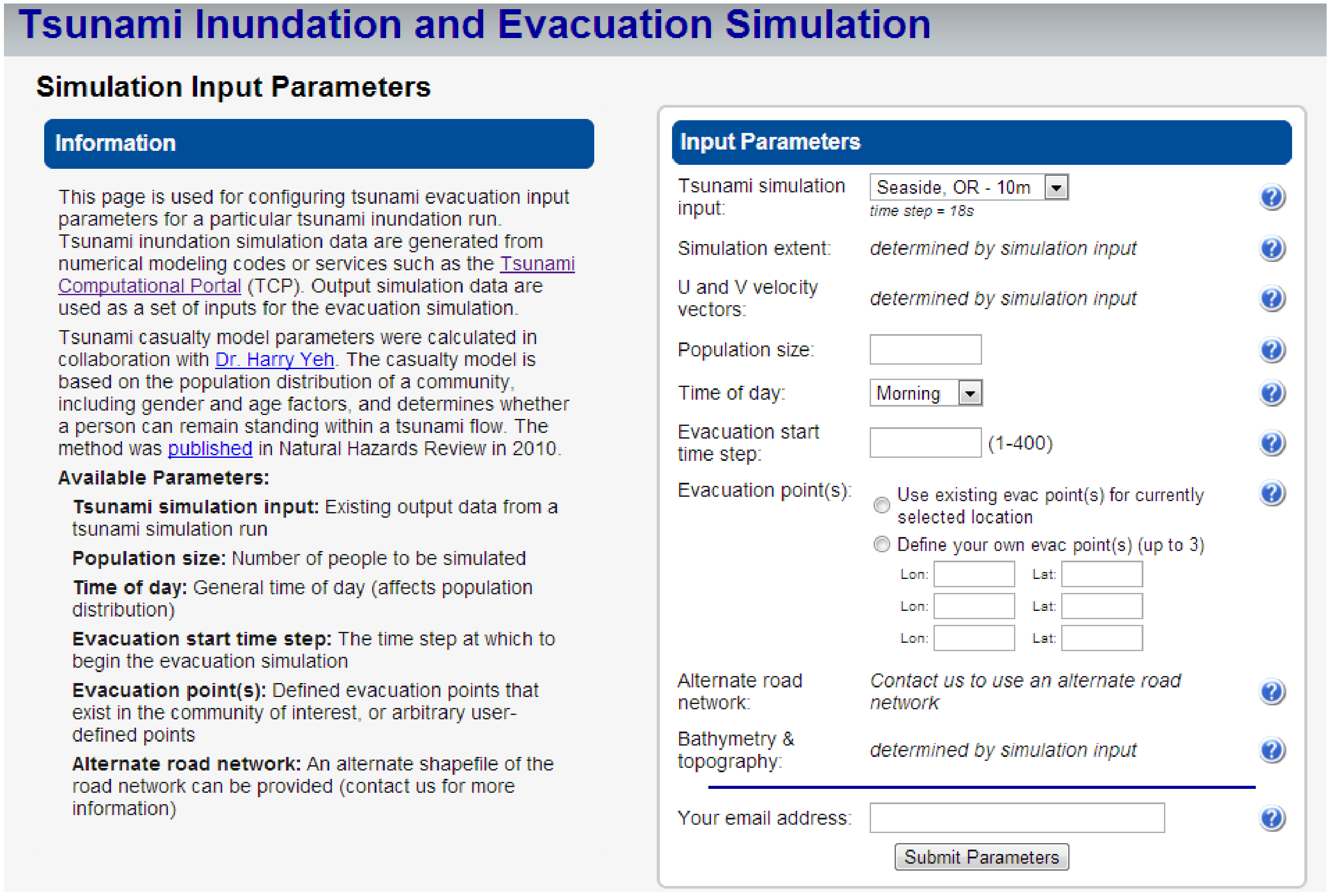


Figure 3.5: Generalized database schema diagram representing the major database tables used by the simulation framework. Simulation outputs (including time-series data) are written to various "Job" tables with matching IDs that relate back to the metadata used to generate the simulation run.

3.4.3 Input Data Sources and Data Preparation

Input data for the simulation framework come from three primary sources: (1) Input parameters defined by a user on the settings web page (Figure 3.6), (2) tsunami simulation modeling output data, and (3) GIS data for the modeled community. The settings web page allows the user to select a predefined set of tsunami simulation output data to use as an input into the simulation model, as well as the initial population size, general time of day (morning, afternoon, evening, night), the time



Tsunami Inundation and Evacuation Simulation

Simulation Input Parameters

Information

This page is used for configuring tsunami evacuation input parameters for a particular tsunami inundation run. Tsunami inundation simulation data are generated from numerical modeling codes or services such as the Tsunami Computational Portal (TCP). Output simulation data are used as a set of inputs for the evacuation simulation.

Tsunami casualty model parameters were calculated in collaboration with Dr. Harry Yeh. The casualty model is based on the population distribution of a community, including gender and age factors, and determines whether a person can remain standing within a tsunami flow. The method was published in Natural Hazards Review in 2010.

Available Parameters:

Tsunami simulation input: Existing output data from a tsunami simulation run

Population size: Number of people to be simulated

Time of day: General time of day (affects population distribution)

Evacuation start time step: The time step at which to begin the evacuation simulation

Evacuation point(s): Defined evacuation points that exist in the community of interest, or arbitrary user-defined points

Alternate road network: An alternate shapefile of the road network can be provided (contact us for more information)

Input Parameters

Tsunami simulation input:	Seaside, OR - 10m *time step = 18s*
Simulation extent:	*determined by simulation input*
U and V velocity vectors:	*determined by simulation input*
Population size:	
Time of day:	Morning
Evacuation start time step:	(1-400)
Evacuation point(s):	○ Use existing evac point(s) for currently selected location ○ Define your own evac point(s) (up to 3) Lon: Lat: Lon: Lat: Lon: Lat:
Alternate road network:	*Contact us to use an alternate road network*
Bathymetry & topography:	*determined by simulation input*
Your email address:	

Figure 3.6: Settings web page for defining the input parameter settings used by the simulation model.

step at which to begin the evacuation process, and the evacuation points to use as destinations. Tsunami simulation output data consist of a wave series file representing water depth at each time step of the simulated tsunami event, as well as U and V velocity vector files that are used to determine the force and direction of the tsunami flow.

GIS data were essential to the development of the simulation framework. Data representing roads, bridges, and taxlots were obtained from Clatsop County, OR (the county in which Seaside is located). At the time of initial development, a detailed GIS layer representing buildings (structures) in Seaside could not be obtained. Instead, a structures layer was digitized from high-resolution (0.5 m) aerial imagery provided by the State of Oregon. The digitized layer was georeferenced and converted to shapefile format, then loaded into the spatial database. The Clatsop County GIS layer representing taxlots contained attributes such as structure age and type, year built, and a three-digit "stat_class" code representing a classification of building type and land use.

After loading the taxlot GIS data into the spatial database, PostGIS spatial queries were developed to identify the intersection of taxlots with the digitized structure data, spatially assigning taxlot ID, street address, and the stat_class code as attributes to each structure polygon within the database. The resulting attributed structures data were used to (1) identify starting locations for the initial population distribution, and (2) coarsely distribute the starting population among structure type (i.e., residential vs. commercial) based on the general starting time of the simulation. A similar assignment of residential vs. commercial/industrial building types based on the stat_class attribute was used by Dominey-Howes *et al*. (2010) for estimating building vulnerability and property loss from a tsunami event affecting the Seaside community.

In this study, all simulated individuals were routed along the established road network in the Seaside community. Each individual moved toward one of three official evacuation locations (whichever location is closest to the individual at the start of the simulation process), identified by DOGAMI (2005). Routing along the road network was performed using the Dijkstra shortest-path algorithm, as initially described in Dijkstra (1959). Several steps were followed to prepare the road network for use in this manner:

1. A spatial data layer representing roads in the area was obtained in Esri shapefile format.
2. The roads layer was imported into the existing PostgreSQL/PostGIS spatial database.
3. Elevated road segments (bridges) were identified and their average height above ground level was recorded in the database.
4. A representation of the roads layer as a network of edges and vertices was produced.

Following preparation of the roads network, the Dijkstra shortest-path algorithm was applied to the simulated population in order to route them along the prepared network. The assumed evacuation speed, based on a pedestrian walking rate for average adult males (1.65 m/s) and females (1.37 m/s) suggested by Eubanks (1994) and used in Yeh (2010), was used to advance each individual a certain distance along the road network at each time step. The routed location of each individual at each time step was recorded in the spatial database and later used in the modeling process.

3.4.4 Casualty Determination

To determine the status of each individual at each time step, certain assumptions must be made about generalized body types to determine whether an individual is overtaken by the water depth and velocity at their location. Yeh (2010) evaluated gender and age factors in tsunami casualties and, using mean values of anthropometric measured data from Kroemer *et al*. (1997), developed casualty curves representing the critical conditions that cause casualties during tsunami inundation flows. The casualty curves are plotted on charts of inundation depth vs. flow speed, with distinctions made among male and female body types, and adults vs. children. Table 3.2 contains the anthropometric values used in this simulation model. In this study, adult male and female body types are evaluated.

Casualty determination is calculated by a series of algorithms that take into account the anthropometric values, water depth, force, and direction of the tsunami flow.

These algorithms are implemented on the processing server in a C++ program that iterates across all individuals still standing at each time step of the simulation. The first evaluations determine (1) whether there is any water present at that time step (if false, no casualty exists), and (2) whether the water speed is equivalent to zero (if true, no casualty exists). Next, each individual's hip height is compared to the water depth at the current time step. Following Yeh (2010), values representing the wetted area of a human body (A), the centroid of the area A relative to ground level (Ay_0), and the submerged volume of the individual's body (V) are calculated for each individual. If the water depth is shallow (below hip height), A, Ay_0, and V are calculated using body parameters representing the lower half of the body; otherwise, the values are calculated using the full set of body parameters.

Table 3.2: Mean values of anthropometric measured data for U.S. adult male and females (Kroemer *et al*. 1997), used by Yeh (2010) to calculate casualty curves for individuals subjected to tsunami flow. The values in this table are used in the casualty determination algorithm built into the simulation model.

Age and gender	**Height (m)**	**Weight (kg)**	**Foot width *a* (m)**	**Hip breadth *b* (m)**	**Shoulder breadth *c* (m)**	**Hip height h_b (m)**	**Shoulder height h_c (m)**	**Foot length *d* (m)**	**Abdominal depth *e* (m)**	**Chest depth *f* (m)**	**Moment arm *x* (m)**
Adult male	175.6	78.0	0.202	0.367	0.492	0.928	1.443	0.101	0.236	0.243	0.300
Adult female	162.9	62.0	0.180	0.385	0.433	0.862	1.334	0.090	0.219	0.239	0.278

The C++ code block below handles the conditional calculations involved in casualty determination (the code that implements these algorithms is provided in more detail in Appendix E). The set of equations in the first conditional statement represents the calculations performed if water depth is below hip height.

```
if ( h < hb ) {

  A = a*h+(b-a)*((h*h)/(2.*hb));

  Ay0 = ((a/2.)*(h*h)+((b-a)/(3*hb))*powf(h,3.));

  V = h * 3.1415926 * ((a + (b - a) * (h/hb))/2.) *
   ((d + (e-d) * (h/hb))/2.);

} else {

  A = ((a + b)/2.)*hb+b*(h-hb)+((c-b)/(2.*(hc-hb))) *
   (h-hb)*(h-hb);

  Ay0 = ( (a/2.)*hb*hb+((b-a)/3.)*hb*hb+(1./(6.*(hb-hc))) *
   (3*c*h*h*hb-(2*b+c)*powf(hb,3.)+3*b*hb*hb*hc+h*h *
   (2.*(b-c)*h-3.*b*hc)));

  V = hb * 3.1415926 * ((a + (b - a) * (h/hb))/2.) *
   ((d + (e-d) * (h/hb))/2.0) + (h-hb)*3.1415926 *
   ((b+(c-b) * ((h-hb)/(hc-hb)))/2.) * ((e+(f-e) *
   ((h-hb)/(hc-hb)))/2.0);
}
```

An individual can be displaced by a tsunami flow by two different failure modes. The first is based on force balance and occurs when the force of the tsunami flow surpasses the individual's frictional force (i.e., their ability to remain standing against the flow). When the flow force exceeds the frictional force, the individual is assumed to be brought down by the tsunami flow and displaced in the flow direction (i.e., carried away). The second mode is based on moment balance and occurs when the moment of the flow exceeds the moment of the individual – that is, the rotational force exerted by the flow exceeds the individual's ability to remain standing, causing them to be knocked over into the flow. Following calculation of the values A, Ay_0, and V, the code below determines whether an individual will succumb to either force imbalance or moment imbalance, or remain standing, based on the anthropometric values and data describing water depth, velocity, and direction at their location. If they are determined to be a casualty, they are marked as such in the database at that time step, and not evaluated in subsequent time steps.

```
if (w < (sigma*V)) {
  Location currentPersonLocation = p->get_location();
  p->set_dead(true);
  return;
}

wsv = w-sigma*V;
um = sqrt((2.*wsv*x)/(Cd*Density*Ay0));
uf = sqrt((2.*ff*wsv)/(Cd*Density*A));
minu = HSDMIN((um),(uf));

if ((speed/minu) < 1.) {
  return;
} else {
  p->set_dead(true);
}
```

3.4.5 Simulation Model

The simulation model operates on a time-step basis, as defined by the tsunami inundation input data used in the model run. The state of all objects is calculated and evaluated at each time step, resulting in model outputs that are written to the filesystem (e.g. time-series grids representing tsunami inundation), and to the database (e.g., time-series data representing the position and state of each modeled individual at each time step). Figure 3.7 illustrates the modeling process and the flow of logic in the simulation model, while Figure 3.8 describes the sub-process that evaluates each person's casualty status at each time step as the simulation progresses.

The simulation model accepts user-defined parameters as inputs that determine, in part, the initial population distribution in the community of interest. Based on the taxlot information related to the structures data stored in the spatial database, structures are identified as residential or commercial (including businesses, schools, medical facilities, etc.). The randomly generated population is distributed among the structures in the community depending upon the general time of day assigned by the user (i.e., early morning, daytime, evening, or night). Table 3.3 displays the percentages of occupancy assigned by structure type, depending upon the simulated time of day – in general, more people are assumed at home at night than during the daytime (the simulation model assumes the event occurs on a weekday), based on data from the American Time Use Survey (2012). This population distribution represents the starting condition for the simulation model.

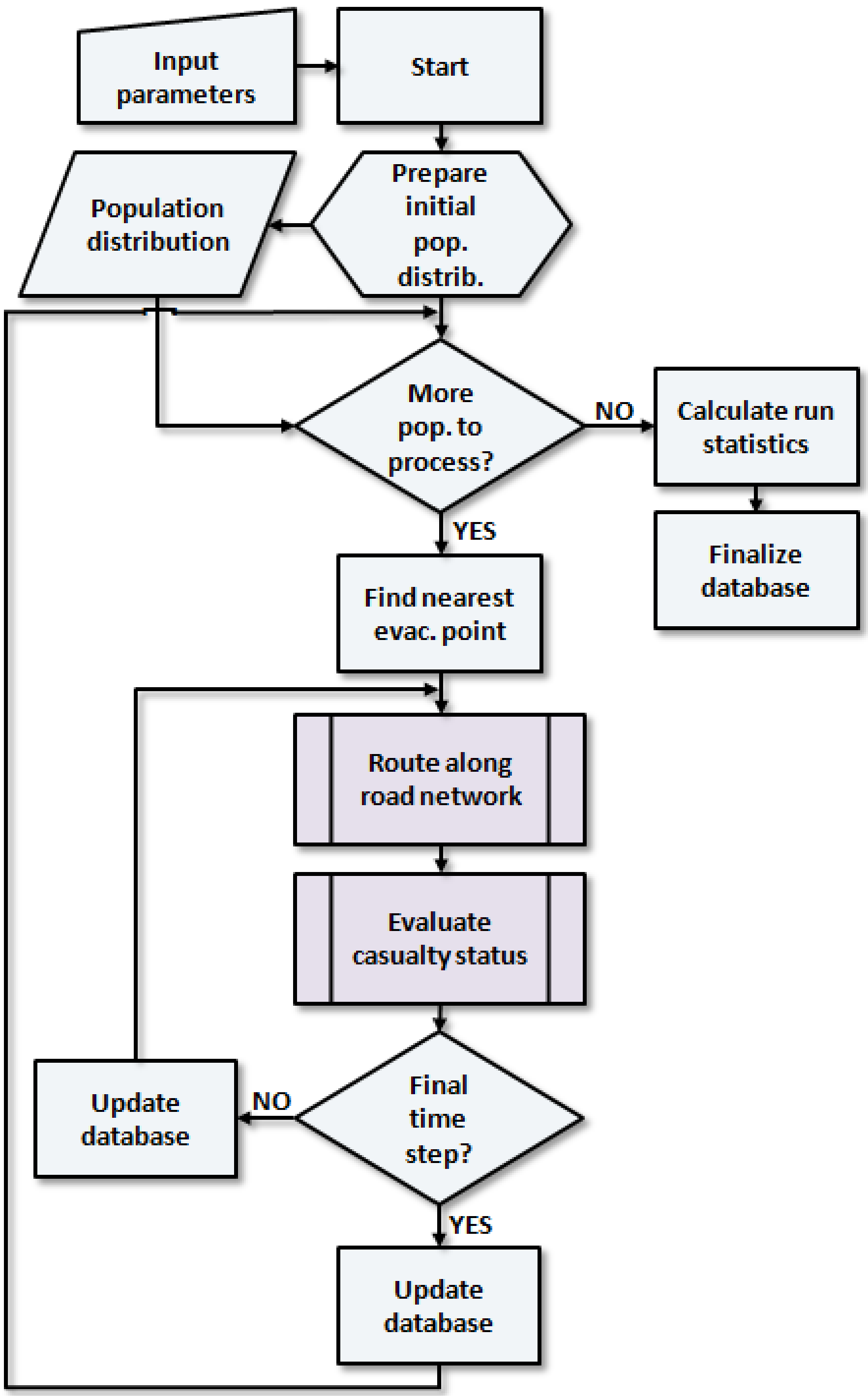


Figure 3.7: Flowchart representing logic flow in the simulation model. The "Route along road network" and "Evaluate casualty status" sub-processes are described in more detail elsewhere in the document.

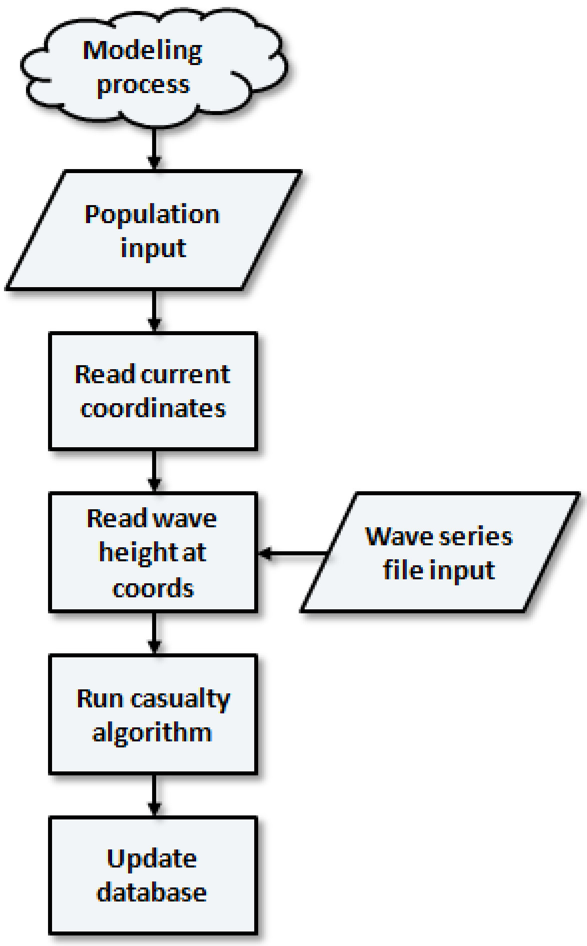


Figure 3.8: Flowchart representing the casualty evaluation sub-process. This sub-process executes the casualty determination algorithm as part of a loop process in the larger simulation model. The casualty status for each person is evaluated at each time step within the loop, and the status is written to the database as part of the time-series information stored for each person.

Table 3.3: Estimated percentages of occupancy by structure type across different periods of a typical weekday.

General location	Early Morning	Daytime	Evening	Night
Residential	70	20	60	90
Commercial	30	80	40	10

Once assigned to a structure, a person is ready to be moved onto the road network during the first time step of the simulation. By modeling the movement of the population in a synchronized fashion with the tsunami inundation data (i.e., time steps), the simulation model assumes that all members of the population receive the tsunami warning at the same time, and begin moving simultaneously.

3.4.6 Output Data Products

As indicated in Figures 3.6 and 3.7, the simulation modeling process generates multiple output data products for each model run. The output data products are summarized in Table 3.4. A Python script run early in the processing chain of events generates a randomized initial population distribution, using spatial database queries to place individuals in known structures. This output data product is produced as a file, which then serves as an input data source for subsequent steps of the modeling process. Water depth at each time step is read from a binary wave series input file by a C++ program that calls a GDAL function to generate a TIFF image representing water depth for each time step. The TIFF images are used for visualizing water depth in the results interface, and are generated once for the related simulation and then stored for later use (i.e., they are not generated dynamically every time the simulation is viewed).

Table 3.4: Output data products generated by the simulation framework.

Output Data Product	**Format**	**Description**
Population distribution	ASCII file	File containing initial randomly-generated population distribution assigned to structures in the study area; created by Python script.
Water depth	Binary files	TIFF images generated via C++ and GDAL, one per time step representing water depth across the domain, used as a semi-transparent map overlay.
Space-time path	Spatial database	Data representing each simulated individual's location in space and time, for each time step. Used for queries and generation/animation of space-time paths.
Casualties	Database	Record of each individual's casualty status at each time step as calculated by the casualty determination algorithm.

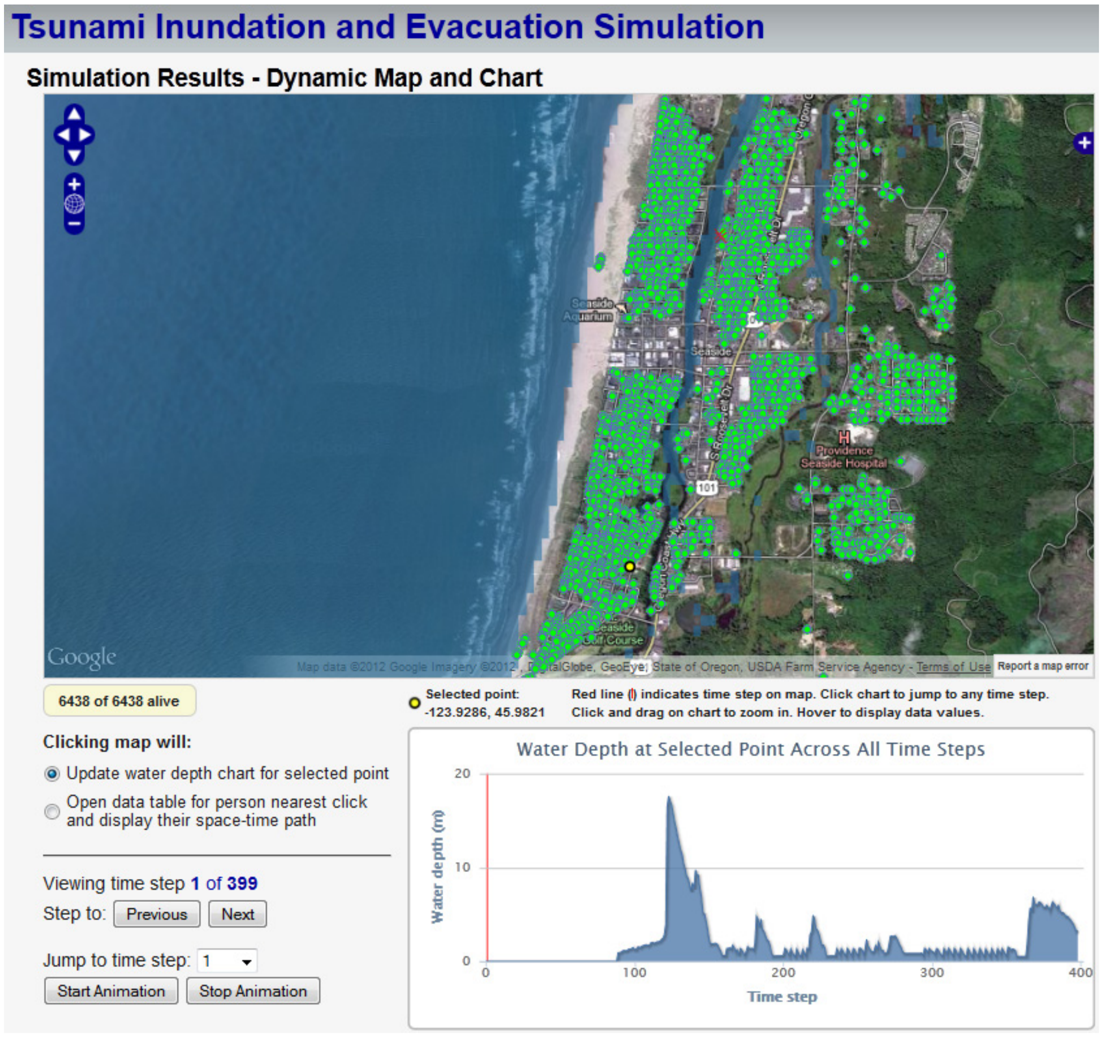
The routing algorithm advances each individual along the road network, recording their position in space and time in the spatial database. Those data are used for visualizing the progression of each individual on the animated map, as well as for satisfying queries and displaying an individual's space-time path on the map. As the simulation model advances, the casualty determination algorithm evaluates each individual's casualty status at each time step and records that status in the database.

Those data are used for visualizing each individual's status on the map and are particularly effective for identifying change over time during animation. A custom database function (Appendix F) calculates simulation run summary statistics at the end of the process.

3.4.7 Simulation Results Interface

Once a simulation model run is complete the output can be viewed in the web-based results interface (Figure 3.9). All automated output products (time series population data, individual space-time paths, modeled water depth as TIFFs) are automatically made available in the interface. The results interface contains:

- A dynamic map with multiple base layers that supports animation of the time-series result data and an individual's space-time path.
- An interactive chart displaying water depth across all time steps at any clicked point on the map.
- A dynamic table containing a selected individual's data for each time step.
- Controls for navigating among the set of time steps, starting and stopping the animation, and interpreting a user's map click.


Tsunami Inundation and Evacuation Simulation
Simulation Results - Dynamic Map and Chart
Seaside Aquarium
Seaside
Providence Seaside Hospital
Seaside Golf Course
101
Google
Map data ©2012 Google Imagery ©2012 , TerraMetrics, DigitalGlobe, GeoEye, State of Oregon, USDA Farm Service Agency - Terms of Use
Report a map error
Water Depth at Selected Point Across All Time Steps
Water depth (m)
Time step
0
10
20
100
200
300
400
Previous
Next
Start Animation
Stop Animation

6438 of 6438 alive

Selected point:
-123.9286, 45.9821

Red line (|) indicates time step on map. Click chart to jump to any time step.
Click and drag on chart to zoom in. Hover to display data values.

Clicking map will:
[✓ Update water depth chart for selected point]
Open data table for person nearest click and display their space-time path

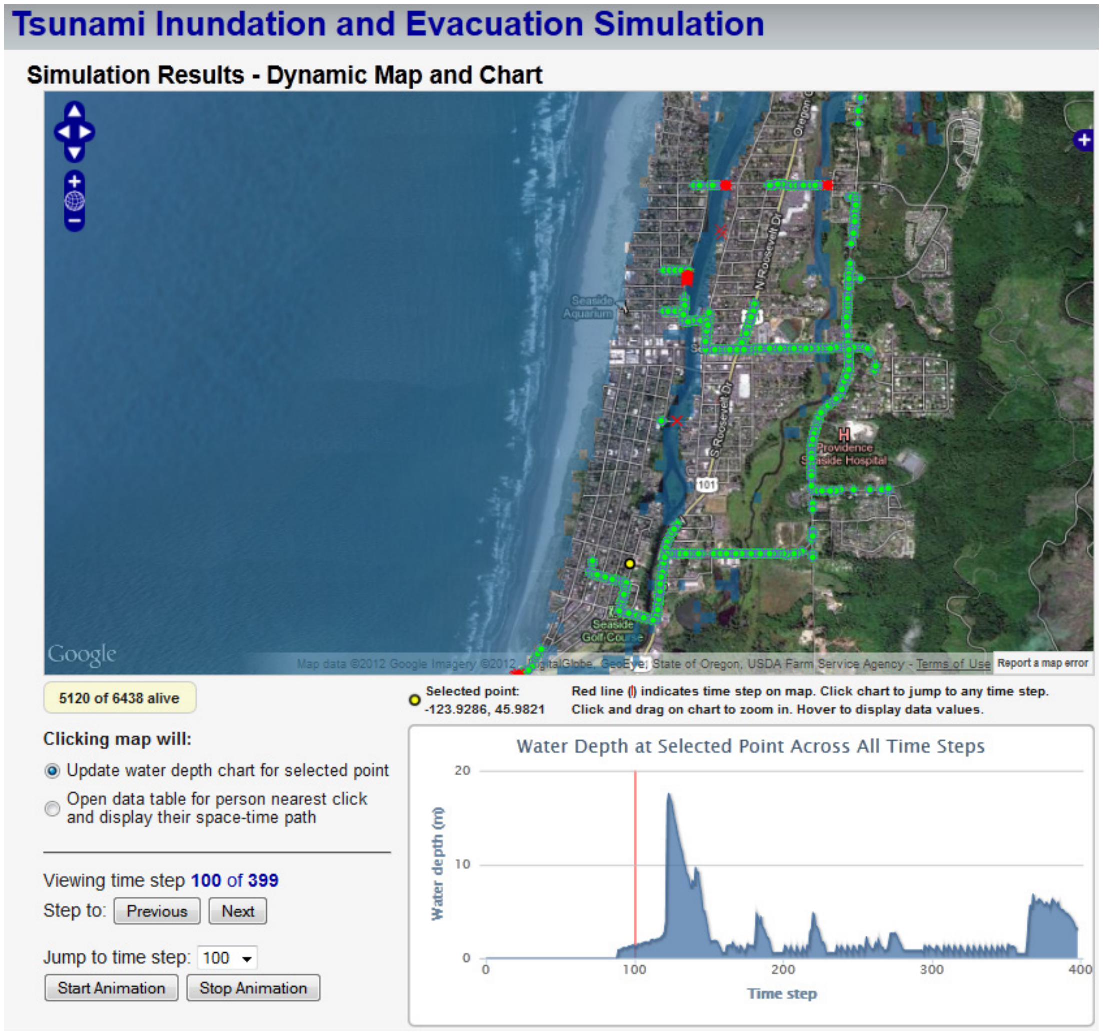
Viewing time step 1 of 399
Step to:
Jump to time step: 1

Figure 3.9: Simulation results web interface. The map and chart are rendered dynamically and are interactively linked (i.e., clicking one updates the other). This figure displays the starting condition, and the layer switcher is expanded to display the available base layer and overlay options.

3.4.7.1 *Mapping and Animation Tool*

The mapping and animation tool enables exploration and visualization of both the tsunami inundation data and the simulated human response (i.e., movement of all simulated individuals across all time steps). By clicking the "Next" or "Previous" buttons the user can step through time steps one by one to view the conditions at each time step. Alternatively, the user can jump directly to a numbered time step of interest, automatically loading the data for that time step. Clicking the "Start Animation" button causes the interface to progressively load the simulation result data for each time step in an automated fashion, enabling the user to view evacuation movement as an interactive animation. All map operations are accessible during animation – panning, zooming, layer substitution and toggling, water depth chart display update, and data table generation can all be performed during animation, as well as when the animation is stopped. At each time step, the database is queried to determine whether an individual's map symbol should be a green circle, indicating they are still evacuating, a red dot, indicating that they were overtaken by the tsunami flow at that time step, or a red X, which indicates their final routed position in subsequent time steps. Figure 3.10 shows the interface at time step 100. In this figure, the routed individuals have already navigated the city streets and are moving along main routes to the evacuation points.

The simulation results web interface was built using PHP code on the server-side, and JavaScript and HTML on the client-side. JavaScript is heavily used in the interface codebase to provide tight integration among the OpenLayers-based mapping/visualization component, the animation controls, and the dynamic chart constructed using the Highcharts package. Additional client-side code sends



Tsunami Inundation and Evacuation Simulation

Simulation Results - Dynamic Map and Chart

5120 of 6438 alive

Selected point:
-123.9286, 45.9821

Red line (|) indicates time step on map. Click chart to jump to any time step.
Click and drag on chart to zoom in. Hover to display data values.

Clicking map will:

[✓ Update water depth chart for selected point]

Open data table for person nearest click and display their space-time path

Viewing time step 100 of 399

Jump to time step: 100

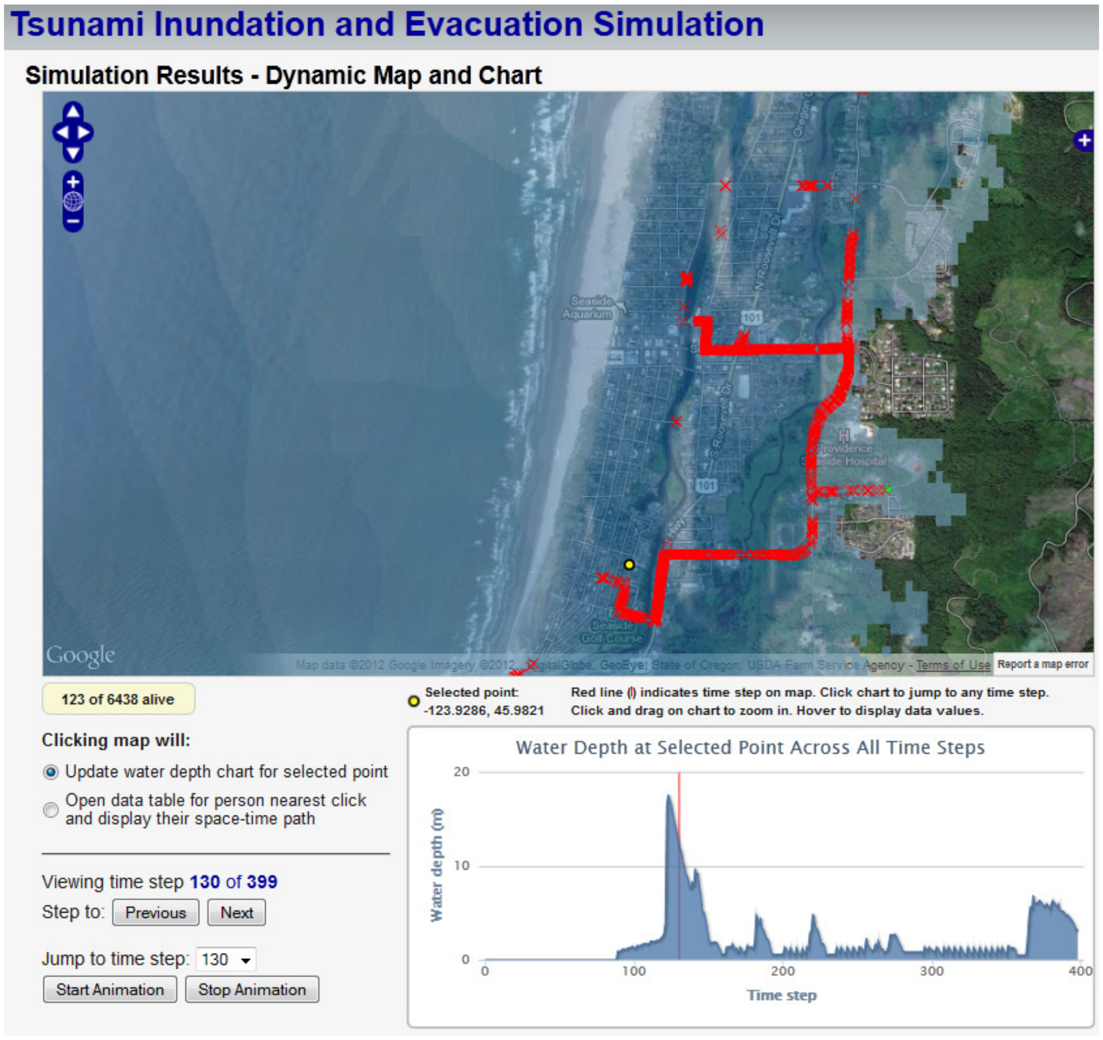
Seaside Aquarium
N Roosevelt Dr
S Roosevelt Dr
101
Providence Seaside Hospital
Seaside Golf Course
Google
Map data ©2012 Google Imagery ©2012 DigitalGlobe, GeoEye, State of Oregon, USDA Farm Service Agency - Terms of Use
Report a map error
Step to: Previous Next
Start Animation
Stop Animation
Water Depth at Selected Point Across All Time Steps
Water depth (m)
20
10
0
0
100
200
300
400
Time step

Figure 3.10: Simulation results web interface displaying time step 100. The red line on the chart moves automatically to indicate the current time step on the water depth profile. Individuals are moving along established evacuation routes.

asynchronous JavaScript and XML (AJAX) requests to the server, which enable dynamic updating and loading of additional content on the results page without requiring a page reload for each request (i.e., the map can be clicked at any time to update the chart or return other data, without reloading the map view).

The OpenLayers client-side mapping toolkit is fully JavaScript-based, enabling tight integration with other client components, both in terms of code (Appendix G) and functionality. The OpenLayers map interface developed for this study includes the four standard Google Maps base layers (streets, terrain [aka "physical"], satellite, hybrid), as well as the Esri Ocean Basemap layer, which is implemented as an ArcGIS REST API layer via OpenLayers. All base layers can be included natively in the OpenLayers code. To include raster and vector layers stored locally (on the website domain serving the map), a server-side mapping engine must be used. The open source package MapServer was used to render wave runup (raster), population (vector), water depth points (vector), and space-time paths (vector) for this study. Each time a map view is generated, OpenLayers sends a request to MapServer for the layer(s) making up the overlay view, and MapServer returns an image that OpenLayers composites with the rendered client-side map to produce the output the user sees in the map window. Through Proj.4, OpenLayers can reproject layers on the fly (e.g., to Google's "Spherical Mercator" projection) for integration with the map view.

The tsunami simulation data used in this study modeled an extreme tsunami inundation event, with a maximum water depth of nearly 20 m in some areas (evident in the chart in Figure 3.10, for example, around time step 120). Figure 3.11 displays the conditions at time step 130, after the first and largest inundation event has arrived on shore.


Tsunami Inundation and Evacuation Simulation

Simulation Results - Dynamic Map and Chart

123 of 6438 alive

Selected point:
-123.9286, 45.9821

Red line (|) indicates time step on map. Click chart to jump to any time step.
Click and drag on chart to zoom in. Hover to display data values.

Clicking map will:

[✓ Update water depth chart for selected point]

Open data table for person nearest click and display their space-time path

Viewing time step 130 of 399

Jump to time step: 130

Google
Map data ©2012 Google Imagery ©2012, DigitalGlobe, GeoEye, State of Oregon, USDA Farm Service Agency - Terms of Use
Report a map error
Seaside Aquarium
Providence Seaside Hospital
Seaside Golf Course
101
Step to: Previous Next
Start Animation
Stop Animation
Water Depth at Selected Point Across All Time Steps
Water depth (m)
20
10
0
0
100
200
300
400
Time step

Figure 3.11: Simulation results web interface displaying time step 130. In this extreme simulation scenario, nearly all individuals are marked as casualties by this time step.

3.4.7.2 *Interactive Chart*

When the user selects the “Update water depth chart for selected point” radio button (the default selection) and clicks a desired point on the map, a yellow marker is placed on the map (as in Figure 3.11), the selected pixel coordinates are transformed to latitude and longitude, and an AJAX call is made to a C++ utility program that reads the binary tsunami wave series file and retrieves the water depths across all time steps at the requested point. The data are returned from the server in JavaScript Object Notation (JSON) format, delivered to the client, and processed by the chart code (Appendix G) to dynamically create the filled line chart without reloading the web page. The dynamic chart has several features (some are displayed in Figure 3.12):

- Hovering the mouse cursor over the chart area displays the water depth value at the time step indicated by the cursor.
- Clicking and dragging the mouse cursor across the chart area causes the chart to zoom in to the selected region to display finer detail. The chart also automatically adjusts the X and Y axes to the new data range if necessary.
- The chart area is active and is linked to the map display – clicking on the chart area causes the map to jump to the selected time step and load the related map data, and also update the time step and casualty count.
- A red, vertical tracking line on the chart area indicates the currently selected time step and moves dynamically when the animation mode is running, making it easy to track the current position and correlate the chart with the map view.

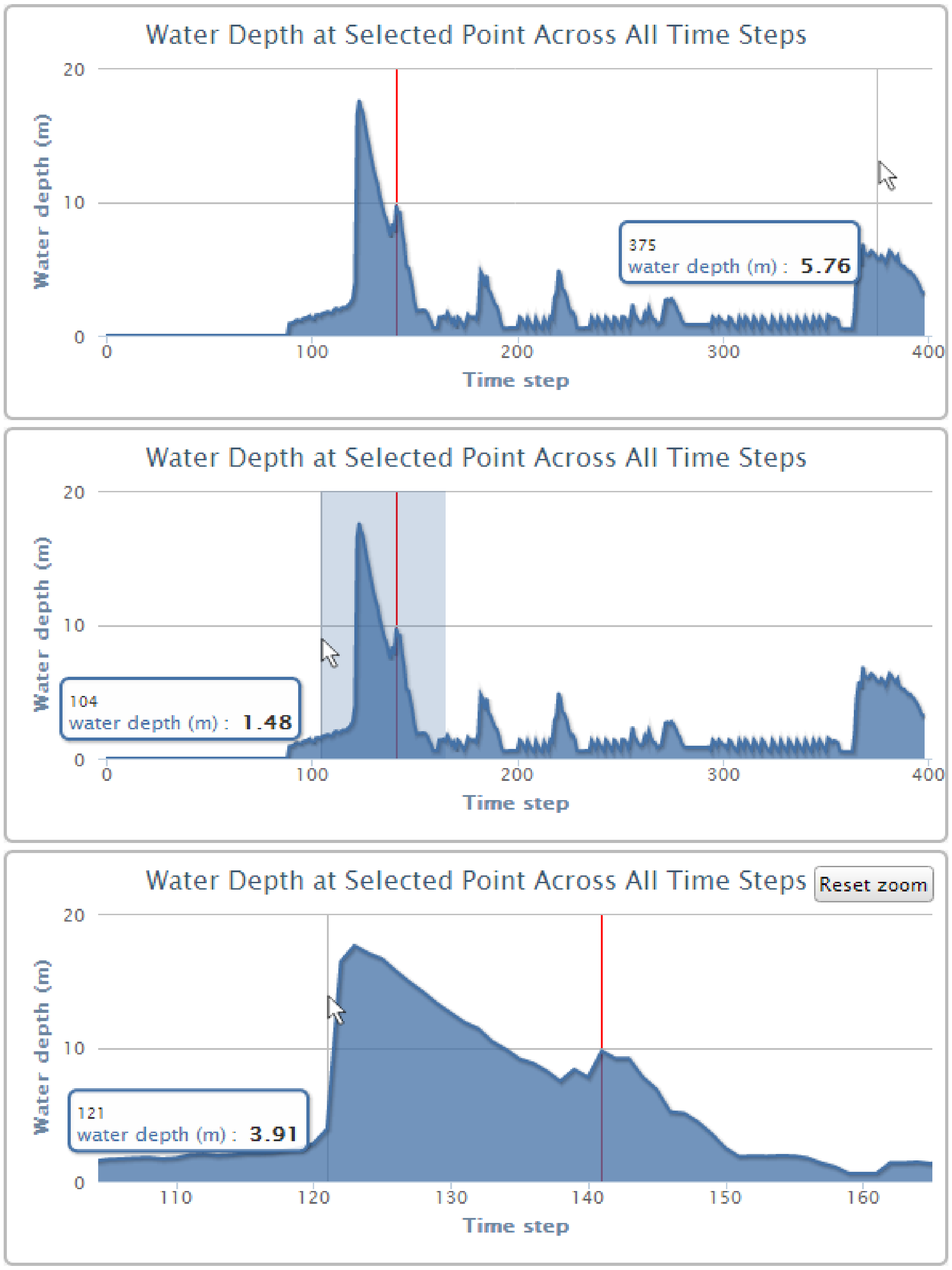


Figure 3.12: Examples of the interactive water depth chart. Each time step in this example represents an 18 second interval. Top: the chart has been clicked to set the map animation to time step 141 (red line), while the cursor hovers over time step 375 to display water depth. Middle: an area is highlighted by dragging the mouse cursor. Bottom: the chart is zoomed in to the highlighted area.

3.4.7.3 *Space-Time Path Construction and Rendering*

An individual's path representing their movement in space and time during the duration of the simulation can be displayed simply by clicking on their map marker. Figure 3.13 displays full and zoomed-in views of a space-time path displayed on the results interface map. Because the space-time path is rendered dynamically as a vector layer based on the current time step and view extent, it is re-rendered correctly as the user performs any map operations such as zooming and panning, and it also becomes animated if the user chooses to start animating the map view. During animation, the individual's current position on the space-time path is highlighted as a yellow segment. Moreover, data retrieval and space-time path rendering can occur at any time (e.g., if an animation is already underway the user can select the "Open data table…" radio button, click near an individual on the map, and the space-time path will automatically be rendered and animated). Clicking near a different individual will update the map with the new space-time path while the animation continues.

When a user selects the "Open data table…" radio button and clicks near an individual on the map, a custom PHP function (getPathPoints(); see the "db.php" section in Appendix G) performs a spatial query via an AJAX call to retrieve all points representing the individual's space-time path. The returned point data are processed into the GeoJSON format by the `ogr2ogr` utility and returned to the client, where custom JavaScript code and OpenLayers are used to create and render a colorized track on the map representing the individual's space-time path.

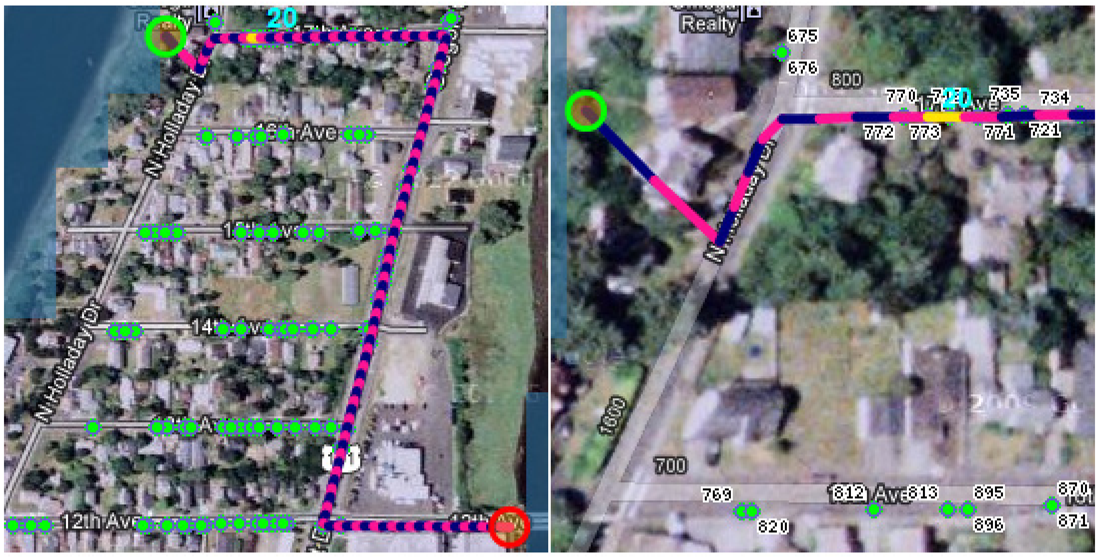


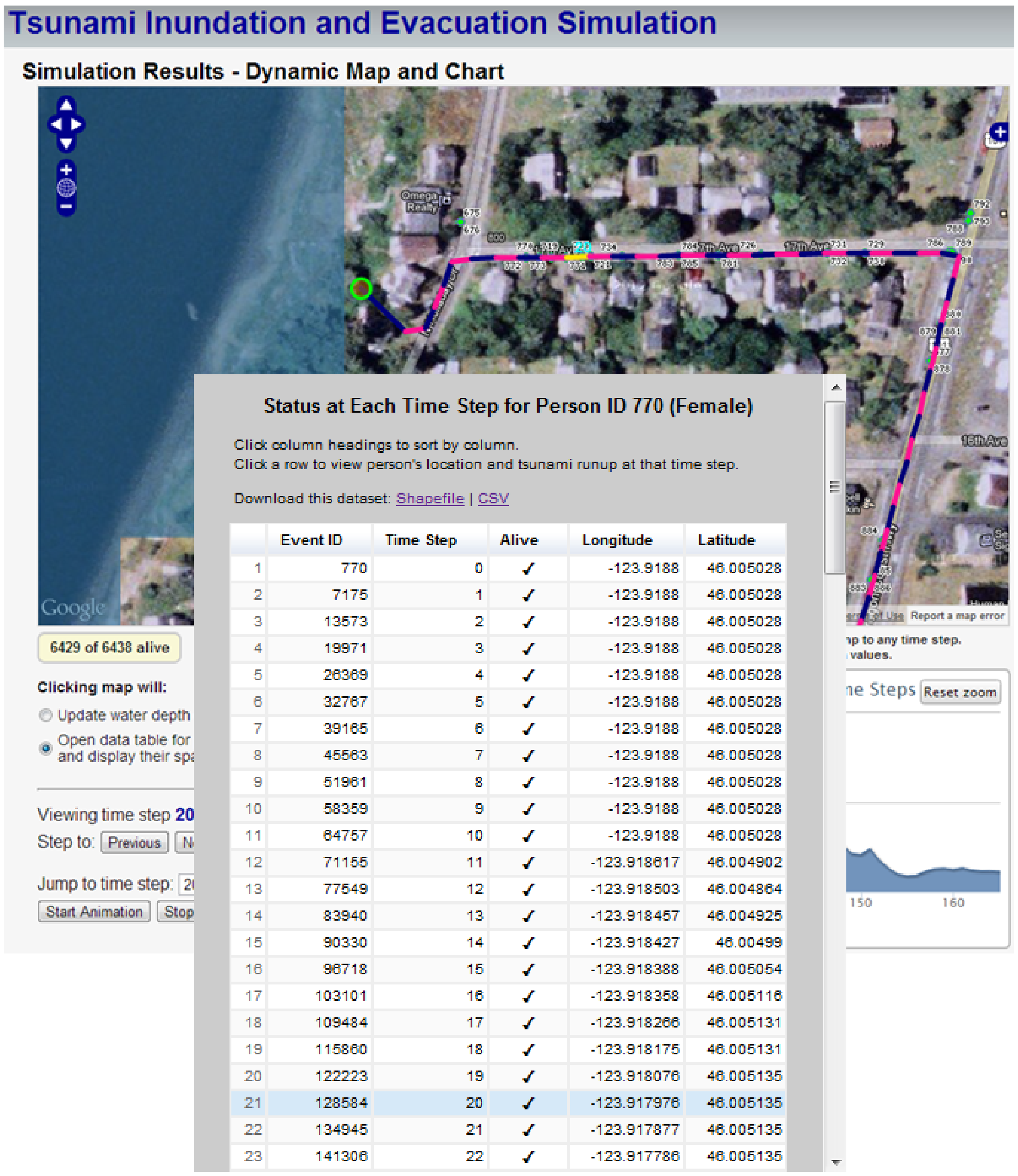
Figure 3.13: Clipped map views representing an individual's fully rendered space-time path (left) and a zoomed-in view of the path origin (right). The currently selected time step is visible as the segment highlighted in yellow.

This is accomplished by using the built-in OpenLayers.Layer.PointTrack vector layer type. Although PostGIS can output data directly to the GeoJSON format via the ST_AsGeoJSON() function, only the geometries can be exported (i.e., no attributes are included). By retrieving an individual's space-time path data directly from the database, both the geometries and attributes can be encoded as GeoJSON via ogr2ogr and passed to OpenLayers, allowing feature labeling as well as vector layer symbolization on the map.

3.4.7.4 *Interactive Data Table*

As the space-time path is rendered on the map, an interactive data table is also constructed and displayed in a popup browser window (Figure 3.14). The columns in the data table can be sorted – clicking a column heading will sort the table on that column, toggling between ascending or descending order. The data table is interactively linked to the results interface window, so that clicking a table row highlights it and causes the results interface to jump to the selected time step indicated in the table, simultaneously updating the map, counters, and red indicator line on the chart. The matching highlighted yellow segment of the individual's space-time path is also updated, giving the user an effective way to view the selected individual's evacuation progress and casualty status at any time step, in the context of the incoming tsunami inundation. Links in the table window provide downloads of the data contained in the table in shapefile or text (comma-separated value) formats, both of which are built dynamically upon the user's request.

The data table (Figure 3.14) was implemented using the DataTable class from the Google Visualization API (2012). Upon detection of a map click, a custom PHP function is called (getTimeSeries(); see the "table.php" section in Appendix G) that (1) performs a spatial query in the database to retrieve that individual's time series data, (2) processes the result, and (3) delivers the formatted data to the google.visualization.DataTable JavaScript object, which in turn renders the data as an interactive table.



Tsunami Inundation and Evacuation Simulation

Simulation Results - Dynamic Map and Chart

Status at Each Time Step for Person ID 770 (Female)

Click column headings to sort by column.
Click a row to view person's location and tsunami runup at that time step.

Download this dataset: Shapefile | CSV

	Event ID	Time Step	Alive	Longitude	Latitude
1	770	0	✓	-123.9188	46.005028
2	7175	1	✓	-123.9188	46.005028
3	13573	2	✓	-123.9188	46.005028
4	19971	3	✓	-123.9188	46.005028
5	26369	4	✓	-123.9188	46.005028
6	32767	5	✓	-123.9188	46.005028
7	39165	6	✓	-123.9188	46.005028
8	45563	7	✓	-123.9188	46.005028
9	51961	8	✓	-123.9188	46.005028
10	58359	9	✓	-123.9188	46.005028
11	64757	10	✓	-123.9188	46.005028
12	71155	11	✓	-123.918617	46.004902
13	77549	12	✓	-123.918503	46.004864
14	83940	13	✓	-123.918457	46.004925
15	90330	14	✓	-123.918427	46.00499
16	96718	15	✓	-123.918388	46.005054
17	103101	16	✓	-123.918358	46.005116
18	109484	17	✓	-123.918266	46.005131
19	115860	18	✓	-123.918175	46.005131
20	122223	19	✓	-123.918076	46.005135
21	128584	20	✓	-123.917976	46.005135
22	134945	21	✓	-123.917877	46.005135
23	141306	22	✓	-123.917786	46.005135

Figure 3.14: Data table with matching map view. A map click on an individual of interest highlights that individual's space-time path for the simulation event, and displays an interactive table of the individual's status at each time step. In this example the evacuation began at time step 10 of the simulation, so the tenth routed segment (20th time step overall) is highlighted on the space-time path.

3.5 Discussion

Casualty simulation is an important method for estimating risk to coastal communities in the event of a tsunami. Simulation models can help make the concept of tsunami inundation real and intelligible to communities that could be at risk. Watching a representation of a tsunami flow inundate land on an interactive map is interesting, but watching how the flow could affect individuals in a community, as well as animate and query their simulated movement over time and space, can provide a deeper understanding of its potential effect.

The simulation framework developed in this study is a prototype system designed to test the automation of simulation model runs and output generation based on user input, and to display the results in an interactive, usable, web-based context. Although significant effort went into examining theoretical aspects of spatiotemporal data representation and analysis, casualty determination approaches, and other important issues, the main focus of the study was on code and framework development in support of the stated study goals. There are many aspects of the prototype system that could be enhanced and improved with additional work – those aspects are discussed in this section, along with alternative approaches.

3.5.1 Casualty Determination and Evacuation Movement

One of the unique strengths of this simulation framework is the incorporation of the casualty model that evaluates various criteria to determine whether a person can remain standing in a tsunami flow at each time step of the simulation. The simulation framework developed by Katada *et al*. (2006) was sophisticated; however, once an area was inundated by a tsunami, all people within the area were automatically assumed dead (i.e., no casualty model was implemented in that framework).

Although this prototype system does include a casualty model, it is currently limited to simulating adult male and female body types. Extending the system to simulate the additional body types defined by Kroemer *et al*. (1997) would likely result in a more realistic assessment of evacuation movement and casualty determination. The similarity in the fixed set of adult male and female body type parameters is evident in the visualization output, where individuals can often be observed traveling an identical route while landing at the same point at each simulated time step. Modeling additional body types based on age and gender, and introducing some degree of variation within the anthropometric parameters representing them, should result in a more realistic simulation model and visualization output. To further extend this system into a true agent-based model, the simulation model could be redesigned to assess the position and group density of nearby individuals at each time step, as well as nearby topographic features (e.g., an individual could cross an empty lot as a shortcut rather than remaining on the road network) and other environmental variables.

In its current implementation, the simulation framework advances simulated humans a measured amount along their routed tracks at each time step, based on the estimated walking speeds of male and female adults. The temporal resolution of the time steps is defined as an input parameter during the tsunami modeling parameterization process that generates the tsunami simulation and is often defined on the order of seconds, but may also be as long as minutes in scale. The simulation framework could be configured to represent human movement on a finer temporal scale by generating position data more frequently than the fixed temporal scale of the tsunami simulation. However, while position data could be updated more frequently, casualty determination data (i.e., indicating whether the subject has become inundated by the tsunami flow) could only be updated on the fixed temporal scale of the tsunami simulation.

Because this study focused on prototype framework design, one of the assumptions made is that all individuals in the modeled population begin moving at the same time (i.e., they all receive a tsunami warning and act upon it simultaneously). Future enhancements could include the ability to vary warning transmission type (radio/TV announcements, siren, police notification, etc.) and timing, as in Katada *et al*. (2006), or perhaps take into account warning transmission using a neighbor-to-neighbor agent based modeling approach as in Nagarajan *et al*. (2012). An additional consideration is the subtle decision factors that are more difficult to quantify. For instance, although the average male may be able to run faster than the average female (cf. Kroemer *et al*. 1997), if the two individuals have a relationship of some kind, *would* the male run faster? The male evacuee may decide to run at the same speed as the female evacuee. Factors such as these could be explored further and perhaps represented in some fashion in a future version of the simulation framework.

3.5.2 Data Sources and Issues

As with most projects that rely upon multiple sources of spatial data as inputs for analysis and interpretation, a certain amount of error is introduced by the data sources. In this study, error could be present in the spatial data representing roads and structures for the Seaside area. Examples of error could include misattribution of structure type, missing structure data, or problems with the road network topology. In a prototype system, however, in most cases these issues are secondary to any larger issues that would cause the data to not function properly when used within the simulation framework.

This simulation framework (indeed, any simulation system requiring such data) would also benefit from higher-resolution tsunami runup data, which would allow more precise estimation of casualties. Low-resolution tsunami inundation data represent a source of potential error in a simulation framework. Since modeled water depth is the same for all individuals within a given grid cell, any individuals matching the casualty model conditions who are spatially located within that grid cell will be evaluated against that water depth, even if their locations are several meters apart at that time step. Advances in high-resolution bathymetry data mapping (e.g., via LiDAR technology), together with better techniques for modeling wave propagation in near-shore environments where bathymetry and topography data merge, will lead to more realistic simulations of wave runup and calculations of water depth per grid cell.

The spatial database was populated with additional structure-related data that were not used in the prototype system. The simulation framework could be extended to include estimation of losses due to structural damage, similar to Dominey-Howes *et al*. (2010), who conducted building damage estimation for the town of Seaside using structural data in the event of a tsunami.

Although data representing temporal phenomena are becoming increasingly important and available (e.g., climate data, sensor network data), representation of spatiotemporal data in spatial databases and generalized GIS tools for working with spatiotemporal data remain limited. Esri has made progress with enabling support for temporal data in their software products – ArcGIS 10 introduced new support and tools for working with temporal data, capabilities which are enabled in both ArcGIS Desktop and in ArcGIS Server applications (ArcGIS Help 2012). Esri's approach is to store time stamps at the feature level rather than storing separate layers to represent snapshots in time. Some relational database software products such as TimeDB (2012) are built specifically to handle temporal data in an object-relational fashion, and typically only store changes to each database object over time to minimize database size and complexity. In the prototype system developed for this project, the database objects (i.e., simulated evacuating individuals) are moving at every time step for which they have not been overtaken by the tsunami flow, so information about their position and state is stored for every time step prior to reaching a casualty state.

3.5.3 Hardware and Software Technology

Fast performance is a challenge in any web-based system. In this prototype system, the spatial database, which drives many of the processes related to the web interface, is stored on a dedicated server with plenty of processing power and memory, fast SCSI disks and internal and external network connections. The database server parameters are optimized for read performance, and the database tables are properly indexed to optimize both spatial and non-spatial query performance. The Apache web server and PHP installation are also optimized for fast generation and transmission of web pages. While all of this is necessary for fast performance in general, it is especially important for the interactive web-based display in this prototype system, where the animation performance depends upon fast transmission of data from the server to the client.

Continued advances in web-based technology for spatial visualization (e.g., Anderson *et al*. 2011) will make more techniques available for the development of complex interactive web-based systems. For instance, web mapping applications that dynamically render spatial features (i.e., as vector data features rather than as a generated map image) may benefit from compression of vector data on transmission from server to client (e.g., Bashar *et al*. 2012). However, although this approach may lessen server-side load under heavy usage, it can impose a performance penalty on the client-side if many features are requested in the map view. Another example is the potential of vector-based rendering via HTML5 or scalable vector graphics (SVG) – each has some important differences, but also represents an important new method for creating map-based graphics in a web browser.

3.5.4 User Interactivity

A goal of this study was to provide an interactive web-based interface for displaying results that included animation capability in an easy-to-use form, while conveying important information about the content. In considering the general user experience, some studies indicate that animations are no better than static diagrams at helping people understand concepts portrayed in the content (Hegarty *et al*. 2003, Mayer *et al*. 2005, Kim *et al*. 2007). However, animations that incorporate interactive controls provide additional benefits over fixed animations and static graphics (Tversky *et al*. 2002, Betrancourt 2005). Map-based animations that include tools for starting, stopping, zooming, panning, etc., give users control over the functioning of the animation, which allows them to discover data and relationships at their own pace. Furthermore, implementing the animated mapping interface in a web browser increases accessibility and, potentially, ease-of-use. The next phase of prototype development will include usability testing conducted by selected users in a controlled environment.

3.6 Conclusion

The intent of this study was to develop a prototype framework for simulating potential human movement in response to a hypothetical tsunami inundation event. Spatiotemporal tsunami inundation data and other spatial data describing the Seaside, Oregon region were used as inputs to the simulation framework. One of the primary goals was to automate the production of simulation output, based on user input, and also automate the inclusion of the output in a web-based data exploration and visualization interface. These goals were accomplished by developing custom software tools on both the server- and client-side, and implementing those tools in a framework system that integrated them with a centralized spatiotemporal database as well as key open source software tools.

As part of the simulation framework, modeled individuals were (1) routed along an existing road network and (2) evaluated at each time step of the simulation using a casualty determination algorithm. Although certain assumptions had to be made (i.e., using a standard set of body parameters for males and females), the inclusion of the casualty model, itself based upon research evaluating human stability in tsunami inundation conditions, gives a firm basis for determination of casualties given water depth and velocity conditions at any time step.

An additional research challenge addressed in this prototype system is the animation of time-series simulation output data within a web-based mapping and visualization interface. The goal was to give the user full control over the interface (including querying the underlying time-series data) at any time, even while the animation is underway. The additional features (dynamic interactive chart, animated space-time path, interactive data table) can provide the user with extra insight into the time-series data, the tsunami inundation conditions, and the casualty state at each time step. This goal was successfully accomplished using a mix of custom code with open source software packages such as OpenLayers, MapServer, Highcharts, and GDAL/OGR.

The prototype system developed in this study demonstrates an effective simulation framework for the automated modeling and visualization of spatiotemporal data. The framework can be expanded upon to provide additional functionality, or used as a foundation for developing a similar system in another discipline that requires the simulation, analysis, and display of spatiotemporal data.

3.7 Acknowledgments

Initial development of the simulation framework was supported by the National Science Foundation under Grant No. SES-0527520. Additional funding was provided by an NWACC 2008 Grant. Toshitaka Katada provided helpful advice early in the study. Eric Geist provided high-resolution runup simulation data for the Seaside, Oregon area that were used in prototype development. Clatsop County personnel provided some of the GIS data used in this study.

3.8 References

Adobe. 2012. Adobe Flex framework, http://www.adobe.com/products/flex.html (last accessed 17 May 2012).

American Time Use Survey. 2012. Bureau of Labor Statistics – American Time Use Survey, http://www.bls.gov/tus (last accessed 2 August 2012).

Anderson, J.C., M. Davis, K. Fujiwara, T. Fang, J. Andres, and M. Nedbal. 2011. Web-based scientific visualization software for geospatial displays and collaborative applications. In Proceedings of the OCEANS '11 Conference. Kona, Hawaii.

Andrienko, G., N. Andrienko, and S. Wrobel. 2007. Visual analytics tools for analysis of movement data. SIGKDD Explorations 9(2): 38-46.

Andrienko, G., N. Andrienko, P. Bak, D. Keim, S. Kisilevich, and S. Wrobel. 2011a. A conceptual framework and taxonomy of techniques for analyzing movement. *Journal of Visual Languages and Computing* 22(3): 213-232.

Andrienko, G., N. Andrienko, and M. Heurich. 2011b. An event-based conceptual model for context-aware movement analysis. *International Journal of Geographical Information Science* 25(9): 1347-1370.

ArcGIS Help. 2012. What's new for temporal data in ArcGIS 10, http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#//00qp00000018000000.htm (last accessed 20 April 2012).

Bashar, M.A., M. Islam, M.A. Chowdhury, M.P. Sajjad, and M.T. Ahmed. 2012. Concept for a web map implementation with faster query response. *Journal of Information Engineering and Applications* 2(1): 30-37.

Bernard, E.N., H.O. Mofjeld, V. Titov, C.E. Synolakis, and F.I. González. 2006. Tsunami: Scientific frontiers, mitigation, forecasting and policy implications. *Philosophical Transactions of the Royal Society* 364: 1989-2007.

Betrancourt, M. 2005. The animation and interactivity principles in multimedia learning. Pages 287-296 *in* R.E. Mayer (ed.), Cambridge handbook of multimedia learning. Cambridge University Press, New York.

Calheiros, R.N., R. Ranjan, A. Beloglazov, C.A.F. DeRose, and R. Buyya. 2010. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience* 41(1): 23-50.

Chen, X. and F.B. Zhan. 2008. Agent-based modelling and simulation of urban evacuation: Relative effectiveness of simultaneous and staged evacuation strategies. *Journal of the Operational Research Society* 59(1): 25-33.

Clerveaux, V., T. Katada, and K. Hosoi. 2008. Tsunami scenario simulator: A tool for ensuring effective disaster management and coastal evacuation in a multilanguage society. *Science of Tsunami Hazards* 27(3): 48-71.

Cova, T.J. and J.P. Johnson. 2003. A network flow model for lane-based evacuation routing. *Transportation Research Part A* 37(7): 579-604.

Dall'Osso, F., M. Gonella, G. Gabbianelli, G. Withycombe, and D. Dominey-Howes. 2009. A revised (PTVA) model for assessing the vulnerability of buildings to tsunami damage. *Natural Hazards and Earth System Sciences* 9: 1557-1565.

Dall'Osso, F., A. Maramai, L. Graziani, B. Brizuela, A. Cavalletti, M. Gonella, and S. Tinti. 2010. Applying and validating the PTVA-3 Model at the Aeolian Islands, Italy: Assessment of the vulnerability of buildings to tsunamis. *Natural Hazards and Earth System Sciences* 10: 1547-1562.

Dalquen, D.A., M. Anisimova, G.H. Gonnet, and C. Dessimoz. 2012. ALF – A simulation framework for genome evolution. *Molecular Biology and Evolution* 29(4): 1115-1123.

Dijkstra, E.W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik* 1(1): 269-271.

DOGAMI. 2005. Tsunami evacuation map: Seaside. Oregon Department of Geology and Mineral Industries publication DOGAMI-TS-EB-SEA-01 (11/05), http://www.oregongeology.org/pubs/tsubrochures/SeasideEvac.pdf (last accessed 2 May 2012).

Dominey-Howes, D., P. Dunbar, J. Varner, and M. Papathoma-Köhle. 2010. Estimating probable maximum loss from a Cascadia tsunami. *Natural Hazards* 53: 43-61.

Dunbar, P., H. McCullough, and G. Mungov. 2011. 2011 Tohoku earthquake and tsunami data available from the National Oceanic and Atmospheric Administration/National Geophysical Data Center. *Geomatics, Natural Hazards and Risk* 2(4): 305-323.

Eckert, S., R. Jelinek, G. Zeug, and E. Krausmann. 2012. Remote sensing-based assessment of tsunami vulnerability and risk in Alexandria, Egypt. *Applied Geography* 32(2): 714-723.

Eubanks, J. 1994. Pedestrian accident reconstruction. Lawyers and Judges, Tucson, AZ.

Geist, E.L. and T. Parsons. 2006. Probabilistic analysis of tsunami hazards. *Natural Hazards* 37: 277-314.

Geist, E.L., V.V. Titov, and C.E. Synolakis. 2006. Tsunami: Wave of change. *Scientific American* 294(1): 56-63.

Gisler, G.R. 2008. Tsunami simulations. *Annual Review of Fluid Mechanics* 40: 71-90.

Google Visualization API. 2012. Google Visualization API Reference, https://developers.google.com/chart/interactive/docs/reference (last accessed 21 July 2012).

Goto, Y., Y. Ogawa, and T. Komura. 2010. Tsunami disaster reduction education using town watching and moving tsunami evacuation animation – trial in Banda Aceh. *Journal of Earthquake and Tsunami* 4(2): 115-126.

Goto, Y., M. Affan, Agussabti, Y. Nurdin, D.K. Yuliana, and Ardiansyah. 2012. Tsunami evacuation simulation for disaster education and city planning. *Journal of Disaster Research* 7(1): 92-101.

Gupta, A.K. 2011. Recent Tohoku tsunami and earthquake: A natural consequence of the movement of the Pacific plate along the trench, east of the Japanese island arc. *Science and Culture* 77(5-6): 175-185.

Hägerstrand, T. 1970. What about people in regional science? *Papers of the Regional Science Association* 24: 7-21.

Hegarty, M., S. Kriz, and C. Cate. 2003. The roles of mental animations and external animations in understanding mechanical systems. *Cognition and Instruction* 21(4): 209-249.

Hornsby, K. and M.J. Egenhofer. 2002. Modeling moving objects over multiple granularities. *Annals of Mathematics and Artificial Intelligence* 36(1-2): 177-194.

Imamura, F., A.C. Yalciner, and G. Ozyurt. 2006. Tsunami modeling manual (TUNAMI model), http://www.tsunami.civil.tohoku.ac.jp/hokusai3/J/projects/manual-ver-3.1.pdf (last accessed 10 February 2012).

Jonkman, S.N., J.K. Vrijling, and A.C.W.M. Vrouwenvelder. 2008. Methods for the estimation of loss of life due to floods: A literature review and a proposal for a new method. *Natural Hazards* 46: 353-389.

Kapler, T. and W. Wright. 2005. GeoTime information visualization. *Information Visualization* 4(2): 136-146.

Karon, J. and H. Yeh. 2011. Comprehensive tsunami simulator for Cannon Beach, Oregon. Final Report submitted to the City of Cannon Beach, May 2011, http://www.tappister.com/assets/content/2011/05/Cannon-Beach-Tsunami-Final-Report-May-2011.pdf (last accessed 22 April 2012).

Katada, T., J. Asada, N. Kuwasawa, and Y. Oikawa. 2000. Development of practical scenario simulator for dissemination of disaster information. *Journal of Civil Engineering Information Processing System* 9: 129-136.

Katada, T. 2003. A preliminary integrated tsunami scenario simulation. Workshop for Integrated Tsunami Scenario Simulation, Oregon State University, http://tsunami.orst.edu/workshop/2003/pdf/Katada.pdf (last accessed 20 April 2012).

Katada, T., N. Kuwasawa, and H. Yeh. 2004. Tsunami scenario simulator. 2nd Workshop for Integrated Tsunami Scenario Simulation, Oregon State University, http://tsunami.orst.edu/workshop/2004/doc/Katada.pdf (last accessed 20 April 2012).

Katada, T., N. Kuwasawa, H. Yeh, and C. Pancake. 2006. Integrated simulation of tsunami hazards. EERI's 8th U.S. National Conference on Earthquake Engineering (8NCEE), San Francisco, CA. Paper No. 1727.

Keon, D., C. Pancake, H. Yeh, and T. Logan. 2009. The Tsunami Computational Portal: Distributed infrastructure for executing and comparing multiple computational models. Association of American Geographers (AAG) Annual Meeting, Cyberinfrastructure Specialty Group, Las Vegas, NV, 22-27 March.

Kim, S., M. Yoon, S.-M. Whang, B. Tversky, and J.B. Morrison. 2007. The effect of animation on comprehension and interest. *Journal of Computer Assisted Learning* 23(3): 260-270.

Kobayashi, K. T. Katada, N. Kuwasara, A. Narita, M. Hirano, and I. Kase. 2007. Collaborative interface for disaster simulation. *In* Proceedings of the 2nd International Conference on Urban Disaster Reduction, Taipei, Taiwan.

Koshimura, S. and M. Takashima. 2005. Remote sensing, GIS, and modeling technologies enhance the synergic capability to comprehend the impact of great tsunami disaster. *In* Proceedings of the 3rd International Workshop on Remote Sensing for Post Disaster Response, Chiba, Japan.

Koshimura, S., T. Katada, H.O. Mofjeld, and Y. Kawata. 2006. A method for estimating casualties due to the tsunami inundation flow. *Natural Hazards* 39(2): 265-274.

Koshimura, S., Y. Namegaya, and H. Yanagisawa. 2009a. Tsunami fragility – A new measure to identify tsunami damage. *Journal of Disaster Research* 4(6): 479-488.

Koshimura, S., T. Oie, H. Yanagisawa, and F. Imamura. 2009b. Developing fragility functions for tsunami damage estimation using numerical model and post-tsunami data from Banda Aceh, Indonesia. *Coastal Engineering Journal* 51(3): 243-273.

Kraak, M. 2003. The space-time cube revisited from a geovisualization perspective. Pages 1988-1996 *in* Proceedings of the 21st International Cartographic Conference (ICC), Durban, South Africa.

Kroemer, K.H.E., H.J. Kroemer, and K.E. Kroemer-Elbert, 1997. Engineering physiology: Bases of human factors/ergonomics, 3rd Ed. Van Nostrand Reinhold, New York.

Langran, G. 1992. Time in geographic information systems. Taylor & Francis, Bristol, PA.

Le, Y. 2012. Challenges in data integration for spatiotemporal analysis. *Journal of Map & Geography Libraries: Advances in Geospatial Information, Collections & Archives* 8(1): 58-67.

Leone, F., F. Lavigne, R. Paris, J-C. Denain, and F. Vinet. 2011. A spatial analysis of the December 26th, 2004 tsunami-induced damages: Lessons learned for a better risk assessment integrating buildings vulnerability. *Applied Geography* 31(1): 363-375.

Lime, S. 2008. MapServer. Pages 65-85 *in* G.B. Hall and M.G. Leahy (eds.), Open source approaches in spatial data handling. Advances in geographic information science 2. Springer-Verlag, Berlin.

Lindell, M.K. and C.S. Prater. 2010. Tsunami preparedness on the Oregon and Washington coast: Recommendations for research. *Natural Hazards Review* 11: 69-81.

Liu, Y., Y. Takeuchi, and N. Okada. 2007. Multi-agent based collaborative modeling for flood evacuation planning – Case study of Nagata, Kobe. *Annals of Disaster Prevention Research Institute* 50(B): 241-249.

Liu, Y., N. Okada, and Y. Takeuchi. 2008. Dynamic route decision model-based multi-agent evacuation simulation – Case study of Nagata Ward, Kobe. *Journal of Natural Disaster Science* 28(2): 91-98.

Lynett, P.J. 2011. Tsunami inundation, modeling of. Pages 1008-1021 *in* R.A. Meyers (ed.), Extreme environmental events: Complexity in forecasting and early warning. Springer, New York.

Lynett, P.J. and P.L.F. Liu. 2011. Numerical simulation of complex tsunami behavior. *Computing in Science & Engineering* 13(4): 50-57.

Maantay, J., A. Maroko, and G. Culp. 2010. Using geographic information science to estimate vulnerable urban populations for flood hazard and risk assessment in New York City. Pages 71-97 *in* P.S. Showalter and Y. Lu (eds.), Geospatial techniques in urban hazard and disaster analysis. Springer, Netherlands.

MapServer. 2012. MapServer – Open source web mapping, http://www.mapserver.org (last accessed 20 July 2012).

Mark, D. 1998. Geospatial lifelines [abstract]. Page 12 *in* O. Günther, T. Sellis, and B. Theodoulidis (eds.), Integrating spatial and temporal databases. Dagstuhl Seminar Report No. 228.

Mas, E., F. Imamura, and S. Koshimura. 2011. Modeling the decision of evacuation from tsunami based on human risk perception. Tohoku Branch Annual Meeting. Sendai: Japan Society of Civil Engineering (JSCE), http://www.tsunami.civil.tohoku.ac.jp/hokusai3/J/shibu/22/erick_11.pdf (last accessed 3 May 2012).

Mas, E., F. Imamura, and S. Koshimura. 2012. An agent based model for the tsunami evacuation simulation. A case study of the 2011 great East Japan tsunami in Arahama Town. *In* Joint Conference Proceedings: 9th International Conference on Urban Earthquake Engineering & 4th Asia Conference on Earthquake Engineering, Tokyo.

Mayer, R.E., M. Hegarty, S. Mayer, and J. Campbell. 2005. When static media promote active learning: Annotated illustrations versus narrated animations in multimedia instructions. *Journal of Experimental Psychology: Applied* 11(4): 256-265.

McCarthy, A.C., N.H. Hancock, and S.R. Raine. 2010. VARIwise: A general-purpose adaptive control simulation framework for spatially and temporally varied irrigation at sub-field scale. *Computers and Electronics in Agriculture* 70(1): 117-128.

Merati, N., E. Gica, and C. Chamberlin. 2007. Building tsunami analysis tools into a GIS workspace. *In* ESRI International User Conference Proceedings (UC1889), San Diego.

Merati, N., C. Chamberlin, C. Moore, V. Titov, and T.C. Vance. 2010. Integration of tsunami analysis tools into a GIS workspace – Research, modeling, and hazard mitigation efforts within NOAA's Center for Tsunami Research. Pages 273-294 *in* P.S. Showalter and Y. Lu (eds.), Geospatial techniques in urban hazard and disaster analysis. Springer, Netherlands.

Miller, H. 1991. Modelling accessibility using space-time prism concepts within geographical information systems. *International Journal of Geographical Information Systems* 5(3): 287-301.

Miller, H. and Y. Wu. 2000. GIS software for measuring space-time accessibility in transportation planning and analysis. *GeoInformatica* 4(2): 141-159.

Miller, H. and S.A. Bridwell. 2008. A field-based theory for time geography. *Annals of the Association of American Geographers* 99(1): 49-75.

Mountain, D. 2005. Visualizing, querying, and summarizing individual spatio-temporal behavior. Pages 181-200 *in* J. Dykes, A.M. MacEachren, M-J. Kraak (eds.), Exploring geovisualization. Elsevier, London.

Nagarajan, M., D. Shaw, and P. Albores. 2012. Disseminating a warning message to evacuate: A simulation study of the behaviour of neighbours. *European Journal of Operational Research* 220(3): 810-819.

NANOOS. 2012. NANOOS Visualization System (NVS), http://www.nanoos.org/nvs/nvs.php?section=NVS-Products-Tsunamis-Evacuation (last accessed 12 July 2012).

Niazi, M.A. and A. Hussain. 2011. A novel agent-based simulation framework for sensing in complex adaptive environments. *IEEE Sensors Journal* 11(2): 404-412.

Oikawa, Y. and T. Katada. 1999. A study on the effect of flood experience on the evacuation activity. *Journal of Japan Society for Natural Disaster Science* 18(1): 103-116.

Omira, R., M.A. Baptista, J.M. Miranda, E. Toto, C. Catita, and J. Catalão. 2010. Tsunami vulnerability assessment of Casablanca-Morocco using numerical modelling and GIS tools. *Natural Hazards* 54: 75-95.

Oregon.gov. 2012. Department of Geology and Mineral Industries – Tsunami Evacuation Route Maps, http://www.oregon.gov/DOGAMI/earthquakes/Coastal/CoastalHazardsMain.shtml (last accessed 15 May 2012).

OWASE. 2012. Owase Dynamic Tsunami Hazard Map. Disaster Social Engineering Laboratory, Gunma University, Japan, http://dsel.ce.gunma-u.ac.jp/simulator/owase/en (last accessed 20 April 2012).

Papathoma, M. and D. Dominey-Howes. 2003. Tsunami vulnerability assessment and its implications for coastal hazard analysis and disaster management planning, Gulf of Corinth, Greece. *Natural Hazards and Earth System Sciences* 3: 733-747.

Papathoma, M., D. Dominey-Howes, Y. Zong, and D. Smith. 2003. Assessing tsunami vulnerability, an example from Herakleio, Crete. *Natural Hazards and Earth System Sciences* 3: 377-389.

Peuquet, D.J. 2002. Representations of space and time. The Guilford Press, New York.

PostGIS. 2012. PostGIS – A spatial extension to PostgreSQL, http://postgis.refractions.net (last accessed 20 July 2012).

PostgreSQL. 2012. PostgreSQL relational database management system, http://www.postgresql.org (last accessed 20 July 2012).

Priest, G.R., C. Goldfinger, K. Wang, R.C. Witter, Y. Zhang, and A.M. Baptista. 2010. Confidence levels for tsunami-inundation limits in northern Oregon inferred from a 10,000-year history of great earthquakes at the Cascadia subduction zone. *Natural Hazards* 54: 27-73.

Raubal, M., S. Winter, S. Teβmann, and C. Gaisbauer. 2007. Time geography for ad-hoc shared-ride trip planning in mobile geosensor networks. *ISPRS Journal of Photogrammetry and Remote Sensing* 62(5): 366-381.

Reese, S., B.A. Bradley, J. Bind, G. Smart, W. Power, and J. Sturman. 2011. Empirical building fragilities from observed damage in the 2009 South Pacific tsunami. *Earth-Science Reviews* 107(1-2): 156-173.

Schneider, P. 2011. Tsunami modeling for Seaside, Oregon. Pages 408-416 *in* Proceedings of the 2011 Solutions to Coastal Disasters Conference. American Society of Civil Engineers, Reston, VA.

Shaw, S-L., H. Yu, and L.S. Bombom. 2008. A space-time GIS approach to exploring large individual-based spatiotemporal datasets. *Transactions in GIS* 12(4): 425-441.

Simonovic, S.P. and S. Ahmad. 2005. Computer-based model for flood evacuation emergency planning. *Natural Hazards* 34: 25-51.

Suppasri, A., E. Mas, S. Koshimura, and K. Imai. 2012. Developing tsunami fragility curves from the surveyed data of the 2011 great East Japan tsunami in Sendai and Ishinomaki plains. Coastal Engineering Journal 54(1): 1250008(1-16).

Synolakis, C.E. and E.N. Bernard. 2006. Tsunami science before and beyond Boxing Day 2004. *Philosophical Transactions of the Royal Society* 364: 2231-2265.

Tang, Z., M.K. Lindell, C.S. Prater, and S.D. Brody. 2008. Measuring tsunami planning capacity on U.S. Pacific Coast. *Natural Hazards Review* 9(2): 91-100.

Tappister. 2012. Cannon Beach tsunami inundation and evacuation study 2010-2011, http://www.tappister.com/cannon-beach (last accessed 29 April 2012).

Tarbotton, C., D. Dominey-Howes, J.R. Goff, M. Papathoma-Kohle, F. Dall'Osso, and I.L. Turner. 2012. GIS-based techniques for assessing the vulnerability of buildings to tsunami: Current approaches and future steps. *Geological Society, London, Special Publications* 361: 115-125.

TCP. 2012. Tsunami Computational Portal, http://tsunamiportal.nacse.org (last accessed 28 April 2012).

TimeDB. 2012. TimeDB bitemporal relational DBMS, http://www.timeconsult.com/Software/Software.html (last accessed 24 July 2012).

Titov, V. and F.I. González. 1997. Implementation and testing of the Method of Splitting Tsunami (MOST) model. NOAA Tech. Memo. ERL PMEL-112 (PB98-122773), NOAA/Pacific Marine Environmental Laboratory, Seattle. 11 pp.

Titov, V., A.B. Rabinovich, H.O. Mofjeld, R.E. Thomson, and F.I. Gonzalez. 2005. The global reach of the 26 December 2004 Sumatra tsunami. *Science* 309: 2045-2048.

Tversky, B., J.B. Morrison, and M. Betrancourt. 2002. Animation: Can it facilitate? *International Journal of Human-Computer Studies* 57: 247-262.

Uno, K. and K. Kashiyama. 2008. Development of a simulation system for the disaster evacuation based on multi-agent model using GIS. *Tsinghua Science and Technology* 13(S1): 348-353.

Usery, E.L., J. Choi, and M.P. Finn. 2010. Modeling sea-level rise and surge in low-lying urban areas using spatial data, geographic information systems, and animation methods. Pages 11-30 *in* P.S. Showalter and Y. Lu (eds.), Geospatial techniques in urban hazard and disaster analysis. Springer, Netherlands.

Van Ho, Q., P. Lundblad, T. Åström, and M. Jern. 2011. A web-enabled visualization toolkit for geovisual analytics. *Information Visualization* 11(1): 22-42.

Vance, T.C., N. Merati, S.M. Mesick, C.W. Moore, and D.J. Wright. 2007. GeoModeler: Tightly linking spatially-explicit models and data with a GIS for analysis and geovisualization. *In* Proceedings of the 15th ACM International Symposium on Advances in Geographic Information Systems (ACM GIS 2007), Seattle, WA, 7-9 November.

Wolman, D. 2005. Reaching safe ground. Technology Review, 26 July 2005, http://www.technologyreview.com/business/14626 (last accessed 21 April 2012).

Wood, N.J. and J.W. Good. 2004. Vulnerability of port and harbor communities to earthquake and tsunami hazards: The use of GIS in community hazard planning. *Coastal Management* 32(3): 243-269.

Yeh, H. 2010. Gender and age factors in tsunami casualties. *Natural Hazards Review* 11: 29-34.

Yu, H. 2006. Spatial-temporal GIS design for exploring interactions of human activities. *Cartography and Geographic Information Systems* 33(1): 3-19.

Yu, H. 2008. Visualizing and analyzing activities in an integrated space-time environment: Temporal geographic information system design and implementation. *Transportation Research Record* 2024: 54-62.

Yu, H. and S.-L. Shaw. 2007. Revisiting Hägerstrand's time-geographic framework for individual activities in the age of instant access. Pages 103-118 *in* H.J. Miller (ed.), Societies and cities in the age of instant access. Springer, Netherlands.

Yu, H. and S.-L. Shaw. 2008. Exploring potential human activities in physical and virtual spaces: A spatio-temporal GIS approach. *International Journal of Geographical Information Science* 22(4): 409-430.

Yuan, M., D.M. Mark, M.J. Egenhofer, and D.J. Peuquet. 2005. Extensions to geographic representations. Pages 129-156 *in* R.B. McMaster and E.L. Usery (eds). A research agenda for geographic information science. CRC Press, Boca Raton, FL.

Yuan, M. 1996. Modeling semantics, temporal, and spatial information in geographic information systems. Pages 334-347 *in* M. Craglia and E.L. Usery (eds.), Geographic information research: Bridging the Atlantic. Taylor & Francis, London.

Zerger, A. and S. Wealands. 2004. Beyond modelling: Linking models with GIS for flood risk management. *Natural Hazards* 33: 191-208.

Chapter 4: GridStats: A web-based system for calculating climate grid statistics over varying spatial and temporal scales

Authors: Dylan B. Keon, Christopher Daly, Adam Ryan

4.1 Abstract

Raster-based datasets (grids) provide a simple yet powerful data structure for storing numerical data and performing spatial and statistical analyses across several datasets (i.e., map algebra). Each grid can contain a data representation for a given time period, enabling analysis over time. However, grid file sizes can be very large, particularly at high grid cell resolution. Performing statistical calculations on multiple grids across varying spatial and temporal scales can be computationally intensive, typically requiring a specialized geographic information system (GIS) or image processing software package.

The PRISM Climate Group produces modeled climate grids representing precipitation, minimum temperature, maximum temperature, and mean temperature for the conterminous U.S. at 800 m resolution over multiple time scales (daily, monthly, and annual). With currently over 49,000 climate grids available, grid loading and processing times can become prohibitively long, particularly when performing analyses across large temporal and/or spatial scales. Leveraging and extending a custom server-side solution designed in-house for quickly processing multiple grids, the GridStats system was developed to provide fast, targeted statistical summary and analysis capabilities across virtually any set of PRISM climate grids.

The GridStats system is a scalable, extensible, web-based framework that provides fast access and analysis capability to all PRISM climate grids. Any available gridded PRISM climate parameter can be analyzed over user-selectable space and time ranges using an extendable set of statistical methods. The system is accessed via a web browser, which provides user-level control over all parameter settings, as well as dynamic map-based spatial selection aids to help identify and select a bounded region for analysis. When a request is submitted, custom server-side code processes the requested grid set and returns output that can be viewed and explored in the resulting web pages, which feature synchronized input/output dynamic maps for comparison, as well as dynamically-built interactive charts that display both plotted data points and statistical information (mean, ±1 standard deviation, centered moving averages).

The GridStats system is unique in its ability to process large sets of relatively high-resolution (800 m) gridded climate data very quickly, span virtually any combination of spatial and temporal ranges within the available datasets, perform statistical calculations over space and time, and render the output in dynamic tools that allow further data exploration, while requiring no specialized software apart from a standard web browser.

4.2 Introduction

Climate data, such as accumulated precipitation on a monthly time scale, are typically stored in raster GIS formats (grids). Inherent to each grid are elements of time (the time interval represented by the grid) and space (the spatial extent covered by the grid and the grid resolution). Large collections of climate grids enable the analysis of climate parameters over varying spatial and temporal scales. By altering the spatial extent (i.e., clipping a grid to a defined extent) and selecting a time series of interest (e.g., a set of monthly grids representing January 2000 to December 2003), any number of analyses can be performed on the collection of grids over time. One possibility is to perform a statistical analysis through each set of co-located time-series grid cells, resulting in an output grid with each cell containing the calculated output statistic value. This approach is known generally as map algebra (Tomlin 1990) and is available in any GIS software package that supports raster processing, such as ArcGIS (2012), Geographic Resources Analysis Support System (GRASS) (2012), or Idrisi (2012). Although web-based mapping and services have grown significantly in scope and availability since the introduction of Esri's ArcIMS in 2000 and the introduction of the Google Maps application programming interface (API) in 2005, web-based raster analysis capabilities remain limited.

4.2.1 PRISM Climate Grids

The Precipitation-elevation Regressions on Independent Slopes Model (PRISM) is a statistical-topographic model that uses point data (weather station observations) and a digital elevation model (DEM) to calculate modeled grids representing climate parameters across the conterminous United States (CONUS) (Daly *et al.* 1994, Daly *et al.* 1997). The PRISM Climate Group at Oregon State University has produced gridded climate data at 4 km resolution on monthly and annual time scales since 1995 using a sophisticated knowledge-based approach incorporating extensive inference, refinement, and quality assurance procedures (Daly *et al.* 2002, Daly *et al*. 2008). More recently, the PRISM Climate Group partnered with the Northwest Alliance for Computational Science and Engineering (NACSE) and has begun automated production of CONUS climate grids at 800 m resolution on a daily time scale, using a Linux-based computational cluster to facilitate time-effective processing of daily data. Grids produced in this manner represent the climate parameters precipitation (ppt), minimum temperature (tmin), maximum temperature (tmax), and mean temperature (tmean).

PRISM staff often need to make a quick assessment of a particular climate parameter across space or time, or may need to load hundreds of grids to perform a larger statistical summary or analysis. Across all time series, there are currently over 49,000 climate grids stored on the PRISM filesystem. Performing grid processing operations over large spatial or temporal scales can be time-prohibitive due to (1) moving large quantities of data over the network, and (2) processing large quantities of data on a computer workstation. Moreover, sophisticated web-based grid querying utilities under development by NACSE and PRISM need fast access to grids for real-time processing. A server-side solution was required not only for fast grid access and processing, but also for performing real-time statistical summaries and analyses on selected grid sets over space and time.

4.2.2 Real-time Grid Calculations

By nature of their format, grids provide a simple yet powerful data structure for storing numerical data and performing spatial and statistical analyses across multiple datasets. Each grid can contain a data representation for a snapshot in time (or an interval), enabling analysis over both space and time. Specialized locally-installed GIS software is typically used to perform grid-based analysis. However, the software is usually installed on a particular computer and, in many cases, requires a purchased license before the software can be used. GRASS GIS is the only full-featured open source raster GIS software package, but for most people it requires a steeper learning curve than commercial software, even with the Quantum GIS (QGIS 2012) software package as a front end. Data size and format can also serve as impediments to fast processing of grids, particularly if large quantities of gridded data need to be loaded across a network for processing.

With a powerful server or clustered server environment, centralizing grid storage and processing on the server-side can speed delivery of the output product to the client, whether the client is a locally-installed GIS software package or a web browser. This is the driving concept behind ArcGIS Server (ArcGIS 2012), which recently began offering more server-side grid processing capabilities (ArcGIS Resource Center 2012), and similar software products. However, dynamic grid processing (i.e., not just delivering data, but rather performing complex grid processing operations and delivering the output) in a server environment remains a challenge. Programmatic approaches offer the most flexibility for designing a server-side grid processing solution, but can also require significant effort to build. Significant advances in open source GIS software over the past decade have resulted in powerful server-side software components such as the Geospatial Data Abstraction Layer (GDAL 2012), OGR Simple Features Library library (OGR 2012), MapServer (Lime 2008, MapServer 2012), and PostGIS (PostGIS 2012), all of which can potentially be leveraged and combined with custom software to create a grid processing system.

If the grid processing work can be moved to the server, and the client only requires operations such as data exploration, visualization, and querying of the output data, custom web-based tools become ideal candidates for a client interface. Requiring no specialized software apart from a standard web browser, web-based tools have the benefit of being accessible from virtually anywhere. With the current panoply of advanced JavaScript-based client-side libraries designed to enhance the user experience, such as OpenLayers (OpenLayers 2012), jQuery (jQuery 2012), and Highcharts (Highcharts 2012), and the availability of powerful techniques for making asynchronous calls to server-side processes to retrieve data, web-based interfaces designed today can be more useful and powerful than ever before.

Using the internet as the basis for a grid analysis framework requires inherently fast approaches to grid processing and delivery. Although it is possible, for example, to develop a web-based system that e-mails the user when a computationally-intensive job is complete, the user experience is greatly enhanced by near-real-time processing and delivery of the output products. This necessitates a streamlined grid processing solution on the server-side, capable of quickly handling complex grid processing operations as well as multiple simultaneous requests.

4.2.3 Project Goals

The intent of this project was to implement a usable web-based approach for analyzing PRISM climate grids over varying spatial and temporal scales. Leveraging the structured system of PRISM climate grids and extending a previously-developed grid processing toolkit, the GridStats system was developed with these goals in mind:

1. Determine the feasibility of performing statistical analyses across multiple grids in a real-time web environment.
2. Create a web-based system for calculating statistics across multiple grids over user-defined spatial and temporal scales.
3. Leverage the custom server-side solution (the GridServer) designed in-house for quickly processing multiple grids and extend its point-based processing capability to process regions across multiple grids.

4. Design a system that can handle the processing power necessary for calculating analyses across the high resolution (800 m) PRISM grids.
5. Design user interactivity features that help the user properly define input parameters and interpret the output data.
6. Incorporate dynamic mapping tools, both for setting the initial area of interest and for exploring the statistical output grids.
7. Develop dynamic, interactive charts that display the output data and allow their exploration, plus incorporate in-chart statistical reference aids.
8. Attempt to build the system framework, statistical analysis, and grid processing code using open source software tools.

A range of hardware, software, database, mapping, analysis, and development techniques were used in the design and implementation of the GridStats system. The overarching goal of the project was to combine this complex set of elements into a system that provides automated, easy-to-use exploration and analysis of large datasets in a web-based context. Developing a modular framework that can be used with additional climate parameters, raster data types, and statistical methods was also an important consideration in this project.

4.3 Related Work

Several topics associated with this project such as climate mapping, web-based grid processing, temporal map algebra, and calculation of grid statistics have been examined by others. In this section, related research is discussed in the context of these topics, with particular attention given to the central theme of processing grids across space and time.

4.3.1 Web-based Grid Processing

The concept of real-time web-based processing of raster datasets has existed for several years (e.g., Baumann 2001). Websites processing grids (either by querying grids and delivering the data to the user, or by clipping grids and delivering a grid product) have typically used processing scripts that either call a GRASS GIS utility on the server to process a set of grids, or a custom-written software tool to perform grid querying or clipping. Many examples of these data analysis and delivery approach exist on websites today (e.g., PRISM Data Explorer 2012, USGS National Elevation Dataset 2012).

Prior to the advent of the Open Geospatial Consortium (OGC) Web Coverage Service (OGC WCS 2010), no standards existed for the processing and delivery of raster datasets. The WCS standard defines procedures for the processing and delivery of time-series raster-based datasets ("coverages") on remote servers, with delivery of the output to the requesting server. Since its introduction, OGC WCS capabilities have been added to certain spatial data infrastructures (SDIs), enabling access and analysis capabilities to any client. For example, Bernard and Ostländer (2007) demonstrated the use of a WCS server that included a map algebra service for assessing climate change vulnerability in Arctic regions. Foerster *et al.* (2011) described the use of the OGC WCS and WPS (Web Processing Service) standards for multiple grid-based spatial analysis scenarios.

The GDAL library contains driver support for the OGC WCS standard, handling any WCS coverage as a raster dataset, with GDAL acting as a client to the WCS server. In the current version (10.1) of ArcGIS, Esri enables geoprocessing services to be published with ArcGIS for Server. Esri also indicates in documentation (ArcGIS Resource Center 2012) that server-side processing can be performed with raster functions and published via ArcGIS for Server. ArcGIS for Server also includes support for the OGC WCS standard.

4.3.2 Web-based Climate Grid Analysis

Certain websites allow the user to retrieve data values across multiple time-series climate grids and calculate statistics across the selected grid sets. The PRISM Climate Group developed and continues to support the PRISM Data Explorer (2012), designed several years ago and built around a mix of technologies including MapServer, JpGraph, and GRASS GIS. The user can select a point location of interest via a map click or by entering coordinates manually, select a month, and then select a year range for which to retrieve data for that month (i.e., all Januaries from 1934 to 1967). When the request is submitted, a report is returned with information about the selected grid cell, the data values for that grid cell by year, and a filled line chart graphing the data values across the time series. The system works by sending a request to a script that calls a GRASS GIS utility to dynamically fetch the data values from the requested grids and return them to the browser. The PRISM Data Explorer extracts values from 4 km resolution PRISM grids. It works well for extracting data over time for a single month or all months in a range, but uses dated technology and lacks interactive tools for examining the output or for performing more sophisticated queries.

WestMap, a climate grid analysis website developed through a collaboration of the University of Arizona, the Western Regional Climate Center (WRCC), the Desert Research Institute (DRI), and PRISM, enables fairly extensive analysis of PRISM climate grids (Comrie *et al*. 2006, Glueck *et al*. 2008, WestMap 2012). A WestMap user can select a location of interest using a number of methods (states, counties, climate divisions, hydro units, by pixel, or by polygon). The first four location selection methods use a custom imagemap-selection method, while the last two use a Google Maps API-based map to set a point or define a polygon. After selecting a point or region of interest, the user can choose to create either a time series chart or a map of the region. Time series charts display parameter data for the selected time interval, as well as a running mean plotted as points, and an optional summary statistics report for the selected data values. Three types of maps can be generated for any of the climate parameters: (1) anomaly, expressed as a difference or a percent of normal, (2) a composite of several years, or (3) a difference between two years (or two year ranges).

WestMap covers states in the mountainous western US (west of Kansas) and analyzes PRISM 4 km climate grids. The website is written in the PHP: Hypertext Preprocessor (PHP) language, with a combination of C, Fortran, Perl, and other languages used to dynamically retrieve data from PRISM climate grids and perform analyses on them. Output charts and maps are static (i.e., there is no interactive component once the products are generated). Output values cannot be retrieved from the resulting maps, and cannot be viewed on the chart (although there is an option to print them below the chart). The generated maps and charts can be saved as images.

The Nature Conservancy, in collaboration with the University of Washington and the University of Southern Mississippi, developed the ClimateWizard website, a tool that allows the user to select a state or country and learn how climate has changed over time in that region (Girvetz *et al.* 2009, ClimateWizard 2012). As with the other climate grid analysis tools mentioned in this section, ClimateWizard uses the PRISM 4 km climate grids for analysis (in addition to other data sources). Various general circulation models and emission scenarios can be selected for future climate projections, and resulting maps are displayed in an Esri-based interactive mapping tool. Cell values can be retrieved by clicking on the interactive map, and are plotted on an interactive chart that displays the model and value for each data point for the modeled scenarios (the data values are also supplied in a table). Data and map images can be exported from the tool. A separate custom query interface (ClimateWizard Custom 2012) allows more advanced analysis of global regions (50 km resolution) or regions in the U.S. (4 km to 12 km resolution). Advanced analysis options include departure analysis (from the 1961-1990 normals) and linear trend analysis.

4.3.3 Map Algebra and Grid Statistics

Map algebra, the operation of performing calculations through aligned grid cells to produce a new grid, is a key concept in grid processing (Tomlin 1990, Tomlin 1994). Programmatically, map algebra calculations on grid sets representing data across space and time (i.e., the space-time cube) can be approached as multidimensional array operations, where each grid is a 2-dimensional (2D) array of rows and columns, and the set of time-series grids aligned for processing represents the third dimension. By calculating through columns of the 2D arrays (Figure 4.1), map algebra can be performed to produce an output array, with each output array element containing a calculated value. This approach is discussed in detail by Frank (2005), who described an extended version of Tomlin's map algebra for handling calculations across spatiotemporal raster datasets. Frank's extended map algebra provides consistent operations for single maps, time series data, and stacks of gridded time series data (as in Figure 4.1).

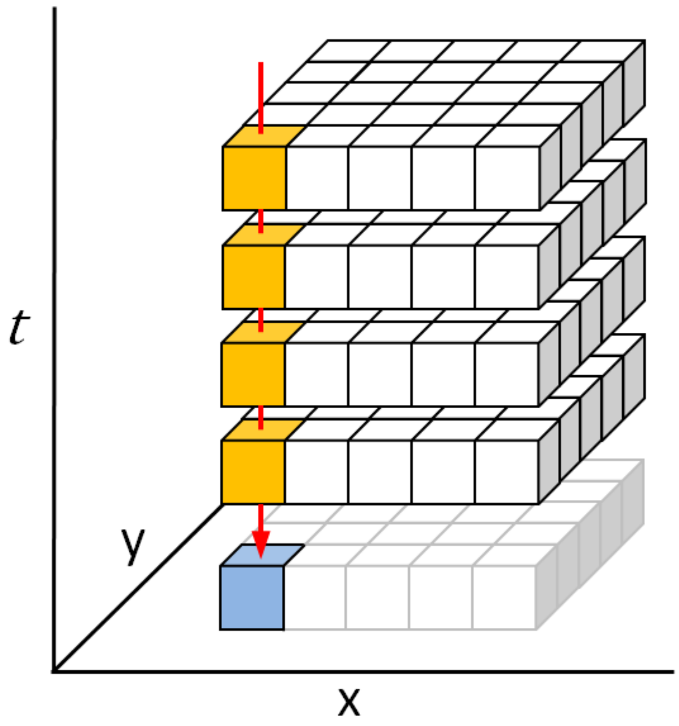


Figure 4.1: Example of map algebra for spatiotemporal data. This diagram illustrates calculating through grid cells (or 2D columns in a 3D array) over time to produce an output grid (or 2D array) containing the calculated values.

Similar to Frank (2005) in concept, Mennis *et al.* (2005) described cubic map algebra functions for spatiotemporal analysis, extending standard map algebra to three dimensions. Their cubic map algebra approach was tested by summarizing Normalized Difference Vegetation Index (NDVI) anomaly values during El Nino/Southern Oscillation (ENSO) phases across different land covers. Later, Mennis (2010) extended and formalized this approach as “multidimensional map algebra” (MMA), an extension of the conventional map algebra. MMA functions operate on multidimensional data in many forms, including 2-D in space and 1-D in time (as represented in Figure 4.1).

An additional map algebra approach for processing spatiotemporal data was proposed by Gutierrez *et al.* (2007), who compared raster map algebra techniques to Online Analytical Processing (OLAP) data cubes and described their manipulation as multi-dimensional arrays. Cordiero *et al.* (2009) described another map algebra approach to extend geoalgebra and image algebra, but primarily focused on spatial representations and manipulations related to cellular automata rather than considering the temporal dimension.

Grid-based statistical operations have been available in GIS software packages for many years. The ESRI ARC/INFO GRID module was added in version 6.0 of ARC/INFO (released in 1991) and supported command-line access to raster operations, including statistical functions. The core functionality of GRID was later used as the basis for the ArcGIS Spatial Analyst (ArcGIS Spatial Analyst 2012) extension, where statistical grid calculations are performed in ArcGIS today. GRASS GIS has supported a wide range of grid-based statistical functions since the 1980s, and continues to be under active development. The open source statistical package R (R Project 2012) provides support for spatial statistics on grids via the sp R package (Sp R 2012), and can be called from within a GRASS environment using the R/GRASS interface (R/GRASS 2012). Bivand *et al.* (2008) describe many examples of applied spatial data analysis on grids using R. Access to R from the Python programming language is available via the rpy2 module (Rpy2 2012), enabling programmatic calculation of statistical functions across grids imported as arrays. The NumPy (NumPy 2012) and SciPy (SciPy 2012) Python modules also support a range of statistical functions, and NumPy supports fast operations on grids as multidimensional arrays.

4.3.4 Server-side Grid Processing Systems

Server-side grid operations can require significant processing power, particularly for large spatial and temporal scales, but also during periods of concurrent usage. Having a properly-configured server dedicated to grid processing operations is one way ensure good performance. However, increased usage, higher-resolution gridded datasets, and larger quantities of data can all lead to increased load on the processing server. Other possibilities exist such as distributed computing over scientific grids (e.g., Giuliani *et al.* 2011), or the use of Amazon's Elastic Compute Cloud (Amazon EC2 2012). Siládi *et al.* (2012) used Amazon's EC2 to perform parallel processing to predict depth of snow cover, using inverse distance weighting interpolation calculations in the cloud.

Auer (2012) describes an intriguing approach to real-time web GIS analysis using the Web Graphics Library (WebGL) JavaScript API. WebGL is based on the OpenGL 3D graphics library, and runs natively in the user's browser, where it has access to the graphics processing unit (GPU) on the graphics card. The GPU on modern graphics cards is capable of performing fast parallel processing of computational tasks. Auer outlines a process for performing WebGL-based parallel spatial analysis of grids. Bryan (2012) also used a GPU-processing-based approach in assessment and modeling of socio-ecological systems, running tests across a range of processing types. Bryan realized a performance gain using parallel GPU processing that was several orders of magnitude faster than with standard GIS software on a computer workstation. Although parallel GPU approaches to grid processing are promising, they may require more research before their reliable implementation in a production environment can be realized.

4.4 The GridStats System

The GridStats system was designed to give PRISM Climate Group staff the means to quickly analyze and compare a point (single grid cell) or a region (via a drawn bounding box or other defined polygon boundary) over varying spatial and temporal scales. The system was designed from the ground up, making extensive use of existing open source software packages as well as other grid processing software tools developed in-house (GridServer, Biltools).

4.4.1 Design Intent

The GridStats system was designed to meet user interface goals as well as particular development goals guiding future development. The user interface was designed to be fully interactive, offering full control over all spatial and temporal scale selections. Results pages have clearly-indicated controls and intuitive visualizations of the output data and analysis results.

In addition to developing the key functional code driving the grid processing operations, particular attention was given to modularity of the code design. Both the frontend and backend codebases are designed such that additional univariate and multivariate statistical methods, polygon-based area selections, and climate parameters can be added fairly easily, without re-engineering server-side or client-side code.

4.4.2 System Architecture

The GridStats system features an interactive web-driven utility that allows input selections to be defined on a settings web page, while output data can be viewed, explored, and exported on a set of results pages. Complex JavaScript-based client-side code manages input selection, asynchronous queries and responses, output formatting, and output visualization. Behind the scenes, a complicated system of custom server-side software tools manage grid logistics, input/output, processing, and statistical analysis of climate grids across varying spatial and temporal scales. Figure 4.2 summarizes the interconnected pieces that define the system architecture.

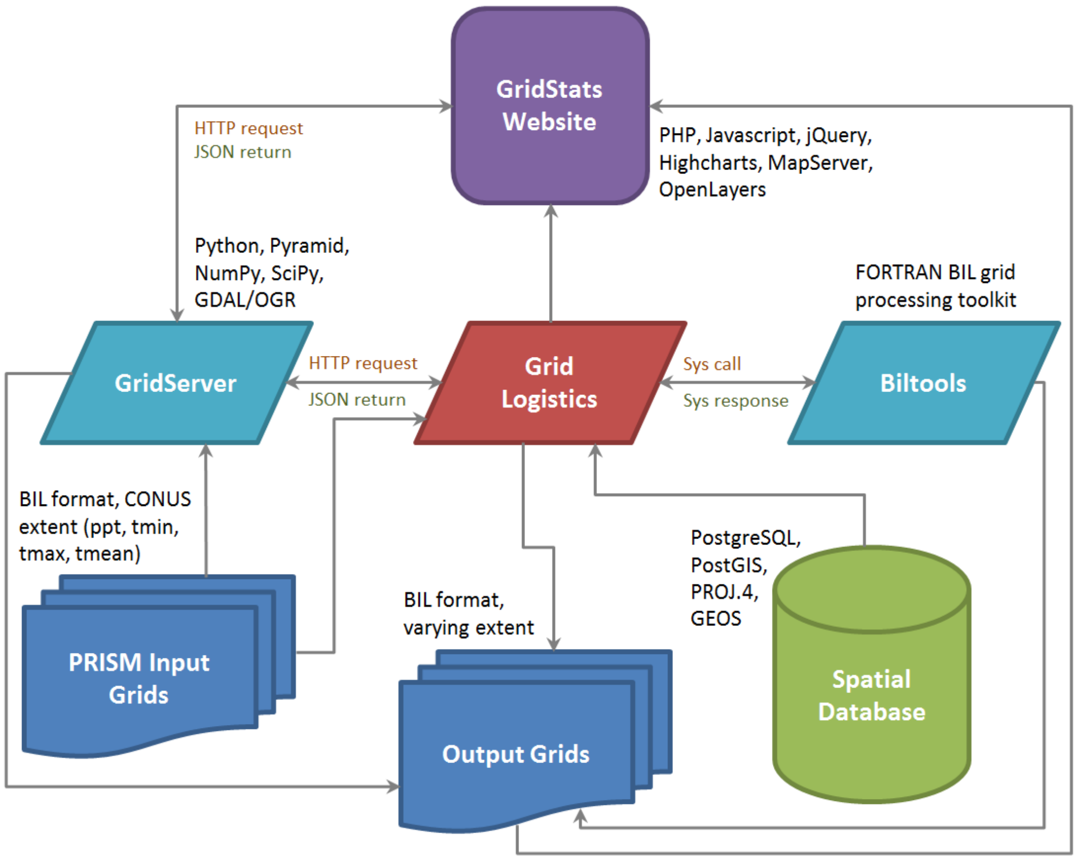

GridStats Website
HTTP request
JSON return
PHP, Javascript, jQuery, Highcharts, MapServer, OpenLayers
Python, Pyramid, NumPy, SciPy, GDAL/OGR
FORTRAN BIL grid processing toolkit
GridServer
HTTP request
JSON return
Grid Logistics
Sys call
Sys response
Biltools
BIL format, CONUS extent (ppt, tmin, tmax, tmean)
PostgreSQL, PostGIS, PROJ.4, GEOS
BIL format, varying extent
PRISM Input Grids
Output Grids
Spatial Database

Figure 4.2: GridStats system architecture. After input parameters are submitted via the website, the processing engine sends a request to either the GridServer or biltools, depending on the request type. Output grids are written to the filesystem and output data are returned in JavaScript Object Notation (JSON) format to the processing engine and used in reporting and charting. All output is immediately available in the results website.

4.4.3 Hardware and Software

The GridStats system requires fairly significant computational resources in order to process *n* number of grids across varying spatial and temporal scales and return the output to the user as quickly as possible. Dedicated database, web, and grid processing servers (Table 4.1) distribute the processing load among multiple machines.

Table 4.1: Dedicated servers used by the GridStats system. The servers listed here are all connected via a 1 gigabit local network.

Specification	**Web Server**	**Database Server**	**Grid Processing Server**
Server type	Dell PowerEdge R710	Sun SF X4170	Dell PowerEdge C2100
CPU	Intel Xeon E5630 (four 4-core CPUs)	Intel Xeon X5560 (four 4-core CPUs)	Intel Xeon X5650 (four 6-core CPUs)
Memory	32 GB	32 GB	64 GB
Disk storage type	Local	EMC NS-480 NAS	Local
Disk quantity	8	9 (half allocated to this server)	24
Disk capacity	320 GB	600 GB	600 GB
Disk speed	10K RPM	15K RPM	15K RPM
Disk interface	SCSI	Fibre channel	SAS
RAID level	RAID 5	RAID 5	RAID 6
RAID controller type	hardware	hardware	hardware
Operating system	Linux (RHEL 6)	Linux (RHEL 6)	Linux (RHEL 6)

The GridStats system relies upon a range of open source software packages (Table 4.2), as well as custom code written in PHP, Javascript, and HTML (about 5000 lines of code, not including GridServer modifications written in Python). All spatial data queries, processing, and server-side and client-side manipulations are performed using open source software tools and custom-written software tools and utility scripts.

Table 4.2: Open source software packages used by the GridStats system, and the purpose for which they are used.

Software Package	**Purpose**
PostgreSQL *(postgresql.org)*	Relational database system – provides storage and management of input and output data.
PostGIS *(postgis.refractions.net)*	Module that provides support for spatial objects and operations within PostgreSQL, and enables spatial queries.
GEOS *(geos.osgeo.org)*	Library incorporating the OpenGIS Simple Features operations, which are used within PostgreSQL/PostGIS.
PROJ.4 *(proj.osgeo.org)*	Library that handles coordinate system transformations on spatial objects stored in the database.
GDAL *(gdal.org)*	Library for manipulating, converting, and processing raster data (grids) in multiple formats.
OGR *(gdal.org/ogr)*	Library for processing vector datasets and converting output from PostgreSQL/PostGIS to GeoJSON.
MapServer *(mapserver.org)*	Server-side software that processes map layers for the OpenLayers web mapping and visualization interface.
OpenLayers *(openlayers.org)*	Client-side mapping tool – enables web mapping and animated visualization of maps and output data.
Highcharts *(highcharts.com)*	Client-side charting tool – enables interactive charting capabilities via Javascript.
jQuery (*jquery.com*)	JavaScript library for event handling, AJAX transactions, HTML DOM changes, and cross-browser compatibility.
Python *(python.org)*	Scripting language used for server-side grid processing and associated tasks.
NumPy (*numpy.scipy.org*)	Numerical Python package, used for processing grids as multidimensional arrays and calculating statistics.
SciPy (*scipy.org*)	Scientific tools for Python, used for calculating certain statistics.
PHP *(php.net)*	Server-side web programming language that drives the GridStats website.
Pyramid (*pylonsproject.org*)	WSGI web framework that provides a fast, lightweight web server and enables HTTP requests to Python utilities.
Apache *(httpd.apache.org)*	Web server used for serving website pages and handling PHP-generated content.

4.4.4 Input Data Sources

The GridStats system is built around the availability of a key dataset (the PRISM 800 m climate grids) and various supporting spatial datasets used for display, manipulation, and grid clipping operations. These input data sources are stored on the filesystem or in the spatial database, depending upon type.

4.4.4.1 *PRISM Climate Grids*

Automated PRISM modeling processes produce climate grids at both daily and monthly time scales (annual grids are calculated from monthly data) for the CONUS. The spatial extent of the climate grids is identical (i.e., CONUS) across all grids, as is the grid cell resolution (800 m). Table 4.3 lists the available PRISM modeled climate grids by parameter and time series.

All PRISM climate grids are stored in the BIL (band interleaved by line) format (ESRI 1999, NDIIPP 2012), also described as the EHdr (ESRI .hdr labeled) format in GDAL documentation (GDAL Raster Formats 2012). The BIL format supports coordinate systems via a `.prj` file and georeferencing via the matching `.hdr` file. As a widely-used, generalized binary raster data format, BIL is well-supported in GDAL raster processing operations, as well as in commercial software products from Esri and image processing software vendors such as ENVI and Imagine. Stored in BIL format, each PRISM grid requires approximately 85 MB of disk space. Across all parameters and time series, there are currently over 49,000 available grids occupying nearly 5 TB of disk space on the filesystem.

Table 4.3: Climate parameters and availability of 800 m resolution time-series grids produced by the PRISM Climate Group. Automated climate modeling is performed on a daily basis, with the most recently-available grids typically representing yesterday, last month, or last year, depending upon the time series of interest.

	Time Series		
Parameter	**Daily**	**Monthly**	**Annual**
Precipitation (ppt)	1980-01-01 to <yesterday>	1895-01 to <last month>	1895 to <last year>
Minimum Temperature (tmin)	1970-01-01 to <yesterday>	1895-01 to <last month>	1895 to <last year>
Maximum Temperature (tmax)	1970-01-01 to <yesterday>	1895-01 to <last month>	1895 to <last year>
Mean Temperature (tmean)	1970-01-01 to <yesterday>	1895-01 to <last month>	1895 to <last year>

4.4.4.2 *Additional Spatial Data*

In addition to the PRISM climate grids, a set of vector-based GIS spatial layers (Table 4.4) are used by GridStats. These layers were imported into the spatial database and are used in both the web interface and in the grid processing system.

In the web interface, the initial settings page gives the user the ability to overlay the states, counties, watersheds, and PLSS layers as spatial reference aids when selecting a location of interest. Additionally, the state, county, and watershed layers are enabled as "active selections," whereby a user can select a polygon of interest (e.g., the state of Idaho) from a menu and be zoomed directly to the highlighted polygon on the map, while that polygon is also selected as the region of interest. When a state, county, or watershed area-based request is submitted, the spatial extent information for that polygon is extracted from the spatial database and used for clipping the grid to isolate the associated grid cells for analysis.

Table 4.4: Vector-based spatial data stored in the spatial database and used in both the web interface and the grid processing system.

Spatial Data	**Type**	**Source**
US state boundaries	Polygon	National Atlas (*nationalatlas.gov/atlasftp.html*)
US county boundaries	Polygon	National Atlas (*nationalatlas.gov/atlasftp.html*)
Watershed boundaries (8-digit HUCs)	Polygon	US Geological Survey (*water.usgs.gov/huc.html*)
Public Lands Survey System (PLSS) *	Polygon	US DOI Bureau of Land Management (*blm.gov/wo/st/en/prog/more/gcdb.html*)

* Currently used only for spatial reference on maps – not used in the grid processing system.

4.4.5 Spatial Database

The database used by the GridStats system is stored and managed in an instance of the open source PostgreSQL relational database management system (PostgreSQL 2012) with the open source PostGIS module (PostGIS 2012) included to support the storage, manipulation, and querying of spatial data as database objects.

PostgreSQL/PostGIS allow the execution of advanced spatial queries and can be accessed directly via web programming languages such as PHP and web mapping tools such as MapServer (Lime 2008, MapServer 2012). The PostgreSQL database server instance runs on a dedicated machine as described in Table 4.1.

Spatial data used in this project were converted to the geographic WGS84 coordinate system (EPSG 4326) prior to loading into the spatial database. This conversion was accomplished via the `ogr2ogr` command-line utility (packaged with the OGR library), using this command:

```
ogr2ogr -t_srs EPSG:4326 state_bnd.shp states_4326.shp
```

The converted data were tested and then prepared for insertion into the spatial database via the `shp2pgsql` command-line utility (packaged with the PostGIS module), using this command:

```
shp2pgsql -I -s 4326 states_4326.shp gis.states | psql -d my_db
```

The last section of that command (`psql -d my_db`) takes the output from `shp2pgsql` via a Unix "pipe" and imports it directly to the spatial database using the PostgreSQL `psql` command-line utility. All supporting vector-based spatial datasets were processed and inserted into the spatial database in this manner. GiST spatial indexes were generated for all of the geometry columns in the resulting spatial tables.

4.4.6 The GridServer

The GridServer is a server-side grid processing software system designed to accept a request, process a grid or a set of grids, and return the resulting data to the requesting entity. The GridServer was developed by NACSE specifically for PRISM operations, although it is extensible to any set of gridded data with a regular spatial extent, and provides core functionality to the GridStats system for quickly performing statistical operations across multiple PRISM climate grids.

4.4.6.1 *Purpose*

The purpose of the GridServer is to process requests for grid data as fast as possible and return the output data. Requests typically originate from a web-based application, with the possibility of a large number of concurrent users, so grid processing must occur very quickly. The GridServer was initially designed to receive a latitude/longitude point request, map it to a single grid cell, retrieve data from that grid cell over time (from a single grid to potentially thousands of time-series grids), organize the output data, and return it to the requestor. As part of the GridStats project, functionality was added to the GridServer to support region-based processing using a rectangular bounding box or a state, county, or watershed polygon boundary.

4.4.6.2 *System Design*

The GridServer is written in Python and wrapped in a multi-threaded web application using the Pyramid (2012) framework. Pyramid is a WSGI (Web Server Gateway Interface)-based system that utilizes Paster, a fast, lightweight web server. Running on a dedicated server (described previously in Table 4.1), the system handles HTTP-based requests from a web application or other client. The GridServer makes use of a number of open source libraries such as NumPy, SciPy, and GDAL/OGR.

The GridServer establishes a Unix system file handle to every available PRISM climate grid during initialization. By eliminating the system overhead involved with establishing a file handle and opening a file programmatically, the GridServer can achieve very fast seek times to over 49,000 grids (that quantity is increasing daily). The maximum number of file handles available to the user agent running the GridServer process was changed from the Linux system default of 4096 to 65,536 to accommodate the large number of file handles required by the GridServer. The system itself can accommodate a much larger number of file handles, and the user agent's allocation will be increased further as larger quantities of PRISM climate grids are produced.

NACSE performed extensive testing of grid request processing times over the network-based NFS filesystem vs. local disk storage and found that request processing was consistently 6-7 times faster using local disk storage. The primary set of PRISM grids are stored on the internal NFS filesystem to provide network access and ease of regular backups and need to remain there, so to maximize GridServer performance a local copy of the grids was made on the dedicated GridServer machine. This secondary copy is synchronized with the primary source (1) anytime a new grid is produced, and (2) on a nightly basis via the *rsync* and *cron* Unix system utilities. The GridServer maintains Unix system file handles on this local, synchronized set of grids and refreshes all data file handles hourly, or whenever a refresh is requested.

When a request is received, the GridServer builds a list of the required grids based on the requested time series and start/end dates, reads relevant data from the grids using the established file handles, and processes the grids according to the request. For point-based requests, both the actual grid values and processed data are written, along with matching metadata, to a JSON object that is passed back to the requestor. For area-based requests, the data are first copied into a NumPy N-dimensional array (ndarray). Once the relevant grid data are contained in an ndarray, it becomes much easier to manipulate the data for processing than working with the grids in their native binary format, and the NumPy package provides a wealth of numeric processing functions that operate on the ndarray data structure. The retrieved data are then written to a new ndarray, processed as needed according to the request, and passed into a function that generates a new grid (or set of grids) in BIL format. This is another benefit of the BIL format – the binary data structure is non-proprietary and relatively easy to reproduce via Python. Output grids are written to a filesystem location accessible to the web server. Metadata describing the grids are returned in a JSON object, similar to point data requests.

4.4.6.3 *GridStats Usage*

After a user makes their selections on the initial settings page and clicks the Submit button, a request is sent to the GridServer for one of several grid processing methods. An example area-based HTTP request looks like this:

```
http://<server.domain>:<port>/stats_area_years?minlon=-122.43271
&minlat=41.33694&maxlon=-121.29877&maxlat=43.56572
&start=1982&end=1988&param=ppt&pstat=mean
```

This request is for annual precipitation data for the years 1982-1988, for a region defined by a bounding box drawn on the map on the initial settings page. The requested statistic is "mean," which performs the mean calculation for every grid cell across the years 1982-1988 within the requested region.

4.4.7 Server-side Processing

The GridStats system relies upon a range of server-side components, including a range of open source packages and modules used directly or accessed via APIs, and custom-written software tools. Hardware and software components enabling server-side GridStats operations are listed and described in sections 4.2 and 4.3. Their usage is described in this section, particularly for the components involved in calculating grid statistics.

4.4.7.1 *Calculating Grid Statistics*

Requests spanning the CONUS extent (i.e., the entire PRISM grid extent) are handled by making a system call to `bilcalc`, one of the Fortran utilities included in Biltools. The `bilcalc` command currently supports min, max, mean, and sample standard deviation calculations. All other grid statistic calculations are handled by the GridServer.

The GridServer's initial design intent was to handle point-based requests, retrieve data from matching grid cells across all grids as fast as possible, and return the data to the requestor. The GridStats system leverages the GridServer's core functionality of quickly retrieving and processing grids, as well as the initial point-based request processing methods. However, in developing GridStats, the GridServer's point-based methods were extended to calculate a set of summary statistics across all grid cells retrieved by each point-based request, with new area-based methods developed for this purpose.

The GridServer (in addition to other GridStats framework components, including both the client-side and server-side code driving the website) utilizes some key open source GIS code libraries. In particular, for this project the GDAL/OGR library enables reading, reprojection, and general manipulation of raster datasets (via GDAL), and reading, writing, reprojection, and spatial queries of vector datasets (via OGR). All operations are handled programmatically, either by direct command-line calls, integration of Python modules, or using the GDAL/OGR features compiled into other open source GIS software such as MapServer.

4.4.7.1.1 Extending the GridServer

Additional functions and methods were added to the GridServer to enable area-based requests, which are defined by either drawing a rectangular bounding box on a map or selecting a desired state, county, or watershed polygon boundary. For rectangular bounding box requests, the GridServer uses the bounding coordinates to clip the rectangular region from each input grid, read the clipped regions into a NumPy ndarray, conduct statistical calculations on the array elements, and return the processed output data as a BIL grid, along with associated metadata.

Area requests that use irregular polygons as the bounding region require more complicated logic and processing than rectangular regions. When the user selects a polygon region on the settings web page, the matching spatial database table name is recorded in the background, along with the numerical identifier representing the selected polygon. That information is included in the GridServer request, triggering a clip-by-polygon operation rather than a normal clip-by-bounding-box operation. The Python code driving the GridServer imports the OGR module, which enables a query to be issued to the spatial database to retrieve the encoded geometry for the polygon of interest. The form of that query is:

```
SELECT AsBinary(the_geom) AS wkb_geometry FROM <spatial_table> WHERE
gid = <id>;
```

The GDAL module (also imported by the GridServer Python code) is then used to rasterize the bounding polygon to a grid matching the resolution and alignment of the input grids. This is done as an inclusive process, such that all grid cells either contained by the polygon or intersecting its boundary are included in the rasterized layer. Once properly retrieved and rasterized in this manner, the resulting layer can be used as a Boolean spatial mask to process and return only the grid cells in the polygon of interest.

Selecting a region of interest based on a polygon is an important feature that allows the user to calculate grid statistics over space and time based on significant natural (e.g., watershed) or administrative (e.g., state or county) boundaries. For instance, a user could quickly calculate the minimum precipitation over the years 1990-2010 in the Grand Canyon Watershed in Arizona (this request takes 0.86 seconds to return the output grid and data). The GridStats system currently includes US states, US counties, and watersheds (8-digit HUCs) as selectable polygon layers, but additional polygon layers can easily be added to the framework.

For all area-based methods, the GridServer code runs extensive validity checks against the requested bounding region to ensure that:

1. The requested region does not lie entirely outside of the PRISM grid mask extent (e.g., the bounding box was drawn entirely in Canada).
2. Any areas of the requested region drawn partially outside of the PRISM grid mask extent (e.g., a bounding box was drawn that includes part of California and extends several km into the Pacific Ocean) have the grid cells properly set to "no data" (-9999) in the output grids, and are excluded from statistical calculations.

Table 4.5 summarizes the grid statistics available via each server-side tool and request type. Point requests calculate a set of statistics through the set of retrieved grid cells (i.e., the statistic is calculated through time), and the resulting output is returned as a summary set and displayed on the website. Area requests perform statistical operations using two different methods, and return a grid (or set of grids, for Principal Components Analysis) along with the related data values. Requests for processing across the entire grid extent (CONUS) are handled by Biltools, which are a set of Fortran tools written by the PRISM Climate Group specifically for processing statistics or basic map algebra operations across the entire CONUS. The output from an "entire grid" operation is a single BIL grid with each grid cell representing the calculated statistic through time.

Table 4.5: Currently implemented statistics listed by request type and the server-side tools used for calculating them. The statistics listed for point-based requests are returned as a complete set, while area-based requests process and return a single output statistic.

	Server-side Tool and Request Type		
Statistic	**GridServer (point method)**	**GridServer (area method)**	**Biltools (entire grid)**
Sum	X		
Range	X		
Min	X	X	X
Max	X	X	X
Mean	X	X	X
Median	X		
Sample Standard Deviation	X	X	X
Standard Error	X		
Principal Components Analysis (PCA)		X	

4.4.7.1.2 Area-based Methods

The GridServer's area-based methods (example in Appendix H) calculate statistics in two ways, with both approaches automatically calculated for every area-based request. The first approach is "temporal first," in which the requested statistic is calculated through time (i.e., calculated vertically through each aligned grid cell, with each grid representing a snapshot of values in time), ultimately producing a spatial output grid in which each grid cell contains the output statistic value calculated through time (Figure 4.3). This technique matches the multidimensional map algebra approach described by Mennis (2010) where the data cube is 2D over space and 1D through time, and is similar to the summary operation described by Frank (2005).

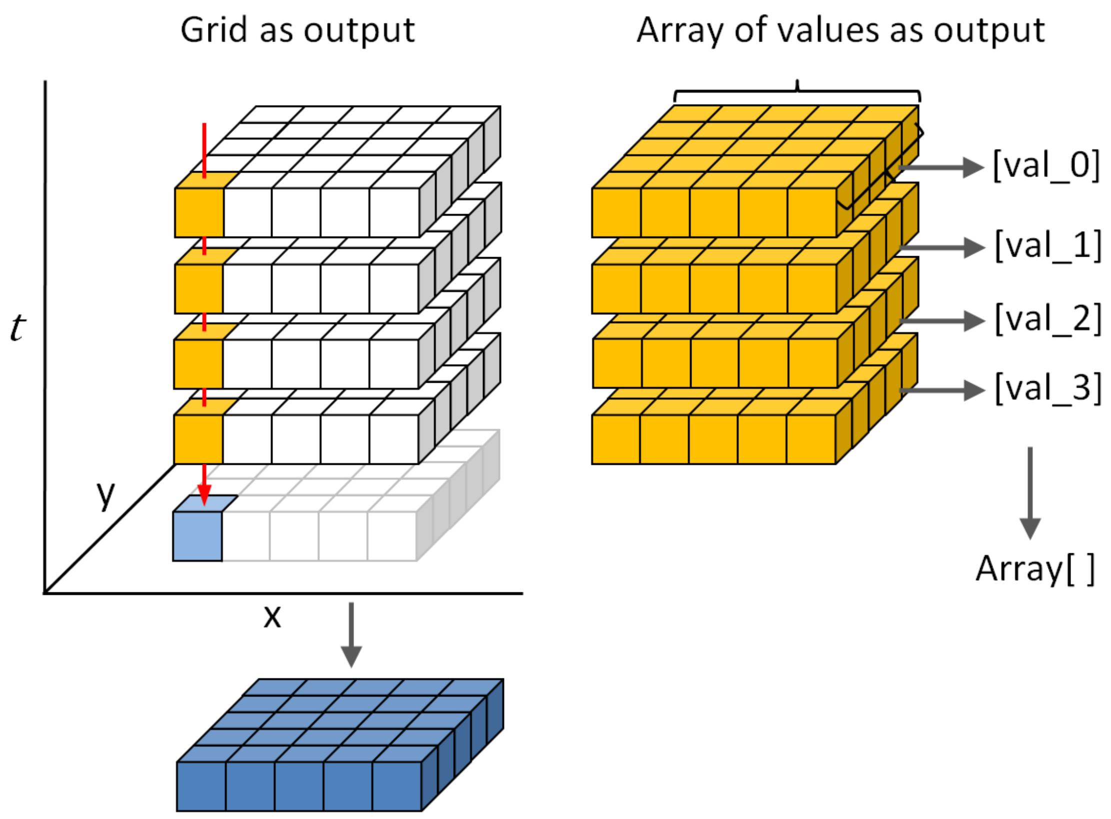


Figure 4.3: The "temporal first" (grid as output) and "spatial first" (array of values as output) approaches to area-based calculations implemented in the GridServer. In the first method, an output grid is produced containing the map algebra value calculated for each grid cell. In the second, the output array contains a single value for each grid, representing the statistic calculated across all grid cells in that grid.

In the GridStats web interface, the output grid produced by the "temporal first" method is automatically displayed as an output layer in the dynamic mapping tool. The GridServer's Python code uses masked NumPy arrays to perform these calculations. The following code sample is an abbreviated version of the function that calculates all temporal grid statistics, using mean as an example (Appendix I contains additional Python code related to this approach):

```
1    def get_temporal_grid(self, data):
2      fill_val = -9999.
3      mdata = numpy.ma.masked_array(data, numpy.isnan(data))
4      grid = numpy.ma.mean(mdata, axis=0)
5      outgrid = grid.filled(fill_val)
6      return outgrid
```

Explanation of this code sample:

Line 1: The Python function is defined. The `data` argument represents a NumPy ndarray containing the input grid range selected by the user.

Line 2: The `fill_val` variable is set to the value that will be used to fill any "no data" grid cells (e.g., in the ocean) that happen to exist in the output grid. The standard PRISM "no data" value is -9999, which is used for both masking operations and map classification purposes.

Line 3: A masked NumPy array is created from the input grids in the `data` ndarray. Masked NumPy arrays set any "no data" values designated as `nan` in the input grids to a special value that the NumPy module ignores during processing. This allows any "no data" areas to be excluded from statistical calculations.

Line 4: The NumPy function `numpy.ma.mean` is called to calculate the mean on the masked ndarray produced in the previous line. The important `axis=0` argument causes the function to calculate the mean statistic through columns of the ndarray (i.e., performing map algebra by calculating vertically through aligned grid cells). This produces a 2-dimensional array containing the data representing the output grid, which is stored in the `grid` variable.

Line 5: Any "no data" values in the array are filled using the fill value defined in line 1, and stored in the `outgrid` variable.

Line 6: The output grid is returned to the calling function. Once returned, the 2D array is converted to the binary BIL grid format and written to disk as a map-ready output grid.

The second area-based calculation performed by the GridServer is "spatial first," in which the requested statistic is calculated over space, producing a single output statistic value for each input grid. In contrast to the "temporal first" approach of calculating *through* all aligned grid cells over time, this approach calculates *across* all grid cells over space, in effect calculating an overall areal statistic for each input grid (refer to previous Figure 4.3). The resulting output statistic values are collected and returned to the GridStats website, where they are plotted on an interactive chart, allowing the user to visualize spatial trends in the selected region over time. As with the "temporal first" approach, the GridServer's Python code uses masked NumPy arrays to perform this type of calculation. The following code sample is an abbreviated version of the function that calculates all spatial grid statistics, using mean as an example (Appendix I also contains additional Python code related to this approach):

```
1    def get_spatial_values(self, data):
2      rnd = 2
3      spatial_vals = []
4      mdata = numpy.ma.masked_array(data, numpy.isnan(data))
5      spatial_vals.append(round(numpy.ma.mean(data), rnd))
6      return spatial_vals
```

Explanation of this code sample:

Line 1: The Python function is defined. The `data` argument represents a NumPy ndarray containing the input grid range selected by the user.

Line 2: A value is assigned to a variable used for rounding the final output values. In the full Python function this variable is set conditionally based on the climate parameter being processed (precipitation is calculated to two decimal places, while temperature variables are calculated to one decimal place).

Line 3: The `spatial_vals` list is initialized to hold the final spatial output values, which will be indexed identically to the `data` ndarray holding the input grid data.

Line 4: As previously described in the "temporal first" code explanation, a masked NumPy array is created from the input grids in the `data` ndarray.

Line 5: The NumPy function `numpy.ma.mean` is called to calculate the mean on the masked ndarray produced in the previous line. Note that in contrast to the mean calculation in the "temporal first" code explanation, this function call does not define an `axis` argument. By excluding the `axis` argument, the `numpy.ma.mean` function calculates the mean across all elements in the flattened array (i.e., all grid cells across a single input grid). This approach produces a single output statistic value for each processed grid (the full Python function loops over all input grids in the requested set, calculating the output statistic for each grid). As each value is calculated, it is appended to the `spatial_vals` list.

Line 6: The `spatial_vals` list is returned to the calling function. The data are packaged as part of the JSON return and dynamically plotted on the time-series chart on the GridStats results web page.

Figures displaying the area-based output data rendered on maps and plotted in charts are provided in section 4.8.2.

4.4.7.1.3 Calculating Univariate Statistics in Python

By making requests to the GridServer, the GridStats system can calculate a number of univariate statistics (listed previously in Table 4.5) that are used for reporting and charting (point-based requests), and to calculate output grids representing the statistic of interest (area-based requests). Sums are currently calculated only for precipitation-related point requests, and are performed manually by simply summing the values of the individual aligned grid cells over the requested time interval. The basic min and max statistics are calculated for point requests by passing NumPy arrays into the Python-based functions `numpy.min` and `numpy.max`, respectively. For area-based requests involving masked values, the `numpy.ma.min` and `numpy.ma.max` functions are used. Range (the difference between min and max) is also calculated only for point-based requests, but is available for both precipitation and temperature climate parameters.

The mean of the input data is calculated via `numpy.mean` (for point requests) or `numpy.ma.mean` (for area requests), using Formula 4.1.

$$\bar{x} = \frac{\sum x}{n} \qquad 4.1$$

Standard deviation is also calculated using a NumPy function. All standard deviation requests (whether point- or area-based) calculate sample standard deviation using `numpy.std` (or `numpy.ma.std` for masked array operations). PRISM climate grids are modeled using data obtained from irregular station networks (i.e., a sample across the CONUS), making sample standard deviation the correct statistic vs. population standard deviation. An example function call looks like the following statement, where `data` is a masked NumPy ndarray:

```
output = numpy.ma.std(data, ddof=1)
```

The `ddof=1` statement instructs `numpy.ma.std` to use `n-1` degrees of freedom, thereby performing the sample standard deviation calculation (Formula 4.2).

$$s = \sqrt{\frac{\sum(x - \bar{x})^2}{n-1}} \qquad 4.2$$

For point-based requests that calculate sample standard deviation, the standard error of the mean is also calculated. This operation uses the `scipy.sem` function, as standard error is only available as a built-in function in the SciPy package, and calculates the statistic using Formula 4.3. The function uses `n-1` degrees of freedom by default, so no explicit `ddof` argument must be given.

$$SE_{\bar{x}} = \frac{s}{\sqrt{n}} \qquad 4.3$$

4.4.7.1.4 Calculating PCA in Python

Principal Components Analysis is the sole multivariate statistical technique currently available in the GridStats system. This method was implemented for two reasons: (1) to identify and produce the principal components (as output grids) that explain the preponderance of variance in a set of input grids meaningful to the user based on climate parameter and spatial and temporal scales, and (2) to serve as a test method for implementing multivariate techniques in the GridServer and the GridStats output visualization tools.

The heavily-commented Python function developed for calculating PCA is included in full in Appendix J, with key steps described here:

1. As described in prior sections, a NumPy ndarray holds the input grids and is converted to a masked array for processing.

2. The initial ndarray is 3D (each grid's cells are stored as subarrays of rows and columns). It is flattened to 2D (each grid's cells are then all stored in a single array, indexed to retain proper ordering) to prepare for further processing.
3. The data are mean-centered (i.e., normalized by the mean). This occurs across arrays and through columns.
4. The data are scaled by standard deviation across arrays, through columns.
5. The diagonalized covariance matrix of the mean-centered, scaled data is calculated across all subarrays. This is possible since the arrays were flattened in step 2.
6. Using the NumPy linear algebra module, the eigenvalues and eigenvectors of the covariance matrix are calculated.
7. The eigenvalue and eigenvector arrays are index-matched to ensure they remain in the correct ordering associated to one another. The eigenvalue array is sorted in decreasing order, and the eigenvector array is then sorted to match.
8. The eigenvalues are converted to percentages of explained variance and are stored in a list.
9. The principal components corresponding to each eigenvector are calculated using the formula described in Appendix J. In short, the eigenvectors from the first column of the eigenvector array are calculated against the first set of input grids. This is repeated across the columns of the eigenvector array. This formula is identical to the GRASS GIS approach used in the *r.covar* and *r.pca* modules (Neteler and Mitasova 2002, GRASS 2012).
10. The 2D output array containing the component arrays is returned to the calling function, along with the indexed list of eigenvalues expressed as percentages. The component arrays are processed into binary BIL grids, with one grid representing each component. The eigenvalue percentages are used to list the variance explained by each component grid.

After processing is complete, the set of component grids is immediately available in the GridStats output web page, implemented as a selectable list that automatically loads a grid into the dynamic output mapping tool. Visualization of the principal components as grids is of questionable benefit, although grid cell values do have meaning relative to one another. Each grid cell within a component grid contains a principal component score, which can be useful in interpreting the relative within-component contribution of each grid cell to the variance of the input grid set explained by that component. Grid cell values can be queried in the output mapping tool. The set of component grids can also be exported as a .zip file and used as inputs in further analysis.

4.4.7.2 *Data Output Products*

Point-based (i.e., single grid cell) and area-based requests are handled by the GridServer, while requests spanning the entire CONUS grid extent are handled by the Biltools toolkit. Table 4.6 lists the request types and related output products.

Table 4.6: GridStats spatial scale request types and the possible data output products generated by each request type.

	Data Output Product		
Spatial Scale Request Type	**BIL grid (full extent)**	**BIL grid (partial extent)**	**Data values as text (JSON)**
Point (single grid cell)			X
Rectangular bounding box		X	X*
Area clipped by polygon boundary		X	X*
Full grid extent (CONUS)	X		

* Data are returned as the univariate statistic output value calculated across all grid cells in the requested region (one data value per input grid).

Each BIL output grid (for both full and partial extents) consists of a set of five files: A binary `.bil` file plus matching `.hdr`, `.prj`, `.stx`, and `.aux.xml` files. PRISM BIL grids all use the following standard `.hdr` format (the example below represents a clipped region):

```
BYTEORDER      I
LAYOUT         BIL
NROWS          269
NCOLS          137
NBANDS         1
NBITS          32
BANDROWBYTES   548
TOTALROWBYTES  548
PIXELTYPE      FLOAT
ULXMAP         -122.433333334365
ULYMAP         43.566666658865
XDIM           0.00833333333
YDIM           0.00833333333
NODATA         -9999
```

Point (single grid cell) and area methods return data values as text (JSON format). In the point method, two data arrays are embedded in the JSON return: (1) the set of actual values for the selected climate parameter, extracted from the matching grid cell across all requested time-series grids, and (2) a set of summary statistic values (sum [precipitation only], range, min, max, mean, median, standard deviation, standard error) calculated across the set of data values in (1).

The GridServer area methods embed one data array in the JSON return – the set of univariate statistic output values calculated across all grid cells in the requested region. The statistic is calculated for each input grid by processing all grid cells in the requested region, such that one data value per input grid is returned. The JSON return also includes a filesystem pointer to the output BIL grid produced via the "temporal first" calculation where the requested statistic was calculated through each grid cell, producing an output grid representing the statistic calculated over the requested time series.

4.4.8 Website Design, Client-side Tools, and User Interaction

The GridStats website uses a range of JavaScript-based client-side software tools that perform a number of functions, including:

- enhance user interaction with the website
- validate menu selections and notify user of any issues prior to submission
- dynamically load menu options based on a selection in another menu
- communicate with the server to query data and update the user's view
- constrain selection options based on time series limitations and spatial scale

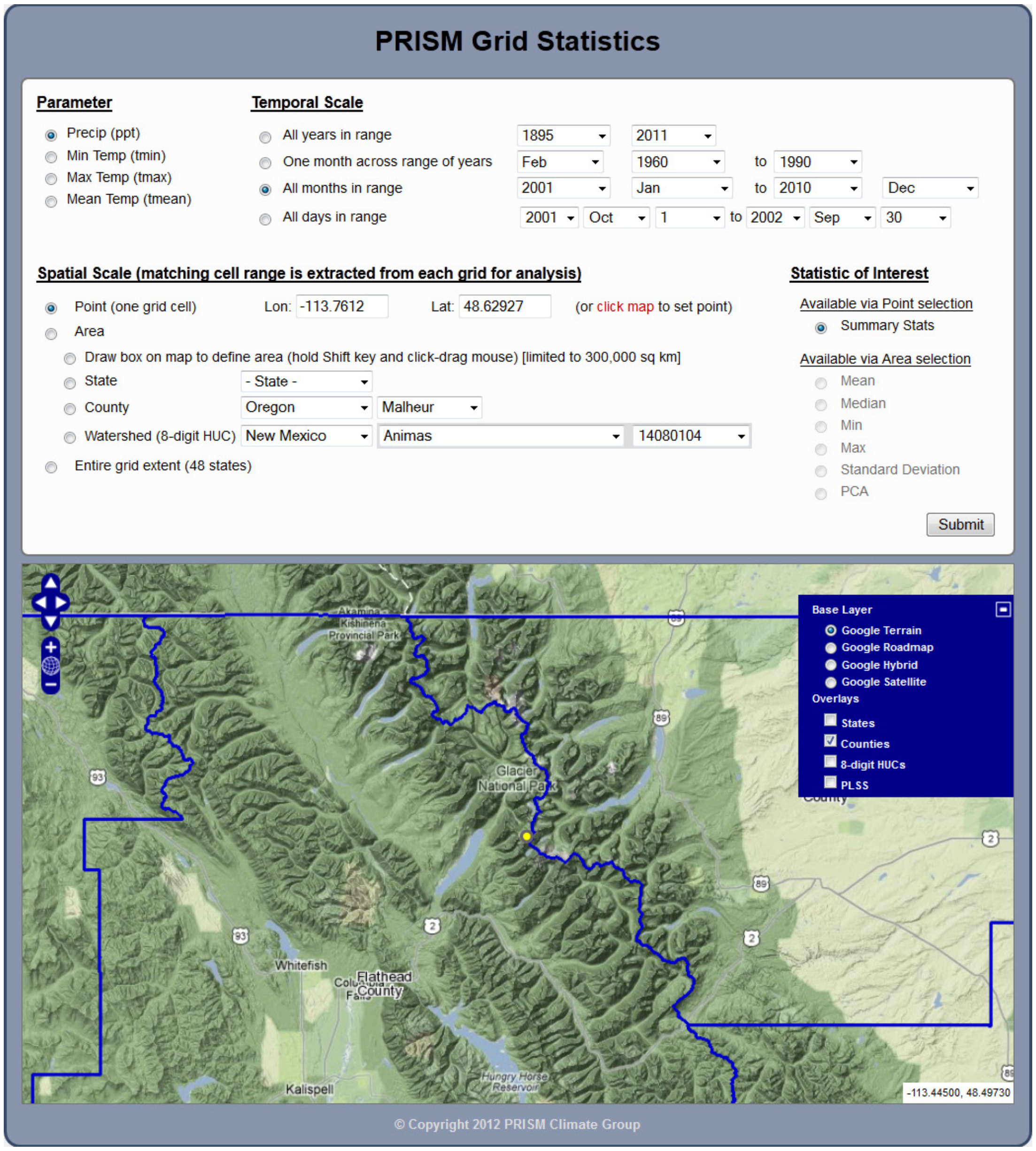
- set key variables sent to the server on submission
- provide dynamic, interactive mapping and charting tools on results pages

4.4.8.1 *Input Settings*

The settings page of the GridStats website (Figure 4.4) gives the user control over all spatial and temporal scale settings, location of interest, and selection of the desired climate parameter and statistic of interest. Selections are made primarily via a series of radio buttons and dropdown menus (Figure 4.5), although point coordinates may be entered directly into text entry boxes. The values (and in some cases, availability) of the selectors depend upon selections made in other areas of the web page (i.e., Summary Stats are only available for point-based requests). That logic is controlled on the client-side using JavaScript-based code. All parameter settings are saved on the user's computer using cookies and reloaded the next time they visit the website (or even when they simply click the browser's Back button). This makes it easy for the user to switch to a different data time series, for example, while keeping all other parameter settings constant.

4.4.8.1.1 Menu Selections and Validation

The available climate parameters are largely an independent set of selections, with the exception that mean temperature data are only available on monthly and annual time scales. When a user clicks the radio button next to "Mean Temp (tmean)," the



PRISM Grid Statistics

Parameter

[✓ Precip (ppt)]
Min Temp (tmin)
Max Temp (tmax)
Mean Temp (tmean)

Temporal Scale

All years in range	1895	2011						
One month across range of years	Feb	1960	to	1990				
[✓ All months in range]	2001	Jan	to	2010	Dec			
All days in range	2001	Oct	1	to	2002	Sep	30	

Spatial Scale (matching cell range is extracted from each grid for analysis)

[✓ Point (one grid cell)] Lon: -113.7612 Lat: 48.62927 (or click map to set point)

Area

Draw box on map to define area (hold Shift key and click-drag mouse) [limited to 300,000 sq km]

State: - State -

County: Oregon, Malheur

Watershed (8-digit HUC): New Mexico, Animas, 14080104

Entire grid extent (48 states)

Statistic of Interest

Available via Point selection

[✓ Summary Stats]

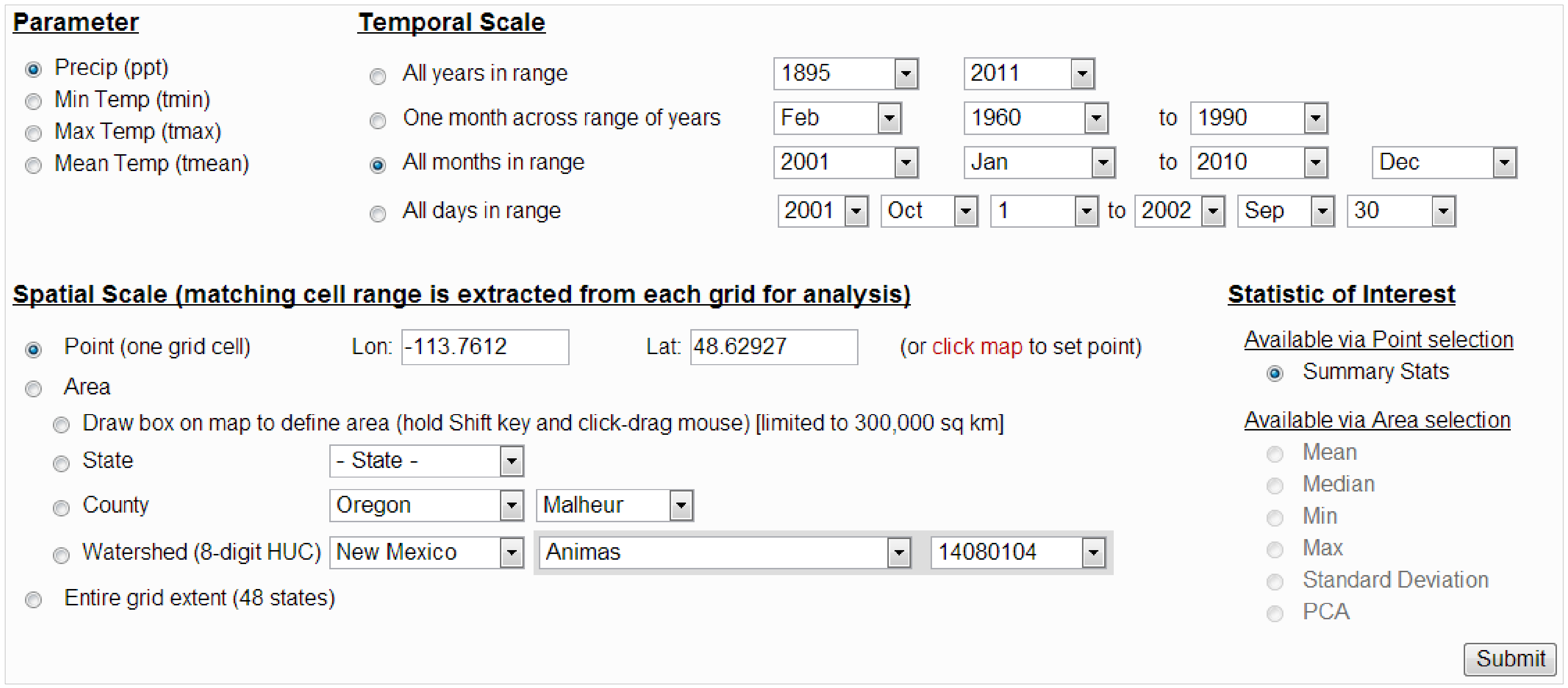
Available via Area selection

Mean
Median
Min
Max
Standard Deviation
PCA

Submit

© Copyright 2012 PRISM Climate Group

Figure 4.4: The GridStats settings page, ready for request submission. A (yellow) point has been selected at the center of the image, in Glacier National Park. The selected climate parameter is precipitation, and the selected temporal scale is all months in the range January 2001 to December 2010. “Summary Stats” is selected as the default statistic of interest, since this is a point-based request. A larger version of the settings section is included in Figure 4.5.


Parameter

- [✓ Precip (ppt)]
- Min Temp (tmin)
- Max Temp (tmax)
- Mean Temp (tmean)

Temporal Scale

All years in range	1895	2011						
One month across range of years	Feb	1960	to	1990				
[✓ All months in range]	2001	Jan	to	2010	Dec			
All days in range	2001	Oct	1	to	2002	Sep	30	

Spatial Scale (matching cell range is extracted from each grid for analysis)

- [✓ Point (one grid cell)] Lon: -113.7612 Lat: 48.62927 (or click map to set point)
- Area
  - Draw box on map to define area (hold Shift key and click-drag mouse) [limited to 300,000 sq km]
  - State: - State -
  - County: Oregon, Malheur
  - Watershed (8-digit HUC): New Mexico, Animas, 14080104
- Entire grid extent (48 states)

Statistic of Interest

Available via Point selection

- [✓ Summary Stats]

Available via Area selection

- Mean
- Median
- Min
- Max
- Standard Deviation
- PCA

Figure 4.5: Settings section of the main GridStats web page. Spatial scale menus are dynamic, and zoom the map directly to the selected polygon when a selection is made. Extensive client-side validation code checks selected values prior to submission. In this example, the area-based statistic selections are currently disabled since the "Point" method is selected (in that case, Summary Stats is selected by default).

"All days in range" row in the Temporal Scale section is automatically disabled. Clicking any of the other climate parameter radio buttons causes the daily data selectors to be automatically re-enabled. Although it is possible to achieve this functionality by writing the controls directly in low-level JavaScript, it can be handled in a much more logical and straightforward manner using the jQuery library (jQuery 2012). Every selector on the settings page contains a unique identifier (ID) – jQuery allows the manipulation of any object's properties and attributes by obtaining a handle on the object's ID and manipulating the document object model (DOM) to make the necessary change in the user's browser. This approach is used throughout the GridStats website. jQuery statements to disable and enable a selector look like these examples, taken from the JavaScript code for the settings page:

```
$('#days_in_rng').attr('disabled', 'disabled');
$('#days_in_rng').removeAttr('disabled');
```

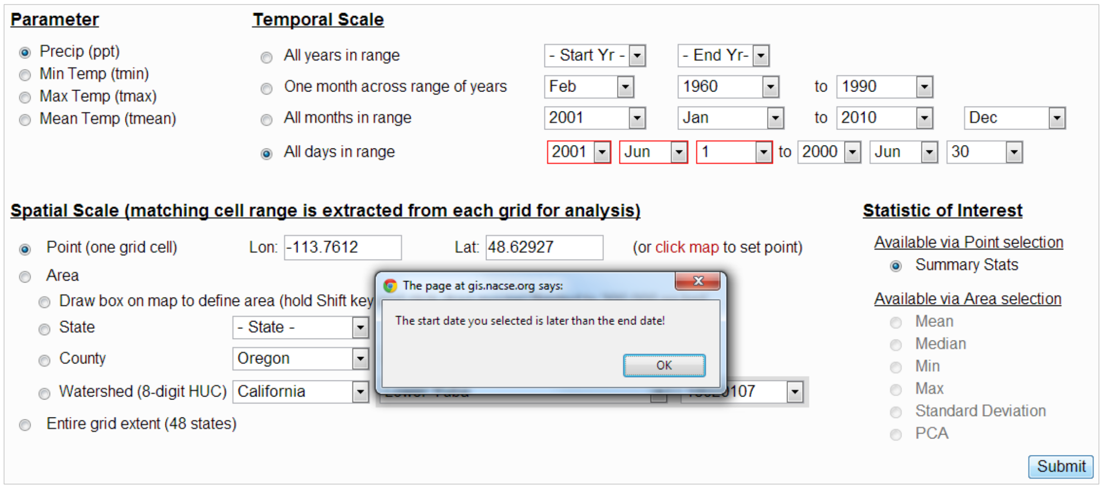
Additional JavaScript and jQuery code samples are included in Appendix K.

In the Temporal Scale section, all of the dropdown menus are filled with years, months, and days by PHP functions when the page loads. The menu-filling functions determine any date limits based on internal settings read from a configuration file where those limits are defined. For example, daily precipitation data are currently available from 1980 onward, while daily temperature data are available from 1970 onward. When the user selects "Precip" as the climate parameter, the start year and end year dropdown menus in the "All days in range" row are automatically modified to disable the years 1970-1979 as selections. This is handled via custom jQuery code, using the `.change()` handler that fires events (e.g., a function call to the code that disables years 1970-1979) whenever a user makes a selection on the menu of interest.

Date handling and validation in the settings page is extensive and complicated, since many possible cases must be accounted for in the logic. Some of the cases that are addressed using jQuery and custom date comparison functions include:

- If the current year is selected in a monthly-related temporal scale, the current month and any future months in the current year must be disabled.
- If the current month is August and the user has already selected “Nov” as the month of interest, and the user then selects the current year, the month must snap to the last available month, which would be “Jul” in this case.
- The “day” menus range 1-31 and must disable any day options that do not exist in the currently selected month (e.g., if “Jun” is selected, the day “31” must be disabled).
- If the currently selected year is a leap year, and the currently selected month is “Feb,” then day 29 must be made available as a selection, while it is disabled for Feb in non-leap years.
- If the selected start year (or year/month, or year/month/day) is later than the end year (or year/month, or year/month/day), the user must be alerted so that they can fix the mistake prior to submission.

An example of a date validation alert is displayed in Figure 4.6. The website code handles known date limitations (e.g., no day 31 in June) by disabling relevant options using jQuery. Otherwise, the user has full control over menu selections, and any mistakes (e.g., start year = 2010, end year = 2002) are caught by the validation framework code. This level of structured client-side validation captures any request configuration issues on the client-side prior to submission, greatly limiting errors on the server-side, which can cause problems that are more difficult to report to the user and potentially resource-intensive for the processing server.


Parameter
- ● Precip (ppt)
- ○ Min Temp (tmin)
- ○ Max Temp (tmax)
- ○ Mean Temp (tmean)

Temporal Scale
- ○ All years in range: - Start Yr - / - End Yr-
- ○ One month across range of years: Feb, 1960 to 1990
- ○ All months in range: 2001, Jan to 2010, Dec
- ● All days in range: 2001, Jun, 1 to 2000, Jun, 30

Spatial Scale (matching cell range is extracted from each grid for analysis)
- ● Point (one grid cell) Lon: -113.7612 Lat: 48.62927 (or click map to set point)
- ○ Area
  - ○ Draw box on map to define area (hold Shift key
  - ○ State: - State -
  - ○ County: Oregon
  - ○ Watershed (8-digit HUC): California; 0107
- ○ Entire grid extent (48 states)

Statistic of Interest

Available via Point selection
- ● Summary Stats

Available via Area selection
- ○ Mean
- ○ Median
- ○ Min
- ○ Max
- ○ Standard Deviation
- ○ PCA

Submit

The page at gis.nacse.org says:

The start date you selected is later than the end date!

OK

Figure 4.6: Example of a date validation alert where, for “All days in range,” the user has selected a start date that is later than the selected end date. The validation code will catch any such error and continue to alert the user until a valid selection is made.

4.4.8.1.2   Map-based Selection

The interactive mapping tool on the input settings page is driven by custom code based around the open source OpenLayers JavaScript API. The map includes the four standard Google Maps base layers (set to Terrain by default), and allows the user to navigate to any location in the CONUS. A location of interest smaller than the full CONUS extent can be selected by (1) clicking on the map to identify a point (previously shown in Figure 4.4), (2) entering longitude and latitude coordinates directly, or (3) selecting a region of interest using one of four available methods:

- Drawing a box on the map to denote a rectangular region of interest.
- Selecting a state from the CONUS.
- Selecting a county within a CONUS state.
- Selecting a watershed (8-digit USGS HUC) within a CONUS state.

The OpenLayers-based map interprets a mouse click-drag event as a panning movement (e.g., panning the current view from east to west). If the "Draw box on map" radio button is selected, the user can select a bounding box while holding down the Shift key and performing a click-drag event (Figure 4.7). The drawn box remains on the map and dynamically calculates the estimated area of the box in km^2, which is printed inside the box. A different box can be drawn anywhere, erasing the previously-drawn box, and clicking to any other spatial scale selection method automatically erases the box from the map view (as well as from the behind-the-scenes code setting the bounding box coordinates).

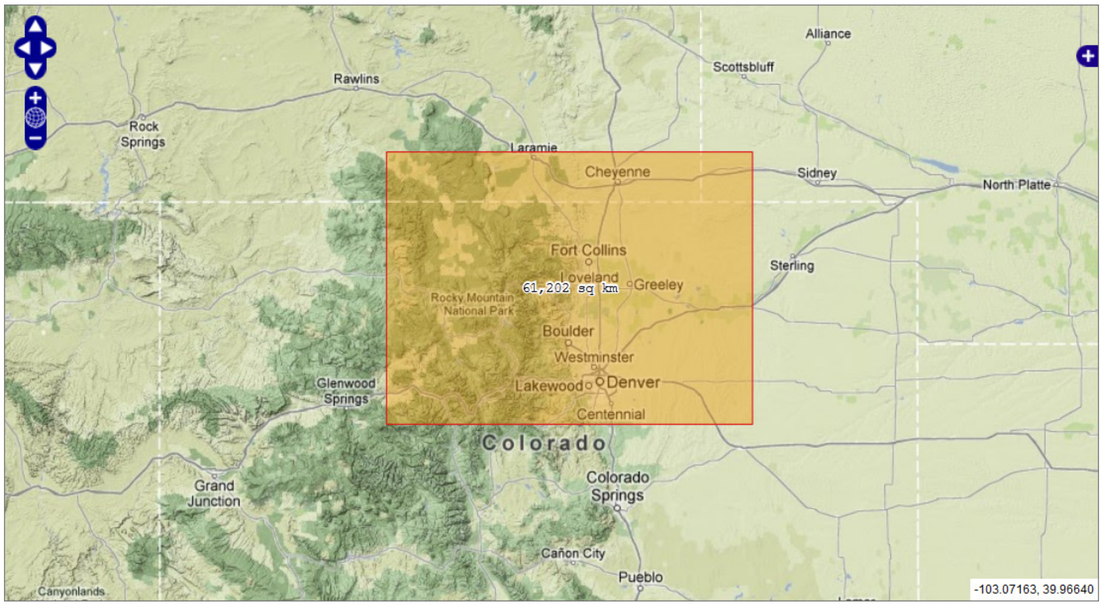


Figure 4.7: Example of a bounding box selection drawn on the input settings page mapping tool. The box remains on the map after it is drawn (until erased by another selection), and contains the calculated area for the bounded region (64,812 km^2 in this example).

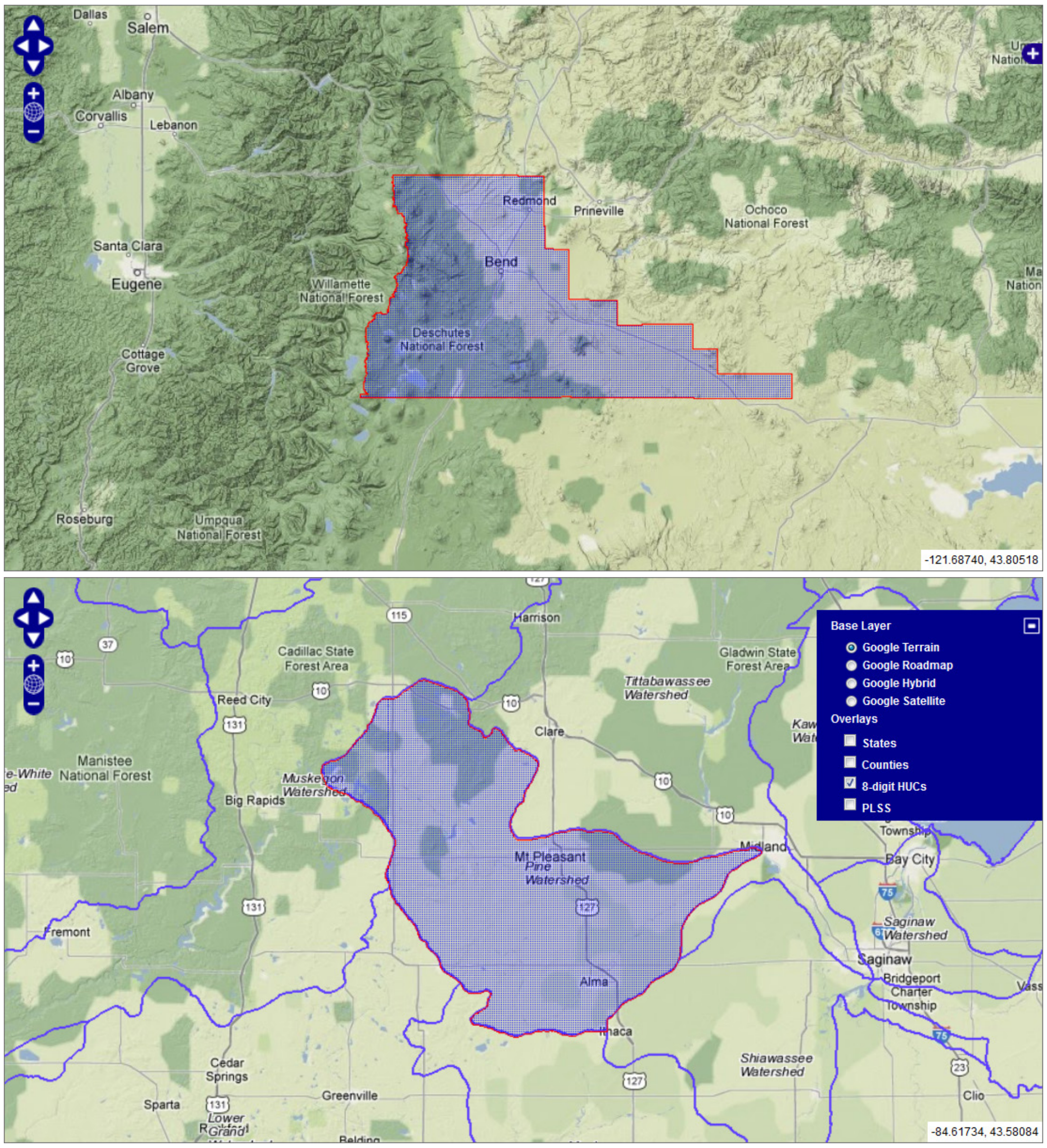
The code that enables the selection of a region based on a polygon is more complex than for other selection types. When a selection is made in the “State” menu, for instance, the map zooms directly to the selected state, displays it as an outlined and shaded polygon, and sets information in the background to prepare the related GridServer request. This is accomplished using a series of client-side and server-side requests (the latter performed asynchronously via an AJAX approach):

1. Selection information is sent to a JavaScript function that processes the request and uses the jQuery `$.getJSON()` function to make a server-side request to a PHP script.

2. The PHP script determines the request type, calls the proper database query function, queries the database to retrieve the geometry for the requested polygon, and converts the geometry to a JSON-encoded text string.
3. The `$.getJSON()` function on the client-side receives the JSON-encoded data, sets layer visibility on the map, zooms the map to the extent of the polygon, and renders the polygon on the map.
4. Client-side code also assigns the table name and unique feature ID to a set of inputs used to prepare the GridServer request.

The County and Watershed menus require an extra layer of asynchronous queries. When a state is selected in the first menu, the adjacent menu is updated dynamically via an AJAX database query (i.e., when Georgia is selected, only the counties that exist in Georgia are displayed in the second menu). The same approach is used for the Watershed menu, with the added complexity that the watershed name and HUC code menus are linked – when a selection is made in one of them, the other menu updates to match. This allows the user to select a watershed either by name or by HUC code. Figure 4.8 displays two examples of selected polygons. Any watershed polygon that intersects with the selected state boundary is included in the dynamically-built menu list, using a spatial database query to determine the intersecting polygons (`$state` is passed in to the query as a variable):

```
SELECT DISTINCT h.gid, h.huc_name, h.huc_code FROM gis.huc250k h,
gis.states s WHERE s.state_name = '$state' AND
ST_Intersects(s.the_geom, h.the_geom) ORDER BY h.huc_name,
h.huc_code;
```


Base Layer

[✓ Google Terrain]

Google Roadmap

Google Hybrid

Google Satellite

Overlays

☐ States

☐ Counties

☑ 8-digit HUCs

☐ PLSS

Figure 4.8: Examples of polygon selection on the input settings page mapping tool. Top: Deschutes County, Oregon is selected via the "County" menu set. Bottom: The Pine Watershed (HUC #04080202) in Michigan is selected via the "Watershed" menu set. The bottom image also displays the layer switcher, in which the nationwide 8-digit HUC layer is toggled for display as a spatial reference aid, and the rest of the available feature layers and Google Maps base layers are visible.

4.4.8.2 *Results Pages*

After submitting a request on the settings page, the GridServer (or Biltools) returns the output data to a single results page. Depending on the request type, different sets of PHP code are dynamically included in the results page to render four different result page types: (1) point, (2) univariate area, (3) multivariate (PCA) area, (4) entire grid. Point results include a statistics and input settings report, plus a dynamically-generated interactive chart. Area results include split, synchronized dynamic maps containing rendered, classified input and output grids, as well as an interactive chart. Entire grid results include split maps as in the area results, but do not include an interactive chart.

4.4.8.2.1 Dynamic Charting

The point-based and area-based results pages include a dynamically-generated interactive chart. The charts are built from server-side result data passed to client-side code, using the open source Highcharts JavaScript library (Highcharts 2012). Highcharts is highly configurable and flexible and supports a wide range of chart types, many of which can be combined as needed into a single chart. In the GridStats system, a mix of line, spline, and column chart types are used, depending upon the climate parameter and data series type.

In addition to the data points, basic statistics are plotted on each chart as aids in understanding the data. The standard deviation of the plotted data is drawn as a solid line, while dashed lines representing ±1 standard deviation (SD) are plotted above and below the mean line. Figure 4.9 displays an example chart generated for a point in Glacier National Park, MT. The chart contains minimum temperature time-series data points plotted as a spline for the months January 2001 to December 2010. The chart is fully zoomable along the x-axis, giving the user the ability to view a section of a dense chart in more detail. Data values for each point can be viewed by hovering over the chart with the mouse cursor. The legend is interactive, allowing the user to toggle line visibility on a per-data-series basis, and collapsible, giving the user the ability to view the entire chart area if any data points happen to be obscured by the legend. When toggling a data series off, the Highcharts code automatically rescales the y-axis to maximize the vertical spread of data in the chart area. This can also be helpful for gaining a better sense of the shape of a data line (i.e., by turning off all data series except for one, more detail can often be viewed in that remaining series). Figure 4.10 displays a zoomed-in view, with the mean and mean ±1 SD lines toggled off.

Centered moving averages are also plotted on the chart, allowing the user to view a smoothed representation of the data signal over various moving average time windows. Each time series includes two lines representing different centered moving average windows: 10 and 30 for daily data, 5 and 12 for monthly data, and 5 and 10 for annual data.

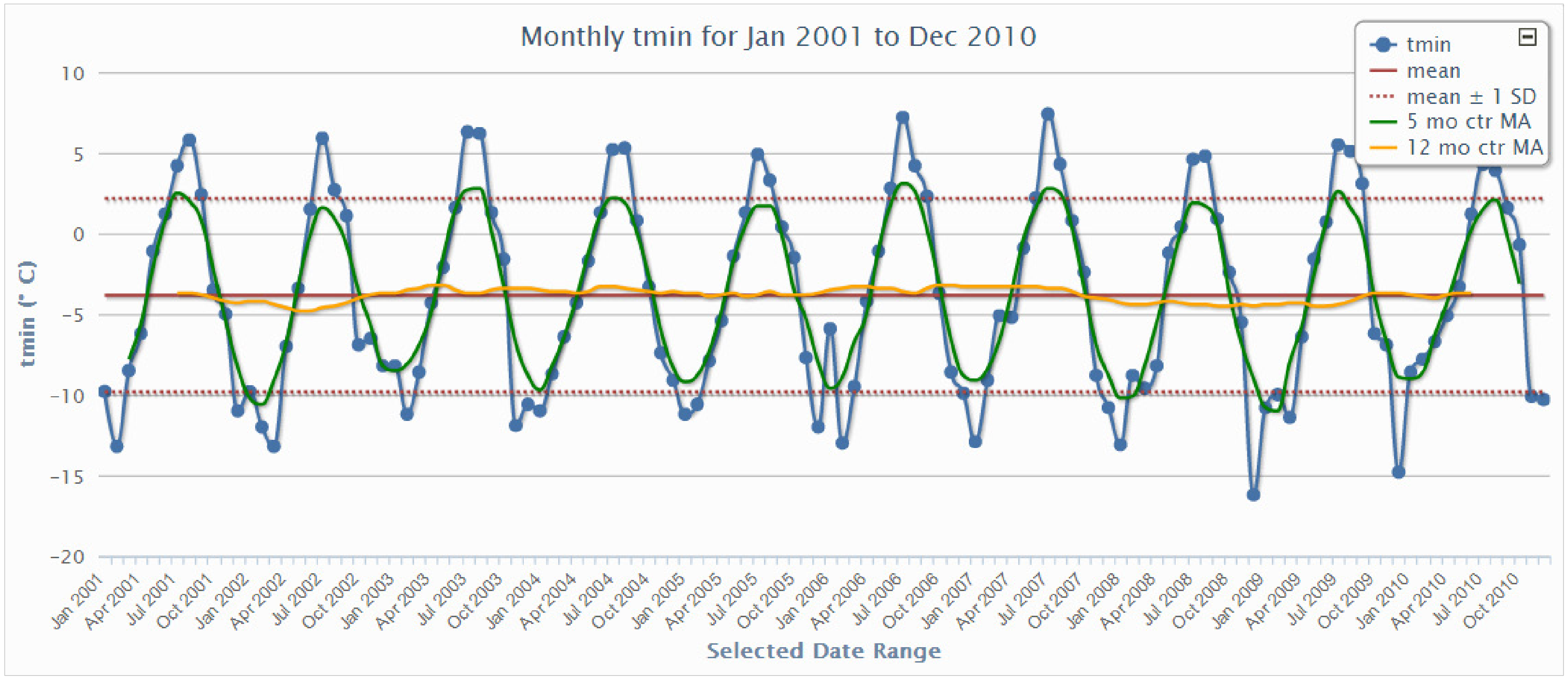


Figure 4.9: Monthly minimum temperature for the inclusive months from January 2001 to December 2010, for a point in Glacier National Park, MT. The user can zoom into any section of the chart, and any data series can be hidden or shown by clicking on the legend item to toggle visibility. The legend is also collapsible to allow the user to view the full chart area.

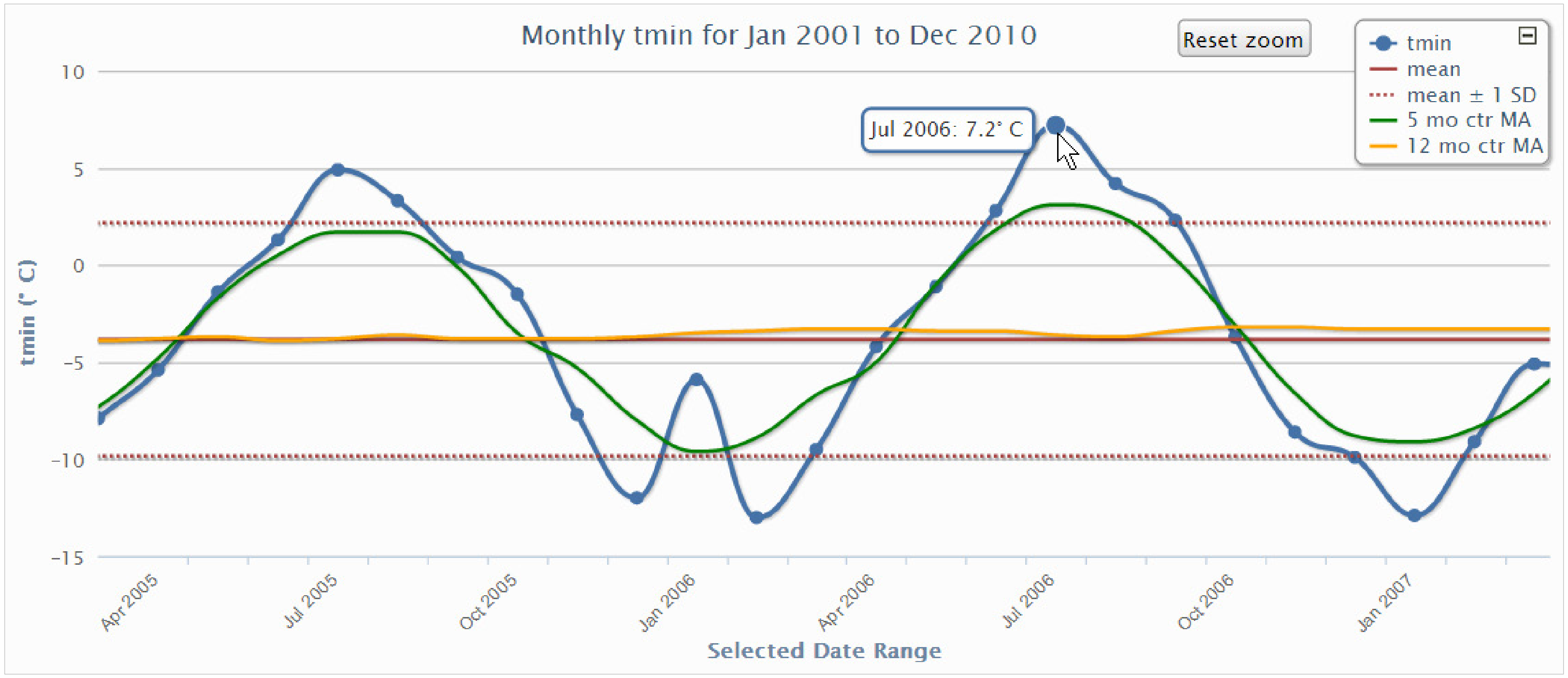


Figure 4.10: A zoomed-in view of the full chart in Figure 4.9. The mean and mean ±1 SD lines have been toggled off in this view. The Reset Zoom button appears any time the chart does not display the full data range. The mouse cursor is hovered over the July 2006 data point, displaying the data value. Hovering over either of the centered moving average lines displays their calculated values at each point.

The centered moving averages are calculated by different methods depending upon the moving average window parity. The simplest case (when the moving average window is an odd number) currently applies only to the size 5 window used for monthly and annual data. The case is relatively simple because the calculation sums the current windowed data value with the two adjacent data values on each side of the data value, and then divides by 5 to calculate the moving average at that point. Formula 4.4 displays the approach used for this particular case.

$$\hat{f}(t) = \frac{(y_{t-2} + y_{t-1} + y_t + y_{t+1} + y_{t+2})}{5} \qquad 4.4$$

Cases where the moving average window is an even number require a more complex method to ensure a centered average for each data point. For example, using a size 4 window, two data points could be used on the trailing side of the window, and one on the other (or vice-versa). Using this approach, the moving average will never be centered over the current windowed data value. To calculate a true centered moving average for an even numbered window, a more efficacious approach involves determining the moving average by calculating both cases; that is, for a size 4 window, use two data points on the trailing side for case one, then two data points on the leading side for case two.

After performing those calculations, the two resulting values are then averaged to produce a centered value at time *t*, as displayed in Formula 4.5 (after Hyndman 2010).

$$\hat{f}(t) = \frac{1}{2}\left[\frac{y_{t-2} + y_{t-1} + y_t + y_{t+1}}{4}\right] + \frac{1}{2}\left[\frac{y_{t-1} + y_t + y_{t+1} + y_{t+2}}{4}\right] \qquad 4.5$$

For the example of a size 4 window, Formula 4.5 reduces to:

$$\hat{f}(t) = \frac{1}{8}y_{t-2} + \frac{1}{4}y_{t-1} + \frac{1}{4}y_t + \frac{1}{4}y_{t+1} + \frac{1}{8}y_{t+2} \qquad 4.6$$

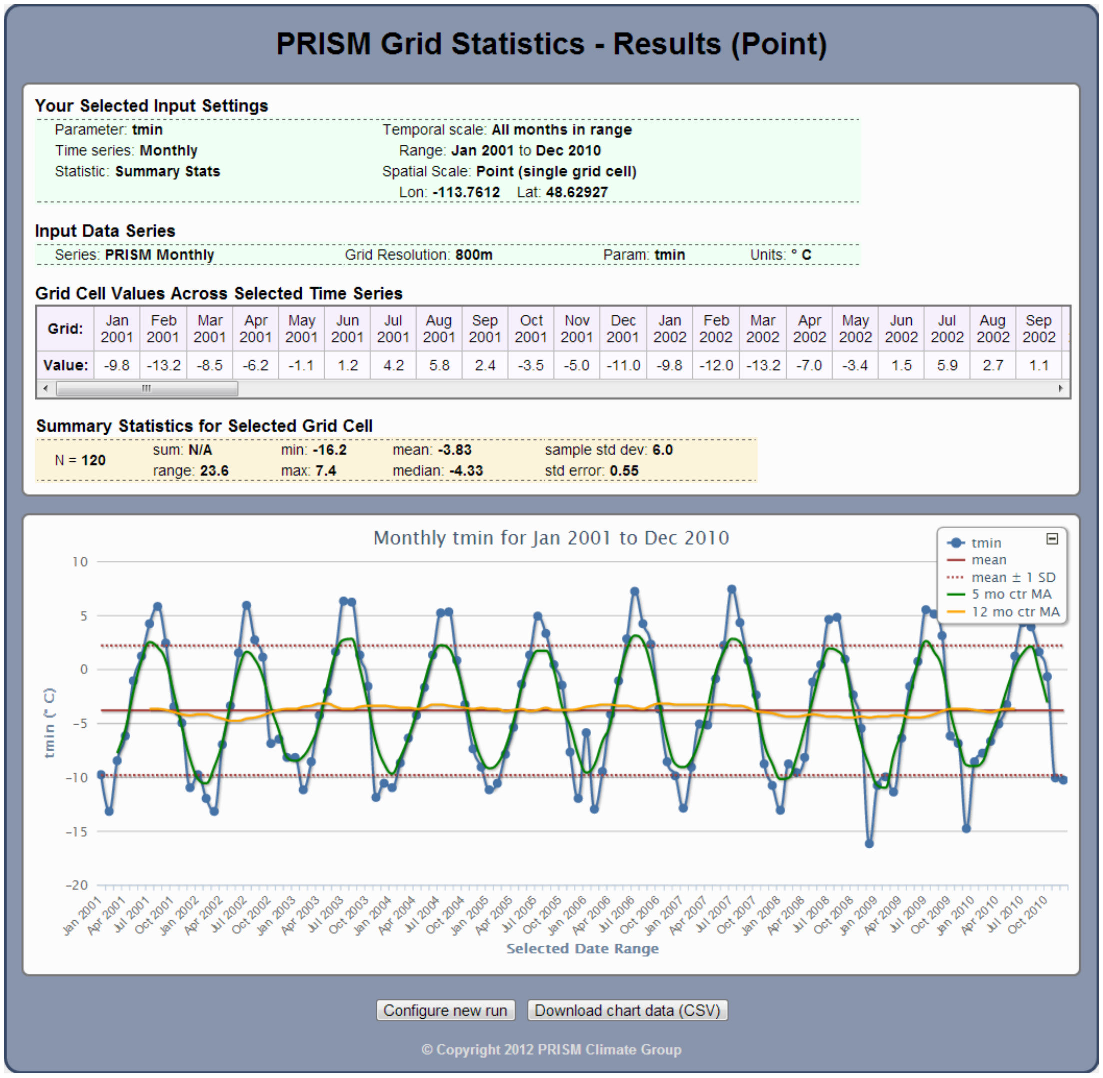
The coefficients in Formula 4.6 vary depending upon the requested window size – for a size 10 window, the inner and endpoint coefficients would become 0.1 and 0.05, respectively, with 11 data values included in the centered moving average calculation. In both even and odd window cases, the moving average is not calculated for the initial and final number of values in the data range equal to half the input window size (i.e., for a size 5 window, moving average values are not calculated for the first two and last two data points in the series, due to a lack of trailing and leading data for the moving window to capture at each end of the series). For odd window cases, a standard floor() function is used to determine the number of unusable values at each end of the data series (e.g., floor(win/2) where win=5 gives a value of 2).

Formulas 4.4 and 4.6 are implemented in the PHP code developed for preparing chart data (included in Appendix L). Once time-series data are returned from a GridServer request, the code processing the output web page sends the data array and requested window size to the centeredMovingAverage() function, which calculates the centered moving average across the temporal chart range. The function handles both even- and odd-window cases, and is therefore called twice to prepare the data used for plotting the two centered moving average lines included in each chart. Once the data are prepared, the arrays are converted to text strings using the PHP implode() function, and passed to JavaScript for Highcharts to render as chart splines.

4.4.8.2.2 Point Results

After the user submits a point-based request and the GridServer returns data from the stack of matching grid cells, GridStats takes the user to the point results page (Figure 4.11). The top area of the page displays the input settings used for submitting the request, as well as information about the data series and selected climate parameter. Returned grid cell values populate a table, which is horizontally-scrollable to accommodate large data returns. The GridServer returns output statistic values for min, max, mean, median, sample standard deviation, and standard error, all of which are included in a summary statistics report below the grid cell values table. The website code sums the number of data values and calculates the data range from the min and max values, also included in the summary statistics report.

The bottom area of the point results page includes the dynamic chart, described in detail in the previous section (4.8.2.1). The returned grid cell values are also used to draw the spline for the main data series on the chart.



PRISM Grid Statistics - Results (Point)

Your Selected Input Settings

Parameter: **tmin**
Time series: **Monthly**
Statistic: **Summary Stats**
Temporal scale: **All months in range**
Range: **Jan 2001** to **Dec 2010**
Spatial Scale: **Point (single grid cell)**
Lon: **-113.7612** Lat: **48.62927**

Input Data Series

Series: **PRISM Monthly** Grid Resolution: **800m** Param: **tmin** Units: **° C**

Grid Cell Values Across Selected Time Series

Grid:	Jan 2001	Feb 2001	Mar 2001	Apr 2001	May 2001	Jun 2001	Jul 2001	Aug 2001	Sep 2001	Oct 2001	Nov 2001	Dec 2001	Jan 2002	Feb 2002	Mar 2002	Apr 2002	May 2002	Jun 2002	Jul 2002	Aug 2002	Sep 2002
Value:	-9.8	-13.2	-8.5	-6.2	-1.1	1.2	4.2	5.8	2.4	-3.5	-5.0	-11.0	-9.8	-12.0	-13.2	-7.0	-3.4	1.5	5.9	2.7	1.1

Summary Statistics for Selected Grid Cell

N = **120** sum: **N/A** range: **23.6** min: **-16.2** max: **7.4** mean: **-3.83** median: **-4.33** sample std dev: **6.0** std error: **0.55**

Figure 4.11: Example results page for a point-based request. Input settings, data series, grid cell values across the requested time series, and a table of summary statistics for those values are all included, in addition to the interactive chart.

The GridServer processes and returns point data requests very quickly – it can calculate point-based summary statistics through 1000 time-series grids and return the data values in less than a second (Table 4.7). For the tests summarized in Table 4.7 the GridServer was restarted between each request, forcing it to drop and reallocate grid file handles to eliminate any possible performance bias due to caching. Random point locations and non-overlapping date ranges were also used (except for the largest request of 10,000 grids, for which a random point was used but overlapping dates were unavoidable). GridServer requests for these tests were issued in this form (with new randomly-selected point coordinates inserted for each request), using the *time* and *wget* command-line utilities to measure query response time:

```
time wget -qO test.txt "http://<server.domain>:<port>/
stats_point_days?lon=-104.48871&lat=38.76029&start=19800101
&end=19800102&param=ppt"
```

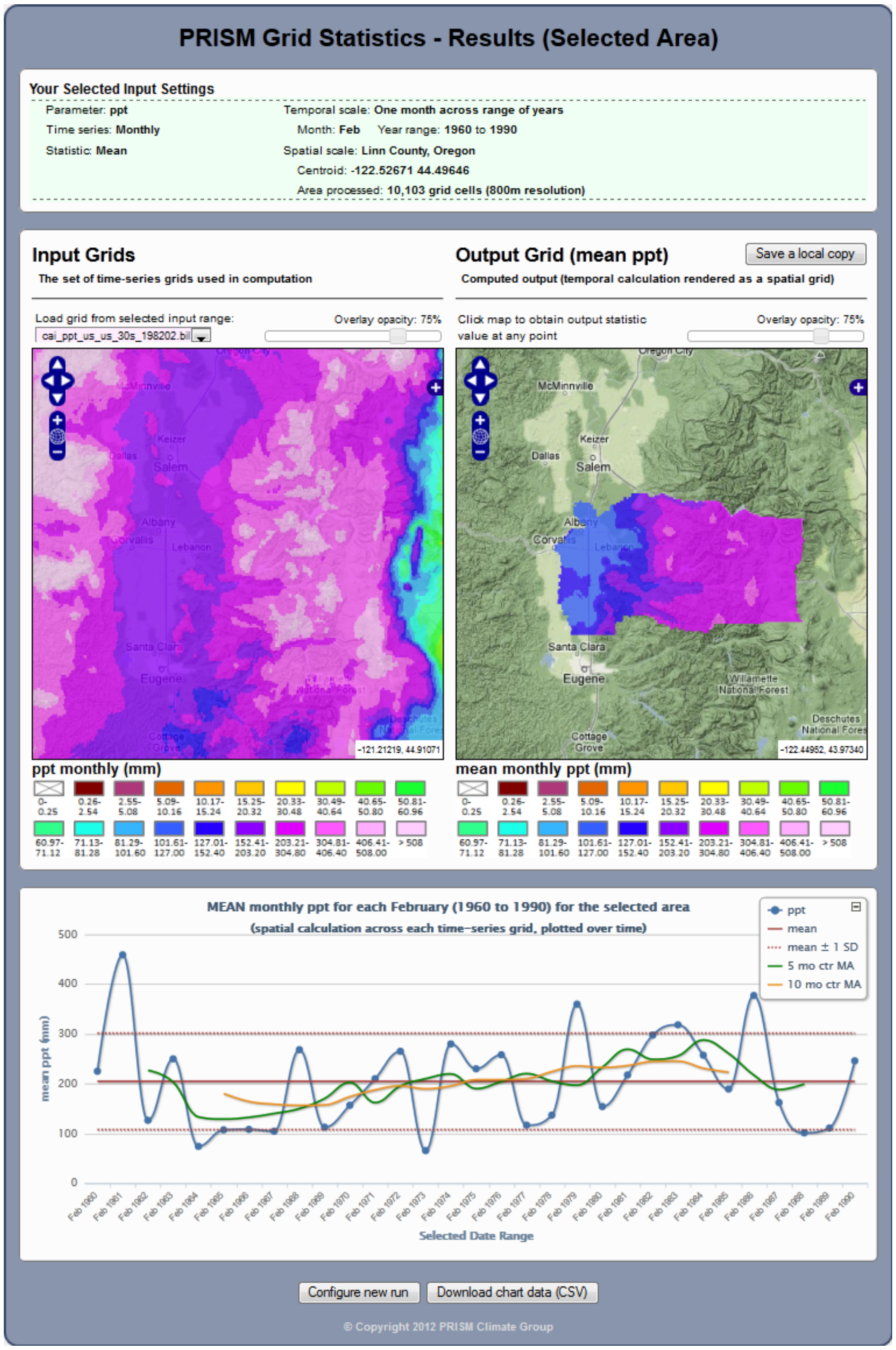
4.4.8.2.3 Area Results

After an area-based request is processed, GridStats takes the user to the area results page (Figure 4.12), which includes different content depending upon the request type (PCA and "entire grid" returns are handled differently than other area-based results). All area result pages contain a report summarizing the input settings and some output information, including the centroid of the selected area (calculated in the spatial database) and the number of cells processed per grid.

Table 4.7: Sample GridServer response times for varying temporal scales and number of grids. Each processed request was point-based (i.e., requesting summary statistics), with precipitation as the selected climate parameter on a daily time scale.

Requested date range (daily ppt data)	**Number of grids in request**	**Initial response time (s)**	**Average subsequent response time (10 at same location) (s)**
1980-01-01 – 1980-01-02	2	0.08	0.06
1980-02-01 – 1980-02-10	10	0.13	0.06
1980-03-01 – 1980-06-08	100	0.19	0.08
1980-07-01 – 1983-03-27	1,000	0.97	0.21
1983-04-01 – 1988-09-20	2,000	1.50	0.38
1988-10-01 – 2002-06-09	5,000	3.84	0.77
1980-01-01 – 2007-05-18	10,000	8.12	1.46

All area result pages also contain a pair of interactive maps, with input grids on the left side and output grids on the right (Figures 4.12, 4.13). When a user is taken to the results page, the grid representing the processed output polygon area is loaded in the Output Grid map window and both maps are automatically zoomed to the extent of that polygon. Each interactive map is a distinct OpenLayers-based map window containing Google Maps base layers, the feature layer set also available in the input settings map, and all other standard controls. However, the maps are synchronized so that all pan, zoom, and query operations that occur in one map window also occur in the other.



PRISM Grid Statistics - Results (Selected Area)

Your Selected Input Settings

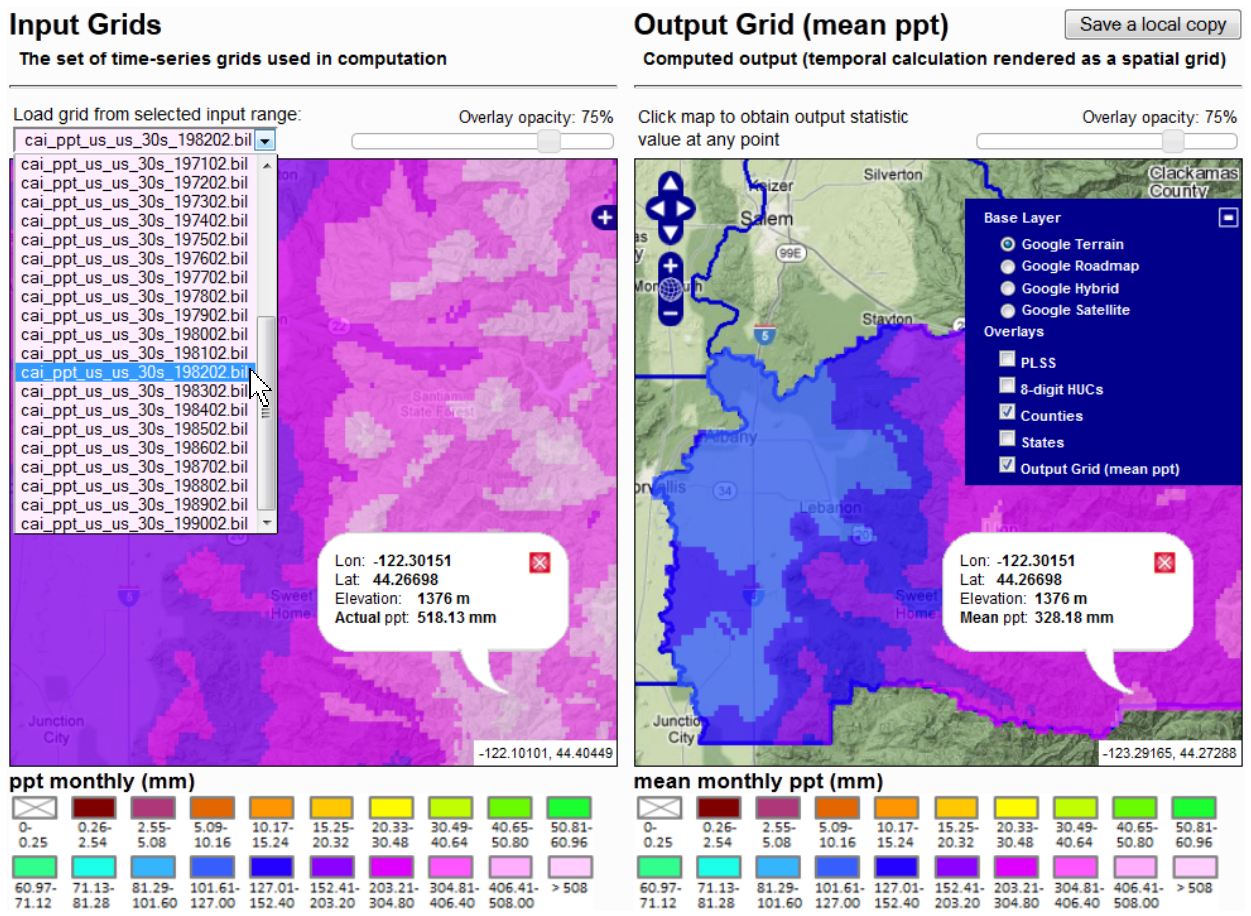
Parameter: **ppt**
Time series: **Monthly**
Statistic: **Mean**
Temporal scale: **One month across range of years**
Month: **Feb** Year range: **1960 to 1990**
Spatial scale: **Linn County, Oregon**
Centroid: **-122.52671 44.49646**
Area processed: **10,103 grid cells (800m resolution)**

Input Grids
The set of time-series grids used in computation
Load grid from selected input range: cai_ppt_us_us_30s_198202.bil
Overlay opacity: 75%

Output Grid (mean ppt)
Computed output (temporal calculation rendered as a spatial grid)
Click map to obtain output statistic value at any point
Overlay opacity: 75%

MEAN monthly ppt for each February (1960 to 1990) for the selected area
(spatial calculation across each time-series grid, plotted over time)

Figure 4.12: Example results page for an area-based request (Linn County, Oregon). Mean precipitation for every February in the year range 1960-1990 was calculated as an output grid (i.e., over time – as viewed in the map), and the mean across each monthly grid was calculated, with the means charted below.



Input Grids

The set of time-series grids used in computation

Load grid from selected input range: cai_ppt_us_us_30s_198202.bil

Overlay opacity: 75%

cai_ppt_us_us_30s_197102.bil
cai_ppt_us_us_30s_197202.bil
cai_ppt_us_us_30s_197302.bil
cai_ppt_us_us_30s_197402.bil
cai_ppt_us_us_30s_197502.bil
cai_ppt_us_us_30s_197602.bil
cai_ppt_us_us_30s_197702.bil
cai_ppt_us_us_30s_197802.bil
cai_ppt_us_us_30s_197902.bil
cai_ppt_us_us_30s_198002.bil
cai_ppt_us_us_30s_198102.bil
cai_ppt_us_us_30s_198202.bil
cai_ppt_us_us_30s_198302.bil
cai_ppt_us_us_30s_198402.bil
cai_ppt_us_us_30s_198502.bil
cai_ppt_us_us_30s_198602.bil
cai_ppt_us_us_30s_198702.bil
cai_ppt_us_us_30s_198802.bil
cai_ppt_us_us_30s_198902.bil
cai_ppt_us_us_30s_199002.bil

Lon: **-122.30151**
Lat: **44.26698**
Elevation: **1376 m**
Actual ppt: 518.13 mm

-122.10101, 44.40449

ppt monthly (mm)

☒ 0-0.25	🟥 0.26-2.54	🟪 2.55-5.08	🟧 5.09-10.16	🟧 10.17-15.24	🟨 15.25-20.32	🟨 20.33-30.48	🟩 30.49-40.64	🟩 40.65-50.80	🟩 50.81-60.96
🟩 60.97-71.12	🟦 71.13-81.28	🟦 81.29-101.60	🟦 101.61-127.00	🟦 127.01-152.40	🟪 152.41-203.20	🟪 203.21-304.80	🟪 304.81-406.40	🟪 406.41-508.00	🟪 > 508

Output Grid (mean ppt)

Save a local copy

Computed output (temporal calculation rendered as a spatial grid)

Click map to obtain output statistic value at any point

Overlay opacity: 75%

Base Layer
● Google Terrain
○ Google Roadmap
○ Google Hybrid
○ Google Satellite

Overlays
☐ PLSS
☐ 8-digit HUCs
☑ Counties
☐ States
☑ Output Grid (mean ppt)

Lon: **-122.30151**
Lat: **44.26698**
Elevation: **1376 m**
Mean ppt: 328.18 mm

-123.29165, 44.27288

mean monthly ppt (mm)

☒ 0-0.25	🟥 0.26-2.54	🟪 2.55-5.08	🟧 5.09-10.16	🟧 10.17-15.24	🟨 15.25-20.32	🟨 20.33-30.48	🟩 30.49-40.64	🟩 40.65-50.80	🟩 50.81-60.96
🟩 60.97-71.12	🟦 71.13-81.28	🟦 81.29-101.60	🟦 101.61-127.00	🟦 127.01-152.40	🟪 152.41-203.20	🟪 203.21-304.80	🟪 304.81-406.40	🟪 406.41-508.00	🟪 > 508

Figure 4.13: Map section of the results page for an area-based request. The paired maps are synchronized for zooming, panning, and queries. Base layers and overlays can be toggled independently, and the Input Grids map allows loading of grids used in the analysis. In this example a point in Linn County, Oregon was clicked, returning the mean precipitation across all Februaries from 1960-1990 for that grid cell, with the actual Feb. 1982 value returned on the input map.

This simplifies the process of comparing the output grid with any of the input grids; above the Input Grids map window is a dropdown menu containing a list of all input grids, and clicking any of those grid names loads the selected grid in the Input Grids map window. By navigating around the maps and clicking any point, the underlying input and output grid values of the matching grid cell are displayed in map popups (Figure 4.13) that also contain the longitude, latitude, and elevation of that grid cell. An underlying digital elevation model (DEM) grid of the same spatial extent and 800 m resolution is queried to obtain the elevation. Additionally, opacity slider controls were added to each map as a visual aid for adjusting how much of the underlying base layer can be viewed through the rendered data grid.

For these maps, point based queries are handled using an AJAX approach to avoid reloading the page for each query. The grid files are queried directly by making a system call to the GDAL `gdallocationinfo` command-line utility, a raster query tool that retrieves a single grid cell value based on a longitude and latitude coordinate pair. This utility is able to quickly index into a BIL grid and return a single cell value. The same utility is used to query the underlying DEM grid.

The output grid calculated from an area-based request represents the “temporal first” calculation, in which each output grid cell represents the statistic calculated through all stacked input grid cells. The GridServer also performs the “spatial first” calculation, whereby the output statistic is calculated across all cells for each input grid. That output is available in an interactive chart (Figure 4.14), configured similarly to the other GridStats charts. PCA results are displayed similarly to other area results, except that the chart is a histogram containing the percentage of variance explained by each principal component, and a dropdown menu is added above the Output Grid map, containing a selectable list of component grids arranged in descending order of variance explained (Figure 4.15).

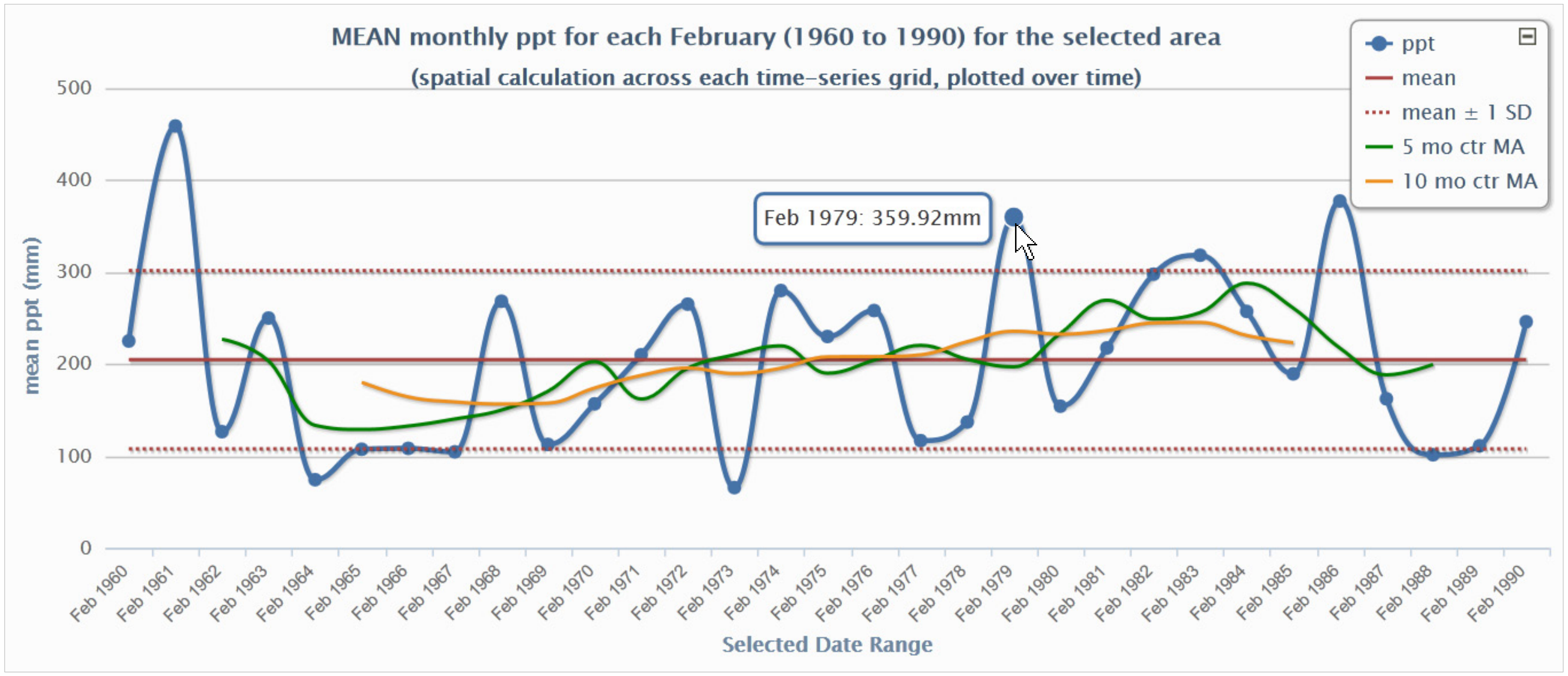


Figure 4.14: Chart section of the results page for an area-based request. The data series on this chart represents the "spatial first" calculation, in which for each grid, the mean is calculated across all grid cells to produce a single value. That set of values is plotted as the data series on this chart. Spatial trends in February mean precipitation across the selected area (Linn County, Oregon) for 1960-1990 can be viewed in this chart. As with all other GridStats charts, the mean line, ±1 standard deviation, and centered moving averages are included.

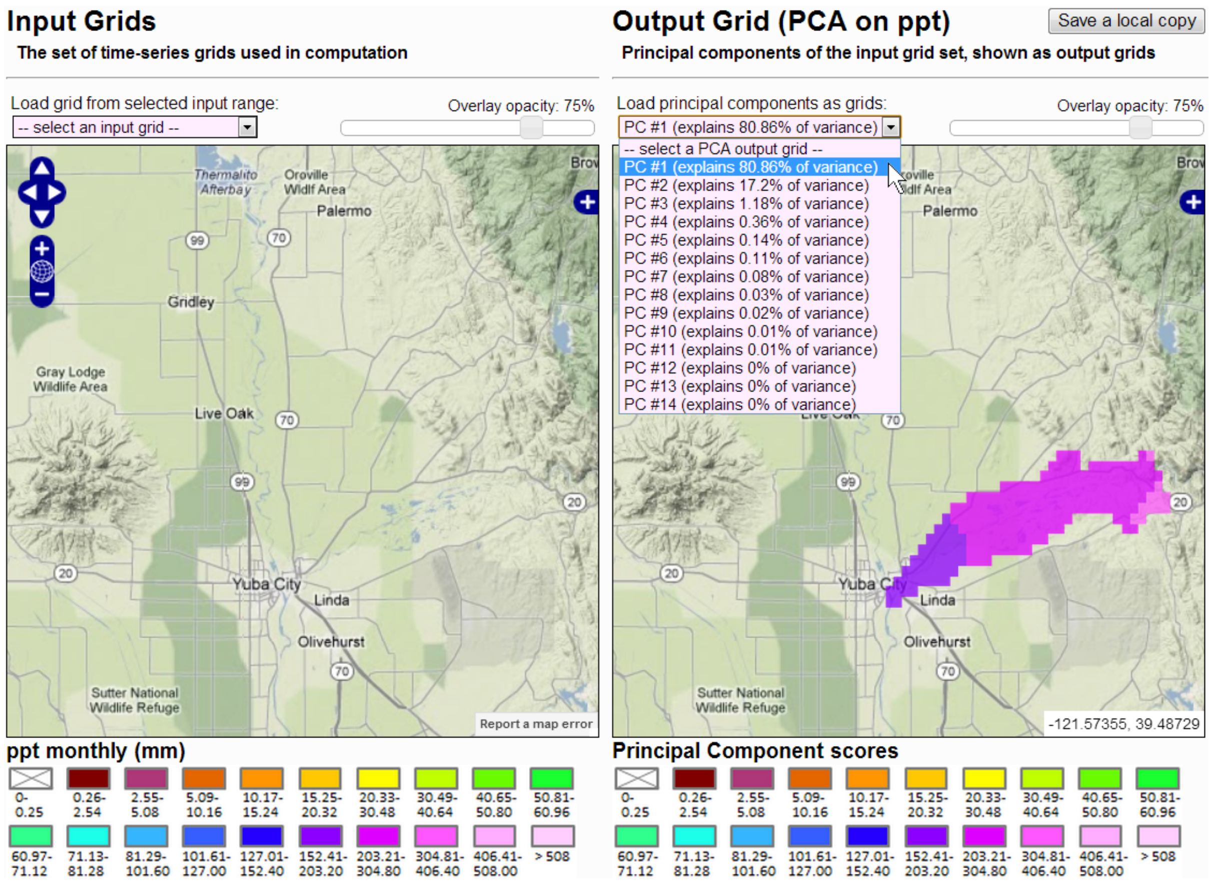

Input Grids
The set of time-series grids used in computation
Load grid from selected input range:
-- select an input grid --
Overlay opacity: 75%
Output Grid (PCA on ppt)
Save a local copy
Principal components of the input grid set, shown as output grids
Load principal components as grids:
PC #1 (explains 80.86% of variance)
Overlay opacity: 75%
-- select a PCA output grid --
PC #1 (explains 80.86% of variance)
PC #2 (explains 17.2% of variance)
PC #3 (explains 1.18% of variance)
PC #4 (explains 0.36% of variance)
PC #5 (explains 0.14% of variance)
PC #6 (explains 0.11% of variance)
PC #7 (explains 0.08% of variance)
PC #8 (explains 0.03% of variance)
PC #9 (explains 0.02% of variance)
PC #10 (explains 0.01% of variance)
PC #11 (explains 0.01% of variance)
PC #12 (explains 0% of variance)
PC #13 (explains 0% of variance)
PC #14 (explains 0% of variance)
Thermalito Afterbay
Oroville Wldlf Area
Palermo
Gridley
Gray Lodge Wildlife Area
Live Oak
Yuba City
Linda
Olivehurst
Sutter National Wildlife Refuge
Report a map error
-121.57355, 39.48729
ppt monthly (mm)
Principal Component scores
0-0.25
0.26-2.54
2.55-5.08
5.09-10.16
10.17-15.24
15.25-20.32
20.33-30.48
30.49-40.64
40.65-50.80
50.81-60.96
60.97-71.12
71.13-81.28
81.29-101.60
101.61-127.00
127.01-152.40
152.41-203.20
203.21-304.80
304.81-406.40
406.41-508.00
> 508

Figure 4.15: Map section of the results page for a PCA area-based request, displaying the result of a PCA run on monthly precipitation for the months January 1999 to February 2000, for the Lower Yuba Watershed, California. The principal components are generated as output grids, with each component explaining a percentage of the variance included in the input grid set. All component grids are made available in a dropdown menu for easy inclusion in the Output Grid map, with the percentage variance explained included in the list.

The GridServer processes and returns area request data remarkably fast, considering how many grid cells it is often tasked with processing in a single request. It can calculate a statistic across a one-degree block (one degree of latitude and longitude, which includes 14,641 grid cells), processing this region across 100 time-series grids, and return the resulting output grid and "spatial first" data values in 2.2 seconds (Table 4.8). For the tests summarized in Table 4.8 the GridServer was again restarted between each request, forcing it to drop and reallocate grid file handles to eliminate any possible performance bias due to caching. Disparate bounding box and polygon locations and non-overlapping date ranges were also used. GridServer requests for these tests were issued in this form (with new randomly-selected bounding boxes or polygons selected for each request):

Bounding box request:

```
time wget -qO test.txt "http://<server.domain>:<port>/
stats_area_months?minlon=-122.43271&minlat=41.33694&maxlon=
-121.29877&maxlat=43.56572&start=19820101&end=19880101&
param=ppt&pstat=mean"
```

Polygon request:

```
time wget -qO test.txt "http://<server.domain>:<port>/
stats_area_months?start=19820101&end=19880101&
param=ppt&pstat=mean&stname=gis.counties&stid=782"
```

Table 4.8: Sample GridServer response times for varying temporal and spatial scales. Each request was area-based (using mean as the statistic), with precipitation as the selected climate parameter on a monthly time scale. The response tasks include calculating an output grid and writing it to the filesystem, and calculating and returning the "spatial first" data values.

Requested date range (monthly ppt data)	**Number of grids in request**	**Spatial extent**	**Number of cells processed per grid**	**Initial response time (s)**	**Average subsequent response time (10 at same location) (s)**
Jan 1920 – Feb 1920	2	1-degree box (-120,43 to -119,44)	14,641	0.16	0.10
Jan 1922 – Oct 1922	10	1-degree box (-81,37 to -80,38)	14,641	0.46	0.19
Jan 1930 – Apr 1938	100	1-degree box (-94,41 to -93,42)	14,641	2.22	1.24
Jan 1940 – Apr 1948	100	3-degree box (-100,40 to -97,43)	130,321	6.78	5.72
Jan 1950 – Dec 1951	24	Aucilla Watershed, Georgia (03110103)	8,505	0.67	0.33
Jan 1960 – Dec 1961	24	Otter Watershed, Vermont (02010002)	10,032	0.71	0.41
Jan 1970 – Dec 1971	24	Wallowa County, Oregon	20,384	1.13	0.43
Jan1980 – Dec 1981	24	Elko County, Nevada	81,266	1.37	0.96
Jan 1990 – Dec 1992	36	Pennsylvania	214,900	2.97	2.69
Jan 2000 – Dec 2002	36	California	1,405,430	19.50	16.32

4.4.8.2.4 Data Export

The statistical output and data products resulting from GridStats requests can be used for reports, publications, or further analysis. The GridStats system is a useful toolkit for generating grids representing an output statistic for either a defined region or the entire CONUS, as well as for generating statistical summaries for a particular point. When a grid (or set of grids, in the case of PCA) is produced as output, either by an area or entire grid request, the grids are stored in a temporary filesystem location made accessible to the web mapping tools. To download output grids, the user can simply click the "Save a local copy" button, which triggers a function that collects the related files and delivers them to the user as a downloadable compressed .zip file.

Data values used for generating the interactive charts are also available for download. Every time a chart is created, the data are fed into a function that generates a comma-separated value (.csv) text file. When the user clicks the "Download chart data" button, the file is delivered to their browser. After saving the file on their computer, the user can simply open it in a spreadsheet program such as Microsoft Excel, which will automatically import data in .csv format. Data values and moving average values are included in the .csv file. Moving averages are not particularly easy to calculate in Microsoft Excel; exporting them from GridStats provides an easy way to import them to a spreadsheet for further exploration.

4.5 Discussion

The system developed in this study integrates a wide range of components for the purpose of performing fast statistical processing and visualization of PRISM climate grids in a web-based environment, and is designed in a manner that gives the user simple yet effective controls for defining a request, plus helpful, interactive tools for interpreting and exploring the output. As system usage and scope expands, additional consideration must be given to system features, optimization, and scalability. This section discusses those issues and mentions some ideas for possible system expansion.

4.5.1 Grid Statistical Calculations

The intent for the initial phase of GridStats development was to design a working system that included a few basic statistical processing functions, as well as one more complicated multivariate technique (PCA) as a proof-of-concept. Those development goals were met and have been successfully tested across a range of spatial and temporal scales, using a variety of processing methods. The implementation of those statistical methods in the grid processing framework lays the groundwork for adding additional univariate and multivariate statistical techniques.

Advanced statistical techniques under consideration for inclusion in the GridStats system include linear regression and *k*-means clustering. Both techniques have established Python methods in NumPy and SciPy, and could therefore be integrated into the current system framework. Another possible feature addition is inhomogeneity testing of climate data. An example of a temporal climate inhomogeneity is urban areas acting as heat sources, where changes in land use (i.e., rural to urban) over time have caused local warming that is potentially not representative of the surrounding rural region. Changes in weather station locations or instrumentation are examples of inhomogeneities commonly encountered in the climate record. The goal of testing would be to determine whether any non-climatic inhomogeneities exist in a data series defined by particular temporal or spatial scales, and to quantify the magnitude of such an inhomogeneity.

4.5.2 Optimization

Extensive testing of data formats for containing the PRISM climate grids was performed prior to this project, resulting in the acceptance of BIL as the standard PRISM grid format and conversion of all existing climate grids from ASCII Grid format to BIL (a procedure made very efficient by the `gdal_translate` command-line utility). The BIL format offers the best performance for fast grid access via both locally-installed GIS software and server-based grid processing services. The PRISM Climate Group and NACSE have developed Python-based grid processing utilities for the GridServer and other tools that utilize NumPy ndarrays for fast programmatic processing of PRISM climate grids. This method is already highly optimized and will likely remain in place as a core technique going forward.

However, it should be noted that the PostGIS Raster project (PostGIS Raster 2012) is under active development and may offer unique possibilities for storing PRISM climate grids in the database and querying subsets of data from them. Although it is unlikely that this would be a faster approach than reading BIL grids directly into NumPy arrays, it merits attention going forward.

The web server machine is optimized for fast operation and multithreaded processing of requests. Performance gains might be achieved by recompiling the MapServer CGI executable, stripping out any features unnecessary for GridStats operations. The database server is optimized for fast reads, with testing performed on key parameter adjustments in the `postgresql.conf` file depending upon available system memory and other specifications, and the best-performing values assigned to those parameters. Spatial queries and operations in the database can be resource-intensive, but all spatial tables used in the GridStats system contain spatial GiST indexes on the geometry columns, as well as standard B-tree indexes on any regularly-queried non-spatial fields (gid, huc_code, etc.). Again, the next step for database optimization would be hardware upgrades, which are planned in the coming months.

The grid processing server is dedicated to processing requests for climate grids and is already optimized for fast performance for that purpose. Further optimization of the grid processing server would entail hardware upgrades (faster CPU, more CPU cores, and additional memory). However, optimization of the existing

Python/Pyramid/paster framework running the GridServer may be possible. In particular, performance gains are likely possible by implementing the nginx HTTP server (Nginx 2012), a fast server with support for FastCGI and uwsgi that uses an event-driven architecture and a very small memory footprint. Nginx also supports simple load balancing and fault tolerance, potentially enabling multiple GridServer instances to be running simultaneously on the same server, with the load balancer shuttling requests to GridServer instances as needed. Initial testing of nginx in the GridServer system realized twofold gains in requests processed per second vs. the paster server. As usage of the GridStats system expands, the framework will likely be upgraded to use nginx as the underlying HTTP server on the grid processing machine(s).

4.5.3 Scalability

The PRISM climate grid set currently contains over 49,000 files. The dedicated grid processing server has a file handle limit of 65,536 set in the /etc/security/limits.conf file. The server has a maximum open file limit of 6,515,139 so there is plenty of headroom available for increasing the GridServer allocation; the limit will indeed be increased as more grids are added to the filesystem. However, each instance of the GridServer ties up the number of file handles in the limit (65,536). If 20 instances of the GridServer are running in an nginx configuration, and the file limit has been changed to 131,072 (for instance), then 2,621,440 file handles will be claimed by GridServer instances alone. While this still provides adequate headroom for multiple instances of the GridServer running simultaneously, it is a potential consideration as the collection of PRISM climate grids continues to expand, along with increased demand for statistical processing.

The bulk of the processing load takes place on the grid processing server. With their current hardware specifications and software configurations, the web server and database server can handle a significantly higher level of GridStats requests. If additional grid processing capability is needed due to increased demand, or the necessity arises for running very computationally intensive statistical calculations over large spatial and/or temporal scales, additional grid processing machines may be added to the system architecture. In this scenario, a load balancer would be placed in front of the grid processing servers to manage requests among them based on current processing load and queued requests. One of two disk configurations would be utilized: (1) mirror a full local copy of the PRISM climate grids on each grid processing server, or (2) divide the grids among the grid processing servers, using the load balancer to route requests to the proper server based on the temporal or spatial parameters used to divide them among servers.

As demand for services increases, another solution is Amazon's EC2 (Amazon EC2 2012). Amazon's EC2 supports unlimited amounts of distributed processing at a reasonable cost, although implementation and management of the system would require significant retooling of the current framework. Grid-based processing represents another possibility for scaling; that is, processing tasks using a scientific grid to perform distributed computing (e.g., Giuliani *et al.* 2011). However, PRISM currently has access to a computational cluster that could be expanded to handle additional load, which would likely be considered before a distributed solution, including the development of a parallel processing approach to handle multiple grid calculations.

4.6 Conclusion

The GridStats system represents a unique web-based approach for calculating PRISM climate grid statistics over varying spatial and temporal scales. It calculates statistics across the highest resolution PRISM climate grids available (800 m), at the highest temporal scale available (daily). Successful design, implementation, and testing of the GridStats system have demonstrated the effectiveness of:

- Using a web-based interface as a front-end to powerful grid processing software components.
- Accessing the GridServer as a fast and reliable system for performing statistical calculations across multiple grids, and extending the GridServer codebase to perform additional methods such as area-based calculations.
- Creating custom tools using open source software such as Python, NumPy, and GDAL to perform fast processing of large multidimensional gridded datasets.
- Using custom map-based tools and a menu validation system for simplifying the user experience of defining temporal and spatial scales for analysis.

- Dynamically building map- and chart-based interactive visualization aids for further exploration of output data.

The GridStats system and the GridServer will scale effectively as additional time-series grids are added to existing PRISM climate parameters grid sets and as new climate parameter grid types are added in the future. Dedicated hardware, a fast network, hardware and software optimization, and continued development will ensure the reliability of the system and keep it up-to-date as data products and software packages are added and improved.

The GridServer is designed in a modular fashion, making it relatively easy to add statistical methods. Additional methods and visualization techniques will be added as tool usage increases and user feedback is incorporated.

4.7 Acknowledgments

Matt Doggett (PRISM Climate Group) developed the "biltools" Fortran utilities used for calculating grid statistics across the entire PRISM grid extent. Ben Steinberg (NACSE) provided helpful comments and suggestions regarding server-side implementation.

4.8 References

Amazon EC2. 2012. Amazon Elastic Compute Cloud (Amazon EC2), http://aws.amazon.com/ec2 (last accessed 20 September 2012).

ArcGIS. 2012. Esri ArcGIS – Mapping & analysis for understanding our world, http://www.esri.com/software/arcgis (last accessed 20 September 2012).

ArcGIS Resource Center. 2012. Server-side processing with raster functions, http://resources.arcgis.com/en/help/main/10.1/index.html#/Server_side_processing_with_raster_functions/0155000003w0000000 (last accessed 20 September 2012).

ArcGIS Spatial Analyst. 2012. ArcGIS Spatial Analyst extension, http://www.esri.com/software/arcgis/extensions/spatialanalyst (last accessed 20 September 2012).

Baumann, P. 2001. Web-enabled raster GIS services for large image and map databases. Pages 870-874 *in* A. Min Tjoa and R. Wagner (eds.), 12th International Workshop on Database and Expert Systems Applications (DEXA 2001), Munich, Germany, 3-7 September. IEEE Computer Society.

Bernard, L. and N. Ostländer. 2008. Assessing climate change vulnerability in the arctic using geographic information services in spatial data infrastructures. *Climatic Change* 87: 263-281.

Bivand, R.S., E.J. Pebesma, and V. Gómez-Rubio. 2008. Applied spatial data analysis with R. Springer, New York.

Bryan, B.A. 2012. High-performance computing tools for the integrated assessment and modelling of socialeecological systems. *Environmental Modelling & Software*. Available online at http://dx.doi.org/10.1016/j.envsoft.2012.02.006.

ClimateWizard. 2012. ClimateWizard, http://www.climatewizard.org (last accessed 20 September 2012).

ClimateWizard Custom. 2012. Climate Wizard Custom, http://www.climatewizardcustom.org (last accessed 20 September 2012).

Comrie, A.C., K. Redmond, M.F. Glueck, and H. Reinbold. 2006. Interactive web-based access and analysis tools for the Western Climate Mapping Initiative (WestMap). EOS, Transactions, American Geophysical Union 87(52), Fall Meeting Supplement, Abstract IN13A-1161.

Cordiero, J.P.C., G. Câmara, U.M. de Freitas, and F. Almeida. 2009. Yet another map algebra. *Geoinformatica* 13: 183-202.

Daly, C., R.P. Neilson, and D.L. Phillips. 1994. A statistical-topographic model for mapping climatological precipitation over mountainous terrain. *Journal of Applied Meteorology* 33: 140-158.

Daly, C., G. Taylor, and W. Gibson. 1997. The PRISM approach to mapping precipitation and temperature. Pages 10-12 *in* Proceedings of the 10th AMS Conference on Applied Climatology, Reno, NV, October 20-23.

Daly, C., W.P. Gibson, G.H. Taylor, G.L. Johnson, and P. Pasteris. 2002. A knowledge-based approach to the statistical mapping of climate. *Climate Research* 22: 99-113.

Daly, C., M. Halbleib, J.I. Smith, W.P. Gibson, M.K. Doggett, G.H. Taylor, J. Curtis, and P.P. Pasteris. 2008. Physiographically sensitive mapping of climatological temperature and precipitation across the conterminous United States. *International Journal of Climatology* 28(15): 2031-2064.

ESRI. 1999. Extendable image formats for ArcView GIS 3.1 and 3.2. ESRI White Paper J-8018, July 1999.

Foerster, T., B. Schäffer, B. Baranski, and J. Brauner. 2011. Geospatial web services for distributed processing – Applications and scenarios. Pages 245-286 *in* P. Zhao and L. Di (eds.), Geospatial web services: Advances in information interoperability. Information Science Reference, Hershey, PA.

Frank, A.U. 2005. Map algebra extended with functors for temporal data. Pages 194-207 *in* J. Akoka *et al.* (eds.), Lecture Notes in Computer Science 3770. Springer-Verlag, Berlin.

GDAL. 2012. Geospatial Data Abstraction Library, http://www.gdal.org/frmt_various.html (last accessed 05 September 2012).

GDAL Raster Formats. 2012. Various supported GDAL raster formats, http://www.gdal.org/frmt_various.html (last accessed 05 September 2012).

Girvetz, E.H., C. Zganjar, G.T. Raber, E.P. Maurer, P. Kareiva, and J.J. Lawler. 2009. Applied climate-change analysis: The Climate Wizard tool. PLoS ONE 4(12): e8320. doi:10.1371/journal.pone.0008320.

Giuliani, G., N. Ray, and A. Lehmann. 2011. Grid-enabled spatial data infrastructure for environmental sciences: Challenges and opportunities. *Future Generation Computer Systems* 27: 292-303.

Glueck, M.F., A.C. Comrie, K. Redmond, H.J. Reinbold, J.T. Abatzoglou, and C. Daly. 2008. The WestMap stakeholder-driven interactive data access and analysis interface for fine-scale climate data. American Meteorological Society 88th Annual Meeting, January 20-24, New Orleans. Available online at https://ams.confex.com/ams/88Annual/techprogram/paper_130602.htm (last accessed 20 September 2012).

GRASS. 2012. GRASS GIS: r.covar, http://grass.fbk.eu/gdp/html_grass62/r.covar.html (last accessed 07 September 2012).

Gutierrez, A.G. and P. Baumann. 2007. Modeling fundamental geo-raster operations with array algebra. Pages 607-612 *in* Proceedings of the Seventh IEEE International Conference on Data Mining, October 28-31, Omaha, NE.

Highcharts. 2012. Highcharts JS: Interactive JavaScript charts for your web projects, http://www.highcharts.com (last accessed 13 September 2012).

Hyndman, R.J. 2010. Moving averages. Pages 866-869 *in* M. Lovirc (ed.), International encyclopedia of statistical science. Springer, New York.

Idrisi. 2012. Idrisi – geospatial software for monitoring and modeling the Earth system, http://www.clarklabs.org (last accessed 20 September 2012).

jQuery. 2012. jQuery JavaScript library, http://www.jquery.com (last accessed 13 September 2012).

Lime, S. 2008. MapServer. Pages 65-85 *in* G.B. Hall and M.G. Leahy (eds.), Open source approaches in spatial data handling. Advances in geographic information science 2. Springer-Verlag, Berlin.

MapServer. 2012. MapServer – Open source web mapping, http://www.mapserver.org (last accessed 20 September 2012).

Mennis, J., R. Viger, and C.D. Tomlin. 2005. Cubic map algebra functions for spatio-temporal analysis. *Cartography and Geographic Information Science* 32: 17-32.

NDIIPP. 2012. Sustainability of digital formats: Band interleaved by line (BIL) image file, http://www.digitalpreservation.gov/formats/fdd/fdd000283.shtml (last accessed 06 September 2012).

Neteler, M. and H. Mitasova. 2002. Open source GIS: A GRASS GIS approach. 3rd edition. Springer, New York.

Nginx. 2012. Nginx – a free, open-source, high-performance HTTP server, http://www.nginx.org (last accessed 20 September 2012).

NumPy. 2012. NumPy – the fundamental package for scientific computing with Python, http://numpy.scipy.org (last accessed 20 September 2012).

OGC WCS. 2010. OGC® WCS 2.0 Interface Standard – Core. Open Geospatial Consortium document reference #OGC 09-111r3. http://portal.opengeospatial.org/files/?artifact_id=41437 (last accessed 20 September 2012).

OGR. 2012. OGR Simple Features Library, http://www.gdal.org/ogr (last accessed 20 September 2012).

OpenLayers. 2012. OpenLayers: Free maps for the web, http://www.openlayers.org (last accessed 20 September 2012).

PostGIS. 2012. PostGIS – A spatial extension to PostgreSQL, http://postgis.refractions.net (last accessed 20 September 2012).

PostGIS Raster. 2012. PostGIS Raster – An ongoing project aiming at developing raster support in PostGIS, http://trac.osgeo.org/postgis/wiki/WKTRaster (last accessed 20 September 2012).

PostgreSQL. 2012. PostgreSQL relational database management system, http://www.postgresql.org (last accessed 20 September 2012).

PRISM Data Explorer. 2012. PRISM Data Explorer, http://prismmap.nacse.org/nn/index.phtml (last accessed 20 September 2012).

Pyramid. 2012. Pyramid open source web application framework, http://www.pylonsproject.org/projects/pyramid (last accessed 18 August 2012).

QGIS. 2012. Quantum GIS – A user-friendly open source geographic information system, http://www.qgis.org (last accessed 20 September 2012).

R Project. 2012. The R project for statistical computing, http://www.r-project.org (last accessed 20 September 2012).

R/GRASS. 2012. Short introduction to geostatistical and spatial data analysis with GRASS and R statistical data language, http://grass.fbk.eu/statsgrass/grass_geostats.html (last accessed 20 September 2012).

SciPy. 2012. SciPy – Scientific tools for Python, http://www.scipy.org (last accessed 20 September 2012).

Siládi, V., L. Huraj, N. Polčák, and E. Vesel. 2012. A parallel processing of spatial data interpolation on computing cloud. Pages 193-198 *in* Proceedings of the Fifth Balkan Conference in Informatics (BCI '12). ACM, New York.

Sp R. 2012. Spatial data in R, http://r-spatial.sourceforge.net (last accessed 20 September 2012).

Tomlin, C.D. 1990. Geographic information systems and cartographic modeling. Prentice Hall, New York.

Tomlin, C.D. 1994. Map algebra: One perspective. *Landscape and Urban Planning* 30(1-2): 3-12.

USGS National Elevation Dataset. 2012. USGS National Elevation Dataset (NED), http://ned.usgs.gov (last accessed 20 September 2012).

WestMap. 2012. WestMap – Climate analysis & mapping toolbox, http://www.cefa.dri.edu/Westmap (last accessed 20 September 2012).

Chapter 5: Conclusion

This dissertation explored project-based methods of handling spatiotemporal data and examined them within a web-based context, considering five related aspects: Storage, representation, processing, analysis, and visualization. By their nature, spatiotemporal data are multidimensional (i.e., the “space-time cube”), requiring specialized approaches in their manipulation and display. Challenges posed by the handling of spatiotemporal data in this context are not insurmountable, but it is difficult to generalize solutions to any of the five aspects given the relative heterogeneity of the data and their inherent complexity. However, this dissertation found that it *is* possible to develop methods for handling spatiotemporal data using targeted solutions, and generalize the resulting methods within the category of data under consideration (e.g., the PRISM climate grids). For instance, the grid processing system described in Chapter 4 was written in a generalized manner that could extend to any set of input grids with a regular extent. With additional development, the grid processing system could likely be modified to work with sets of input grids that vary in spatial extent. The web-based data exploration and visualization framework described in Chapter 4 is already developed in a generalized manner that would likely need no further modification to work with additional datasets.

The web browser is a viable application platform for accepting user input and enabling the exploration and visualization of output, particularly when augmented by client-side libraries and toolkits that enable the development of rich internet applications (RIA). As a common user interface that is able to broker client/server communication, the web browser facilitates the automated processing of spatiotemporal data based on user input, calling any set of server-side functions necessary to process the data according to the user's request. This paradigm is extensible – new server-side methods can be added and called with minimal changes to the client-side codebase.

5.1 Web-based Approaches and New Technology

At the outset it was unclear whether the proposed projects could be fully developed and implemented within a web-based platform. Standard web pages did not provide enough client-side flexibility and support for asynchronous client/server interaction to support the complexity required by the user interfaces, and many of the now-popular JavaScript libraries enabling advanced functionality were under development and not yet well-known. The advancement of web technology moves at a fast pace; project development plans were altered several times as client-side software technology advanced during the course of the dissertation. Although low-level JavaScript code could be written to perform many of the functions enabled by client-side software tools and methods used in this project, their availability made the development of RIA much more practical and feasible. Chapters 3 and 4 demonstrate RIA that use rich client-side JavaScript libraries such as OpenLayers

(2012), jQuery (2012), and Highcharts (2012) for data retrieval, processing, and visualization.

A number of next-generation web technologies are currently under development. The nascent but quickly-growing category of rich JavaScript application platforms such as Backbone (2012) and Meteor (2012) will enable next-generation development of RIA. These platforms support the development of entire applications in JavaScript, with the same APIs available on both the client and server, which enables tight communication and synchronization between the client and the server. Also central to the advancement of RIA is HTML5 (2012), the fifth revision of the HyperText Markup Language standard, which includes new syntactic features such as <video> elements, integrates scalable vector graphics (SVG) support, and obviates the need for a server-side web software framework such as Adobe Flash or Microsoft Silverlight.

Although initial development of RIA can be more complex than for standard internet applications (Toffetti *et al.* 2011), the benefits of asynchronous client/server interaction, minimized bandwidth usage, sophisticated user interfaces, and flexibility and extensibility for future upgrades makes the RIA approach an attractive option. A great deal of custom client-side code was developed for the projects described in this dissertation. Although new client-side and RIA technologies may supplant some of the software packages used in these projects, the core codebase should remain adaptable and endure as changes are made to the underlying frameworks.

5.2 Automated Data Processing

Automated data processing is an important element in the group of procedures that enable real-time web-based analysis and visualization of spatiotemporal data, but generalization of server-side data processing code poses a challenge. In the projects described in Chapters 2-4, specialized server-side code was developed to perform automated data processing tasks. It is not unusual to develop specialized code for handling particular processing tasks; however, the challenge lies in designing the code in an extensible fashion, so that additional methods can be added without extensive code rewrites. The GridServer, described in Chapter 4, is developed in this fashion – additional methods for processing PRISM climate grids over varying spatial and temporal scales can be developed and "plugged in" to the system.

In some cases making direct system calls to server-side functions makes sense; Chapter 2 demonstrates an example of this approach, which works well for snapping a selected data window to an existing bathymetry grid, for example, as the user moves through the interface one tab at a time. However, a typically more robust approach leverages the client/server architecture by using a server-side web server gateway interface (WSGI) framework such as the Python-based Pyramid (2012). This type of framework facilitates the direct execution of specialized server-side

Python processing code and can be designed to return the output as a JavaScript object notation (JSON) object via a hypertext transfer protocol (HTTP) call for further analysis and visualization. This approach is used in Chapter 3, where, using an asynchronous JavaScript and XML (AJAX) approach, a map click creates a call to a server-side function that retrieves water depths in a single grid cell across all time steps and returns a JSON object that is processed on the client-side to automatically update the interactive chart. Chapter 4 describes an approach where HTTP-based requests are sent to the server to process point- or area-based requests for statistical analysis of climate grid data over time. The resulting data are returned to the client as a JSON object, which is used to update interactive map-based and chart-based visualizations.

5.3 Spatiotemporal Data Representation and Visualization

Chapter 2 describes the time-series output data format produced by the Tsunami Computational Portal (TCP) modeling process – the output data are written to a binary file containing water depth values at each time step. In essence, the spatiotemporal data stored in this two-dimensional (2D) file are arranged in the "temporal snapshot" representation. Similarly, the PRISM climate grids described in Chapter 4 each represent a temporal snapshot on daily, monthly, or annual scales. In Chapter 3, the simulation framework generates time-series data representing the movement of each simulated individual across all time steps. Those data are written to a spatial database, with each record representing an individual's status and location at that time step.

The binary time-series files described in Chapters 2 and 4 represent temporal snapshots by default. Although the project described in Chapter 3 could possibly store temporal information using an object-oriented identity-based change approach such as Hornsby and Egenhofer (2000), or a more general atomic representation such as the one described by Goodchild *et al.* (2007), the temporal snapshot approach is a straightforward and efficacious representation that satisfies the database query and web interface requirements. The snapshot data are properly indexed in the spatial database (both B-tree and spatial generalized search tree indices are automatically created on the necessary columns), resulting in the very fast execution of temporal queries to display data in real-time animations and other visualizations in the web interface.

One of the primary study goals was to automatically generate web-based visualizations from output data products. The projects described in Chapters 2-4 each generate different types of visualizations, from non-interactive tsunami propagation animations (Chapter 2), to interactive animations overlaid on a map-based tool with a linked chart (Chapter 3), to synchronized, map-based visualization of statistical output combined with an interactive chart (Chapter 4). The animations produced in Chapter 2 occur as an automated server-side process. All visualization tools in Chapters 3 and 4 were built using JavaScript-based code developed as part of the overarching rich internet applications (RIA) approach, and enhanced by dynamic server-side processing of spatiotemporal data.

5.4 Summary

Rich internet applications, such as the web-based interfaces described in Chapters 2-4, represent a viable alternative to traditional locally-installed software applications for specific tasks. While RIA will likely never replace full-featured software applications such as ArcGIS (2012), they can (i.e., Chapter 4) provide specific processing functionality that takes advantage of real-time client/server architecture to quickly produce and display output in an interactive tool. In this sense, the web browser is an equalizing application that provides a common user interface platform and is available to anyone with a computer and an internet connection.

Spatiotemporal data, which are inherently multidimensional, are well-suited for targeted server-side processing and analysis tools. Their complexity and typically large storage size on disk can be prohibitive to manage in a local setting. Server-side processing of the data leverages fast data access and high-performance hardware to perform the operation remotely and deliver just the output product to the client. With careful planning, server-side processing of spatiotemporal data can be automated, with the automation developed in a manner generalizable to other data types or disciplines.

5.5 References

ArcGIS. 2012. Esri ArcGIS – Mapping & analysis for understanding our world, http://www.esri.com/software/arcgis (last accessed 20 September 2012).

Backbone. 2012. Backbone.js JavaScript application. http://www.backbonejs.org (last accessed 28 September 2012).

Goodchild, M., M. Yuan, and T. Cova. 2007. Towards a general theory of geographic representation in GIS. *International Journal of Geographic Information Science* 21(3): 239-260.

Highcharts. 2012. Highcharts JS – Interactive JavaScript charts for your web projects, http://www.highcharts.com (last accessed 13 September 2012).

Hornsby, K. and M.J. Egenhofer. 2000. Identity-based change: A foundation for spatio-temporal knowledge representation. *International Journal of Geographical Information Science* 14(3): 207-224.

HTML5. 2012. HTML5 – the new HTML standard, http://www.w3schools.com/html/html5_intro.asp (last accessed 28 September 2012).

jQuery. 2012. jQuery JavaScript library, http://www.jquery.com (last accessed 13 September 2012).

Meteor. 2012. Meteor JavaScript application, http://www.meteor.com (last accessed 28 September 2012).

OpenLayers. 2012. OpenLayers – free maps for the web, http://www.openlayers.org (last accessed 20 September 2012).

Pyramid. 2012. Pyramid open source web application framework, http://www.pylonsproject.org/projects/pyramid (last accessed 18 August 2012).

Toffetti, G., S. Comai, J.C. Preciado, and M. Linaje. 2011. State-of-the-art and trends in the systematic development of rich internet applications. *Journal of Web Engineering* 10(1): 70-86.

Chapter 6: Bibliography

ADCIRC. 2012. A (Parallel) Advanced Circulation Model for Oceanic, Coastal and Estuarine Waters, http://www.adcirc.org (last accessed 10 February 2012).

Adobe. 2012. Adobe Flex framework, http://www.adobe.com/products/flex.html (last accessed 17 May 2012).

Allan, J.C., P.D. Komar, P. Ruggiero, and R. Witter. 2012. The March 2011 Tōhoku tsunami and its impacts along the U.S. West Coast. *Journal of Coastal Research* 28(5): 1142-1153.

Amante, C. and B.W. Eakins. 2009. ETOPO1 1 arc-minute global relief model: Procedures, data sources and analysis. NOAA Technical Memorandum NESDIS NGDC-24, 19 pp, March.

Amazon EC2. 2012. Amazon Elastic Compute Cloud (Amazon EC2), http://aws.amazon.com/ec2 (last accessed 20 September 2012).

American Time Use Survey. 2012. Bureau of Labor Statistics – American Time Use Survey, http://www.bls.gov/tus (last accessed 2 August 2012).

Anderson, J.C., M. Davis, K. Fujiwara, T. Fang, J. Andres, and M. Nedbal. 2011. Web-based scientific visualization software for geospatial displays and collaborative applications. In Proceedings of the OCEANS '11 Conference. Kona, Hawaii.

Andrienko, G., N. Andrienko, and S. Wrobel. 2007. Visual analytics tools for analysis of movement data. SIGKDD Explorations 9(2): 38-46.

Andrienko, G., N. Andrienko, P. Bak, D. Keim, S. Kisilevich, and S. Wrobel. 2011a. A conceptual framework and taxonomy of techniques for analyzing movement. *Journal of Visual Languages and Computing* 22(3): 213-232.

Andrienko, G., N. Andrienko, and M. Heurich. 2011b. An event-based conceptual model for context-aware movement analysis. *International Journal of Geographical Information Science* 25(9): 1347-1370.

ArcGIS. 2012. Esri ArcGIS – Mapping & analysis for understanding our world, http://www.esri.com/software/arcgis (last accessed 20 September 2012).

ArcGIS Help. 2012. What's new for temporal data in ArcGIS 10, http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#//00qp00000018000000.htm (last accessed 20 April 2012).

ArcGIS Resource Center. 2012a. How time is supported in spatial data, http://resources.arcgis.com/en/help/main/10.1/index.html#//005z00000004000000 (last accessed 27 September 2012).

ArcGIS Resource Center. 2012b. Server-side processing with raster functions, http://resources.arcgis.com/en/help/main/10.1/index.html#/Server_side_processing_with_raster_functions/0155000003w0000000 (last accessed 20 September 2012).

ArcGIS Spatial Analyst. 2012. ArcGIS Spatial Analyst extension, http://www.esri.com/software/arcgis/extensions/spatialanalyst (last accessed 20 September 2012).

Armstrong, M.P. 1988. Temporality in spatial databases. Proceedings: GIS/LIS '88 2: 880-889.

Atkins, D.E., K.K. Droegemeier, S. Feldman, H. Garcia-Molina, M.L. Klein, D.G. Messerschmitt, P. Messina, J.P. Ostriker, and M.H. Wright. 2003. Revolutionizing science and engineering through cyberinfrastructure: Report of the National Science Foundation blue-ribbon advisory panel on cyberinfrastructure, http://www.nsf.gov/cise/sci/reports/atkins.pdf (last accessed 24 February 2012).

Backbone. 2012. Backbone.js JavaScript application. http://www.backbonejs.org (last accessed 28 September 2012).

Barkan, R., U.S. ten Brink, and J. Lin. 2008. Trans-Atlantic tsunamis: Simulations of the 1755 Lisbon and of hypothetical Puerto Rico trench earthquake tsunamis. American Geophysical Union (AGU) Fall Meeting, San Francisco, CA, December 15-19.

Barkan, R., U.S. ten Brink, and J. Lin. 2009. Far field simulations of the 1755 Lisbon earthquake: Implications for tsunami hazard to the U.S. East Coast and the Caribbean. *Marine Geology* 264(1-2): 109-122.

Barkan, R. and U. ten Brink. 2010. Tsunami simulations of the 1867 Virgin Island earthquake: Constraints on epicenter location and fault parameters. *Bulletin of the Seismological Society of America* 100(3): 995-1009.

Bashar, M.A., M. Islam, M.A. Chowdhury, M.P. Sajjad, and M.T. Ahmed. 2012. Concept for a web map implementation with faster query response. *Journal of Information Engineering and Applications* 2(1): 30-37.

Baumann, P. 2001. Web-enabled raster GIS services for large image and map databases. Pages 870-874 *in* A. Min Tjoa and R. Wagner (eds.), 12th International Workshop on Database and Expert Systems Applications (DEXA 2001), Munich, Germany, 3-7 September. IEEE Computer Society.

Beller, A., T. Giblin, K.V. Le, S. Litz, T. Kittel, and D. Schimel. 1991. A temporal GIS prototype for global change research. Proceedings: GIS/LIS '91 2: 752-765.

Bernard, E.N. 2005. The U.S. National Tsunami Hazard Mitigation Program: A successful state-federal partnership. *Natural Hazards* 35(1): 5-24.

Bernard, E.N., H.O. Mofjeld, V. Titov, C.E. Synolakis, and F.I. González. 2006. Tsunami: Scientific frontiers, mitigation, forecasting and policy implications. *Philosophical Transactions of the Royal Society* 364: 1989-2007.

Bernard, L. and N. Ostländer. 2008. Assessing climate change vulnerability in the arctic using geographic information services in spatial data infrastructures. *Climatic Change* 87: 263-281.

Betrancourt, M. 2005. The animation and interactivity principles in multimedia learning. Pages 287-296 *in* R.E. Mayer (ed.), Cambridge handbook of multimedia learning. Cambridge University Press, New York.

Bivand, R.S., E.J. Pebesma, and V. Gómez-Rubio. 2008. Applied spatial data analysis with R. Springer, New York.

Blain, C.A. and K. Kelly. 2001. Software design description for the Advanced Circulation Model (ADCIRC). NRL Memorandum Report, NRL/MR/7320-01-8271, Naval Research Laboratory, Department of the Navy, 12/2001.

Brassel, K.E. and R. Weibel. 1988. A review and conceptual framework of automated map generalization. *International Journal of Geographical Information Science* 2(3): 229-244.

Brown, B., M. Chui, and J. Manyika. 2011. Are you ready for the era of 'big data?' McKinsey Quarterly, October, https://www.mckinseyquarterly.com/Are_you_ready_for_the_era_of_big_data_2864 (last accessed 20 September 2012).

Bryan, B.A. 2012. High-performance computing tools for the integrated assessment and modelling of socialeecological systems. *Environmental Modelling & Software*. Available online at http://dx.doi.org/10.1016/j.envsoft.2012.02.006.

Buckley, A.R., M. Gahegan, and K. Clarke. 2005. Geographic visualization. Pages 313-333 *in* R. B. McMaster and E. L. Usery (eds.), A research agenda for geographic information science. CRC Press, Boca Raton, FL.

Calheiros, R.N., R. Ranjan, A. Beloglazov, C.A.F. DeRose, and R. Buyya. 2010. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience* 41(1): 23-50.

Chen, X. and F.B. Zhan. 2008. Agent-based modelling and simulation of urban evacuation: Relative effectiveness of simultaneous and staged evacuation strategies. *Journal of the Operational Research Society* 59(1): 25-33.

Clerveaux, V., T. Katada, and K. Hosoi. 2008. Tsunami scenario simulator: A tool for ensuring effective disaster management and coastal evacuation in a multilanguage society. *Science of Tsunami Hazards* 27(3): 48-71.

ClimateWizard. 2012. ClimateWizard, http://www.climatewizard.org (last accessed 20 September 2012).

ClimateWizard Custom. 2012. Climate Wizard Custom, http://www.climatewizardcustom.org (last accessed 20 September 2012).

Comrie, A.C., K. Redmond, M.F. Glueck, and H. Reinbold. 2006. Interactive web-based access and analysis tools for the Western Climate Mapping Initiative (WestMap). EOS, Transactions, American Geophysical Union 87(52), Fall Meeting Supplement, Abstract IN13A-1161.

Cordiero, J.P.C., G. Câmara, U.M. de Freitas, and F. Almeida. 2009. Yet another map algebra. *Geoinformatica* 13: 183-202.

Cova, T.J. and J.P. Johnson. 2003. A network flow model for lane-based evacuation routing. *Transportation Research Part A* 37(7): 579-604.

Dall'Osso, F., M. Gonella, G. Gabbianelli, G. Withycombe, and D. Dominey-Howes. 2009. A revised (PTVA) model for assessing the vulnerability of buildings to tsunami damage. *Natural Hazards and Earth System Sciences* 9: 1557-1565.

Dall'Osso, F., A. Maramai, L. Graziani, B. Brizuela, A. Cavalletti, M. Gonella, and S. Tinti. 2010. Applying and validating the PTVA-3 Model at the Aeolian Islands, Italy: Assessment of the vulnerability of buildings to tsunamis. *Natural Hazards and Earth System Sciences* 10: 1547-1562.

Dalquen, D.A., M. Anisimova, G.H. Gonnet, and C. Dessimoz. 2012. ALF – A simulation framework for genome evolution. *Molecular Biology and Evolution* 29(4): 1115-1123.

Daly, C., R.P. Neilson, and D.L. Phillips. 1994. A statistical-topographic model for mapping climatological precipitation over mountainous terrain. *Journal of Applied Meteorology* 33: 140-158.

Daly, C., G. Taylor, and W. Gibson. 1997. The PRISM approach to mapping precipitation and temperature. Pages 10-12 *in* Proceedings of the 10th AMS Conference on Applied Climatology, Reno, NV, October 20-23.

Daly, C., W.P. Gibson, G.H. Taylor, G.L. Johnson, and P. Pasteris. 2002. A knowledge-based approach to the statistical mapping of climate. *Climate Research* 22: 99-113.

Daly, C., M. Halbleib, J.I. Smith, W.P. Gibson, M.K. Doggett, G.H. Taylor, J. Curtis, and P.P. Pasteris. 2008. Physiographically sensitive mapping of climatological temperature and precipitation across the conterminous United States. *International Journal of Climatology* 28(15): 2031-2064.

Dijkstra, E.W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik* 1(1): 269-271.

DOGAMI. 2005. Tsunami evacuation map: Seaside. Oregon Department of Geology and Mineral Industries publication DOGAMI-TS-EB-SEA-01 (11/05), http://www.oregongeology.org/pubs/tsubrochures/SeasideEvac.pdf (last accessed 2 May 2012).

Dominey-Howes, D., P. Dunbar, J. Varner, and M. Papathoma-Köhle. 2010. Estimating probable maximum loss from a Cascadia tsunami. *Natural Hazards* 53: 43-61.

Dunbar, P. and H. McCullough. 2011. Global tsunami deposits database. *Natural Hazards* 63(1): 267-278.

Dunbar, P., H. McCullough, and G. Mungov. 2011. 2011 Tohoku earthquake and tsunami data available from the National Oceanic and Atmospheric Administration/National Geophysical Data Center. *Geomatics, Natural Hazards and Risk* 2(4): 305-323.

Eckert, S., R. Jelinek, G. Zeug, and E. Krausmann. 2012. Remote sensing-based assessment of tsunami vulnerability and risk in Alexandria, Egypt. *Applied Geography* 32(2): 714-723.

ESRI. 1999. Extendable image formats for ArcView GIS 3.1 and 3.2. ESRI White Paper J-8018, July 1999.

Esri CityEngine. 2012. Esri CityEngine: Smart 3D city models, http://www.esri.com/software/cityengine (last accessed 12 October 2012).

Eubanks, J. 1994. Pedestrian accident reconstruction. Lawyers and Judges, Tucson, AZ.

Foerster, T., B. Schäffer, B. Baranski, and J. Brauner. 2011. Geospatial web services for distributed processing – Applications and scenarios. Pages 245-286 *in* P. Zhao and L. Di (eds.), Geospatial web services: Advances in information interoperability. Information Science Reference, Hershey, PA.

Frank, A.U. 2005. Map algebra extended with functors for temporal data. Pages 194-207 *in* J. Akoka *et al.* (eds.), Lecture Notes in Computer Science 3770. Springer-Verlag, Berlin.

Gahegan, M., M. Wachowicz, M. Harrower, and T-M. Rhyne. 2001. The integration of geographic visualization with knowledge discovery in databases and geocomputation. *Cartography and Geographic Information Science* 28(1): 29-44.

Garbin, D.A. and J.L. Fisher. 2010. Open source for enterprise geographic information systems. *IT Professional* 12(6): 38-45.

Garrett, J.J. 2005. Ajax: A new approach to web applications, http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications (last accessed 28 September 2012).

GDAL. 2012. Geospatial Data Abstraction Library, http://www.gdal.org/frmt_various.html (last accessed 05 September 2012).

GDAL Raster Formats. 2012. Various supported GDAL raster formats, http://www.gdal.org/frmt_various.html (last accessed 05 September 2012).

GEBCO. 2010. General bathymetric chart of the oceans: The GEBCO_08 grid, http://www.gebco.net/data_and_products/gridded_bathymetry_data/documents/gebco_08.pdf (last accessed 11 February 2012).

Geist, E.L. and T. Parsons. 2006. Probabilistic analysis of tsunami hazards. *Natural Hazards* 37: 277-314.

Geist, E.L., V.V. Titov, and C.E. Synolakis. 2006. Tsunami: Wave of change. *Scientific American* 294(1): 56-63.

George, D.L. and R.J. LeVeque. 2006. Finite volume methods and adaptive refinement for global tsunami propagation and local inundation. *Science of Tsunami Hazards* 24(5): 319-328.

GeoServer. 2012. GeoServer – a Java-based software server that allows users to view and edit geospatial data, http://www.geoserver.org (last accessed 28 September 2012).

Girvetz, E.H., C. Zganjar, G.T. Raber, E.P. Maurer, P. Kareiva, and J.J. Lawler. 2009. Applied climate-change analysis: The Climate Wizard tool. PLoS ONE 4(12): e8320. doi:10.1371/journal.pone.0008320.

Gisler, G.R. 2008. Tsunami simulations. *Annual Review of Fluid Mechanics* 40: 71-90.

Giuliani, G., N. Ray, and A. Lehmann. 2011. Grid-enabled spatial data infrastructure for environmental sciences: Challenges and opportunities. *Future Generation Computer Systems* 27: 292-303.

Glueck, M.F., A.C. Comrie, K. Redmond, H.J. Reinbold, J.T. Abatzoglou, and C. Daly. 2008. The WestMap stakeholder-driven interactive data access and analysis interface for fine-scale climate data. American Meteorological Society 88th Annual Meeting, January 20-24, New Orleans. Available online at https://ams.confex.com/ams/88Annual/techprogram/paper_130602.htm (last accessed 20 September 2012).

GMT. 2012. Generic Mapping Tools, http://gmt.soest.hawaii.edu (last accessed 24 January 2012).

Goff, J.R., S.L. Nichol, and D. Kennedy. 2010. Development of a palaeotsunami database for New Zealand. *Natural Hazards* 54(2): 193-208.

Goff, J., C. Chagué-Goff, D. Dominey-Howes, B. McAdoo, S. Cronin, M. Bonté-Grapetin, S. Nichol, M. Horrocks, M. Cisternas, G. Lamarche, B. Pelletier, B. Jaffe, and W. Dudley. 2011. Palaeotsunamis in the Pacific Islands. *Earth-Science Reviews* 107: 141-146.

Goff, J.R., C. Chagué-Goff, and J.P. Terry. 2012. The value of a Pacific-wide tsunami database to risk reduction: Putting theory into practice. *Geological Society, London, Special Publications* 361: 209-220.

Goodchild, M.F., R.P. Haining, S. Wise, *et al.* 1992. Integrating GIS and spatial analysis – Problems and possibilities. *International Journal of Geographic Information Science* 6(5): 407-423.

Goodchild, M.F., M.J. Egenhofer, K.K. Kemp, D.M. Mark, and E. Sheppard. 1999. Introduction to the Varenius project. *International Journal of Geographical Information Science* 13(8): 731-745.

Goodchild, M., M. Yuan, and T. Cova. 2007. Towards a general theory of geographic representation in GIS. *International Journal of Geographic Information Science* 21(3): 239-260.

Google Earth. 2012. Google Earth, http://earth.google.com (last accessed 20 September 2012).

Google Maps. 2012. Google Maps, http://maps.google.com (last accessed 20 September 2012).

Google Visualization API. 2012. Google Visualization API Reference, https://developers.google.com/chart/interactive/docs/reference (last accessed 21 July 2012).

Goth, G. 2012. The science of better science. *Communications of the Association for Computing Machinery* 55(2): 13-15.

Goto, Y., Y. Ogawa, and T. Komura. 2010. Tsunami disaster reduction education using town watching and moving tsunami evacuation animation – trial in Banda Aceh. *Journal of Earthquake and Tsunami* 4(2): 115-126.

Goto, Y., M. Affan, Agussabti, Y. Nurdin, D.K. Yuliana, and Ardiansyah. 2012. Tsunami evacuation simulation for disaster education and city planning. *Journal of Disaster Research* 7(1): 92-101.

GRASS. 2012. GRASS GIS: r.covar, http://grass.fbk.eu/gdp/html_grass62/r.covar.html (last accessed 07 September 2012).

Gupta, A.K. 2011. Recent Tohoku tsunami and earthquake: A natural consequence of the movement of the Pacific plate along the trench, east of the Japanese island arc. *Science and Culture* 77(5-6): 175-185.

Gutierrez, A.G. and P. Baumann. 2007. Modeling fundamental geo-raster operations with array algebra. Pages 607-612 *in* Proceedings of the Seventh IEEE International Conference on Data Mining, October 28-31, Omaha, NE.

Hägerstrand, T. 1970. What about people in regional science? *Papers of the Regional Science Association* 24: 7-21.

Hegarty, M., S. Kriz, and C. Cate. 2003. The roles of mental animations and external animations in understanding mechanical systems. *Cognition and Instruction* 21(4): 209-249.

Highcharts. 2012. Highcharts JS: Interactive JavaScript charts for your web projects, http://www.highcharts.com (last accessed 13 September 2012).

Hornsby, K. and M.J. Egenhofer. 2000. Identity-based change: A foundation for spatio-temporal knowledge representation. *International Journal of Geographical Information Science* 14(3): 207-224.

Hornsby, K. and M.J. Egenhofer. 2002. Modeling moving objects over multiple granularities. *Annals of Mathematics and Artificial Intelligence* 36(1-2): 177-194.

HTML5. 2012. HTML5 – the new HTML standard. http://www.w3schools.com/html/html5_intro.asp (last accessed 28 September 2012).

Huang, W., C.-L. Huang, and C.-H. Wu. 2006. The development of a computational grid portal. In Proceedings of the sixth IEEE international symposium on cluster computing and the grid, Singapore, May 16-19.

Hyndman, R.J. 2010. Moving averages. Pages 866-869 *in* M. Lovirc (ed.), International encyclopedia of statistical science. Springer, New York.

Idrisi. 2012. Idrisi – geospatial software for monitoring and modeling the Earth system, http://www.clarklabs.org (last accessed 20 September 2012).

Imamura, F., A.C. Yalciner, and G. Ozyurt. 2006. Tsunami modeling manual (TUNAMI model), http://www.tsunami.civil.tohoku.ac.jp/hokusai3/J/projects/manual-ver-3.1.pdf (last accessed 10 February 2012).

Janik, C., M. Bailey, D. Keon, C. Pancake, and H. Yeh. 2007. Web-based tsunami visualization. 9th International Conference on Fluid Control, Measurements, and Visualization (FLUCOME), Tallahassee, FL, September 16-19.

jQuery. 2012. jQuery JavaScript library, http://www.jquery.com (last accessed 13 September 2012).

Jonkman, S.N., J.K. Vrijling, and A.C.W.M. Vrouwenvelder. 2008. Methods for the estimation of loss of life due to floods: A literature review and a proposal for a new method. *Natural Hazards* 46: 353-389.

Johnston, D., D. Paton, G.L. Crawford, K. Ronan, B. Houghton, and P. Bürgelt. 2004. Measuring tsunami preparedness in coastal Washington, United States. *Natural Hazards* 35(1): 173-184.

Kapler, T. and W. Wright. 2005. GeoTime information visualization. *Information Visualization* 4(2): 136-146.

Karon, J. and H. Yeh. 2011. Comprehensive tsunami simulator for Cannon Beach, Oregon. Final Report submitted to the City of Cannon Beach, May 2011, http://www.tappister.com/assets/content/2011/05/Cannon-Beach-Tsunami-Final-Report-May-2011.pdf (last accessed 22 April 2012).

Katada, T., J. Asada, N. Kuwasawa, and Y. Oikawa. 2000. Development of practical scenario simulator for dissemination of disaster information. *Journal of Civil Engineering Information Processing System* 9: 129-136.

Katada, T. 2003. A preliminary integrated tsunami scenario simulation. Workshop for Integrated Tsunami Scenario Simulation, Oregon State University, http://tsunami.orst.edu/workshop/2003/pdf/Katada.pdf (last accessed 20 April 2012).

Katada, T., N. Kuwasawa, and H. Yeh. 2004. Tsunami scenario simulator. 2nd Workshop for Integrated Tsunami Scenario Simulation, Oregon State University, http://tsunami.orst.edu/workshop/2004/doc/Katada.pdf (last accessed 20 April 2012).

Katada, T., N. Kuwasawa, H. Yeh, and C. Pancake. 2006. Integrated simulation of tsunami hazards. EERI's 8th U.S. National Conference on Earthquake Engineering (8NCEE), San Francisco, CA. Paper No. 1727.

Keon, D., C. Pancake, H. Yeh, and T. Logan. 2009. The Tsunami Computational Portal: Distributed infrastructure for executing and comparing multiple computational models. Association of American Geographers (AAG) Annual Meeting, Cyberinfrastructure Specialty Group, Las Vegas, NV, 22-27 March.

Keon, D. 2010. Database management systems. Pages 671-675 *in* B. Warf (ed.), Encyclopedia of geography. SAGE Publications, Thousand Oaks, CA.

Kim, S., M. Yoon, S.-M. Whang, B. Tversky, and J.B. Morrison. 2007. The effect of animation on comprehension and interest. *Journal of Computer Assisted Learning* 23(3): 260-270.

Kobayashi, K. T. Katada, N. Kuwasara, A. Narita, M. Hirano, and I. Kase. 2007. Collaborative interface for disaster simulation. *In* Proceedings of the 2nd International Conference on Urban Disaster Reduction, Taipei, Taiwan.

Koshimura, S. and M. Takashima. 2005. Remote sensing, GIS, and modeling technologies enhance the synergic capability to comprehend the impact of great tsunami disaster. *In* Proceedings of the 3rd International Workshop on Remote Sensing for Post Disaster Response, Chiba, Japan.

Koshimura, S., T. Katada, H.O. Mofjeld, and Y. Kawata. 2006. A method for estimating casualties due to the tsunami inundation flow. *Natural Hazards* 39(2): 265-274.

Koshimura, S., Y. Namegaya, and H. Yanagisawa. 2009a. Tsunami fragility – A new measure to identify tsunami damage. *Journal of Disaster Research* 4(6): 479-488.

Koshimura, S., T. Oie, H. Yanagisawa, and F. Imamura. 2009b. Developing fragility functions for tsunami damage estimation using numerical model and post-tsunami data from Banda Aceh, Indonesia. *Coastal Engineering Journal* 51(3): 243-273.

Kowalik, Z. and T.S. Murty. 1993. Numerical simulation of two-dimensional tsunami runup. *Marine Geodesy* 16: 87-100.

Kraak, M. 2003. The space-time cube revisited from a geovisualization perspective. Pages 1988-1996 *in* Proceedings of the 21st International Cartographic Conference (ICC), Durban, South Africa.

Kraak, M.-J. 2008. Geovisualization and time – New opportunities for the space-time cube. Pages 293-318 *in* M. Dodge, M. McDerby, and M. Turner (eds.), Geographic visualization: Concepts, tools, and applications. Wiley, West Sussex, England.

Kroemer, K.H.E., H.J. Kroemer, and K.E. Kroemer-Elbert, 1997. Engineering physiology: Bases of human factors/ergonomics, 3rd Ed. Van Nostrand Reinhold, New York.

Lakshmanan, V. 2012. Automated analysis of spatial grids: Motivation and challenges. *Geotechnologies and the Environment* 6: 1-18.

Langran, G. and N.R. Chrisman. 1988. A framework for temporal geographic information. *Cartographica* 25(3): 1-14.

Langran, G. 1992. Time in geographic information systems. Taylor & Francis, Bristol, PA.

Le, Y. 2012. Challenges in data integration for spatiotemporal analysis. *Journal of Map & Geography Libraries: Advances in Geospatial Information, Collections & Archives* 8(1): 58-67.

Leone, F., F. Lavigne, R. Paris, J-C. Denain, and F. Vinet. 2011. A spatial analysis of the December 26th, 2004 tsunami-induced damages: Lessons learned for a better risk assessment integrating buildings vulnerability. *Applied Geography* 31(1): 363-375.

Li, X. and M.-J. Kraak. 2012. Explore multivariable spatio-temporal data with the time wave: Case study on meteorological data. Pages 79-92 *in* Advances in spatial data handling and GIS. Lecture notes in geoinformation and cartography, part 3. Springer, Berlin.

Lime, S. 2008. MapServer. Pages 65-85 *in* G.B. Hall and M.G. Leahy (eds.), Open source approaches in spatial data handling. Advances in geographic information science 2. Springer-Verlag, Berlin.

Lindell, M.K. and C.S. Prater. 2010. Tsunami preparedness on the Oregon and Washington coast: Recommendations for research. *Natural Hazards Review* 11: 69-81.

Liu, P.L.-F., Y-S. Cho, M.J. Briggs, C.E. Synolakis, and U. Kanoglu. 1995. Run-up of solitary waves on a circular island. *Journal of Fluid Mechanics* 302: 259-285.

Liu, P.L.-F., S.-B. Woo, and Y.-S. Cho. 1998. Computer programs for tsunami propagation and inundation, http://ceeserver.cee.cornell.edu/pll-group/doc/comcot_technical_manual.pdf (last accessed 14 February 2012).

Liu, Y., Y. Takeuchi, and N. Okada. 2007. Multi-agent based collaborative modeling for flood evacuation planning – Case study of Nagata, Kobe. *Annals of Disaster Prevention Research Institute* 50(B): 241-249.

Liu, Y., N. Okada, and Y. Takeuchi. 2008. Dynamic route decision model-based multi-agent evacuation simulation – Case study of Nagata Ward, Kobe. *Journal of Natural Disaster Science* 28(2): 91-98.

Lohr, S. 2012. The age of big data. The New York Times, February 11, http://www.nytimes.com/2012/02/12/sunday-review/big-datas-impact-in-the-world.html (last accessed 20 September 2012).

Lynett, P.J. 2011. Tsunami inundation, modeling of. Pages 1008-1021 *in* R.A. Meyers (ed.), Extreme environmental events: Complexity in forecasting and early warning. Springer, New York.

Lynett, P.J. and P.L.F. Liu. 2011. Numerical simulation of complex tsunami behavior. *Computing in Science & Engineering* 13(4): 50-57.

Maantay, J., A. Maroko, and G. Culp. 2010. Using geographic information science to estimate vulnerable urban populations for flood hazard and risk assessment in New York City. Pages 71-97 *in* P.S. Showalter and Y. Lu (eds.), Geospatial techniques in urban hazard and disaster analysis. Springer, Netherlands.

MacEachren, A., M. Wachowicz, R. Edsall, D. Haug, and R. Masters. 1999. Constructing knowledge from multivariate spatiotemporal data: Integrating geographical visualization with knowledge discovery in database methods. *International Journal of Geographical Information Science* 13(4): 311-334.

MapServer. 2012. MapServer – Open source web mapping, http://www.mapserver.org (last accessed 20 September 2012).

Mark, D. 1998. Geospatial lifelines [abstract]. Page 12 *in* O. Günther, T. Sellis, and B. Theodoulidis (eds.), Integrating spatial and temporal databases. Dagstuhl Seminar Report No. 228.

Marks, K. and W. Smith. 2006. An evaluation of publicly available global bathymetry grids. *Marine Geophysical Research* 27(1): 19-34.

Mas, E., F. Imamura, and S. Koshimura. 2011. Modeling the decision of evacuation from tsunami based on human risk perception. Tohoku Branch Annual Meeting. Sendai: Japan Society of Civil Engineering (JSCE), http://www.tsunami.civil.tohoku.ac.jp/hokusai3/J/shibu/22/erick_11.pdf (last accessed 3 May 2012).

Mas, E., F. Imamura, and S. Koshimura. 2012. An agent based model for the tsunami evacuation simulation. A case study of the 2011 great East Japan tsunami in Arahama Town. *In* Joint Conference Proceedings: 9th International Conference on Urban Earthquake Engineering & 4th Asia Conference on Earthquake Engineering, Tokyo.

Mayer, R.E., M. Hegarty, S. Mayer, and J. Campbell. 2005. When static media promote active learning: Annotated illustrations versus narrated animations in multimedia instructions. *Journal of Experimental Psychology: Applied* 11(4): 256-265.

McCarthy, A.C., N.H. Hancock, and S.R. Raine. 2010. VARIwise: A general-purpose adaptive control simulation framework for spatially and temporally varied irrigation at sub-field scale. *Computers and Electronics in Agriculture* 70(1): 117-128.

McLachlan, J., D. Foster, S. Clayden, and S. Barry. 2005. Long-term forest and landscape dynamics. Pages 125-141 *in* D.R. Foster and J.D. Aber (eds.), Forests in time: The environmental consequences of 1,000 years of change in New England. Yale University Press, New Haven, CT.

Mennis, J., R. Viger, and C.D. Tomlin. 2005. Cubic map algebra functions for spatio-temporal analysis. *Cartography and Geographic Information Science* 32: 17-32.

Merati, N., E. Gica, and C. Chamberlin. 2007. Building tsunami analysis tools into a GIS workspace. *In* ESRI International User Conference Proceedings (UC1889), San Diego.

Merati, N., C. Chamberlin, C. Moore, V. Titov, and T.C. Vance. 2010. Integration of tsunami analysis tools into a GIS workspace – Research, modeling, and hazard mitigation efforts within NOAA's Center for Tsunami Research. Pages 273-294 *in* P.S. Showalter and Y. Lu (eds.), Geospatial techniques in urban hazard and disaster analysis. Springer, Netherlands.

Meteor. 2012. Meteor JavaScript application. http://www.meteor.com (last accessed 28 September 2012).

Microsoft LayerScape. 2012. Microsoft LayerScape, http://research.microsoft.com/en-us/projects/layerscape (last accessed 12 October 2012).

Miller, H. 1991. Modelling accessibility using space-time prism concepts within geographical information systems. *International Journal of Geographical Information Systems* 5(3): 287-301.

Miller, H. and Y. Wu. 2000. GIS software for measuring space-time accessibility in transportation planning and analysis. *GeoInformatica* 4(2): 141-159.

Miller, H. and S.A. Bridwell. 2008. A field-based theory for time geography. *Annals of the Association of American Geographers* 99(1): 49-75.

Misra, I., S.M. Moorthi, R.K. Gambhir, and R. Ramakrishnan. 2011. Evolutionary rapid development using open source framework for geospatial data processing. *Trends in Information Management* 7(1): 31-40.

Mountain, D. 2005. Visualizing, querying, and summarizing individual spatio-temporal behavior. Pages 181-200 *in* J. Dykes, A.M. MacEachren, M-J. Kraak (eds.), Exploring geovisualization. Elsevier, London.

Murphy, K. 2012. Tsunami debris: Huge dock washes up on Oregon coast. Los Angeles Times, 6 Jun 2012, http://articles.latimes.com/2012/jun/06/nation/la-na-nn-huge-dock-tsunami-20120606 (last accessed 20 July 2012).

Nagarajan, M., D. Shaw, and P. Albores. 2012. Disseminating a warning message to evacuate: A simulation study of the behaviour of neighbours. *European Journal of Operational Research* 220(3): 810-819.

NANOOS. 2012. NANOOS Visualization System (NVS), http://www.nanoos.org/nvs/nvs.php?section=NVS-Products-Tsunamis-Evacuation (last accessed 12 July 2012).

National Research Council. 2011. Tsunami warning and preparedness: An assessment of the U.S. tsunami program and the nation's preparedness efforts. The National Academies Press, Washington, D.C. 284 pp.

NDIIPP. 2012. Sustainability of digital formats: Band interleaved by line (BIL) image file, http://www.digitalpreservation.gov/formats/fdd/fdd000283.shtml (last accessed 06 September 2012).

Neteler, M. and H. Mitasova. 2002. Open source GIS: A GRASS GIS approach. 3rd edition. Springer, New York.

Nginx. 2012. Nginx – a free, open-source, high-performance HTTP server, http://www.nginx.org (last accessed 20 September 2012).

Nguyen, T.T. 2009. Indexing PostGIS databases and spatial query performance evaluations. *International Journal of Geoinformatics* 5(3): 1-9.

Niazi, M.A. and A. Hussain. 2011. A novel agent-based simulation framework for sensing in complex adaptive environments. *IEEE Sensors Journal* 11(2): 404-412.

NOAA. 2012. Japan tsunami marine debris. NOAA Marine Debris Program, http://marinedebris.noaa.gov/info/pdf/japanfaq.pdf (last accessed 10 February 2012).

NOAA Center for Tsunami Research. 2012. NOAA Center for Tsunami Research: Tsunami Modeling and Research, http://nctr.pmel.noaa.gov/model.html (last accessed 14 February 2012).

NumPy. 2012. NumPy – the fundamental package for scientific computing with Python, http://numpy.scipy.org (last accessed 20 September 2012).

OGC WCS. 2010. OGC® WCS 2.0 Interface Standard – Core. Open Geospatial Consortium document reference #OGC 09-111r3. http://portal.opengeospatial.org/files/?artifact_id=41437 (last accessed 20 September 2012).

OGR. 2012. OGR Simple Features Library, http://www.gdal.org/ogr (last accessed 20 September 2012).

Oikawa, Y. and T. Katada. 1999. A study on the effect of flood experience on the evacuation activity. *Journal of Japan Society for Natural Disaster Science* 18(1): 103-116.

Omira, R., M.A. Baptista, J.M. Miranda, E. Toto, C. Catita, and J. Catalão. 2010. Tsunami vulnerability assessment of Casablanca-Morocco using numerical modelling and GIS tools. *Natural Hazards* 54: 75-95.

Open Geospatial Consortium, Inc. 2010. OpenGIS® implementation standard for geographic information - simple feature access - part 2: SQL option. Reference number OGC 06-104r4, http://portal.opengeospatial.org/files/?artifact_id=25354 (last accessed 8 February 2012).

OpenLayers. 2012. OpenLayers: Free maps for the web, http://www.openlayers.org (last accessed 20 September 2012).

Openshaw, S. 1987. An automated geographical information system. *Environment and Planning A* 19: 431-436.

Openshaw, S., M. Charlton, C. Wymer, and A. Craft. 1987. A Mark 1 geographical analysis machine for the automated analysis of point data sets. *International Journal of Geographical Information Systems* 1(4): 335-358.

Oregon.gov. 2012. Department of Geology and Mineral Industries – Tsunami Evacuation Route Maps, http://www.oregon.gov/DOGAMI/earthquakes/Coastal/CoastalHazardsMain.shtml (last accessed 15 May 2012).

OWASE. 2012. Owase Dynamic Tsunami Hazard Map. Disaster Social Engineering Laboratory, Gunma University, Japan, http://dsel.ce.gunma-u.ac.jp/simulator/owase/en (last accessed 20 April 2012).

Papathoma, M. and D. Dominey-Howes. 2003. Tsunami vulnerability assessment and its implications for coastal hazard analysis and disaster management planning, Gulf of Corinth, Greece. *Natural Hazards and Earth System Sciences* 3: 733-747.

Papathoma, M., D. Dominey-Howes, Y. Zong, and D. Smith. 2003. Assessing tsunami vulnerability, an example from Herakleio, Crete. *Natural Hazards and Earth System Sciences* 3: 377-389.

Perl. 2012. Perl: Practical Extraction and Report Language, http://www.perl.org (last accessed 13 February 2012).

Peuquet, D.J. 1984. A conceptual framework and comparison of spatial data models. *Cartographica* 21(4): 66-113.

Peuquet, D.J. 1994. It's about time: A conceptual framework for the representation of temporal dynamics in geographic information systems. *Annals of the Association of American Geographers* 84(3): 441-461.

Peuquet, D.J. and N. Duan. 1995. An event-based spatiotemporal data model (ESTDM) for temporal analysis of geographical data. *International Journal of Geographical Information Systems* 9(1): 7-24.

Peuquet, D.J. 2001. Making space for time: Issues in space-time data representation. *GeoInformatica* 5(1): 11-32.

Peuquet, D.J. 2002. Representations of space and time. The Guilford Press, New York.

PostGIS. 2012. PostGIS – A spatial extension to PostgreSQL, http://postgis.refractions.net (last accessed 20 September 2012).

PostGIS Raster. 2012. PostGIS Raster – An ongoing project aiming at developing raster support in PostGIS, http://trac.osgeo.org/postgis/wiki/WKTRaster (last accessed 20 September 2012).

PostgreSQL. 2012. PostgreSQL relational database management system, http://www.postgresql.org (last accessed 20 September 2012).

Priest, G.R., C. Goldfinger, K. Wang, R.C. Witter, Y. Zhang, and A.M. Baptista. 2010. Confidence levels for tsunami-inundation limits in northern Oregon inferred from a 10,000-year history of great earthquakes at the Cascadia subduction zone. *Natural Hazards* 54: 27-73.

PRISM Data Explorer. 2012. PRISM Data Explorer, http://prismmap.nacse.org/nn/index.phtml (last accessed 20 September 2012).

PROJ.4. 2012. PROJ.4 Cartographic Projections Library, http://proj.osgeo.org (last accessed 21 January 2012).

Pyramid. 2012. Pyramid open source web application framework, http://www.pylonsproject.org/projects/pyramid (last accessed 18 August 2012).

QGIS. 2012. Quantum GIS – A user-friendly open source geographic information system, http://www.qgis.org (last accessed 20 September 2012).

R Project. 2012. The R project for statistical computing, http://www.r-project.org (last accessed 20 September 2012).

R/GRASS. 2012. Short introduction to geostatistical and spatial data analysis with GRASS and R statistical data language, http://grass.fbk.eu/statsgrass/grass_geostats.html (last accessed 20 September 2012).

Raubal, M., S. Winter, S. Teβmann, and C. Gaisbauer. 2007. Time geography for ad-hoc shared-ride trip planning in mobile geosensor networks. *ISPRS Journal of Photogrammetry and Remote Sensing* 62(5): 366-381.

Reese, S., B.A. Bradley, J. Bind, G. Smart, W. Power, and J. Sturman. 2011. Empirical building fragilities from observed damage in the 2009 South Pacific tsunami. *Earth-Science Reviews* 107(1-2): 156-173.

Schmidt, J., C. Piret, N. Zhang, B.J. Kadlec, D.A. Yuen, Y. Liu, G.B. Wright, and E.O.D. Sevre. 2010. Modeling of tsunami waves and atmospheric swirling flows with graphics processing unit (GPU) and radial bias functions (RBF). *Concurrency and Computation: Practice and Experience* 22: 1813-1835.

Schneider, P. 2011. Tsunami modeling for Seaside, Oregon. Pages 408-416 *in* Proceedings of the 2011 Solutions to Coastal Disasters Conference. American Society of Civil Engineers, Reston, VA.

Schultz, N. and M. Bailey. 2012. Using extruded volumes to visualize time-series datasets. Pages 127-148 *in* J. Dill, R. Earnshaw, D. Kasik, J. Vince, and P.C. Wong (eds.), Expanding the frontiers of visual analytics and visualization. Springer, London.

SciPy. 2012. SciPy – Scientific tools for Python, http://www.scipy.org (last accessed 20 September 2012).

Sharma, S, U.S. Tim, and S. Gadia. 2012. AutoConViz: Automating the conversion and visualization of spatio-temporal query results in GIS. *Geo-spatial Information Science* (in press). Available online first at http://dx.doi.org/10.1080/10095020.2012.714099 (last accessed 27 September 2012).

Shaw, S-L., H. Yu, and L.S. Bombom. 2008. A space-time GIS approach to exploring large individual-based spatiotemporal datasets. *Transactions in GIS* 12(4): 425-441.

Showstack, R. 2011. Oceanographer tracks marine debris from the Japan tsunami and other incidents. *Eos Transactions American Geophysical Union* 92(37): 306.

Siládi, V., L. Huraj, N. Polčák, and E. Vesel. 2012. A parallel processing of spatial data interpolation on computing cloud. Pages 193-198 *in* Proceedings of the Fifth Balkan Conference in Informatics (BCI '12). ACM, New York.

Simonovic, S.P. and S. Ahmad. 2005. Computer-based model for flood evacuation emergency planning. *Natural Hazards* 34: 25-51.

Snodgrass, R.T. 2000. Developing time-oriented database applications in SQL. Morgan Kauffman, San Francisco.

Sp R. 2012. Spatial data in R, http://r-spatial.sourceforge.net (last accessed 20 September 2012).

Suleimani, E. 2004. UAF Tsunami Code, http://tsunamiportal.nacse.org/documentation/UAF_TsunamiCode.pdf (last accessed 14 February 2012).

Suppasri, A., E. Mas, S. Koshimura, and K. Imai. 2012. Developing tsunami fragility curves from the surveyed data of the 2011 great East Japan tsunami in Sendai and Ishinomaki plains. Coastal Engineering Journal 54(1): 1250008(1-16).

Synolakis, C.E. and E.N. Bernard. 2006. Tsunami science before and beyond Boxing Day 2004. *Philosophical Transactions of the Royal Society* 364: 2231-2265.

Tang, Z., M.K. Lindell, C.S. Prater, and S.D. Brody. 2008. Measuring tsunami planning capacity on U.S. Pacific Coast. *Natural Hazards Review* 9(2): 91-100.

Tappister. 2012. Cannon Beach tsunami inundation and evacuation study 2010-2011, http://www.tappister.com/cannon-beach (last accessed 29 April 2012).

Tarbotton, C., D. Dominey-Howes, J.R. Goff, M. Papathoma-Kohle, F. Dall'Osso, and I.L. Turner. 2012. GIS-based techniques for assessing the vulnerability of buildings to tsunami: Current approaches and future steps. *Geological Society, London, Special Publications* 361: 115-125.

TCP. 2012. Tsunami Computational Portal, http://tsunamiportal.nacse.org (last accessed 28 April 2012).

TimeDB. 2012. TimeDB bitemporal relational DBMS, http://www.timeconsult.com/Software/Software.html (last accessed 24 July 2012).

Titov, V. and F.I. González. 1997. Implementation and testing of the Method of Splitting Tsunami (MOST) model. NOAA Tech. Memo. ERL PMEL-112 (PB98-122773), NOAA/Pacific Marine Environmental Laboratory, Seattle. 11 pp.

Titov, V., A.B. Rabinovich, H.O. Mofjeld, R.E. Thomson, and F.I. Gonzalez. 2005. The global reach of the 26 December 2004 Sumatra tsunami. *Science* 309: 2045-2048.

Toffetti, G., S. Comai, J.C. Preciado, and M. Linaje. 2011. State-of-the-art and trends in the systematic development of rich internet applications. *Journal of Web Engineering* 10(1): 70-86.

Tomlin, C.D. 1990. Geographic information systems and cartographic modeling. Prentice Hall, New York.

Tomlin, C.D. 1994. Map algebra: One perspective. *Landscape and Urban Planning* 30(1-2): 3-12.

Tøssebro, E. and M. Nygård. 2011. Representing topological relationships for spatiotemporal objects. *Geoinformatica* 15: 633-661.

Tufte, E.L. 1983. The visual display of quantitative information. Graphics Press, Cheshire, CT.

Tversky, B., J.B. Morrison, and M. Betrancourt. 2002. Animation: Can it facilitate? *International Journal of Human-Computer Studies* 57: 247-262.

Uno, K. and K. Kashiyama. 2008. Development of a simulation system for the disaster evacuation based on multi-agent model using GIS. *Tsinghua Science and Technology* 13(S1): 348-353.

Usery, E.L., J. Choi, and M.P. Finn. 2010. Modeling sea-level rise and surge in low-lying urban areas using spatial data, geographic information systems, and animation methods. Pages 11-30 *in* P.S. Showalter and Y. Lu (eds.), Geospatial techniques in urban hazard and disaster analysis. Springer, Netherlands.

USGS National Elevation Dataset. 2012. USGS National Elevation Dataset (NED), http://ned.usgs.gov (last accessed 20 September 2012).

Van Hemert, J., J. Koetsier, L. Torterolo, I. Porro, M. Melato, and R. Barbera. 2011. Generating web-based user interfaces for computational science. *Concurrency and Computation: Practice and Experience* 23: 256-268.

Van Ho, Q., P. Lundblad, T. Åström, and M. Jern. 2011. A web-enabled visualization toolkit for geovisual analytics. *Information Visualization* 11(1): 22-42.

Vance, T.C., N. Merati, S.M. Mesick, C.W. Moore, and D.J. Wright. 2007. GeoModeler: Tightly linking spatially-explicit models and data with a GIS for analysis and geovisualization. *In* Proceedings of the 15th ACM International Symposium on Advances in Geographic Information Systems (ACM GIS 2007), Seattle, WA, 7-9 November.

Wang, S. and Y. Liu. 2009. TeraGrid GIScience gateway: Bridging cyberinfrastructure and GIScience. *International Journal of Geographical Information Science* 23(5): 631–656.

Wen, T.-H., M.-H. Lin, and C.-T. Fang. 2012. Population movement and vector-borne disease transmission: Differentiating spatial-temporal diffusion patterns of commuting and noncommuting dengue cases. *Annals of the Association of American Geographers* 102(5): 1026-1037.

WestMap. 2012. WestMap – Climate analysis & mapping toolbox, http://www.cefa.dri.edu/Westmap (last accessed 20 September 2012).

Wolman, D. 2005. Reaching safe ground. Technology Review, 26 July 2005, http://www.technologyreview.com/business/14626 (last accessed 21 April 2012).

Wood, N.J. and J.W. Good. 2004. Vulnerability of port and harbor communities to earthquake and tsunami hazards: The use of GIS in community hazard planning. *Coastal Management* 32(3): 243-269.

Worboys, M.F. 1992. A model for spatio-temporal information. Proceedings: The 5th International Symposium on Spatial Data Handling 2: 602-611.

Yeh, H.H. 1991. Tsunami bore runup. *Natural Hazards* 4(2): 209-220.

Yeh, H. 2010. Gender and age factors in tsunami casualties. *Natural Hazards Review* 11: 29-34.

Yu, H. 2006. Spatial-temporal GIS design for exploring interactions of human activities. *Cartography and Geographic Information Systems* 33(1): 3-19.

Yu, H. 2008. Visualizing and analyzing activities in an integrated space-time environment: Temporal geographic information system design and implementation. *Transportation Research Record* 2024: 54-62.

Yu, H. and S.-L. Shaw. 2007. Revisiting Hägerstrand's time-geographic framework for individual activities in the age of instant access. Pages 103-118 *in* H.J. Miller (ed.), Societies and cities in the age of instant access. Springer, Netherlands.

Yu, H. and S.-L. Shaw. 2008. Exploring potential human activities in physical and virtual spaces: A spatio-temporal GIS approach. *International Journal of Geographical Information Science* 22(4): 409-430.

Yuan, M. 1996. Modeling semantics, temporal, and spatial information in geographic information systems. Pages 334-347 *in* M. Craglia and E.L. Usery (eds.), Geographic information research: Bridging the Atlantic. Taylor & Francis, London.

Yuan, M. 1999. Use of a three-domain representation to enhance GIS support for complex spatiotemporal queries. *Transactions in GIS* 3(2): 137-159.

Yuan, M. 2001. Representing complex geographic phenomena in GIS. *Cartography and Geographic Information Science* 28(2): 83-96.

Yuan, M., D.M. Mark, M.J. Egenhofer, and D.J. Peuquet. 2005. Extensions to geographic representations. Pages 129-156 *in* R.B. McMaster and E.L. Usery (eds). A research agenda for geographic information science. CRC Press, Boca Raton, FL.

Zerger, A. and S. Wealands. 2004. Beyond modelling: Linking models with GIS for flood risk management. *Natural Hazards* 33: 191-208.

Zhang, L. and J. Yi. 2010. Management methods of spatial data based on PostGIS. Pages 410-413 *in* Proceedings of the Second Pacific-Asia Conference on Circuits, Communications and System (PACCS), August 1-2.

Zhong, C., T. Wang, W. Zeng, and S.M. Arisona. 2012. Spatiotemporal visualization: A survey and outlook. Pages 299-317 *in* S.M. Arisona, G. Aschwanden, J. Halatsch, and P. Wonka (eds.), Communications in computer and information science, vol 242: Digital urban modeling and simulation. Springer, New York.

APPENDICES

Appendix A: Conversion of Gridded Bathymetry ASCII (xyz) Datasets to Spatial Database Tables

Gridded bathymetry and coastal topography datasets used in the Tsunami Computational Portal were typically obtained as ASCII text files in “xyz” (or similar) format. These text files contain space-delimited values of longitude, latitude, and seafloor depth in m. The ASCII text must be parsed into a format that can be ingested into the spatial database (with added information of a numeric ID value and Cartesian x and y grid values). The Perl script below reads xyz text files and converts them to a series of input statements that are written to a file, which can be ingested directly by the spatial database. The script also generates a GiST index to help maximize efficiency of spatial queries, and performs other table indexing and cleaning operations.

A modified version of this script that writes records directly to the database was also developed, but runs more slowly than the script below. The PostgreSQL COPY command was also explored as an alternative to INPUT statements, but COPY simply streams data and cannot process function calls (e.g., GeometryFromText() to create point objects). Many scripting options exist, but the script below sufficed for all grid loading operations.

```
#!/usr/bin/perl
#
# xyz2sql.pl
#
# This script parses space-delimited text files in xyz or yxz
# format, such as bathymetry data.  The text file will be parsed
# into an SQL file that can be ingested into a PostGIS-enabled
# PostgreSQL database by running a command such as:
#
#    psql –h <host> -d <your_database> -f <sql_output_file>
#
# as long as you have write permission on that database.
#
# This script also adds grid-based x and y values (0 0, 0 1, 0 2,
# etc). It requires a "unit" value to define how many rows in the
# text file constitute a unit (row or column) of the grid.  This is
# dependent upon your data, so you will need to manually take a
# look at your text file to determine this value.  In my case I was
# dealing with text files where a block of 500 text rows had a +/-
# constant latitude, with longitude increasing.  At the end of 500
# rows, the latitude changed to a more-or-less fixed value, and the
# longitude started over at the lowest value.  So, my "unit" value
# was 500.
#
# The script takes a number of command line arguments, which can be
# reviewed by running the script with no arguments.  The script
# creates the necessary table and geometry column, as well as a
# GiST index and indexes on the grid x & y columns. It also does a
# VACUUM ANALYZE on the table.

use strict;
use warnings;

# disable command buffering, so that 'Processing....done' works
$| = 1;

if(scalar(@ARGV) != 8) {
die "
  Usage: xyz2sql.pl <input file> <input format> <unit> <db> <table> <geom>
<srid> <output file>\n
  <input file>  Name of input text file (with path if outside current
directory)
  <input format>  Must choose one of [xyz|yxz|zyx|xyzxy|xyzyx]
        Note:   xyzxy and xyzyx refer to data like 311 0 580 128.34445
34.28819
                where x and y refer to grid coordinates.
                Grid coordinates MUST be the first xy pair, lat/lon the
last xy pair
  <unit> Number of rows that make up a discrete unit (usually defined by
latitude)
  <db>  Name of PostgreSQL database, which must be spatially enabled with
PostGIS
  <table>  Name of new table (cannot contain a decimal point)
  <geom>  Name of geometry column in the new table
  <srid>  Desired SRID for geometry (-1 == undefined)\n
  <output file>  Name of output sql file (with full path if outside current
directory)\n
  Example:
```

```
  ./xyz2sql.pl bathy_500.xyz xyz 776 tsunami bathy_500 geom -1
bathy_500.sql
";
}

if($ARGV[1] !~ /xyz|yxz|zyx|xyzxy|xyzyx/) { die "  WARNING: <format> must
be one of [xyz|yxz|zyx|xyzxy|xyzyx]!\n  Script did not run!\n"; }
if($ARGV[4] =~ /\./) { die "  WARNING: <table> cannot contain a decimal
point!\n  Script did not run!\n"; }

my $infile = $ARGV[0]; #input file
my $format = $ARGV[1]; #whether data is xyz, yxz, etc
my $unit = $ARGV[2]; #how many rows make up a 'unit'
my $db = $ARGV[3]; #db name
my $table = $ARGV[4]; #table name
my $geom = $ARGV[5]; #geometry column
my $srid = $ARGV[6]; #srid number (-1 == undefined)
my $outfile = $ARGV[7]; #output file

my @depth;
my @lat;
my @lon;
my @x;
my @y;
my $x;
my $y;
my $z;

open(IN, $infile) || die "Could not open '$infile' - $!";

my $inCount = 1;
print "\n  => Parsing $infile";
# parse and suck the text file into arrays
while(<IN>) {
        #my $line = $_ unless $_ =~ /^#/;
        my $line = $_;
        chomp($line);
        $_ =~ /^\s*([+-]?\d+\.?\d*[eE]?[+-]?\d*)\s+([+-]?\d+\.?\d*[eE]?[+-
]?\d*)\s+([+-]?\d+\.?\d*[eE]?[+-]?\d*)/;
        my $a = sprintf("%.10g", $1);
        my $b = sprintf("%.9g", $2);
        my $c = sprintf("%.6g", $3);
        # NOTE - these are just the formats I've seen so far...
        # more may need to be added
        if($format eq 'xyz') {
                push @lon, $a;
                push @lat, $b;
                push @depth, $c;
        } elsif($format eq 'yxz') {
                push @lon, $b;
                push @lat, $a;
                push @depth, $c;
        } elsif($format eq 'zyx') {
                push @lon, $c;
                push @lat, $b;
                push @depth, $a;
        } elsif($format eq 'xyzyx') {
                $line =~ /^(\d+)\s(\d+)\s(-?\d+\.?\d+?)\s(-?\d+\.\d+)\s(-
?\d+\.\d+)/;
                push @x, $1;
                push @y, $2;
                push @lon, $5;
                push @lat, $4;
                push @depth, $3;
        } elsif($format eq 'xyzxy') {
                $line =~ /^(\d+)\s(\d+)\s(-?\d+\.?\d+?)\s(-?\d+\.\d+)\s(-
?\d+\.\d+)/;
                push @x, $1;
                push @y, $2;
                push @lon, $4;
                push @lat, $5;
                push @depth, $3;
        }
        if($inCount % 100000 == 0) {print ".";}
        if($inCount % 1000000 == 0) {print $inCount;}
        $inCount++;
}
print "...done\n";

close(IN);

# count latitude values to get a total row count
my $count = scalar(@lat);

open(OUT, "> $outfile") || die "Could not open '$outfile' - $!";

# create table - currently creates x and y cols no matter what -
# we add x & y values below if they don't exist
my $sql1;
#if($format !~ /xyzxy|xyzyx/) {
#       $sql1 = "CREATE TABLE $table (id int, depth int);";
#} else {
        $sql1 = "CREATE TABLE $table (id int, x smallint, y smallint, depth
numeric);";
#}
print OUT "$sql1\n";

# create geometry column
#print "  => Adding geometry column ...";
print OUT "SELECT AddGeometryColumn('$table', '$geom', $srid, 'POINT',
2);\n";

# insert data
print "  => Printing $count rows to $outfile";
# x,y needs to start at 1,1 to work with ARSC's Fortran code
my $xpos = 0; #initialize at 0, will go to 1 at next increment
my $ypos = 1; #initialize at 1
for(my $i = 0; $i < $count; $i++) {
        if($i != 0 && $i % 100000 == 0) {print ".";}
        if($i != 0 && $i % 1000000 == 0) {print $i;}
        my $id = $i + 1; #so that id column begins at 1
        my $sql2;

        if($i != 0 && $i % $unit == 0) {
                $xpos = 0; #reset at each $unit rows - will go to 1 at next
increment
                $ypos++; #increment 1 for each $unit rows
        }
        $xpos++; #increment 1 for every row

        if($format !~ /xyzxy|xyzyx/) {
                $sql2 = "INSERT INTO $table (id, x, y, depth, $geom) VALUES
($i, $xpos, $ypos, $depth[$i], GeometryFromText('POINT($lon[$i] $lat[$i])',
$srid));";
        } else {
                $sql2 = "INSERT INTO $table (id, x, y, depth, $geom) VALUES
($i, $depth[$i], $x[$i], $y[$i], GeometryFromText('POINT($lon[$i]
$lat[$i])', $srid));";
        }
        print OUT "$sql2\n";
}
print OUT "COMMIT;\n";
print "...done\n";

# create GiST index
#print "  => Creating GiST index $table\_index ...";
print OUT "CREATE INDEX $table\_gist_index ON $table USING gist ($geom
gist_geometry_ops);\n";
print OUT "CREATE INDEX $table\_x_index ON $table (x);\n";
print OUT "CREATE INDEX $table\_y_index ON $table (y);\n";

# vacuum analyze to update table stats
#print "  => Performing VACUUM ANALYZE on table ...";
print OUT "VACUUM ANALYZE $table;\n";

close(OUT);

print "\n  Job finished\n\n";
```

Appendix B: PostgreSQL Database Functions for Calculating Parameters of Gridded Bathymetry Datasets

Two database functions were written in the PL/pgSQL procedural language to perform calculations that must be executed on each gridded bathymetry or topography dataset added to the spatial database. Because the calculations each require a large number of successive queries on the gridded datasets, the most efficient solution was to write them as database functions and run them using the database engine.

Maximum Allowable Time Step

In the TCP parameterization system, the maximum allowable time step for each gridded dataset must be known prior to job configuration, and prior to the dataset being used in model runs. The PL/pgSQL database function below calculates the maximum allowable time step based on a standard formula that can be summarized thusly:

```
    (spacing in m) / (wave velocity) = time to move wave
```

Spacing in m is dependent upon latitude and is calculated based on the maximum latitude represented by the bathymetry dataset. Wave velocity is dependent upon ocean depth and is calculated based on the maximum ocean depth value contained within the bathymetry dataset. Both max_lat and max_depth are queried dynamically within the database function and included in the calculations. Once complete, the calculated max_time_step values are written to a database table that contains metadata describing each gridded dataset.

```
--
-- This function calculates maximum allowable time step for
-- each gridded dataset.
-- It is called via max_time_step()
--
-- * delta T = (1855 * cos(radians(max_lat))*spacing/60) / sqrt(2 * g *
abs(max_depth))
-- * g = gravitational constant (9.8)
-- * use abs(max_depth) because our depths might be stored as negative
values whereas most modelers store depths as positive values
-- * need to use radians() with cos()
CREATE OR REPLACE FUNCTION max_time_step()
RETURNS VOID AS '
DECLARE
  max_lat NUMERIC;
  max_depth NUMERIC;
  spc NUMERIC;
  g CONSTANT FLOAT := 9.8;
  m CONSTANT INTEGER := 1855;
  result NUMERIC;
  curs1 refcursor;
  row RECORD;
BEGIN
  FOR row in SELECT db_table FROM input_grids WHERE max_allowable_time_step
IS NULL LOOP

    RAISE NOTICE ''processing %'', row.db_table;

    OPEN curs1 FOR EXECUTE ''SELECT max(y(geom)) FROM '' ||
quote_ident(row.db_table);
    FETCH curs1 into max_lat;
    CLOSE curs1;

    OPEN curs1 FOR EXECUTE ''SELECT min(depth) FROM '' ||
quote_ident(row.db_table) || '' where depth > -30000 '';
    FETCH curs1 into max_depth;
    CLOSE curs1;

    OPEN curs1 FOR EXECUTE ''SELECT spacing FROM input_grids WHERE db_table
= '''''' || quote_ident(row.db_table) || '''''''';
    FETCH curs1 into spc;
    CLOSE curs1;

    result := (m * cos(radians(max_lat))*(spc/60)) / sqrt(2 * g *
abs(max_depth));
```

```
    EXECUTE ''UPDATE input_grids SET max_allowable_time_step = '' ||
quote_literal(result) || '' WHERE db_table = '''''' ||
quote_ident(row.db_table) || '''''''';

  END LOOP;
  RETURN;
END;
' LANGUAGE 'plpgsql';
```

Number of Time Steps Required to Cross Grid Extent

In addition to the maximum allowable time step, the number of time steps required for a wave to cross the maximum grid extent dimension must also be calculated before a gridded dataset can be used in the TCP parameterization system. The parameter max_dist (i.e., the maximum grid extent dimension in m) is determined as the maximum of either the latitude or longitude extent across the grid, which are queried dynamically within the function. The max_dist value is divided by wave velocity to determine the total time required to cross the maximum grid extent. The resulting num_time_steps value is stored in a database table that contains metadata describing each gridded dataset and is used to limit model run parameterization to allowable values, so that a TCP user won't accidentally enter unacceptable values.

```
--
-- This function calculates the number of time steps required to
-- cross the max grid extent, for each gridded dataset.
-- It is called via num_time_steps()
--
-- * following gives us distance in meters:
--    y_dist = (maxy - miny) * 60 * 1861
--    x_dist = (maxx - minx) * 60 * 1855 * cos(radians(miny))
-- * total_time = max_dist/sqrt(g * abs(avg_depth))
-- * use abs(avg_depth) because our depths might be stored as negative
values whereas most modelers store depths as positive values
-- * need to use radians() with cos()
CREATE OR REPLACE FUNCTION num_time_steps()
```

```
RETURNS VOID AS '
DECLARE
    minx NUMERIC;
    miny NUMERIC;
    maxx NUMERIC;
    maxy NUMERIC;
    max_dist NUMERIC;
    avg_depth NUMERIC;
    result NUMERIC;
    y_dist NUMERIC;
    x_dist NUMERIC;
    ym CONSTANT INTEGER := 1861;
    xm CONSTANT INTEGER := 1855;
    g CONSTANT FLOAT := 9.8;
    row RECORD;
    curs1 REFCURSOR;
BEGIN
    FOR row in SELECT db_table FROM input_grids WHERE
time_to_cross_max_domain IS NULL LOOP

        RAISE NOTICE ''PROCESSING %'', row.db_table;

        OPEN curs1 FOR EXECUTE ''SELECT min(x(geom)) FROM '' ||
quote_ident(row.db_table);
        FETCH curs1 into minx;
        CLOSE curs1;
        --RAISE NOTICE ''minx = %'', minx;

        OPEN curs1 FOR EXECUTE ''SELECT min(y(geom)) FROM '' ||
quote_ident(row.db_table);
        FETCH curs1 into miny;
        CLOSE curs1;
        --RAISE NOTICE ''miny = %'', miny;

        OPEN curs1 FOR EXECUTE ''SELECT max(x(geom)) FROM '' ||
quote_ident(row.db_table);
        FETCH curs1 into maxx;
        CLOSE curs1;
        --RAISE NOTICE ''maxx = %'', maxx;

        OPEN curs1 FOR EXECUTE ''SELECT max(y(geom)) FROM '' ||
quote_ident(row.db_table);
        FETCH curs1 into maxy;
        CLOSE curs1;
        --RAISE NOTICE ''maxy = %'', maxy;

        OPEN curs1 FOR EXECUTE ''SELECT avg(depth) FROM '' ||
quote_ident(row.db_table) || '' where depth > -30000 '';
        FETCH curs1 into avg_depth;
        CLOSE curs1;
        --RAISE NOTICE ''avg_depth = %'', avg_depth;

        y_dist = (maxy - miny) * 60 * ym;
        x_dist = (maxx - minx) * 60 * xm * cos(radians(miny));
        --RAISE NOTICE ''y_dist = %'', y_dist;
        --RAISE NOTICE ''x_dist = %'', x_dist;

        IF y_dist > x_dist
        THEN
            result := y_dist/sqrt(g * abs(avg_depth));
        ELSE
            result := x_dist/sqrt(g * abs(avg_depth));
        END IF;
        result := round(result,0);
        RAISE NOTICE ''  result = %'', result;

        EXECUTE ''UPDATE input_grids SET time_to_cross_max_domain = '' ||
quote_literal(result) || '' WHERE db_table = '''''' ||
quote_ident(row.db_table) || '''''''';

    END LOOP;
    RETURN;
END;
' LANGUAGE 'plpgsql';
```

Appendix C: Selected PHP Functions and JavaScript Code Related to Spatial Data Handling in the Portal Interface

Subgrid Selection

When a relatively coarse-resolution grid is selected in the portal interface, any grids of finer resolution are made available for selection as subgrids; however, they are only made available if they meet the 5:1 or 3:1 grid spacing ratio requirement. The two functions below inspect metadata stored for each grid in the database, determine which grids of finer resolution can be made available for selection as aligned subgrids, and present the subgrid options to the user.

```
function getPossibleChildGrids( $parent_grid, $level, $indent = 1 )
{
  global $GRID_NAME_LIST;
  $grids = '';

  $query = 'SELECT * FROM input_grids ' .
     'WHERE enabled = true AND (region_id = ' . $parent_grid['region_id'] .
" OR region_id = 0) " .
     'AND ( spacing = ' . round(($parent_grid['spacing'] / 3),10) . ' OR
spacing = ' . round(($parent_grid['spacing'] / 5),10) . ' ) ' .
     'AND ( max_x > ' . $parent_grid['min_x'] . ' AND min_x < ' .
$parent_grid['max_x'] . ' AND ' .
           'max_y > ' . $parent_grid['min_y'] . ' AND min_y < ' .
$parent_grid['max_y'] . ' ) ' .
     'ORDER BY db_table';

  $child_results = queryDB( $query );

  while( $grid = pg_fetch_array( $child_results, null, PGSQL_ASSOC ) )
  {
    $grids .= '<tr>';

    for( $i = 0; $i < $indent * 2; $i++ )
    {
      $grids .= '<td width="3%" rowspan="2"> </td>';
    }

    $num = $grid['input_grids_id'];
    $grids .= "<td width=\"5%\" align=\"right\"><input type=\"radio\"
name=\"master_grid_id\" id=\"master_grid_id_$num\" value=\"$num\"/></td>\n"
.
```

```
        "<td colspan=\"" . (9 - $indent) . "\">" .
                          "<label for=\"master_grid_id_$num\"><b>" .
$grid['db_table'] . '</b></label>&nbsp;&nbsp;&nbsp;&nbsp;<a
href="http://tsunamiportal.nacse.org/metadata/' . $grid['metadata_file'] .
        '" target="_blank" style="font-size : 80%">Metadata</a><br/>';
    $grids .= "</td></tr><tr><td></td><td colspan=\"" . (9 - $indent) .
"\"><font style=\"font-size : 80%\">X-Range: " . $grid['min_x'] . ' to ' .
$grid['max_x'] . "<br/>\n" .
        "Y-Range: " . $grid['min_y'] . ' to ' . $grid['max_y'] . "<br/>\n"
.
        "Spacing: " . $grid['spacing'] . ' ' . $grid['units'] .
"</br></font>\n";
    $grids .= "</td></tr>\n";

    $GRID_NAME_LIST[] = $grid['db_table'];

    if( --$level > 0 )
      $grids .= getPossibleChildGrids( $grid, $level, $indent + 1 );
  }
  return $grids;
}


//  getAddSubgrid( $curr_grid )
//
//  Returns a string containing an 'Add Sub Grid' button and a list
//  of possible sub grid spacings to choose from.
//
//  $curr_grid - The Grid Object to display possible sub grid spacings for

function getAddSubgrid( $curr_grid )
{
  $output = '';

  // Obtain a reference to the array of possible spacings for the current
grid
  $spacings = $curr_grid->possible_spacings;
  // Get the tree depth of this grid node
  $node_depth = $curr_grid->getNodeDepth();

  // If there are any possible spacings
  if( count($spacings) > 0 )
  {
    // If this grid meets the sub grid nesting constraints:
    //  - The master grid can contain up to 4 sub grids
    //  - Sub grids of the master grid can contain only one sub grid
    //  - Sub grids of sub grids of the master grid cannot contain any
further sub grids
    if( ( $node_depth == 1 && count($curr_grid->child_grids) < 4) ||
        ( $node_depth == 2 && count($curr_grid->child_grids) < 1 ) )
    {
      // Setup a table to contain the button and spacing options
      $output = "<table width=\"100%\"><tr><td align=\"left\"
width=\"30%\">\n";
      // Setup the 'Add Sub Grid' button
      $output .= "<input type=\"submit\" name=\"add_new_grid[" .
$curr_grid->grid_id . "]\" value=\"Add Sub Grid\"
class=\"submit_small\"/>\n";
      $output .= "</td><td align=\"left\">\n";

      $spacing_checked = false;

      // Loop over each possible sub grid spacing
      foreach( $spacings as $spacing )
      {
        // Create a spacing id to be decoded by the GridController
        $spacing_id = $spacing['spacing'] . '=' . $spacing['units'] . '=' .
$spacing['db_table'];
        if( !$spacing_checked )
        {
          $checked = 'checked="checked"';
          $spacing_checked = true;
        }
        else
        {
          $checked = '';
        }

        // Create a radio button and label for this spacing
        $output .= "<input type=\"radio\" name=\"grid_spacing_" .
$curr_grid->grid_id . "\" id=\"grid_spacing_" . $curr_grid->grid_id .
$spacing['db_table'] . "\" value=\"$spacing_id\" $checked/>\n";
        $output .= "<label for=\"grid_spacing_" . $curr_grid->grid_id .
$spacing['db_table'] . "\">" . $spacing['db_table'] . "</label>\n";
      }
      $output .= "</td></tr></table>\n";
    }
    else
    {
      // This grid cannot contain any more sub grids.  Disallowed by the
nesting constraints.
      $output .= "No more sub grids allowed for this grid<br>\n";
    }
  }
```

Dynamically Calculating Estimated Simulation Time

As the portal user parameterizes a model run, they define the number of seconds simulated by each time step as well as the total number of time steps to calculate. With this information, the portal uses a mix of PHP and JavaScript to calculate the estimated amount of time that will be simulated by the model run. This number changes dynamically as the user adjusts their desired parameters.

```
function displayTotalSize()
{
  var total_points = <?php echo $m_grid->getTotalDataPoints(); ?>;
  var len_time_step = (document.getElementById('Len_Time_Step')).value;
  var num_time_steps = (document.getElementById('Num_Time_Steps')).value;
  var output_freq = (document.getElementById('Output_Frequency')).value;

  var hours_elapsed = (((len_time_step * num_time_steps) / 60) /
60).toFixed(3);

  if( parseFloat(len_time_step) > 0.0 && parseInt(num_time_steps) > 0 &&
parseInt(output_freq) > 0 )
  {
    var size = ( num_time_steps / output_freq ) * total_points * 4;
    var out = '<b class="size_estimator">Estimated total output &asymp; ' +
getNiceSize(size) + '</b><br/>' +
        '<span class="size_estimator" style="font-size : 80%;">Equivalent
of simulating ' + hours_elapsed +
        ' hrs<br/></span>';
    if( out != last_out )
    {
      (document.getElementById('size_estimator')).innerHTML = out;
      last_out = out;
    }
  }
  else
  {
    if( last_out != '' )
    {
      (document.getElementById('size_estimator')).innerHTML = '<br/><span
style="font-size : 80%;"><br/></span>';
      last_out = '';
    }
  }
}
```

Calculating and Drawing Grid Extents on Selection Map

Each user-defined grid extent is displayed on the selection map after the user clicks "Update Display" or proceeds to the next step in the interface. Latitude and longitude values are mapped to pixel coordinates, and the functions below draw the necessary lines to form the extent of each grid and apply a transparent shaded overlay to the extent box.

```
function drawGrid( $img )
{
  global $SEL_GRID;
  global $bg_color, $text_color, $range_color;
  $grid = $this->grid_info;

  $color = $text_color;
  $bg_color = $bg_color;

  if( isset($grid['overlapping']) && $grid['overlapping'] == true )
  {
    $color = colorallocate( $img, 204, 0, 0 );
  }

  if( !is_null($this->grid_bound) && !isset($grid['sibling_grid']) ) {
    $this->grid_bound->drawGrid($img);
  }

  if( !is_null($grid['min_x']) ) {

    if((isset($grid['overlapping']) and $grid['overlapping'] != true ) or
(!isset($grid['overlapping']))){
      imagefilledrectangle( $img, $grid['left'], $grid['top'],
$grid['left'] + $grid['x_length_scaled'], $grid['top'] +
$grid['y_length_scaled'], $bg_color );
    }
    imagerectangle( $img, $grid['left'], $grid['top'], $grid['left'] +
      $grid['x_length_scaled'],
      $grid['top'] + $grid['y_length_scaled'],
      $color );
    $grid['caption'] = $grid['grid_id'] . ': ' . $grid['name'];
    $caption_width = $this->font_width * strlen( $grid['caption'] );
    if( $caption_width > $grid['x_length_scaled'] )
    {
      $grid['caption'] = substr( $grid['caption'], 0,
      intval( $grid['x_length_scaled'] / $this->font_width ) - 2 ) . '..';
    }

    imagestring($img, $this->font_size, $grid['left'] + 3, $grid['top'] +
2, $grid['caption'], $color);

    $table_name_parts = explode( '_', $grid['db_table'] );
    $caption2 = '(' . $table_name_parts[count($table_name_parts) - 1] .
')';
    $caption_width = $this->font_width * strlen( $caption2 );
    if( $caption_width > $grid['x_length_scaled'] )
    {
      $caption2 = substr( $caption2, 0, intval( $grid['x_length_scaled'] /
$this->font_width ) - 3 ) . '..)';
    }

    if( $grid['y_length_scaled'] > (($this->font_height * 2) + 3) )
    {

      imagestring($img, $this->font_size, $grid['left'] + 3, ($grid['top']
+ 2) + $this->font_height + 1 ,
      $caption2, $color);
    }

  }
  $this->drawChildGrids( $img );
}

function drawChildGrids( $img )
{
  $child_grids = $this->child_grids;
  foreach( $child_grids as $idx => $child_grid )
  {
    $child_grid = $child_grids[$idx];
    $child_grid->drawGrid( $img );
  }
}
```

Appendix D: Exporting and Packaging Selected Grid Extent

After a model run is configured and submitted for processing, an automated system prepares the input configuration parameters and the associated gridded data. The gridded data must be exported in the binary format required by the model codes that will use the data on the computational system at ARSC.

Extracting Gridded Data as Binary Files

PostgreSQL supports the development of database functions via the libpq library, a C-based application programming interface. In the TCP system, where xyz point data must be converted to a specific binary format, a C-based program provided the most efficient means of exporting the data and performing the proper conversion.

```
# This code generates the output_bin_dfile() database
# function, which is used to export gridded TCP data
# in the binary format required by ARSC.
#
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <sys/stat.h>
#include <unistd.h>
#include "executor/spi.h"

typedef unsigned char byte;

extern int output_bin_dfile( text *tbl_name, text *out_file, int x_min, int
y_min, int x_max, int y_max, int flip_signs )
{
  FILE *fpout;
  char *tablename;
  char *outfilename;
  char outfilepath[256];
  char tmpPath[256] = "";
  bool flipSigns = ( flip_signs == 0 ? false : true );
  char query[512];
  unsigned int i, rows;
  int result;

  bool nullDepth = false;
  Datum datumDepth;
  float4 floatDepth;
  float bigendDepth;
  byte *lEnd = ((byte *) &floatDepth);
  byte *bEnd = ((byte *) &bigendDepth);
  byte cnt = 0;

  // Prepare tablename
  tablename = DatumGetCString(DirectFunctionCall1( textout,
PointerGetDatum( tbl_name ) ));

  // Prepare outfilename
  outfilename = DatumGetCString(DirectFunctionCall1( textout,
PointerGetDatum( out_file ) ));

  // Build the query statement
  sprintf( query, "SELECT depth::float4 FROM %s WHERE x >= %i AND x <= %i
AND y >= %i AND y <= %i ORDER BY y ASC, x ASC;", tablename, x_min, x_max,
y_min, y_max );

  // Open the output file for binary write access
  fpout = fopen(outfilename,"wb");

  if (fpout==NULL)
  {
    elog( ERROR, "Unable to open output file: '%s'", outfilename );
  }

  // Output file is open and ready, query is ready

  SPI_connect();

  // Execute the query
  result = SPI_exec( query, 0 );
  rows = SPI_processed;

  // If the SELECT statement worked, and returned more than zero rows
  if (result == SPI_OK_SELECT && rows > 0)
  {
    // Get the tuple (row) description for the rows
    TupleDesc tupdesc = SPI_tuptable->tupdesc;
    // Get pointer to the tuple table containing the result tuples
    SPITupleTable *tuptable = SPI_tuptable;

    // Loop over each row in the result set (tuple set)
    for( i = 0; i < rows; i++ )
    {
      // Get tuple (row) number i
      HeapTuple tuple = tuptable->vals[i];
      // Store a pointer to the depth value Datum on this row
      datumDepth = SPI_getbinval( tuple, tupdesc, 1, &nullDepth );
      floatDepth = DatumGetFloat4( datumDepth );
      if( nullDepth )
        elog ( ERROR, "NULL depth value on row %i", i );

      if( flipSigns )
        floatDepth *= -1.0;

      // Write the little-endian floatDepth into bigendDepth
      bEnd += 3;
      for( cnt = 0; cnt < 4; cnt++ )
      {
        *bEnd = *lEnd;
        if( cnt < 3 ) {
          lEnd++;
          bEnd--;
        }
      }
      lEnd -= 3;

      // Write the floating point depth value out to the file
      fwrite(&bigendDepth,sizeof(float),1,fpout);
    }
  }

  // Done using the result set
  SPI_finish();

  // Close the output file
  fclose(fpout);

  // CHMOD the file for access by group tportal
  int mode = ( S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP | S_IROTH );
  chmod( outfilename, mode );

  // Free up memory
        pfree(tablename);
        pfree(outfilename);

  return 0;
}
```

Database Export Function

The PL/pgSQL code below imports the grid export C code as a PostgreSQL database function. This function is called whenever a model run is configured and submitted, and is used for exporting gridded spatial data into the binary format required by the model codes at ARSC.

```
CREATE OR REPLACE FUNCTION output_bin_dfile( text, text, int4, int4, int4,
int4, boolean )
RETURNS int4
AS '/private/share/tcp/bin/libpgoutdfile.so'
LANGUAGE 'c';
```

Appendix E: Casualty Calculation Code for the Simulation Model

The casualty calculation code exists as part of a larger codebase that drives the simulation framework. This code is written in C++ and, once compiled, produces the command-line program that is called to determine casualty conditions for every living member of the simulated population at each time step. The casualty calculation takes measured anthropometric values into account, along with the simulated water height, velocity, and direction. The status of each evaluated individual is written to the spatial database. The code also generates a TIFF file for each time step from the tsunami inundation modeling output data. Each TIFF file represents the water depth at that time step of the tsunami simulation.

```
#include "CasualtyCalc.h"

void BuildVisuals(DataOutBlock<GByte> *inBuff, DataOutBlock<GByte>
*outBuff)
{
  int i,j,k;
  int v;
  int idx;
  GByte *buf = inBuff->GetSliceAt(0); //Both genders
  for (i=0;i<Row;i++){
    for(j=0;j<Col;j++){
      v=buf[i*Col+j];
      if (v == 0) {
        for (k=0;k<3;k++)
          outBuff->SetCellAt(j,i,k,(GByte)Zero[k]);
      }
      else {
        idx = (int)floor((float)v/10.); // color map index
        if (idx > 15) idx = 15;
        for (k=0;k<3;k++)
          outBuff->SetCellAt(j,i,k,(GByte)HSDColorMap[idx][k]);
      }
    }
  }
}

bool WriteOutFile(DataOutBlock<GByte> *DB, char *filename)
```

```
{
  char *finalName = new char[strlen(filename) + strlen(oPath) + 2]; // 1
for \0 and 1 for '/' just in case
  strcpy(finalName,oPath);
  if (oPath[strlen(oPath-1)]!='/')
    strcat(finalName,"/");
  strcat(finalName,filename);

  TiffByteWriter *tw = TiffWriterFactory::instanceOf(finalName,ex,DB);
  tw->setNoDataValue(0.);
  tw->WriteTiff();
  delete tw; // must do this explicitly or it won't get written
  delete finalName;
  return (true);
}

bool WriteOutFile(DataOutBlock<GInt16> *DB, char *filename)
{
  char *finalName = new char[strlen(filename) + strlen(oPath) + 2]; // 1
for \0 and 1 for '/' just in case
  strcpy(finalName,oPath);
  if (oPath[strlen(oPath-1)]!='/')
    strcat(finalName,"/");
  strcat(finalName,filename);

  TiffShortWriter *tw = TiffWriterFactory::instanceOf(finalName,ex,DB);
  tw->setNoDataValue(0.);
  tw->WriteTiff();
  delete tw; // must do this explicitly or it won't get written
  delete finalName;
  return (true);
}

bool ReadPeopleFile()
{
  FILE *fp;
  char buf[HSDMAXBUF];
  float x,y;
  float sx,sy;
  int   gid;
  float t;
  int srid;
  Location *l;
  Location *srl;
  int id;
  int idx = 0;
  Location S;

  assert((fp=fopen(People_infile,"r"))!=NULL);

  fgets(buf,HSDMAXBUF,fp);
  sscanf(buf,"%d %d\n",&NumPeople,&srid);
  people = new Person[NumPeople];
  fgets(buf,HSDMAXBUF,fp);
  while (!feof(fp)){
    sscanf(buf,"%d\t%f\t%f\t%f\t%f\t%d\t%f\n",&id,&y,&x,&sy,&sx,&gid,&t);
    l = new Location(x,y,srid);
    srl = new Location(sx,sy,srid);
    if ((ex->World2Screen(*l,S))==true) {
      people[idx].Set_Person(id,l,pt);
      RN->RoutePerson(&people[idx],srl,gid,t);
      idx++;
    }
    fgets(buf,HSDMAXBUF,fp);
  }

  printf("Read in %d people, kept %d\n",NumPeople,idx);
  NumPeople=idx;
  return (true);
}


bool ReadStructureFile()
{
  FILE *fp;
  char buf[HSDMAXBUF];
  float x,y;
  int srid;
  Location l;
  int gid;
  int hc;
  int idx = 0;
  Location S;

  assert((fp=fopen(Structure_infile,"r"))!=NULL);

  fgets(buf,HSDMAXBUF,fp);
  sscanf(buf,"%d %d\n",&NumStructures,&srid);
  structures = new Structure[NumStructures];

  fgets(buf,HSDMAXBUF,fp);
  while (!feof(fp)){
    sscanf(buf,"%d\t%d\t%f\t%f\n",&gid,&hc,&y,&x);
    l.Set_Location(x,y,srid);
    if ((ex->World2Screen(l,S))==true)
      structures[idx++].setStructure(gid,hc,l);
    fgets(buf,HSDMAXBUF,fp);
  }
  printf("Read in %d structures, kept %d\n",NumStructures,idx);
  NumStructures=idx;
  return (true);
}

void
usage(void)
```

```
{
  printf("av[1] - input H Series file\n"\
 "av[2] - input U Series file name\n"\
 "av[3] - input V Series file name\n"\
 "av[4] - input Bathy-Topot file name\n"\
 "av[5] - extent file name\n"\
 "av[6] - People file name\n"\
 "av[7] - Structures file name\n"\
 "av[8] - Number of time slices to iterate across\n"\
 "av[9] - Path to directory where output products will be dumped\n"\
 "av[10] - When to start people running (i.e. which time index)\n"\
 "av[11] - Output table name\n"\
 "av[12] - Road Network name\n"\
"\n\n");
exit(0);
}

bool InitializeOutputDir(void)
{
  DIR *dirp;

  if ((dirp=opendir(oPath))!=NULL)
    closedir(dirp);
  else {
    return((mkdir(oPath,0755)==0 ? true:false));
  }
  return(true);
}

 //av[1] - input H Series file\n"\
 //av[2] - input U Series file name\n"\
 //av[3] - input V Series file name\n"\
 //av[4] - input Bathy-Topot file name"\
 //av[5] - extent file name\n"\
 //av[6] - People file name\n"\
 //av[7] - Structures file name\n"\
 //av[8] - Number of time slices to iterate across\n"\
 //av[9] - Path to directory where output products will be dumped\n"\
 //av[10] - When to start people running (i.e. which time index)\n"\

bool Initialize(int ac, char **av)
{
  if (ac < 8) usage();

  H_infile = strdup(av[1]);
  U_infile = strdup(av[2]);
  V_infile = strdup(av[3]);
  BT_infile = strdup(av[4]);
  extent_infile = strdup(av[5]);
  People_infile = strdup(av[6]);
  Structure_infile = strdup(av[7]);
  Time = atoi(av[8]);
  oPath = strdup(av[9]);
  whenToRun = atoi(av[10]);
  Output_Table_Name = strdup(av[11]);
  Road_infile = strdup(av[12]);
  oPrefix = strdup("");

  assert(InitializeOutputDir());

  ex = new Extent(extent_infile);
  Row = ex->get_Row();
  Col = ex->get_Col();
  pm = new PeopleMover(ex);
  RN = new RoadNetwork(Road_infile);

  dbU = new DataBlock<float>(Row,Col,Time,U_infile);
  dbV = new DataBlock<float>(Row,Col,Time,V_infile);
  dbH = new DataBlock<float>(Row,Col,Time,H_infile);
  dbBT = new DataBlock<float>(Row,Col,1,BT_infile);

  pt = new PopTable();

  ReadPeopleFile();

  dbpr = new DBPopRecorder(Output_Table_Name);

  return (true);
}

char *MakeFilename(char *base, int num)
{
  char *name = new char[strlen(base)+5+strlen(oPrefix)+1+4]; //length of
base + 5 digits + length of prefix + 4 for .tif + 1 for null
  sprintf(name,"%s%s%05d.tif",oPrefix,base,num);
  return (name);
}

void
CalcCasualty(Person *p,int sx, int sy, float *gh, float *gu, float *gv,
float *gbt, int pidx)
{
  PopRow *pr = p->get_ptr();

  float u;
  float v;
  float h;
  float speed;
  float tol=1e-6;

  if (isnan(gh[sy*Col+sx])) return; // no water...no casualty
  h = gh[sy*Col+sx]/100.-(gbt[sy*Col+sx]*-1.);
  if (h < tol) return; // water is so low it's in the noise
  if (h < p->get_height()) return; // the person is on something (bridge)
above the water
```

```
  if (isnan(gu[sy*Col+sx])) u = 0.; // no U speed;
  else  u = gu[sy*Col+sx]/100.; // convert cm/sec to m/sec
  if (isnan(gv[sy*Col+sx])) v = 0.; // no V speed;
  else v = gv[sy*Col+sx]/100.; // convert cm/sec to m/sec

  speed = sqrt(u*u+v*v);

  if (speed < tol) return;  //water speed ~= 0...no casualty

  float a = pr->get_a();
  float b = pr->get_b();
  float c = pr->get_c();
  float d = pr->get_d();
  float e = pr->get_e();
  float f = pr->get_f();
  float hb = pr->get_hb();
  float hc = pr->get_hc();
  float hg0 = pr->get_hg0();
  float x = pr->get_x();
  float w = pr->get_weight()*9.81;
  float A,Ay0,V,um,uf,minu;
  float wsv;

  if (h < hb){
    A = a*h+(b-a)*((h*h)/(2.*hb));
    Ay0 = ((a/2.)*(h*h)+((b-a)/(3*hb))*powf(h,3.));
    V = h * 3.1415926 * ((a + (b - a) * (h/hb))/2.) * ( (d + (e-d) *
(h/hb))/2.0);
  } else {
    A = ((a + b)/2.)*hb+b*(h-hb)+((c-b)/(2.*(hc-hb)))*(h-hb)*(h-hb);
    Ay0 = ((a/2.)*hb*hb+((b-a)/3.)*hb*hb+(1./(6.*(hb-hc)))*(3*c*h*h*hb-
(2*b+c)*powf(hb,3.)+3*b*hb*hb*hc+h*h*(2.*(b-c)*h-3.*b*hc)));
    V = hb * 3.1415926 * ((a + (b - a) * (h/hb))/2.) * ( (d + (e-d) *
(h/hb))/2.0) + (h-hb)*3.1415926 * ((b+(c-b) * ((h-hb)/(hc-hb)))/2.) *
((e+(f-e)*((h-hb)/(hc-hb)))/2.0);
  }

  if (w < (sigma*V)) {
    Location currentPersonLocation = p->get_location();
    p->set_dead(true);
    return;
  }

  wsv = w-sigma*V;
  um = sqrt((2.*wsv*x)/(Cd*Density*Ay0));
  uf = sqrt((2.*ff*wsv)/(Cd*Density*A));
  minu = HSDMIN((um),(uf));

  if ((speed/minu) < 1.) {
    return;
  } else {
    p->set_dead(true);
  }
  return;
}

void RunSimulation(void)
{
  int t;
  int p;
  int TotalDead=0;
  int DeadThisRound;
  Location currentPersonLocation;
  Location  Screen;
  char  *deadTimeSliceFileName;
  char  *livingTimeSliceFileName;
  char  *waterTimeSliceFileName;
  char  *cummDeadFileName;
  char  *visualLivingFileName;
  char  *visualDeadFileName;
  int   x,y;
  float *gh,*gu,*gv,*gbt;
  int i;

  gbt = dbBT->GetSliceAt(0);

  printf("Starting Simulation with %d people\n",NumPeople);

  for(t = 0;t < Time; t++){
    if (TotalDead >= NumPeople) break;

    DeadThisRound=0;
    gh = dbH->GetSliceAt(t);
    gu = dbU->GetSliceAt(t);
    gv = dbV->GetSliceAt(t);
    printf("Calculating casualties for time index %d...",t);

    for(p = 0; p < NumPeople; p++){
      if (people[p].is_dead()) {
        continue;
      }

      if (people[p].is_safe()) continue;

      currentPersonLocation = people[p].get_location();
      if((ex->World2Screen(currentPersonLocation,Screen))==false){
        people[p].set_safe(true);
        continue;
      }
      x=(int)Screen.get_x();
      y=(int)Screen.get_y();

      CalcCasualty(&people[p],x,y,gh,gu,gv,gbt,p);

      dbpr->addRow(p,&people[p],t);
      if (people[p].is_dead()) {
        DeadThisRound++;
        TotalDead++;
      } else {
        if (t >= whenToRun)
          pm->MovePerson(&people[p]);
      }
    }
    printf("%d %s died\n",DeadThisRound,(DeadThisRound == 1 ? "person
has":"people have"));
  }

  dbpr->Flush();

  int deadStats=0;
  int liveStats=0;

  for(p = 0; p < NumPeople; p++){
    if (people[p].is_dead()) deadStats++;
    else liveStats++;
  }

  printf("Total People: %d\n\t Dead: %d (%5.2f%%)\n\tAlive: %d
(%5.2f%%)\n\n\n",
          NumPeople,deadStats,((float)deadStats/(float)NumPeople)*100.,
          liveStats,((float)liveStats/(float)NumPeople)*100.);
}

int
main(int ac, char **av)
{
  // Load global variables from CMD line or issue usage
  assert(Initialize(ac,av));
  RunSimulation();
}
```

Appendix F: Database Function for Automatically Calculating Simulation Run Summary Statistics

Following a simulation run, a set of summary statistics need to be calculated for use in the results web interface. This pl/pgSQL database function calculates the summary statistics and inserts them into a new table that is queried by the web interface. The function is run automatically at the end of a simulation run.

```
/*
  This PL/pgSQL function creates a stats table containing the number of
  simulated people alive at each time step index. The table is given the
  same name as the data table, with '_stats' appended.

  Inputs:
    maintbl: name of the table containing the data for the time series run
    ts: total number of time steps simulated (1-based)

  Example usage:
    SELECT create_stats_table('job_151', 500);
*/

CREATE OR REPLACE FUNCTION create_stats_table(maintbl VARCHAR, ts INTEGER)
  RETURNS VOID
AS $$

DECLARE
  curs1 refcursor;
  max INTEGER;
  ct INTEGER;
  endct INTEGER;
  newtbl VARCHAR := ($1 || '_stats');

BEGIN

  IF EXISTS (
      SELECT *
      FROM   pg_catalog.pg_tables
      WHERE  schemaname = 'public'
      AND    tablename  = quote_ident(newtbl)
     )
  THEN
     RAISE NOTICE 'ERROR: Table public.%', newtbl || ' already exists.
Exiting now.';
     EXIT;
  ELSE
```

```
     EXECUTE 'CREATE TABLE ' || quote_ident(newtbl) || ' (tidx int, alive
int)';
     RAISE NOTICE 'Created table %', newtbl;
  END IF;

  -- select max time step index contained in main data table (can be < ts)
  OPEN curs1 FOR EXECUTE 'SELECT max(tidx) FROM ' || quote_ident(maintbl);
    FETCH curs1 INTO max;
    CLOSE curs1;

  -- select number of people living at end of simulation (time index in
  -- main data table may end prior to full ts value, so this 'endct' var
  -- is used to fill in the count for those final values)
  OPEN curs1 FOR EXECUTE 'SELECT count(id) FROM ' || quote_ident(maintbl)
|| ' WHERE alive IS NULL';
    FETCH curs1 INTO endct;
    CLOSE curs1;

  FOR i IN 0..ts-1 LOOP
    IF max+1 <= i THEN
      EXECUTE 'INSERT INTO ' || quote_ident(newtbl) || ' VALUES (' || i ||
', ' || endct || ')';
    ELSE
      OPEN curs1 FOR EXECUTE 'SELECT count(id) FROM ' ||
quote_ident(maintbl) || ' WHERE tidx = ' || i || ' AND alive IS TRUE OR
alive IS NULL';
        FETCH curs1 INTO ct;
        CLOSE curs1;
      EXECUTE 'INSERT INTO ' || quote_ident(newtbl) || ' VALUES (' || i ||
', ' || ct || ')';
    END IF;
  END LOOP;

  RAISE NOTICE 'Finished';

END;
$$
LANGUAGE 'plpgsql';
```

Appendix G: Web Code for the Simulation Results Page, Map-based Visualization, and Tabular Data

After configuring and running a simulation model run, the user can view a web page that displays the animated, map-based visualization of the simulation results. The code that creates the results page contains a combination of PHP, HTML, and Javascript, and draws upon several open source packages (PHP, MapServer, OpenLayers, HighCharts, and Proj.4). This Appendix includes selected code from certain key files that construct the results page:

- db.php: Functions that connect to the spatial database to retrieve data
- rpc.php: Code that handles Javascript AJAX calls to return data to the client
- table.php: Data table popup web page
- result.js: OpenLayers mapping code and all of the related custom Javascript that drives the animation, timekeeping, charting, and other client-side functionality.

Selected Code from db.php

```
<?
  function getPersonInfo($dbh, $table, $lon, $lat, $tidx) {
    $query = "select pid, mid
              from $table
              where (tidx = $tidx OR alive = 'f')
              and ST_DWithin(ST_GeogFromText('SRID=4326;POINT($lon $lat)'),
geography(geometry), 50)
              order by ST_Distance_Sphere( ST_GeometryFromText('POINT($lon
$lat)',4326), geometry)
              limit 1";
    $result = pg_query($dbh, $query);
    if( pg_num_rows($result) == 0 ) {
      $out = "No person found within 100m of map click<br />\n";
    } else {
      $out = array();
      while( $row = pg_fetch_row($result) ) {
        $out[0] = $row[0];
        $row[1] == 1 ? $out[1] = 'Male' : $out[1] = 'Female';
      }
    }
    return $out;
  }

  // Query to populate google-viz-chart data table
  function getTimeSeries($dbh, $table, $pid) {
    $query = "SELECT id, tidx, alive, ST_X(geometry) AS lon, ST_Y(geometry)
              AS lat FROM $table WHERE pid = $pid ORDER BY tidx";
    $result = pg_query($dbh, $query);
    $rows = pg_num_rows($result);
    $col = 0;
    echo "data.addRows($rows);\n";
    while ($row = pg_fetch_array($result)) {
      $row['alive'] == 't' ? $alive = 'true' : $alive = 'false';
      echo "
data.setCell($col,0,".htmlspecialchars($row['id']).");\n";
      echo "
data.setCell($col,1,".htmlspecialchars($row['tidx']).");\n";
      echo "      data.setCell($col,2,".htmlspecialchars($alive).");\n";
      echo "
data.setCell($col,3,".htmlspecialchars(round($row['lon'],6)).");\n";
      echo "
data.setCell($col,4,".htmlspecialchars(round($row['lat'],6)).");\n";
      $col++;
    }
  }

  /* This version uses ogr2ogr, which supports retrieval of attributes into
     the GeoJSON format, and not just the geometry */
  function getPathPoints($table, $pid) {
    $filename = randString(12);
    $file = '/tmp/' . $filename . '.json';
    $util = "$conf['ogr2ogrPath']";
    $cmd = $util . ' -f "GeoJSON" ' . $file . ' PG:"dbname=<<redacted>>
user=<<redacted>> password=<<redacted>>" -sql "select
ST_Transform(geometry, 900913), tidx from ' . $table . ' where pid=' . $pid
. '"';
    exec($cmd, $output, $ret);
    if ($ret == 0) {
      return $filename;
    } else {
      return 'There was an error.';
    }
  }

?>
```

Selected Code from rpc.php

```
<?
$proc = $_GET['proc'];

if($proc == 'get_wave_heights') {
  $x = $_GET['x'];
  $y = $_GET['y'];
  $minx = $_GET['minx'];
  $miny = $_GET['miny'];
  $maxx = $_GET['maxx'];
  $maxy = $_GET['maxy'];
  $htfile = $_GET['htfile'];
  $extent = array($minx,$miny,$maxx,$maxy);
  $numsteps = $_GET['numsteps'];
  $rows = $_GET['rows'];
  $cols = $_GET['cols'];
  $geo_xy = projectPoint($x, $y, $current_proj, true);
  $img_xy = geoToPix($geo_xy[0], $geo_xy[1], $rows, $cols, $extent);
  $cmd = 'inc/read_sample ' . $cols . ' ' . $rows . ' ' . $numsteps . ' ' .
$htfile . ' ' . $img_xy[0] . ' ' . $img_xy[1];
  $data = exec($cmd);
  $data = str_replace('nan', '0', $data);
  echo json_encode($data, JSON_NUMERIC_CHECK);
} elseif($proc == 'get_path_data') {
  $x = $_GET['x'];
  $y = $_GET['y'];
  $table = $_GET['table'];
  $tidx = $_GET['tidx'];
  $person_info = getPersonInfo($dbh, $table, $x, $y, $tidx); // returns
pid, gender
  if( is_array($person_info) ) {
    $data = getPathPoints($table, $person_info[0]);
    echo json_encode($data);
  } else {
    $d = array(0);
    echo json_encode($d);
  }
} elseif($proc == 'get_count') {
  $table = $_GET['table'];
  $tidx = $_GET['tidx'];
  $data = getCount($dbh, $table, $tidx); // returns int count
  echo json_encode($data);
} else {
  error("Called non-existent procedure: ", $proc);
}

?>
```

Selected Code from table.php

```
<script type='text/javascript' src='https://www.google.com/jsapi'></script>
<script type='text/javascript'>
  var data, table;
  var tidx = <?echo $tidx;?>;

  google.load('visualization', '1', {packages:['table']});
  google.setOnLoadCallback(drawTable);

  function drawTable() {
    data = new google.visualization.DataTable();
    data.addColumn('number', 'Event ID');
    data.addColumn('number', 'Time Step');
    data.addColumn('boolean', 'Alive');
    data.addColumn('number', 'Longitude');
    data.addColumn('number', 'Latitude');
    <? if( is_array($person_info) ) {
         getTimeSeries($dbh, $table, $person_info[0]);
       } else {
         echo "No person exists close enough to your map click.  Please try
again.";
       }
    ?>
    table = new
google.visualization.Table(document.getElementById('popup_table'));
    table.draw(data, {width:500, showRowNumber:true});

    google.visualization.events.addListener(table, 'select',
selectHandler);

    function selectHandler() {
      var selection = table.getSelection();
      var item = selection[0];
      var ts = data.getFormattedValue(item.row, 1);
      var tsint = parseInt(ts);
      window.opener.jumpToTimeStep(tsint);
      window.opener.hiliteSegment(tsint);
    }

    // set initial state of table and space-time path on map
    var max = data.getNumberOfRows();
    if( tidx < max ) {
      table.setSelection([{row:tidx, column:null}]);
      //window.opener.hiliteSegment(tidx);
    }
  }
</script>
```

Selected Code from result.js

```
<script type="text/javascript">

/** Build water depth chart **/

  $(document).ready(function() {
    chart = new Highcharts.Chart({
      chart: {
        renderTo: 'chartdiv',
        type: 'area',
        borderColor: '#bbb',
        borderWidth: 2,
        borderRadius: 5,
        zoomType: 'x',
        resetZoomButton: {
          position: {
            x: -10,
            y: 10
          },
          relativeTo: 'chart'
        },
        events: {
          click: function(event) {
            jumpToTimeStep(Math.round(parseFloat(event.xAxis[0].value)));
          }
        }
      },
      title: {
        text: 'Water Depth at Selected Point Across All Time Steps'
      },
      xAxis: {
        title: {
          text: 'Time Step'
        }
      },
      yAxis: {
        title: {
          text: 'Water Depth (m)'
        }
      },
      loading: {
        labelStyle: {
          top: '45%'
        }
      },
      tooltip: {
        shared: true,
        crosshairs: true
      },
      plotOptions: {
        series: {
          lineWidth: 1
        },
        area: {
          marker: {
            enabled: false,
          }
        }
      },
      credits: {
        enabled:false
      },
      series: [{
        name: 'water depth (m)',
        showInLegend: false,
        data: []
      }]
    });
  });


  /** Set up map and related init **/

  function init() {

    updateCounter(tidx);
    newURL(tidx);

    debugE = document.getElementById('debug');

    var lon = -123.929;
    var lat = 45.996;
    var zoom = 9;

    var ctr_merc = OpenLayers.Layer.SphericalMercator.forwardMercator(lon,
lat);
    ctr_merc_x = ctr_merc.lon;
    ctr_merc_y = ctr_merc.lat;

    // openlayers
    map = new OpenLayers.Map('map', {
      units: "m",
      maxResolution: 156543.0339,
      maxExtent: new OpenLayers.Bounds(-20037508.34, -20037508.34,
20037508.34, 20037508.34),
      projection: epsg900913
      //displayprojection: epsg4326
    } );

    var ls = new OpenLayers.Control.LayerSwitcher();
    map.addControl(ls);

    // base layers
    gterrain = new OpenLayers.Layer.Google(
      "Google Terrain",
      { type: google.maps.MapTypeId.TERRAIN,
        maxZoomLevel: 15,
        minZoomLevel: 4,
        sphericalMercator: true
      });
    groadmap = new OpenLayers.Layer.Google(
      "Google Roadmap",
      { type: google.maps.MapTypeId.ROADMAP,
        minZoomLevel: 4,
        maxZoomLevel: 19,
        sphericalMercator: true
      });
    ghybrid = new OpenLayers.Layer.Google(
      "Google Hybrid",
      { type: google.maps.MapTypeId.HYBRID,
        minZoomLevel: 4,
        maxZoomLevel: 19,
        sphericalMercator: true
      });
    gsatellite = new OpenLayers.Layer.Google(
      "Google Satellite",
      { type: google.maps.MapTypeId.SATELLITE,
        minZoomLevel: 4,
        maxZoomLevel: 19,
        sphericalMercator: true
      });
    esri_ocean = new OpenLayers.Layer.ArcGIS93Rest(
      "ESRI Ocean Basemap",

"http://services.arcgisonline.com/ArcGIS/rest/services/Ocean_Basemap/MapSer
ver/export",
      { layers: "show:0",
        srs: "3857",
        format: "jpg"
      });

    // vector layers
    popl = new OpenLayers.Layer.MapServer(
      "Population",
      "http://newvole.nacse.org/cgi-bin/mapserv", {
        map: '/a1/www/mapfiles/hsd.map',
        layers: 'alive dead alldead',
        format: 'image/png',
        tidx: tidx
      },{
        singleTile: true,
        isBaseLayer: false,
      });
    runup = new OpenLayers.Layer.MapServer(
      "Wave Runup",
      "http://newvole.nacse.org/cgi-bin/mapserv", {
        map: '/a1/www/mapfiles/hsd.map',
        layers: 'runup',
        format: 'image/png',
        data: data
      },{
        singleTile: true,
        isBaseLayer: false,
      });

    var wavept_style = OpenLayers.Util.extend({},
OpenLayers.Feature.Vector.style['default']);
      wavept_style.fillOpacity = 1;
      wavept_style.strokeColor = "black";
      wavept_style.fillColor = "yellow";
      wavept_style.pointRadius = 4;
      wavept_style.strokeWidth = 2;

    wavept = new OpenLayers.Layer.Vector("Wave Chart Point", {
      style: wavept_style,
      isBaseLayer: false,
      displayInLayerSwitcher: true,
      numZoomLevels: gsatellite.numZoomLevels,
      resolutions: gsatellite.resolutions
    });

    runup.setOpacity(.6);

    runup.events.register( "loadend", runup, runupOnload );
    popl.events.register( "loadend", popl, poplOnload );

    map.addLayers([gterrain, groadmap, ghybrid, gsatellite, esri_ocean,
runup, popl]);
    map.addLayer( wavept );
    map.setBaseLayer(ghybrid);
    map.setCenter(new OpenLayers.LonLat(ctr_merc_x, ctr_merc_y), zoom);

    // Handle map click - wave chart vs. person data table
    map.events.register('click', map, function(e) {
      var xy = map.getLonLatFromViewPortPx(e.xy);
      if( $("input[name='mapclick']:checked").val() == 'table' ) { //
display data table
        var geo = getLatLon(xy);
        var call = 'table.php?table=' + table + '&lon=' + geo.x + '&lat=' +
geo.y + '&tidx=' + tidx;
        var config =
'height=700,width=600,scrollbars=yes,toolbar=no,menubar=no,location=no,dire
ctories=no,status=no';
        popwin = window.open(call, 'tablewin', config);
        pathHandler(table, geo.x, geo.y, tidx);
      } else { // update water depth chart
        chart.showLoading();
        getWaveHeights(xy, numsteps, rows, cols, htfile, minx, miny, maxx,
maxy);
        addPoint(xy);
      }
    });

    // initial message above chart
    $('#chartmsg').html(chart_msg_pre + '<span style="color:red;">(click
map to select point)</span>' + chart_msg_suf);
    addPlotLine(tidx+1);

    // starting condition for counter - alive_ct comes from
result/index.php
    $('#alivenum').html(alive_ct[0]);
    $('#totalnum').html(alive_ct[0]);
  }

  function isArray(a) {
    return Object.prototype.toString.apply(a) === '[object Array]';
  }

  function forward() {
    if(tidx < img_max) {
      tidx++;
      newURL(tidx);
      updateCounter(tidx);
      $('#alivenum').html(alive_ct[tidx]);
      redrawLayers();
      if( typeof(stpath_layer) != 'undefined' ) {
        hiliteSegment(tidx);
      }
    }
  }

  function back() {
    if(tidx > img_min) {
      tidx--;
      newURL(tidx);
      updateCounter(tidx);
      $('#alivenum').html(alive_ct[tidx]);
      redrawLayers();
      if( typeof(stpath_layer) != 'undefined' ) {
        hiliteSegment(tidx);
      }
    }
  }

  function redrawLayers() {
    runup.params.data = data;
    popl.params.tidx = tidx;

    if( runup.visibility ) {
      u = runup.grid[0][0].url;
      out = u.replace(/runup_\d{5}/, 'runup_'+img);
      runup.grid[0][0].imgDiv.src = out;
    }
    if( popl.visibility ) {
      popl_loading = 1;
      popl.redraw();
    }

    removePlotLine();
    addPlotLine(tidx);
  }

  function runupOnload() {
    runup_loading = 0;
    animateMap();
  }

  function poplOnload() {
    popl_loading = 0;
    animateMap();
  }

  function animateMap() {
    // hack to prevent OpenLayers from displaying lingering pop'l layer
    // on zoom event during animation
    if(animating) {
      if(popl_loading == 0 && runup_loading == 0) {
        $('#OpenLayersDiv181').hide();
        forward();
      }
    }
  }

  function updateCounter(imageNum) {
    count = document.getElementById('count');
    // Note: the text below makes the count one-based, for user ease
    //count.innerHTML = '<b>'+(tidx+1)+'</b> of <b>'+readable_max+'</b>';
    count.innerHTML = '<b>'+(tidx)+'</b> of <b>'+numsteps+'</b>';
  }

  // this is ONLY used to toggle layer status based on user click
  function layerControl(clickedLayer) {
    if( layerStatus[clickedLayer] == true ) {
      layerStatus[clickedLayer] = false;
    } else {
      layerStatus[clickedLayer] = true;
    }

    layerString()
    newURL(tidx);
  }

  // build new LAYERS string to go in mapserv CGI call
  function layerString() {
    // reset LAYERS string
    layersOn = 'LAYERS=';
    for( i in layerStatus )
    {
      if( layerStatus[i] == true )
      {
        layersOn = layersOn + i + " ";
      }
    }
  }

  function newURL(imageNum) {
    img = sprintf('%05d', imageNum);
    data = model+"/"+run+"/runup_"+img+".tif";
  }

  function imageJump(selectObj) {
    var idx = selectObj.selectedIndex;
    var dest = selectObj.options[idx].value;
    tidx = Math.round(parseFloat(dest));
    newURL(dest);
    updateCounter(dest);
    $('#alivenum').html(alive_ct[dest]);
    if( typeof(stpath_layer) != 'undefined' ) {
      hiliteSegment(idx);
    }
    redrawLayers();
  }

  function jumpToTimeStep(num) {
    tidx = num;
    newURL(num);
    updateCounter(num);
    $('#alivenum').html(alive_ct[num]);
    redrawLayers();
  }

  function addPlotLine(ts) {
    chart.xAxis[0].addPlotLine({
      value: ts,
      color: 'rgb(255, 0, 0)',
      width: 1,
      id: 'waveline'
    });
  }

  function removePlotLine() {
    chart.xAxis[0].removePlotLine('waveline');
  }

  function buttonHandler(imageID,button) {
    test = document.getElementById(imageID);
    test.src = "images/" + button;
    return true;
```

```
}

function getLatLon(xy) {
  var src = new Proj4js.Proj('EPSG:900913');
  var dst = new Proj4js.Proj('EPSG:4326');
  var pt = new Proj4js.Point(xy.lon, xy.lat);
  Proj4js.transform(src, dst, pt);
  return pt;
}

function addPoint(xy) {
  wavept.removeAllFeatures();
  var point = new OpenLayers.Geometry.Point(xy.lon,xy.lat);
  wavept_feature = new OpenLayers.Feature.Vector(point);
  wavept.addFeatures([wavept_feature]);
}

function roundNum(num, dec) {
  var newnum = Math.round(num*Math.pow(10,dec))/Math.pow(10,dec);
  return newnum;
}

function pathHandler(table, x, y, tidx) {
  getPathData(table, x, y, tidx);
}

function removeSTPath() {
  if( typeof(stpath_layer) != 'undefined' ) {
    map.removeLayer(stpath_layer);
    map.removeLayer(stendpt_layer);
  }
}

function drawSTPath(geojson) {
  var features = [];
  var data = new OpenLayers.Format.GeoJSON();
  var file = '/temp/' + geojson + '.json';

  // get geojson output file from handler ajax call to ogr2ogr
  OpenLayers.Request.GET({
    url: file,
    async: false,
    success: function(r){
      features = data.read(r.responseText);
    },
    failure: function(x){
      alert('Could not retrieve path data.');
    }
  });

  stpath_layer = new OpenLayers.Layer.PointTrack('Space-Time Path', {
    isBaseLayer: false
    // styles applied below, per segment
    });

    map.addLayer(stpath_layer);
    stpath_layer.addNodes(features);

    // alternate path segment colors to denote movement per time step
    var c, ct = features.length;
    for(i = 0; i < (ct-1); i++) {
      i % 2 != 0 ? c = '#ff1493' : c = '#000078';
      stpath_layer.features[i].style = OpenLayers.Util.applyDefaults({
        strokeColor: c,
        strokeWidth: 5,
        strokeOpacity: 1.
      });
    }

    stpath_layer.redraw();

    // hilite path segment for current time step
    if( tidx < stpath_layer.features.length ) {
      hiliteSegment(tidx);
    }

    // create styled start/end points for path

    stendpt_layer = new OpenLayers.Layer.Vector('Path Start/End Points');
    stendpt_layer.addFeatures([
      new OpenLayers.Feature.Vector(features[0].geometry),
      new OpenLayers.Feature.Vector(features[ct-1].geometry)
    ]);
    stendpt_layer.features[0].style = OpenLayers.Util.applyDefaults({
      strokeColor: '#00ff00',
      strokeWidth: 3,
      pointRadius: 9
    }, stendpt_layer.styleMap.styles.default.defaultStyle);
    stendpt_layer.features[1].style = OpenLayers.Util.applyDefaults({
      strokeColor: '#ff0000',
      strokeWidth: 3,
      pointRadius: 9
    }, stendpt_layer.styleMap.styles.default.defaultStyle);
    stendpt_layer.redraw();
    map.addLayer(stendpt_layer);

  }

  // Hilite currently selected time step
  function hiliteSegment(tidx) {
    var segstyle = OpenLayers.Util.extend({},
OpenLayers.Feature.Vector.style['default']);
      segstyle.strokeColor= '#ffe303';
      segstyle.strokeWidth= 5;
      segstyle.fontColor= '#0ff';
      segstyle.fontSize= '14px';
      segstyle.fontWeight= 'bolder';
      segstyle.labelXOffset= 15;
      segstyle.labelYOffset= 10;
      segstyle.label= $.trim(tidx);

    if( typeof(prev_tidx) != 'undefined' ) { // restore color of
previously-selected segment
      stpath_layer.features[prev_tidx].style.strokeColor = prev_color;
      stpath_layer.features[prev_tidx].style.label = false;
    }

    if( typeof(stpath_layer.features[tidx]) != 'undefined' ) {
      prev_color = stpath_layer.features[tidx].style.strokeColor;
      prev_tidx = tidx;
      //stpath_layer.features[tidx].style.strokeColor = '#ffe303';
      stpath_layer.features[tidx].style = segstyle;
      stpath_layer.redraw();
    }
  }

/** AJAX calls **/

  function getWaveHeights(xy, numsteps, rows, cols, htfile, minx, miny,
maxx, maxy){
    $.getJSON("../rpc.php?proc=get_wave_heights&x=" + xy.lon + "&y=" +
xy.lat + "&numsteps=" +
              numsteps + "&rows=" + rows + "&cols=" + cols + "&htfile=" +
htfile + "&minx=" +
              minx + "&miny=" + miny + "&maxx=" + maxx + "&maxy=" + maxy,
      function(data) {
        var cd = $.parseJSON(data); // convert to js array object
        var cd_cm = new Array;
        $.each(cd, function(idx, val) {
          cd_cm[idx] = roundNum((val/100), 2); // convert cm to m, round to
2 decimal places, add to new array
        });
        chart.series[0].setData(cd_cm); // assign data series in chart
        var geo = getLatLon(xy);
        $('#chartmsg').html(chart_msg_pre + Math.round(geo.x * 10000)/10000
+
                            ', ' + Math.round(geo.y * 10000)/10000 +
chart_msg_other +
                            chart_msg_suf); // update lat/lon coords above
chart
        chart.hideLoading(); // hide loading state
      }
    );
  }

  function getPathData(table, x, y, tidx){
    $.getJSON("../rpc.php?proc=get_path_data&x=" + x + "&y=" + y +
"&table=" + table + "&tidx=" + tidx,
      function(data) {
        var pd = data;
        isArray(pd) ? alert('Map click was not close enough to a person.')
: drawSTPath(pd);
      }
    );
  }
```

Appendix H: Sample Area-based GridServer Method

The GridServer includes a number of method functions, each written to perform a specific type of processing task. Area-based GridServer requests are more complex than point-based (i.e., single grid cell) requests, requiring validation of the bounding box or spatial mask, clipping, statistical calculation(s), and multiple file output (most of those operations exist in other functions not displayed here). Both the "temporal first" and "spatial first" calculations are performed in every area-based GridServer method function. The Python function below processes requests for statistical calculations a range of months for a defined region.

```
def _stats_area_months(self, request,
                       start=None,
                       end=None,
                       pstat=None,
                       param=None):

  """
  Statistic calculated across area over time (months).
  Returns a result grid (or grids) and data values as output.
  """

  log = request.logger
  start = start or request.start
  end = end or request.end
  param = param or request.param
  pstat = pstat or request.pstat

  ret = {param: {}}

  log.info("  _stats_area_months   start:%s end:%s" % (
start.strftime("%Y/%m"), end.strftime("%Y/%m")))

  # clip bounds
  self.init_bounds(request)

  # get keys for requested monthly grids
  monthly_keys = self.get_monthly_keys( start, end )
  log.debug("    monthly_key handles: %s" % ",".join( monthly_keys ))
  if not self.validate_keys(request, "monthlies", param, monthly_keys):
    log.error("Could not get monthly data")
  else:
    monthly_array = self.get_clipped_mdarray(request, "monthlies", param,
monthly_keys)
    if monthly_array is None:
      log.error("Could not make clipped monthly array")
    else:
      if request.spatial_mask is not None:
        monthly_array[:,request.spatial_mask] = numpy.nan

        if pstat == 'pca':
          pca_grids, pca_eigval = self.get_pca_grids(monthly_array)
          pca_outfiles = []
          for i in range(len(pca_grids)):

            if request.spatial_mask is not None:
              pca_grids[i][request.spatial_mask] = numpy.nan

            name = request.gricket + "_pca" + str(i+1)
            new_filename = name + ".bil"
            new_path = os.path.join( request.output_dir, new_filename )
            new_ds = self.array_2_dataset( pca_grids[i], request.bounds,
new_path )
            pca_outfiles.append( os.path.join(request.gricket,
new_filename) )
          ret["pca_grids"] = pca_outfiles
          ret["pca_eigval"] = pca_eigval
        else:
          spatial_vals = self.get_spatial_values(param, pstat,
monthly_array)
          ret = {param: self._stats_param_output(spatial_vals, param)}
          temporal_grid = self.get_temporal_grid(pstat, monthly_array)

          if request.spatial_mask is not None:
            temporal_grid[request.spatial_mask] = numpy.nan

          name = request.gricket + "_" + pstat
          new_filename = name + ".bil"
          new_path = os.path.join( request.output_dir, new_filename )
          new_ds = self.array_2_dataset(temporal_grid, request.bounds,
new_path )
          ret["grid"] = os.path.join( request.gricket, new_filename )
        cells_used = numpy.count_nonzero(~numpy.isnan(monthly_array[0]))
        ret["extent"] = [request.bounds.leftlon, request.bounds.toplat,
                         request.bounds.rightlon, request.bounds.bottomlat]
        ret["size"] = [request.bounds.cols, request.bounds.rows]
        ret["cells_used"] = cells_used
        return ret
```

Appendix I: GridServer Area-based Statistical Calculations

The GridServer calculates area-based statistics using the “temporal first” and “spatial first” methods. In the “temporal first” method (essentially a form of multidimensional map algebra), the output product is a grid. In the “spatial first” method, the statistic is calculated across all cells in each grid, resulting in a single output value per grid. The array of values produced by that process is the output product. The two Python functions below each get called as part of an area-based GridServer request.

```
def get_spatial_values(self, param, pstat, data):
  ### Computes mean across each grid area and returns the resulting set
  ### of values.
  ### AKA "spatial first"

  if param == 'ppt':
    rnda = 3
    rndb = 2
    rndc = 1
  else:
    rnda = 2
    rndb = 1
    rndc = 1

  spatial_vals = []

  # convert nan's if they exist (ie, bbox is partly over ocean)
  # this approach uses a numpy masked array to mask the nan's

  mdata = numpy.ma.masked_array(data, numpy.isnan(data))

  for i in mdata:
    if( pstat == 'mean' ):
      spatial_vals.append(round(numpy.ma.mean(i), rnda))
    elif( pstat == 'min' ):
      spatial_vals.append(round(numpy.ma.min(i), rndb))
    elif( pstat == 'max' ):
      spatial_vals.append(round(numpy.ma.max(i), rndb))
    elif( pstat == 'median' ):
      spatial_vals.append(round(numpy.ma.median(i), rndb))
    elif( pstat == 'stddev' ):
      spatial_vals.append(round(numpy.ma.std(i, ddof=1), rndc))
  return spatial_vals


def get_temporal_grid(self, pstat, data):
  ### Computes mean thru all columns and returns a new grid (ie, map
  ### algebra).
  ### AKA "temporal first"
  ### axis=0 computes thru columns

  # convert nan's if they exist (ie, bbox is partly over ocean)
  # this approach uses a numpy masked array to mask the nan's

  fill_val = -9999.
  mdata = numpy.ma.masked_array(data, numpy.isnan(data))

  if( pstat == 'mean' ):
    grid = numpy.ma.mean(mdata, axis=0)
  elif( pstat == 'min' ):
    grid = numpy.ma.min(mdata, axis=0)
  elif( pstat == 'max' ):
    grid = numpy.ma.max(mdata, axis=0)
  elif( pstat == 'median' ):
    grid = numpy.ma.median(mdata, axis=0)
  elif( pstat == 'stddev' ):
    grid = numpy.ma.std(mdata, axis=0, ddof=1)

  outgrid = grid.filled(fill_val)
  return outgrid
```

Appendix J: PCA Calculation in Python

The GridStats system allows PCA to be performed across a set of PRISM climate grids for a defined area, resulting in an output set of grids representing the principal components in descending order of percentage of variance explained. The NumPy Python module contains core functionality that can be used to manipulate grids as multidimensional arrays, as well as a set of linear algebra functions (`numpy.linalg`) that enable the calculations necessary for PCA. The Python function below is called by any GridServer area-based method when a PCA-based request is submitted.

```
def get_pca_grids(self, data):
  ### Computes principal components for the requested set of input grids.
  ### Returns the set of principal component arrays to be turned into
  ### grids, and an array of the matching eigenvectors expressed as
  ### percentage of variance explained.

  fill_val = -9999.
  mdata = numpy.ma.masked_array(data, numpy.isnan(data))

  # Get shape of 3D grids array and create a zero'd 2D array the same size,
  # then fill new 2D array with flattened 3D subarrays.
  # This is all to get a flattened array (one entire grid per subarray),
  # so the covariance matrix can be properly calculated across grids.

  d1,d2,d3 = mdata.shape
  mdata2 = numpy.ma.zeros([d1,d2*d3])
  for i in range(len(mdata)):
    mdata2[i] = mdata[i].flatten()

  # Mean-center the flattened array...axis=0 calculates the column mean,
  # which is subtracted from each array element in that column.

  mdata2 -= numpy.ma.mean(mdata2, axis=0)

  # Scale the mean-centered array...axis=0 calculates the column SD,
  # which is then used to divide each array element in that column.

  mdata2 /= numpy.ma.std(mdata2, axis=0, ddof=1)

  # Calculate the covariance matrix of the mean-centered, scaled array.
```

```
covar = numpy.ma.cov(mdata2)

# Use numpy.linalg.eig to calculate the eigenvalues and eigenvectors
# of the covariance matrix. By its nature the covariance matrix is
# always a square array, so we can properly use linalg.eig to do this.

eval, evec = la.eig(covar)

# Get array index for the eigenvalues, such that the largest eigenvalues
# are first in the sort order. The index is used to sort both the
# eigenvalues and the eigenvectors.
# That is, the eigenvectors are properly sorted to maintain their
# association with the matching eigenvalues.

eval_abs = numpy.absolute(eval)
idx = eval_abs.argsort()[::-1]
sorted_eval = eval_abs[idx]
sorted_evec = evec[idx]

# Use the properly sorted (high to low) eigenvalues and associated
# eigenvectors to calculate new arrays representing the principal
# components. The new arrays are stored in the pc_grids numpy array
# and returned. Matching eigenvalues are converted to percentage
# variance explained, and placed in eigval[].
#
# First  PC is calculated as:
#   PC1=(evec_col0_val0*mdata0)+(evec_col0_val1*mdata1)+...
# Second PC is calculated as:
#   PC2=(evec_col1_val0*mdata0)+(evec_col1_val1*mdata1)+...
# and so on...

eval_sum = numpy.sum(sorted_eval)
pc_grids = numpy.zeros([d1,d2,d3])
eigval = []
val = 0

for i in pc_grids:
  i = numpy.ma.zeros([d2,d3])
  for j in range(len(mdata)):
    i += sorted_evec[j][val] * mdata[j]
  pc_grids[val] = i.filled(fill_val)
  eigval.append(round((sorted_eval[val]/eval_sum)*100, 2))
  val += 1

return pc_grids, eigval
```

Appendix K: GridStats Client-side Code Samples

The initial settings page in the GridStats web interface incorporates hundreds of lines of JavaScript code to perform tasks such as date checks, menu updates, and validation of input settings prior to request submission. Most of these tasks are handled using the jQuery JavaScript library and are executed solely on the client-side, while others involve AJAX methods to asynchronously retrieve server-side data to update client-side options. The code samples below are representative of the larger set of client-side code in the system.

Sample Change Function

Each radio button group on the input settings page has dynamic client-side functionality attached to the change event. The jQuery `$.change()` function was used for this purpose rather than explicitly defining an `onchange()` call in the HTML element. This approach allows any necessary logic to be processed anytime the user makes a change to that radio button selection. In this example, the Spatial Scale radio button group captures a change whenever the user clicks among Point, Area, or Entire Grid selections, or the sub-selections under Area (Draw box, State, County, Watershed). Certain commands and functions must be processed depending upon which radio button is selected (e.g., if Area is selected, the "Summary Statistics" choice is disabled, and if "Draw Box" is then selected by default, a function is called to enable box drawing).

```
// for when point scale changes to area scale and vice-versa
$("input[name='spatial']").change(function(e){
  removePolyLayers();
  removeBoxLayer();
  disableBoxDraw();
  disablePoint();
  setStat();
  $('#click_map_text').removeClass('red_text');
  if( $(this).val() == 'point' ) {
    disableArea();
    uncheckSubAreas();
    on_select_map_click();
    $('#click_map_text').addClass('red_text');
  } else if( $(this).val() == 'area' ) {
    removePointLayer();
    enableBoxDraw();
    $('#pca').removeAttr('disabled');
    $('#pca_label').removeClass('disabled');
    $('#draw_box_text').addClass('red_text');
    if( !$("input[name='areas']:checked").val() ) {
      $('#box').prop('checked', true);
    }
  } else if( $(this).val() == 'full' ) {
  $('#draw_box_text').removeClass('red_text');
    removePointLayer();
    uncheckSubAreas();
    if( $('#pca').prop('checked', true) ) {
      $('#pca').attr('disabled', 'disabled');
      $('#pca_label').addClass('disabled');
      $('#mean').prop('checked', true);
    }
  }
});
```

Sample Menu Updater

In the date selection menus for daily data, the selectable days must be adjusted depending upon the currently selected month (i.e., if June is selected, day 31 must be disabled as a possible selection). This happens dynamically due to the `updateDays()` function below, which indexes into the `mo_days` array and automatically enables or disables days depending upon the month. The function properly handles February 29 in both leap years and regular years. The jQuery statements used in this function are dynamically assembled based on input values.

```
var mo_days = Array(31,28,31,30,31,30,31,31,30,31,30,31);

// disables/enables days of month in <select>s based on new selection
// called when changes are made to month dropdowns
// also called from updateMonths()
function updateDays(pre, se) {

  var sy, sm, sd;
  var sy = parseInt($('#'+pre+'_'+se+'yr').val(), 10);
  var sm = parseInt($('#'+pre+'_'+se+'mo').val(), 10);
  var sd = parseInt($('#'+pre+'_'+se+'dy').val(), 10);
  var ld = mo_days[sm-1];

  // assign 29 as last day of Feb in a leap year
  if( sm == 2 ) {
    if( sy % 400 == 0 ) { // leap year
      ld = 29;
    } else if( sy % 100 == 0 ) { // not leap year
      ld = 28;
    } else if( sy % 4 == 0 ) { // leap year
      ld = 29;
    } else { // not leap year
      ld = 28;
    }
  }

  // re-enable any options that were disabled earlier
  $('#'+pre+'_'+se+'dy *').attr('disabled', false);

  // disable any days (i.e., 31) that don't exist in current month
  if( ld < 31 ) {
    for( var j=ld+1; j<=31; j++ ) {
      $('#'+pre+'_'+se+'dy option[value="'+j+'"]').attr('disabled', true);
    }
    // handle leap year - make sure 29 Feb is not disabled
    if( ld == 29 ) {
       $('#'+pre+'_'+se+'dy option[value="29"]').attr('disabled', false);
    }
  }
```

Sample Validator

This sample function validates the selected number of input grids for certain statistics – certain statistics such as PCA require at least three input grids. This function ensures that the minimum number is selected prior to submission, avoiding any server-side errors that would result from such a request. In this function, many of the jQuery statements are dynamically constructed based on current menu selections.

```
var elm = 'border';
var sty = '1px solid red';

function validateNumGrids(pstat) {
  var resp = 'You must select at least 3 grids when calculating
'+pstat+'.';
  if( $('#'+pstat).prop('checked') == true ) {
    if( $('#mo_in_yr').prop('checked') == true ) {
      if( $('#miy_endyr').val() - $('#miy_startyr').val() < 3 ) {
        $('#miy_startyr').css(elm,sty);
        $('#miy_endyr').css(elm,sty);
        alert(resp);
        return false;
      }
    }
    if( $('#mo_in_rng').prop('checked') == true ) {
      start = $('#mir_startyr').val() + '/' + $('#mir_startmo').val() + '/'
+ 01;
      end = $('#mir_endyr').val() + '/' + $('#mir_endmo').val() + '/' + 01;
      chk = calcDays(start, end);
      if( chk < 58 ) { // under 3 months, accounting for inclusion of Feb
        $('#mir_startyr').css(elm,sty);
        $('#mir_startmo').css(elm,sty);
        $('#mir_endyr').css(elm,sty);
        $('#mir_endmo').css(elm,sty);
        alert(resp);
        return false;
      }
    }
    if( $('#days_in_rng').prop('checked') == true ) {
      start = $('#dir_startyr').val() + '/' + $('#dir_startmo').val() + '/'
+ $('#dir_startdy').val();
      end = $('#dir_endyr').val() + '/' + $('#dir_endmo').val() + '/' +
$('#dir_enddy').val();
```

```
      chk = calcDays(start, end);
      if( chk < 3 ) {
        $('#dir_startyr').css(elm,sty);
        $('#dir_startmo').css(elm,sty);
        $('#dir_startdy').css(elm,sty);
        $('#dir_endyr').css(elm,sty);
        $('#dir_endmo').css(elm,sty);
        $('#dir_enddy').css(elm,sty);
        alert(resp);
        return false;
      }
    }
  }
  return true;
}
```

Sample AJAX Call

Dynamically-filled menus on the input settings page, such as the County menu and HUC Name menu, must be filled immediately after the user makes a selection in the previous “State” menu. This is done by automatically sending a request to a PHP script on the server (`rpc.php` in the function below), which formulates and submits a query to the database, receives the return, processes it in to a JSON string, and returns it to the calling function on the client. The calling function, built using the jQuery `$.getJSON()` approach, it is able to process the JSON return and dynamically populate the HUC menus based on the State selection. In the case of the Watershed menu, a spatial database query is performed to retrieve all HUCs that intersect with the selected state. That set is returned to this function, which populates the HUC Name and HUC Code menus.

```
function populateHucs(state_name, cookie_load_menu, cookie_zoom_poly){
  $.getJSON("rpc.php?proc=hucs&state2=" + state_name,
    function(data) {
      var ops1 = ['<option value="null">- HUC Name -</option>'];
      var ops2 = ['<option value="null">- HUC Code -</option>'];
      $.each(data, function(i,row) {
        // IDs are used as values so the menus can be synced
        ops1.push('<option value="' + row[0] + '">' + row[1] + '</option>')
        ops2.push('<option value="' + row[0] + '">' + row[2] + '</option>')
      });
      $("#huc_name").html( ops1.join('') );
      $("#huc_code").html( ops2.join('') );
      $('#hucs_loader').html('&nbsp;');
      // need to do from_cookie here rather than in cookie function area,
      // so that these can fire after AJAX call
      if( cookie_load_menu ) {
        $('#huc_name option[value="' + $.cookie('huc_name') +
'"]').attr('selected', 'selected');
        $('#huc_code option[value="' + $.cookie('huc_name') +
'"]').attr('selected', 'selected');
        if( cookie_zoom_poly ) {
          zoomToPoly('huc');
        }
      }
    }
  );
}
```

Appendix L: Calculating Centered Moving Averages

The interactive charts on the GridStats result pages are built dynamically by processing the data returned from a GridServer request and formatting the values to work with the client-side Highcharts JavaScript library. In addition to the data values, centered moving averages are calculated and plotted on the same chart. Each time series includes a pair of moving average windows (e.g., monthly uses 12-month and 5-month windows) on the chart. The centered moving averages are calculated using a PHP function that accepts the data values array and a window size. Odd-numbered windows are fairly easy to calculate, since two data values on either side of the window center are used (e.g., values 1-2 and 4-5 in a size-5 window). Even-numbered windows are more difficult to process and require essentially a double moving average operation to calculate the centered values.

```
/* Creates an array of centered moving average data values */
/* for a given data array and window size.                 */
/* Correctly handles both even- and odd-sized windows.     */

function centeredMovingAvg($data, $win) {

  // determine whether win is odd or even
  $odd = $win % 2 == 0 ? false : true;

  $param = $_POST['param'];
  $dec = $param != 'ppt' ? '%01.1f' : '%01.2f';
  $ct = count($data);

  // set proper halfwin for odd vs. even
  if( $odd ) {
    $halfwin = floor($win/2); //ie, win=5 gives halfwin=2
  } else {
    $halfwin = $win/2; //ie, win=10 gives halfwin=5
  }

  $end = $ct - $halfwin;
  // assign output nulls for unused data points at beginning of range
  for( $i=0; $i<$halfwin; $i++ ) {
    $out[$i] = 'null';
  }

  if( $odd ) {
    // calculate centered moving average across odd window
    for( $i=$halfwin; $i<$end; $i++ ) {
      $sum = 0;
      for( $j=($i-$halfwin); $j<=($i+$halfwin); $j++ ) {
        $sum += $data[$j];
      }
      $out[$i] = sprintf($dec, $sum/$win);
    }
  } else {
    // calculate centered moving average across even window
    // this is a 2-by-X centering method, where X is the even window value
    $cf1 = 1/$win; // coeff for calculating non-endpoint vals
    $cf2 = $cf1/2; // coeff for calculating endpoint vals
    for( $i=$halfwin; $i<$end; $i++ ) {
      $sum = 0;
      for( $j=($i-$halfwin); $j<=($i+$halfwin); $j++ ) {
        // window of 10 will have val index range of 0-10 (11 vals)
        $coeff = $j == ($i-$halfwin) || $j == ($i+$halfwin) ? $cf2 : $cf1;
        $sum += $coeff * $data[$j];
      }
      $out[$i] = sprintf($dec, $sum);
    }
  }

  // assign output nulls for unused data points at end of range
  for( $i=$end; $i<$ct; $i++ ) {
    $out[$i] = 'null';
  }

  // can end up with an array full of nulls with short time spans
  // this messes up chart ... so return false and don't chart it at all
  if( count(array_unique($out)) == 1 && $out[0] == 'null' ) {
    return false;
  } else {
    return $out;
  }
}
```