Performance Evaluation of Transaction Processing Systems

By Walter H. Kohler, Yun-Ping Hsu, Thomas K. Rogers, Wael H. Bahaa-El-Din

Abstract

Performance and price /performance are important attributes to consider when evaluating a transaction processing system. Two major approaches to performance evaluation are measurement and modeling. TPC Benchmark A is an industry standard benchmark for measuring a transaction processing system's performance and price/performance. Digital has implemented TPC Benchmark A in a distributed transaction processing environment. Benchmark measurements were performed on the VAX 9000 Model 210 and the VAX 4000 Model 300 systems. Further, a comprehensive analytical model was developed and customized to model the performance behavior of TPC Benchmark A on Digital's transaction processing platforms. This model was validated using measurement results and has proven to be an accurate performance prediction tool.

Transaction processing systems are complex in nature and are usually characterized by a large number of interactive terminals and users, a large volume of on-line data and storage devices, and a high volume of concurrent and shared database accesses. Transaction processing systems require layers of software components and hardware devices to work in concert. Performance and price/performance are two important attributes for customers to consider when selecting transaction processing systems. Performance is important because transaction processing systems are frequently used to operate the customer's business or handle mission-critical tasks. Therefore, a certain level of throughput and response time guarantee are required from the systems during normal operation. Price/performance is the total system and maintenance cost in

Introduction

dollars, normalized by the performance metric.

Performance Evaluation of Transaction Processing Systems

The performance of a transaction processing system is often measured by its throughput in transactions per second (TPS) that satisfies a response time constraint. For example, 90 percent of the transactions must have a response time that is less than 2 seconds. This throughput, qualified by the associated response time constraint, is called the maximum qualified throughput (MQTh). In a transaction processing environment, the most meaningful response time definition is the end-to-end response time, i.e., the response time observed by a user at a terminal. The end-to-end response time represents the time required by all components that compose the transaction processing system.

The two major approaches used for evaluating transaction processing system performance are measurement and modeling. The measurement approach is the most realistic way of evaluating the performance of a system. Performance measurement results from standard benchmarks have been the most accepted form of performance assessment of transaction processing systems. However, due to the complexity of

Modeling uses simulation or analytical modeling techniques. Compared to the measurement approach, modeling makes it easier to produce results and requires less computing resources. Performance models are also flexible. Models can be used to answer "what-if" types of questions and to provide insights into the complex performance behavior of transaction processing systems, which is difficult (if not impossible) to observe in the measurement environment. Performance models are widely used in research and engineering communities to provide valuable analysis of design alternatives, architecture evaluation, and capacity planning. Simplifying assumptions are usually made in the modeling approach. Therefore, performance models require validation, through detailed simulation or measurement, before predictions from the models are accepted.

This paper presents Digital's benchmark measurement and modeling approaches to transaction processing system performance evaluation. The paper includes an overview of the current industry standard transaction processing benchmark, the

transaction processing systems, such measurements are usually very expensive, very time-consuming, and difficult to perform.

TPC Benchmark A, and a description of Digital's implementation of the benchmark, including the distinguishing features of the implementation and the benchmark methodology. The

performance measurement results that were achieved by using the TPC Benchmark A are also presented. Finally, a multilevel analytical model of the performance behavior of transaction processing systems with response time constraints is presented and validated against measurement results.

TPC Benchmark a-an Overview

The TPC Benchmark A simulates a simple banking environment and exercises key components of the system under test (SUT) by using a simple, update-intensive transaction type. The benchmark is intended to simulate a class of transaction processing application environments, not the entire range of transaction processing environments. Nevertheless, the single transaction type specified by the TPC Benchmark A standard provides a simple and repeatable unit of work. The benchmark can be run in either a local area network (LAN) or a wide area network (WAN) configuration. The related throughput metrics are tpsA-Local and tpsA-Wide, respectively. The benchmark specification defines the general application requirements, database design and scaling rules, testing and pricing guidelines, full disclosure report requirements, and an audit checklist.[1] The following sections provide an overview of the benchmark.

Application Environment

The TPC Benchmark A workload is patterned after a simplified banking application. In this model, the bank contains one or more branches. Each branch has 10 tellers and 100,000 customer accounts. A transaction occurs when a teller enters a deposit or a withdrawal for a customer against an account at a branch location. Each teller enters transactions at an average rate of one every 10 seconds. Figure 1 illustrates this simplified banking environment.

Transaction Logic

The transaction logic of the TPC Benchmark A workload can be described in terms of the bank environment shown in Figure 1. A teller deposits in or withdraws money from an account, updates the current cash position of the teller and branch, and makes an entry of the transaction in a history file. The pseudocode shown in Figure 2 represents the transaction.

Terminal Communication

For each transaction, the originating terminal is required to transmit data to, and receive data from, the system under test. The data sent to the system under test must consist of at least 100 alphanumeric data bytes, organized as at least four distinct fields: Account_ID, Teller_ID, Branch_ID, and Delta. The Branch_ID identifies the branch where the teller is located. The Delta is the amount to be credited to, or debited from, the specified account. The data received from the system under test consists of at least 200 data bytes, organized as the above four input fields and the Account_Balance that results from the successful commit operation of the transaction.

Implementation Constraints

The TPC Benchmark A imposes several conditions on the test environment.

- The tested system must preserve the effects of committed transactions and ensure database consistency after recovering from
  - The failure of a single durable medium that contains database or recovery log data
  - The crash and reboot of the system
  - The loss of all or part of memory
- Eighty-five percent of the accounts processed by a teller must belong to the home branch (the one to which the teller belongs). Fifteen percent of the accounts processed by a teller must be owned by a remote branch (one to which the teller does not belong). Accounts must be uniformly distributed and randomly selected.

Database Design

o The transaction processing system must support atomicity, consistency, isolation, and durability (ACID) properties during the test.	The database consists of four individual files /tables: Branch, Teller, Account, and History, as defined in Table 1. The overall size of the database is determined by the throughput capacity of the system. Ten tellers, each entering transactions at an average rate of

one transaction every 10 seconds, generate what is defined as a one-TPS load. Therefore, each teller contributes one-tenth (1/10) TPS. The history area must be large enough to store the history records generated during 90 eight-hour days of operation at the published system TPS capacity. For a system that has a processing capacity of x TPS, the database is sized as shown in Table 2.

Table 1

Database Entities

Record	Bytes	Fields Required	Description
Branch	100	Branch_ID	Identifies the branch across the range of branches
		Branch_Balance	Contains the branch's current cash balance
Teller	100	Teller_ID	Identifies the teller across the range of tellers
		Branch_ID	Identifies the branch where the teller is located
		Teller_Balance	Contains the teller's current cash balance
Account	100	Account_ID	Identifies the customer account uniquely for the entire database
		Branch_ID	Identifies the branch where the account is held
		Account_Balance	Contains the account's current cash balance
History	50	Account_ID	Identifies the account updated by the transaction
		Teller_ID	Identifies the teller involved in the transaction

Branch_ID	Identifies the branch associated with the teller
Amount	Contains the amount of credit or debit (delta) specified by the transaction.

Table 1 (Cont.)

Database Entities

Record	Bytes	Fields Required	Description
		Time_Stamp	Contains the date and time taken between the BEGIN TRANSACTION and COMMIT_TRANSACTION statements

Table 2

Database Sizing

Number_of_Records	Record Type
1 x x	Branch records
10 xx x	Teller records
100,000 x x	Account records
2,592,000 x x	History records

For example, to process 20 TPS, a system must use a database that includes 20 branch records, 200 teller records, and 2,000,000 account records. Because each teller uses a terminal, the price of the system must include

Benchmark A uses two basic metrics:

- Transactions per second (TPS)-throughput in TPS, subject to a response time constraint, i.e., the MQTh, is measured while the system is in a sustainable steady-state condition.
- Price per TPS (K$/TPS)-the purchase price and five-year maintenance costs associated with one TPS.

Transactions per Second. To guarantee that the tested system provides fast response to on-line users, the TPC Benchmark A imposes a specific response time constraint on the benchmark. Ninety percent of all transactions must have a response time of less than two seconds. The

200 terminals. A test that results in a higher TPS rate is invalid unless the size of the database and the number of terminals are increased proportionately.

Benchmark Metrics

TPC Benchmark A standard defines transaction response time as the time interval between the transmission from the terminal of the first byte of the input message to the system under test to the arrival at the terminal of the last byte of the output message from the system under test.

The reported TPS is the total number of committed transactions that both started and completed during an interval of steady-state performance, divided by the elapsed time of the interval. The steady-state measurement interval must be at least 15 minutes, and 90 percent of the transactions must have a response time of less than 2 seconds.

Price per TPS. The K$/TPS price/performance metric measures the total system price in thousands of dollars, normalized by the TPS rating of the system. The priced system includes all the components that a customer requires to achieve the reported performance level and is defined by the TPC Benchmark A standard as the

- Price of the system under test, including all hardware, software, and maintenance for five years.
- Price of the terminals and network components, and their maintenance for five years.
- Price of on-line storage for 90 days of history records at the published TPS rate, which amounts
- Price of additional products required for the operation, administration, or maintenance of the priced systems.
- Price of products required for application development.

All hardware and software used in the tested configuration must be announced and generally available to customers.

TPC Benchmark a Implementation

Digital's implementation of the TPC Benchmark A goes beyond the minimum requirements of the TPC Benchmark A standard and uses Digital's distributed approach to transaction processing.[2] For example, Digital's TPC Benchmark A implementation includes forms management and transaction processing monitor software that are required in most real transaction processing environments but are not required by the benchmark. The following sections provide an overview of Digital's approach and implementation.

Transaction Processing Software Environment

The three basic functions

to 2,592,000 records per TPS. A storage medium is considered to be on-line if any record can be accessed randomly within one second.

of a general-purpose transaction processing system are the user interface (forms processing), applications management, and database management. Digital has developed a distributed transaction architecture (DECdta) to define how

the major functions are partitioned and supported by components that fit together to form a complete transaction processing system. Table 3 shows the software components in a typical Digital transaction processing environment.

Table 3
Transaction Processing Software Components

Component	Example
Operating system	VMS
Communications	LAT, DECnet
Database	VAX Rdb /VMS
TP monitor	VAX ACMS, DECintact
Forms	DECforms
Application	COBOL

Distributed Transaction Processing Approach

Digital transaction processing systems can be distributed by placing one or more of the basic system functions (i.e., user interface, application manager, database manager) on separate computers. In the simplest form of a distributed transaction processing system, the user forms management to be performed at a remote location, whereas the application is processed at a central location. The Digital transaction processing software components are separable because their clearly defined interfaces can be layered transparently onto a network. How these components may be partitioned in the Digital distributed transaction processing environment is illustrated in Figure 3.

interface component runs
on a front-end processor,
and the application and
database components run
on a back-end processor.
The configuration
allows terminal and

TPC Benchmark A Test Environment

The Digital TPC Benchmark A tests are implemented in a distributed transaction processing environment using the transaction processing software components shown in Figure 3. The user interface component runs on one or more front-end processors, whereas the application and database components run on one or more back-end processors. Transactions are entered from teller terminals, which communicate with the front-end processors. The front-end processors then communicate with the back-end processors to invoke the application servers and perform database operations. The communications can take place over either a local area or a wide area network. However, to simplify testing, the TPC Benchmark A standard allows sponsors to use remote terminal emulators (RTEs) rather than real terminals. Therefore, the TPC Benchmark A tests base performance and price /performance results on two distinctly configured systems, the target system and the test system.

The target system is the initiate transactions and communicate with the front-end processors. Front-end processors communicate with a back-end processor using the DECnet protocol.

The test system is the configuration of components used in the lab to measure the performance of the target system. The test system uses RTEs, rather than user terminals, to generate the workload and measure response time. (Note: In previously published reports, based on Digital's DebitCredit benchmark, the RTE emulated front-end processors. In the TPC Benchmark A standard, the RTE emulates only the user terminals.) The RTE component

- Emulates the behavior of terminal users according to the benchmark specification (e.g., think time, transaction parameters)
- Emulates terminal devices (e.g., conversion and multiplexing into the local area transport [LAT] protocol used by the DECserver terminal servers)
- Records transaction messages and response times (e.g., the starting and ending

times of individual

configuration of hardware and software components that customers can use to perform transaction processing. With the Digital distributed transaction processing approach, user terminals

transactions from each emulated terminal device)

Figure 4 depicts the test system configuration in the LAN environment with one back-end processor, multiple front-end processors, and multiple remote terminal emulators.

TPC Benchmark A Results

We now present the results of two TPC Benchmark A tests based on audited benchmark experiments performed on the VAX 9000 Model 210 and the VAX 4000 Model 300 systems. [3,4] These two systems are representative of Digital's large and small transaction processing platforms. The benchmark was implemented using the VAX ACMS transaction processing monitor, the VAX Rdb/VMS relational database management system, and the DECforms forms management system on the VMS operating system. Tables 4 and 5 show the back-end system configurations for the VAX 9000 Model 210 and the VAX 4000 Model 300 systems, respectively. Table 6 shows the system configuration of the front-end systems.

Component	Product	Quantity
Communications	DECnet-VMS Phase IV	1
TP monitor	VAX ACMS V3.1	1
Dictionary	VAX CDD/Plus V4.1	1
Application	VAX COBOL V4.2	1
Database system	VAX Rdb/VMS V4.0	1
Forms management	DECforms V1.2	1

Table 5

VAX 4000 Model 300 Back-end System Configuration

Component	Product	Quantity
Processor	VAX 4000 Model 300	1
Memory		64 MB
Tape drive	TK70	1

Table 4
VAX 9000 Model 210 Back-end System Configuration

Component	Product	Quanti	Disks	RF31
Processor	VAX 9000 Model 210	1	Operating system	VMS 5.4
Memory		256 MB	Communications	DECnet-VMS Phase IV
Tape drive	TA81	1		
Disk controller	KDM70	2	TP monitor	VAX ACMS V3.1
Disks	RA92	16	Dictionary	VAX CDD/Plus V4.1
Operating system	VMS 5.4	1		
Disk controller	DSSI	3		
Disks	RF31	18		
Tape drive	TA81	1		
Disk controller	KDM70	2		
Disks	RA92	16		

Table 5 (Cont.)

VAX 4000 Model 300 Back-end
System Configuration

Component	Product	Quantity
Application	VAX COBOL V4.2	1
Database system	VAX Rdb/VMS V4.0	1
Forms management	DECforms V1.2	1

Table 6

Front-end Run-time System Configuration

Component	Product	Quantity
Processor	VAXserver 3100 Model 10	10 for VAX 9000 back-end
		3 for VAX 4000 back-end
Memory		16 MB for VAX 9000 back-end
		12 MB for VAX 4000 back-end
Disks	RZ23 (104 MB)	16
Operating system	VMS 5.3	1 for VAX 9000 back-end
	VMS 5.4	1 for VAX 4000 back-end
Communications	DECnet-VMS Phase IV	1
TP monitor	VAX ACMS V3.1	1
Forms_management	DECforms_V1.2	1

Table 7

VAX 9000 Model 210 Maximum Qualified Throughput

System	TPS (tpsA-Local)	Response Time (seconds)		
		Average	90 percent	Maximum

VAX 9000 Model 210	69.4	1.20	1.74	5.82
VAX 4000 Model 300	21.6	1.39	1.99	4.81

Measurement Results

The maximum qualified throughput and response time results for the TPC Benchmark A are summarized in Table 7 for the VAX 9000 Model 210 and the VAX 4000 Model 300 systems.

Both configurations have sufficient main memory and disk drives such that the processors are effectively utilized with no other bottleneck. Both systems achieved well over 90 percent CPU

utilization at the maximum qualified throughput under the response time constraint. In addition to the throughput and response time, the TPC Benchmark A specification requires that several other data points and graphs be reported. We demonstrate these data and graphs by using the VAX 9000 Model 210 TPC Benchmark A results.

- Response Time in Relationship to TPS. Figure 5 shows the 90th percentile and average response times at 100 percent and approximately 80 percent and 50 percent of the maximum qualified throughput. The mean transaction response time still grows linearly with the transaction rate up to the 70 TPS level, but the 90th percentile response time curve has started to rise quickly due to the high CPU utilization and random arrival of transactions.
- Response Time Frequency Distribution. Figure 6 is a graphical representation of the transaction response time distribution. The average, 90th percentile, and maximum transaction response times are also marked on the graph.
- Transactions per Second over Time.

  The results shown in Figure 7 demonstrate the sustainable maximum qualified throughput. The one-minute running average transaction throughputs during the warm-up and data collection periods of the experiment are plotted on the graph. This graph shows that the throughput was steady during the period of data collection.
- Average Response Time over Time. The results shown in Figure 8 demonstrate the sustainable average response time in

  the experiment. The one-minute running average transaction response times during the warm-up and data collection periods of the experiment are plotted on the graph. This graph shows that the mean response time

was steady during the period of data collection.

Comprehensive Analytical Model

Modeling techniques can be used as a supplement or an alternative to the measurement approach. The performance behavior of complex transaction processing systems can be characterized by a set of parameters, a set of performance metrics, and the relationships among them. These parameters can be used to describe the different resources available in the system, the database operations of transactions, and the workload that the transaction processing system undergoes. To completely represent such a system, the size of the parameter set would be too huge to manage. An analytical model simplifies, through abstraction, the complex behavior of a system into a manageable set of parameters and policies. Such a model, after proper validation, can be a powerful tool for many types of analysis, as well as a performance prediction tool. Results can be obtained quickly for any combination of parameters.

A comprehensive analytical model of the performance behavior of transaction processing systems has been developed. The model can be used to study the basic construction of the model and the customization made to model the execution of TPC Benchmark A on Digital's transaction processing systems. The model can also be used to study different transaction processing workloads in addition to the TPC Benchmark A.

Response Time Components

The main metric used in the model is the maximum qualified throughput under a response time constraint. The response time constraint is in the form of "x percent of transaction response times are less than y seconds." To evaluate throughput under such response time constraint, the distribution of transaction response times is determined by first decomposing the transaction response time into nonoverlapping and independent components. The distribution of each component is then evaluated. Finally, the overall transaction response time distribution is derived from the mathematical convolution of the component response time distributions.

The logical flow of a transaction in a front-end and back-end distributed

processing systems with a response time constraint was developed and validated against measurement results. This model is hierarchical and flexible for extension. The following sections describe transaction processing system that is used to implement TPC Benchmark A is depicted in Figure 9. The response time of a transaction consists of three basic components: front-end processing,

back-end processing, and communication delays.

- Front-end processing usually includes terminal I/O processing, forms/presentation services, and communication with the back-end systems. In the benchmark experiments, no disk I/O activity was involved during the front-end processing.

Within the back-end system, the transaction response time is further decomposed into two additional components, CPU delays and non-CPU, nonoverlapping delays. CPU delays include both the CPU service and the CPU waiting times of transactions. Non-CPU, nonoverlapping delays include:

- Logging delays, which include the time for transaction log writes and commit protocol delays
- Database I/O delays, which include both waiting and service times for accessing storage devices
- Back-end processing includes the execution of application, database access, concurrency control, and transaction commit processing. The back-end processing usually involves a high degree of concurrency and many disk I/O activities.
- Communication delays primarily include the communications between the user terminal and the front-end node, and the front-end and back-end interactions.

(Note: These response time components do not overlap with each other.)

The model is configured in a two-level hierarchy, a high level and a detailed level. The use of a hierarchy allows a complex and detailed model that considers many components and involves many parameters to be constructed easily. Because of the hierarchical approach, the model also provides flexibility for modifications and extensions, and validation of separate submodels.

The high-level model assumes the decomposition of transaction response times, as described in the Response Time Components section, and models the

- Other delays, which include delays that result from concurrency control (e.g., waiting for locks) and waiting for messages

behavior of the transaction processing system by an open queuing system, as shown in Figure 10. The queuing system consists of servers and delay centers, which are connected in a

Two-level Approach

queuing network with the following assumptions:

- The front-end processing does not involve any disk I/O operation, and the load on the front-end systems is equally balanced.
- The back-end is a shared-memory multiprocessor system with symmetrical loads on all processors (or it can be simply a uniprocessor).

The front-end CPU is modeled as an M/M/1 queuing center, and the back-end CPU is modeled as an M/M /m queuing center. The transactions' CPU times on the front-end and back-end systems are assumed to be exponentially distributed (coefficient of variation equal to 1) due to the single type of transaction

in the benchmark. (Note: An approximation of M/G/m can be used to consider a coefficient of variation other than 1 for the back-end transaction CPU service time, especially in the multiprocessor case when the bus is highly utilized.) Database I/O, logging I/O, and other delays are modeled as delay centers,

- No intratransaction parallelism exists within individual transaction execution.
- No mutual dependency exists between transaction response time components.
- Transaction arrivals to the processors have a

  Poisson distribution.

These assumptions correspond to Digital's TPC Benchmark A testing methodology and implementation.

distribution. The major input parameters for this high-level model are the

- Number of front-end systems and the front-end CPU service time per transaction
- Number of CPUs in the back-end system and the back-end CPU service time per transaction
- Sum of the back-end database I/O response time, journaling I/O response time, and other delay times (i.e., the mean for the LOD delay center's 2-Erlang distribution)
- Response time constraint (in the form of x percentile less than y seconds)

with appropriate delay distributions. For the model of the TPC Benchmark A workload, the database I/O, journaling I/O, and other communication and synchronization delays are combined into one delay center, called the LOD delay center, which is represented by a 2-Erlang

The main result from the high-level model is the MQTh. This high-level model presents a global picture of the performance behavior and manifests the relationship between the most important parameters of the transaction processing system and MQTh.

Some of the input parameters in the high-level model are dynamic. The CPU service time of a transaction may vary with the throughput or number of processors, and the database I/O or other delays may also depend on the throughput. A good example of a dynamic model is a tightly coupled multiprocessor system, with one bus interconnecting the processors and with a shared common memory (e.g., a VAX 6000 Model 440 system). Such a system would run a single copy of the symmetrical multiprocessing operating system (e.g., the VMS system). The average CPU service time of transactions is affected by both hardware and software factors, such as

a. Hardware contention that results from conflicting accesses to the shared bus and main memory and that causes processor speed degradation and longer CPU service time.

b. Processor synchronization overhead that results from the serialization of accesses to shared data structures. Many operating systems use spin-locks as the mechanism for processor-level synchronization, and the processor spins (i.e., busy-waits) in the case of a conflict. In the model, the busy-wait overhead is considered to be part of the transaction code path, and such contention elongates the transaction CPU service time.

Four detailed-level submodels are used to account for the dynamic behavior of these parameters: CPU-cache-bus-memory, busy-wait, I/O group, and LOD.

The CPU-cache-bus-memory submodel consists of many low-level parameters associated with the workload, processor, cache, bus, and memory components of multiprocessor systems. It models these components by using a mixed queuing network model that consists of both open and closed chains, as shown in Figure 11. The most important output from this submodel is the average number of CPU clock cycles per instruction.

The busy-wait submodel models the spin-lock analysis to derive busy-wait time. The I/O grouping

contention that is associated with the two major VMS spin-locks, called SCHED and IOLOCK8. This submodel divides the state of a processor into several nonoverlapping states and uses probability

submodel models the group commit and group write mechanisms of the VAX Rdb /VMS relational database management system. This submodel affects the path length of transaction because of the amortization of disk I/O processing

among grouped transactions. The LOD submodel considers the disk I/O times and the lock contention of certain critical resources in the VAX Rdb/VMS system.

Integrating the Two Levels of the Model

The two levels of the model are integrated by using an iterative procedure outlined in Figure 12. It starts at the detailed-level submodels, with initial values for the MQTh, the transaction path length, the busy-wait overhead, and the CPU utilization.

By applying the initialized parameters to the submodels, the values of these parameters are refined and input to the high-level model. The output parameters from the high-level model are then fed back to the detailed-level submodels, and this iterative process continues until the MQTh converges. In most cases, convergence is reached within a few iterations.

Model Predictions

The back-end portion of the model was validated against measurement results from numerous DebitCredit benchmarks (Digital's precursor of the TPC Benchmark A) on many VAX computers with the VMS operating system, running VAX ACMS and VAX Rdb/VMS software. [5] With sufficient detailed parameters available (such as transaction instruction count, instruction cycle time, bus/memory access time, cache hit ratio), the model correctly end-to-end model to the TPC Benchmark A on two VAX platforms, the VAX 9000 Model 210 and the VAX 4000 Model 300 systems, and then compare the results. The benchmark environment and implementation are described in the TPC Benchmark A Implementation section of this paper.

Because both the VAX 9000 Model 210 and the VAX 4000 Model 300 systems are uniprocessor systems, there is no other processor contending for the processor-memory interconnect and memory

estimated the MQTh and many intermediate results for several multiprocessor VAX systems. The model was then extended to include the front-end systems. In this section, we discuss applying this complete subsystems. Such contention effects can therefore be ignored when modeling a uniprocessor system. The transaction processing performance prediction for the VAX 9000 Model 210 system is a successful

example of the application of our analytical model. We needed an accurate estimate of TPC Benchmark A performance on the VAX 9000 Model 210 system before a VAX 9000 system was actually available for testing. The high-level (MQTh) model was used with estimated values for the input parameters, LOD and transaction CPU service time. The estimated LOD was based on previous measurement observations from the VAX 6000 systems. The other parameter, back-end transaction CPU service time, was derived from the

- Timing information of the VAX 9000 CPU
- Memory access time and cache miss penalty of the VAX 9000 CPU
- Prediction of cache hit ratio of the VAX 9000 system under the TPC Benchmark A workload
- Transaction path length of the TPC Benchmark A implementation
- Instruction profile of the TPC Benchmark A implementation

The high-level model predicted a range of MQTh, with a high end of 70 TPS and with a strong probability that the

Additional predictions were made later, when an early prototype version of the VAX 9000 Model 210 system was available for testing. A variant of the DebitCredit benchmark, much smaller in scale and easier to run, was performed on the prototype system, with the emphasis on measuring the CPU performance in a transaction processing environment. The result was used to extrapolate the CPU service time of the TPC Benchmark A transactions on the VAX 9000 Model 210 system and to refine the early estimate. The results of these modifications supported the previous high-end estimate of performance of 70 TPS and refined the low-end performance to be 62 TPS. The final, audited TPC Benchmark A measurement result of the VAX 9000 Model 210 system showed 69.4 TPS, which closely matches the prediction. Table 8 compares the results from benchmark measurement and the analytical model outputs.

Table 8
Measurement Compared to Model Predictions

System	Measured MQTh	Modeled MQTh

high-end performance was achievable.

VAX 9000 Model 210	69.4
	70.0
VAX 4000 Model 300	21.5
	20.8

The VAX 4000 Model 300 TPC Benchmark A results were also used as a validation case. VAX 4000 Model 300

systems use the same CMOS chip as the VAX 6000 Model 400 series and the same 28-nanosecond (ns) CPU cycle time. However, in the VAX 4000 series, the CPU-memory interconnect is not the XMI bus but a direct primary memory interconnect. This direct memory interconnect results in fast main memory access. The processor, cache, and main memory subsystems are otherwise the same as in the VAX 6000 Model 400 systems. Therefore, the detailed-level model and associated parameters for the VAX 6000 Model 410 system can be used by ignoring the bus access time. The TPC Benchmark A measurement results are within 7 percent of the model prediction, which means that our assumption on the memory access time is acceptable.

Conclusion

Performance is one of the most important attributes in evaluating a transaction processing system. However, because of the complex nature of transaction processing systems, a universal assessment of transaction processing system performance is impossible. The performance of a transaction processing system is performance by different vendors. But it is only one transaction processing benchmark that represents a limited class of applications. When evaluating transaction processing systems performance, a good understanding of the targeted application environment and requirements is essential before using any available benchmark result. Additional benchmarks that represent a broader range of commercial applications are expected to be standardized by the Transaction Processing Performance Council (TPC) in the coming years.

Performance modeling is an attractive alternative to benchmark measurement because it is less expensive to perform and results can be compiled more quickly. Modeling provides more insight into the behavior of system components that are treated as black boxes in most measurement experiments. Modeling helps system designers to better understand performance issues and to discover existing or potential performance problems. Modeling also provides solutions for improving performance by

workload dependent, configuration dependent, and implementation dependent. A standard benchmark, like TPC Benchmark A, is a step toward a fair comparison of transaction processing

modeling different tuning or design alternatives. The analytical model presented in this paper was validated and used extensively in many engineering performance studies. The model also helped

the benchmark process to size the hardware during preparation (e.g., the number of RTE and front-end systems needed, the size of the database) and to provide an MQTh goal as a sanity check and a tuning aid. The model could be extended to represent additional distributed configurations, such as shared-disk and "shared-nothing" back-end transaction processing systems, and could be applied to additional transaction processing workloads.

Acknowledgments

The Digital TPC Benchmark A implementation and measurements are the result of work by many individuals within Digital. The authors would like especially to thank Jim McKenzie, Martha Ryan, Hwan Shen, and Bob Tanski for their work in the TPC Benchmark A measurement experiments; and Per Gyllstrom and Rabah Mediouni for their contributions to the analytical model and validation.

References

1. Transaction Processing Performance Council, TPC Benchmark A Standard Specification (Menlo Park, CA: Waterside Associates, November 1989).
2. Transaction Processing Systems Handbook (Maynard: Digital Equipment Corporation, Order No. EC-H0650-57, 1990).
3. TPC Benchmark: A Report for the VAX 9000 Model 210 System (Maynard: Digital Equipment Corporation, Order No. EC-N0302-57, 1990).
4. TPC Benchmark: A Report for the VAX 4000 Model 300 System (Maynard: Digital Equipment Corporation, Order No. EC-N0301-57, 1990).
5. L. Wright, W. Kohler, and W. Zahavi, "The Digital DebitCredit Benchmark: Methodology and Results," Proceedings of the International Conference on Management and Performance Evaluation of Computer Systems (December 1989): 84-92.