Tools and Techniques for Preliminary Sizing of Transaction Processing Applications

By William Z. Zahavi, Frances A. Habib, and Kenneth J. Omahen

Abstract

Sizing transaction processing systems correctly is a difficult task. By nature, transaction processing applications are not predefined and can vary from the simple to the complex. Sizing during the analysis and design stages of the application development cycle is particularly difficult. It is impossible to measure the resource requirements of an application which is not yet written or fully implemented. To make sizing easier and more accurate in these stages, a sizing methodology was developed that uses measurements from systems on which industry-standard benchmarks have been run and employs standard systems analysis techniques for acquiring sizing information. These metrics are then used to predict future transaction resource usage.

Introduction

The success or failure of a business, based on the level of performance the application provides. In transaction processing, poor application performance can translate directly into lost revenues.

The risk of implementing a transaction processing application that performs poorly can be minimized by estimating the proper system size in the early stages of application development. Sizing estimation includes configuring the correct processor and proper number of disk drives and controllers, given the characteristics of the application.

The sizing of transaction processing systems is a difficult activity. Unlike traditional applications such as mail, transaction processing applications are not predefined. Each customer's requirement is different and can

The transaction processing marketplace is dominated by commercial applications that support businesses. These applications contribute substantially

vary from simple to complex. Therefore, Digital chose to develop a sizing methodology that specifically meets the unique requirements of transaction processing

customers. The goal of this effort was to develop sizing tools and techniques that would help marketing groups and design consultants in recommending configurations that meet the needs of Digital's customers. Digital's methodology evolved over time, as experience was gained in dealing with the real-world problems of transaction processing system sizing.

The development of Digital's transaction processing sizing methodology was guided by several principles. The first principle is that the methodology should rely heavily upon measurements of Digital systems running industry-standard transaction processing benchmarks. These benchmarks provide valuable data that quantifies the performance characteristics of different hardware and software configurations.

The second principle is that systems analysis methodologies should be used to provide a framework for acquiring sizing information. In particular, a multilevel view of a customer's business is adopted. This approach recognizes that a manager's view of the business

The third principle is that the sizing methodology must employ tools and techniques appropriate to the current stage of the customer's application design cycle. Early in the effort to develop a sizing methodology, it was found that a distinction must be made between preliminary sizing and sizing during later stages of the application development cycle. Preliminary sizing occurs during the analysis and design stages of the application development cycle. Therefore, no application software exists which can be measured. Application software does exist in later stages of the application development cycle, and its measurement provides valuable input for more precise sizing activities.

For example, if a customer is in the analysis or design stages of the application development cycle, it is unlikely that estimates can be obtained for such quantities as paging rates or memory usage. However, if the application is fully implemented, then tools such as the VAXcluster Performance Advisor (VPA) and the DECcp capacity planning products can be used for sizing. These

functions performed by an organization is different from a computer analyst's view of the transaction processing activity. The sizing methodology should accommodate both these views.

tools provide facilities for measuring and analyzing data from a running system and for using the data as input to queuing models.

and Techniques for Preliminary Sizing of Transaction Processing Applications

The term sizing, as used in this paper, refers to preliminary sizing. The paper presents the metrics and algebra used in the sizing process for DECtp applications. It also describes the individual tools developed as part of Digital's transaction processing sizing effort.

Sizing

The purpose of sizing tools is twofold. First, sizing tools are used to select the appropriate system components and to estimate the performance level of the system in terms of device utilization and user response times. Second, sizing tools bridge the gap between business specialists and computer specialists. This bridge translates the business units into functions that are performed on the system and, ultimately, into units of work that can be quantified and measured in terms of system resources.

In the sections that follow, a number of important elements of the sizing methodology are described. The first of these elements is the platform on which the transaction processing system will be implemented. It is assumed that the used to describe the work performed by the business. The Sizing Metrics and Sizing Formulas sections describe the algorithms that use platform and business metric information to perform transaction processing system sizing.

Platforms

The term platform is used in transaction processing sizing methodology to encompass general customer preferences for the hardware and software upon which the transaction processing application will run.

The hardware platform specifies the desired topology or processing style. For example, processing style includes a centralized configuration and a front-end and back-end configuration as valid alternatives. The hardware platform may also include specific hardware components within the processing style. (In this paper, the term processor refers to the overall processing unit, which may be composed of multiple CPUs.)

The software platform identifies the set of layered products to be used by the transaction processing application,

customer will supply general preferences for the software and hardware configuration as part of the platform information. The Levels of Business Metrics section details the multilevel approach with each software product identified by its name and version number. In the transaction processing environment, a software platform is composed of the transaction processing monitor, forms manager,

database management system, application language, and operating system.

Different combinations of software platforms may be configured, depending on the hardware platform used. A centralized configuration contains all the software components on the same system. A distributed system is comprised of a front-end processor and a back-end processor; different software platforms may exist on each processor.

Levels of Business Metrics

The term business metrics refers collectively to the various ways to measure the work associated with a customer's business. In this section, various levels of business metrics are identified and the relationship between metrics at different levels is described. [1] As mentioned earlier, the levels correspond to the multilevel view of business operation typically used for systems analysis. The organization or personnel most interested in a metric in relation to its business operation is noted in the discussion of each metric.

usage requires that a set of metrics be defined. These metrics reflect the business activity and the system load. The business metrics are the foundation for the development of several transaction processing sizing tools and for a consistent algebra that connects the business units with the computer units.

The business metrics are natural forecasting units, business functions, transactions, and the number of I/Os per transaction. The relationship among these levels is shown in Figure 1. In general, a business may have one or more natural forecasting units. Each natural forecasting unit may drive one or more business functions. A business function may have multiple transactions, and a single transaction may be activated by different business functions. Every transaction issues a variety of I/O operations to one or more files, which may be physically located on zero, one, or more disks. This section discusses the business

The decomposition of the business application requirements into components that can be counted and quantified in terms of resource metrics but does not discuss the physical distribution of I/Os across disks, which is an implementation-specific item.

A natural forecasting unit is a macrolevel indicator of business volume. (It is also called a key volume indicator.) A business generally uses a volume indicator to measure the level of success of the business. The volume is often measured

in time intervals that reflect the business cycle, such as weekly, monthly, or quarterly. For example, if business volume indicators were "number of ticket sales per quarter," or "monthly production of widgets," then the corresponding natural forecasting units would be "ticket sales" and "widgets." Natural forecasting units are used by high-level executives to track the health of the overall business.

Business functions are a logical unit of work performed on behalf of a natural forecasting unit. For example, within an airline reservation system, a common business function might be "selling airline tickets." This business function may consist of multiple interactions with the computer (e.g., flight inquiry, customer credit check). The completion of the sale terminates the business function, and "airline ticket" acts as a natural forecasting unit for the enterprise selling the tickets. The measurement metric for business functions is the number of business function occurrences per hour. Business functions may be used by middle-level managers to track

by a user. Each of the interactions mentioned in the above business function is a transaction. The measurement metric for a transaction is the number of transaction occurrences per business function. Transactions may be used by low-level managers to track the activity of their groups.

The bulk of commercial applications involves the maintaining and moving of information. This information is data

that is often stored on permanent storage devices such as rotational disks, solid state disks, or tapes. An I/O operation is the process by which a transaction accesses that data. The measurement metric for the I/O profile is the number of I/O operations per transaction. I/O operations by each transaction are important to programmers or system analysts.

In addition to issuing I/Os, each transaction requires a certain amount of CPU time to handle forms processing. (Forms processing time is not illustrated in Figure 1.) The measurement metric for forms processing time is the expected number of fields. The number of input and output fields per form

the activity of their departments.

A transaction is an atomic unit of work for an application, and transaction response time is the primary performance measure seen are important metrics for users of a transaction processing application or programmer/system analysts.

Tools and Techniques for Preliminary Sizing of Transaction Processing Applications

By collecting information about a transaction processing application at various levels, high-level volume indicators are mapped to low-level units of I/O activity. This mapping is fundamental to the transaction processing sizing methodology.

Performance goals play a particularly important role in the sizing of transaction processing systems. [2] The major categories of performance goals commonly encountered in the transaction processing marketplace are bounds for

- Device utilization(s)
- Average response time for transactions
- Response time quantiles for transactions

For example, a customer might specify a required processor utilization of less than 70 percent. Such a constraint reflects the fact that system response time typically rises dramatically at higher processor utilizations. A common performance goal for response time is to use a transaction's average response time and response time quantiles. For example, the proposed directly into decreased productivity and lost revenues.

When a customer generates a formal Request For Proposal (RFP), the performance goals for the transaction processing system typically are specified in detail. The specification of goals makes it easier to define the performance bounds. For customers who supply only general performance goals, it is assumed that the performance goal takes the form of bounds for device utilizations.

Overall response time consists of incremental contributions by each major component of the overall system:

- Front-end processor
- Back-end processor
- Communications network
- Disk subsystem

A main objective in this approach to sizing was to identify and use specific metrics that could be easily counted for each major component. For instance, the number of fields per form could be a metric used for sizing front-end processors because that number is specific

system should have an average response time of x seconds, with 95 percent of all responses completing in less than or equal to y seconds, where x is less than y. Transaction response times are crucial for businesses. Poor response times translate

and easily counted. As the path of a transaction is followed through the overall system, the units of work appropriate for each component become clear. These units become the metrics for sizing that particular component. The focus of this paper is on processor sizing

with bounds on processor utilization. Processors generally constitute the major expense in any proposed system solution. Mistakes in processor sizing are very expensive to fix, both in terms of customer satisfaction and cost.

Sizing Metrics

Transaction processing applications permit a large number of users to share access to a common database crucial to the business and usually residing on disk memory. In an interactive transaction processing environment, transactions generally involve some number of disk I/O operations, although the number is relatively small compared to those generated by batch transaction processing applications. CPU processing also is generally small and consists primarily of overhead for layered transaction processing software products. Although these numbers are small, they did influence the sizing methodology in several ways.

Ratings for relative processor capacity in a transaction processing environment were developed to reflect the ability of a processor to support disk I/O activity (as observed generated by a transaction provides a good prediction of the required amount of CPU processing. [3] Numerous industry-standard benchmark tests for product positioning were run on Digital's processors. These processors were configured as back-end processors in a distributed configuration with different software platforms.

The base workload for this benchmark testing is currently the Transaction Processing Performance Council's TPC Benchmark A (TPC-A, formerly the DebitCredit benchmark). [4,5,6] The most complete set of benchmark testing was run under Digital's VAX ACMS transaction processing monitor and VAX Rdb/VMS relational database. Therefore, results from this software platform on all Digital processors were used to compute the first sizing metric called the base load factor.

The base load factor is a high-level metric that incorporates the contribution by all layered software products on the back-end processor to the total CPU time per I/O operation. Load factors are computed by dividing the total CPU utilization by the number of achieved disk I/O operations per second.

in benchmark tests). In addition, empirical studies of transaction processing applications showed that, for purposes of preliminary sizing, the number of disk I/Os

(The CPU utilization is normalized in the event that the processor is a Symmetrical Multiprocessing [SMP] system, to ensure that its value falls within the range of 0 to 100 percent.) The calculation

of load factor yields the total CPU time, in centiseconds (hundredths of seconds), required to support an application's single physical I/O operation.

The base load factors give the CPU time per I/O required to run the base workload, TPC-A, on any Digital processor in a back-end configuration using the ACMS/Rdb. The CPU time per I/O can be estimated for any workload. This generalized metric is called the application load factor.

To relate the base load factors to workloads other than the base, an additional metric was defined called the intensity factor. The metric calculation for the intensity factor is the application load factor divided by the base load factor. The value in using intensity factors is that, once estimated (or calculated for running applications), intensity factors can be used to characterize any application in a way that can be applied across all processor types to estimate processor requirements.

Intensity factors vary based on the software platform used. If a selected DECtp software platform.

To estimate an appropriate intensity factor for a nonexistent application, judgment and experience with similar applications are required. However, measured cases from a range of DECtp applications shows relatively little variation in intensity factors. Guidelines to help determine intensity factors are included in the documentation for Digital's internally developed transaction processing sizing tools.

The work required by any transaction processing application is composed of two parts: the application /database and the forms management. This division of work corresponds to what occurs in a distributed configuration, where the forms processing is off-loaded to one or more front-end processors. Load factors and intensity factors are metrics that were developed to size the application/database. To estimate the amount of CPU time required for forms management, a forms-specific metric is required. For a first-cut approximation, the expected number of (input) fields is used as the sizing metric.

software platform other than a combined VAX ACMS and VAX Rdb/VMS platform is selected, the estimate of the intensity factor must be adjusted to reflect the resource usage characteristics of the	This number is obtained easily from the business-level description of the application.

and Techniques for Preliminary Sizing of Transaction Processing Applications

This section describes the underlying algebra developed for processor selection. Different formulas to estimate the CPU time required for both the application /database and forms management were developed. These formulas are used separately for sizing back-end and front-end processors in a distributed configuration. The individual contributions of the formulas are combined for sizing a centralized configuration.

The application/database is the work that takes place on the back-end processor of a distributed configuration. It is a function of physical disk accesses. To determine the minimal CPU time required to handle this load, processor utilization is used as the performance goal, setting up an inequality that is solved to obtain a corresponding load factor. The resulting load factor is then compared to the table of base load factors to obtain a recommendation for a processor type. To reinforce this dependence of load factors on processor types, load factor x refers to the associated processor type x in the following time per transaction, expressed in centiseconds per transaction. By multiplying this product by the transactions per second rate, an expression for processor utilization is derived. Thus processor utilization (expressed as a percentage scaled between 0 and 100 percent) is the number of transactions per second, times the number of I/Os per transaction, times load factor x, times the intensity factor.

The performance goal is a CPU utilization that is less than the utilization specified by the customer. Therefore, the calculation used to derive the load factor is the utilization percentage provided by the customer, divided by the number of transactions per second, times the number of I/Os per transaction, times the intensity factor.

Once computed, the load factor is compared to those values in the base load factor table. The base load factor equal to or less than the computed value is selected, and its corresponding processor type, x, is returned as the minimal processor required to handle this workload.

The four input parameters that need to be estimated for inclusion in this

calculations.

One method for estimating the average CPU time per transaction is to multiply the number of I/Os per transaction by the load factor x and the intensity factor. This yields CPU inequality are

- Processor utilization performance goal (traditionally set at around 70 percent, but may be set higher for Digital's newer, faster processors)

- Target transactions per second (which may be derived from Digital's multilevel mapping of business metrics)
- I/Os per transaction (estimated from application description and database expertise)
- Intensity factor (estimated from experience with similar applications)

Note: Response time performance goals do not appear in this formula. This sizing formula deals strictly with ensuring adequate processor capacity. However, these performance parameters (including the CPU service time per transaction) are fed into an analytic queuing solver embedded in some of the transaction processing sizing tools,

which produces estimates of response times.

Forms processing is the work that occurs either on the front-end processor of a distributed configuration or in a centralized configuration.

It is not a function of physical disk accesses; rather, forms processing is CPU intensive. To estimate the CPU time (in

where y equals the CPU time for forms processing; a equals the CPU time per form per transaction instance, depending on the forms manager used; b equals the CPU time per field per transaction instance, depending on the forms manager used; z equals the expected number of fields; and c equals the scaling ratio, depending on the processor type.

This equation was developed by feeding the results of controlled forms testing into a linear regression model to estimate the CPU cost per form and per field (i.e., a and b. The multiplicative term, c, is used to eliminate the dependence of factors a and b on the hardware platform used to run these tests.

Sizing Tools

Several sizing tools were constructed by using the

above formulas as starting points. These tools differ in the range of required inputs and outputs, and in the expected technical sophistication of the user.

The first tool developed is for quick, first-approximation processor sizing. Currently embodied as a DECalc spreadsheet,

seconds) required for forms processing, the following simple linear equation is used:

$$y = c(a + bz)$$

with one screen for processor selection and one for transactions-per-second sensitivity analysis, it can handle back-end, front-end, or centralized sizing. The first screen shows the range of processors required, given the target processor utilization, target transactions

per second, expected number of fields, and the possible intensity factors and number of I/Os per transaction. (Because the estimation of these last two inputs generally involves the most uncertainty, the spreadsheet allows the user to input a range of values for each.) The second screen turns the analysis around, showing the resulting transaction-per-second ranges that can be supported by the processor type selected by the user, given the target processor utilization, expected number of fields, and possible intensity factors and number of I/Os per transaction.

The basic sizing formula addresses issues that deal specifically with capacity but not with performance. To predict behavior such as response times and queue lengths, modeling techniques that employ analytic solvers or simulators are needed. A second tool embeds an analytic queuing solver within itself to produce performance estimates. This tool is an automated system (i.e., a DECtp application) that requests information from the user according to the multilevel workload characterization product characteristics (e.g., processor and disk) and measured DECtp applications. The user can search through the measured cases to find a similar case, which could then be used to provide a starting point for estimating key application parameters. The built-in product characteristics shield the user from the numeric details of the sizing algorithms.

A third tool is a spin-off from the second tool. This tool is a standalone analytic queuing solver with a simple textual interface. The tool is intended for the sophisticated user and assumes that the user has completed the level of analysis required to be able to supply the necessary technical input parameters. No automatic table lookups are provided. However, for a completely characterized application, this tool gives the sophisticated user a quick means to obtain performance estimates and run sensitivity analyses. The complete DECtp software platform necessary to run the second tool is not required for this tool.

Data Collection

To use the sizing tools

methodology. This starts from general business-level information and proceeds to request successively more detailed information about the application. The tool also contains a knowledge base of Digital's fully, certain data must be available, which allows measured workloads to be used to establish the basic metrics. Guidance in sizing unmeasured transaction processing applications is highly dependent on

developing a knowledge base of real-world transaction processing application descriptions and measurements. The kinds of data that need to be stored within the knowledge base require the data collection tools to gather information consistent with the transaction processing sizing algebra.

For each transaction type and for the aggregate of all the transaction types, the following information is necessary to perform transaction processing system sizing:

a. CPU time per disk I/O

b. Disk I/O operations per transaction

c. Transaction rates

d. Logical-to-physical disk I/O ratio

The CPU to I/O ratio can be derived from Digital's existing instrumentation products, such as the VAX Software Performance Monitor (SPM) and VAXcluster Performance Advisor (VPA) products. [7] Both products can record and store data that reflects CPU usage levels and physical disk I/O rates.

The DECtrace product collects event-driven data. It can collect DECtrace product can be used to track the rate at which events occur.

The methods for determining the logical-to-physical disk I/O ratio per transaction remain open for continuing study. Physical disk I/O operations are issued based on logical commands from the application. The find, update, or fetch commands from an SQL program translate into from zero to many thousands of physical disk I/O operations, depending upon where and how data is stored. Characteristics that affect this ratio include the length of the data tables, number of index keys, and access methods used to reach individual data items (i.e, sequential, random).

Few tools currently available can provide data on physical I/O operations for workloads in the design stage. A knowledge base that stores the logical-to-physical disk I/O activity ratio is the best method available at this time for predicting that value. The knowledge base in the second sizing tool is beginning to be populated with application descriptions that include this type of information. It is anticipated that,

resource items from layered software products, including VAX ACMS monitor, the VAX Rdb/VMS and DBMS database systems, and if instrumented, from the application program itself. As an event collector, the as this tool becomes widely used in the field, many more application descriptions will be stored in the knowledge base. Pooling individual application experiences into one central repository will create a valuable

source of knowledge that may be utilized to provide better information for future sizing exercises.

Acknowledgments

The authors would like to acknowledge our colleagues in the Transaction Processing

Systems Performance Group whose efforts led to the development of these sizing tools, either through product characterization,

system support, objective critique, or actual tool development. In particular, we would like to acknowledge the contributions made by Jim Bouhana to the development of the sizing methodology and tools.

References

1. W. Zahavi, and J. Bouhana, "Business-Level Description of Transaction Processing Applications," CMG '88 Proceedings (1988): 720-726.
2. K. Omahen, "Practical Strategies for Configuring Balanced Transaction Processing Systems," IEEE COMPCON Spring '89 Proceedings (1989): 554-559.
3. W. Zahavi, "A First Approximation Sizing Technique-The I/O Operation as a Metric of CPU Power," CMG '90 Conference Proceedings (forthcoming Dececember 10-14, 1990).
4. "TPC BENCHMARK A-Standard Specification," Transaction Processing Performance Council (November 1989).
5. "A Measure of Transaction Processing Power," Datamation, Vol. 31, No. 7 (April 1, 1985): 112-118.
6. L. Wright, W. Kohler, W. Zahavi, "The Digital DebitCredit Benchmark: Methodology

and Results," CMG '89 Conference Proceedings (1989): 84-92.
7. F. Habib, Y. Hsu, K. Omahen, "Software Measurement Tools for VAX/VMS Systems," CMG Transactions (Summer 1988): 47-78.