Journal of Parallel and Distributed Computing **61**, 609–640 (2001)
doi:10.1006/jpdc.2000.1710, available online at http://www.idealibrary.com on IDEAL®

Architectural Effects of Symmetric Multiprocessors on TPC-C Commercial Workload[1]

Xing Du

Software and Systems Development Lab, Hewlett-Packard Company, Bellevue, Washington 98007

Xiaodong Zhang

Department of Computer Science, College of William and Mary, Williamsburg, Virginia 23187

Yingfei Dong

Department of Computer Science, University of Minnesota, Minneapolis, Minnesota 55455

and

Lin Zhang

LXB World Marketing (Canada), Vancouver, British Columbia, Canada V5T 1P5

Received November 1, 1997; revised November 1, 1998; accepted November 14, 2000

Commercial transaction processing applications are an important workload running on symmetric multiprocessor systems (SMPs). They differ dramatically from scientific, numeric-intensive, and engineering applications because they are I/O bound, and they contain more system software activities. Most SMP servers available in the market have been designed and optimized for scientific and engineering workloads. A major challenge of studying architectural effects on the performance of a commercial workload is the lack of easy access to large-scale and complex database engines running on a multiprocessor system with powerful I/O facilities. Experiments involving case studies have been shown to be highly time-consuming and expensive. In this paper, we investigate the feasibility of using queuing network models with the support of simulation to study the SMP architectural impacts on the performance of commercial workloads. We use the commercial benchmark TPC-C as the workload. A bus-based SMP machine is used as the target platform. Queueing network modeling is employed to characterize the TPC-C workload on the SMP. The system components such as processors, memory, the

[1] This work is supported in part by the National Science Foundation under Grants CCR-9400719 and CCR-9812187, by the Air Force Office of Scientific Research under Grant AFOSR-95-1-0215, and by the Office of Naval Research under Grant ONR-95-1-1239. The work was done when Xing Du was with the Department of Computer Science, College of William and Mary.

0743-7315/01 $35.00

memory bus, I/O buses, and disks are modeled as service centers, and their effects on performance are analyzed. Simulations are conducted as well to collect the workload-specific parameters (model parameterization) and to verify the accuracy of the model. Our studies find that among disk-related parameters, the disk rotation latency affects the performance of TPC-C most significantly. Among I/O buses and number of disks, the number of I/O buses has the deepest impact on performance. This study also demonstrates that our modeling approach is feasible, cost-effective, and accurate for evaluating the performance of commercial workloads on SMPs, and it is complementary to the measurement-based experimental approaches.

Key Words: commercial workload; performance evaluation; queueing network model; symmetric multiprocessor (SMP); TPC-C.

1. INTRODUCTION

Symmetric multiprocessor systems (SMPs) have become a standard parallel processing platform for various applications. One important usage of such systems is the execution of commercial workloads, which represent one of the most rapidly growing market segments. Commercial workload applications range from airline reservation systems to wholesale systems. Multiprocessor systems are used to increase the throughput of the commercial workload and to reduce the response times for each client.

Improving the performance of commercial workloads on SMPs is a hot research topic interesting to both academia and industry. Commercial applications differ dramatically from scientific, numeric-intensive, and engineering applications. They contain more sophisticated system software activities and use relatively slow I/O devices. Performance of commercial workloads is determined by so many factors (both hardware and software) that it is hard and time-consuming to evaluate. Based on an intensive experimental study conducted at Western Research Lab at DEC, Barroso *et al.* [1] recently summarized the challenges and difficulties of the performance evaluation of commercial workloads—the lack of experimental resources. It is difficult and expensive to have a large-scale and complex database engine running on a multiprocessor system with powerful I/O facilities for performance evaluation. In addition, the experiments for case studies have been shown to be highly time-consuming.

If performance models are sufficiently accurate for the evaluation of commercial workloads, it is certainly cost-effective and complementary to the measurement-based experiments. In this paper, we investigate the feasibility of using queuing network models with the support of simulation to study the SMP architectural impacts on performance of commercial workloads. We characterize commercial workloads and model their performance on SMPs. Using our model, we further quantitatively evaluate the architectural impacts of SMPs on the performance of commercial workloads. In detail, we use the TPC Benchmark C (TPC-C) [18], a standard commercial workload benchmark, as the workload. A bus-based SMP system with modern disk drives is used as the target machine. A queueing network model is employed to evaluate performance of the TPC-C workload running on

such a machine. The model is verified by program-driven simulation and is found to be accurate. We quantitatively analyze the impacts of processors, memory, and disks using the TPC-C metric *tpmC* (transaction-per-minute) as well. Experimental and modeling results show that among disk-related parameters, the disk rotation latency affects performance of TPC-C most significantly. Among I/O buses and number of disks, the number of I/O buses has the most significant impact on performance. Our study also demonstrates that queueing network modeling is a feasible, cost-effective, and accurate way to evaluate the performance of commercial workloads on SMPs, and is complementary to the measurement-based experimental approaches.

The rest of the paper is organized as follows. Section 2 gives an overview of the workload benchmark TPC-C and the architecture of SMP. In Section 3, we use the queueing network model to characterize the behavior of TPC-C on an SMP. Section 4 discusses the model parameterization and simulation environments. Architectural impacts on the performance of TPC-C are quantitatively evaluated in Section 5. Section 6 discusses related work. We give our conclusions and future work in the Section 7.

2. TPC-C WORKLOAD AND SMP ARCHITECTURE

2.1. TPC-C Overview

The Transaction Processing Performance Council (TPC) is a nonprofit organization founded to define commercial workload benchmarks and to disseminate its benchmark performance data to the industry. Its benchmarks are widely used by computer manufacturers and database providers to test, evaluate, and demonstrate the performance of their products. Currently, there are two benchmarks in the TPC benchmark suite: TPC-C and TPC-D, which represent two major categories of commercial applications: those supporting business operations and those supporting business analysis. TPC-C is an on-line transaction processing (OLTP) benchmark. It is a mixture of read-only and update intensive transactions that simulate a complete computing environment where a population of terminal operators executes transactions against a database. TPC-D is a benchmark which represents a broad range of decision support applications that require complex, long running queries against large complex databases. Examples of such applications are data mining and predictive data modeling. TPC-C and TPC-D represent completely different application domains. This paper focuses on investigating the performance of TPC-C applications.

Benchmark TPC-C contains representative transactions of an industry which must manage, sell, or distribute a product or service. Specifically, TPC-C simulates a wholesale company with a number of geographically distributed warehouses and their sale districts. Customers call the company to place a new order or request the status of an existing order. Orders are composed of 10 order lines (ordering 10 items at one time on the average). One percent of all order lines are for items which are not in stock at the regional warehouse and must be supplied by another warehouse. The system is also used to enter payments from customers, process orders for delivery, and to examine stock levels to identify potential supply shortages.

TABLE 1

Database Tables and Transactions Statistics of TPC-C

Database table		Transaction				
Name	Size	New-Order (I)	Payment (II)	Order-Status (III)	Delivery (IV)	Stock-Level (V)
Warehouse (1)	W	J (0.02)	U, S (0.19)			
District (2)	W*10	S, U (0.04)	U, S (0.19)			S (0.33)
Stock (3)	W*100k	S, U (0.45)				J (0.33)
Item (4)	100k	S (0.21)				
Customer (5)	W*30k	J (0.02)	S, U (0.53)	S (0.78)	U (0.16)	
Orders (6)	W*30k +	I (0.02)		S (0.11)	S, U (0.28)	
New-Order (7)	W*9k +	I (0.02)			S, D (0.28)	
Order-Line (8)	W*300k +	I (0.21)		S (0.11)	U, S (0.28)	J (0.33)
History (9)	W*30k +	I (0.09)				

Note. W is the number of warehouses, $k = 1000$, S, I, J, U, D represent the database queries *select*, *insert*, *join*, *update*, and *delete*, respectively). The number in each transaction type column is the probability for that type of transaction to access a database table.

Technically, the benchmark is composed of nine individual database tables: Warehouse, District, Customer, Order, New-Order, Order-Line, Stock, Item, and History. We represent them in Tables 1 to 9, respectively. Five types of transactions are defined, which operate on these tables. They are New-Order, Payment, Order-Status, Delivery, and Stock-Level. We use Roman numerals I to V to denote them, respectively. To implement the transactions, the following database queries are used: `select`, `insert`, `update`, `join`, and `delete`. We summarize the database tables, the transactions, and how the transactions operate on database tables in Table 1, where W is the number of warehouses and k is 1000. The probability for a transaction type to access a database table is listed in parentheses as well. For example, the probability for transaction I (New-Order) using `select`(S) and `update` (U) operations to access database Table 3 (Stock) is 0.45, which means 45% of the database operations are performed on database Table 3.

The transactions are generated using emulated users. An emulated user selects a transaction type, inputs a transaction of that type (keying), waits for the output of the transaction, and thinks after getting the output on the screen. This procedure is shown in Fig. 1.

Transaction selection occurs at random but is based on a minimum mixed percentage for each transaction type. Actually, the TPC-C benchmark requires an emulated user to execute at least one III, one IV, and one V transaction for every 10 I and 10 II type transactions. The minimum mixed percentage is listed in Table 2. There is no minimum for the New-Order transaction as its measured rate is used as the reported throughput. We will discuss it later. For each transaction type, the think time is taken independently from a negative exponential distribution and is computed from

$$T_{think} = ln(r) \times T_{mean-think},$$

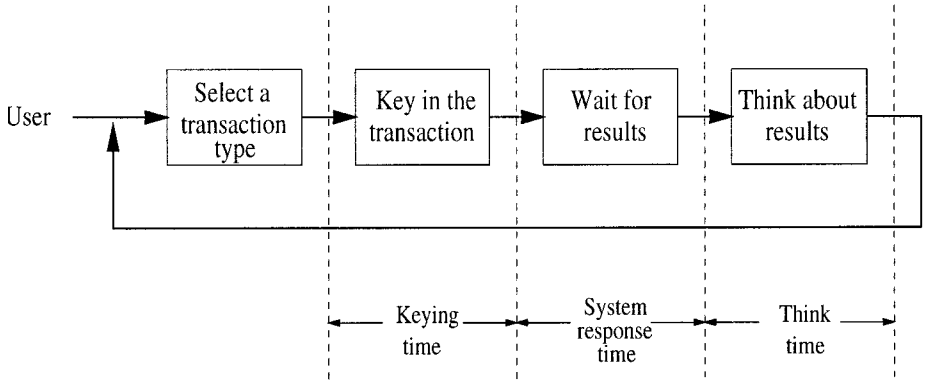

FIG. 1. The four steps taken by an emulated user to execute a transaction.

where T_{think} is the think time, r is a random number uniformly distributed between 0 and 1, and $T_{mean-think}$ is the mean think time. The minimum mean think time and minimum keying time for different transaction types are defined in the benchmark as well. Table 2 lists these requirements.

Multiple emulated users may generate transactions concurrently to work against the same database through a client-server mode. The TPC-C performance metric measures the total number of transactions of type I (New-Order) completed per minute in the server. It is expressed and reported in the unit of transactions-per-minute C (*tpmC*).

2.2. SMP Architecture

Bus-based symmetric multiprocessors (SMPs) are by far the most popular server products for commercial applications. One typical SMP architecture is shown in Fig. 2. Generally speaking, an SMP consists of several identical processors. Each processor has its own cache and is connected to the shared main memory by a memory bus. The number of processors varies from 2 to 8 for small-scale SMPs.

TABLE 2

Transaction Mix and Minimum Time Constraints (Adopted from the TPC-C Specification)

Transaction type	Minimum mixed percentage	Minimum keying time (s)	Minimum mean think time (s)
New-Order (I)	n/a	18	12
Payment (II)	43.0	3	12
Order-Status (III)	4.0	2	10
Delivery (IV)	4.0	2	5
Stock-Level (V)	4.0	2	5

Note. The minimum mixed percentage for New-Order is not defined (n/a) because New-Order is used as the reported throughput.

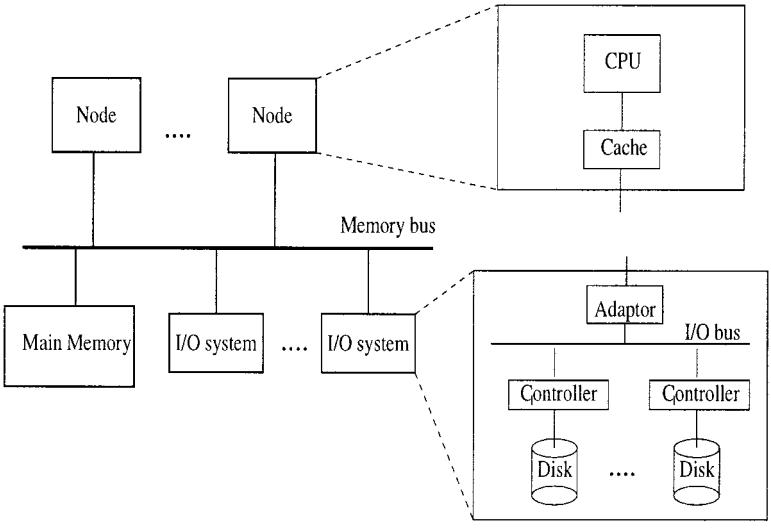

FIG. 2. The SMP architecture consists of a couple of CPU/cache pairs which are interconnected with the main memory and with I/O systems through a shared memory bus.

For multiprocessors with a small number of processors and large caches, the bus and the single memory can only satisfy a limited memory demand from each processor. I/O buses such as PCI or SCSI are used to connect I/O devices. An adaptor links the memory bus and an I/O bus. The memory bus is used both by processors when they access memory and by I/O devices when they transfer data between memory and I/O devices [5].

Disks are the I/O devices we consider in the paper. To improve performance, modern disks usually release I/O buses when they are waiting for the disk heads to reach the expected disk sectors. Specifically, for such a disk drive to read data, when the disk is searching for the required data on a specific track and sector, it releases its I/O bus for other disks on the same bus to use. When the disk head reaches its location, it reads the data from the disk to the disk cache. After that, the disk tries to regain the I/O bus and then transfer the data from the disk cache to memory. Contention may occur when regaining the I/O bus. The release-and-regain of I/O buses increases the average disk access time for an individual read because of the release-and-regain overhead and I/O bus contention. However, it improves I/O bus utilization by the use of concurrency. For write operations, some disk drives support "immediate-reporting," which means that as soon as the data is written to the disk cache, it is reported as complete. But because of volatile write-cache problems, we do not consider such drives in the study. We assume all writes are "not immediate-reported" and are completed only when the data are written to the disk media, not just the write cache. Ruemmler and Wilkes [13] give a detailed description of these typical disk drives. To simplify the discussion, we further assume that all disks are of the same type and of the same size.

This is a typical, noncustomized SMP architecture, which is often used as departmental servers to run medium-size, transaction-based applications represented by the benchmark TPC-C.

3. ANALYTICAL MODEL

Analytical modeling provides a quick and approximate performance analysis and evaluation in comparison with simulation and measurements. The queueing network model [9] is a simple and efficient way to analyze system performance. It has been widely used in various applications (e.g., [7, 17, 20]). In this section, we derive a queueing network model for the SMP architecture and use the exact solution for closed queueing network models, the Mean Value Analysis (MVA), to compute the performance of workload TPC-C on SMPs.

3.1. Queueing Network Models

The queueing network model is used to characterize a computer system as a network of queues to be evaluated analytically. A network of queues is a collection of service centers, which represent system resources, and customers, such as users or transactions. Analytical evaluation is performed by using numerical methods to efficiently solve a set of equations induced by the network of queues and its parameters.

Based on the number of service centers involved, a model system can be classified as either a single service center system or a multiple service centers system. Two important parameters regarding a service center are the workload intensity and the service demand (the average service requirement of a customer, sometimes called the service time). For specific parameter values, we may obtain performance measures such as utilization, residence time, queue length, and throughput by solving simple equations derived from a set of fundamental laws and rules such as Little's Law and the Forced Flow Law [9].

The workload intensity may be classified into two major types: transaction workload and terminal workload. The transaction workload has its intensity specified by a parameter λ, indicating the rate at which requests (customers) arrive. It has a population (customers) that varies over time. The terminal workload has its intensity specified by two parameters: N, indicating the number of terminals (customers), and Z, indicating the average time for customers to use terminals between interactions ("think" time). Models with transaction workload are sometimes referred to as *open models*, since there is an infinite stream of arriving customers. Meanwhile, models with terminal workload are referred to as *closed models*, since a fixed number of customers repeatedly request services. Algorithms used to evaluate open models differ from those used for closed models. In addition, models can be classified as either *single class models* or *multiple class models* depending on whether or not customers are indistinguishable from one another.

Queueing network models have become important tools in the design and analysis of computer systems. For many applications, it achieves sufficient accuracy at relatively low cost. Detailed descriptions of queueing network models can be found in the literature, such as [9].

3.2. Modeling TPC Benchmark on SMPs

The TPC-C benchmark is a terminal workload where a fixed number of customers (emulated users) request services, think about results, and start another transaction

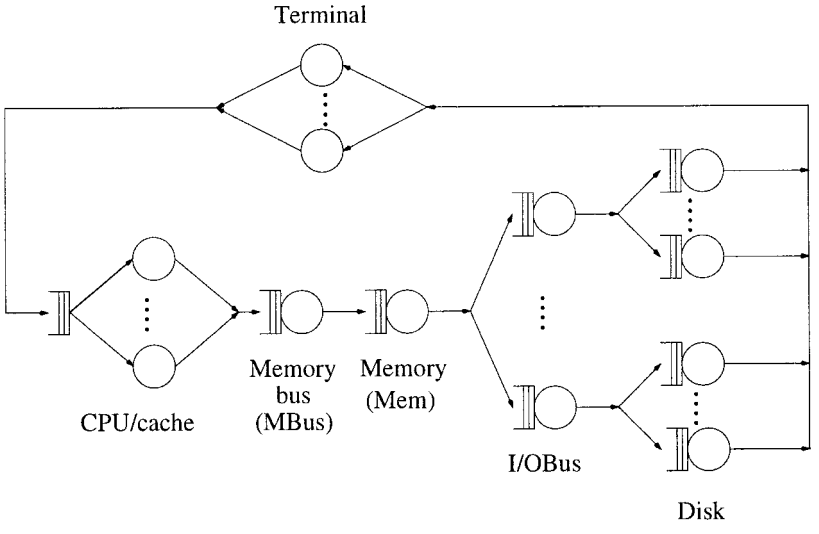

FIG. 3. A single class, closed model of the TPC-C workload on an SMP system.

again.[2] This process is repeated. The customers are indistinguishable, and have the same type of behavior. So the TPC-C workload is considered as a single class terminal workload. In principle, an SMP can be viewed as a service center with several terminals requesting services from it. In practice, an SMP is composed of multiple subordinate service centers as shown in Fig. 3. The service centers that we are interested in are CPUs, the memory bus, the memory, I/O buses, and disks. We include the cache along with its corresponding CPU and do not consider its effect separately for two reasons. First, there has been literature available on investigating cache behavior of commercial workloads (e.g., [3, 11]). This paper focuses on other architectural impacts. Second, compared with the average memory access latency and disk access latency, the average cache access latency is extremely small. In terms of accuracy and efficiency, it is reasonable not to characterize it individually in the model. However, we include its effects in CPU centers.

The model notations are given in Table 3.

Based on the TPC-C specification, the think time, Z_i, of a TPC-C transaction type i ($i = I, ..., V$), is composed of menu response time, keying time, and transaction think time. Usually, the minimum values for these times are used in order to maximize the *tpmC*. Thus, we have

$$Z_i = min(menu\ response\ time_i) + min(keying\ time_i) + min(transaction\ think\ time_i).$$

So the average (weighted mean) think time for a terminal, which is a mix of five transaction types, is

$$Z = \sum_{i=I}^{V} Z_i P_i.$$

[2] *Note:* the term *transaction* used in the TPC-C context is different from that used in queueing network models.

TABLE 3

Model Notations

r	The service center: CPUs, the memory bus, the memory, an I/O bus, or a disk
$(IOBus, i)$	the j th I/O bus.
$(DISK, i, j)$	the j th disk on the i th I/O bus.
L	the number of I/O buses.
K	the number of disks per I/O bus.
N	the number of terminals.
Z	the average think time for a terminal.
R	the average response time of the system.
X	the throughput of the system.
D_r	the service time at center r.
Q_r	the waiting time at center r.
R_r	the response time at center r. We have $R_r = D_r + Q_r$.
U_r	the utilization of the resource r.
P_i	the probability of a transaction of type i ($i = \mathrm{I}, ..., \mathrm{V}$).
V_r	the number of visits to service center r.
S_r	the average service time of service center r per visit.

Note. Regarding K, the number of disks per I/O bus, we assume the number of disks on each I/O bus is equal (symmetric). The total number of disks are $K \times L$.

According to the TPC-C specification, at least 90% of all menu selections must have a menu response time of less than 2 s. Thus, we set $min(mean\ response\ time_i) = 0$. Using the above formula and the values in Table 2, we have 9.63 s for the minimum keying time, and $-11.36ln(r)$ s for the minimum transaction think time, where r is a random variable uniformly distributed between 0 and 1 ($r \in [0, 1]$). Thus, we have

$$Z = 9.63 - 11.36 \times ln(r). \tag{3.1}$$

The five transactions have different database operations working on different database tables. Transaction I (New-Order) consists of entering a complete order through a single database transaction. It is the metric of the benchmark. It operates on eight database tables using standard data management functions such as `join`, `select`, `insert`, and `update`. We use $J(t1, t2)$ to denote the operation to join two tables $t1$, and $t2$, $S(t)$, $I(t)$, $U(t)$, and $D(t)$ to denote to select a tuple from, insert a tuple to, update a tuple in, and delete a tuple from table t, respectively. Here t, $t1$, and $t2$ are generic terms for database tables numbered from 1 to 9. Its operations are listed in Table 4, where `o_ol_cnt` is the item number in an order which varies from 5 to 15 and has the average value of 10.

Transaction II (Payment) supports two ways to input a payer (customer): the customer identifier number or the customer's last name. Assume there are `name_cnt` customers having the same last name. When the customer is selected by his/her last name, on average, it will take `name_cnt/2` times `select` operations (retrieve and check them one by one) to get the exact customer. In addition, if the customer's credit type is "BC" (Bad Credit) rather than "GC" (Good Credit), some

TABLE 4

The Database Operations of the Five Transaction Types

Transaction	Database operations
New-Order (I)	$J(5,1) + S(2) + U(2) + I(6) + I(7) + o\_ol\_cnt \times (S(4) + S(3) + U(3) + I(8))$
Payment (II)	$U(1) + S(1) + U(2) + S(2) + P_{name} \times (S(5) + name\_cnt/2 \times S(5)) + P_{id} \times S(5) + P_{BC} \times (S(5) + U(5)) + P_{GC} \times U(5) + I(9)$
Order-Status (III)	$P_{name} \times (S(5) + name\_cnt/2 \times S(5)) + P_{id} \times S(5) + S(6) + S(8)$
Delivery (VI)	$dist\_per\_ware \times (S(7) + D(7) + S(6) + U(6) + U(8) + S(8) + U(5))$
Stock-Level (V)	$S(2) + J(8,3)$

Note. J, S, U, I, D stands for database operations *join*, *select*, *update*, *insert*, and *delete*, respectively. We use numbers 1 through 9 to denote each of nine database tables, respectively.

additional information must be retrieved and appended to database Table 5. The second line in Table 4 gives the average operations where P_{name} and P_{id} are probabilities of customers selected by "name" and by "id", respectively, and P_{BC} and P_{GC} are probabilities of a customer to have the credit type of "BC" and "GC" respectively.

The III transaction (Order-Status) queries the status of a customer's last order. It is a read-only database transaction. As with transaction II, the customer can be selected either by his/her last name or by his/her identifier number.

The IV transaction (Delivery) processes new orders which have not been delivered yet for each district within the same warehouse. The number of districts in a warehouse is denoted by `dist_per_ware`. The fourth line in the table gives the operations of this transaction, which consists of both read and write on databases.

The V transaction (Stock-Level) determines the number of recently sold items that have a stock level below a specified threshold. It is a read-only transaction and only works against three database tables.

The average distribution for a database access to a database table is the ratio of the number of accesses of the table to the number of accesses of all tables during a specific time interval. It is computed based on the transaction type distribution in Table 2 and the access patterns to database tables for each transaction type defined in Table 4.

The service centers used by a TPC-C transaction in an SMP include CPUs, the memory, the memory bus, and I/O systems. Thus, the average response time (R) of an SMP for one transaction can be estimated as

$$R = R_{CPU} + R_{MBus} + R_{Mem} + R_{I/O}. \tag{3.2}$$

Multiple CPUs form a special load dependent service center whose service time is not constant but depends on the number of requests queued at the center. Suppose there are two CPUs and they are identical. A request needs time t for a CPU to process. If there is no request in the waiting queue, the service time for the request is t. However, when there are one or more independent requests in the queue, the average service

time[3] becomes $t/2$. This kind of center is called a flow equivalent service center (FESC). We model the multiple CPUs and their caches as an FESC. Thus the average response time for the center to process a TPC-C transaction is determined not only by the number of requests in the queue but also by the probability of that number of requests in the queue. Therefore, the average response time of the CPU center for a transaction is

$$R_{CPU} = V_{CPU} \sum_{j=1}^{n} \frac{j}{\mu(j)} P(j-1 \mid n-1), \tag{3.3}$$

where V_{CPU} is the number of visits to the CPU center per transaction, $\mu(j)$ is the average service rate (per cycle) when there are j transactions, and $P(j \mid n)$ is the probability of j transactions presenting in the queue when the number of transactions in the model is n. It varies from 0 to N when applying the MVA algorithm to iteratively compute the throughput of the whole system. $P(j \mid n)$ can also be regarded as the queue length distribution for the CPU center and is computed by

$$P(j \mid n) = \begin{cases} \dfrac{X(n)}{\mu(j)} P(j-1 \mid n-1) & j = 1, \ldots, n \\ 1 - \displaystyle\sum_{i=1}^{n} P(i \mid n) & j = 0. \end{cases} \tag{3.4}$$

All other service centers in the model are load independent. Their response times can be computed based on the service times. The service times of the memory bus (MBus), the memory (Mem), and I/O buses (IOBus) for a TPC-C transaction can be established in terms of the number of visits (V) per transaction and the average service time per visit (S).

To effectively utilize the bandwidth of memory buses, modern SMPs usually employ the split-transaction protocol, which splits a memory access activity into two components: a memory request component and a reply component to allow the memory bus to be released for other uses between the request and reply stages [6]. Consequently, the service time of the memory bus, D_{MBus}, for a transaction consists of two parts: time for requests and time for replies. Thus, we have

$$D_{MBus} = V_{Mbus} \times (S_{MBusRequest} + S_{MBusReply}),$$

where V_{MBus} is the number of memory bus visits per transaction, $S_{MBusRequest}$ and $S_{MBusReply}$ are the bus service times for request and reply stages, respectively. In addition, we know that the number of memory bus visits equals to the sum of the

[3] In real applications, requests are, to some extent, dependent on each other because of factors such as synchronization. So it results in a higher average service time, and a slightly complicated computation as we will discuss below.

number of CPUs' visits to the memory ($V_{CPU-Mem}$) and the number of disks' visits to the memory ($V_{Disk_{M}em}$). Therefore, the above service time is rewritten as

$$D_{M\,Bus} = (V_{CPU-Mem} + V_{Disk-Mem}) \times (S_{M\,BusRequest} + S_{M\,BusReply}). \tag{3.5}$$

Assume $S_{M\,Bus}$ is the average service time per visit for a non-split-transaction bus. We then have

$$S_{M\,Bus} > S_{M\,BusRequest} + S_{M\,BusReply},$$

so the memory bus service time is reduced by using the split-transaction bus.

However, the split-transaction bus protocol increases the average memory access latency because the memory bus must be acquired twice and contention for the bus exists for the second time the bus is acquired. As with the memory bus, the memory is accessed by visits from both CPU and disk operations. Logically, the data transferred between the memory and disks are in the unit of a disk block (page), whose size is an integer factor of the data size transferred between the memory and CPUs (cache line). Physically, the transfer of a disk block is accomplished by transferring a sequence of cache-line-size sub-blocks because of the physical features of the memory modules and the width of the data line of the memory bus. Assume V_{Disk} is the number of visits to disks for a transaction, clearly we have

$$V_{Disk-Mem} = \frac{Disk\ block\ size}{Cache\ block\ size} \times V_{Disk}.$$

The memory service time for a request from disks for a cache-line-size subblock is the same as the memory service time for a request of the same data size from CPUs. Consequently, the service time of the memory for a transaction is calculated as

$$\begin{aligned} D_{Mem} &= V_{Mem} \times (S_{Mem} + Contention_{M\,Bus}) \\ &= (V_{CPU-Mem} + V_{Disk-Mem}) \times (S_{Mem} + Contention_{M\,Bus}), \end{aligned}$$

where V_{Mem} is the number of visits to the memory in a transaction, which includes the number of memory visits from CPUs ($V_{CPU-Mem}$) and the number of memory visits from disks ($V_{Disk-Mem}$), and S_{Mem} is the memory service time per visit. $Contention_{M\,Bus}$ is the waiting time for the memory bus because of contention. According to the principles of the mean value analysis, the contention time for the memory bus can be approximated by the product of the memory bus service time for a request and the number of bus requests from other sources encountered by this request. The weighted mean of the memory bus service time is

$$\frac{S_{M\,BusRequest} + S_{M\,BusReply}}{2}.$$

The latter is determined by

$$\frac{U_{M\,Bus} - U_{M\,Bus}(Mem)}{1 - U_{M\,Bus}}$$

where $U_{M\,Bus}$ is the utilization of the memory bus and $U_{M\,Bus}(Mem)$ is the utilization of the memory bus caused by accesses from memory (not disks). Combining the above equations, we have the equation for the average memory service time:

$$D_{Mem} = (V_{CPU-Mem} + V_{Disk-Mem}) \times \left(S_{Mem} + \frac{S_{M\,BusRequest} + S_{M\,BusReply}}{2} \times \frac{U_{M\,Bus} - U_{M\,Bus}(Mem)}{1 - U_{M\,Bus}} \right). \quad (3.6)$$

Clearly, the split-transaction bus requires the memory system to be sophisticated enough to handle multiple overlapping memory access activities. Multiple memory modules should be available for concurrent accesses. The use of multiple memory modules will decrease the average service time (S_{Mem}). Since the number of memory accesses incurred by TPC-C transactions are much larger than the number of memory modules available, we assume the memory access distribution among memory modules is uniform and independent. Thus, the effect of multiple memory modules can be modeled by adjusting the memory service time (S_{Mem}) based on the number of memory modules.

I/O buses are exclusively used by disk accesses. The service time for the ith bus is computed as

$$D_{IO\,Bus,\,i} = V_{IO\,Bus,\,i} \times S_{IO\,Bus},$$

where $V_{IO\,Bus,\,i}$ is the number of visits to the ith I/O bus per TPC-C transaction, and $S_{IO\,Bus}$ is the service time per visit (we assume that all I/O buses have the same service time per visit). However, the number of visits to an I/O bus equals to the sum of the numbers of visits caused by visits to all disks attached to that I/O bus, so we rewrite the above equation as

$$D_{IO\,Bus,\,i} = \sum_{j=1}^{K} V_{Disk,\,i,\,j} \times S_{IO\,Bus},$$

where $V_{Disk,\,i,\,j}$ is the number of visits per transaction to the jth disk on the ith I/O bus. However, as we discussed before, the read and write operations work differently. The read operation's I/O bus service time is

$$S_{IO\,Bus\,Req} + S_{IO\,Bus\,Rep},$$

while the write operation's I/O bus service time is

$$S_{IO\,Bus} + S_{seek} + S_{latency} + S_{transfer}$$

because it holds the I/O bus when it seeks, waiting, and transferring data to disks. $S_{IO\,Bus\,Req}$ and $S_{IO\,Bus\,Rep}$ are the I/O bus service times for requesting a disk and replying to the data, respectively. So we modify the above equation to get

$$D_{IO\,Bus,\,i} = \sum_{j=1}^{K} (V^{read}_{Disk,\,i,\,j} \times (S_{IO\,Bus\,Req} + S_{IO\,Bus\,Rep}) + V^{write}_{Disk,\,i,\,j} \times (S_{IO\,Bus} + S_{seek} + S_{latency} + S_{transfer})), \tag{3.7}$$

where $V^{read}_{Disk,\,i,\,j}$ and $V^{write}_{Disk,\,i,\,j}$ are the number of read and write operations on disk $(Disk, i, j)$, respectively.

To access a block in a disk, the disk head first moves to the cylinder where that block is located (the *seek* operation), it then waits there for the block to arrive, and finally, when the block is rotated under the disk head, it accesses it (the *transfer* operation). We use S_{seek}, $S_{latency}$, and $S_{transfer}$ to denote the time spent on the above three steps, respectively. Based on the discussion of disk drives in Section 2.2, the disk read and write operations work differently. In addition to seeking, waiting, and transferring, a read operation spends some time waiting for its I/O bus (to regain it) after it has finished the transfer of data from disk to disk cache. Consequently, the average service time for read operations is

$$D^{read}_{Disk,\,i,\,j} = V^{read}_{Disk,\,i,\,j}(S_{seek} + S_{latency} + S_{transfer} + S_{Dcache} + Contention_{IO\,Bus,\,i}),$$

where $Contention_{IO\,Bus,\,i}$ is the waiting time for the I/O bus i. As with the computation of contention time for the memory bus, we have

$$Contention_{IO\,Bus,\,i} = (P_w S_{IO\,Bus} + P_r(S_{IO\,Bus\,Req} + S_{IO\,Bus\,Rep})) \times \frac{U_{IO\,Bus,\,i} - U_{IO\,Bus,\,i}(j)}{1 - U_{IO\,Bus,\,i}},$$

where P_w and P_r are the probabilities of I/O operations to be write and read, $U_{IO\,Bus,\,i}$ is the utilization of I/O bus i, and $U_{IO\,Bus,\,i}(j)$ is the utilization of I/O bus i caused by disk j on it.

In contrast, the write operation does not have disk cache and contention overhead, and its service time is

$$D^{write}_{Disk,\,i,\,j} = V^{write}_{Disk,\,i,\,j}(S_{seek} + S_{latency} + S_{transfer}).$$

Consequently, the average disk service time for the jth disk on the i th I/O bus per transaction is estimated by

$$D_{Disk,\,i,\,j} = D^{read}_{Disk,\,i,\,j} + D^{write}_{Disk,\,i,\,j}. \tag{3.8}$$

We have obtained the service time for each I/O bus and disk. However, because the disks can be running in parallel, it makes the computation of the whole service time for the I/O system a little complicated. Since a disk releases its I/O bus when it searches and transfers the required data between disk and disk cache, disks may

run in parallel. However, an I/O bus cannot be in operation in parallel with other I/O buses because operations of an I/O bus will involve the use of the memory bus, and in the system there is only one memory bus. They are performed sequentially. Consequently, the service time for the I/O system is

$$D_{I/O} = \sum_{i=1}^{L} D_{IO\,Bus,\,i} + D_{Disk}, \tag{3.9}$$

where D_{Disk} is the average response time for all disks for a transaction. For a transaction, its disk operations are performed sequentially because they are requested after the previous operation has finished even though some of them are performed on different disks and may be executed in parallel. When several transactions are executed concurrently, this parallelism is exploited. As a result, the average service time for the whole disk system is determined by the way in which disk operations are distributed among all disks. Clearly, we have

$$\max_{i=1,\,j=1}^{i=L,\,j=K} D_{Disk,\,i,\,j} \leqslant D_{Disk} \leqslant \sum_{i=1,\,j=1}^{i=L,\,j=K} D_{Disk,\,i,\,j}.$$

The approximate value for D_{Disk} is computed based on the probabilities (utilizations) of disks working in parallel and/or sequentially. The following expression gives an example about how to calculate the average disk service time for a two-disk system:

$$U_1 U_2 max(D_1, D_2) + U_1(1 - U_2) D_1 + U_2(1 - U_1) D_2.$$

The two disks have utilizations U_1 and U_2, and service times D_1 and D_2, respectively. A similar computation is employed to get the average disk service time for I/O systems consisting of more than two disks.

There are two kinds of mappings in the system: allocations of database tables to the available disks, and connections of the disks to the available I/O buses. Here are the notations for the two mappings f and F that define the data allocation and disk connections of the system:

$$f: \{1, 2, 3, 4, 5, 6, 7, 8, 9\} \to \{disk_1, ..., disk_{K \times L}\}$$
$$F: \{disk_1, ..., disk_{K \times L}\} \to \{IO\,Bus_1, ..., IO\,Bus_L\}.$$

Function f describes how the 9 database tables are allocated to $K \times L$ disks, and function F describes how $K \times L$ disks are connected to the L I/O buses. Based on those two mappings, the total number of accesses to disks, and the database access distribution table (Table 5), we compute the number of visits to each disk. The two mapping f and F affect the value of D_{Disk}. By changing the mappings f and F, we can evaluate disk configuration and data allocation effects on performance.

We apply the MVA algorithm [9] for the single class closed model. The computation is started by initiating both the terminal population n and the queue

TABLE 5

Access Distributions to the Nine Database Tables

Database table	1	2	3	4	5	6	7	8	9
Probability	0.0907	0.1129	0.2157	0.1108	0.2937	0.0246	0.0202	0.1278	0.0036

length Q to be 0. First, the response times for all service centers except for the CPU center are computed based on

$$R_i = D_i(1 + Q_i)$$

using Eqs. (3.5), (3.6), and (3.9), where i is $MBus$, Mem, and I/O respectively. The response time for the CPU center is given in Eq. (3.3). Second, Little's Law is applied

$$X = \frac{n}{Z + \sum R_i}$$

to compute the throughput for that number of terminals (n), where i is $CPUs$, $MBus$, Mem, and I/O, respectively. The waiting time for each center is computed as

$$Q_i = XR_i,$$

and the utilization of each center is obtained through

$$U_i = XD_i.$$

Next increase n by 1, and compute the response time for each center again. This process is repeated N times until finally we get the throughput X for a system consisting of N terminals. This is the throughput for the SMP running N TPC-C terminals. The average TPC-C benchmark metric $tpmC$ is computed as

$$tpmC = \sum_{1 \leqslant i \leqslant N} X_{i,I} = P_I \sum_{1 \leqslant i \leqslant N} X_i = P_I X, \tag{3.10}$$

where $X_{i,I}$ is the throughput of transaction I on terminal i, and X_i is the throughput of all types of transactions on terminal i. The probability of transactions of New-Order (P_I) comes from Table 2. Based on Eq. (3.10), we compute the measure of TPC-C benchmark ($tpmC$) on an SMP system.

4. PARAMETERIZATION AND MODEL VERIFICATION

4.1. Simulator

We built the SMP simulator for two purposes: (1) to run the TPC-C benchmark on it to collect the benchmark's parameters required by the model. Examples of such parameters are the numbers of visits to memory, disks, and others; (2) to

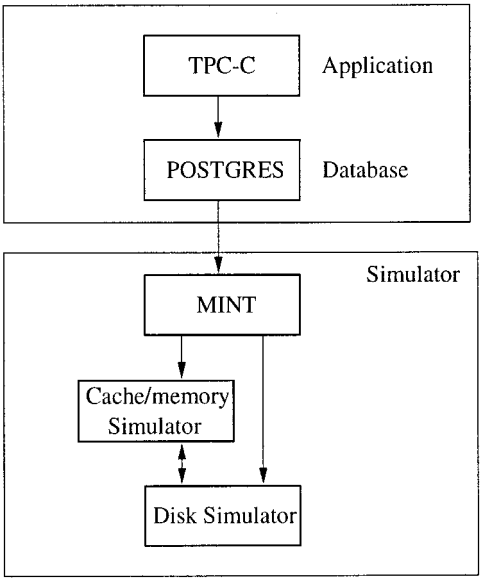

FIG. 4. The simulation environment consists of an SMP simulator, database system POSTGRES, and the TPC-C workload.

verify the model by comparing the simulation results with the model results. We used simulation rather than measurements to verify the model because of the lack of instrumenting and monitoring tools in a physical SMP machine to get the model parameters, and the financial difficulty to get a sufficient number of different SMP configurations.

The simulator we built consists of two parts: multiple processors and their memory system, and I/O buses and disks. In the model study, we include the effects of cache in the CPU center. In the simulator, for correctness, we simulate the cache behavior. In detail, a MINT-based [19] memory simulator was built to simulate the bus-based cache coherent cache/memory system. The disks and I/O buses are simulated using the disk simulator [8] implemented at Dartmouth. The database system running on the simulator was POSTGRES [14], which was developed by the University of California at Berkeley. We modified MINT, the disk simulator, and POSTGRES and integrated them. The work includes writing a back-end for MINT to simulate the memory system, attaching the disk simulator to MINT, modifying POSTGRES to collect disk access traces, and finally compiling POSTGRES using `non_shared` option to generate codes that can be run on the MINT-based simulator. We implemented the TPC-C benchmark based on the specification using C and POSTGRES's SQL. Without loss of generality, we scaled down the database size of TPC-C 100 times to reduce the simulation time. We also reduced the cache and memory sizes accordingly. The software structure is shown in Fig. 4.

4.2. Parameterization

First, the simulator was used to collect the TPC-C benchmark parameters which are required by the queueing network model. They are the number of visits to

TABLE 6

The Number of Visits to CPUs, the Memory, and Disks of the Five Types of TPC-C Transactions

Transaction	CPU	Memory read	Memory write	Disk read	Disk write
New order	22847137	4405731	3177453	150	570
Payment	13849985	2691023	1885840	116	38
Order status	8407681	1639657	1141375	105	0
Delivery	42757246	8469415	5874990	183	692
Stock level	7698052	1487017	1051811	99	0

service centers such as CPUs, the memory system, and disks. Table 6 gives these numbers of CPU accesses, the memory read/write number, and the disk read/write number for each TPC-C transaction type. We collected these values by running each transaction on an SMP with one processor, one I/O bus and one disk. Other parameters such as the numbers of visits to the memory bus and I/O buses can be derived directly from those values.

The values of architectural parameters of the model, most of which are the service times at service centers, are adopted from modern SMP systems and based on [4]. Table 7 lists the average service times in cycles for CPUs, memory, and disks. Other architectural parameters used by the simulator are listed in Table 8. The disk parameters are given based on the disk and I/O bus configuration values reported in [13].

4.3. Model Verification

Before we verified the model, we first collected the average effective service rates (per cycle) for CPU centers. The CPU centers that we are interested in are a one-CPU center, a two-CPU center, a three-CPU center, and a four-CPU center. We

TABLE 7

Average Service Times (in Cycles) for CPUs, the Memory, I/O Buses, and Disks Where $M = 10^6$

Name	Value	Description
CPI	1	S_{CPU}, Processor speed = 200 MHz
Memory latency	100	S_{Mem}
Memory bus request latency	2	$S_{MBusRequest}$
Memory bus reply latency	10	$S_{MBusReply}$
I/O bus latency	5120	S_{IOBus}, SCSI 2 bus
I/O bus request latency	120	$S_{IOBusReq}$
I/O bus reply latency	4850	$S_{IOBusRep}$
Disk cache latency	150	S_{Dcache}
Disk seek time	1 M	S_{seek} Average seek time
Disk rotation latency	1.5 M	$S_{latency}$, Average rotation latency
Disk transfer time	42000	$S_{transfer}$, Disk sector size = 256 bytes

TABLE 8

Architectural Parameters of an SMP System

Name	Value
Processor speed	200 MHz
Max. number of processors	4
Cache block size	32 bytes
Cache size	16 Kbytes
Memory size	256 Kbytes
Disk sector size	256 bytes
Disk cache size	128 Kbytes
Max. number of I/O buses (L)	2
Max. number of disks (K)	2

collected the effective service rates from the simulation. The simulator was set to simulate those four types of SMPs. Each system with a different number of CPU centers has one I/O bus and one disk. The effective service rate functions are shown in Fig. 5. The function values beyond 16 are very close to the values when the number of requests is 16, so we do not show those values in the figure. We use them in the model analysis of different I/O configurations.

We compare the modeling results with the simulation results on nine SMP configurations, where the number of terminals is 48 ($N = 48$). The nine configurations are given in Table 9.

The comparative model and simulation results are presented in Fig. 6. The model results of *tpmC* for all configurations are slightly larger than the simulation results. The possible reasons are: Our queueing network model does not precisely model

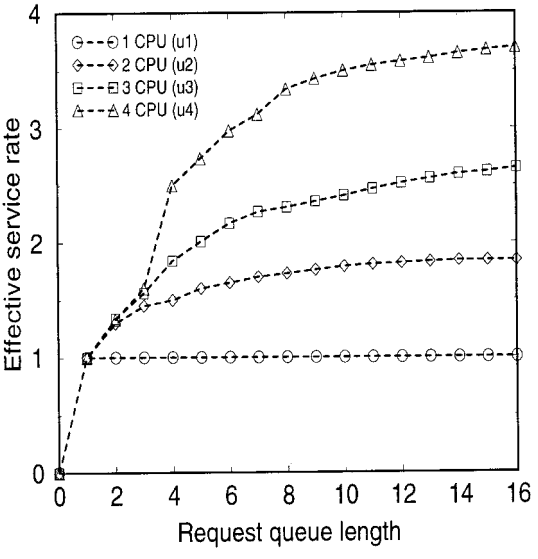

FIG. 5. Effective service rate functions (μ) for one-CPU, two-CPU, three-CPU, and four-CPU centers. This measures the number of services per cycle. The function values beyond 16 are very close to the values when the number of requests is 16. They are not included in the figure.

TABLE 9

SMP Configurations for Model Verification and Performance Analysis

Configuration name	SMP configuration			Note
	No. of CPUs	No. of I/O buses	No. of disks	
C1	1	1	1	base configuration
C2	2	1	1	
C3	2	1	2	
C4	2	2	2	1 disk on each I/O bus
C5	2	1	4	
C6	2	2	4	2 disks on each I/O bus
C7	4	1	2	
C8	4	1	4	
C9	4	2	4	2 disks on each I/O bus

the cache behavior, but the simulator does. In the model, we only apply a rule of thumb on the cache hit rate (90% of memory accesses are cache hits). Even though the actual cache hit rate may be larger than 90% in the simulation, the cache coherence overhead may affect performance negatively. Another contributing factor is synchronization. We model it through the effective service rates of the CPU centers, which are obtained from an SMP with one I/O bus and one disk. However the actual values for other SMP configurations may vary. The third factor is the mean value analysis itself. It is only a mean value modeling approach. But the quantitative difference between the model and simulation results is within a range of 11.2% and is acceptable. The POSTGRES we used for experiments is POSTGRES 4.2. We modified the POSTGRES in order to collect the I/O access profiles. Before each I/O calls, we added code to write the requested data addresses and their lengths to a trace file. One disadvantage of POSTGRES 4.2 is that it only supports

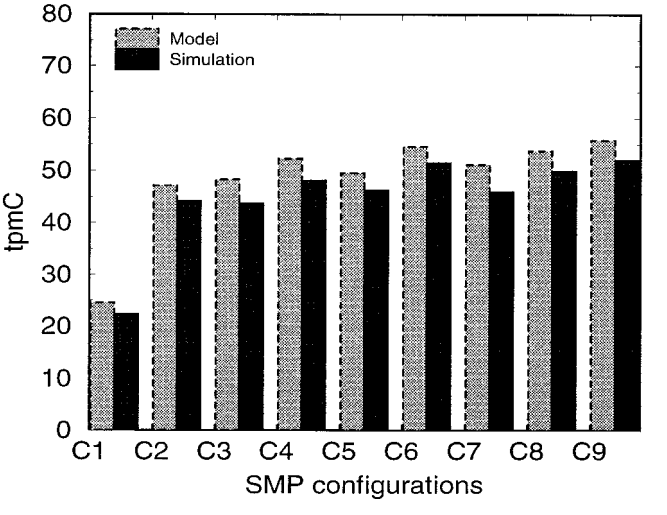

FIG. 6. Comparisons between model and simulation results for nine SMP configurations.

data locking at the level of tables, which limits the concurrency. (Newer releases of PostgresSQL 6.5 or above supports data locking at tuple levels.) We have tried to minimize the locking effect by reducing the TPC-C table sizes to a factor of 100, which makes the results with locking at the table level close to the results with locking at the tuple level. The model itself does not cover the synchronization overheads between transactions. This is another reason why the modeled results are somewhat better than the simulated results (see Fig. 6).

To verify whether the model and simulation results are close to a real system, we measured performance of TPC-C on a 4-CPU SMP machine. The machine is a SUN SPARCstation 20 with four 100 MHz hyperSPARC processors. It has one I/O bus and one disk. We set the architectural parameters of the model to the same values as the SPARCstation, and we compared the measurement results with the model results. The difference between them is close (15.3%). This indicates that the model results are sufficiently accurate to characterize the behaviors of SMP systems in terms of simulation and measurements.

5. PERFORMANCE ANALYSIS

In this section, we quantitatively analyze the architectural effects of SMPs on performance of TPC-C based on *tpmC*. The architectural factors we consider include the number of CPUs, the memory system latency and the I/O bus latency, the disk seek time, and the disk rotation latency. The data allocation effect is also investigated. The objectives of the analysis are to apply the model to provide insights into architectural impacts of TPC-C on performance and to find which architectural parameters affect the performance of a commercial workload most significantly. The evaluation results can be used as inputs to the design of future SMPs and other types of multiprocessors running commercial workloads.

To ease discussions and comparisons of performance of TPC-C on various SMP configurations, an SMP with one CPU, one I/O bus, and one disk is used as the baseline SMP configuration. Its architectural parameters are shown in Table 7. Performance of other SMPs are compared with that of the baseline SMP. The relative performance improvement is used as the metric throughout the analysis. It is defined as a ratio in the form of

$$\frac{tpmC(C) - tpmC(C1)}{tpmC(C1)} \times 100\%,$$

where C is any SMP configuration, and $C1$ is the baseline SMP.

5.1. Effects of Multiple Processors

We first analyze the effects of multiple processors on the performance of TPC-C. In this evaluation, a one-I/O bus and one-disk architecture is used. The CPU center is adjusted by changing the number of CPUs. It is reflected by the effective service rate μ of the center. We use the four rate functions defined in Fig. 5, which represent the service rates of one-CPU, two-CPU, three-CPU, and four-CPU centers respectively.

TABLE 10

CPU Center Effects on *tpmC*

CPU center	μ_1	μ_2	μ_3	μ_4
tpmC	24.6 (0%)	47.2 (91.9%)	51.9 (110.0%)	53.5 (117.5%)

Note. The relative improvement ratios are given in the parenthesis.

Table 10 gives the performance of workload with $N = 48$, and their relative improvement ratios. The largest improvement happens when the CPU center changes from a single CPU to a dual CPU center. It yields an improvement by a factor of 91.1%. However, after that, with the increase of the number of CPUs, the improvements are moderate and become smaller and smaller. From this analysis, we find that only increasing the processing power (by increasing the number of processors) of a system will not significantly improve the performance of the TPC-C commercial workload on an SMP when the number of CPUs is more than 2. The reason for this is that I/O subsystems are not scaled up proportionally to the increase of the number of processors. The four-CPU configuration has the same amount of I/O capability as those with one, two, and three CPUs. When the I/O subsystems are saturated, increasing the number of CPUs does not improve the overall performance.

5.2. Effects of Memory System Latencies and I/O Bus Latencies

The memory system latency measures the access delay to the memory bus and the access delay to the memory. We used a 2-CPU SMP as the platform, whose service rate function is μ_2, and which has one I/O bus and one disk. The total load on the memory system will not be changed by different combinations of CPU, disk and I/O bus for a given N (we set $N = 48$). Table 11 shows the variance of *tpmC* when we reduce the memory latency (S_{Mem}) from 100 cycles to 50 cycles and reduce the memory bus reply latency ($S_{MBusRep}$) from 10 to 5 cycles. We also give its relative improvement ratios (inside parentheses). Performance of the SMP with 100 cycles of memory bus latency has a 91.9% relative improvement over that on the baseline SMP. When we reduce the memory service time to 90 cycles and the memory bus reply latency to 9 cycles, its relative *tpmC* improvement increases to

TABLE 11

Memory Bus Latency Effects on *tpmC*

S_{Mem}	100	90	80	70	60	50
$S_{MBusRep}$	10	9	8	7	6	5
$S_{MBusReq}$	2	2	2	2	2	2
tpmC	47.2 (91.9%)	52.9 (115.0%)	56.7 (130.5%)	58.3 (137.0%)	59.1 (140.2%)	59.6 (142.2%)
U_{IOBus}	40.3%	45.6%	53.4%	60.8%	67.7%	67.9%

Note. The relative improvement ratios are given in parentheses. The I/O bus utilizations are also given to explain the behavior of the system.

115.0%. But when the latency is reduced to 70 cycles or less, only a slight performance improvement is observed. This observation indicates that when the memory service time becomes 70 cycles or less and the memory bus reply latency becomes 7 cycles or less, the memory system is no longer the bottleneck of this SMP system and will not improve the performance of the workload even if we continue to reduce the latency. This is verified by the I/O bus utilizations. When the latency is 100 cycles, its I/O bus utilization is only a little more than 40%. However, it jumps to more than 60% after the memory service time reduces to less than 70 cycles. The utilization of more than 60% indicates that the I/O bus is becoming the new system performance bottleneck.

The effects of I/O bus latencies may vary with different I/O bus configurations. We evaluate the effects of I/O bus latencies on five SMP configurations. They are C2, C3, C5, C4, and C6 listed in Table 9. We present results of typical data allocation mappings. The mappings, f, used for one, two and four disks are defined as

$$\text{1 disk: } f_1(x) = 1 \qquad x = 1, 2, 3, 4, 5, 6, 7, 8, 9 \tag{5.11}$$

$$\text{2 disks: } f_2(x) = \begin{cases} 1 & x = 1, 3, 5, 7, 9 \\ 2 & x = 2, 4, 6, 8 \end{cases} \tag{5.12}$$

$$\text{4 disks: } f_4(x) = \begin{cases} 1 & x = 1, 3 \\ 2 & x = 2, 4 \\ 3 & x = 5, 7, 8 \\ 4 & x = 6, 9. \end{cases} \tag{5.13}$$

Up to 4 disks and up to 2 I/O buses are used in this study. When there are 4 disks and 2 I/O buses, we assume disks 1 and 2 are connected to I/O bus 1 and disks 3 and 4 to I/O bus 2. The mapping function f_1 in (5.11) means that all 9 database tables are allocated to a single disk, while the mapping function f_2 in (5.12) means database tables 1, 3, 5, 7, and 9 are allocated on disk 1, and database tables 2, 4, 6, and 8 are on disk 2. Similarly, f_4 allocates database tables in four disks. We start with the utilizations of the memory bus, I/O buses, and disks. Table 12 lists these utilizations. For all configurations, we observe that the I/O bus

TABLE 12

Disks, I/O Buses, and the Memory Bus Utilizations for Five SMP Configurations

	U_{Disk}				$U_{I/O\,Bus}$		
Configuration	1	2	3	4	1	2	$U_{M\,Bus}$
C2	63.4%	—	—	—	40.3%	—	58.7%
C3	41.2%	25.7%	—	—	31.5%	—	62.6%
C5	20.7%	10.2%	19.6%	15.9%	33.5%	—	68.6%
C4	41.2%	—	25.7%	—	12.3%	17.5%	68.2%
C6	20.7%	10.2%	19.6%	15.9%	14.0%	18.3%	70.5%

Note. In some configurations, certain disks and/or I/O bus 2 are not available. The symbol (—) means that the device is not available for that configuration.

utilizations are lower than the sum of all disk utilizations. This is because the disk seek and disk rotation latency are the major times for a disk access. Meanwhile, during this period of time, the I/O bus is released. The memory bus is used not only by processors but also by I/O operations when they require data transfer, so its utilizations are higher than that of I/O buses. When there is only one disk and one I/O bus, the disk utilization is very high (63.4%). Increasing the number of disks on one I/O bus does not seem very effective. But when the number of I/O buses increases to 2, the workload on I/O buses is balanced between the two, and can be executed concurrently. The response time of a disk access is reduced, and consequently, the memory bus utilizations increase.

Table 13 lists *tpmC* values and their relative performance improvement ratios (inside parentheses) on different SMP configurations (C2, C3, C4, C5, and C6) by reducing I/O bus latency 512 cycles a time. The results indicate that reducing I/O bus latencies in most cases moderately increases the throughput because it reduces the I/O bus waiting times. For example, on C6, when we reduce the latency to 3072, it gains 141.1% relative performance, which is 8.4% better than that of the latency of 5120. For configurations consisting of one bus (C2, C3, and C5), increasing the number of disks gains less performance than reducing I/O bus latencies. For a given latency, a configuration with more disks only slightly outperforms the configuration with fewer disks. A similar situation occurs in the 2-bus cases, but 2-bus configurations outperform 1-bus counterparts significantly because their workload is balanced on two buses. In terms of the impact significance on performance of the TPC-C workload, our study ranks the following parameters from the most significant to the least significant: the I/O bus latency, the number of I/O buses, and the number of disks.

5.3. *Effects of Disk Parameters*

The disk-related architectural parameters are disk seek time, disk rotation latency (the average time for the disk head to reach the required sector after the head has been on the track where the sector is), and disk transfer time. In this section, using the model results, we show how those parameters affect the performance of TPC-C.

The SMPs we used are C2, C3, C4, C5, and C6. All used a 2-CPU center with the service rate function μ_2 and the number of terminals was set to $N = 48$. We

TABLE 13

Effects of I/O Bus Latencies on *tpmC* for Five SMP Configurations

I/O bus latency	C2	C3	C5	C4	C6
3072	56.8 (130.9%)	57.4 (133.3%)	57.6 (134.2%)	58.9 (139.4%)	59.3 (141.1%)
3584	55.3 (124.8%)	55.9 (127.2%)	56.7 (130.5%)	57.0 (131.7%)	57.1 (132.1%)
4096	53.6 (117.9%)	54.1 (119.9%)	55.0 (123.6%)	56.4 (129.3%)	56.4 (129.3%)
4608	50.1 (103.7%)	51.7 (110.2%)	52.4 (113.0%)	55.2 (124.4%)	55.9 (127.2%)
5120	47.2 (91.9%)	48.3 (96.3%)	49.6 (101.6%)	52.3 (112.6%)	54.7 (122.4%)

Note. The improvements in comparison with the base SMP configuration are given in parentheses. The relative improvement ratios are given in parentheses.

TABLE 14

Disk Seek Time Effects on *tpmC* for Five SMP Configurations

Disk seek time	C2 1 bus, 1 disk	C3 1 bus, 2 disks	C5 1 bus, 4 disks	C4 2 buses, 2 disks	C6 2 buses, 4 disks
0.6M	52.3 (112.6%)	52.6 (113.8%)	53.1 (115.9%)	60.4 (145.5%)	60.9 (147.6%)
0.7M	51.4 (108.9%)	52.0 (111.4%)	52.7 (114.2%)	59.4 (141.5%)	60.1 (144.3%)
0.8M	49.6 (101.6%)	51.4 (108.9%)	52.1 (111.8%)	58.2 (136.6%)	59.3 (141.1%)
0.9M	48.3 (96.3%)	50.9 (106.9%)	51.3 (108.5%)	57.9 (135.4%)	58.9 (139.4%)
1M	47.2 (91.9%)	48.7 (98.0%)	50.1 (103.7%)	56.3 (128.9%)	57.1 (132.1%)

Note. The disk seek time is increased by 0.1 Million cycles on each line. The disk rotation latency and disk transfer time are fixed, and their values are 1.5M and 42000 cycles, respectively. The relative improvement ratios are given in parentheses.

reduced the value of each disk related parameter by a constant quantum of time: 0.1 million cycles for the disk seek time, 0.15 million cycles for the disk rotation latency, and 4200 cycles for the disk transfer time. Tables 14–16 summarize the results. From the tables, we see that among all disk related parameters, disk rotation latency (Table 15) affects performance most significantly (up to 157.3% in comparison with the baseline SMP configuration). Regarding disk rotation latency, each of the three SMPs has only one I/O bus (C2, C3, and C5). Reducing the rotation latency affected performance on C2 most significantly (16.7% in comparison with its counterpart having disk rotation latency of 1.5M). It affected C3 by 13.3%, and C5 by 10.5%. The reason is that on C2, all workloads are allocated on the single disk, and any improvement will result in significant reduction of its response time. But for the configurations of two and four disks, its effects are not as significant as that on one disk SMP. For the configurations of two buses (C4 and C6), the improvement is not as significant as that on the one bus SMP. This can be seen from the relative improvement compared with those having the original rotation latency (= 1.5M), which are 11.0 and 10.9% for C4 and C6, respectively.

We observed similar behavior when reducing the disk seek time, but the performance improvement is moderate (Table 14). The disk transfer time plays the least

TABLE 15

Disk Rotation Latency Effects on *tpmC* for Five SMP Configurations

Disk rotation latency	C2 1 bus, 1 disk	C3 1 bus, 2 disks	C5 1 bus, 4 disks	C4 2 buses, 2 disks	C6 2 buses, 4 disks
0.9M	55.1 (124.0%)	55.2 (124.4%)	55.4 (125.2%)	62.5 (154.1%)	63.3 (157.3%)
1.05M	54.6 (122.0%)	54.9 (123.2%)	55.0 (123.6%)	61.3 (149.2%)	62.5 (154.1%)
1.2M	51.3 (108.5%)	51.7 (110.2%)	52.9 (115.0%)	60.1 (144.3%)	61.4 (149.6%)
1.35M	48.7 (98.0%)	49.5 (101.2%)	51.8 (110.6%)	58.2 (136.6%)	58.7 (138.6%)
1.5M	47.2 (91.9%)	48.7 (98.0%)	50.1 (103.7%)	56.3 (128.9%)	57.1 (132.1%)

Note. The disk rotation latency is increased by 0.15 Million cycles on each line. The disk seek time and disk transfer time are fixed, and their values are 1M and 42000 cycles, respectively. The relative improvement ratios are given in parentheses.

TABLE 16

Disk Transfer Time Effects on *tpmC* for Five SMP Configurations

Disk transfer time	C2 1 bus, 1 disk	C3 1 bus, 2 disks	C5 1 bus, 4 disks	C4 2 buses, 2 disks	C6 2 buses, 4 disks
25200	50.3 (104.5%)	51.1 (107.7%)	51.5 (109.3%)	58.6 (138.2%)	58.7 (138.6%)
29400	50.1 (103.7%)	50.6 (105.7%)	51.2 (108.1%)	58.4 (137.4%)	58.5 (137.8%)
33600	49.2 (100.0%)	49.5 (101.2%)	51.0 (107.3%)	58.0 (135.8%)	58.1 (136.2%)
37800	48.7 (98.0%)	48.9 (98.8%)	50.6 (105.7%)	57.6 (134.1%)	57.7 (134.6%)
42000	47.2 (91.9%)	48.7 (98.0%)	50.1 (103.7%)	56.3 (128.9%)	57.1 (132.1%)

Note. The disk transfer time is increased by 4200 cycles on each line. The disk seek time and disk rotation latency are fixed, and their values are 1M and 1.5M cycles, respectively. The relative improvement ratios are given in parentheses.

role (at most 138.6%) in the performance improvement. For example, for C2, the performance improved only by 6.5% when the transfer time was reduced by 40%, while when reducing the rotation latency by 40%, it yields 16.7% improvement for C2. It indicates that disks spend most of their time on seeking and locating the data rather than transferring them.

From this study, we know that among all disk-related architectural parameters, the disk rotation latency plays the most important role in performance improvement, while the disk seek time is the second, and the disk transfer time is the least factor.

5.4. Effects of Data Allocation

Using the model, we also evaluate the performance of TPC-C on an architecture with a given number of disks and I/O buses but with different data (database table) allocations. Since a 4-disk and 2-I/O bus configuration provides many alternatives to allocate data, we use it as the platform to study the data allocation effects. The number of CPUs is 2, and the number of terminals is 48.

For a system of 4-disk and 2-I/O bus architecture, there are several ways to allocate the data among four disks. Table 17 lists some candidates, and their relative performance improvement ratios. Allocation A distributes the database tables almost evenly (based on the number of database tables) among all disks. Allocation B intends to distribute all data of database tables approximately evenly among disks (with $W = 1$). Allocations C and D distribute data on a configuration with one disk on each I/O bus. They are used to compare with those configurations (A and B) using more than one disk on one I/O bus. Allocation E provides the best performance among all our tested allocations in terms of *tpmC*. We will explain later why this allocation yields better performance than any other allocation we investigated.

Utilizations of I/O buses and disks are presented in Table 18. For allocation A, because the database tables 1, 2, and 3 are in the same disk, and they are the major data accessed by the major transactions I and II, the result is a very high utilization for I/O Bus 1 (72.7%), and relatively high disk utilization for DISK 1 and 2 (58.3 and 45.4% respectively). On the other hand, the workload on I/O Bus 2 and its

TABLE 17

Five Data Allocation Candidates and Their Performance in *tpmC* and Relative Improvement Ratios in Comparison with the Baseline SMP

	I/O Bus 1		I/O Bus 2		
Allocation	DISK 1	DISK 2	DISK 3	DISK 4	*tmpC*
A	{1 2 3}	{4 5}	{6 7}	{8 9}	49.3 (100.4%)
B	{8}	{1 2 4 7}	{3}	{5 6 9}	55.4 (125.2%)
C	{1 2 3 4 5}	{}	{6 7 8 9}	{}	48.2 (95.9%)
D	{1 2 4 7 8}	{}	{3 5 6 9}	{}	53.6 (117.9%)
E	{8 4}	{9}	{6}	{1 2 3 5 7}	58.7 (138.6%)

Note. The baseline SMP has only one disk and all data are put on that disk. The database tables are numbered from 1 to 9.

disks are comparatively small. This gives about 100.4% increase of overall performance in comparison with the baseline SMP with one I/O bus and one disk. Merging two disks' data (allocation C) reduces the improvement to 95.9%. In comparison with A, the utilization of DISK 1 increases to 69.8%. Since all accesses to the database tables 1–5 are conducted on DISK 1, there is no overlapping of the usage of I/O buses and the disks. Consequently, the response time increases while the utilization of I/O buses reduces. Even in such a case, it still outperforms by 95.9% compared with the one with the single I/O bus. Allocation B distributes the data almost evenly among the disks. But since the access probabilities for each data item are not equal, this leads also to the imbalance of workload among disks and I/O buses. This is reflected in Table 18. I/O Bus 1 is still the hot spot. However, compared with A, we find it is much better balanced; thus it yields a 125.2% improvement. Allocation D is similar to B. Allocation E has the most balanced database table allocation with regard to both the amount of data and data access probabilities for each transaction. Database table 8 is the largest one, while database table 9 is less frequently accessed. Putting them on different disks but on the same I/O bus minimizes the response time for accessing database table 8, which is beneficial for transactions I, III, IV, and V. Less frequent accesses to database table 9 makes it possible to use the idle cycles of I/O Bus 1 while DISK 1 is busy seeking blocks. Database table 6 is a relatively large and frequently accessed table. It is stored separately in a disk

TABLE 18

Utilizations of I/O Buses and Disks for the Five Types of Data Allocations

Allocation	I/O Bus 1	I/O Bus 2	DISK 1	DISK 2	DISK 3	DISK 4
A	72.7%	31.4%	58.3%	45.4%	10.6%	8.7%
B	67.8%	41.1%	57.3%	48.9%	25.3%	27.4%
C	58.7%	20.3%	69.8%	0%	43.3%	0%
D	51.4%	30.8%	76.8%	0%	64.2%	0%
E	45.7%	38.5%	54.2%	47.3%	38.6%	46.1%

to reduce its effect on the access of other database tables. The remaining database tables are in the fourth disk. This setting balances the data traffic flows among the memory bus and I/O buses, thus yielding the best performance among all our tested allocations. It improves the performance by a factor of 138.6% in comparison with the baseline SMP and 24.3% in comparison with putting all data on one disk.

Through this study, we find that given an SMP configuration, the performance of TPC-C can be improved moderately by allocating the database tables properly. This is because the access rates for different database tables are quite different. Allocating frequently accessed tables on one disk or on disks on one I/O bus will incur high disk or I/O bus utilization demand, thus making it become the bottleneck of the system. In contrast, allocating them separately to balance the workload will yield better performance. This is a relatively easy way to improve performance without upgrading any hardware devices. Our experiments show this promise. In practice, some hardware performance instrumenting and monitoring mechanisms can be used to gather utilizations of all devices. Performance can be tuned based on adjusting data allocations.

6. RELATED WORK

Maynard *et al.* [11] compare the cache/memory performance between scientific and commercial workloads in a uniprocessor system through trace-driven simulations. A large number of user processes, a high percentage of operating system activities, and distinctive branch behavior make commercial data not cache efficient. The authors of [11] also find that increasing the associativity of second-level caches can reduce miss rates.

Trancoso *et al.* [16] investigate memory performance of decision support system —a kind of commercial workload benchmarked by TPC-D—in cache-coherent shared-memory multiprocessors. Three representative queries are selected from TPC-D, and their memory performance is analyzed. For each query, sequential and index accesses to the database data are evaluated. They find that both kinds of accesses can exploit spatial locality and therefore can benefit from relatively long cache lines.

Leutenegger and Dias [10] model the TPC-C benchmark for single-node and distributed database management systems. Data access skew is quantified. A LRU buffer is assumed to be used. They examine the effect of packing hot tuples into pages and show the price/performance benefit as well.

There are also several papers concerning performance on specific SMP machines with specific hardware supporting facilities. Piantedosi and Sathaye [12] evaluate the performance of commercial workloads in DEC AlphaSevers using memory channels. A memory channel is a special high speed network dedicated to the data transfer between memory of several machines. It reduces significantly the overhead needed to share memory resident data between computers, thus decreasing the number of disk accesses, and improving the performance of the workloads.

Similar to the conclusions reached by Maynard *et al.*, Eickemeyer *et al.* [3] find growing miss rates due to commercial applications. They evaluate the use of multithreaded uniprocessors for commercial applications in order to overlap the computation with the access to memory. So when a thread is blocked by a memory access caused by a cache miss, another thread may take control of the processor to do some effective computation. They observe that multithreading can provide significant improvements for uniprocessor commercial computing environments.

Thakkar and Sweiger [15] evaluate the performance of an OLTP benchmark TP1 (ancestor of TPC-A), which simulates a banking (deposit/withdraw) system workload on a Sequent Symmetry system. Because the system bus is designed specifically as a pipeline bus optimized for data transfer between CPU and memory, cache-to-cache transfer has a higher overhead than cache-to-memory. Thus, when cache is relatively large, and a process migrates from a processor to another, cache-to-cache transfer is required, which may incur higher overhead and may affect the overall performance of the system. The paper investigates the effects of two scheduling schemes, affinity and nonaffinity, on the performance of TP1.

Compared with scientific applications, cache and memory plays a smaller role in the performance of commercial workloads. As [11] states, about half (43% for TPC-C) of the instructions of commercial applications executed involve disk I/O. Consequently, memory and disks are the bottleneck of the system. Multiple processors may reduce the execution time of computational parts of the workloads, but they do not release any burden on I/O devices. It might affect positively or negatively the overall performance of workloads. Multiple disks usually provide more storage capacity for the workloads. But if they are configured poorly, and data are allocated improperly, they may affect the performance as well. The memory bus and I/O buses are other critical resources and could easily become the bottlenecks of the system. Our research differs from the above-cited work in several respects. We focus on

1. commercial workload performance on general symmetric multiprocessor systems,
2. the architectural (especially I/O) impact on performance, and
3. the queuing network models supported by simulation.

7. CONCLUSIONS AND FUTURE WORK

Symmetric multiprocessors have become a popular parallel server for running database-oriented commercial applications. Performance evaluation is the first step toward improving the performance of commercial applications on SMPs. Experiment-based evaluation requires significant computing resources and time. In this paper, we investigate the feasibility of using a queueing network model supported by simulation to characterize commercial workloads on SMPs. Although our modeling results may not be as precise as the results from intensive simulation and measurements, they are sufficiently accurate and acceptable for characterizing the TPC-C workload on SMPs and revealing insights into SMP architectural effects on performance of commercial

workloads. A unique feature of this approach is its efficiency in terms of timing and computing resources.

In summary, we have made the following contributions in this performance study:

- We developed a queueing network model to characterize and analyze the behavior of a commercial workload on SMP systems.
- We also built a MINT-based SMP simulator integrated with a disk simulator and the POSTGRES database for the TPC-C execution. The simulation environment was used to collect the input parameters for the model (parameterization) and to verify the model. The simulation results show that the model is sufficiently accurate.
- Using the model, we quantitatively analyzed the SMP architectural impact on performance of TPC-C. Our studies indicate that among disk-related parameters, the disk rotation latency affects performance of TPC-C most significantly. Among I/O buses and number of disks, the number of I/O buses has the most significant impact on performance.

We have also investigated the initial data allocation policies of TPC-C. We show that performance could be effectively improved by allocating the database tables among several disks based on utilizations of each table, of disks, and of I/O buses. In a two-I/O bus and four-disk architecture, we achieved a performance improvement by a factor of 24.3% using a selective allocation policy.

The use of parallel disks [2] in an SMP is an effective way to improve performance because it reduces average disk rotation time and disk transfer time from concurrent disk operations. Since the I/O bus latency plays an important role in performance, accommodating faster I/O buses in SMPs is another way to improve commercial workload performance. For a given configuration, employing proper data allocation is also a promising approach for enhancing performance while not upgrading any hardware. Our experiments only tested various allocations in the units of database tables. An alternative method is to allocate the data in smaller units, such as tuples, based on some semantics of the data. All those approaches are static: data are allocated before execution. Dynamic allocation is another interesting issue. The time spent on transferring data among disks is definitely the major overhead, and it will offset performance gain. However, a reconfigurable connection mechanism (rather than a simple I/O bus) between disks will provide low-cost support for data redistribution and make this approach attractive. All those issues are open and need further investigation. Disk buffers and database buffers also help reduce the number of disk accesses. Currently, we are studying data access patterns of commercial workloads. Several buffer management strategies are proposed and evaluated for their effectiveness. Since scalability is a major limit on bus-based shared memory multiprocessors, we are investigating other alternatives as well. With the availability of high-speed networks and powerful workstations with large disk storage capability, networks of workstations are a cost-effective candidate for commercial applications. We are studying the performance of commercial workloads supported by a distributed database management system on networks of workstations.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive comments and suggestions on the paper. We appreciate Z. Zhu's discussions and comments on this work. We are thankful to Neal Wagner for reading the paper and for his comments.

REFERENCES

1. L. A. Barroso, K. Gharachorloo, and E. Bugnion, Memory system characterization of commercial workloads, *in* "Proceedings of the 25th International Symposium on Computer Architecture," June, 1998.
2. P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson, RAID: High-performance, reliable secondary storage, *ACM Comput. Surveys* (June 1994).
3. R. J. Eickemeyer, R. E. Johnson, S. R. Kunkel, M. S. Squillante, and S. Liu, Evaluation of multithreaded uniprocessors for commercial application environments, *in* "Proceedings of ISCA'96," pp. 203–212, 1996.
4. M. Heinrich *et al.*, The performance impact of flexibility in the Stanford FLASH multiprocessor, *in* "Proceedings of the 6th ASPLOS," pp. 274–285, 1994.
5. J. L. Hennessy and D. A. Patterson, "Computer Architecture: A Quantitative Approach," second ed., Morgan Kaufmann, San Francisco, 1996.
6. W. W. Hsu and J-K. Peir, Buses, *in* "CRC Handbook of Computer Science and Engineering" (A. B. Tucker, Ed.), pp. 427–446, CRC Press, Boca Raton, FL, 1997.
7. H. Jiang, L. N. Bhuyan, and J. K. Muppala, MVAMIN: Mean value analysis algorithms for multistage interconnection networks, *J. Parallel Distrib. Comput.* **12** (1991), 189–201.
8. D. Kotz, S. B. Toh, and S. Radhakrishnan, "A Detailed Simulation Model of the HP 97560 Disk Drive," TR PCS-TR94-220, Department of Computer Science, Dartmouth College, 1994.
9. E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik, "Quantitative System Performance —Computer System Analysis Using Queueing Network Models," Prentice–Hall, New York, 1984.
10. S. T. Leutenegger and D. Dias, A modeling study of the TPC-C benchmark, "ACM SIGMOD'93," pp. 22–31, 1993.
11. A. M. G. Maynard, C. M. Donnelly, and B. R. Olszewski, Contrasting characteristics and cache performance of technical and multi-user commercial workloads, *in* "The Sixth International Conference on Architectural Support for Programming Languages and Operating Systems," pp. 145–156, October 1994.
12. J. A. Piantedosi and A. S. Sathaye, Performance measurement of TruCluster systems under the TPC-C benchmark, *Digital Tech. J.* **8**(3) (1996), 46–57.
13. C. Ruemmler and J. Wilkes, An introduction to disk drive modeling, *IEEE Comput.* (March 1994), 17–28.
14. M. Stonebraker and G. Kemnitz, The POSTGRES next generation database management system, *Comm. ACM* **34**(10) (1991), 78–92.
15. S. S. Thakkar and M. Sweiger, Performance of an OLTP application on symmetry multiprocessor system, *in* "Proceedings of ISCA'90," pp. 228–238, 1990.
16. P. Trancoso *et al.*, The memory performance of DSS commercial workloads in shared-memory multiprocessors, *in* "The Third International Sympoisum on High-Performance Computer Architecture," pp. 250–260, February 1997.
17. D. Towsley, Approximate models of multiple-bus multiprocessor systems, *IEEE Trans. Comput.* **35**(3) (1986), 220–228.
18. Transaction Processing Performance Council, TPC Benchmark C, "TPC Benchmark C Stanford Specification, Revision 3.3.3," April 16, 1998.

19. J. E. Veenstra and R. J. Fowler, MINT: A front end for efficient simulation of shared-memory multiprocessors, *in* "Proceedings of the Second International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems," pp. 201–207, 1994.
20. Q. Yang and L. N. Bhuyan, A queueing network model for a cache coherence protocol on multiple-bus multiprocessors, *in* "Proceedings of International Conference on Parallel Processing," pp. 130–137, 1988.

XING DU received his B.S. and Ph.D. in computer science from Nanjing University, China, in 1986 and 1991, respectively. He is a software design engineer at Software and Systems Development Lab of Hewlett-Packard Company. He worked at the Oracle Corporation between 1999 and 2000. He was a research associate in the University of Texas at San Antonio and in the College of William and Mary between 1995 to 1998. He worked as a research scientist at the University of Virginia between 1998 to 1999. His research interests are parallel/distributed systems and software engineering.

XIAODONG ZHANG is a professor of computer science at the College of William and Mary. He received his B.S. in electrical engineering from Beijing Polytechnic University, China, in 1982, and his M.S. and Ph.D. in computer science from University of Colorado at Boulder, in 1985 and 1989, respectively. His research interests are parallel and distributed systems, computer memory systems and scientific computing. He is an associate editor of *IEEE Transactions on Parallel and Distributed Systems*, and has chaired the IEEE Computer Society Technical Committee on Supercomputing Applications.

YINGFEI DONG is a Ph.D. candidate in computer science at the University of Minnesota. He received his B.S. and M.S. in computer science at Harbin Institute of Technology, in 1989 and 1992, respectively. He was a research associate in the High Performance Computing and Software Lab at the University of Texas at San Antonio from 1995 to 1997. His research interests are in the areas of networking systems. He is a member of ACM.

LIN ZHANG is a member of the technical staff at LXB World Marketing Company in Canada. She participated in the work on commercial workload evaluation of shared-memory systems in the High Performance Computing and Software Lab at the University of Texas at San Antonio from 1996 to 1997. She received her B.S. in computer science from Nanjing University, China, in 1986.