A Capacity Planning Study of Database Management Systems with OLAP Workloads

By

XILIN CUI

A thesis submitted to the
School of Computing
in conformity with the requirements for the
Degree of the Master of Science

Queen's University
Kingston, Ontario, Canada
October, 2003

Abstract

Computer capacity planning is the process of monitoring and projecting computer workload and specifying the most cost-effective computing environment that meets the current and future demands for computer services. The growing complexity of database systems stimulates the development of software tools to probe and predict system performance and to help in system design.

The main contributions of this thesis are a study of DBMS tuning factors influencing its performance under OLAP workload, the design and validation of a queueing network model to capture the main features of the DBMS behavior, and the use of a quantitative approach to project the DBMS performance with OLAP workload.

We built a workload model for OLAP, which is based on the TPC-H benchmark, to investigate the impact of DBMS tuning factors (such as buffer pool, sort heap, prefetching and number of I/O servers) on OLAP performance. We found there are certain thresholds on buffer pool and sort heap size in a DBMS with OLAP workload. The relationships between these thresholds, database size and hardware configurations were further investigated.

We propose a queueing network model (QNM) to represent the DBMS under study and give the results of preliminary experiments to validate this model. We have indicated, through discussion and example, how to modify the parameters of this model

to represent various common changes to the hardware (CPU and disk upgrade) and workload (number of users and database size). Amdahl's law is used to estimate the effect of multiple disks on DBMS performance. A mathematical model is proposed and used to predict the disk demands for different disks.

Keywords: Capacity Planning, Database Management System (DBMS), Queueing Network Model (QNM), Online Analytical Processing (OLAP).

Acknowledgements

I would like to take this opportunity to express my sincere thanks to my supervisor Dr. Patrick Martin for his guidance, advice and support during my thesis research and graduate study at Queen's University. Pat has been an excellent advisor, and I am immensely grateful for the help he has given me. His talents in looking at the big picture and asking high-level clarifying questions, combined with the depth of his technical knowledge have provided me with incredible amounts of insightful feedback on my research.

In the last two years, I've also been very fortunate to work with Wendy Powley. Wendy has been an astute technical advisor and a supportive mentor. She provided counsel in many areas, including experiments set up, lab technical support and thesis writing. Easy-going and fun loving, she communicates her joy in teaching and advising students. I have greatly enjoyed working with her. My thanks also go to the members of database lab, especially Wenhu Tian, who have saved the day on many occasions, and I am very thankful for their invaluable help.

Special thanks are given to the school of computing and information science at Queen's University for providing me the opportunity to pursue graduate studies. Also thank IBM Canada, CITO, NSERC and Ontario government for the financial support.

Finally, I would like to thank my parents, my wife Joy and my son John, for their love, support, understanding, patience, and encouragement through the years of my life. I thank them from the bottom of my heart.

Contents

List of Tables

List of Figures

Chapter 1 Introduction

More emphasis is typically placed on the functionality of a computer system rather than it's performance. In the past, E-commerce companies believed that the most urgent priority they had was to get their business up and running. However, this attitude is changing [49]. Steve Johnson of Andersen Consulting states: "It is quite clear that there's much more attention now on performance…helping companies that are going to survive this dot.com shakeout to really focus on the thing that makes a difference to their business" [37].

There are multiple ways to measure the performance of a system. The most commonly used performance metrics are response time (R) and throughput (X) [21]. The response time is defined as the time interval from the instant a command is input to the system to the instant the corresponding reply begins to appear at the terminal. The throughput is generally considered as a measure of the system's productivity, that is, the number of programs or transactions processed during the measurement period. Essentially, they are the same aspect of the system viewed from different perspectives. Users are interested in the response time while system managers are interested in throughput. Given infinite resources, the expected quality of service can always be

provided, but with limited resources, capacity planning and resource management is needed.

Computer capacity planning is the process of monitoring and projecting computer workload and specifying the most cost-effective computing environment that meets the current and future demands for computer services [23]. Much of the work on capacity planning was done in the 1970's and the 1980's, while CPU speeds were relatively slow [23, 38, 13, 16]. Since then, research has been sparse in this area. The reasons for this lack of interest could be the relatively low cost of PC's and their building components, and the PC's upgrade speed is faster than the demands of applications.

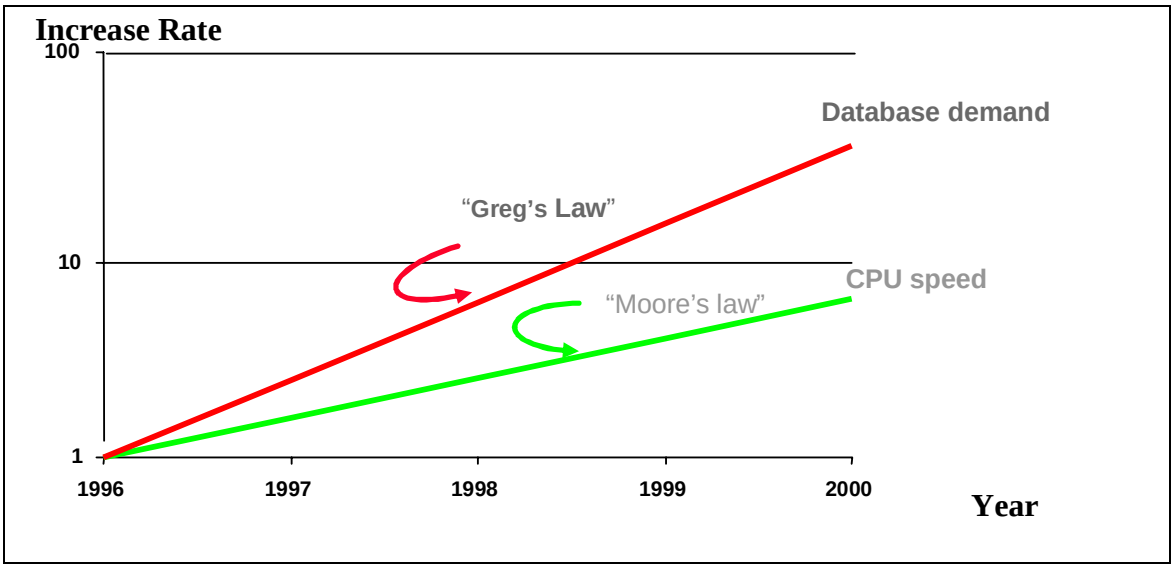


Figure 1-1. The Gap Between Database Demand and CPU Update Speed

Since 1996, with the development of the World Wide Web (WWW), and the emergence of very large multimedia databases, database demand has doubled every 9-12 months according to Greg's law [34]. At the same time processor speed has doubled every 18 months based on Moore's law, and disk speed increase rate is very minimal. As

shown in Figure 1-1, the gap between database demand and CPU speed is increasing with time, and the difference with disk is even greater, which indicates a need for capacity planning.

1.1 Motivation

According to Dataquest [43], commercial server applications such as database services, file services, media and email services are the dominant applications run on the server machines. Database applications occupied 32% of the server volume in 1995 and the share is increasing. Corporate profits and organizational efficiency are becoming increasingly dependent upon their database server systems, and those systems are becoming so complex and diverse that they are difficult to control and maintain. To monitor and predict their performance is a challenge for the database administrators (DBAs).

Database servers pose both hardware and software configuration challenges. Large hardware systems present numerous hardware configuration parameters such as the number and speed of disks, I/O controllers, the disk I/O bus, and the processor I/O bus. In addition, various policies, such as the amount of caching provided by the I/O controllers must be specified.

Database management systems (DBMSs) are themselves complicated software systems with numerous configuration “knobs”. Popular commercial DBMSs provide on the order of 200 initialization parameters to control runtime management issues such as

the buffer pool size and management strategy, the degree of multithreading/processing, logging, disk read-ahead (prefetching), and specific memory management alternatives. Although default values are often provided for these configuration parameters, they do not necessarily match the requirements of the intended workloads.

The growing complexity of database systems stimulates the development of software tools to help in system design and tools to predict and probe system performance. The goal of capacity planning is to predict the most effective computing environment to meet future demands given a small number of parameters. Of specific interest to this thesis is DBMS design and planning. In particular, we note the following:

- During the early stages of database development, DBAs need to gain insight into the desired performance of the system in order to make reasonable choices concerning the speed and number of processors, the number of disks required to store the database, and the distribution of the database tables over these disks. It is more cost effective to spend the time in design, and to correct inefficiencies before an application reaches production than to be forced into a costly upgrade because the database server cannot provide adequate response to the end user community.

- At run time, the DBA must make modifications to the DBMS configuration and tune the system. For example, the DBA may decide to increase the buffer pool size in order to decrease the response time of the system. Accurate sizing of the buffer pool requires that the relationship between buffer pool size and projected response time be quantified.

- Like the saying ‘a chain is only as strong as its weakest link’, database performance is often set by its weakest component. The process of system tuning is really that of finding the weakest component and making it stronger. The DBA needs a tool to help determine the bottleneck in the DBMS and to suggest a solution to remove the bottleneck.

- The workload increases when companies expand their business and allow more customers to access and use their systems. When a DBMS reaches saturation its performance degrades. There is a need to predict the impact of expansion on the performance of the system in order to take the right action to keep the performance levels intact.

- When a system upgrade becomes necessary, a DBA usually has more than one option. The performance of each alternate configuration must be evaluated in order to choose the best price/performance ratio.

The database community widely recognizes two major types of commercial database workloads: *online transaction processing (OLTP)* and *online analytical processing (OLAP),* also called *decision support systems (DSS)*. OLTP systems, such as airline reservation systems, handle the *operational* aspects of day-to-day business transactions. OLAP systems provide *historical* support for forming business decisions. Microsoft Research’s Dr. Philip Bernstein [4] estimates that decision support systems (DSS) account for about 35% of the demand on a database server, a percentage that is increasing over time. We focus on OLAP capacity planning in our research.

1.2 Objective of the Research

The main objective of this thesis is to build an analytical model for database management systems that can be used to predict the performance of these systems in a scientific way. A second main objective is to lay the foundation for a software tool that allows users to examine their design decisions and estimate the performance of the resulting system with a certain hardware configuration. Although our experiments use DB2 Universal Database (DB2/UDB) [18], the techniques that we use are general and can be applied to any other DBMSs. This work is a continuation and extension of previous work on capacity planning for OLTP workloads [56]. In this thesis we specifically focus on capacity planning for OLAP workloads.

The secondary objectives of this research are the following:

- To investigate the impact of database design and hardware configuration factors on OLAP performance, and to quantify these impacts.
- To find the relationships between important tuning parameters such as bufferpool size, sort heap size and OLAP performance. To quantify the relationship between database size and OLAP response time.
- To characterize and generalize OLAP transactions, and to develop a workload model for an OLAP system.
- To build an analytical model to represent the DBMS system under study, validate this model and use it to predict OLAP performance for different user populations and hardware configurations.

1.3 Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 outlines the general approaches and procedures of capacity planning using an analytical model approach and reviews the previous work in this area. In Chapter 3, we discuss the impact of factors, such as buffer pool, sort heap, prefetching, and number of I/O servers on OLAP performance, followed by the development of an OLAP workload model. In Chapter 4 we propose a queuing network model to represent the DBMS under study and provide a detailed description of a set of experiments to validate this model. Chapter 5 gives a concrete example to show the functionality and scope of this model. We summarize the thesis and suggest some future work in Chapter 6.

Chapter 2 Background and Related Work

Database management systems (DBMSs) are key components of most large organizations' information technology infrastructure and often require significant amounts of both computer and human resources. It is therefore reasonable for an organization to try and provide the most cost-effective configuration for its DBMS. The problem of determining this configuration is made more difficult by the fact that a DBMS workload, and hence its demand for resources, may change significantly over time. Capacity planning estimates the amount of hardware resources, such as CPU, memory, and disk, needed to satisfy a set of performance requirements under a changing workload for a certain application.

In this chapter, we first introduce the general approaches to capacity planning, and outline the advantages and disadvantages of each approach. We then describe the analytical model approach in more detail. In Section 2.2, we review the related work on DBMS capacity planning.

2.1 General Approaches to Capacity Planning

2.1.1 General Approaches

In general, there are four approaches to capacity planning: intuitive, experimental, simulation and analytical. The intuitive approach is based on the planner's personal perceptions of the specific circumstances, and not on a clear scientific or mathematical basis. This approach may produce excellent results if the company is fortunate enough to have gifted and experienced planners or if the system's configuration is simple and small. However, this kind of intuitive planning implies a high risk when the computer system is complex.

The historic approach is a modification of the intuitive approach in which the planner relies on a history of system performance measurements to predict future workloads and performance. Although this approach is more scientific than the intuitive one, it is still subjective and is prone to failure in the case where a workload reaches its threshold and causes a performance revolution.

Experimental approaches can lead to a good understanding of what settings are required for good system performance. However, the experimental setup must be carefully chosen to avoid inconsistent results to obtain decisive conclusions. Also altering the system to try out different configurations would be in many cases prohibitively costly.

Another approach to capacity planning is to use simulation. A simulation model is a program written to represent the dynamic behavior of a system. Simulation models have the ability to model computer systems at any desired level of detail and are therefore

powerful tools in system analysis. However, simulation modeling can be relatively costly. It usually involves writing and debugging computer programs, collecting and validating data on a number of parameters, and the running of programs, which can require substantial computational resources. In DBMS capacity planning, simulation is not a suitable choice because the database management system (DBMS) itself is a complex software program. Building a DBMS simulation model is essentially developing another DBMS.

The fourth approach is the analytical modeling approach. The analytical approach consists of building a high level mathematical model to represent the system resources. This kind of model is usually built using queueing network models (QNM) [23]. Queueing network models are sets of mathematical formulae that are devised to capture the relationships between workload and performance measures. (Refer to Appendix A for more details on queueing network models). With specific parameter values, it is possible to evaluate performance measures of interest by solving some simple equations. This model can easily be developed since its parameters are based on directly measurable quantities. The resulting model is flexible because future changes to the system can be reflected by adjusting the parameters. Theoretically, an analytical model can be constructed at any desired level of detail. However, when the model is large, computation becomes prohibitively complex and expensive. To make an analytic model solvable, it may be necessary to make questionable simplifying assumptions, thus placing uncertainty on the significance of the results obtained.

The analytical model is easy to develop, not too complex to use and update, and gives fairly good estimates and results. It is an approach suitable for DBMS capacity

planning. In this thesis, we use the analytical modeling approach as a basis for a tool for capacity planning of database management systems. The analytical modeling approach represents a good intermediate solution. Although it is less accurate than actual experimentation and simulation approaches, analytical modeling is much simpler than simulation, more reliable than intuition, and more flexible than experimentation.

2.1.2 Procedures of Capacity Planning Using Analytical Modeling

There are two different strategies for the capacity planning process using an analytical model, namely a component approach and a system modeling approach. With the component approach the various resource components are treated independently and utilization and performance of each resource is monitored and projected. The system modeling approach attempts to extract the essential aspects of the system's operational characteristics and represent them in mathematical formulae, computer programs, or both. The system modeling approach is methodical in that it enforces the gathering, organizing, evaluation, and understanding of information from the entire system. Once constructed, the model can provide great insight about the system and can be used to predict system behavior under future scenarios. In our research, we use and focus on the system modeling approach.

Queueing network modeling is a general and widely used system modeling approach. A queueing network model represents a computer system as a network of queues. Each queue is associated with a "service center", which represents a system

resource (e.g. CPU, disk) in demand by clients. The complexity of the model can be controlled through specifying the number and types of service centers, routing characteristics, and the number of workload classes. A capacity planning study based on queueing network modeling usually consists of the following steps [21]:

1. The basis of modeling is a thorough understanding of the system being modeled. Therefore, the first step is to have a clear understanding of the hardware configuration of the system (CPU, disks, memory) and its characteristics such as CPU speed, disk transfer rates, whether queuing is required and what queuing discipline is used (i.e. first in, first out, or priority).
2. Identify workload components and obtain measurement data regarding each workload component. Forecast the rate of growth for each workload component.
3. Construct a queueing network model based on the workload and system description as obtained above.
4. Calculate performance measures from the model, with model parameters defined from measurement data obtained through a monitoring tool. Validate the model by comparing the calculated performance quantities with their measured counterparts. Proceed to the next step only after the model is validated. Otherwise, go back to previous steps and modify the model.
5. Use the validated model to predict the performance of each workload class under the projected future workload, or examine the various alternatives for system changes that can be used to avoid the performance degradation.

This thesis roughly follows the above procedure, Chapter 3 covers steps 1 and 2, Chapter 4 realizes steps 3 and 4, and Chapter 5 corresponds to step 5.

2.2 Literature Review

2.2.1 Capacity Planning Using Queueing Network Modeling

In 1971, Buzen proposed modeling systems using queueing network models and published efficient algorithms for solving some important models [6]. The models are an abstraction of the computer systems they model, so they are easier to create than general purpose simulation models. They can also be used interactively because they are solved analytically. Since then, many advances have been made in modeling computer systems with queuing networks, including faster solution techniques and accurate approximation techniques [23, 38]. Lazowska [23] and Ferrari [13] emphasize the techniques and tools developed in performance evaluation. These tools include hardware and software monitors, simulation, analytical modeling and benchmarking. Lam [21] examines the theory and practices of computer capacity planning in a more rigorous manner than previous studies. Menasce [29] relates the models to the real world using clear and accurate mathematical modeling, and later extends the models to present concepts and formulas for effectively planning web-based systems, and shows how to use quantitative methods to analyze web-based applications [27].

Previous attempts that use queueing network models as analytic tools on database system performance studies include a stochastic model of processor scheduling under IMS [22], studies of concurrency controls and locking for updates [14, 19, 20, 36, 42, 53]

and a hierarchical performance model [31]. Some researchers have proposed performance prediction tools and applied them to specific database design issues [24, 32]. In some research, the kinds of analytic approaches used initially at the physical database design level were applied at the logical database level as well. Oren and Ashim [33] suggest a way of estimating accesses to components of the data model for alternative logical database designs. Teorey and Fry [45] extend an analytic evaluation model to include the step from a requirements specification to a logical database structure.

File organization and access is an active research area that is closely related to database design and management. Several prediction packages based on simulation have been produced for evaluating file organizations [39]. Teorey and Das [44] and Pezzaro [35] suggest some similar packages based on analysis. Teorey and Oberlander [46] specify a model based on a combination of analysis and simulation for evaluating a range of design decisions including buffering and overflow strategies. The analytic model of Gambino and Gerristen supports the optimization of several detailed representations of design decisions [15]. Cardenas [7] and Sagamang [8] introduced analytical models for selection among alternative file structures. They specify the set of parameters needed to describe the file or database content, the access and modification activity, and the storage device characteristics. Specific file structures are dealt with, and factors such as index selection, block sizes, data encoding, and mapping to storage devices are taken into account. Allen [1] suggests an analytic model for evaluating general file organizations involving accesses to directories and data. Other papers have covered the evaluation of database organizations and selection of access paths [25, 30, 47, 51, 52].

DBMS vendors have developed capacity planning tools that allow users to estimate the performance of their products. One of these tools is the DB2/UDB Estimator developed by IBM [12]. The DB2/UDB Estimator provides the user with an estimate of the performance and cost of running DB2/UDB for OS/390. The DB2/UDB Estimator forecasts the cost of future applications or modifications to existing applications, and identifies the cost associated with different hardware and software design approaches. This tool can help designers and database system administrators in different cases and at different phases of the design and administration life cycles. However, the tool's orientation towards DB2/UDB limits its scope. The assumptions and simplifications on SQL statements, table definitions, and transactions of the database in DB2/UDB Estimator can also lead to imprecise estimation of performance.

There are many different approaches to capacity planning in the database performance research. Most approaches fit into one of the following three categories [40]:

- **Evaluation models**: Given a specific computer configuration, estimate the performance of the application using this configuration.
- **Selection models**: From a set of configuration design choices, identify the design with the best performance according to certain criteria.
- **Optimization models**: From a set of configuration design choices characterized by a set of parameters, chose the set of parameters that optimizes performance.

Essentially these three types of models are closely related. Selection models can be built from an evaluation model simply by trying out each design option. Similarly, optimization models are selection models applied to a large set of design alternatives.

Experience has shown that queueing models can predict the response time of a system to within thirty to fifty percent accuracy [40]. The computation of the performance indices for conventional systems requires a few seconds of processing time. For very large systems, with thousands of customers and varied workload components, an exact solution of the queueing network model is not feasible, and one must settle for bounds on performance [55] or approximate solutions [54]. In order to handle the high computational complexity, many packages for solving queueing network models were developed [9].

Another point we can draw from the literature is that the more information that is supplied, the greater the accuracy of the model. Also, the more information provided, the more difficult it is to build the model, and the model will be less adaptable. There is a trade off between accuracy, cost and adaptability in capacity planning. The challenge is to come up with a scheme that is rich enough to be useful, and yet simple enough to be manageable.

2.2.2 Database Workload Benchmarks: OLTP vs. OLAP

Before modeling DBMS performance, we must first understand the intended workload. There are two major types of database workloads: *online transaction processing (OLTP)* and *online analytical processing (OLAP)*. The two database

workloads have different characteristics. OLTP uses short, moderately complex queries that read and/or modify a relatively small portion of the overall database, which translates into small random disk accesses. OLTP workloads typically have a high degree of multiprogramming due to the large number of concurrent users. In contrast, OLAP queries are typically long running, moderate to very complex queries, which often scan large portions of the database in a read-mostly fashion. This access pattern translates into large sequential disk accesses. Updates are propagated either through periodic batch runs or through background “trickle” update streams. The multiprogramming level in OLAP systems is typically much lower than that of OLTP systems.

The biggest challenge for database system capacity planning is that both OLTP and OLAP workloads are quite complex. In order to simplify and standardize this issue, the Transaction Processing Performance Council (TPC) [48] defines several industry standard database benchmarks. Two common benchmarks, TPC-C and TPC-D model OLTP and OLAP workloads respectively.

In April 1999, two new benchmarks, TPC-R and TPC-H, replaced TPC-D as the industry's standard benchmarks for decision support applications. Unable to integrate two workloads typical for decision support systems, the TPC constituted two separate benchmarks, TPC-R for a *reporting workload* and TPC-H for an *ad-hoc querying workload*. An ad-hoc querying workload simulates an environment in which users connect to the database system and issue individual queries that are not known in advance.

The workloads in TPC-H and TPC-R consist of the execution of 22 read-only queries in both single and multi-user mode and two refresh functions. These queries are

formalized from real world business questions. They simulate generated ad-hoc queries (TPC-H) and reporting queries (TPC-R), and generate intensive disk and CPU activity on the database server.

We use the TPC-H benchmark as OLAP workload in our research. Details of the TPC-H benchmark can be found in Appendix B.

2.2.3 Previous Work

Previous research in our group is focused on capacity planning for an OLTP workload. A closed queueing network model (QNM) with one CPU service center, one disk service center, a memory service center and a number of users was proposed by Zawawy [56]. This QNM models the behavior of the OLTP transactions and determines their relationships with the DBMS resources. The OLTP workload is partitioned into five classes based on the five different types of transactions in TPC-C.

Zawawy studied the service demands of CPU, disk and memory under various population sizes and buffer pool sizes and concluded the following:

- The disk demand is not affected by the variation of the population size if the queuing time is not included. The disk demand depends on the number of disk accesses, which in turn greatly depends on the buffer pool size. An equation was derived to represent the relationship between disk service demand and buffer pool size based on Belady's virtual memory study [3].
- The CPU demand per transaction is independent of the number of transactions concurrently in the system, i.e. the population size, because by definition the CPU

demand does not include the queuing time for the CPU. There is a threshold for buffer pool size for the CPU demand beyond which, all data are in the buffer pool and there is no need for disk access until the transaction is committed. The CPU demand is independent of the buffer pool size. In the case of multiple CPUs, the CPU demand was equally divided between the CPUs.

- The memory service demand has no effect on the system performance if it satisfies the minimum requirements of the operating system and DB2/UDB processes and buffer pools.

A set of experiments was run to validate and use the model. The estimated results produced by the model were mostly in the 10 percent error range compared to the actual detected values, with the exception being when the population size is small and the system is under-utilized.

This QNM model is not suitable for an OLAP workload. First, there are 22 queries in TPC-H, the benchmark for OLAP workload. We cannot map all 22 queries one-to-one into the QNM workload as Zawawy did with TPC-C benchmark (there are only five transaction classes in TPC-C), because the computational complexity of a QNM increased exponentially with the number of workload classes. We have to characterize and group 22 queries into a few general transactions. Second, as we discussed in the previous section, TPC-C transactions are relatively short and access the disk randomly, so the disk demand is greatly related to the buffer pool. An equation could be derived between disk demand and buffer pool size in a QNM model of OLTP. The TPC-H queries are quite complex and sequentially scan large amounts of data. So the buffer pool behaves differently and the equation derived for TPC-C is not suitable. Even more, the

TPC-H queries use extensive sorting functions, so the model should account for the use of sort memory. Third, the typical number of clients running simultaneously in an OLAP workload is far smaller than for an OLTP workload.

Chapter 3 Factors Affecting OLAP Performance and Workload Model Development

Generally, the critical element in the creation of performance prediction models for database systems is the representation of the accesses to, and manipulations of, the database contents. A meaningful representation requires a thorough understanding of the system and user behavior. The first step in conducting capacity planning is to have a clear understanding of the configurations of the systems under study, and a familiarity with the typical workload system. In this chapter, we first introduce the computing environments used for our research. In Section 3.2, we discuss the factors in a DBMS that affect OLAP workload performance. Section 3.3 develops and validates an OLAP workload model, based on TPC-H benchmark, to be used by the queuing network model presented in Chapter 4.

3.1 System Configuration

We chose to run our experiments on three different computer systems to show the independence of our approach from the hardware configuration and the adaptability of our model. Since some of our experiments need to be run on systems with multiple CPUs and/or multiple disks (to illustrate different processor and disk configurations), and we have chosen computer systems that can provide us with such a capability.

The first system used in our experiments is an IBM Netfinity 5000, with dual PIII 400MHz processors, a 512KB cache, 1 GB ECC SDRAM memory, an Open Bay hard disk drive (four IBM-PCCO DGHS09Y UW (ultra wide) SCSI SCA disks, and one IBM-PSG DNES-309170Y Ultra Wide SCSI disk), Dual Channel Wide Ultra SCSI and PCI/ISA controllers.

The second computer system is an IBM-Xseries 240 with dual 1 GHZ CPUs, four PCI/ISA controllers, and 24 Seagate ST 318436LC SCSI disks. With two processors on its motherboard, we choose upon startup whether the system runs with either a single CPU or dual CPUs, allowing us to run our multiple processor experiments. By distributing the database tables over multiple disks, we can choose how many disk controllers and disk drives are used.

The third computer is an IBM PowerServer 704 with a single 200 MHZ CPU, 1 GB of RAM, and 16 Seagate ST 34520W SCSI hard disk drives.

Throughout the discussion of our experiments, we refer to the first system as Jaguar, the second system as Baserver, and the third as Cougar. The operating system on all machines is Windows NT Server version 4.0. Jaguar and Baserver run DB2/UDB

V8.1, and Cougar runs DB2/UDB V7.3. We assume the difference in DBMS versions does not affect our study.

3.2 DBMS Tuning for OLAP Workload

Database server performance depends on two things: hardware environment and software configuration. Hardware environment includes CPU, RAM, and disk subsystem speed. When the database is small, that is, up to a hundred thousand rows per table, the performance is mainly dependent on the CPU speed since most of the data can be kept in main memory. As the database grows larger, less and less of the data will fit in memory. Most databases are too large to fit in RAM. So caches, caching strategies, and caching options are important to DBMS performance. Eventually, the database becomes many times larger than the size of available RAM. At that point, performance becomes most dependent on disk speed. That is, how fast can the server get to the data. So often, rather than upgrading a machine, the RAM capacity and/or the disk subsystem are upgraded. The DBA may spend time giving the database server hints on how to cache the data, so it can use the available RAM more effectively. This is another research area, software configuration and DBMS tuning.

With the rapid deployment of new computer technologies over the last few years, DBMSs are used for OLTP, OLAP, data warehousing, e-commerce and multimedia applications [10, 11]. These applications have a wide range of resource demands and haphazard resource allocations may lead to severe resource contention or severe performance degradation. The default DBMS configuration settings are often not

appropriate and are usually not optimal. A well-tuned configuration for one workload is unsuitable for another type of workload. For example, a change from an OLTP to an OLAP workload requires an adjustment of more than 15 parameters in DB2/UDB [18]. Self-tuning and autonomic computing, which shifts some or all tuning responsibilities from the DBA to the DBMS itself, is the direction of next generation DBMSs [5, 26].

In this section, we cover the impact of software configuration and some important DBMS tuning parameters on OLAP performance such as:

- Buffer pool, the area of memory into which database pages are temporarily read and modified.
- Sort heap, the memory pages to be used for sorts.
- Prefetching, which reads the data needed by a query before it is referenced by the query, so that the query does not have to wait for I/O to be performed.
- I/O server, which is used on behalf of the database agents to perform prefetch I/O and asynchronous I/O.

The impact of the hardware environment (disk and CPU) on OLAP performance and capacity planning are investigated in Chapter 4 when the queueing network model is developed and validated.

3.2.1 Buffer Pool

The purpose of the buffer pool is to reduce the number of disk accesses to improve database system performance, since data can be accessed much faster from

memory than from a disk. During buffer pool set up and configuration, the general rule of thumb is to keep objects with different access patterns separate. That is, objects with a high re-reference access rate should be kept separate from those with a high sequential access rate.

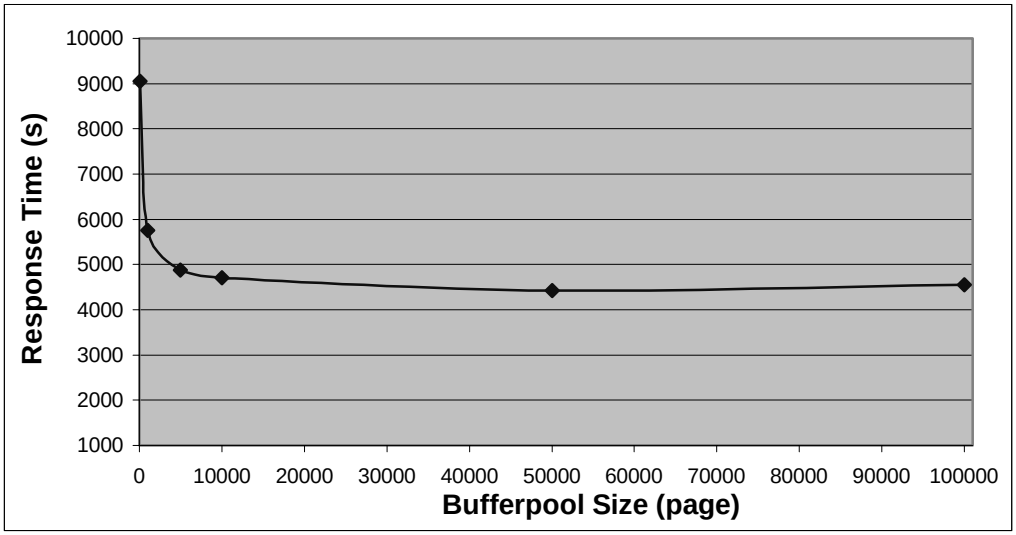


Figure 3-1 Effect of Buffer Pool Size on TPC-H Query Response Time (Cougar, Database size 1GB, Sort heap size 1000 pages)

We measured the response time of the 22 queries in TPC-H using different buffer pool sizes with different configurations. The experiments are designed and run with all the other parameters fixed except the one being investigated, thus isolating it's effect. Figure 3-1 shows the sum of the 22 TPC-H queries' response times versus buffer pool size. We see that the response time decreases as the buffer pool is enlarged up to a certain size. Beyond this size, the response time is a constant, which means there is a threshold for the buffer pool size. The performance of TPC-H is independent of buffer pool size beyond this threshold.

The performance for an OLTP workload, on the other hand, is greatly dependent on the buffer pool size. Typically, the response time decreases with an increase in buffer pool size. The main reason behind this different behavior is the difference in the data access patterns in OLTP and OLAP workloads, namely small random disk accesses in OLTP and large sequential disk accesses in OLAP. During the large sequential disk accesses, data is read once and then can be swapped out of the buffer pool, which requires only a small buffer pool. The number of disk accesses is independent of the buffer pool size in this case. For a random data access pattern, the buffer pool can cache the most frequently used data to reduce the number of disk access. The larger the buffer pool, the larger its cache ability, which results in fewer disk accesses. In TPC-H, the database indexes are heavily used and accessed randomly. Once these indices can be held in memory, the performance of TPC-H is significantly improved.

Table 3-1 Buffer Pool Snap Shot for TPC-H in Different User Populations (Baserver, Database size 1G)

Number of Clients	3	4	5	6	9	15
Data Logical reads	10444219	14627124	18425431	21535474	23765047	27482429
Data Physical reads	8877247	11806594	14996350	17362409	26032733	36971918
Data Hit rate (%)	15.0	19.3	18.6	19.4	-9.5	-34.5
Index Logical reads	5352666	8127561	10017386	11785633	10005468	12171868
Index Physical reads	248038	320718	427068	471324	636513	747034
Index Hit rate (%)	95.4	96.1	95.7	96.0	93.6	93.9

In order to verify our theory, we monitor the buffer pool usage using the database snapshot monitor. Table 3-1 lists the buffer pool logical reads (the overall reads), and physical reads (the actual read from disks) for data and index, and the calculated hit rate

((logical read –physical read)/logical read) for different user populations (that is, a varied number of clients).

From Table 3-1 we can see that the hit rates for index pages are very high (>93%), while the hit rates for data are pretty low (<20%). The data hit rates even become negative when the client number exceeds 9, which means the actual number of reads from disk (Physical reads) is larger than the transaction needs (Logical reads). This is due to prefetching, which we will discuss in Section 3.2.3. When the prefetching is turned off, the data hit rate of 15 clients increased to 10%, but the performance degrades. The overall response time of the 22 TPC-H queries increases from 125 minutes to 186 minutes.

We have determined that there is a certain threshold on buffer pool size for a specific database system with TPC-H workload. To further this investigation, we need to find the exact value and impact factors of this threshold. We designed three sets of experiments to study the buffer pool effect on TPC-H performance with different DBMS parameters (sort heap), hardware configurations (CPU speed and number, disk number) and database sizes.

3.2.1.1 Sort Heap Effect

A database management system is a complex software system, so changing one parameter may impact another parameter. Sort heap is an important DBMS parameter for OLAP workload because of the number of large sorts in this workload. The impact between buffer pool and sort heap was investigated and results showed that they are

independent of each other. From Figure 3-2 we can see that the buffer pool thresholds for different sort heap sizes (1000, 10000, 100000 pages) are the same.

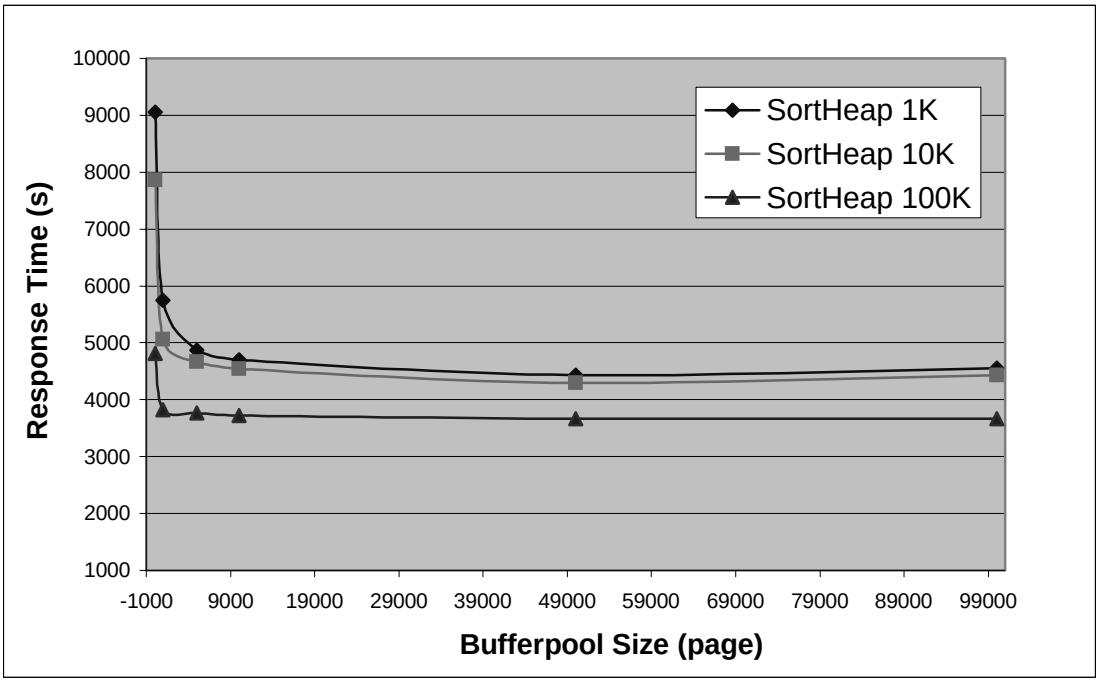


Figure 3-2 Buffer Pool Size on TPC-H Query Response Time as a Function of Sort Heap Size (Cougar, Database size 1GB)

3.2.1.2 Hardware Effect

We also studied the impact of hardware configuration on buffer pool threshold. Figure 3-3 shows the results of different CPU speeds (Cougar 200MHZ, Jaguar 400MHZ, Baserver 1GHZ), and Figure 3-4 shows the results of multiple CPUs. The effect of multiple disks, where the database is distributed among 3, 4, and 12 disks, respectively, is given in Figure 3-5. From the figures, we can see that TPC-H performance varies between the hardware configurations, but the buffer pool threshold is the same for

different configurations. We conclude that this threshold is independent of system hardware configurations.

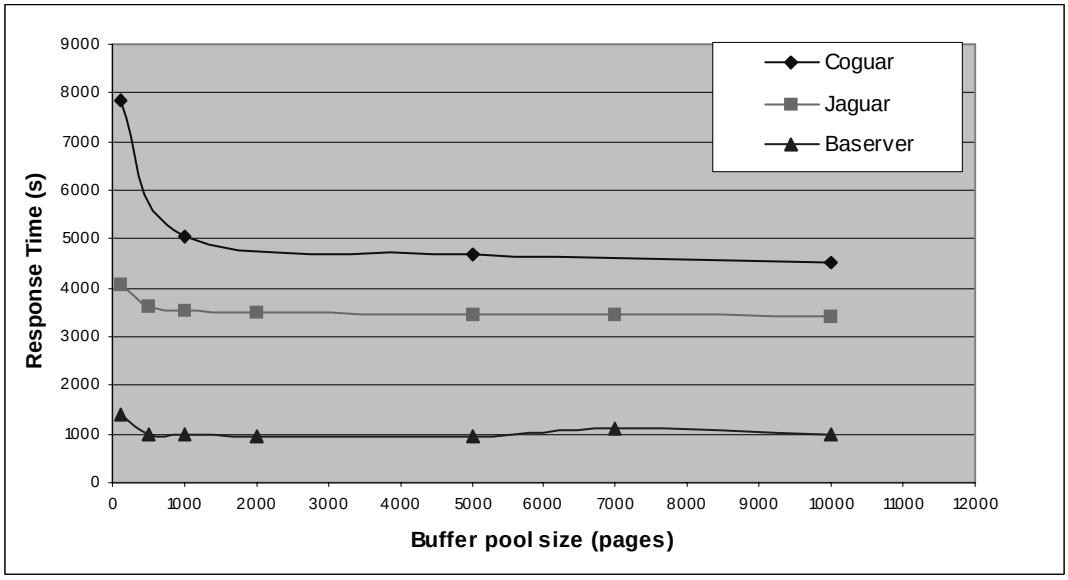


Figure 3-3 Effect of Buffer Pool Size on TPC-H Query Response Time as a Function of CPU speed (Database size 1GB)

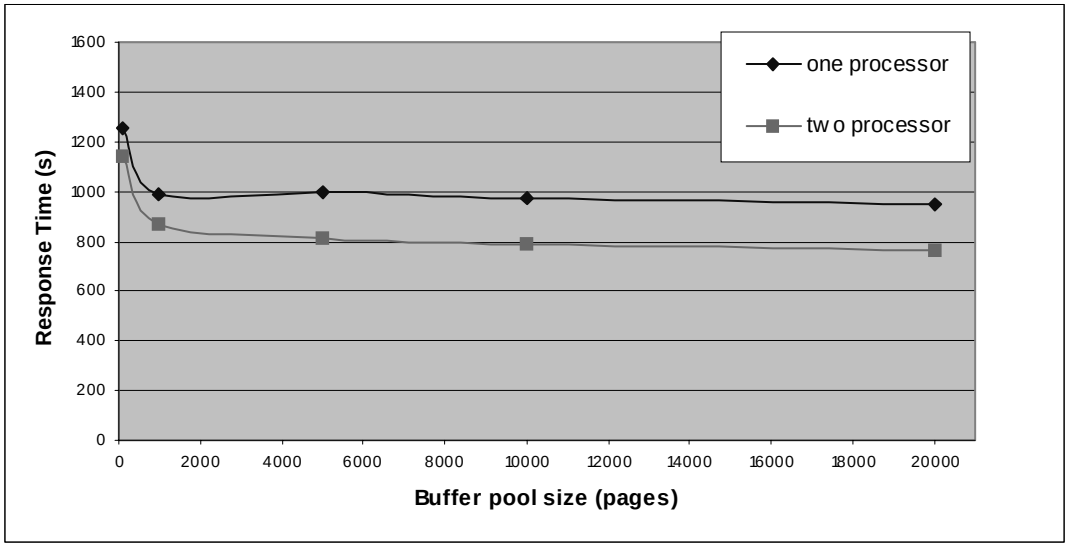


Figure 3-4 Effect of Buffer Pool Size on TPC-H Query Response Time with Multiple CPUs (Database size 1GB, Baserver)

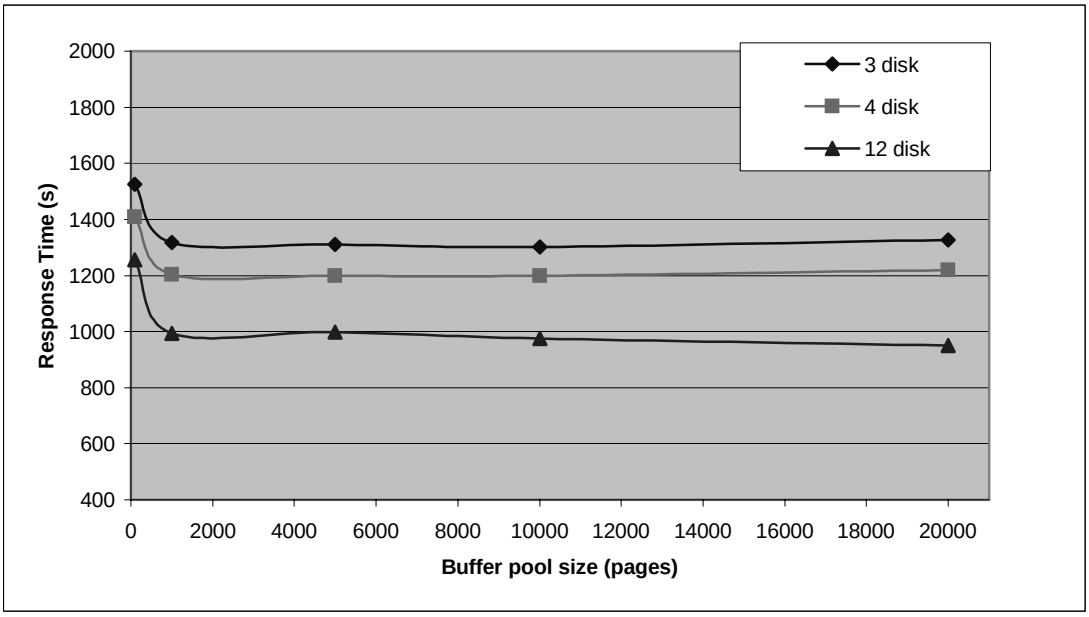


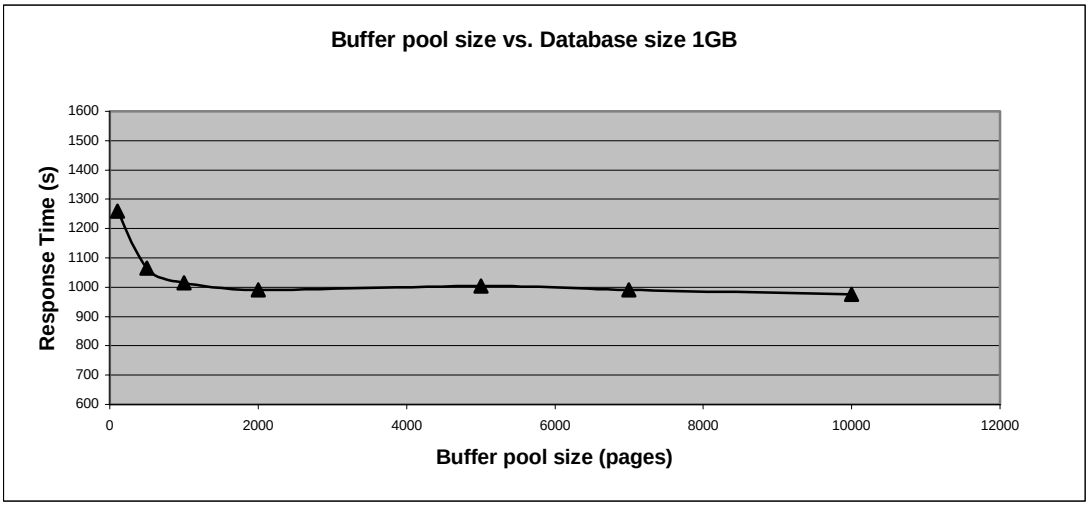
Figure 3-5 Effect of Buffer Pool Size on TPC-H Query Response Time with Multiple Disks (Database size 1GB, Baserver)

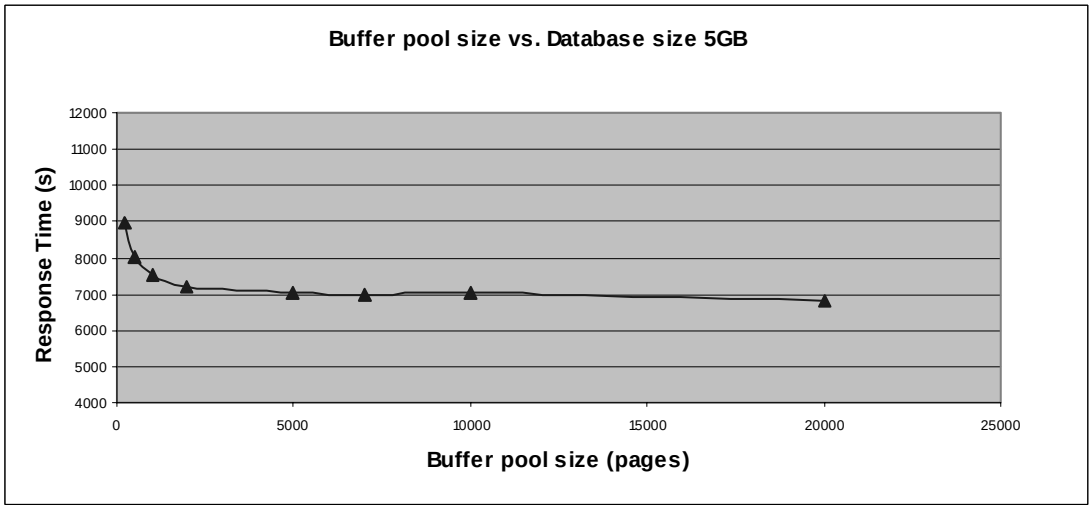
3.2.1.3 Database Size Effect

We hypothesize that the buffer pool threshold is dependent upon the database size and especially the index size. We looked at the threshold values for three database sizes, 1GB, 5GB and 10GB, and the results are shown in Figure 3-6. The results verify our hypothesis and indicate that the buffer pool threshold increases with database size.

3.2.1.4 Summary

In summary, there is a threshold for the buffer pool size for TPC-H workloads. After this threshold, the performance is independent of the size of buffer pool. This threshold is not correlated with sort heap parameter in DBMS, nor the hardware configuration. It is closely related to, and increases along with, the database size.





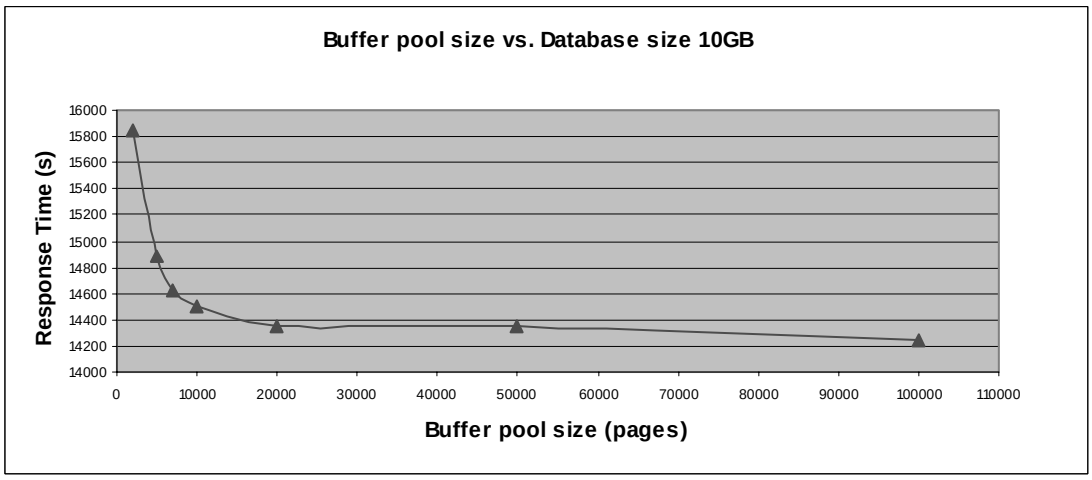


Figure 3-6 Buffer Pool Size on TPC-H Query Response Time as a Function of Database Size (Baserver)

In all subsequent experiments, we use a buffer pool size 5,000 pages for the 1GB database, 7,000 pages for the 5GB database and 20,000 pages for the 10 GB, which exceed their buffer pool thresholds. This eliminates any impact by the buffer pool and simplifies the analyses.

3.2.2 Sort Heap

OLAP queries are typically long running, moderate to very complex queries with frequent multi-way joins. Among the 22 queries of the TPC-H benchmark, 18 queries have an “order by” clause and 15 queries have a “group by” clause. The sort heap memory, which is used to sort, join, and calculate, should play an important role in DBMS with TPC-H workloads.

Figure 3-7 shows the graph of sort heap size vs. response time for the 22 TPC-H queries. From this figure we can see that, similar to buffer pool size, there is a threshold for sort heap size. After this threshold, the query response time is independent of the size of the sort heap. The value of the threshold is related to the workload. Each query uses a different amount of data, and has different requirements for sorting and grouping. When the sort heap size is small, some of the temporary results must be written to disk. The sort heap threshold is the size of sort heap that satisfies the required memory for that query’s data management, thus reducing the number of disk accesses and decreasing the query’s response time. We plot the sum of response time of all 22 queries, the sum of response times of all queries except query 9, and the response time of just query 9 in Figure 3-7. Query 9 plays a dominant role with respect to sort heap among the 22 queries. If we look

at query 9 more closely in the figure (the small icon in Figure 3-7), we see that the response time of query 9 dropped from 848.1 sec to 308.2 sec when sort heap size changed from 82040 to 82050 pages. A change of only 10 pages in the sort heap improves its performance by more than 60%.

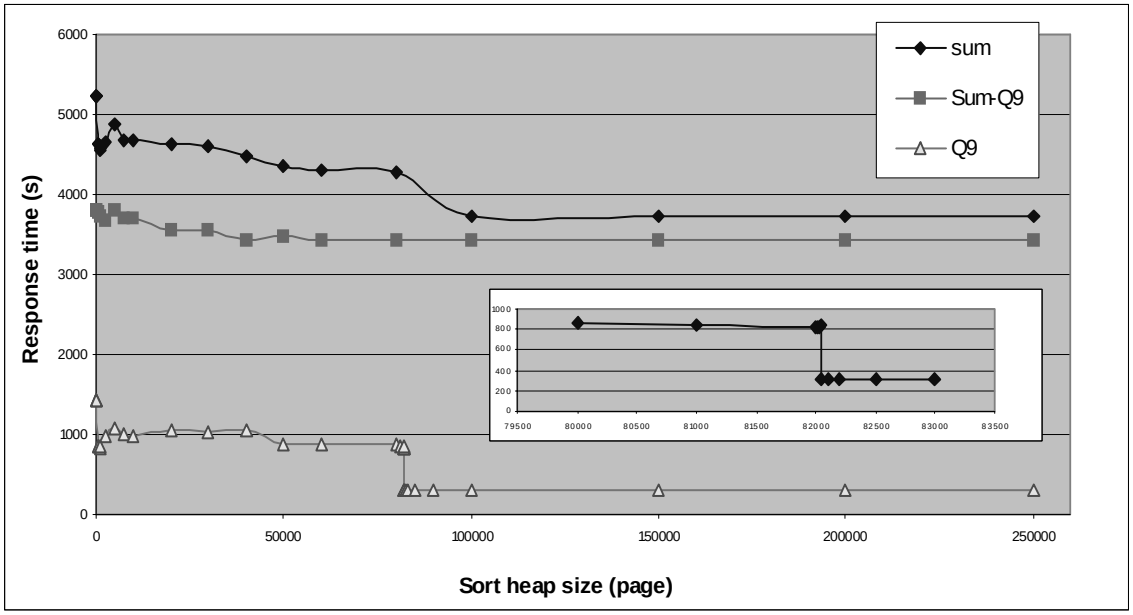


Figure 3-7 Effect of Sort Heap Size on TPC-H Query Response Time (Database size 1GB, Cougar)

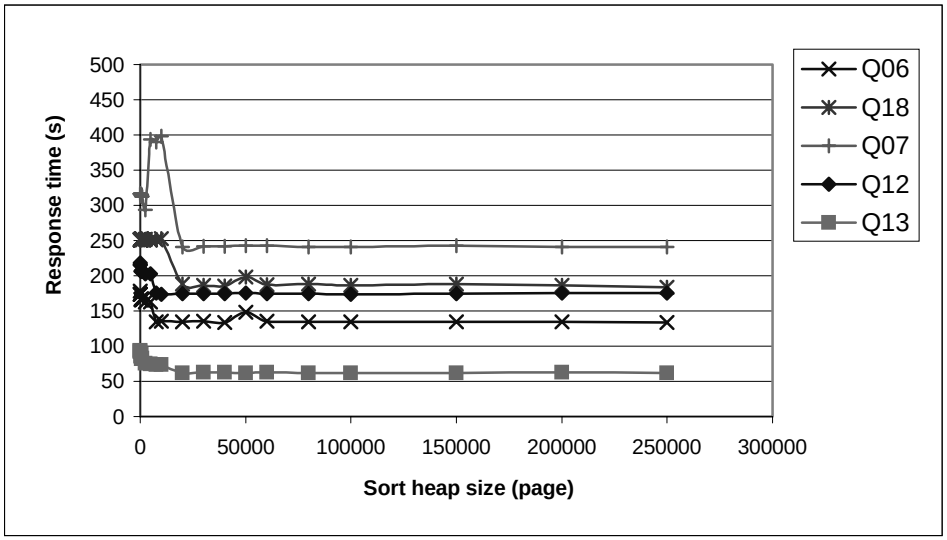


Figure 3-8 Effect of Sort Heap Size on Some TPC-H Query's Response Time (Database size 1GB, Cougar)

Another point that should be noted is that not all queries are sensitive to a change in sort heap size, only queries 6, 7, 9, 12, 13, and 18 are affected by the sort heap size. From Figure 3-8 we can see that the sensitivities are different for the above 6 queries, and each has a different threshold. Among them, query 9 has the largest threshold.

We also investigated the effect of hardware (number of disks, number of CPUs and CPU speed) and database size on the sort heap threshold. Figure 3-9 shows the impact of number of disks, where a 1GB TPC-H database is distributed across 3, 4, 8, and 12 disks, respectively. The response times under each configuration are different, but the shape of the graph and the sort heap threshold are the same. We conclude that the sort heap threshold is independent of the disk configuration.

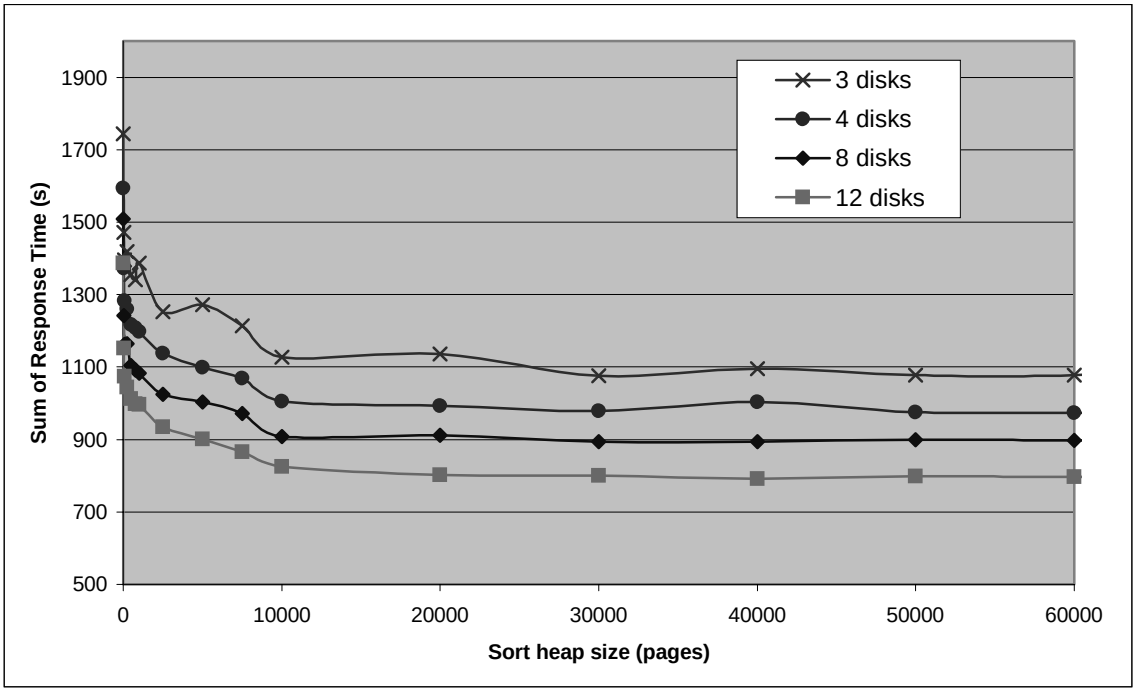


Figure 3-9 Effect of Sort Heap Size on TPC-H Query Response Time as a Function of Disk Distribution (Database size 1GB, Baserver)

We experimented with three systems to verify the impact of CPU configuration on sort heap threshold. Figure 3-10 shows the results of different CPU speeds (Cougar 200MHZ, Jaguar 400MHZ, and Baserver 1GHZ), and Figure 3-11 shows the results of multiple CPUs. From Figure 3-10 we can see that the CPU speed has an impact on the shape of the graph and the sort heap threshold increases as CPU speed decreases (10,000 pages for Baserver, 20,000 pages for Jaguar, and 100,000 pages for Cougar). The slower processor needs more sort heap memory. The number of CPUs does not impact the sort heap threshold, which indicates that there are no parallel sorts in current system (Figure 3-11).

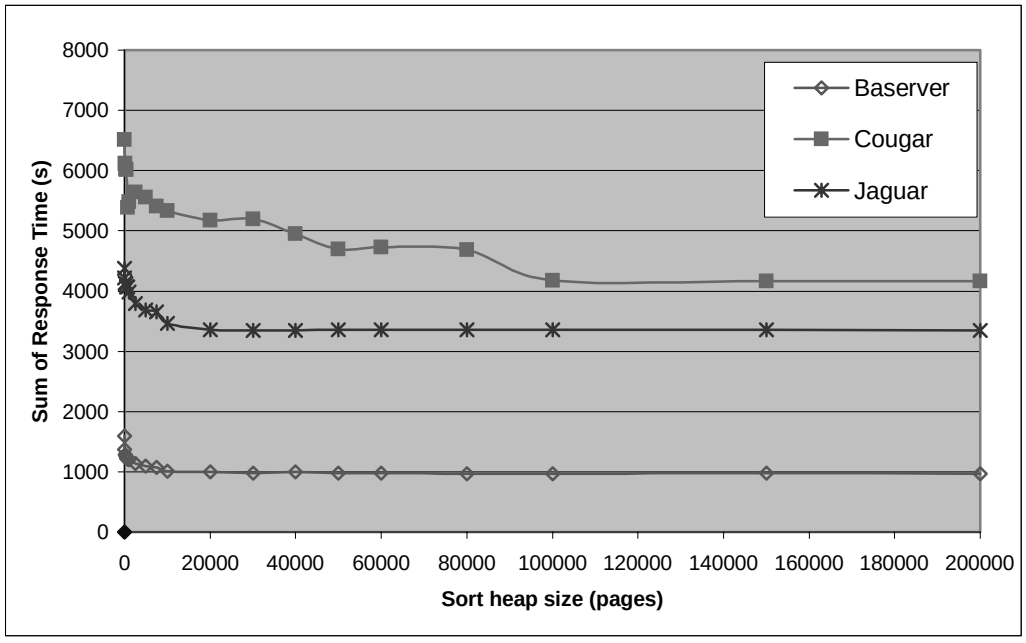


Figure 3-10 Effect of Sort Heap Size on TPC-H Query Response Time as a Function of CPU Speed (Database size 1GB)

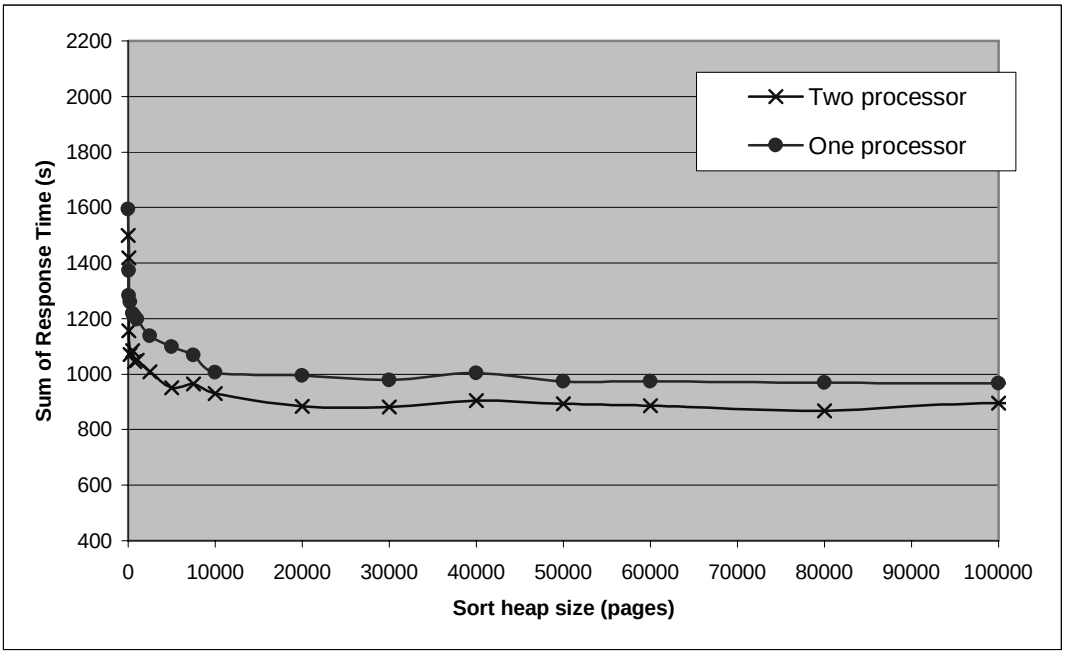


Figure 3-11 Effect of Sort Heap Size on TPC-H Query Response Time as a Function of Number of CPUs (Database size 1GB, Baserver)

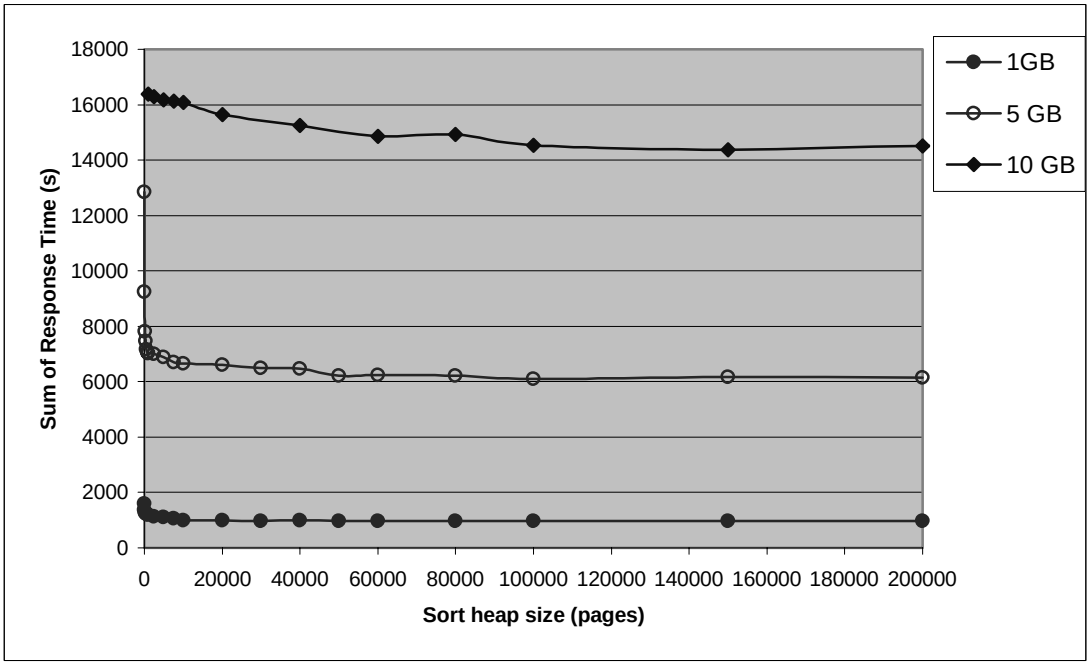


Figure 3-12 Effect of Sort Heap Size on TPC-H Query Response Time as a Function of Database Size (Baserver, One CPU, Four Disks)

The impact of database size on the sort heap threshold is shown in Figure 3-12. As expected, we found that the sort heap threshold increased along with the database size: 10,000 pages for the 1GB database, 100,000 pages for the 5 GB database, and 150,000 pages for the 10 GB database.

In summary, sort heap is another DBMS tuning parameter that impacts performance. We found that there is a threshold for sort heap size. This threshold is dependent on the query content, and different queries have different thresholds. The disk configuration has no effect on the sort heap threshold, while CPU speed plays an important role. We found that a slower processor usually needs more sort heap memory and has a larger sort heap threshold. The required sort heap threshold increases along with the database size.

3.2.3 Prefetching and I/O Servers

OLAP queries typically scan large portions of the database, which translates into large sequential disk accesses. Prefetchers (I/O servers) are designed to deal with this matter. During the experiments on buffer pool hit rate of Section 3.2.1 we found that prefetching decreases the buffer pool hit rate, but reduces the average response time by 30%.

Dennis Shasha and Philippe Bonnet [41] have tested the influence of prefetching on the throughput of DB2/UDB V7.1 with an OLAP workload. They found that the throughput increases as prefetch size increase up to a certain point (about to 64 KB) and

then it is a constant. Sixteen pages (64 KB) is the default value for table prefetching size in DB2/UDB. We use this value in our experiments.

The size of prefetching is predefined when creating the tablespace. We cannot modify prefetching size once the table space is created, but prefetching can be turned on or off via configuration parameters. From Table 3-2 we can see that prefetching decreases the response time by more than 20% in every set of experiments. We use prefetching on as default in our experiments except where we explicitly state otherwise.

Table 3-2 The Effect of Prefetching on TPC-H Response Time (s) as a Function of Buffer Pool Size and Sort Heap Size (Cougar, Database size 1GB)

Buffer pool (page)	1,000			5,000			10,000		
Sort heap (page)	1,000	10,000	100,000	1,000	10,000	100,000	1,000	10,000	100,000
Prefetch off	7779.2	6490.0	5040.2	7070.8	6072.0	4985.2	6718.8	6083.0	4947.8
Prefetch on	5739.8	5095.2	3825.8	4848.8	4666.2	3770.8	4573.8	4679.4	3731.2

Table 3-3 The Effect of I/O Server on TPC-H Response Time (s) (Cougar, Database size 1GB)

	Prefetch Off	Number of I/O servers (Prefetch on)							
		1	2	3	5	10	20	30	100
Response Time	5310.6	4127.1	3734.6	3719.6	3717.5	3723.1	3719.6	3725.0	3722.9

I/O servers are used on behalf of the database agents to perform prefetch I/O and asynchronous I/O by utilities such as backup and restore. The number of I/O servers in DB2/UDB can be from 1 to 255 (the default number of I/O servers in DB2/UDB is 20). We have examined the influence of the number of I/O servers on the performance of TPC-H queries (Table 3-3), and found that one I/O server slowed down the response time, while the response times are relatively constant starting from two I/O servers. This

experiment shows that 2 I/O servers are enough for the current system with one disk. We will use the value (20) for the number of I/O servers in our experiments, to keep all experiments having the same number of I/O servers.

3.3 Workload Characterization and Workload Model Development

Workload characterization is a critical step in the capacity planning process and experience has shown that the dominant source of error lies in this phase. Workload characterization identifies the workload components and obtains measurement data for each workload component. It is a quantitative description of the workload's characteristics [27]. Usually it consists of the following four steps: identification of the basic component of the workload, partitioning the workload, measurement and parameterization.

3.3.1 Workload Identification

The first step in workload characterization is the identification of the basic components, or building blocks, of the workload. Workload is composed of a set of commands that are repeated over and over, each command with a certain frequency. Depending on the level at which the workload is viewed, the choice of the set will vary. For a DBMS, the level of the basic component of the workload could be anywhere from high-level batch jobs that are sets of business transactions, to single business transactions,

to SQL commands, all the way down to CPU instructions. The lower the choice of the basic component level, the larger the number of classes, which complicates the measurement and quantification of the workload, and makes the calculations potentially computationally challenging.

Throughout the study, we use the TPC-H benchmark as the database system workload. TPC-H [48] is the standard benchmark for online analytical processing (OLAP) applications, and is commonly used in database studies. It is specified by the Transaction Processing Performance Council (TPC) [48]. See Appendix B for details on the TPC-H benchmark. The queries of TPC-H benchmark are formalized from real world business questions. The specification of the target performance goals by users is usually done at the business transaction level. Therefore the best choice for the basic transaction level is the business question level.

3.3.2 Partitioning the Workload

A workload can be viewed as a collection of heterogeneous components depending on factors such as the resource usage level. Because of this heterogeneity, the representation of a workload by a single class often lacks accuracy. However, defining many classes can cause a dramatic increase in the computational efforts. Based on factors such as resource utilization, functionality, type of transaction, and applications, a workload partitioning can be done using techniques such as averaging and clustering. The degree to which one should refine the class definition is guided by the computation complexity and goals of the capacity planning study.

We simplify and classify all 22 TPC-H queries into a small number of general workload classes based on their resource usage. The CPU demand and disk demand of all queries are classified into three categories (simple, complex, very complex) using a K-means clustering algorithm. The combination of these three categories from CPU and disk produces 6~9 general query classes, e.g. CPU simple/disk simple (SS), CPU simple/disk complex (SC), CPU simple/disk very complex (SVC) etc. All 22 TPC-H queries can be mapped into these general queries based on their CPU and disk demand values.

3.3.3 Measurement

We use the Windows NT performance monitor to collect performance data about the system. The Windows NT performance monitor is a lightweight application bundled with the Windows NT operating system [50]. It can display statistics for system resources such as processor, memory, file handler, thread, and process. The performance monitor also allows applications such as DB2/UDB to publish their own statistics. It can collect data at any user-specified interval, and export the data from collected charts, logs or reports to a spreadsheet, word processor or database for further manipulation. It is a very light application in terms of computer resource consumption has a minimal effect on the system's performance. We found it had less than a 3% effect on our test systems.

In order to find the parameters of the queueing network model, we ran a set of experiments and collected data about the CPU and disk. The NT performance monitor provides us with a set of more than 200 parameters (called counters). The counters that are of interest to us are: the **%Processor Time** which is the percentage of non-idle

processor time spent servicing DB2/UDB, and **%Disk Time** which is the percentage of elapsed time that the selected disk drive is busy servicing read or write requests.

3.3.4 Workload Parameterization

From the perspective of the queueing network model, the workload is a set of formalized parameters. The workload measures must be transformed into input values of the queueing network model. The choice of the workload parameters depends on the type of the workload applied to the system. There are two types of workloads: a *closed* system workload and an *open* system workload. A *closed* system workload is characterized by a constant number of users/transactions. An *open* system workload is characterized by a variable number of users/transactions. Both batch and terminal workload systems are *closed* systems, while a transaction workload is considered to be an *open* system. As discussed above, we modeled the workload of the DBMS as a batch workload. A batch workload is characterized by a parameter *N*, indicating the average number of active jobs (customers). A batch workload has a fixed population. Customers that have completed service can be thought of as leaving the model and being replaced instantaneously from a backlog of waiting jobs.

Before it can be completed, each transaction needs a certain amount of service from each resource (CPU, disk). The service time needed from each resource, called the service demand, varies depending on the class of the transaction. To compute the service demand for a certain class of transactions from a resource, we use the following utilization law [23] (See Appendix A for details of utilization law):

$$Demand = \frac{Utilization * Time}{Completions}$$

Where *Demand* is the service time a class of transactions requires from a resource, *Utilization* is the percentage of time this resource is busy serving this class, *Time* is the total time the experiment is run, and *Completions* is the number of completed transactions of this class.

The CPU demand represents the total CPU processing time spent over multiple visits for one query to be completed. The disk demand is the total disk busy time to serve this query before it is committed. The CPU (or the disk) demand is calculated by dividing the product of the utilization of the CPU (or the disk) and the total interval time by the number of completions. This way, the estimated CPU (or disk) demand represents the amount of time the resource is busy serving one query. The CPU (or disk) demand does not include the queueing time, which is the time a query spends waiting for the resource to be free. The queueing time is taken into account in the computation of the response time in the queueing network model.

We run 22 TPC-H queries one by one, and measure their %Processor Time and %Disk Time using the Windows NT performance monitor to calculate their CPU demand and disk demand. The CPU demand and disk demand of each general query, identified with the K-means clustering algorithm, can be calculated by averaging all TPC-H queries belong to this class. The frequency of this general workload class is the percent of all queries in this class over 22 TPC-H queries.

Chapter 4 Queueing Network Model Development and Validation

Queueing network models are commonly used in capacity planning to model large computer systems. A model is constructed from information about the computer system configuration and measurements of resource requirements for each of the workloads modeled. The model is solved and the resulting performance metrics (response time, throughput, resource utilization, etc.) are compared to measured performance. The model is calibrated to the computer system. Then, it is used to study the effect of increases in workload and resource demands, and of configuration changes.

In this chapter, we present a general queueing network model of a DBMS, then modify and extend this model with different CPU and disk configurations and different database sizes to test the model's flexibility, portability and adaptability.

4.1 Building the Model

A queueing network model (refer to Appendix A) represents the behavior of queries in the DBMS. We model a query in terms of its use of three main resources, namely, CPU, disk and main memory. An OLAP query can be characterized as a series of *logical page accesses* and *page processes* (the period to process data on a page). A *logical page access* first tries the main memory. If the page requested is not in the buffer pool then it is read from the disk, a logical page access turns out to be a physical page access. The probability that a logical page access involves a disk access in OLAP queries is very high based on the fact that the OLAP queries scan large portion of the data instead of re-referencing them. The *page process* includes sort, group and calculate to give the final query result.

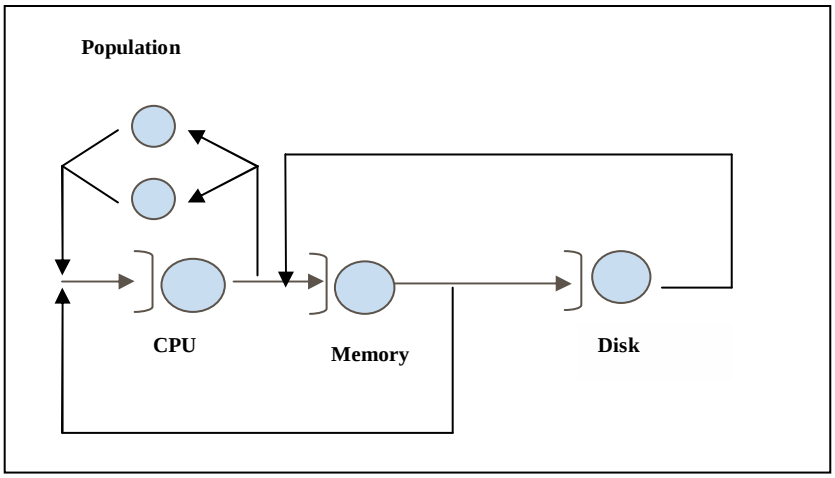


Figure 4-1. DBMS Queueing Network Model

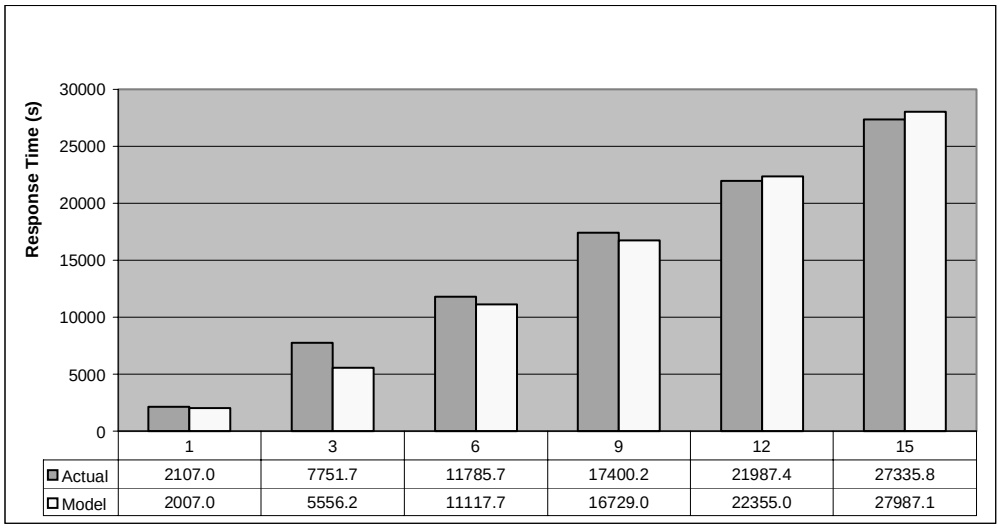
Figure 4-1 shows a queueing network model that represents a DBMS on a computer with a single CPU, disk drive, and a number of users concurrently using the system. A queue is associated with each resource. A client generates a query, which is

composed of a set of logical page accesses and processes. Considering the complexity of OLAP queries, the concurrent users of OLAP database server should be far less than OLTP users. The maximum number of concurrent users in our system is 15. Each of these clients submits a query and waits for the response of that query. Once the response is returned, the client composes a new query and submits it to the server. As discussed in Chapter 3, we model the workload of the DBMS as a batch workload. A closed queueing network model with a finite population is used.

A closed QNM requires three input parameters: the number of service centers, the service demand of each service center for every workload class, and the frequency of each workload class. There are three service centers in our QNM: CPU, disk and main memory. We consider the buffer area and sort heap to be the main memory resource in this study. The time to transfer data in and out of main memory is negligible so the service demand of the main memory is not significant. As we discussed in the previous section, the OLAP performance is independent of the size of buffer pool and sort heap when beyond their thresholds. Our QNM can be reduced to two service centers (CPU and disk) if the DBMS is well tuned (buffer pool and sort heap are larger than their thresholds).

One point that should be noted is that the QNM can model multiple users using the system concurrently, e.g. client *A* uses CPU service and client *B* uses disk service at the same time, but QNM cannot model one user using multiple service centers concurrently, e.g. client *A* uses the CPU service and the disk service at the same time. This scenario is common in DBMSs. The DBMS can prefetch data from disk while the CPU performs a sort operation. If the time spent on main memory is ignored, the actual

response time for each query should be close to the sum of CPU service time and disk service time. We found that the sum of the CPU and the disk service demand is always larger than the actual response time due to the asynchronous activity, and these asynchronous actions cannot be captured by the QNM. The asynchronous time was estimated by the difference between the sum of CPU and disk demand and the actual response time. Half of the asynchronous time was deducted from the CPU service demand and half from the disk service demand to reflect the asynchronous actions in the DBMS.



	1	3	6	9	12	15
Actual	2107.0	7751.7	11785.7	17400.2	21987.4	27335.8
Model	2007.0	5556.2	11117.7	16729.0	22355.0	27987.1

Figure 4-2. The Response Time vs. Client Number (Baserver, Database size 1GB)

In our experiments, we use a C language implementation of the TPC-H benchmark driver to simulate a typical OLAP workload on DB2/UDB DBMS. A set of Perl scripts (each script running the 22 TPC-H queries in random order) is used to simulate multiple clients using DBMS concurrently.

The way the script works is that as soon as a query is committed, another query is immediately submitted to the system, thus maintaining a fixed number of queries in the system at any instant. Therefore, since our system has a constant number of queries in the system, we model the workload of the DBMS as a batch workload.

In the experiments, we first run 22 TPC-H queries one by one, and collect the system usages using Windows NT performance monitor. The CPU service demand and disk service demand, and the frequency of each workload class are measured and calculated as described in Chapter 3. The service demands for each individual query are obtained independently. Then we run the system with multiple clients (multiple Perl scripts running concurrently), and measure their response times as the system performance indices.

We validate our queueing network model against DB2/UDB performance with TPC-H benchmark. For a closed QNM, there are three input parameters, the number of workload classes, the frequency of each workload class, and the service demands in each service center for those workload classes. As described above, system measurements are made and used to calculate the QNM input parameters. The performance indices are estimated using the QNM with these calculated parameters by Mean Value Analysis (MVA) tool [29]. The model is validated by comparing the calculated indices with the performance indices that measured directly from the system. Results for the response time metric are shown in Figure 4-2. In most cases the errors are within 10%.

Calculating the weight of CPU demand and disk demand in the overall service demand (sum of the CPU demand and the disk demand), we found that only 5 TPC-H

queries are CPU intensive, while 17 are disk intensive. Of the service demand, 81% is for disk and only 19% is for CPU. Based on the resource utilities calculated from QNM (Figure 4-3), we see that the disk utilization is very high (>90% when there are more than two concurrent users), and CPU utilization is low (~20%) for different population size. All the signs show that the disk is the bottleneck for our system.

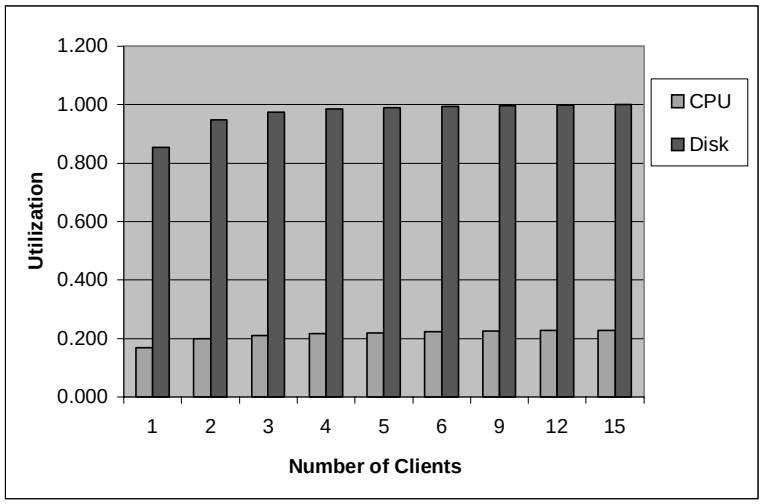


Figure 4-3. The QNM Calculated CPU and Disk Utilities for Different Population Sizes (Baserver, Database size 1GB)

4.2 Adaptability - System Extensions

We create and validate a queueing network model of a baseline system (1 CPU and 1 Disk) in Section 4.1. In this section we discuss the flexibility of this model and show how it can be used to predict system performance when disks or processors are added to the system.

4.2.1 Multiple Disks

In the baseline system, we found that the disk is the system bottleneck. To improve the performance, the first choice is to upgrade the disk system. Here we investigate the effect of adding disks to the system. Each disk was added along with a separate disk controller to eliminate the impact of the controller's bandwidth, and all database tables are evenly distributed on all disks. An example result with 9 clients is shown in Figure 4-4. The calculated response times, using MS Excel workbooks (ClosedQN.XLS) developed by Menasce [29] for solving parallel multiple service centers' QNM, are also shown in Figure 4-4.

From Figure 4-4 we can see that the total TPC-H response time (the sum of all 22 queries) decreases dramatically by increasing the number of disks. Note that the calculated values are always smaller than the actual measured values. The actual disk system is not optimized as expected in the QNM. In the QNM it is assumed that all disks are fully parallel and that there is no interaction between them. This assumption is not true for a DBMS and disk system. The locking in a DBMS and the relationships between different database tables limit the parallelism of the disk system, and this limitation depends on the workload and the database table distribution. We have to account for this in our QNM.

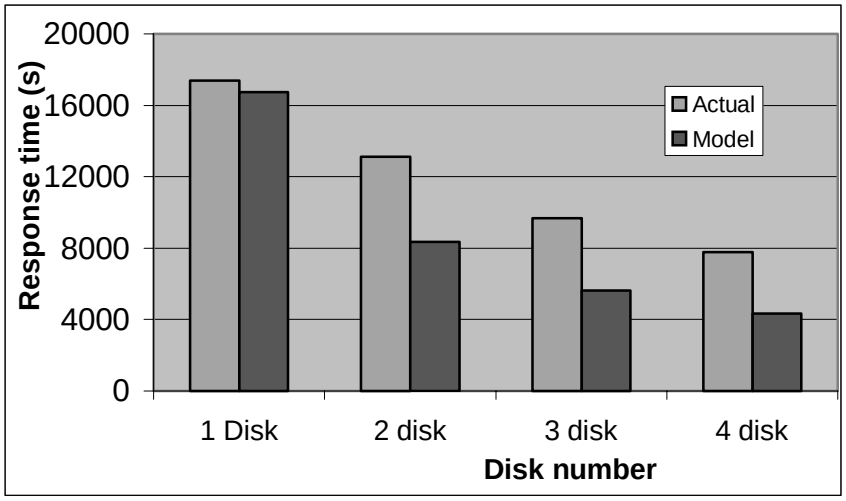


Figure 4-4. The Effect of Multiple Disks on TPC-H Performance (Baserver, Database size 1GB, one processor, 9 clients)

Some models are frequently used in estimating the aggregate processing capacity of multiprocessor systems [17]. Among them, the well-known Amdahl's law [2] is based on the serial fraction of a workload and would be a good fit for our problem. Amdahl's law is:

$$A(P) = \frac{P}{1+\sigma(P-1)}$$

where P is the number of processors in the system and $A(P)$ is the actual capacity. The parameter, σ, where $(0<\sigma<1)$, known as the seriality constant, refers to the serial fraction of the workload that cannot be made to execute in parallel. We tested different values of σ and found that 0.30 is a good fit for our system. Table 4-1 lists the corresponding P and A(P) values. From Table 4-1 we can see that the discrepancy between P and A(P) increases with adding more disks. The actual disk capacity is 2.11 when the system has 4 parallel disks ($\sigma = 0.30$).

Table 4-1. The Disk Number (P) and Calculated Disk Number A(P) Using Amdahl Law

P	A(P)						
	$\sigma = 0$	$\sigma = 0.1$	$\sigma = 0.15$	$\sigma = 0.2$	$\sigma = 0.3$	$\sigma = 0.4$	$\sigma = 0.5$
1	1.00	1.00	1.00	1.00	1.00	1.00	1.00
2	2.00	1.82	1.74	1.67	1.54	1.43	1.33
3	3.00	2.50	2.31	2.14	1.88	1.67	1.50
4	4.00	3.08	2.76	2.50	2.11	1.82	1.60
5	5.00	3.57	3.13	2.78	2.27	1.92	1.67

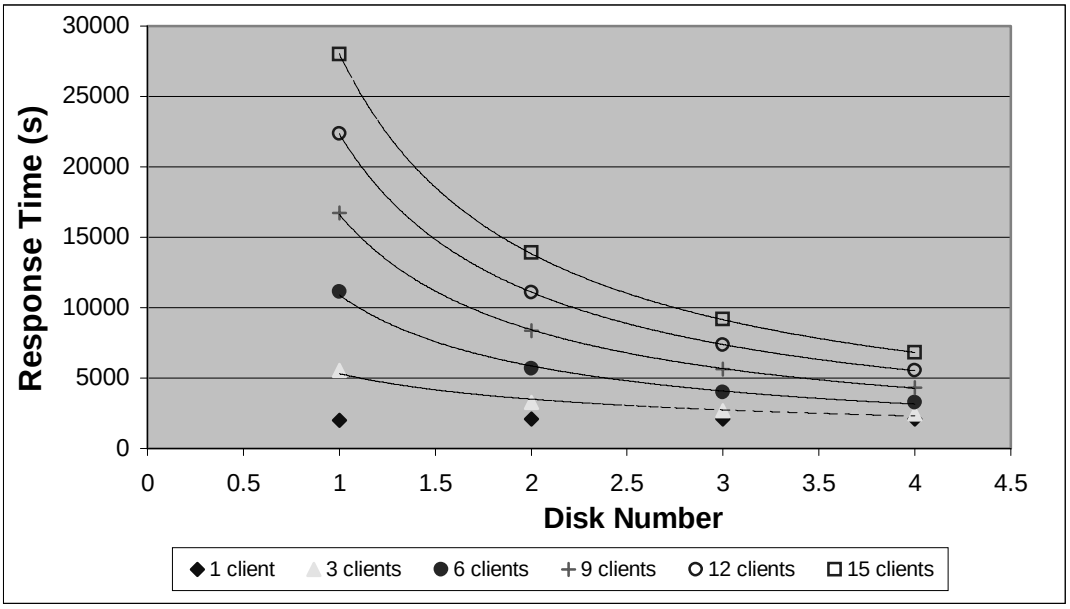


Figure 4-5. The Calculated Response Time Using QNM vs. Disk Number (Baserver, Database size 1GB)

Figure 4-5 plots and curve fits the calculated response time using our QNM vs. disk number for different numbers of clients. The final calculated response time can be obtained using this figure and Table 4-1. For example, the response time for a 4 disk system (P=4, σ=0.30) is the y-axis value at disk number (2.11) in Figure 4-5. Figure 4-6 shows the results of the QNM estimated values using Amdahl law and actual measured

response times for different numbers of clients using different disk configurations. The estimated values are within 10% of the actual values.

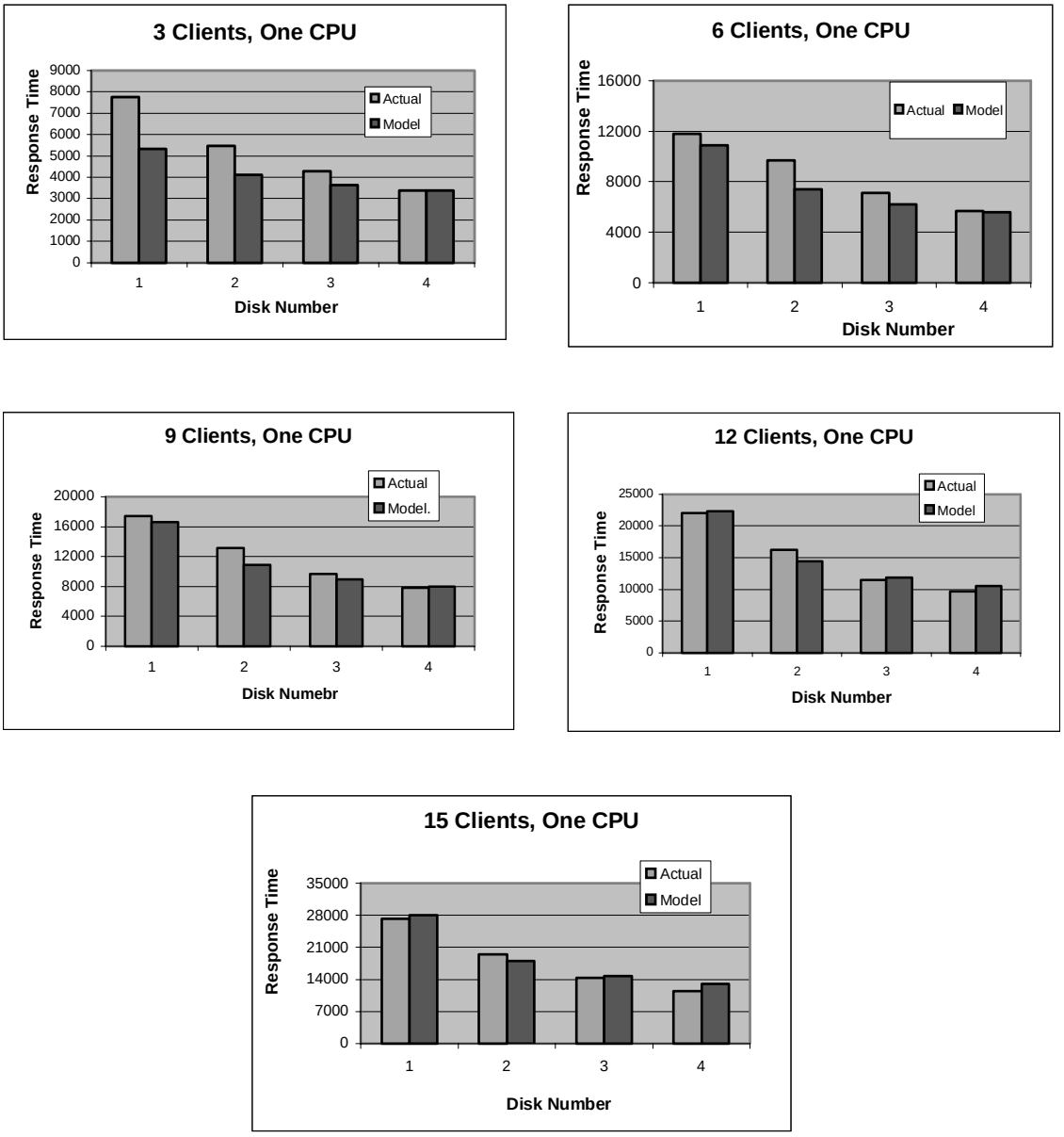


Figure 4-6. The Response Time vs. Disk Configurations (Baserver, Database size 1GB)

In order to test the correctness of this strategy, another system (Jaguar with 3 disks) was examined. The results for different population sizes are shown in Figure 4-7. Comparing the estimate response times from the model with the actual measured response time, the estimated error is within 10%. We also found that the seriality constant (σ=0.30) is the same as the previous system (Baserver), indicating that σ is related to the workload characteristics, not the system configuration.

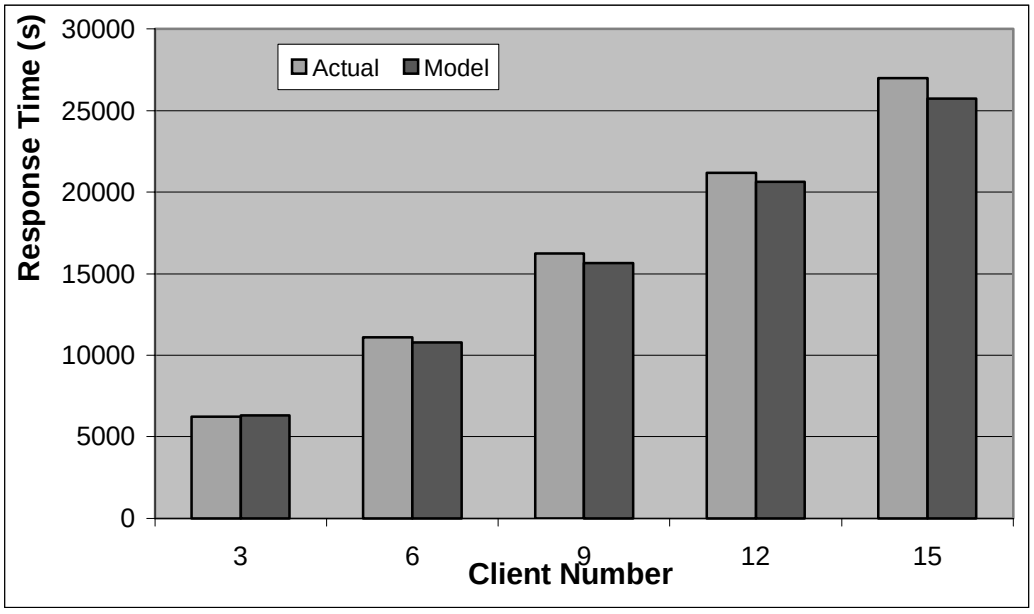


Figure 4-7. The Response Time vs. Client Number (Jaguar, Database size 1GB, 3 disks)

4.2.2 Multiple CPUs

We also investigated the effect of adding processors to the system. We assume that the CPU services are shared equally among the multiple processors, and the measurements from the NT monitor proved this hypothesis. Because the CPU is not the

bottleneck of the current system, adding one CPU only improves the TPC-H performance by 5-10%, while adding one disk led to a 20-30% improvement.

An example result is shown in Figure 4-8. The actual and the calculated response times are shown in Figure 4-8. The multiple processor effect [17] is not seen in the current system. The reason may be that CPU is not the bottleneck of current system, or that two processors are not enough to make a difference.

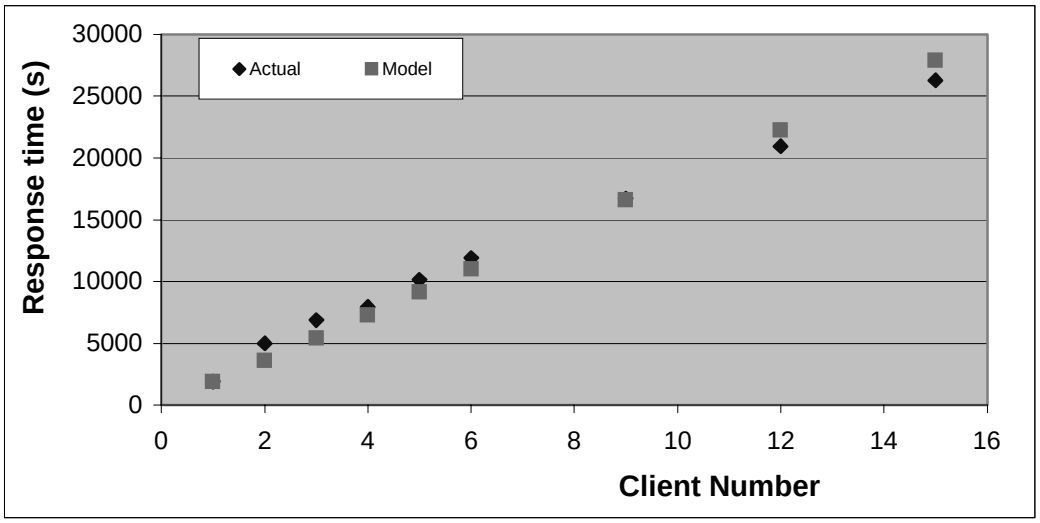


Figure 4-8. The Response Time vs. Different Population Size (Baserver, Database size 1GB, 1 disk, two Processors)

4.3 Adaptability – New System Configurations

In this section we discuss the portability of our QNM, that is, based on a known system, can we estimate and forecast the performance of a different system? Is our QNM portable from one system to another system?

4.3.1 Different Disk Specifications

Every disk can be characterized by basic quantities such as seek time, latency time, and transfer rate. In addition to these disk characteristics, the exact value of disk service demand is also dependent on block size, seek pattern, and I/O subsystem contention. If we use the same workloads and DBMS tuning parameters, it is possible to estimate the changes in disk service demands that will result from replacing one type of disk with another.

All our TPC-H queries are read only. They can be viewed as composed of a set of logical reads. If a logical read is not satisfied in the buffer area, it results in a physical disk access, which is further classified into two general disk access patterns: random read (RR) and sequential read (SR). The disk service time for a random read (RRT) and sequential read (SRT) can be calculated using the following equations [29]:

$$RRT = SeekTime + \frac{LatencyTime}{2} + TransferTime$$

$$SRT = \frac{SeekTime}{RunLength} + TransferTime + \frac{[1/2 + (RunLength - 1)(1 + DiskUtility)/2] \times LatencyTime}{RunLength}$$

where RunLength is the block size of a sequential read. For our system, we use the prefetch size (64 KB) as RunLength. The disk utility is very high (>90%) in our system. In this case, the SRT equation can be simplified into the following equation:

$$SRT = \frac{SeekTime}{RunLength} + TransferTime + LatencyTime$$

From the buffer pool snapshot function in DB2/UDB we can get the total number of physical reads and the total buffer pool read time for each query. Based on the calculated RRT and SRT using the disk specification and the above equation, the number of random reads (RR) and sequential reads (SR) can be estimated using the following two equations:

$$Physical\text{Re}ad = RR + SR$$

$$\text{Re}adTime = RR \times RRT + SR \times SRT$$

These numbers (random read and sequential read) are associated with the workload queries and will not change for different disk systems. Given a disk specification (seek time, latency time, and transfer rate), we can calculate RRT and SRT, and we can get the buffer pool read time for a specific query using the known RR and SR. We also observe that there is a linear relationship between buffer pool read time and disk demand (Figure 4-9). Based on this linear equation, we can calculate the disk demand from the buffer pool read time. Figure 4-10 shows the estimated disk demand and the actual disk demand of one system (Jaguar) for the general workload classes discussed in Section 3.3.2. Consequently, once we know the disk demand, we can plug it into our QNM and estimate the performance for a new disk specification. Figure 4-11 shows the results.

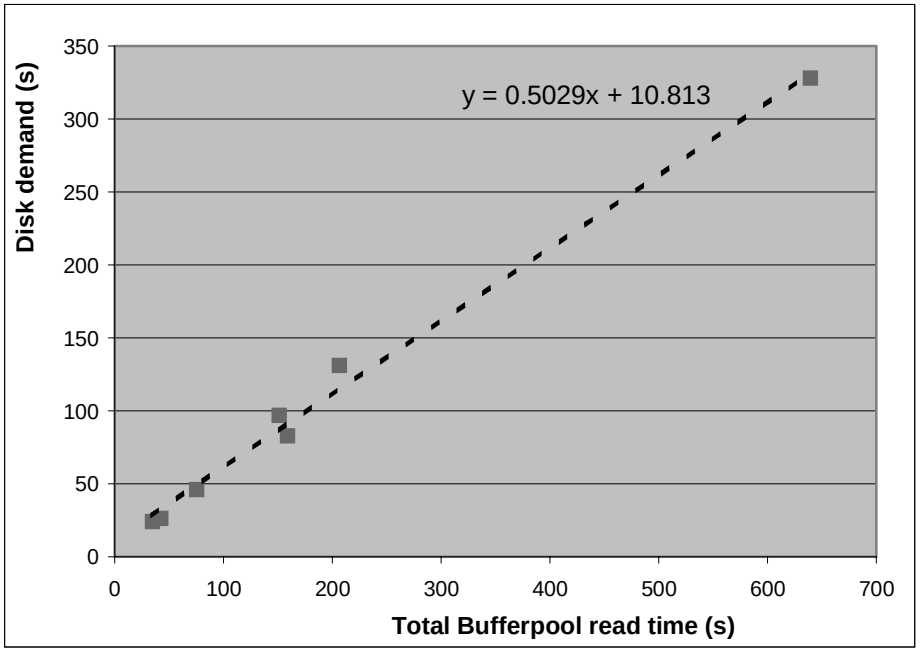


Figure 4-9. The Linear Correlation Between Total Buffer Pool Read Time and Disk Demand (Baserver, database size 1GB, 1 disk, 1 CPU)

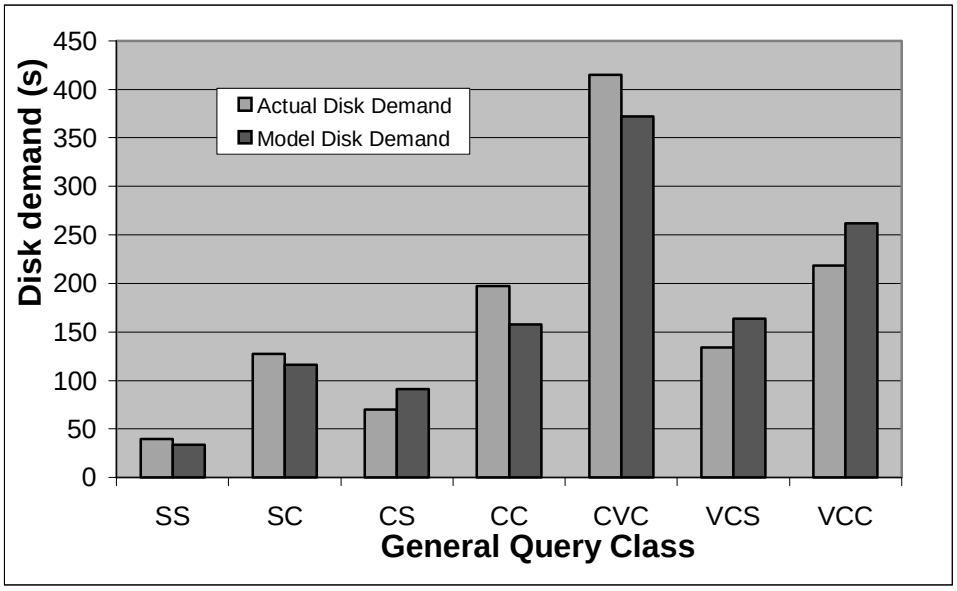


Figure 4-10. Disk Demand Prediction (Jaguar, database size 1GB, 1 disk, 1 CPU)

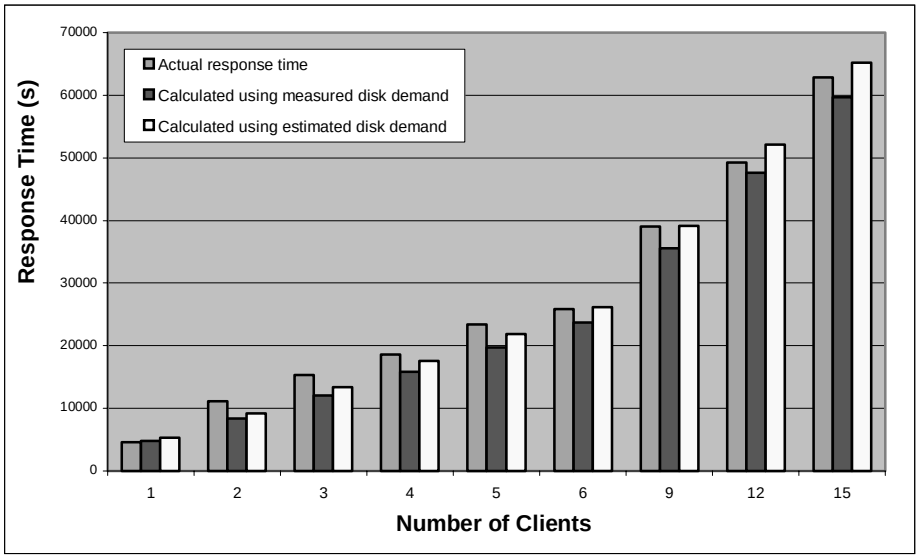


Figure 4-11. The Response Time (Actual, calculated using measured disk demand, and calculated using estimated disk demand) for Different Population Sizes (Cougar, database size 1GB, 1 disk, 1 CPU)

Using this approach, we can estimate the impact of any change in disk specification (seek time, latency time and transfer rate) on a given workload. We can also predict the performance of a new disk system based on a known disk system, that is, our QNM is portable from one disk system to another disk system.

4.3.2 Different CPU speeds

Another common hardware configuration change is the upgrade of a CPU within a family of processors of the same architecture. Usually this is one of the easiest changes to evaluate using a queueing network model. The relative instruction execution rates among processors within a family generally are known and publicized by vendors and user groups. The primary parameter change in QNM is to multiply the CPU demand by the ratio of the current CPU's processing rate to that of the new one.

In our DBMS system, we found that the CPU demand is not linear with the CPU processing rate. We checked three different CPU systems: Cougar (200 MHZ), Jaguar (400 MHZ), and Baserver (1000 MHZ). Their relative ratio of CPU demand is 1:1.7:3.4 , instead of the expected 1:2:5. Figure 4-12 lists the CPU demands and CPU speed with only one client using the system for some type of TPC-H queries (e.g. Q1 is the first query in TPC-H, and Q22 is the No.22 query in TPC-H benchmark). Based on the specifications, the CPU speed encompasses all of the processors (e.g. bus, CPU) in our granularity model. One reason for this non-linear relationship between CPU speed and CPU demand could be the CPU is not the bottleneck for the current system, changing the CPU speed will not result the significant changes in CPU demand.

Since we cannot use a linear equation, we employ curve-fitting techniques to estimate the effect of CPU speed on CPU demand. Using Figure 4-12, we can get the CPU demand of each TPC-H query for a certain CPU speed, which can be further generalized and plugged into our QNM to estimate the performance under this CPU speed.

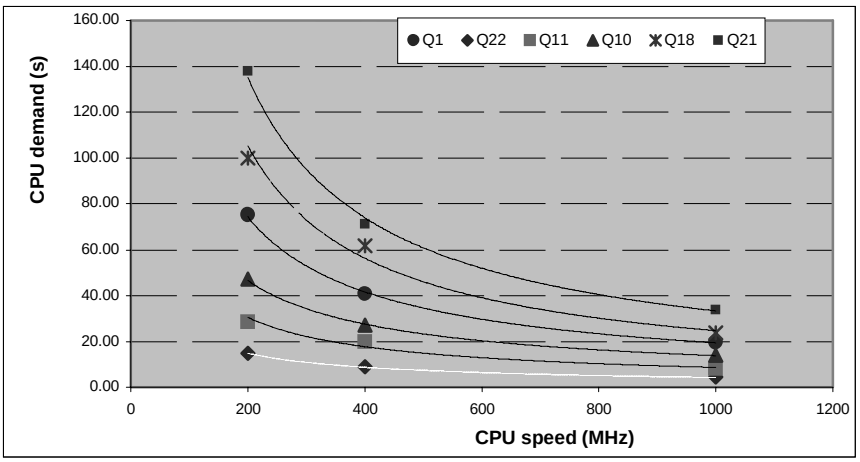


Figure 4-12. The CPU Speed Effect on CPU Demand (Database size 1GB, 1 disk, 1 CPU)

4.4 Adaptability – New Workloads

In this section we discuss the adaptability of our QNM to different workloads. We focus on two kinds of changes to the workload, namely in the number of users and in the database size. Changing the number of users of DBMS is equivalent to changing the intensities of the workload components, which is a frequently studied workload change in QNM that could be reflected by modifying the appropriate workload intensity input parameters. For example, a workload forest, ("a 50% increase in the number of active users"), can be represented in the model by multiplying the frequencies of each workload class by 1.5. This approach is used through out our calculation and shows good accuracy (Figures 4-2, 4-6, 4-7, 4-8, 4-12).

Changing the database size has a complex effect on DBMS memory (shown by studies on buffer pool and sort heap presented in Chapter 3), CPU demand and disk demand. We tested three database sizes 1GB, 5 GB and 10GB. Using the buffer pool snapshot in DB2/UDB we obtain the number of logical reads, physical reads and buffer pool read time for each query. The numbers of logical reads and physical reads increases linearly with database size, i.e. 5 times and 10 times the reads of 1GB database for the 5GB database and 10GB database, respectively. The increase in buffer pool read time, however, is less than linear. We calculate the number of random reads and sequential reads using the method in Section 4.3.1. Comparing with the 1GB database, the number of random reads increases 4-fold, the number of sequential reads increases 6.3-fold in the 5 GB database, and in the 10 GB database 9.7-fold and 11.8-fold. The percentage

increase in random reads is less than the percentage of database growth, and the percentage increase in sequential reads is more than the percentage of database growth. This is a result of an increase in asynchronous actions in the larger databases. Measured asynchronous action shows 33.5% asynchronous activity for the 1GB and 48.1% asynchronous activity for the 5GB database. This explains why the buffer pool read time is less than the linear estimate for a larger database.

We monitor the performance indices using the NT monitor and calculate the CPU demand and disk demand for each TPC-H query for different database sizes. Figure 4-13 and 4-14 show the results of some queries. From these graphs we observe that the disk demand increases much faster than the CPU demand, the CPU demand increases linearly with the database size, while the disk demand increases exponentially with the database size. We have already determined that the disk is the bottleneck for the current system. A larger database increases the burden on the disk and further aggravates the bottleneck. The performance is greatly impacted by this change. The response time in the 5GB database is 12% more than the value of 5-fold 1GB, and changes to 25% for the 10GB database.

Based on the above analysis, the CPU demand and disk demand cannot be calculated by multiplying the database size. The best choice is the curve fitting as shown in Figures 4-13 and 4-14. After getting the CPU demand and disk demand, the performance with new database size could be estimated by plugging them into QNM.

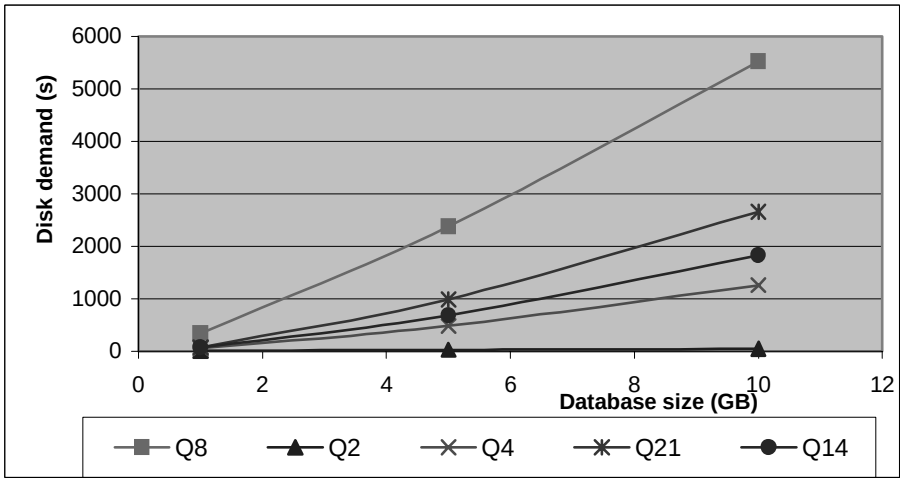


Figure 4-13. The CPU Demands of Some TPC-H Queries as a Function of Different Database Sizes (Baserver, 4 disk, 2 CPU)

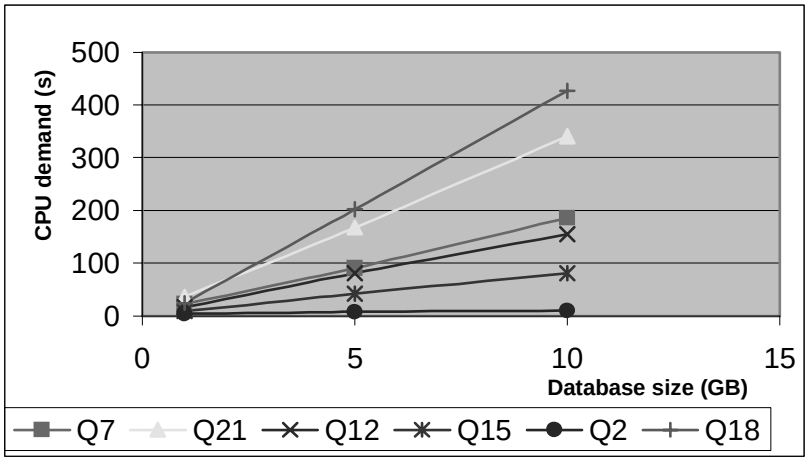


Figure 4-14. The Disk Demands of Some TPC-H Queries as a Function of Different Database Sizes (Baserver, 4 disk, 2 CPU)

4.5 Discussion and Summary

The most challenging aspect of database system analysis using queueing network models is not the technical details of defining, parameterizing, and evaluating the models.

Rather, it is the process of tailoring the general "methodology" of queueing network modeling to a specific database system. Because queueing network models are abstract, many assumptions are made in conducting a modeling study. These assumptions are motivated by simplicity, adequacy of measurements, and ease of evaluation.

In this chapter, we first built a baseline QNM to represent the DBMS using OLAP workloads with CPU, disk and memory as three separate service centers. Based on our examination in Chapter 3, because OLAP performance is independent of the size of buffer pool and sort heap when beyond their thresholds, the memory service is constant and can be ignored. Our QNM was reduced to two service centers (CPU and Disk) if the DBMS is well tuned (buffer pool and sort heap are larger then their thresholds).

Then we extend this model to different disk and CPU configurations and workloads to increase its flexibility for multiple disks and multiple CPUs, portability for different disk specifications and CPU speeds, and adaptability for different number of users and database sizes. DBMS is a comprehensive system. A simple QNM cannot capture all the aspects in it. Some equations and schemes are developed for the OLAP workloads besides the QNM: Amdahl's law is used to estimate the effect of multiple disks on DBMS performance. A mathematical scheme, based on the disk configurations (seek time, latency time, and transfer rate), DBMS internal indices (physical read, read time) from buffer pool, and workload characteristics (sequential scan run length), is proposed and used to predict disk demand for different disks.

During the process of model development and validation, we learned:

- A clear understanding of the objectives of a modeling study can contribute to simplicity in the model and in the modeling effort.

- Because queueing network models are abstract, many assumptions are made during a modeling study. It is important to be explicit concerning the assumptions that are made, the motivations for their introduction, and the arguments for their plausibility.
- Conducting a modeling study is an iterative process because of dependencies that exist among the definition of the model, the measurements used to parameterize the model, the techniques used to evaluate the model, and the objectives of a particular modeling study.
- Workload characterization is a challenging, inherently imprecise process. Useful insights can be obtained despite this imprecision. Characterizing a workload systematically helps to achieve flexibility.
- Valuable insights are gained throughout the modeling cycle, not merely during the projection phase.

Chapter 5 A Case Study

In Chapter 4 we describe a baseline QNM and extend it to various system configurations. We verify that the model is deemed to be a valid fit to represent the DBMS under study and, therefore, can be used to evaluate the performance indices under different workloads and system configurations.

To provide a more comprehensive idea of how our scheme could be used in a capacity planning study, we present a typical capacity planning in a scenario that the workload (the number of users) is increasing. We use our scheme to examine several possible upgrades and then justify the selection of one of them.

5.1 Description of the Example System

We use an IBM-Xseries 240 as our example system, with one 1GHz CPU, and one disk. The disk specifications are listed in Table 5-1. The database size is 1GB. The DBMS is tuned with buffer pool (5,000 pages) and sort heap (10,000 pages) beyond their

thresholds, prefetching turned on and the DB2/UDB default prefetching size of 16 pages (64 KB).

Table 5-1. The Disk Specifications of Example System

Type	Rotate Per Minute (RPM)	Transfer Rate (MB/Sec)	Average Seek Time (ms)	Capacity (GB)
Seagate ST 318436LC SCSI	7800	160	5.9	18.37

In our capacity planning scenario, the number of concurrent users is 6, and the response time (sum of all 22 TPC-H queries) is 11,786 seconds. The workload is expected to increase by 100%. It is required that the response time should not exceed 12,000 seconds with the increased workload.

In the next section, we build a QNM for the example and use the QNM to determine if there is any potential degradation in the system performance with the new workload. We then show how to use our model to project the effect of different upgrade choices to assist in making a cost-effective decision. At the end of this chapter, we analyze and discuss the results.

5.2 Building a Baseline Model

A queueing network model that has two service centers (CPU and Disk) and six concurrent users is used to represent the current system. The CPU demand and disk demand for each TPC-H query are measured on the current system using the Windows NT performance monitor. The TPC-H queries are further classified into eight general

workload classes (SS, SC, SVC…) using the method discussed in the previous chapters. Table 5-2 lists all the parameters for the QNM. *Number of Request per Class* is calculated by multiplying the frequency of each general workload class with the number of clients (which is 6 in the current system).

Table 5-2. The Parameter Values of QNM for Example System (6 clients)

Workload Class	SS	SC	SVC	CS	CC	CVC	VCS	VCC
Number of Requests per Class	1.36	0.27	0.27	0.55	1.64	0.27	1.09	0.55
CPU demand (s)	6.4	5.6	7.5	6.1	9.6	19.9	17.5	15.3
Disk demand (s)	33.0	102.6	329.6	26.2	100.2	335.5	26.4	94.6

Plugging the parameters into the QNM, we calculate the response time and resource utilization. The response time is 11,118 seconds, which is close to the actual measured value 11,786 seconds. The utilization for CPU and disk is 22.3% and 99.3%, respectively.

Doubling the number of concurrent users to 12 results in a 100% increase in the workload. In the QNM, the CPU demand and disk demand per query do not change with the increase in the number of users. The only change is the *Number of Requests per Class*. Using the new set of parameters, we estimate that the response time will jump to 22,355 seconds, which exceeds the maximum allowable response time. Therefore the database administrator must take corrective actions to prevent the system performance from degrading.

5.3 Upgrade Solutions

An administrator can consider multiple options to control the situation, but before making a decision, the administrator has to make sure that whatever the corrective action will be, it has to bring down the response time to below 12,000 seconds. The options that are available to the administrator are upgrading the disk, upgrading the CPU or a combination of both.

There are two general upgrade methods: scale out and scale up [28]. Scaling out means that more devices of the same type are added, while scaling up means replacing the existing devices with faster ones. To make a decision on which way to go, the administrator can use our QNM to estimate the response time under each of these cases.

5.3.1 Disk Scaling Out

Using the baseline QNM, we found that the disk is the bottleneck for the current system (utilization of disk is 99.3%). To improve the performance, the first choice is to upgrade the disk system. Disk scaling out is an easy and cheap way to upgrade the disk system. For simplicity in our study, each disk is added along with a separate disk controller to eliminate the impact of controller's bandwidth, and all database tables are evenly distributed on all disks.

The CPU demand and disk demand do not change. The only change in the QNM is the number of disk service centers. As in Chapter 4, we uses Amdahl's law with the seriality constant σ=0.30 to account for effects of the multiple disks.

Using our QNM, the estimated response time for the increased workload is 14,462 seconds if one disk is added to the disk system, and 11,828 seconds if two disks are added to the disk system. We verified the results by employing the disk systems and measuring their performance. The measured response time for 2 disks system is 16,231 seconds, and 11,481 seconds for 3 disks system. Based on the above results, we can see that at least two more disks are needed in order to bring the response time below the required 12,000 seconds.

5.3.2 Disk Scaling Up

Another choice to upgrade the disk is to replace the current disk with a faster one. We assume that the specification of the new disk is given in Table 5-3.

Table 5-3. The Specifications of New Disk System

Rotate Per Minute (RPM)	Transfer Rate (MB/Sec)	Average Seek Time (ms)
15600	320	3.0

Based on these three disk parameters, we can calculate the random read time (RRT) and sequential read time (SRT) using the equations in Chapter 4. We know that the number of disk reads for each query will not change with the disk specifications. The buffer pool read time for each query can be calculated based on its known values for random reads (RR), sequential reads (SR), and RRT and SRT. The disk demand can be inferred from the buffer pool read time based on the linear relationship between buffer pool read time and disk demand. The CPU demand will remain constant. All the parameters of QNM for the upgraded disk system are listed in Table 5-4. Plugging them

into our QNM, we calculate the response time for the increased workload with the faster disk system to be 12,898 seconds, which is still higher than the required 12,000 seconds. Replacing the disk with a faster disk does not appear to be a valid option.

Table 5-4. The Parameter Values of QNM for Upgraded Disk System (12 clients)

Workload Class	SS	SC	SVC	CS	CC	CVC	VCS	VCC
No. Requests per Class	2.73	0.55	0.55	1.09	3.27	0.55	2.18	1.09
CPU demand (s)	6.4	5.6	7.5	6.1	9.6	19.9	17.5	15.3
Disk demand (s)	19.7	51.3	173.3	21.5	49.3	173.3	29.9	63.3

5.3.3 CPU Scaling Up

Upgrading a CPU within a family of processors is another common hardware configuration change. Usually this is one of the easiest changes to evaluate using queueing network model. We assume that the processor speed changes from 1 GHz to 2 GHz. In our discussion of Chapter 4, we found that the CPU demand is not linear with respect to the CPU processing rate. We plotted the CPU demand vs. three different CPU speeds for TPC-H queries in Figure 4-12. The CPU demand with the new processor speed can be inferred and calculated from this graph. Plugging them into the QNM, we obtain a response time of 22,436 seconds. Replacing the CPU with a faster CPU does not make any performance improvement.

5.3.4 CPU Scaling Out

Rather than acquiring a faster CPU, it is sometimes possible to acquire a second processor to form a tightly coupled multiprocessor system. We show how to use our model to predict the performance of adding an extra CPU.

The CPU demand and disk demand for each workload class remain constant when moving to a dual processor. The only change in the QNM is the number of CPU service centers. The modeled response time is 22,245 seconds for the dual processor system. We further verified this result by adding one processor to the example system, and measured the response time with increased workloads. The actual value is 20,942 seconds.

5.3.5 Mixed Solution

Instead of upgrading the disk system or the CPU separately, the QNM can be used to estimate the effects of combinations of the above choices. In the following we give one example of a combination of choices in Section 5.3.1 and 5.3.4, i.e. adding one CPU together with one disk to the example system. In this case the baseline QNM becomes a model with two CPU service centers and two disk service centers. Solving this model, we obtain the response time for increased workloads, which is 14,491 seconds. The actual measured response time is 15,288 seconds.

5.4 Discussion of the Results

The principal value of a validated queueing network model is its utility as a basis for performance projection. In this chapter, we have shown, through a concrete example, how our QNM is applied in a capacity planning study, how to use the QNM to find the bottleneck, how to gain an in-depth knowledge on the multiple options and answer various what-if questions without any actual system set up, and how to estimate and quantify the effect of proposed upgrade options to find the most cost-effective solution.

As mentioned in the previous chapter, if we want to improve the performance, we should first find the bottleneck resource, the resource with the highest utilization, and then look into how to improve or remove this bottleneck. In the baseline QNM of Section 5.2, we found that disk utilization (99.3%) is much higher than CPU utilization (22.3%). The disk is the bottleneck of example system, and this bottleneck is further aggravated by increases in workload.

Upgrading the disk system is much more effective than upgrading CPU in our example scenario. Adding two disks cuts the response time from 22,355 seconds to 11,828 seconds and fulfills the performance requirement (response time less than 12,000 seconds). Any attempt to improve performance by tweaking other resources (e.g. upgrading the CPU) does not result in any significant improvements in the overall system. Some of the upgrades may even degrade the performance, because they increase the queue length of bottleneck resource. For example, when the processor speed changes from 1 GHz to 2 GHz, the modeled response time increased from 22,355 seconds to 22,436 seconds.

Scaling up is usually more effective than scaling out because of the multiple resource effects [2]. The response time is 14,462 seconds when adding one disk to the example system, while it is 12,898 seconds when the disk speed is doubled. However, scaling up is more expensive than scaling out, and it is often limited by the technology development.

Chapter 6 Conclusions and Future Work

OLAP applications typically involve very large amounts of data and place significant demands on DBMS resources. Capacity planning techniques are needed to avoid the pitfalls of inadequate resources and to meet users' performance expectations in a cost-effective manner. This thesis attempts to lay a foundation for capacity planning studies for DBMSs using OLAP workloads.

6.1 Thesis Contributions

Although in this thesis we specifically use DB2/UDB and the Windows NT performance monitor in our study, the techniques developed in the thesis are not specific to these platforms and can be applied to any other database management system or operating system.

The main contributions of this thesis are a study of DBMS factors influencing OLAP performance, the design and validation of a queueing network model to capture the main features of the DBMS behavior, and the use of a quantitative approach to project DBMS performance with an OLAP workload.

We built a workload model for OLAP, which is based on the TPC-H benchmark and investigated the impact of DBMS tuning factors (such as buffer pool, sort heap, prefetching and number of I/O servers) on OLAP performance. We found the following:

- There are certain thresholds on buffer pool and sort heap size in DBMS with OLAP workload. We investigated the relationships between these thresholds, database size and hardware configurations.
- The buffer pool threshold is not correlated with the sort heap parameter, nor with the hardware configuration. It is closely related to, and increases along with, the database size.
- The sort heap threshold is dependent on the query content and different queries have different thresholds. The disk configuration has no effect on the sort heap threshold, while CPU speed plays an important role.
- A slower processor usually needs more sort heap memory and has a larger sort heap threshold. The required sort heap threshold increases along with the database size.
- Prefetching improves OLAP performance and should be always turned on.

Based on these findings, we proposed a queueing network model (QNM) to represent the system under study and gave the results of preliminary experiments to validate this model. In the thesis,

- We have indicated, through discussion and example, how to modify the parameters of this model to represent various common changes to the

hardware (CPU and disk upgrade) and workload (number of users and database size).

- Amdahl's law is used to estimate the multiple disk effect on DBMS performance when changing the number of disks.
- A mathematical approach, based on the disk configurations (seek time, latency time, and transfer rate) and DBMS internal indices (physical read, read time) from the buffer pool, is proposed and used in predicting the disk demands for different disks.

Queueing network models are effective approaches to capacity planning. This is due to the fact that queueing network models achieve a favorable balance between accuracy and efficiency.

In terms of accuracy, a large body of experience [23, 27] and the results in this thesis indicate that queueing network models can be expected to be accurate to within 5 to 10% for utilizations and throughputs and to within 10 to 30% for response times. This level of accuracy is consistent with the requirements of a wide variety of design and analysis applications.

In terms of efficiency, we have indicated in the previous section that queueing network models can be defined, parameterized, and evaluated with a relatively low cost. Definition is eased by the close correspondence between the attributes of queueing network models and the attributes of DBMSs. Parameterization is eased by the relatively small number of high-level parameters. Evaluation is eased by the recent development of algorithms and tools [27].

6.2 Future Work

A number of related issues require further study:

- First, systems with larger number of CPUs and disks and larger database size need to be tested to verify and quantify the multiple processor effect, the relationships between CPU demand and processor speed, CPU demand and database size, and disk demand and database size.
- The second issue is to integrate our model with the previous QNM using OLTP workload [56], using a general workload model.
- Third, similar studies for DBMSs using other benchmark workloads such as TPC-W (Web workload) should be conducted. The future direction of this model is to allow the user to define the database and workload (SQL statements, batch files, utilities such as copy, recover and rebuild), and then use this model to estimate the performance of different systems under the specified workloads.

Like programming, modeling is more of an art than a science. The more information is supplied, then the more accurate the model. Also, the more information provided, the more difficult it is to build the model, and the model will be less adaptable. There is a trade-off between accuracy, cost and adaptability in capacity planning. The challenge is to come up with a scheme that is rich enough to be useful, and yet simple enough to be manageable.

Bibliography

1. F. Allen. “A Predictive Performance Evaluation Technique for Information Systems”, *Proceeding of 4th International Symposium on Modeling and Performance Evaluation*, 1979.

2. G. Amdahl. “Validity of The Single Processor Approach to Achieving Large Scale Computing Capabilities”, *Proceeding of The American Federation of Information Processing Societies Conference*, 1967.

3. L. Belady. “A Study of Replacement Algorithms for a Virtual-Storage Computer”, *IBM Systems Journal* 5 (2), July 1966.

4. P. Bernstein. “Database Technology: What’s Coming Next?”, *Keynote address at High Performance Computer Architecture-4*, February 1998.

5. K. Brown, M. Mehta, M. Carey and M. Livny. “Towards Automated Performance Tuning for Complex Workloads”, *Proceeding of 20th International Very Large Database Conference*, Santiago, Chile, September 1994.

6. J. P. Buzen. *Queuing Network Models of Multiprogramming*, Ph.D. Thesis, Harvard University, Cambridge, MA, 1971.

7. A. F. Cardenas. “Evaluation and Selection of File Organization – a Model and System”, *Communications of the Association for Computing Machinery*, 16, September 1973, pp. 540-48.

8. A. F. Cardenas and J. P. Sagamang. “Modeling and Analysis of Database Organization – the Doubly Chained Tree Structure”, *Information Systems* 1, 2 (April 1975), pp. 57-67.

9. K. M Chandy and D. Neuse. “Linearizer: A Heuristic Algorithm for Queueing Network Models of Computing Systems”, *Communications of the Association for Computing Machinery*, 25(2), February 1982, pp. 126-134.

10. S. Chaudhuri and U. Dayal. “An Overview of Data Warehousing and OLAP Technology”, *The Association for Computing Machinery SIGMOD Record* 26(1), March 1997.

11. M. Chen, J. Han and P. Yu. “Data mining: An Overview from Database Perspective”, *IEEE Tran. on Knowledge and Data Eng.*, 8(6) December 1996.

12. *DB2 Estimator Help*, DB2 Estimator V7, IBM Corp. 1997, 2000.

13. D. Ferrari. *Computer Systems Performance Evaluation*, Prentice Hall, 1978.

14. B. I. Galler. *Concurrency Control Performance Issues*, Ph.D. thesis, University of Toronto.

15. T. J. Gambino and R. Gerristen. “A Database Design Decision Support System”, *Proceeding of 3rd International Very Large Database Conference*, Tokyo, 1977, pp. 533-44.

16. G. S. Graham. “Queuing Network Models of Computer System Performance”, *Computing Surveys*, Vol. 10, NO. 3, September 1978.

17. N. J. Gunther. “Understanding the MP Effect: Multiprocessing in Pictures”, *Proceeding of Computer Measurement Group Conference*, 1996.

18. IBM, *DB2 Universal Database*. http://www.software.ibm.com/data/db2/udb.

19. K. B. Irani and H.-L. Lin. "Database Concurrency Control: Queueing Network Models for Concurrent Transaction Processing in a Database System", *Proceeding of the Association for Computing Machinery SIGMOD International Conference on Management of Data*, May 1979.

20. K. Irani, S. Purkayastha, and T. J. Teorey. "A Designer for DBMS-processable Logical Database Structures", *Proceeding of 5th Association for Computing Machinery SIGMOD International Conference*, Boston 1979.

21. S. Lam and K. H. Chan. *Computer Capacity Planning: Theory and Practice*, Academic Press, 1987.

22. S. S. Lavenberg and G. S. Shedler. "Stochastic Modeling of Processor Scheduling with Application to Database Management Systems", *IBM Journal of Research & Development 20*, September 1976.

23. E. Lazowska, J. Zahorjan, S. Graham, and K. Sevcik. *Quantitative System Performance: Computer System Analysis Using Queueing Network Models*, Prentice Hall, Englewood Cliffs, N. J., 1984.

24. M. Lefler, M. Stokrp and C. Wong. "DBSim: A Simulation Tool for Predicting Database Performance", *Proceeding of the Association for Computing Machinery SIGMOD International Conference on Management of Data*, Montreal, Quebec, Canada, 1996.

25. S. T. March and D. G. Severance. “A Mathematical Modeling Approach to the Automatic Selection of Database Designs”, *Proceeding of the Association for Computing Machinery SIGMOD Conference* 1978, pp. 52-65.

26. P. Martin, H. Li, M. Zheng, K. Romanufa and W. Powley. “Dynamic Reconfiguration Algorithm: Dynamic Tuning Buffer Pools”, *Proceeding of Database and Expert Systems Applications Conference*, 2000.

27. D. Menasce, and V. A.F. Almeida. *Capacity Planning for Web Performance: Metrics, Models, & Methods*, Prentice Hall 1998.

28. D. A. Menasce, and V. A. F. Almeida. “Challenges in Scaling E-business Sites”, *Proceeding of Computer Measurement Group Conference*, 2000.

29. D. Menasce, V. A.F. Almeida, and L. Dowdy. *Capacity Planning and Performance Modeling: From Mainframes to Client-Server Systems*, Prentice Hall, 1994.

30. T. H. Merrett. “Database Cost Analysis: a Top-Down Approach”, *Proceeding of the Association for Computing Machinery SIGMOD Conference* 1977, pp. 135-143.

31. I. Miyamoto. “Hierarchical Performance Analysis Models for Database Systems”, *Proceedings of 1st International Very Large Database Conference*, 1975.

32. D. Motzkin, R. Ellendula, M. Kamali and S. Tiwari. “A Tool for Performance Evaluation of Database Systems for Small Computer Systems”, *Proceeding of the Association for Computing Machinery Symposium on Applied Computing*, Nashville, Tennessee, United States, 1995, pp. 420.

33. O. Oren and F. Ashim. “Statistics for the Usage of a Conceptual Data Model as a Basis for Logical Database Design”, *Proceeding of 5th International Very Large Database Conference*, 1979, pp. 140-45.

34. D. A. Patterson and K. K. Keeton. “Hardware Technology Trends and Database Opportunities”, *Keynote address at the Association for Computing Machinery SIGMOD’98 Conference*, Seattle, Washington, June 1998. Available from http://www.cs.berkeley.edu/~pattrsn/talks/sigmod98-keynote-color.pdf

35. M. T. Pezzaro. *Analytic Evaluation of Physical Database Designs*, 1980.

36. D. Potier and P. LeBlanc. “Analysis of Locking Policies in Database Management Systems”, *Communications of the Association for Computing Machinery* 23, 10, October 1980.

37. D. Sarris and J. Hofer. “Capacity Planning for e-Commerce Systems With Benchmark Factory”, *White Paper*. Available from http://www.capacityplanning.com/whitepaper

38. C. H. Sauer and K. M. Chandy. *Computer Systems Performance Modeling*, Prentice Hall, 1981.

39. M. E. Senko, V. Y. Lum, and P. J. Owens. “A File Organization Evaluation Model”, *Proceeding of International Federation for Information Processing Conference*, 1968, pp. 514-19.

40. K. Sevcik. “Data Base System Performance Prediction Using an Analytical Model”, *IEEE Computer*, February 1981.

41. D. Shasha and P. Bonnet. *Talk notes*, available from http://www.distlab.dk/dbtune/.

42. C. U. Smith and J. C. Browne. “Analysis of Software Designs: Representation and Evaluation of Competitive and Interference Effects of Concurrency and Blocking”, *Proceeding of Performance ’80 Conference*, Toronto, May 1980.

43. P. Stenstrom, E. Hagersten, D. J. Lijia, M. Martonosi and M. Venugopal. “Trends in Shared Memory Multiprocessing”, *IEEE Computer*, December 1997, pp. 44-50.

44. T. J. Teorey and K. S. Das. “Application of an Analytical Model to Evaluate Storage Structures”, *Proceeding of the Association for Computing Machinery SIGMOD Conference*, 1976, pp. 9-19.

45. T. J. Teorey and J. P. Fry. “The Logical Record Approach to Database Design”, *Computer Survey* 12, June 1980, pp. 179-211.

46. T. J. Teorey and L. B. Oberlander. “Network Database Evaluation Using Analytical Modelling”, *Proceeding of The American Federation of Information Processing Societies NCC 47*, Anaheim, 1978.

47. F. W. Tompa. “Choosing an Efficient Internal Schema”, *Systems for Large Databases*, North Holland Publishing Co. 1976, pp. 65-77.

48. *Transaction Processing Performance Council*. http://www.tpc.org

49. J. Vijavan. “Capacity Planning More Vital Than Ever”, *Computer World*, February 1999, pp 1.

50. *Windows NT Performance Monitor Online Help*. http://www.microsoft.com

51. S. B. Yao and D. Dejong. “Evaluation of Database Access Paths”, *Proceeding of the Association for Computing Machinery SIGMOD Conference* 1978, pp. 66-77.

52. S. B. Yao and A. G. Merten. “Selection of File Organizations Using an Analytical Model”, *Proceeding of 1st International Very Large Database Conference*, 1975, pp 255-67.

53. P. S. Yu, D. M. Dias and S. S. Lavenberg. “On the Analytical Modeling of Database Concurrency Control”, *Journal of the Association for Computing Machinery (JACM)* Volume 40 Issue 4, September 1993.

54. J. Zahorjan. “The Approximate Solution of Large Queueing Network Models”, Report CSRG-122, *Computer Systems Research Group*, University of Toronto 1980.

55. J. Zahorjan, K. C. Sevcik, D. L. Eager, and B. I. Galler. “Balanced Job Bound Analysis of Queueing Networks”, *Communications of the Association for Computing Machinery*, 1981.

56. H. Zawawy. *Capacity Planning for Database Management Systems Using Analytical Modeling*, Master thesis, School of Computing, Queen's University, December 2001.

Appendix A Queueing Network Modeling [21, 23]

A.1 What Is a Queueing Network Model?

Queueing network models (QNM) can be viewed as a subset of the techniques of queuing theory specialized to the representation of computer systems. It is an approach to computer system modeling in which the computer system is modeled as a network of queues evaluated analytically. A network of queues is a collection of service centers, which represent system resources, and users (or transactions) referred to as customers. Analytic evaluation is pursued by solving a set of equations induced by the network of queues and its parameters.

The strong correspondence between the attributes of queueing network models and the attributes of computer systems makes the process of representing a computer system by a queueing network model relatively straightforward. For example, service centers in queueing network models logically correspond to hardware resources and their

software queues in computer systems, and customers in queueing network models logically correspond to users or transactions in computer systems.

Queueing network models have a wide set of features and attributes that can be used to represent computer systems. As an example, there exists more than one way to specify the workload intensity. One method would be the rate at which customers arrive. Another approach is to state the number of customers in the model (This alternative best suits batch workloads). A third approach is to specify the number of customers and the average time that each customer "spends thinking" between interactions (This alternative best represents interactive workloads).

Another example of this richness is the multi-class workload model. Most computer systems have several identifiable workload components. Using queueing network models, it is possible to distinguish between a system's workload components by making use of multiple workload classes, each of which has its own workload intensity and service demands.

Queueing network models can be open or closed models. An open queueing network model represents a system with an infinite customer population where customers arrive, receive service, and exit. The number of customers in the system varies dynamically. A closed QN model represents a system with a fixed finite number of customers. Customers arrive, circulate among the system resources, and exit after completely receiving service. The number of customers in the system at any time is fixed.

A.2 Service Centers in QNM

A.2.1 Single Service Center

In a single service center, customers arrive at the service center, wait in the queue if necessary, receive service from the server, and depart. In fact, this service center and its arriving customers constitute a basic queueing network model.

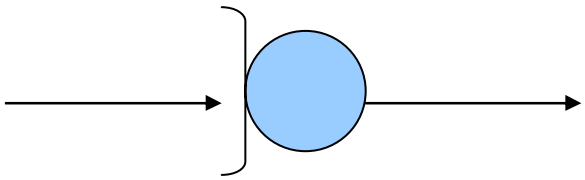


Figure A.2-1. A Single Service Center

A model such as the one in Figure A.2-1 has two parameters: the workload intensity and the service demand. The workload intensity in this case is the rate at which customers arrive (e.g. one customer every two seconds, or 0.5 customers/second). The service demand is the average service requirement of a customer (e.g. 1.25 seconds). For specific parameter values it is possible to evaluate this model by solving some simple equations, to yield performance measures such as utilization (the proportion of time the server is busy), residence time (the average time spent at the service center by a customer, both queueing and receiving service), queue length (the average number of customers at the service center, both waiting and receiving service), and throughput (the rate at which customers pass through the service center).

A.2.2 Multiple Service Centers

Characterizing a contemporary computer system by two parameters is an oversimplification of reality. Figure A.2-2 represents a more practical model in which each system resource (in this case a CPU and three disks) is represented by a separate service center.

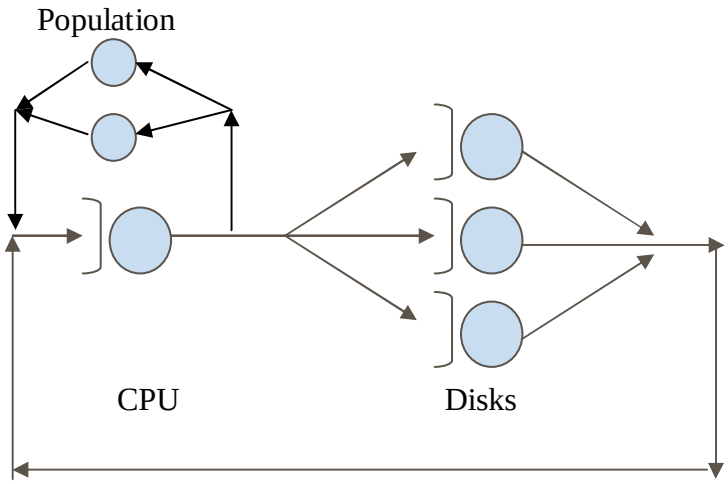


Figure A.2-2. A Multi-resource Queueing Network Model

The parameters of this model are similar to those of the previous one: the workload intensity, which is the rate at which customers arrive, and the service demand. However, this time a separate service demand is provided for each service center. If the customers in the model represent transactions in the system, then the workload intensity corresponds to the rate at which users submit these transactions to the system, and the service demand at each service center represents the total service requirement per transaction at the corresponding resource in the system. The customers are arriving, circulating among the service centers, and then leaving the system. The pattern of

circulation among the centers is not important, however; only the total service demand at each center matters.

A.3 Parameterization, and Evaluation of QNM

The parameterization process of queueing network models is relatively straightforward. For example, to calculate the CPU service demand for a customer in a queueing network model of an existing system, we would observe the system in operation and would measure two quantities: the number of seconds that the CPU was busy, and the number of transactions that were processed. Then, we would divide the busy time by the number of transactions completed to determine the average number of seconds of CPU service needed for each transaction to complete.

There are two main approaches to evaluating queueing network models. The first involves calculating bounds on performance measures, rather than specific values. This approach involves determining upper and lower bounds on performance parameter values such as response time for a particular set of inputs (workload intensity and service demands). The virtue of this approach is that the calculations are simple enough to be carried out by hand, and the resulting bounds can contribute significantly to understanding the system under study.

The second approach, called the mean value analysis (MVA), involves calculating the values of the performance measures. While the algorithms for doing this are sufficiently complicated that the use of computer programs is necessary, it is important to emphasize that these algorithms are extremely efficient. Approximate MVA algorithms

have been devised in order to reduce the complexity of calculations while keeping an acceptable accuracy. The running time of the most efficient general algorithm of the MVA grows as the product of the number of service centers with the number of customer classes, and is independent of the number of customers in each class.

The performance of any particular system is determined by carefully analyzing certain performance measures like response times, throughputs and utilization. The response time is the average time interval elapsed between the instant a transaction is submitted to the system for processing until the answer begins to appear at the user's terminal. Throughput is the average number of transactions or jobs executed per unit time. In other words, it is the rate at which requests are serviced. Utilization is the percentage of time the device is being used, during a given time interval. A device is saturated if its utilization approaches 100%.

A.4 Fundamental Laws and Mean Value Analysis

A.4.1 Utilization Law

The utilization of a resource is equal to the product of the throughput of that resource and the average service requirement at that resource. The utilization law is stated as:

$$U = X * S$$

where,

- U is the utilization of the resource
- X is the throughput and

- S is the average service requirement

A.4.2 Little's Law

Little's Law states that the average number of requests in a system is equal to the product of the throughput of that system and the average time spent in that system by a request. Little's law is stated as:

$$N = X * R$$

where,

- N is the average number of requests in the system
- X is the throughput and
- R is the system residence time per request

A.4.3 Forced Flow Law

This law is based on the fact that the flows or throughput in all parts of a system must be proportional to each other. The forced flow law is stated as:

$$X_k = V_k * X$$

where,

- X_k is the throughput at a resource k in the system
- V_k is the visit count of the resource k and
- X is the throughput of the system

A.4.4 Mean Value Analysis (MVA)

Mean value analysis is the technique most commonly used to resolve closed queueing networks models such as the model used in this thesis. The mean value analysis is based on three equations:

1 – The service center residence time equation:

$$R_k(N) = D_k * (1 + Q_k(N-1))$$

where,

- $R_k(N)$ is the residence time at center k
- D_k is the time needed at resource k for a transaction to be completed and
- $Q_k(N-1)$ is the queue length at resource k when there are N-1 customers in the system

2 - Little's Law applied to the queueing network as a whole:

$$X(N) = N / (Z + \Sigma^K R_k(N))$$

where ,

- X(N) is the system throughput
- $R_k(N)$ is the residence time at center k when there are **N** customers in the system and
- Z is the think time

3 - Little's Law applied to the service centers individually:

$$Q_k(N) = X(N) * R_k(N)$$

where,

- $Q_k(N)$ is the queue length at resource k when there are N customers in the system
- X(N) is the throughput of the system and
- $R_k(N)$ is the residence time at center k

First a solution for the trivial case when the population size is 1 (N=1) is produced. In this case there is no queueing at any of the resources ($Q_k(0) = 0$) and the residence time at each of the resources is equal to the service demand from each resource. As a result, the queue length for when there is one customer in the system is obtained. Based on this solution, the 3 equations are resolved again for the case when N=2, and so on. In our study we use the MS Excel workbooks (ClosedQN.XLS) developed by Menasce [27] to solve the QNM.

Appendix B TPC-H Benchmark [48]

The Transaction Processing Performance Council (TPC) is a non-profit corporation founded to define transaction processing and database benchmarks and to distribute objective, variable TPC performance results to the industry. The TPC-D Benchmark (TPC-D) was the first database benchmark defined by TPC to measure the performance and price/performance of an analytical processing system (or decision support system). In April 1999, two new benchmarks, TPC-R and TPC-H, replaced TPC-D as the industry's standard benchmarks for decision support applications; TPC-R for a *reporting workload* and TPC-H for an *ad-hoc querying workload*. An ad-hoc querying workload simulates an environment in which users connect to the database system and issue individual queries that are not known in advance. In this thesis we focus on an ad-hoc querying workload and use TPC-H benchmark as the base of our workload model.

B.1 Database Design and Population

The purpose of this benchmark is to reduce the diversity of operations found in an information analysis application, while retaining the application's essential performance

characteristics, namely, the level of system utilization and the complexity of operations. The database design and the data populating the database have been chosen to have broad industry-wide relevance while maintaining a sufficient degree of ease of implementation. This benchmark illustrates decision support systems that examine large volumes of data, execute queries with a high degree of complexity, and give answers to critical business questions. The TPC-H database consists of eight separate and individual tables (the base tables) as listed in Table B.1-1.

Table B.1-1. The Tables and Cardinalities in TPC-H Benchmark (Scale Factor: SF)

Table Name	Cardinality (in rows)	Length of Typical Row (in bytes)	Typical Table Size (in MB)
SUPPLIER	10,000*SF	159	2*SF
PART	200,000*SF	155	30*SF
PARTSUPP	800,000*SF	144	110*SF
CUSTOMER	150,000*SF	179	26*SF
ORDERS	1,500,000*SF	104	149*SF
LINEITEM	6,000,000*SF	112	641*SF
NATION	25	128	< 1
REGION	5	124	<1

Scale factors used for the test database are chosen from the set of fixed scale factors defined as follows: 1, 10, 30, 100, 300, 1,000, 3,000, and 10,000. The minimum required scale factor for a test database is SF=1, where the database is approximately 1GB.

B.2 Queries and Refresh Functions

The TPC Benchmark consists of a suite of business oriented ad-hoc queries and concurrent data modifications. TPC-H evaluates the performance of various decision support systems by the execution of sets of queries against a standard database under controlled conditions. The TPC-H queries give answers to real-world business questions, simulate generated ad-hoc queries, include a rich breadth of operators and selectivity constraints, and generate intensive activity on the part of the database server component of the system under test. Table B.2-1 lists the queries' number, name, and corresponding business questions.

Table B.2-1. The Queries and Refresh Functions of TPC-H Benchmark

Number	Name	Business Question
Q1	Pricing Summary Report Query	Reports the amount of business that was billed, shipped, and returned
Q2	Minimum Cost Supplier Query	Finds which supplier should be selected to place an order for a given part in a given region
Q3	Shipping Priority Query	Retrieves the 10 unshipped orders with the highest value
Q4	Order Priority Checking Query	Determines how well the order priority system is working and gives an assessment of customer satisfaction
Q5	Local Supplier Volume Query	Lists the revenue volume done through local suppliers
Q6	Forecasting Revenue Change Query	Quantifies the amount of revenue increase that would have resulted from eliminating certain company-wide discount in a given percentage range in a given year
Q7	Volume Shipping Query	Determines the value of goods shipped between certain nations to help in the re-negotiation of shipping contracts
Q8	National Market Share Query	Determines how the market share of a given nation within a given region has changed over two years for a given part type
Q9	Product Type Profit	Determines how much profit is made on a given

	Measure Query	line of parts, broken out by supplier nation and year
Q10	Returned Item Reporting Query	Identifies customers who might be having problems with the parts that are shipped to them
Q11	Important Stock Identification Query	Finds the most important subset of suppliers' stock in a given nation
Q12	Shipping Modes and Order Priority Query	Determines whether selecting less expensive modes of shipping is negatively affecting the critical-priority Orders by causing more parts to be received by customers after the committed date
Q13	Customer Distribution Query	Seeks relationships between customers and the size of their orders
Q14	Promotion Effect Query	Monitors the market response to a promotion such as TV advertisements or a special campaign
Q15	Top Supplier Query	Determines the top supplier so it can be rewarded, given more business, or identified for special recognition
Q16	Parts/Supplier Relationship Query	Finds out how many suppliers can supply parts with given attributes
Q17	Small-Quantity-Order Revenue Query	Determines how much average yearly revenue would be lost if orders were no longer filled for small quantities of certain parts
Q18	Large Volume Customer Query	Ranks customers based on their having placed a large quantity order
Q19	Discounted Revenue Query	Reports the gross discounted revenue attributed to the sale of selected parts handled in a particular manner
Q20	Potential Part Promotion Query	Identifies suppliers in a particular nation having selected parts that may be candidates for a promotional offer
Q21	Suppliers Who Kept Orders Waiting Query	Identifies certain suppliers who were not able to ship required parts in a timely manner
Q22	Global Sales Opportunity Query	Identifies geographies where there are customers who may be likely to make a purchase

B.3 Execution Rules and Performance Metrics

The TPC-H operations are modeled as follows:

- The database is continuously available 24 hours a day, 7 days a week, for ad-hoc queries from multiple end users and data modifications against all tables, except possibly during infrequent (e.g., once a month) maintenance sessions.
- The TPC-H database tracks, possibly with some delay, the state of the OLAP database through on-going refresh functions which batch together a number of modifications impacting some part of the decision support database.
- To achieve the optimal compromise between performance and operational requirements, the database administrator can set, once and for all, the locking levels and the concurrent scheduling rules for queries and refresh functions.

To measure the performance of a system using the TPC-H Benchmark, the test sponsor will execute runs composed of:

- **A power tes**t, to measure the raw query execution power of the system when connected with a single active user. In this test, a single pair of refresh functions are executed exclusively by a separate refresh stream and scheduled before and after the execution of the queries.
- **A throughput tes**t, to measure the ability of the system to process the most queries in the least amount of time. In this test, several pairs of refresh functions are executed exclusively by a separate refresh stream, and scheduled as defined by the test sponsor.

Benchmark results are highly dependent upon workload, specific application requirements, and systems design and implementation. Relative system performance will vary as a result of these and other factors. Therefore, TPC-H should not be used as a substitute for a specific customer application benchmarking when critical capacity planning and/or product evaluation decisions are contemplated. The results described in this thesis are not audited and are not to be interpreted as representative of the system under study.

Glossary of Acronyms

DB2/UDB	DB2 Universal Database
DBA	Database Administrator
DSS	Decision Support System
DBMS	Database Management System
MVA	Mean Value Analysis
OLAP	Online Analytical Processing
OLTP	Online Transaction Processing
QNM	Queueing Network Model
RR	Random Read
RRT	Random Read Time
SQL	Structured Query Language
SR	Sequential Read
SRT	Sequential Read Time
TPC	Transaction Processing Performance Council
TPC-C	Transaction Processing Performance Council Benchmark C
TPC-D	Transaction Processing Performance Council Benchmark D
TPC-H	Transaction Processing Performance Council Benchmark H
TPC-R	Transaction Processing Performance Council Benchmark R

Vita

Name: Xilin Cui

Education: M. Sc. in Computing and Information Science

Queen's University, Kingston, Ontario, Canada, 2001-2003

B. Sc. in Computing and Information Science

Queen's University, Kingston, Ontario, Canada, 2000-2001

B. Sc. in Chemistry

Nankai University, Tianjin, P. R. China, 1987-1991

Experience: Research Assistant, Computing and Information Science

Queen's University, 2001-2003

Teaching Assistant, Computing and Information Science

Queen's University, 2001-2003

Awards: Nortel Networks upper year Awards, 2000-2001

Queen's Graduate Awards, 2001-2002

Ontario Graduate Scholarship, 2002-2003

Publication: X. Cui, P. Martin, and W. Powley. "A Study of Capacity Planning for Database Management Systems with OLAP Workloads", *Proceeding of Computer Measurement Group's 2003 International Conference*, December 2003.