Capacity Planning in UNIX Environments

91-30-5

Xueping Deng
Department of Computer Science
Oregon State University
Corvallis, OR 97331-3202

Capacity Planning in UNIX Environments

by
Xueping Deng

A Project Report
submitted to
Dept. of Computer Science
Oregon State University

in partial fulfillment of
the requirements for the
degree of
Master of Science

Completed April, 1991
Commencement June 1992

Contents

Computer Capacity Planning in UNIX Environments

Abstract

Capacity Planning is a gradually recognized need in UNIX environments. However, the effort to address this problem has been more or less event driven and is often discarded after crises are over. The main cause to this situation is that the tools to support the CP function in UNIX environments are lacking so that the CP function can not be standardized and each project has to reinvent "the wheel".

In this project, we investigate the current status of CP and to narrow the gap between the need and lack of tools by building a prototype to support CP in UNIX environments. In building the prototype, we use queueing network models to capture the characteristics of a system and use expert system technology to capture the expertise of CP in UNIX.

We have built a queueing network model for Sequent system and then integrated this model into the expert system of the prototype.We have also encoded some simple knowledge into the expert system. With a user-friendly interface, the prototype is able to help the user do What-if analysis. The prototype is now running on Macintosh platform with satisfactory performance and promising potentials.

Chapter 1. Introduction

[CADY90] [LAM87] Capacity planning (CP) is an indispensable business function in most enterprises, whether their output is paint, insurance policies or information. Capacity planning in computer installations permits effective use of capital and high levels of job satisfaction and morale for all staff members. The CP function performed in UNIX installations is the concern of this research project.

In this introduction, we presents a brief look at the CP function in UNIX, and describe the goals of this research project. We conclude with the summaries of the remaining chapters of the report.

1.1 The Goals of the Research Project

CP is the matching of hardware, software, workload, and service level objectives for the current systems and possible alternatives both for the present and the future [BLAYLO83] . The objective of CP is to provide consistent, cost-effective service, and to avoid under- or over-configuration [LAZOWS85].

CP has evolved toward an indispensable business and technical function in computer environments since the development of computers in 1940's. However, [KRAUSE88] since the UNIX environment is relatively new, and is not as widely distributed throughout the corporate world as some other environments such as MVS, CP in this environment has not been set up widely as a standard function, and it is not well supported. Despite the fact that there has been a gradual recognized need for this function, the effort to address this problem has been event-driven; usually discarded after crises are over. One of the causes for this situation is that, to date, it has not been commercially feasible to develop the tools like those that are available to MVS users. That is, the tools to support CP function in UNIX are lacking[KRAUSE88] [FEDER84].

One goal of this research project is to investigate what is available and what is needed to support CP in UNIX environments. The investigation reveals that UNIX environments are equipped with some data collecting utilities, but the tools to make use of this raw data to help the capacity planner are lacking.

Consequently, the second goal of the research is to explore a solution to the problem of lack of tools by building a prototype. The prototype uses expert system technology to automate and integrate such CP activities as data collection, modelling, performance prediction, and automatic reconfiguration to meet performance objectives. The modelling and performance prediction functions of the tool are based on queueing network modelling technique, which has emerged as the technology of choice in many CP applications. The main components of the tool include an expert system, a graphics interface, and a set of reduction programs. The prototype is now running on Macintosh platform.

1.2 Summary of Each Chapter

In chapter 2, we briefly review the definition of CP and the approaches and tools used in CP. After investigating the current status of the CP function in the context of UNIX environments, we summarize what's needed to support the CP function, providing a basis for the work on a prototype CP tool.

In Chapter 3, we discuss desirable features and the top-level view of a prototype tool called Capacity Planner. We conclude with its development decisions.

In Chapter 4, we briefly discuss the theory basis for modelling a system. Based on the discussion, we build queueing network models for Sequent system, which are parameterized and validated with the available measurement data.

In Chapter 5, discussion is focused on the application of expert system technology to CP. We discuss the possible inputs, outputs for the expert system of Capacity Planner. We conclude with the discussion on the current version of the expert system.

In Chapter 6, We give a user manual for Capacity Planner.

Finally, in Chapter 7 we summarize the main result of the research project and discuss the future work for this project.

Chapter 2. Capacity Planning Reviewed

[COOPER80]Computer capacity planning consists of two elements: one technical and the other managerial. We explore only the technical element in this research project. In practice, however, we can not ignore the other element at all.

In this chapter, we survey the different views of the technical aspect of CP and investigate the current status of the art of CP in UNIX environments. The last section discusses the features needed for a tool to efficiently support CP functions, which provides a basis for later chapters on designing and implementing a prototype tool.

2.1 Definition of Capacity Planning

The definition of computer capacity planning is closely related with the tuning and optimization of the system performance, though there are actually as many definitions as there are capacity planners. Differences among the definitions result from different perspectives of the capacity planners.

One view divides the performance management of an installation into two types of activities, one strategic and the other tactical. The former one focuses on the longer term aspects of planning for the resources of the installation, and is called capacity planning or capacity management. The latter focuses on the shorter term, almost day-to-day activities [HOWARD90].

Another view is that capacity planning is a new initiative that arose from the field traditionally known as performance evaluation and tuning. From this perspective, capacity planning is itself a stage in a longer development process. Experience as a system programmer is a prerequisite to becoming a computer capacity planner, and the tools used for performance evaluation are appropriate for solving CP problems [HUGO83].

A functional view of capacity planning is derived from its main purpose: to provide an agreed-upon service at a reasonable cost. Service management, cost

management, workload management and system resource management are the components of capacity planning for the functional viewpoint [HUGO83].

In the chronological view, we divide CP functions into past, present and future. The management of current workloads and systems and planning of future ones are based on data bases of historical records. The popularity of this point of view probably derives from the natural focus it provides on planning and prediction, which is the current management "scare" item. Thus, in terms of importance, planning is seen as the major function, to be followed in importance by current systems management [HUGO83].

These views may differ in their perspectives, but their underlying concept is essential the same: computer capacity planning is to monitor and project computer workloads and to plan for changing or expanding computer configurations to satisfy future demands in a cost-effective manner[LAM87]. It is an on-going process instead of an event. It is simple in concept, based on a well understood business principle -- the need to provide a service at a price acceptable to the market. Behind this simplicity, however, lies a host of complexities, technical and otherwise.

The capacity planning process consists of five steps [HOWARD90]:
1. Identifying all workloads that the system must support,
2. Profiling each workload in terms of its resource demands and time interdependencies,
3. Aggregating the workloads,
4. Determining the aggregate resource requirements, and
5. Finding equipment to match the aggregate resource requirements.

2.2 Current Approaches and Tools for Capacity Planning

The approaches capacity planners take in the capacity planning process may vary, but they are usually grouped as varying over a spectrum according to the approaches taken in step 2 and 4.

[LAZOWS84] At one end of this spectrum is intuition and trend extrapolation. This approach is simple, but it imposes a high demand for the degree of experience and

insight that can yield reliable intuition. Another extreme is the experimental evaluation of alternatives. Experimentation is always valuable, because it is accurate. It is also inflexible and expensive -- often prohibitively so. Between these two extremes are the approaches often grouped as modelling. These approaches attempt to distill, from the mass of details that is the system itself, exactly those aspects that are essential to the system's behavior. These approaches are more reliable than intuition, because they are more methodical. They also enhance experimentation because they provide guidance as to which experiments are necessary. In this research project, we take this modelling approach.

No matter what approaches is taken, the capacity planer has to make use of a variety of tools and methodologies. Even the the approach of intuition and trend extrapolation requires the capacity planner having some knowledge of the current performance of the system before he/she can base his/her intuition and can make the trend extrapolation. Especially when the system is becoming more complex and the increasing complexity of systems means that capacity planning is becoming more knowledge intensive. It is impossible to approach the CP problems by intuition and calculation on the back of an envelop. Advanced planning tools and measurement tools are indispensable in modern CP.

There are three classes of CP tools[HOWARD90]. Measurement tools help determine utilization of resources and workloads of the system. Analysis tools are used to study and reduce the measurement data. Projection tools predict capacity needs for future periods based on measurements, reduced data, and user requirements, so that aggregate resource requirements can be determined and the capacity of the system can be planned.

Usually, measurement tools are either hardware or software monitors designed to collect various performance data, that can be used both for performance evaluation (PE) and/or CP[FERRAR83]. They are not dependent on which approach is taken in CP because all approaches have to use them. Analysis and prediction tools are used mostly in the modelling approach, they may also be used to understand and interpret the experiment data produced in experiment approach.

2.3 Capacity Planning in UNIX Environment

As discussed in Chapter 1, the UNIX operating system is relatively new, and is not as widely distributed throughout the corporate world as some of others such as MVS. The CP function has not been woven into UNIX environment and it has been poorly supported both technically and managerially, the most fundamental cause for this problem is the lack of generic CP tools and procedures. In the commercial product list of capacity planning tools, including analysis and projection tools, in [HOWARD90], more than 90% of the products are devoted to the MVS environment. Different projects then have to handle their CP functions in their own ways - usually as an afterthought. Project-specific measures which are difficult to modify in response to a changing needs and not easily to be adapted to other applications arose as a result. Because of this infeasibility, capacity planners in UNIX environments have not developed the same generic tools like those available to MVS users and UNIX shops have not had the opportunity to build an industry established Computer Performance Evaluation structure. UNIX applications have to constantly "re-invent the wheel" for their performance and capacity concerns.

The utilities available in UNIX to collect the raw data are mainly system utilities [FEDER84]. There are also user-developed applications. Some of the most often used utilities are listed below [KRAUSE88]:
1. Timing command such as timex(1). This kind of utility collects data which can be used to answer questions such as - " What's the response time for some application?" - but does not supply any supporting data.

2. Resource Utilization Measurement Utilities such as ps(1), df(1M). Their measurement is often system-oriented, and not easily reducible to a customer interaction level. It is too time-consuming and tedious to manage the collection and storage of system data. It is almost impossible to extract useful information from the raw data for CP issues such as "When we will need a new disk drive?"

3. Profiling utility profil(2). It can be used to produce profile data when new transactions are compiled. This data is useful in predicting the demand of future workload, it helps not much for measuring the on-going workload.

4. System accounting. UNIX system accounting utilities come close to providing performance measurement data at a transaction level, which is essential to extract necessary information for CP purpose. They do monitor some system resource data

(CPU, memory, user and system times) on individual commands, as run by individual users. However, there are drawbacks:
a. Information is accessible only in a few rigid formats.
b. Most useful data groupings (by command, by user) are too voluminous to easily store or access. For example, a typical two hour interval on Sequent system may take up several MB of disk storage.
c. UNIX accounting only records completed processes. Continuously running daemons are not recorded, which may give misleading snapshot views of the measured environment.
d. The commands are not user friendly. One needs a certain UNIX sophistication and privilege to use them intelligently. And, once a user turns it on, it's no longer accessible to others.

5. Home grown monitors[SALAWU85]. Capacity planners need specific monitoring systems to provide a combination of system and customer-oriented information in a readily accessible and easy to maintain form, because the UNIX environment does not currently provide such generic tools. However, this approach may become expensive in several respects:
a. This type of monitors is intended as an internal tool, to provide support that the external customer only sees as a system by-product, which is not as tangible as other project enhancement such as new products or future enhancements. It becomes a hard work effort to justify.
b. Tools developed are often project specific, and cannot be ported easily to other applications, reforcing the "re-invent the wheel" syndrome.
c. If the required UNIX system expertise is not available in-house, external sources are needed, at greater cost.

2.4 What is needed

From above discussion, it's clear that CP is becoming an indispensable function in UNIX environment and generic tools are needed to support this function, no matter what approach is taken in the CP process.

There are measurement utilities available in UNIX environment, as discussed above. They are not dedicated to CP: they can produce volumes of raw data which can be meaningless to the capacity planner without any reduction.

Analysis tools can boil down the raw data and produce useful numbers such as input configurations and workloads for CP. Projection tools can then make use of those number to compute response time, throughput, utilizations, etc. so that CP questions may be answered. These tools are essential to CP process, but they are lacking in the UNIX environment. The have to be provided to supported the CP function in the UNIX environment.

The tools mentioned above are traditional tools which are good at translating huge amount of measurement data into information that may contribute to more informed decision making. [STROEB85] However, expertise in the field of capacity planning, and understanding and knowledge of the system under study are required to correctly interpret these information and relate it to end-user performance. Expert system technology offers considerable promise in bridging this gap between the specialized, technical challenges of the capacity planning process and the performance needs of the customers. By now, a few knowledge-based tools are already available in the market, mostly for MVS environment. More research and effort has been put into this respect.

Chapter 3. Capacity Planner - a Prototype

In this chapter, we discuss the design issues about a prototype tool named Capacity Planner which is an integrated and generic tool to support CP functions in the UNIX environment.

3.1 The design of Capacity Planner

3.1.1 Out look of Capacity Planner

The tool makes use of expert system technology to automate the routine and time consuming capacity planning tasks of data collection, modelling, performance prediction, and automatic reconfiguration to meet performance objectives. The core of the tool is an expert system built with Nexpert Object. The expert system is interfaced with SuperCard to make it more user-friendly. The expert system acquires user input through the SuperCard interface and displays its reasoning results or other information through the same interface to the user. It fetches the collected or reduced data from the target system through AppleTalk. The top-level view of the tool is shown in Figure 1.

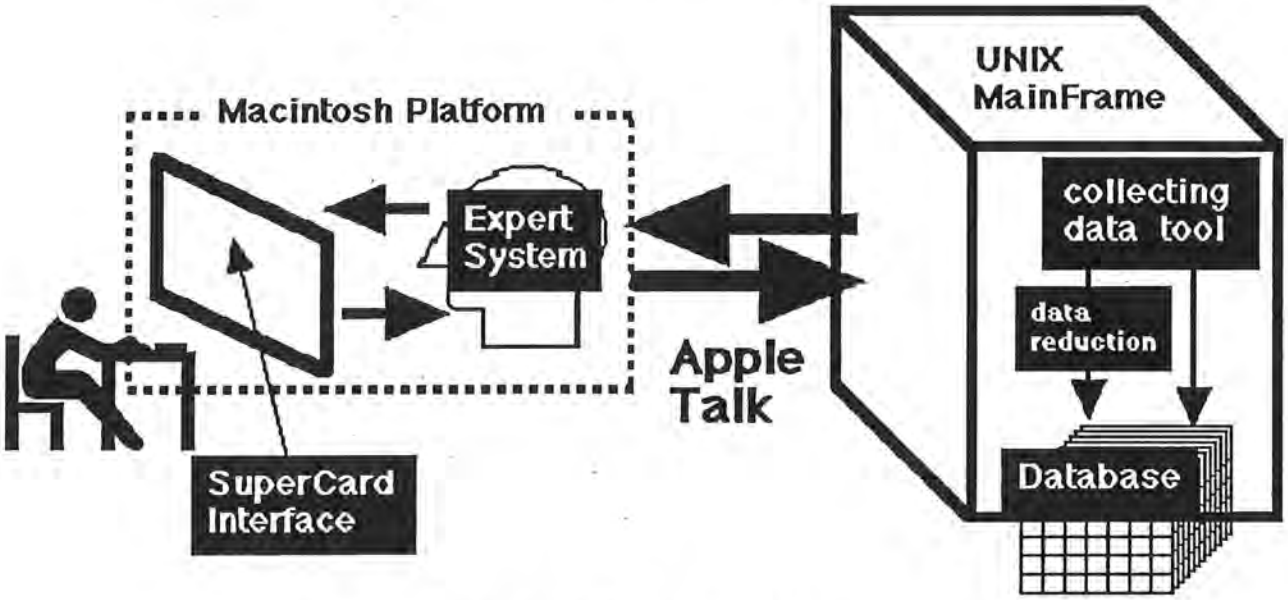


Figure.1 Outlook of Capacity Planner

The tool is designed to have the following features, the features presented in italic have not been implemented:

-- Graphic driven, user-friendly interface

-- *Automatic data collection and data reduction*

-- Queueing network modelling

* *Memory subsystem modelling*
* *I/O subsystem modelling*
* System modelling

-- Batch, terminal and transaction processing modelling

-- WhatIf analyses with comments about system performance

-- *Expert suggestions for optimal configurations*

-- *Online help*

3.1.2 Interface considerations

The interface is built with SuperCard. It consists of a menu system and a set of windows through which information is displayed and the user enter inputs. The main concern for the interface is for the structure of menu through which the user can issue commands.The design of the menu specifies the overall structure of the Interface, and defines the functionalities of Capacity Planner. We list the menu items below and give their appearance in Figure 2.

a) File -- maintain text file. This menu allows the user to save, print or open some saved suggestions/comments which may be given by the expert system.

b) Edit -- edit text file. These functions are the standard operations available with all Macintosh applications.

c) Expert -- maintain knowledge base and restart inference engine.

1) Load knowledge base -- Load knowledge into memory.
2) Save knowledge base -- Save current knowledge base.
3) Clear knowledge base -- Clear current knowledge in the memory.
4) Restart planner -- Reset the knowledge base and start a new session for the Capacity Planner.

d) Modelling -- retrieve data from database and evaluating the baseline models.

1) Auto -- Data is to be collected and reduced automatically.
2) Manual -- Data is to be collected and reduced manually.

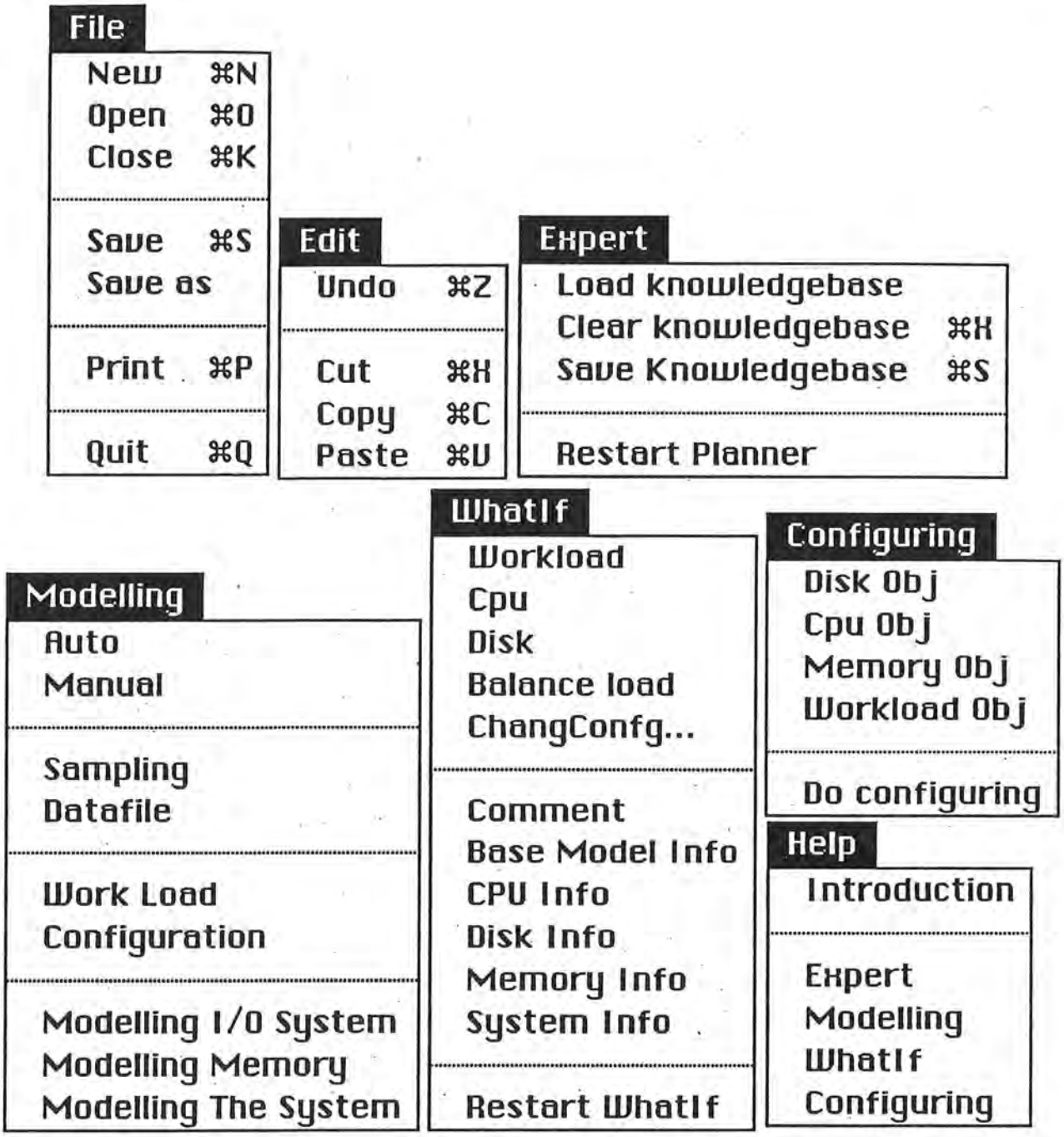


Figure.2 Menus of Capacity Planner

3) Sampling -- In Auto mode, allows the user to specify sampling parameters, such as the the length and the interval of the sampling.

4) Datafile -- In Auto mode, allows the user to provide the tool with existing data files

5) Workload -- In Manual mode, allows the user to input data related to workload, such as number of classes of workload, etc.

6) System confg.In Manual mode, allow the user to input data related to system configuration.

7) Modelling the system

8) Modelling memory subsystem

9) Modelling disk subsystem

e). WhatIf -- allows the user to make changes on workload, or configuration of the system so as to answer what-if questions based on the baseline model.

1) Workload -- Make changes on workload.

2) Cpu -- Make changes for cpus.

3) Disk -- Make changes for disks.

4) Balance load -- Balance load on disks.

5) Comment -- Display comments made by the expert system.

6) Base model Info -- Display information about base model.

7) System Info -- Display system info. such as response time, throughput, etc.

8) Cpu Info -- Display detailed cpu info.

9) Disk Info -- Display detailed disk info.

10). Memory Info -- Display detailed memory info.

11). Change confg... -- Make changes on configuration of the system.

12).Restart WhatIf -- Clean up any changes and start a new WhatIf analysis session.

f) Configuring -- allows the user to specify performance objectives so that the expert system can help in configuring the system to achieve those objectives.

j) Help -- offering online help as how to use the tool.

3.2 Decisions concerning the development of Capacity Planner

1. Sequent system

We have chosen Sequent Symmetry system as the target system. It offers a UNIX environment. In addition to the basic system utilities discussed in Chapter 2, Symmetry systems are equipped with extra utilities which are useful for the purpose of capacity planning. Although they are not dedicated to capacity planning, these utilities, together with those available in common UNIX environment, provide a firm base to start the design of Capacity Planner. The extra utilities are: 1). monitor(1); 2). SPM (System Performance Monitor); 3). logmon(1) (This is actually the log format of monitor); 4). repmon(1).

We use Logmon and Accton(system accounting) to collect the raw data for the project, because the data collected by these two is closest to the information needed for CP purpose. We use other data collecting utilities as reference tools. Repmon is used to sort out the logmon data, while other reduction programs are developed to reduce the Accton data and to correlate the two set of data.

2. Macintosh platform

The Macintosh platform provides a good environment to explore ideas, because many tools are already there, such as SuperCard. These tools are very easy to use and the results are satisfactory. Not surprisingly, we choose Macintosh as the platform to build the prototype.

3. Queueing network modelling technique

In this project, we use queueing network modelling technique to build a model for the system, on which other CP issues such as What-if analysis, and configuration problems are based. Queuing network modeling is especially suitable for modeling computer systems[LAM87][LAZOWS84]. The technique yields relatively accurate results with relatively low cost, that is, it achieves a favorable balance between accuracy and cost.

4. Tools used

Nexpert Object is very easy and comfort to use, mainly because of its user-friendly graphical user interface. It also has other important features, such as database links. Besides, it can be totally controlled by some application, say, an application written in C or other language. This makes it very easy to add an interface on top of the expert system.

SuperCard provides an excellent environment to build a user-friendly interface, because it gives the user total control over the entire interface, including all menus, command keys, windows, etc., and it includes tools that allow the user to manipulate graphics, text, and numbers.

Another tool chosen is Think C. It has a fairly good development environment. Code resources can be easily built with Think C and then installed into the rules of expert system so as to provide it with numerical ability.

Chapter 4. Modelling the System

[BARNES79] [INGRAH87] We can build different models to capture different properties of the same system. The widely used models for the performance of computer systems are petri net models and queueing network models[POTIER85], although some authors claim that simulation models can also be built for the system [HOOVER89] [HOWARD90]. We can evaluate the models with either analytic approach or simulation approach. In this chapter, we discuss queueing network modelling technique in the context of computer systems. We also discuss the steps in construct a queueing network model and the analytic approach used to evaluate the model. Finally, we discuss the research on modelling the Sequent system with queueing network technique.

4.1 Queueing network modelling technique

4.1.1 Queueing network models for computer systems

A queueing network model for a computer system consists of a set of equations built with queueing theory[HOOVER89]. The model represents the system as a network of queues. Each queue is associated with a service center which represents a system resource, for example, a cpu, a disk, for which user jobs/transactions compete[LAM87]. In capacity planning, we can use queueing network models to model a single resource (such as a cpu, a disk, etc), a subsystem (such as disk I/O and memory subsystems) and the system as a whole[LAZOWS84] , although we have only built the system level model in the project. The queues can be decomposed into queues of different classes which correspond to the different components of the workload of the system.

When a model is constructed at system level , cpus are often the first choices as service centers. Each cpu can always be represented as a single service center. When the cpus are tightly coupled, that is, they share a single pool of memory and a single queue of jobs, they can be represented as a single service center, which results in a central-sever queueing network model. Accordingly, there is a multi-sever queueing network model when each cpu is represented as a single service center[LAZOWS84]. In both cases, each disk is represented as a service center, usually because a request to a certain disk can not be

completed by another disk even if this disk is free, that is, disks can not be tightly-coupled together. Delay due to other I/O subsystem components(e.g., channels) is represented by calculating an appropriate effective service demand for the according disk service center.

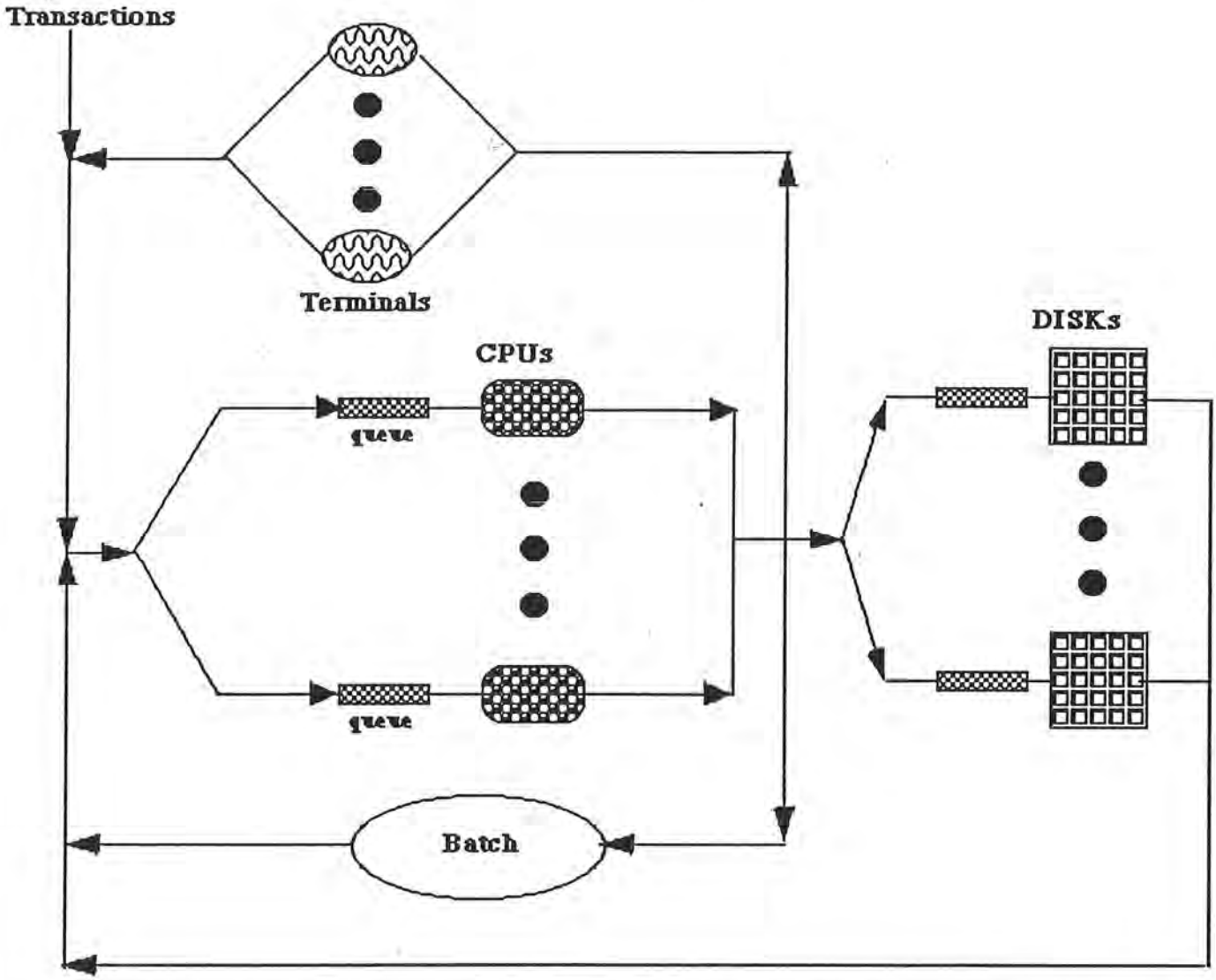


Figure.3 A multi-sever queueing net work model for computer systems

Besides the service centers, a model also includes a source of customers, or workloads in the language of computer systems. Three kinds of workload components may be included in the model: terminal, batch jobs and transactions. The total number of jobs in the system is fixed in the case of batch jobs. With terminal jobs, the sum of the number of jobs in the system plus the number of terminals at which the user is thinking, that is, the terminal is idle, is fixed. Both of these cases constitute "closed" systems as the rate of arrival of jobs is

directly influenced by the rate at which the system can process jobs, both of these workloads are called closed load. On the other hand, transaction load is refereed as "open", since the arrival of jobs to the system is not affected by the departure of jobs from the system. Of cause, each of these workloads can in turn be decomposed into different workload classes. A multi-sever queueing net work model with three classes of workload is shown in Figure 3.

Under this model, a job exiting from a service center becomes input to other service centers in a manner as specified in the model parameters, and then exits the system as it finishes. This description corresponds closely to job execution in actual computer systems. The complexity of the model can be controlled by specifying the number and types of service centers, routing characteristics, and the number of workload classes.

[LAZOWS84] Memory imposes a constraint on the number of "threads of control" that can be active simultaneously. Memory management imposes overhead on the resources of the system. Usually, we make implicit assumptions about this constraint so that we can ignore memory explicitly in the model. Typical assumptions related with each type of workload are listed below:

• Batch workloads' intensity is described by its population. The assumption is that the system had a memory constraint which could be expressed in terms of a specific number of jobs (i.e., that all jobs required the same amount of memory), and that there was a sufficient backlog of work that the system was continuously operating at its maximum multiprogramming level.

• Terminal workload is described by its population and average think time. The assumption is that the system had a fixed number of interactive users, and that enough memory existed to accommodate as many of these users as might concurrently require it (i.e, that there was no memory constraint).

• Transaction workloads' intensity is described by the arrival rate. The assumption is that there was no memory constraint, e.g., there was central subsystem population of a transaction workload.

These assumptions, if not strictly correct, are at least robust and good enough for many modelling studies. Of course, they are not always adequate, in which case, queueing

network models can be extended to include memory effects on system performance, or simulation modelling technique might be used.

4.1.2 Inputs/Outputs of Queueing Network Models

[LAZOWS85] We summarize the inputs and outputs for a queueing network model of a computer system in Table 1 and Table 2.

customer description	C, the number of customer classes For each class c: its workload intensity, one of: lc, the arrival rate (for transaction workloads), or Nc, the population (for batch workloads), or Nc and Zc, the think time (for terminal loads)
center description	K, the number of service centers
service demand	For each class c and center k: Dc,k, the service demand

Table 1 Model inputs

system	aggregate	R, average system response time X, system throughput Q, average number in system
measures	per class	Rc, average class c system response time X, class c system throughput Q, average class c number in system
center	aggregate	Uk, utilization of center k Rk, average residence time at center k Xk, throughput at center k Qk, average queue length at center k
measures	per class	Uc,k, class c utilization of center k Rc,k, class c residence time at center k Xc,k, class c through put at center k Qc,k, class c queue length at center k

Table 2 Model outputs

4.1.3 Fundamental Laws for Queueing Network Models

As we have discussed above, queueing network models use a set of equations to capture the characteristics of a system. We give the foundation to derive the equations here, more detailed discussion on queueing theory can be found in [BARNES79], [BOLKER87], [CADY90], [DENNIN77], [HOWARD90], [LAZOWS84] .

The fundamental laws describe the algebraic relationships among those basic quantities represented in a queueing network model. We are interested in these quantities because the relationships among them abstract the performance of the system.

The utilization law: Uk = XkSx = XDk

That is, the utilization of a resource is equal to the product of the throughput of that resource and the average service requirement at that resource.

Little's law: N = XR

Little's law states that the average number of requests in a system must equal the product of the throughput of that system and the average time spent in that system by a request. This law is central to the algorithms for evaluating queueing network models. The nice thing about the law is that it requires only very weak assumptions so that it can be widely applicable: it can be applied at many different levels, to a single resource, to a subsystem, or to the system as a whole.

The response time law: R=(N/X) - Z

The response time law is a variant of Little's law used at a system level. When we look at Little's law at a system level, the residence time corresponds to the sum of system response time and user think time. Given the system throughput, number of terminal users and the average think time for each user, we derive the response time law from Little's law.

The forced flow: Xk = VkX

This law states that the flows (throughputs) in all parts of a system must be proportional to one another. That is, the various components of a system must do comparable amounts of work.

4.1.4 Algorithms for Evaluating of Queueing Network Models

We have different algorithms to evaluate a model with different workloads. There are algorithms for models with open workloads, closed workloads, or mixed workloads. We have assumed the system has mixed workloads, so we give an algorithm for closed loads and then an algorithm for mixed loads, because the latter is based on the former. Further discussion on these algorithms can be found in [LAZOWS84] [DENNIN77].

An algorithm for queueing network models with closed workloads:

1. Set $Q_{c,k} = N_c/K$ for all c,k.
2. Approximate $A_{c,k}$, the arrival instant queue length at center k seen by an arriving class c customer, with $h_c[Q_{1,k}, \ldots, Q_{C,k}]$, for all c, k.

$$h_c[Q_{1,k}, \ldots, Q_{C,k}] = [(N_c-1)Q_{c,k}/N_c] + \mathbf{E}Q_{j,k} \quad j \text{ ranges from 1 to C and } j \neq c.$$

$\mathbf{E}$ means the sum over j.

3.Apply following equations to compute a new set of $Q_{c,k}$ for all c, k.

- For each class, Little's law applies to the system as a whole

$$X_c = N_c/(Z_c + \mathbf{E}R_{ck}) \quad k \text{ ranges from 1 to K}$$

- For each class, Little's law applies to the service center individually

$$Q_{c,k} = X_c R_{c,k}$$

- For each class, the service center residence time equations

$$R_{c,k} = D_{c,k}[1+A_{c,k}]$$

4. If $Q_{c,k}$ resulting from Step 3 do not agree to within some tolerance (e.g., 0.1%) with those used as inputs in step 2, return to Step 2 using the new Q_{ck}.

An algorithm for queueing network models with mixed workloads:

Let {O} be the set of open workloads and {C} the set of closed workloads.

1. For each center k, obtain its utilization by each open class:

$$U_{c,k} = \lambda_c D_{c,k} \quad \text{where c in } \{O\}$$

and add them together to get the total utilization $U\{o\}k$ by all open workloads for the center.

2. Solve the closed model consisting of the K centers and the closed workloads (but no open loads). The service demand $D_{c,k}^*$ of each load in {C} at each center k in the closed model is set to:

$$D_{c,k}^* = D_{c,k}/(1-U\{o\}k) \text{ for c in } \{C\}$$

where Dc,k is the service demand of class c at center k in the original mixed model. The throughputs, queue lengths, and residence times obtained from the solution of this model are the performance measures for the corresponding closed workloads in the mixed model. Utilizations can be computed by applying the utilization law to the original set of service demands Dc,k.

3. Residence times and queue lengths for the open classes can be computed using the performance measures of the closed loads:

$$R_{c,k} = D_{c,k}(1+Q\{c\}k)/(1-U\{o\}k) \quad \text{for c in } \{O\}$$

$$Q_{c,k} = \lambda_c R_{c,k} \quad \text{for c in } \{O\}$$

where Q{c}k is the total queue length of all closed classes at center k obtained from the solution of the closed model in Step 2.

4.1.5 Steps in constructing a queueing network model

The following steps are necessary to build a queueing network model for a computer system:

1. Data collection

System configuration data and performance data are essential for both constructing and validating the model. Examples of the necessary data are: utilization of the components of the system such as CPU, I/O devices; and the number of jobs of each class completed during a period. All these data have to be grouped according to the base units defined in first step.

2. Workload Identification

To develop a queueing network model of a large scale computer system, it is important to identify the workloads that are to be represented by separate job classes in the model. This step is a crucial step because the level of detail that can be obtained from a given model is a direct function of the job classes that are initially specified. This step involves identifying which of those three workloads comprise the overall workload for the system. The identified workloads can be further decomposed after some investigation.

Another important thing to do in this step is to define the base unit for each kind of workload which quantifies that workload. The base units are the building blocks for the workload of the system. Measurement data should be acquired in terms of these units.

3. Service center description

The service centers represent system resources where congestion or delay may result. There are many ways this representation can be made. Different ways of representation are suitable in answering different CP questions. The purpose of this step is to locate the system resources of interest in order to answer specific CP questions. Also, the service demand on these resources from each class of the workload has to be determined .

4. Model parameterization

After the workload and the service centers are identified for the model, this step is relatively straight forward. It mainly involves calculating the input parameters for the model from the collected or reduced data.

5. Model evaluation and validation

There are two methods to evaluate queueing network models. The first involves calculating bounds on performance measures. The second involves calculating the values of the performance measures. The first is simple enough to be carried out by hand while the second is complicated and computer programs for doing this are necessary. The second method is used in the project.

Validation of the model includes comparing measured performance values from step 3 with the values calculated in this step. If the two sets of values do not match well, we need to check each step and to find out the causes. Typical causes are unsuitable assumptions in each step; wrong definitions of the parameters for the model; the raw data from different utilities is not collected in the same period; and so on. We also need to make adjustment and repeat construction process until the result is satisfactory.

4.2 Research on the Model for Sequent system

Queueing network modelling has been well developed for the use of computer capacity planning. There are commercial queueing network modelling packages available in the market. However, almost all these are MVS-oriented, that is, their terminology is defined in terms of MVS environments and their input data is available with the tools in MVS environments. In this project, we did not use this kind of package, instead, we built a multi-sever queueing network model with the inputs defined in terms of the available raw data. We discuss each step in constructing this model in this section.

4.2.1 Modelling the Sequent system

The Sequent system is a true multiprocessor in which all CPUs share a single pool of memory, all CPUs are identical, and any CPU can execute both user code and kernel code. CPUs can be added or removed without modifying the operating system or user applications. CPUs automatically schedule themselves to ensure that all CPUs are kept busy as long as there are executable processes available. An application can consist of multiple instruction streams, all accessing shared data structures in the memory. The system has 8 cpus and 9 disks, which are chosen to be the service centers in the queueing network model. With these features in mind, we make following assumptions in building the model for the system following the steps described in 4.1.5.

1). The memory does not impose a constraint on the system performance, that is, there is enough memory and it is not a bottleneck.

2). System overhead is evenly distributed among the workload.

3. The influence of other system resources other than CPUs and Disks is accounted for by the parameters of the model.

4). The number of processes completed during the observation interval is significantly greater than the number present at the beginning or end.

5). The queueing processes for CPUs are evenly distributed among CPUs so that every cpu has exactly same queue length, same utilization, etc.

6). FCFS scheduling principle.

Now, we follow the steps to build the model.

1. Data collection and reduction

As discussed in Chapter 2, we use Logmon and Accton to collect the necessary raw data over a specified time period. Logmon provides raw data from which some of the performance measurements such as resource utilization are calculated, while Accton provides a base to calculate other performance measurements, such as throughput, and a base to compute the model parameters such as population of the system. The combination of Logmon data and Accton data provides the rest of the model parameters. The reduction utilities for CP purpose are not available. We have built these utilities based on the definition of the model and its input parameters. The results of the effort are a set of programs written in Awk. We have organized these programs together with a script. The script outputs both the performance measurement as well as the model parameters with Logmon and Accton data as input file. This script runs on the main frame and is not integrated with the function of automatic data collection.

Under this circumstance, the data collection and reduction functions and the validation of the model have not been integrated into the tool, rather, they have been manually manipulated to produce the input parameter files for the model. These input parameter files are stored on the mainframe and are accessed by the expert system through AppleTalk.

2.Workload Identification

The three kinds of workloads: terminal, batch and transaction were included in the model. After some investigation, we found out that the target system is basically an interactive system. The batch workload, if any, was ignorable as compared to the terminal workload, and there was no transaction load for the target system. As a result, we treat the batch load and transaction load as of zero intensity. The terminal workload, in turn, can be further decomposed into different components. However, all the terminal load was grouped into one class if not otherwise mentioned.

Also in the project, the base unit of measurement is defined as UNIX processes. Although there are some shortcomings in using processes as workload building blocks, there are no better reliable alternatives with the available data, There are two direct shortcomings. The first follows from the fact that much work in UNIX is done by forked processes. Resources consumed by forked processes should be charged to the original user-initiated process, but that information is not available. The second involves measuring a baseline response time. Commands like Vi have a single accounting record for the entire editing session. It is impossible to divide the single record to calculate calculate the effective response time for each editing operation[TORNHE87] .

3. Service center description

For the time being, we have only built the model at system level. Accordingly, we choose cpus and disks of the system as service centers for the model. We can either represent all the cpus as a single service center or represent each cpu a service center, as we will see later in this chapter.

4. Model parameterization

The parameterization of queueing network models is relatively straightforward. The problem here is that it is difficult to locate the useful data necessary to calculate the parameters, because some of the needed data is not available among huge amount of data that can be collected. For example, the data needed to calculate two related values, the

response time of the system and the think time of the terminal users, are not directly available. We have to use Little's Law to calculated system response time while the Response Time Law is used to compute the think time.

After analyizing the logmon data and the system accounting data, we have come up with the definitions for the performance measurements and the model parameters in terms of the available data and the model:

Performance measurement,

a). ThroughPut: X= completed Procs/interval(7200) /* 7200 sec. is sampling interval. */

b). ResponseTime: R= Avg.Act.Procs/X --- Little's Law

c). Cpu Utilization: Uc= tot%/800 /* 800 is total cup % for 8 cpus. */

d). Disk Utilization: Ud = XF / 35(or 65)

Model parameters,

a). Population: P = number of logon users

b). Cpu Service Demand: Dc = Uc/X

c). Disk Service Demand: Dd = Ud / X

d). Think Time: Z = Population/X - R --- the Response Time Law

5. Evaluate and validate the model

We evaluate the model with the algorithm given in 4.1.4. The computed performance values from the evaluation in turn are compared to the measured values to validate the model. The measured values are computed according to the above definitions from the collected data.

In the next three sections, we present the work on modelling Sequent system with different models. The models are validated against more than ten sample data, though we only list the validation on the data collected during 2:00pm-4:00pm of Nov.30, 1991 for the central sever model and two more for the multi-sever model. We get the most satisfactory result from themulti-sever model, and we use this model in our project.

4.2.2 A central-sever queueing network model

We can view the tightly-coupled multiprocessors as a single service center, since in the system there is a single queue of processes for all cpus. The throughput of this service center is ideally the sum of those of the individual cpus. Consequently, the straightforward

approach to modelling n tightly-coupled cpus is to create a single center representing them in the model, and to divide the service demands of all customers of all workloads at that center by n. This results in a central-sever queueing network model, which is our first effort to capture the performance of the system, as shown in Figure 4 .

Obviously, we make two more assumptions when representing the tightly-coupled cpus with a single service center. One is that the total service rate of the n cpus equal to n times the rate of a single cpu. Another one is that the effective service rate of a multi-processor is constant, that is,

The steps to build this central-sever queueing network model is basically the same as discussed in 4.3.1, except the definition for Cpu Service Demand has changed. Now that we have only one service center to represent the eight cpus,

$$Dc = Uc/(X*8)$$

The model parameters are:

Population: P = 534 #of service centers K= 10 (1 cpu and 9 disks)

Think time: Z = 72.3703 (sec.)

Cpu Service Demand(sec./process): Dc = 0.00931 (sec./process)

Disk Service Demand(sec./process):

disk0 = 0.0369	disk1 = 0.0573	disk2 = 0.0614
disk3 = 0.0205	disk4 = 0.0041	disk5 = 0.0816
disk6 = 0.0082	disk7 = 0.0164	disk8 = 0.0328

The result of validating this model is satisfactory, as shown below. However, the result is not as satisfactory as the multi-sever model. The main reason is that the assumptions made for representing all the cpus with a single service center do not hold well for the system. In practice, the total service rate of the n cpus can be significantly less than n times the rate of a single cpu because of competition for software locks and interference in accessing main memory. Another thing is that the effective service rate of a multi-processor is not constant, but depends on the processes queued at the center.

	Measured	Computed
(2.00 - 4.00pm of Nov.30)		
ResponseTime	4.1500	0.6410
Queue Length	24	13.2127
ThroughPut	6.9783	7.3209

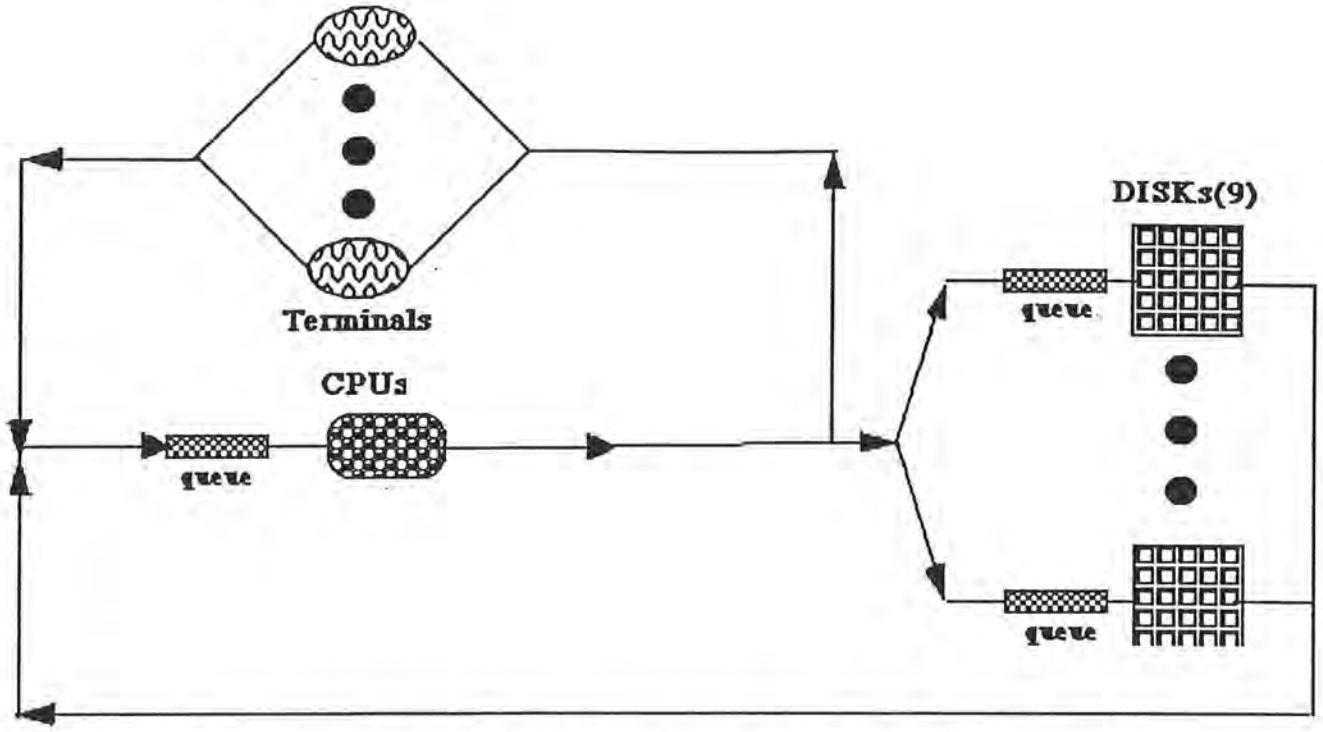


Figure.4 A central-sever queueing network model for Sequent system

CpuUtilization	0.52	0.5448
DiskUtilization		
disk0	0.2571	0.2697
disk1	0.4000	0.4196
disk2	0.4285	0.4496
disk3	0.1428	0.1498
disk4	0.0285	0.0299
disk5	0.5692	0.5972
disk6	0.0571	0.0599
disk7	0.1142	0.1199
disk8	0.2285	0.2398

4.2.3 A multi-sever queueing network model

In a multi-sever model, we represent each cpu as a single sever. The multi-sever model for Sequent system is shown in Figure 5.

The result of validating this model is very satisfactory, including the case when something very strange was happening to the system during 9:30am-11:30am of Dec. 12, 1991.

Model parameters:

Population: P = 534 #of service centers K= 17 (8 cpu and 9 disks)

Think time: Z = 72.3703 (sec.)

Cpu Service Demand(sec./process): Dc = 0.0745 (sec./process)

Disk Service Demand(sec./process):

disk0 = 0.0369	disk1 = 0.0573	disk2 = 0.0614
disk3 = 0.0205	disk4 = 0.0041	disk5 = 0.0816
disk6 = 0.0082	disk7 = 0.0164	disk8 = 0.0328

	Measured	Computed
(2.00 - 4.00pm of Dec.3, heavy Cpu utilization)		
ResponseTime	4.1428	3.5017
Queue Length	26	25.5663
ThroughPut	7.2080	7.2991
CpuUtilization	0.7125	0.7215
DiskUtilization		
disk0	.4	0.4050
disk1	0.4857	0.4918
disk2	0.4285	0.4339
disk3	0.1428	0.1446
disk4	0.0571	0.0578
disk5	0.5384	0.5452
disk6	0.0857	0.0868
disk7	0.4	0.4050
disk8	0.2285	0.2314

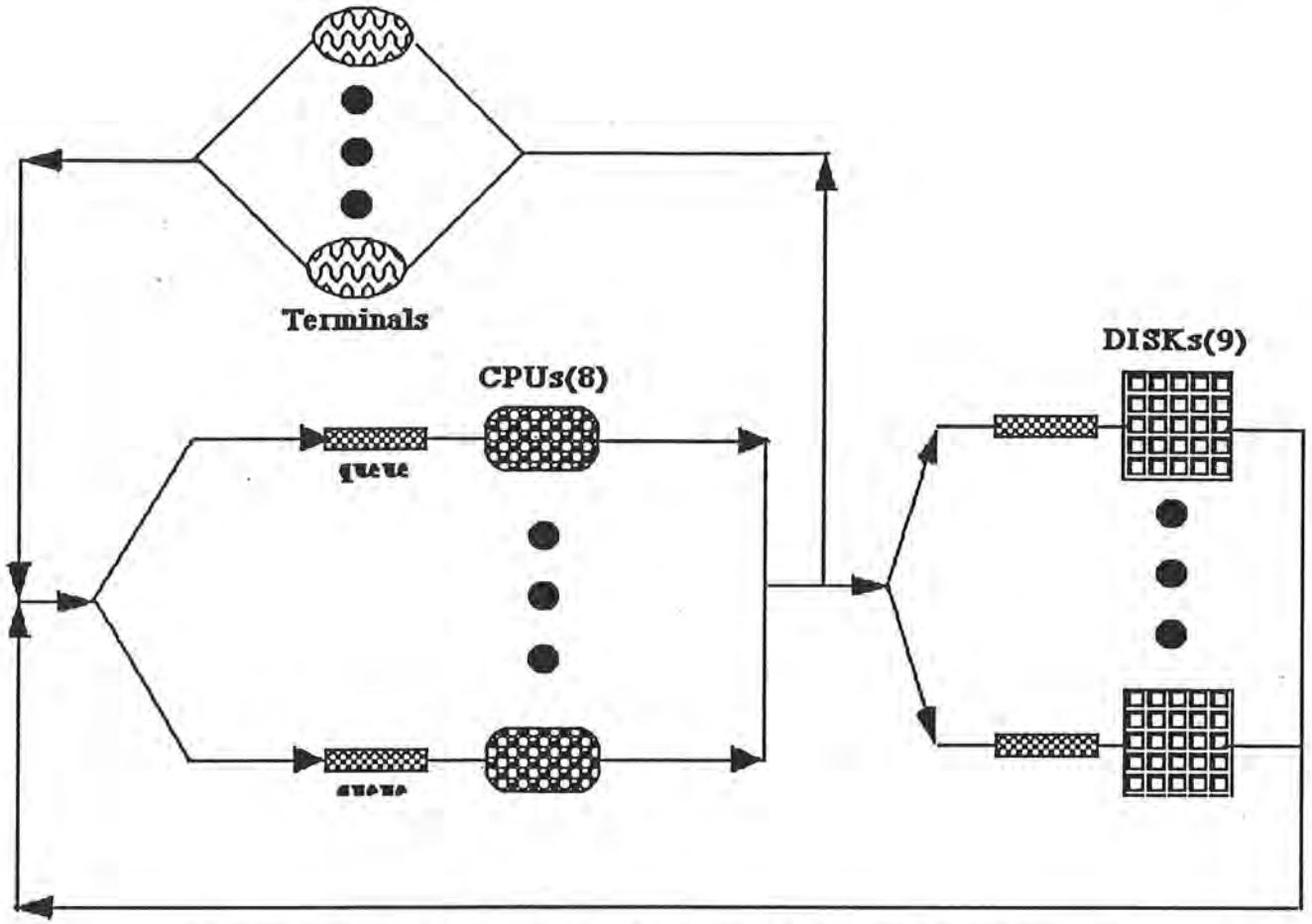


Figure.5 A multi-sever queueing network model for Sequent system

(2.00 - 4.00pm of Nov.30, light Cpu utilization)

ResponseTime	4.1500	1.8358
Queue Length	24	13.2127
ThroughPut	6.9783	7.1961
CpuUtilization	0.52	0.5362
DiskUtilization		
disk0	0.2571	0.2651
disk1	0.4000	0.4125
disk2	0.4285	0.4419
disk3	0.1428	0.1475
disk4	0.0285	0.0294
disk5	0.5692	0.5870
disk6	0.0571	0.0589
disk7	0.1142	0.1178

disk8	0.2285	0.2357

(9.30 - 11.30am of Dec.12, unusual case)

ResponseTime	25.5000	6.9906
Queue Length	171.0000	63.2658
ThroughPut	6.9783	9.0501
CpuUtilization	0.4850	0.6398
DiskUtilization		
disk0	0.3142	0.4146
disk1	0.4000	0.5276
disk2	0.4000	0.5276
disk3	0.2571	0.3392
disk4	0.1142	0.1507
disk5	0.7428	0.9800
disk6	0.0571	0.0753
disk7	0.1142	0.1507
disk8	0.2285	0.3015

4.2.4 Multi-sever queueing network models with more than one workload

We have tried to decompose the terminal workload of the system into to different classes such as user-based workloads(based on alias files), command-based workloads(based on the cpu time, disk I/O info from accounting data), or combinations of the two. The effort is not very successful. The main reason is that we do not have the information of how the system resource consumption is distributed among the different classes, though we can manage to group the workload into different classes.

Chapter 5. Expert System Technology

[KINSBE87] [MADEA85] Expert system technology is making its way into computer applications in many different fields. In this chapter, we start with a brief discussion on the application of this technology, and the suitability to make use of expert system in computer CP. We then discusses the inputs/outputs of the expert system. We conclude with the discussion on the current version of the expert system in the project.

5.1 Making use of expert system technology

[MADEA85][ARTIS85]The advent of AI technology has propelled information systems into a new era. The key in AI technology is to develop systems to solve high-level problems normally accomplished using human expertise, thus the concept of an "expert system".

[LEVINE85] [LEVINE84] An expert system is a computer program that uses knowledge and inference procedures to solve problems that are difficult enough to require significant human expertise for their solution. An expert system contains knowledge of a specific area of expertise which it uses to solve problems at a high level of performance, similar to that of a human expert. The application of expert system technology in fields such as health service (for example, MYCIN and PUFF), science (for example, GENESIS and HEURISTICS), and manufacturing (for example, CALLISTO), aerospace industries, and computer design, has achieved great success, and consequently, has attracted commercial and industrial interest.

Expert system technology offers considerable promise for addressing many of the problems associated with the maintenance, delivery and usability of computer systems. Typically, CP of computer systems provides a very good place to explore and make use of expert system technology.

[STROEB85][BERRY90]The CP process is knowledge-intensive. It requires high level knowledge and experience of the capacity planners. Although we may build many traditional tools to assist in CP, such as performance measurement packages, data reduction

programs, performance prediction models, and configuration guidelines, a good deal of expertise is needed to use them effectively. Also, knowledge of the particular system under study and understanding of performance analysis in general are essential for the capacity planner to start and then achieve the desired destination in the capacity planning process. [OSTROW85] Highly trained technological specialists with broad experience are not easy to find. Expert system technology provides a promising opportunity to satisfy this growing demand by encapsulating expert capacity planning knowledge into automated management system.

5.2 Inputs for the capacity expert system

Expert system is a program encoded with knowledge of specific area. Like a human expert needs fact or information before solving a problem in the specific area, expert system also needs inputs based on which the encoded knowledge can be used to solve a problem. Possible inputs for the expert system in the project are listed below.

1. Present capacity and performance of the system. This includes the utilization of the hardware components, such as cpu, disk, memory and channel; response time of the system and the components; through put of the system and the components.
2. Present workload characteristics. The characteristic consist of the demand of different workload classes on different components. That is, how much a job of some class needs the cpu time, disk I/O and memory, etc.; the level of the workload, e.g., the arrival rate of transactions, the population of terminal users and the multiprogramming level.
3. Forecast of the future workload. Answers to questions such as "How much the present workload will increase"; "What and how much will be the new application of the system" provides the forecast of the future workload for the system.
4. Projections of the system performance. Based on the will-be changes on the system, such as the increase of the workload, the change of the configuration of the system, projections of the system performance answer questions such as, "What will be the utilization of the hardware components, such as cpu, disk, memory and channel"; "What will be the response time of the system and the components"; "What will be the Through put of the system and the components".
5. Objectives for the performance of the system

The objectives for the system can be inspected from the user-point of view as well as from the system point of view.

From the user point of view, the objectives includes: timeliness - how promptly the user receives some output from the system, this is often measured as response time (or turnaround time for batch jobs); accuracy - how many errors are discovered by the user in the output; availability - how much time the system is available to the user; cost - how much the user is to be charged for the service offered by the system.

From the system point of view, the objectives includes: utilization of the hardware components, such as cpu, disk, memory and channel; response time of the system and the components; throughput of the system and the components.

The objectives listed above are most of the possible objectives, and there exists interrelationship among them. In practice, often only one or more of them are used.

5.3 Outputs for the capacity expert system

With the information listed in section 5.2, and with some encoded knowledge, the capacity expert should be able to give CP suggestions. We list the the examples of possible suggestions below.

1. Upgrade or add cpus
When cpu is the bottleneck, increasing the capacity of cpu can improve the performance of the system.
2. Upgrade or add disks
When other components behave well, or when application with great demand disk is added, increasing the capacity of the disk may be necessary.
3. Increase memory
Memory may be extended when it becomes a constraint for the system.
4. Increase or upgrade channels
When disks are upgraded or added, the channels may become bottleneck for the disk subsystem. Channels then need to be upgraded or added.
5. Balance load on disks
There may be occasions when some of the disks are always visited while others remain idle. The problem is that the frequently accessed files are concentrated on these disks. The relocation and then evenly distribution of those heavily used files among all disks can balance the load on the disks and accordingly improve the performance of the system.
6. Reschedule the load of the system

When the I/O-intenstive jobs and cpu-intensive jobs are scheduled together respectively, the capacity of the system can not be made use of to the best extent. Rescheduling them may be a good choice to improve the system performance level using the same capacity.

5.4 Current version of the capacity expert system

Nexpert Object provides the inference engine to house the program structure and a good environment to develop the knowledge base. The effort in this project has been on designing and implementing the knowledge base.

The knowledge base built with Nexpert has both objects and rules. The objects are used to represent the objects of interest in the problem domain, while the rules are used to capture the problem-solving knowledge or expertise based on the facts about and relationships among those objects.

1. Objects/Classes in the knowledge base

For the object part of the knowledge base, we have CPU, DISK, MEMORY three classes which correspond to cpus, disks and memory in a computer system. These classes are defined with many properties of interest, such as utilization, etc. The instances of these classes can be created later automatically by the tool based on input configuration information. We can also save the knowledge base with objects for later use. We also have objects/classes which are not correspond to any system resources in a computer system. These objects/classes are mainly the hypotheses.

2. Rules in the knowledge base

[HELLER85] For the rule part, we have encoded some simple rules into the knowledge base so that the expert system can help the user do basic WhatIf analysis. The rules can be grouped several types:

1). Retrieve data ;

2). Determine if the workload is open, closed or mix;

4) Transfer the changes (if any);

3). Evaluate the model;

5) Checking the performance and giving comments and simple suggestions.

We have listed the rules of the current expert system in Appendix A as reference.

Chapter 6. User Manual for Capacity Planner

Capacity Planner is now running on Macintosh. In this chapter, we provide a kind of user manual for Capacity Planner, which reviews its functionalities and explain how to use it.

6.1 Capacity Planner Functionalities

In this section, we only discuss the implemented functionalities. As presented in chapter 3, we have not implemented all the functionalities for Capacity Planner.

6.1.1 Data reduction function

This function involves extracting information from available raw data to satisfy CP need with a set of reduction programs. The programs have mainly been written with Awk and are put together with a script which runs on the Sequent system and can, with logmon and accounting input files, output the information in a format acceptable by the expert system. We have not integrated these programs into the tool yet. You have to collect logmon data and system accounting data.

6.1.2 Modelling function

After you have collected and reduced the raw data, you can feed the extracted information into Capacity Planner and it then builds a baseline model which can be used in later analyses. Since we have validate the model, what you need to do is just to select the menu item as shown in Figure 6.

6.1.3 What-If analysis function

What-If analysis is the most common CP activities. This analysis is to forecast the influence on system performance if some changes are made to the system. In Capacity Planner, we base this analysis on the baseline model and the input from the user. That is, when the user inputs the changes, such as adding or removing terminal users, Capacity Planner

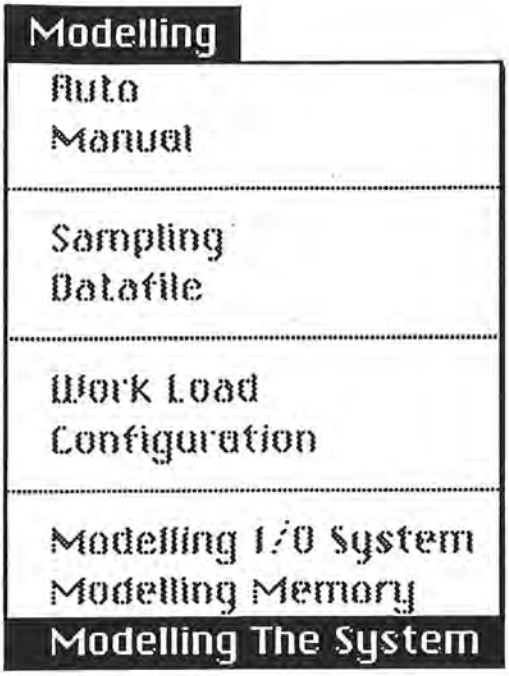


Figure.6 Modelling Menu

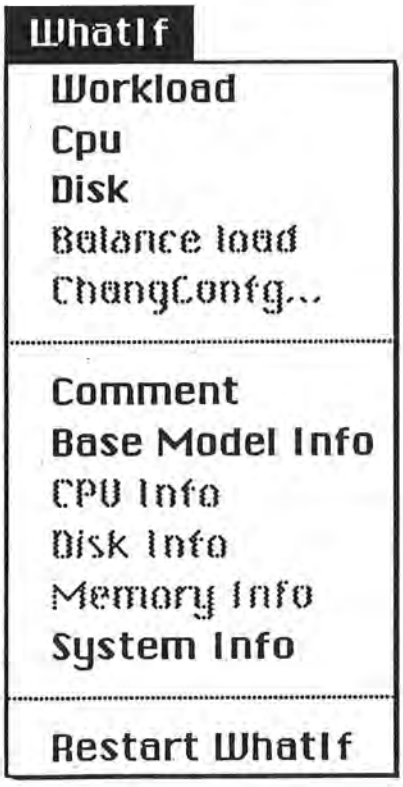


Figure.7 What-If menu

translates the changes into the term of queueing models and adjusts the parameters for the baseline model so as to project the performance for the system.

Before starting the What-If analysis, you may want to have a look at the baseline model which provides you with information for cpus, disks and the terminal users. You can access this information through the menu item "Base Model Info" of menu "WhatIf".

The system performance information provided by Capacity Planner includes average response time, throughput(processes/sec.), Cpu utilization, average disk utilization, and a response time indicator, as shown in Figure 8.

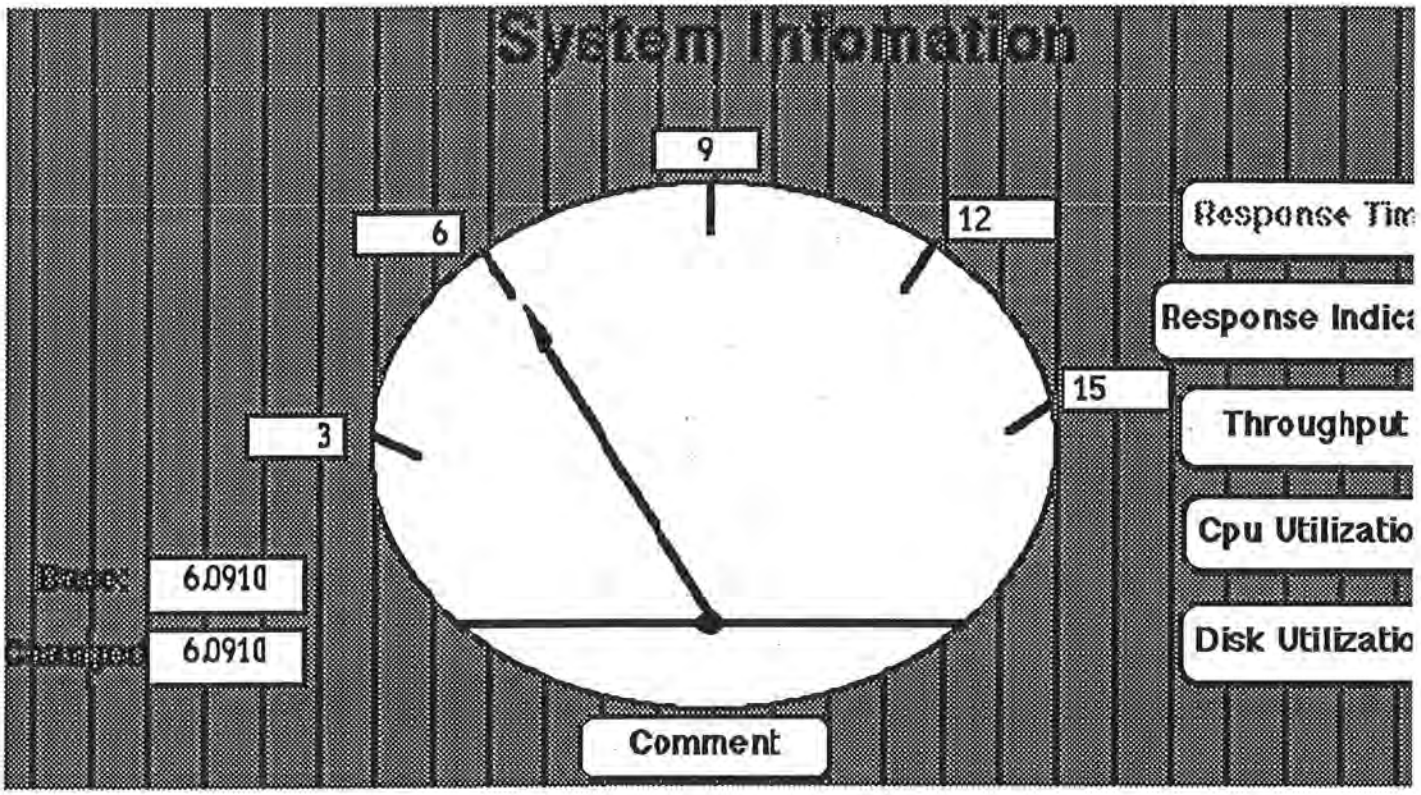


Figure.8 System information provided by Capacity Planner

Changes you can make are:

1. Number of terminal users. That is, you can either add or remove terminal users. This change is made through the window shown in Figure 9. We have assumed the max number of terminal users are 800. When you add terminal users till more than 800, the dial will not show that, but the value will appear in bottom box.

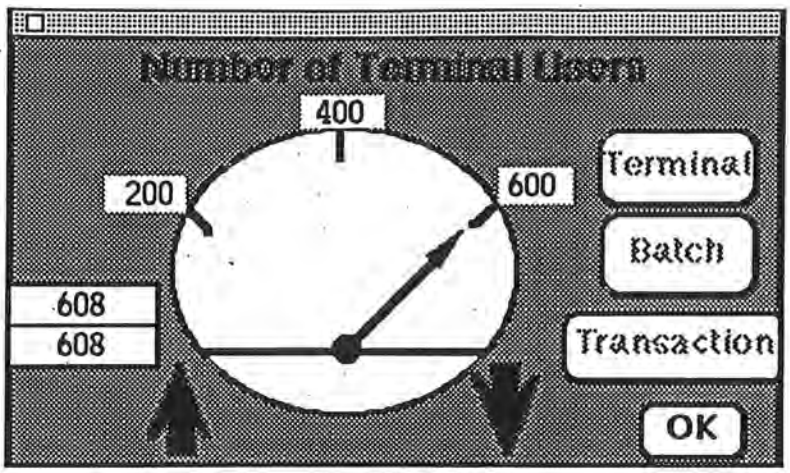


Figure.9 Making changes on workload

2. Speed or number of cpus. The baseline speed pf cpus is assume to be 1. When you increase the speed, say, to 1.5, you have increased the speed of cpus by 50%. Similarly, when you decrease the speed of cpus, say, to .75, you have decrease the speed of cpus by 25%. The baseline value of number of cpus is 8 for Sequent system, you can either add more or remove cpus. The changes are made through the window shown in Figure 10.

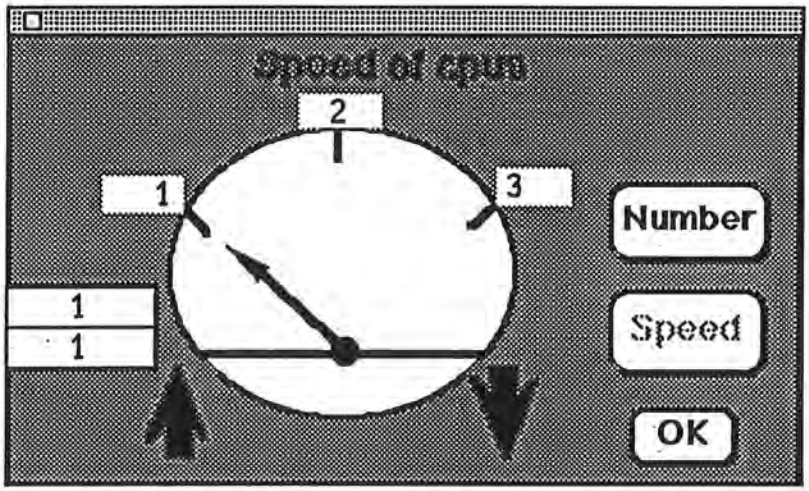


Figure.10 Making changes on number and speed of cpus

3. Speed of every disk. You can change the speed of every disk similar to changing that of cpus. The change is made through the window shown in Figure 11.

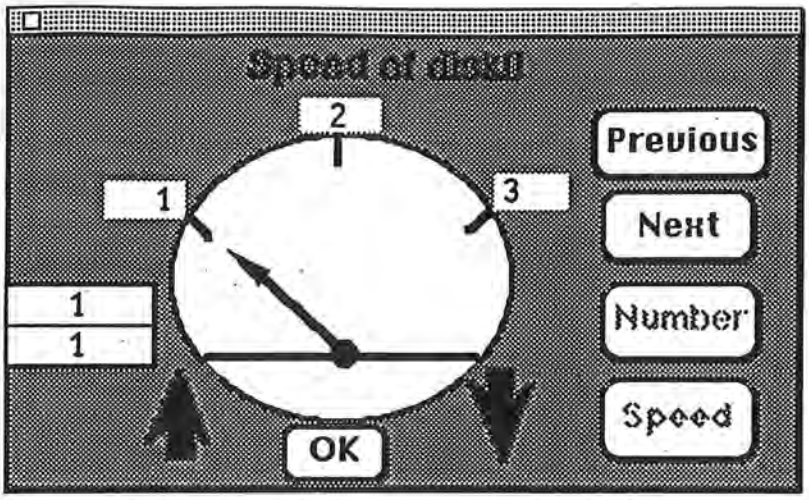


Figure.11 Making changes on speed of disks

Capacity Planner also provides comments on performance of the system. Right now, it can track what changes you have made, check system performance and resource utilization to provide some simple comments/suggestions, as shown in Figure 12.

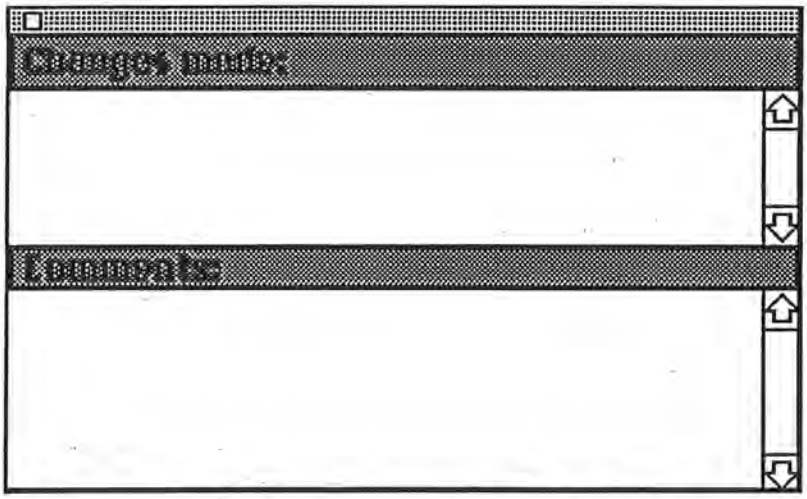


Figure.12 Displaying comments and changes made

6.2 An Example

Now, we will walk through an example showing 'how' to use Capacity Planner. We will start from the very begining - from collecting the raw data.

1. Data collection and reduction

Logmon and system accounting data is needed. You have to collect these two data files over same time period. Then, you can input the data files into the reduction programs to produce the parameter file needed to build the baseline model.

2. Start up.

The visible part of Capacity Planner to the user is the SuperCard interface. This interface is a project of SuperCard (just like an application in C). It can be launched by double click on the project, as shown in Figure 13.

Figure.13 Capacity Planner

The initial appearance of Capacity Planner is shown in Figure 14.

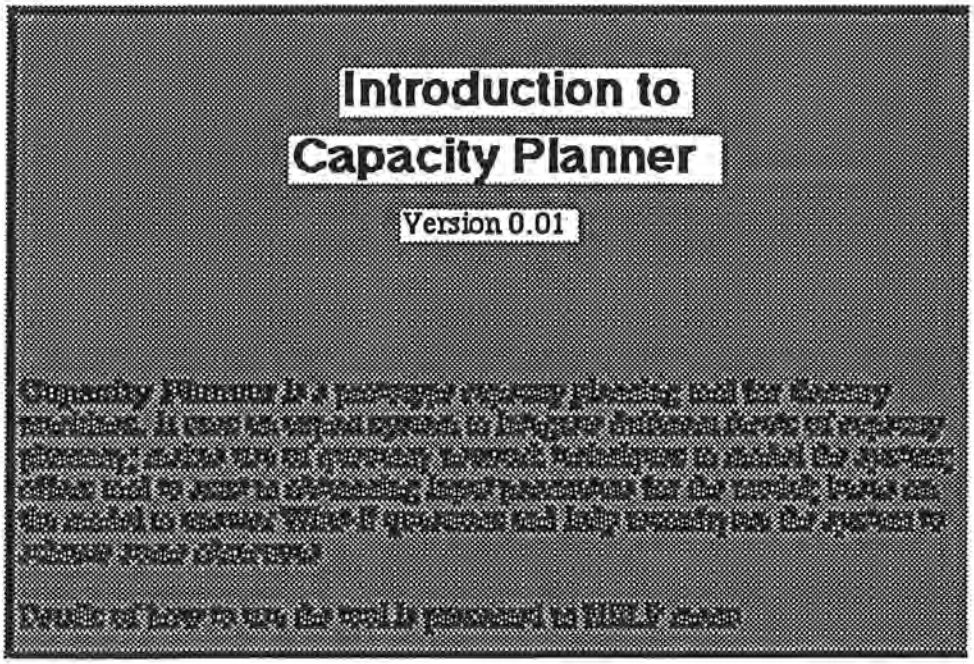


Figure.14 The initial appearance of Capacity Planner

3. Load knowledge base

After the tool is set up, the user can specify which knowledge base to be loaded, or the default one will be loaded. The menu item "Load knowledge" of menu "Expert" allows the

user to do so. The link between the Supercard and the expert system has been initialized on the start of the tool, the knowledge base must be loaded before the planner can be used though.

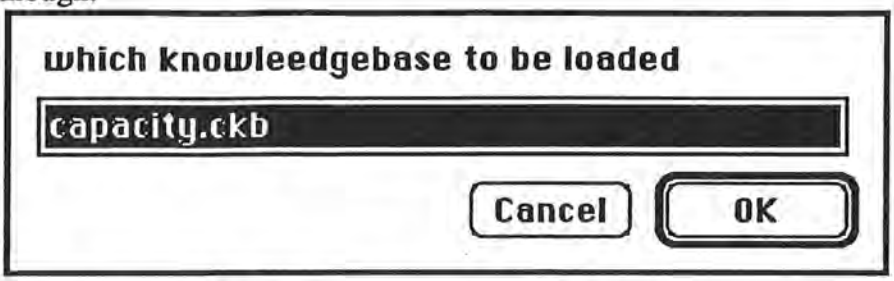
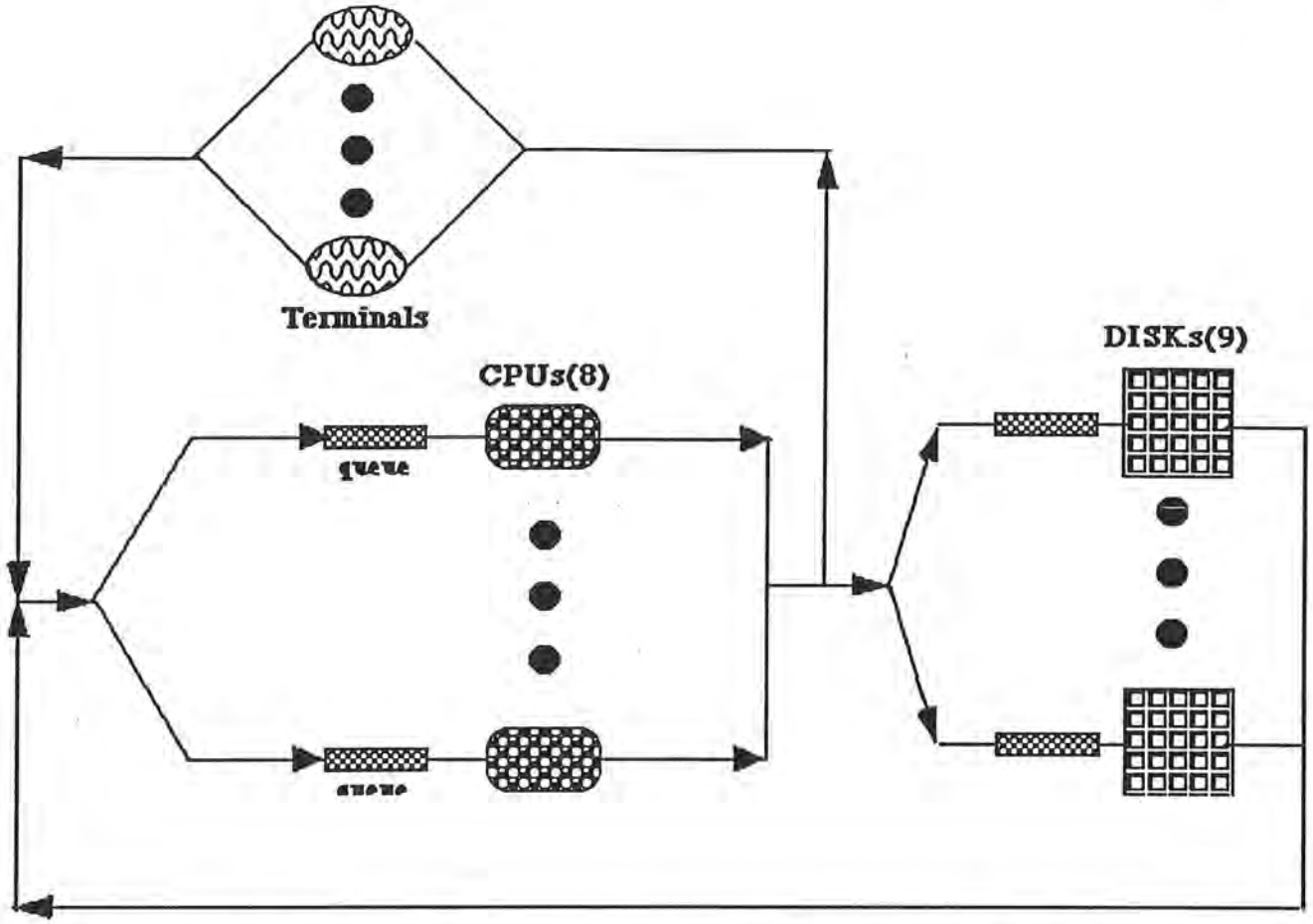


Figure.15 Loading knowledge base

The WhatIf analyses and Configuring functions of the planner are disabled right now, that's because a baseline model is needed to complete these functions.

4. Build baseline model

Two options of modelling the system are provided: "Manual" and "Auto". When "Manual" is selected, the user has to do all the data collection and reduction, and then input the result into the tool. "Auto", on the other hand, means the planner should automatically collect



Note: Click on terminal, memory, cpu or disk to see the detail information.

Figure.16 Baseline Model Information

and reduce the data and build baseline model for the system. By now, only Manual mode is implemented. That is, you have to collect raw data, feed the raw data into the reduction programs to produce a file "parameter.nxp", and put the file into the same folder as Capacity Planner.

So, choose "Manual" menu item from menu "Modelling" and then builds the baseline model by selecting menu item "Modelling the System". You can access the details of the baseline model by choosing "Basemodel Info" from menu WhatIf. After you see the model as shown in Figure 7, click the terminals, cpus, or disks to get their detailed information

5. What-If analysis

Now that you have built the baseline model, you can start "What If" analyses at this stage. The WhatIf analyses and Configuring functions of the planner can be based on models for the system, the I/O subsystem or the memory subsystem, though only the system level model has been built.

You can have a look at the system level information by selecting menu item "SystemInfo". If no change has been made yet, only the base value is shown in the window named "System Information", as shown in Figure 8.

Now, you can make changes to the system and look at the effect. For example, add 100 terminal users to the system, add one more cpu and upgrade disk 5 to match the increased load. After making the changes through according windows, click button OK, the expert system then projects the performance measurement based on the changes and the baseline model. The influence of the changes is displayed in the "System Information" window, as shown in Figure 17: the system response time has increased, etc.; click button "comment", you can see the comments made by the expert system, as also shown in Figure 17.

You may want to keep making more changes or start a new What-If analysis session. To clean up the changes and begin a new WhatIf analyses session, just restart WhatIf.

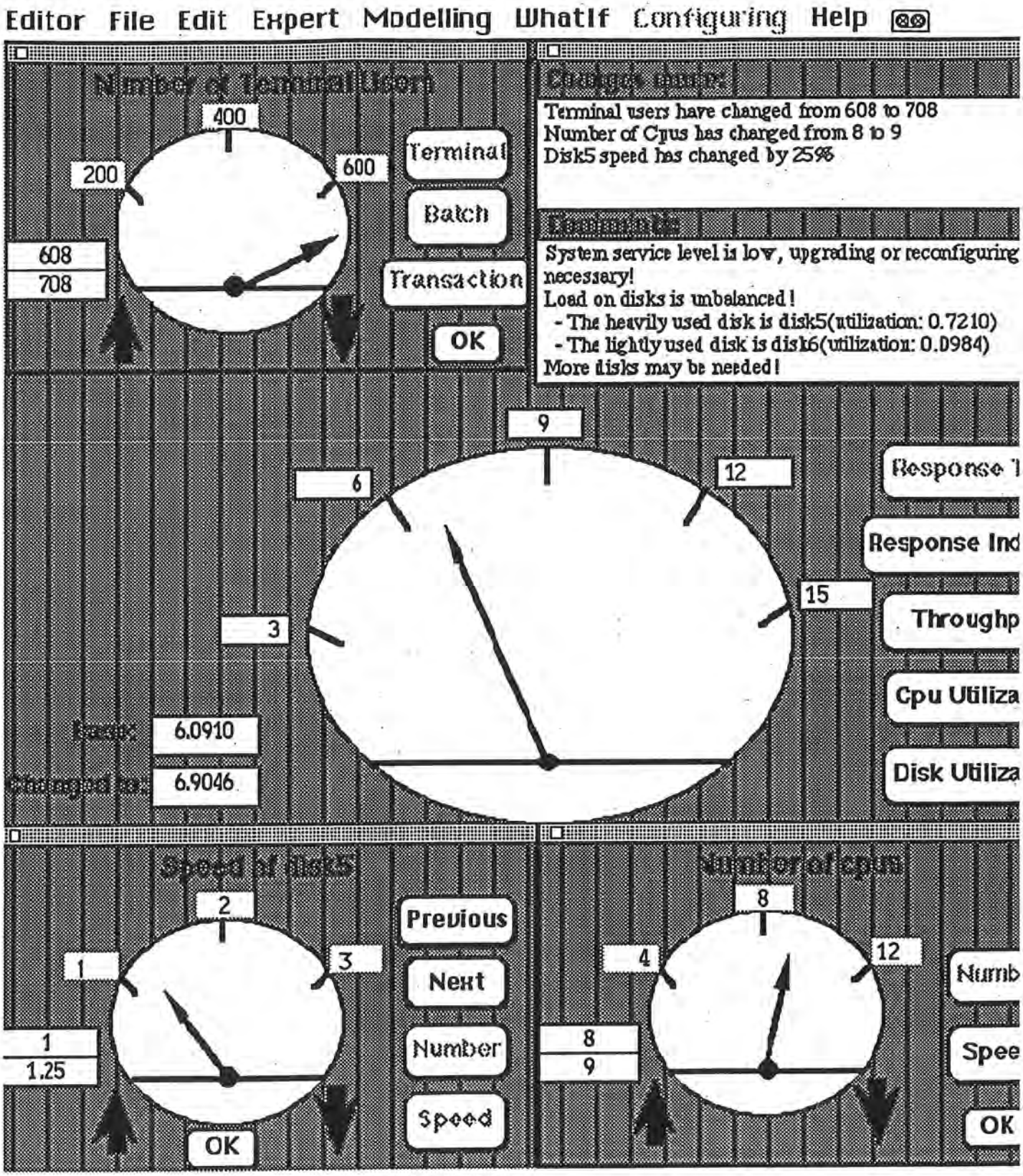


Figure.17 Doing What-If analyses

Chapter 7. Conclusion

7.1 Conclusion

In this project, we have surveyed the current status of CP function and its supporting tools in UNIX environment. From this survey, it is clear that basic raw data for CP purpose is available in UNIX environment, but the tools to make use of it to support CP are lacking, while the need for CP is gradually recognized.

We have built a prototype tool to explore the possibility of bridging this gap. We have built a queueing network model for Sequent system and have integrated it into the prototype. We have also mapped the problem domain into the expert system of the prototype and encoded it with some basic knowledge.The prototype tool is now running on MAC platform with relatively satisfactory performance, valuable capacities and promising potentials.

7.2 Future Work

As discussed above, part of the specified features of Capacity Planner has been implemented. Yet, much remains to be done. A list of possible future work is given below.

1. Build more efficient reduction package.
Since the reduction programs in the tool are mostly written in Awk, they can not be as efficient as the programs written in lower level programing languages such as C. Converting these programs into C or other languages can make the data reduction much more efficient.

2. Integrate data reduction and collection utilities into Capacity Planner.
Right now, the user has to turn on/off the data collecting utilities. The collected data files have to be manually reduced with the reduction programs. It will be nice if the user can issue some command through the tool and the tool will then automatically take care of this process.

3. Model memory and disk subsystems.

Especially when memory becomes a constraint for the system, it needs to be modelled and analyzed to solve the CP problems. Also, disk subsystems play an important role in the overall performance of the system. The details of these subsystems are often of interest to the capacity planners. If they can be modelled, the models will provide capacity planners with great help.

4. Make use of simulation modelling approach.
As discussed in Chapter 4, many assumptions are made before models could be built for a system. Simulation models, though they are much more expensive, are more flexible and accommodate less strict assumptions than queueing network models. When the assumptions do not hold well, or when the assumptions have resulted in some untrackable models for queueing network technique, simulation models or the combination of the two can be good alternatives.

5. Get the distribution of response times.
Mean value approach has been adopted in evaluating the model, which produce mean value for all the performance measures, including response time, the most visible performance index to users of computer systems. End-users see individual response times, not the average. Therefore, the distribution of response times is important to CP studies.

6. Encode more practical expertise
As mentioned before, only simple rules have been encoded into the knowledge base. It's essential to acquire more practical knowledge and encoded it into the knowledge base.

REFERENCES

[ARTIS78] H.P. Artis, 1978. *Capacity Planning for the MVS Computer System,* in Ferrari. Ed. Performance of Computer Installations, North-Holland, 25-54.

[ARTIS85] H. P. Artis, 1985. *Using Expert Systems for Analyzing RMF Data,* CMG '85 conference proceeding, 653-657.

[BARNES79] M. F. Barnes, 1979. *Measurement and Modelling Methods for Computer Systems performance studies,* Longton Information Systems Ltd.

[BERRY90] R. Berry, J. Hellerstein, 1990. *Expert Systems for Capacity Management,* CMG'90 transactions, 85-92.

[BLAYLO83] J. W. Blaylock, 1983. *Capacity Planning for Multiptocessor and Multicomputer systems,* CMG '83 conference proceeding, 37-44.

[BOLKER87] E. D. Bolker, 1987. *A Capacity Planning/Queueing Theory Primer or How Far Can You Go on the Back-of -an-Envelop?* CMG '87 conference proceeding, 734-735.

[BOWERM83] J. R. Bowerman, 1983. *A Tool for Capacity and Configuration Analysis on Large Scale Interactive Systems,* CMG '83 conference proceeding, 76-83.

[BRONNE80] L. Bronner, 1980. *Overview of the capacity Planning Process for Production Data Processing,* IBM System Journal, vol 19, no.1, 4-27.

[CABREA81] L.F. Cabrea, 1981. *Benchmarking UNIX - a Comparative Study, Experimental Computer Performance and Evaluation,* Ferrari, D. Spadoni M.(eds), 205-215.

[CADY90] J. Cady, B. Howarth, 1990. *Computer System Performance management and Capacity Planning,* Prentice Hall.

[COOPER80] J.C. Cooper, 1980. *A Capacity Planning Methodology,* IBM System Journal, vol 19, no.1, 28-45.

[DAWSON88] J.L Dawson, 1988. *Interactive Expert Assistant for Computer Performance Evaluation,* CMG '88 conference proceeding, 244-247.

[DEESE88] D.R. Deese, 1988. *Designing an Expert System for Computer Performance Evaluation,* proceedings of CMG'88, 75-80.

[DENNIN77] P.J. Denning, J.P. Buzen, 1977. *Operational Analysis of Queueing Networks,* proceedings of the Third International Symposium on Measuring, Modelling and Evaluating Computer Systems, 151-172.

[DOMANS88] B. Domanski, S.W. Soberman, 1988. *Expert Systems for Computer Performance Evaluation: on MetaRules and Knowledge Engineering,* CMG '88 conference proceeding, 249-253.

[FEDER84] J. Feder, 1984. *The UNIX System: the Evolution of UNIX System Performance,* AT&T Bell Laboratories Technical Journal, Vol. 63, No. 8, Oct. 1984, 1791-1814.

[FERRAR83] D. Ferrari, etc. 1983. *Measurement and Tuning of Computer Systems,* Prentice-Hall, Inc.

[GADE81] S.L. Gade, 1981. *Tools for Research in Computer Workload Characterization and Modelling,* Experimental Computer Performance and Evaluation, D. Ferrari and M. Spadoni (eds), North-Holland Publishing Company, 235-247.

[HACKEN87] S. R. Hackenberg, 1987. *A Heuristic Modelling Approach to CICS Capacity Planning,* CMG '87 conference proceeding, 252-254.

[HARING83] G. Haring, R. Posch, 1983. *A Simple Functional Description of Computer System,* Journal of Capacity Management, vol.1, no.3, 291-305.

[HELLER85] J. Hellerstein, H. V. Woerkom, 1985. *YSCOPE: a Shell for Building Expert System for Solving Computer Performance Problems,* CMG '85 conference proceeding, 170-180.

[HOOVER89] S.V. Hovver, R.F. Perry, 1989. *Simulation*, Addison-Wesley publishing company.

[HOWARD90] P.C. Howard, Ed. 1990. *IS Capacity Management Handbook Series*, the Institute for Computer Capacity Management.

[HOWE85] D. J. Howe, 1985. *A Basic Approach to a Capacity Planning Methodology*, CMG '85 conference proceeding, 684-689.

[HUGO83] I.St.J. Hugo, 1983. *Capacity Management: A Multi-Dimentional Picture*, Journal of Capacity Management, vol.1, no.3, 199-208.

[INGRAH87] B. A. Ingraham, 1987, *What We Can Learn from other Capacity Planners: a Survey of Quantitative Methods*, CMG '87 conference proceeding, 504-507.

[KINSBE87] J. V. Kinsbergen, 1987. *Expert Systems and MVS Performance Management*, CMG '87 conference proceeding, 233-236.

[KRAUSE88] Krause,A.M., Silverstein, S.D. 1988. *Starting from Scratch: the Evolution of a Capacity/Performance Function in a UNIX environment*, CMG '88 conference proceeding, 738-747.

[LAM87] S.F. Lam, K.H. Chan, 1987. *Computer Capacity Planning*, Academic Press, Inc., Harcourt Brace Jovanovich, Publishers.

[LAZOWS84] E.D. Lazowska, etc. 1984. *Quantitative System Performance - Computer System Analysis Using Queueing Network Models*, Prentice-Hall, Inc.

[LAZOWS85] E.D. Lazowska, 1985. *Capacity Planning Using Queueing Network Models*, CMG '85 conference proceeding, 668-670.

[LEVINE84] A. P. Levine, 1984. *An Expert System for Computer Performance Modeling: Design Issues*, CMG '84 conference proceeding, 227-232.

[LEVINE85] A. P. Levine, 1985. *ESP: an Expert System for Computer Performance Management*, CMG '85 conference proceeding, 181-186.

[LEUNG88] C. H. C. Leung, 1988. *Quantive Analysis of Computer Systems,* Biddles Ltd, Guildford.

[MADEA85] A. T. Madea, L, A. Aho, 1985. *Using Expert System Technology to Evaluate System Resource Planning Alternatives: a Management Perspective,* CMG '85 conference proceeding, 558-563.

[MITRAN87] I. Mitrani, 1987. *Modelling of Computer and Communication systems,* the University Press, Cambidge .

[OSTROW85] J. M. Ostrowski, 1985. *Implementing Expert System Technology into the Field of Computer Performance Evaluation,* CMG '85 conference proceeding, 245-249.

[PIRKLE83] W. L. Pirkle, J. Tyan & T. Liu, 1983. *TUFF: a Capacity Planning System That Thinks for Itself,* CMG '83 conference proceeding, 21-26.

[POTIER85] D. Poiter, ed., 1985. *Modelling Techniques and Tools for Performance Analysis,* Elsevier Scince Publishers B.V.

[SALAWU85] K. O. Salawu, 1985. *Capacity Planning for UNIX Systems,* CMG '85 conference proceeding, 76-79.

[SAUER79] C. H. Sauer, E. A. Macnair, 1979. *Queueing Network Software for Systems Modelling,* Software - Practice and Experience, Vol. 9, 369 - 380.

[SAUER81] C. H. Sauer, K. M. Chandy, 1981. *Computer Systems Performance Modelling,* Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632.

[SCHILL80] D.C. Schiller, 1980. *System Capacity and Performance Evaluation,* IBM System Journal, vol 19, no.1, 45-67.

[SHUB87] A. H. Shub, 1987. *PROFS Capacity Planning Issues,* CMG '87 conference proceeding, 110-112.

[STROEB85] G. J. Stroebel, R. D. Baxter and M. J. Denney, 1985. *A caapcity Planning Expert System for IBM System/38,* CMG '85 conference proceeding, 204-212.

[SVOBOD76] L. Svobodova, 1976. *Computer Performance Measurement and Evaluation Methods: Analysis and Applications,* American Elsevier Publishing Company, Inc.

[TORNHE87] J. Jerik Tornheim, R. T. Williams, 1987. *Workload Characterization Using UNIX System Accounting,* CMG '87 conference proceeding, 351-355.

Appendix A

List of Rules

RULE : Rule 1
If
<|DISK|>.responsetimeBatch is KNOWN
And <|DISK|>.responsetimeTerm is KNOWN
And <|DISK|>.responsetimeTrans is KNOWN
Then CacuDiskRes.yes
is confirmed.
And SUM(<|DISK|>.responsetimeBatch)+SUM(<|DISK|>.responsetimeTerm) is assigned to |DISK|.responseTime

RULE : Rule 2
If
CacuOutputClose.yes is assigned to CacuOutputClose.yes
And CacuOutputOpen.yes is assigned to CacuOutputOpen.yes
And CacuOutputMix.yes is assigned to CacuOutputMix.yes
Then CacuOutput.yes
is confirmed.
And Reset <|resetHypos|>.yes
And CacuSysCpu.yes is assigned to CacuSysCpu.yes
And CacuSysDisk.yes is assigned to CacuSysDisk.yes
And
SUM(<|DISK|>.responsetimeBatch)+SUM(<|DISK|>.responsetimeTerm)+SUM(<|cpu|>.responsetimeTerm)+SUM(<|cpu|>.responsetimeBatch) is assigned to system.responseTime

RULE : Rule 3
If
there is no evidence of openLoad
And there is evidence of closeLoad
Then CacuOutputClose.yes
is confirmed.
And Execute
"apprMva.exe"(@TYPE=EXE;@ATOMID=<|cpu|>,<|DISK|>,workLoadTerm,workLoadBatch;)

RULE : Rule 4
If
there is evidence of openLoad
And there is evidence of closeLoad
Then CacuOutputMix.yes
is confirmed.
And <|cpu|>.servdemandTerm/(1-<|cpu|>.utilizationTrans) is assigned to <|cpu|>.servdemandTerm

And <|cpu|>.servdemandBatch/(1-<|cpu|>.utilizationTrans) is assigned to <|cpu|>.servdemandBatch
And <|DISK|>.servdemandTerm/(1-<|DISK|>.utilizationTrans) is assigned to <|DISK|>.servdemandTerm
And <|DISK|>.servdemandBatch/(1-<|DISK|>.utilizationTrans) is assigned to <|DISK|>.servdemandBatch
And Execute "apprMva.exe"(@TYPE=EXE;@ATOMID=<|cpu|>,<|DISK|>,workLoadTerm,workLoadBatch;)

RULE : Rule 5
If
CacuOutputClose.yes is assigned to CacuOutputClose.yes
And there is evidence of openLoad
Then CacuOutputOpen.yes
is confirmed.
And <|cpu|>.servdemandTrans*(1+<|cpu|>.queLenTerm+<|cpu|>.queLenBatch)/(1-<|cpu|>.utilizationTrans) is assigned to <|cpu|>.responsetimeTrans
And workLoadTrans.arrivalRate*<|cpu|>.responsetimeTrans is assigned to <|cpu|>.queLenTrans
And <|DISK|>.servdemandTrans*(1+<|DISK|>.queLenTerm+<|DISK|>.queLenBatch)/(1-<|DISK|>.utilizationTrans) is assigned to <|DISK|>.responsetimeTrans
And workLoadTrans.arrivalRate*<|DISK|>.responsetimeTrans is assigned to <|DISK|>.queLenTrans

RULE : Rule 6
If
cpu0.utilizationTerm+cpu0.utilizationBatch+cpu0.utilizationTrans is assigned to system.cpuUtil
Then CacuSysCpu.yes
is confirmed.
And satCpu is assigned to satCpu

RULE : Rule 7
If
AVERAGE(<|DISK|>.utilizationTerm)+AVERAGE(<|DISK|>.utilizationBatch)+AVERAGE(<|DISK|>.utilizationTrans) is assigned to system.diskUtil
And <|DISK|>.utilizationTerm+<|DISK|>.utilizationBatch+<|DISK|>.utilizationTrans is assigned to <|DISK|>.utilization
And AVERAGE(<|DISK|>.utilizationR) is assigned to system.diskUtilR
And MAX(<|DISK|>.utilization) is assigned to system.max
And MIN(<|DISK|>.utilization) is assigned to system.min
Then CacuSysDisk.yes
is confirmed.
And unbalanceDisk is assigned to unbalanceDisk
And getName is assigned to getName

```
RULE : Rule 8
If
        <|workClassC|>.Increase is not equal to 0
Then CLoadIncrease.yes
        is confirmed.
        And <|workClassC|>.population+<|workClassC|>.Increase is assigned to
<|workClassC|>.population
        And Reset <|outputHypo|>.yes
        And resetCLoadIncrease is assigned to resetCLoadIncrease
        And Execute "sugg.exe"(@TYPE=EXE;@ATOMID=CacuOutput.yes;)


RULE : Rule 9
If
        <|workClassC|>.population is not equal to 0
Then closeLoad
        is confirmed.


RULE : Rule 10
If
        balanceDisk is assigned to balanceDisk
        And increaseDiskRes is assigned to increaseDiskRes
        And reduceDiskRes.yes is assigned to reduceDiskRes.yes
Then confgDiskRes
        is confirmed.


RULE : Rule 11
If
        cpu0.speedincrease is not equal to 0
Then CpuServDemandChange.yes
        is confirmed.
        And <|cpu|>.servdemandBatch/(1+cpu0.speedincrease) is assigned to
<|cpu|>.servdemandBatch
        And <|cpu|>.servdemandTerm/(1+cpu0.speedincrease) is assigned to
<|cpu|>.servdemandTerm
        And Reset <|outputHypo|>.yes
        And Execute "sugg.exe"(@TYPE=EXE;@ATOMID=CacuOutput.yes;)


RULE : Rule 20
If
        disk0.speedincrease is not equal to 0
Then DiskServDemandChange.yes
        is confirmed.
        And disk0.servdemandBatch/(1+disk0.speedincrease) is assigned to
disk0.servdemandBatch
        And disk0.servdemandTerm/(1+disk0.speedincrease) is assigned to
disk0.servdemandTerm
        And Reset <|outputHypo|>.yes
        And Execute "sugg.exe"(@TYPE=EXE;@ATOMID=CacuOutput.yes;)
```

RULE : Rule 19
If
disk2.speedincrease is not equal to 0
Then DiskServDemandChange.yes
is confirmed.
And disk2.servdemandBatch/(1+disk2.speedincrease) is assigned to disk2.servdemandBatch
And disk2.servdemandTerm/(1+disk2.speedincrease) is assigned to disk2.servdemandTerm
And Reset <loutputHypol>.yes
And Execute "sugg.exe"(@TYPE=EXE;@ATOMID=CacuOutput.yes;)

RULE : Rule 18
If
disk1.speedincrease is not equal to 0
Then DiskServDemandChange.yes
is confirmed.
And disk1.servdemandBatch/(1+disk1.speedincrease) is assigned to disk1.servdemandBatch
And disk1.servdemandTerm/(1+disk1.speedincrease) is assigned to disk1.servdemandTerm
And Reset <loutputHypol>.yes
And Execute "sugg.exe"(@TYPE=EXE;@ATOMID=CacuOutput.yes;)

RULE : Rule 17
If
disk3.speedincrease is not equal to 0
Then DiskServDemandChange.yes
is confirmed.
And disk3.servdemandBatch/(1+disk3.speedincrease) is assigned to disk3.servdemandBatch
And disk3.servdemandTerm/(1+disk3.speedincrease) is assigned to disk3.servdemandTerm
And Reset <loutputHypol>.yes
And Execute "sugg.exe"(@TYPE=EXE;@ATOMID=CacuOutput.yes;)

RULE : Rule 16
If
disk4.speedincrease is not equal to 0
Then DiskServDemandChange.yes
is confirmed.
And disk4.servdemandBatch/(1+disk4.speedincrease) is assigned to disk4.servdemandBatch
And disk4.servdemandTerm/(1+disk4.speedincrease) is assigned to disk4.servdemandTerm
And Reset <loutputHypol>.yes
And Execute "sugg.exe"(@TYPE=EXE;@ATOMID=CacuOutput.yes;)

RULE : Rule 15
If

```
        disk5.speedincrease is not equal to 0
Then DiskServDemandChange.yes
        is confirmed.
        And disk5.servdemandBatch/(1+disk5.speedincrease) is assigned to
disk5.servdemandBatch
        And disk5.servdemandTerm/(1+disk5.speedincrease) is assigned to
disk5.servdemandTerm
        And Reset <|outputHypo|>.yes
        And Execute "sugg.exe"(@TYPE=EXE;@ATOMID=CacuOutput.yes;)


RULE : Rule 14
If
        disk6.speedincrease is not equal to 0
Then DiskServDemandChange.yes
        is confirmed.
        And disk6.servdemandBatch/(1+disk6.speedincrease) is assigned to
disk6.servdemandBatch
        And disk6.servdemandTerm/(1+disk6.speedincrease) is assigned to
disk6.servdemandTerm
        And Reset <|outputHypo|>.yes
        And Execute "sugg.exe"(@TYPE=EXE;@ATOMID=CacuOutput.yes;)


RULE : Rule 13
If
        disk7.speedincrease is not equal to 0
Then DiskServDemandChange.yes
        is confirmed.
        And disk7.servdemandBatch/(1+disk7.speedincrease) is assigned to
disk7.servdemandBatch
        And disk7.servdemandTerm/(1+disk7.speedincrease) is assigned to
disk7.servdemandTerm
        And Reset <|outputHypo|>.yes
        And Execute "sugg.exe"(@TYPE=EXE;@ATOMID=CacuOutput.yes;)


RULE : Rule 12
If
        disk8.speedincrease is not equal to 0
Then DiskServDemandChange.yes
        is confirmed.
        And disk8.servdemandBatch/(1+disk8.speedincrease) is assigned to
disk8.servdemandBatch
        And disk8.servdemandTerm/(1+disk8.speedincrease) is assigned to
disk8.servdemandTerm
        And Reset <|outputHypo|>.yes
        And Execute "sugg.exe"(@TYPE=EXE;@ATOMID=CacuOutput.yes;)


RULE : Rule 22
If
        system.max-<|DISK|>.utilization is precisely equal to 0
Then getName
```

```
        is confirmed.
        And <|DISK|>.name is assigned to system.maxName

RULE : Rule 21
If
        system.min-<|DISK|>.utilization is precisely equal to 0
Then getName
        is confirmed.
        And <|DISK|>.name is assigned to system.minName

RULE : Rule 23
If
        SUM(<|DISK|>.responsetimeTerm)+SUM(<|DISK|>.responsetimeBatch)-
|DISK|.responseTimeObj is less than 0
Then increaseDiskRes
        is confirmed.

RULE : Rule 24
If
        <|workClassO|>.Increase is not equal to 0
Then OLoadIncrease.yes
        is confirmed.
        And <|workClassO|>.arrivalRate*(1+<|workClassO|>.Increase) is assigned to
<|workClassO|>.arrivalRate

RULE : Rule 25
If
        <|workClassO|>.arrivalRate is not equal to 0
Then openLoad
        is confirmed.
        And <|cpu|>.servdemandTrans*workLoadTrans.arrivalRate is assigned to
<|cpu|>.utilizationBatch
        And <|DISK|>.servdemandTrans*workLoadTrans.arrivalRate is assigned to
<|DISK|>.utilizationBatch

RULE : Rule 26
If
        SUM(<|DISK|>.utilizationTerm)+SUM(<|DISK|>.utilizationTrans)+SUM(<|DISK|
>.utilizationBatch)-0.5 is greater than 0
Then overUseDisk
        is confirmed.
        And Execute "comment"(@ATOMID=<|DISK|>.name;@STRING="are saturated,
add more disks.";)

RULE : Rule 27
If
        SUM(<|DISK|>.responsetimeTerm)+SUM(<|DISK|>.responsetimeTrans)+SUM(<
|DISK|>.responsetimeBatch)-|DISK|.responseTimeObj is greater than 0
```

```
        And MAX(<|DISK|>.responseTime) is assigned to max
        And <|DISK|>.responseTime-max is precisely equal to 0
Then reduceDiskRes.yes
        is confirmed.
        And <|DISK|>.speedincrease+diskSpeedIncrs is assigned to
<|DISK|>.speedincrease
        And Reset CacuOutput.yes
        And Reset DiskServDemandChange.yes


RULE : Rule 28
If
        cpu0.speedincrease is KNOWN
Then resetCincrease
        is confirmed.
        And Reset resetCincrease
        And Reset cpu0.speedincrease


RULE : Rule 29
If
        <|workClassC|>.Increase is KNOWN
Then resetCLoadIncrease
        is confirmed.
        And Reset <|workClassC|>.Increase
        And Reset resetCLoadIncrease


RULE : Rule 30
If
        <|workClassC|>.Increase is KNOWN
Then resetWork
        is confirmed.
        And Reset <|workClassC|>.Increase
        And Reset resetWork


RULE : Rule 31
If
        Retrieve "dec12.2.nxp"
@TYPE=NXP;@FWRD=FALSE;@UNKNOWN=TRUE;  \* retrieve the data file named
"dec12.2.nxp". *\
Then retrieveAll
        is confirmed.


RULE : Rule 32
If
        cpu0.utilization-0.9 is greater than or equal to 0
Then satCpu
        is confirmed.
        And Execute "comment"(@ATOMID=cpu0.name;@STRING="is saturated, add
more CPU or Increase the speed of CPU";)
```

RULE : Rule 33
If
ABS(SUM(<|DISK|>.utilizationTerm)+SUM(<|DISK|>.utilizationTrans)+SUM(<|DISK|>.utilizationBatch)-SUM(<<|DISK|>>.utilizationTerm)-SUM(<<|DISK|>>.utilizationTrans)-SUM(<<|DISK|>>.utilizationBatch)) is greater than 10
Then unbalanceDisk
is confirmed.
And Execute "comment"(@ATOMID=<|DISK|>.name,<<|DISK|>>.name;@STRING="have unbalance load, balance the load.";)