Database Buffer Size Investigation for OLTP Workloads

Thin-Fong Tsuei
Sun Microsystems, Inc.
tsuei@eng.sun.com

Allan N. Packer
Sun Microsystems, Inc.
allanp@eng.sun.com

Keng-Tai Ko
Sun Microsystems, Inc.
keng.ko@eng.sun.com

Abstract

It is generally accepted that On-Line Transaction Processing (OLTP) systems benefit from large database memory buffers. As enterprise database systems become larger and more complex, hardware vendors are building increasingly large systems capable of supporting huge memory configurations. Database vendors in turn are developing buffer schemes to exploit this physical memory.

How much will these developments benefit OLTP workloads? Through empirical studies on databases sized comparably to those seen in the real-world, this paper presents the characteristics of an industry-standard OLTP benchmark as memory buffer size changes. We design the experiments to investigate how the database size, the buffer size and the number of CPUs impact performance, in particular the throughput and the buffer hit rate on Symmetric Multiprocessor Systems. The relationships of these major database attributes are plotted and key observations are summarized. We discuss how these relationships change as the number of CPUs changes. We further quantify the relationships: 1) between database buffer data hit rate, buffer size and database size, 2) between throughput, buffer data hit rate and database size and 3) between throughput and number of CPUs. Algorithms, rules-of-thumb and examples are presented for predicting performance, sizing memory and making trade-offs between adding more memory and increasing the number of CPUs.

1. Introduction

The argument that "bigger is better" is often heard when configuring database buffers for On-Line Transaction Processing (OLTP) systems. Database buffers act as memory caches for the database pages stored on disks. Expensive disk accesses are reduced because pages with locality of reference will reside in the buffers over repeated reads and writes[ChDe85, EfWH84, Ston81]. OLTP workloads are characterized by random data accesses with some hot spots. These characteristics benefit from having memory buffers. Database buffer size and how the buffers are allocated to the databases have considerable impact on the performance of OLTP systems.

As real-life enterprise database systems become larger as well as more complex, hardware platform vendors are announcing increasingly large systems capable of supporting huge memory configurations. Addressing schemes that access memory configurations far larger than 4G bytes are being developed. Database vendors, keeping pace with advancements in technology, are developing buffer schemes to exploit such vast physical memory sizes.

With these technology developments and recent signs that memory prices are falling, it is feasible for OLTP servers to take advantage of larger memory buffers. We expect that the marginal benefit of larger database buffer will decrease as the buffer size (number of buffer pages) increases. It is, however, unclear how performance changes relate to variables such as database size, buffer size and the number of CPUs for a database server. In particular, for a given set of these variables, is it possible to estimate or establish a rule-of-thumb for "how big is enough" for a memory buffer. Alternatively, if one of these variables has changed, how should the other variables be adjusted to maintain or improve performance. One practical example is that the performance of database systems usually degrades as the database sizes grow over time, and re-configuration of the systems becomes necessary.

This paper investigates the impact of buffer size on the performance of the TPC-C (Transaction Processing Performance Council Benchmark C) workload [TPC95, Gray93]. TPC-C is a well-known benchmark that emulates read-only and update intensive transactions found in complex OLTP application environments. It has been widely used in the database server industry as a basis of server performance analysis and platform comparisons. This paper provides insight into the behavior of TPC-C with varying memory buffer size, database size and number of CPUs. We design the experiments to examine how throughput and buffer hit rates depend on these variables. A fully-implemented TPC-C-like benchmark is run with a front-end client machine driving a commercial shared memory multiprocessor server to its maximum capacity. Unlike other studies based on limited database sizes [LeDi92, DaYC93], we build a database sized comparably (45.7GB corresponding to 525 warehouses in TPC-C benchmark) to those seen in the real-world OLTP systems on 136 disks on the server, and vary the database size from relatively small to this size. A commercial database engine, IBM's DB2 Common Server Version 2.1.1 [DB2Info], is used in the studies. We vary the buffer size from a typical initial configuration at customer sites to about the maximum size available on our test configuration.

The investigation is divided into two parts. In the first study the number of CPUs is fixed while the database size and the buffer size are varied. Characteristics of the throughput, hit rates of data accesses (data hit rate) and index accesses (index hit rate) to the buffer are summarized. In the second study, the same analysis is repeated on different number of CPUs and the results are compared with those in the first study. The first study confirmed our intuition that the throughput depends on the ratio of the buffer size to the database size. At the same buffer size, the throughput of a

SIGMOD '97 AZ,USA

smaller database benefits more quickly from additional memory buffers. As the marginal benefit decreases, it is much more expensive to obtain the same benefit at larger buffer sizes than at smaller buffer sizes. The optimal buffer size is relatively small compared to the database size. We analyze the index to data access ratio as the buffer size changes. In the second study, we find that the data hit rate remains unchanged across a varying number of CPUs. The relationship between throughput and data hit rate at different number of CPUs varies according to some meaningful ratio.

We further quantify the relationships of the variables evaluated. Experimental data collected is analyzed statistically. The data hit rate is found to be a linear function of $ln(BufferSize/DatabaseScale)$ and the database size. A linear model for estimating throughput based on data hit rate and database size is derived. We present algorithms for predicting performance as buffer size and number of CPUs change. The model's predictions are validated with measurements. We show examples of applying these models and algorithms to estimate performance and configure systems.

While the TPC-C benchmark used in these studies may not be fully representative of OLTP workloads in commercial systems, the observations presented provide a case study for other OLTP systems. The analysis method used in this study may be applied to similar workloads.

The remainder of this paper is organized as follows. Section 2 provides background information on the TPC-C workload and related studies. Section 3 describes the experimental design and configuration. Section 4 discusses the results of the first study and the regression models developed. An algorithm is presented to predict transaction throughput that conforms to TPC-C requirements when only the memory buffer is changed. Section 5 presents the results of the second study. An algorithm which predicts performance as the number of CPUs and the database size changes is described. Section 6 concludes the paper and discusses application of the studies to other database platforms.

2. Background

The TPC-C benchmark is designed to model the activities of a wholesale supplier. It comprises order processing operations representative of an industry that manages, sells, or distributes products or services. Figure 1 shows the tables in a TPC-C database, and the specification for table population and relationships [TPC95]. The

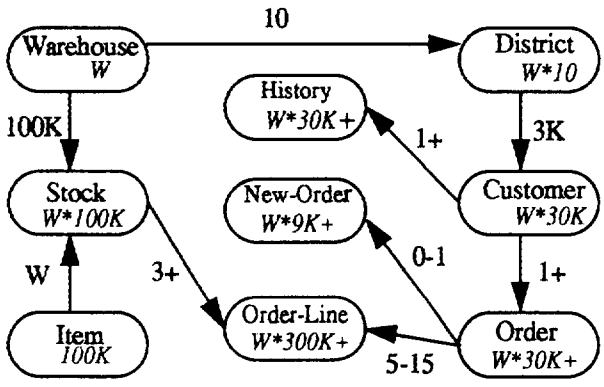


Figure 1. Database Entities, Cardinality and Relationships for TPC-C Benchmark.

database consists of 9 tables: the *warehouses* and their associated sale *districts*. All warehouses maintain *stocks* of the *items* sold. Each district handles its *customer* information and the *new orders* placed by the customers. Every district maintains the status of its *orders* and customer order *history*. Status of each item ordered is tracked through an *order-line* table.

Also shown in Figure 1 is the cardinality of each table. The number of warehouses is the base unit of scaling for TPC-C database size. All other tables, except item, scale with the number of warehouses according to pre-defined ratios (marked next to the arcs in Figure 1). In this paper, we use the terms database scale and database size inter-changeably.

There are 5 transactions in the TPC-C benchmark. The *new order* transaction initiates an order for an average of 10 items. It is a read-write transaction (select, update and insert), normally represents 43-44% of the total transactions submitted, and expects a quick response time. The throughput of this transaction is the performance metric measured by the benchmark. The *payment* transaction is a lighter-weight read-write transaction, composed of at least 43% of the total transactions. The *order status* transaction queries the status of a customer's last order, and is a read-only transaction. The *delivery* transaction processes a batch of pending oldest orders, one for each district for a given warehouse. The transaction is read-write, executed in deferred mode and has a relaxed response time requirement. The *stock level* transaction represents a heavy read-only transaction. Items in the last 20 orders in a district that fall below the stock threshold are counted. The order status, delivery and stock level transactions *each* comprises at least 4% of the total submitted transactions. The performance metric is Transactions Per Minute (TPM) of completed new-order transactions that arrived at the server during the measurement interval. Readers should refer to the TPC-C benchmark specifications [TPC95] for more details.

The TPC-C benchmark is designed to have data access skew. Accesses to customer, item and stock tables are skewed according to pre-specified non-uniform distributions. [LeDi93] evaluated the skewness in the data accesses. They analyzed the buffer hit rates with optimized page packing for hot tuples by simulating accesses to the database buffer.They reported that the largest skew is seen in the stock table. 84% of the accesses to the stock table go to about 20% of the hottest stock tuples, and at the page level, 75% of the accesses go to the hottest 25% of the pages. A smaller skew is seen in the customer table. Assuming a different buffer for each table, they developed a model to predict the throughput for single and multi-node systems using the miss rates from the buffer simulation.

While the work in [LeDi93] is based on simulation and static analysis on accesses to the tables, our work is based mostly on empirical studies by running a fully implemented TPC-C benchmark on a multiprocessing server system with a front end client machine driving the server to its maximum capacity. The configuration in our experiments is much larger and closer to the database sizes seen in real-world OLTP systems. Leutenegger and Dias evaluate the price/performance improvement by packing hot tuples in the same page. We analyze the performance with increasing memory buffer sizes and provide algorithms for price/performance considerations between increasing memory size and adding CPUs.

Database buffer hit rates have been studied either with trace driven simulations[LeDi93], access characteristics derived from the reference traces [DaYC93], or analytically based on indepen-

dent reference model assumption [DaTo90]. For the purpose of simply predicting the hit rate and the transaction throughput, the regression approach used in this paper avoids the simplification assumptions in analytical models and also avoids the complexities of trace-based approaches.

Simple models are frequently used in estimating the aggregate processing capacity of the CPUs in multiprocessor systems [Gunt96]. These models use one parameter to estimate the processing capacity with increasing number of CPUs. The well-known Amdahl's law is based on the serial fraction of a workload [Amda67]. Another approach is to use a geometric series to model the processing capacity contributed by each CPU. The parameter in the series represents the fraction of a CPU capacity expected from an additional CPU. Gunther has proposed a quadratic model on number of processors. The model has a parameter representing the fraction of capacity loss due to overhead [Gunt93]. The parameters in all three models have a value between 0 and 1. [Gunt96] analyzes the underlying dynamics of these three models, and the workload characteristics for which each model is most applicable. In this paper, we will further discuss the use of the geometric model for analyzing the scaling of TPC-C performance with number of CPUs.

3. Experimental Design And Configurations

Several tuning or configuration variables are known to have a major effect on the throughput for an OLTP system. The main factors include the database size, the buffer size, the number of CPUs, the system bus and disk utilization. Database table locks were not observed to cause major contention except when the database scale was very small relative to number of concurrent accesses. Other architectural factors such as the CPU speed (clock speed), the internal and external CPU caches, and the disk speed are crucial to achieving peak throughput. It is not the purpose of this paper to study the impact of architectural factors. We concentrate mainly on the configuration variables and quantify their relationships under the TPC-C workload.

This study is based on step-by-step experiments to analyze how database size and number of CPUs affect the optimal buffer size. The test runs are designed to avoid bottlenecks caused by variables not being investigated. The experimental design is summarized below:

1) Since a saturated system bus can severely limit performance and distort the results, the number of CPUs was selected such that the bus utilization was moderately high but not saturated. In the measured runs, the peak performance was either limited by CPU cycles or I/O latency. System monitor tools were used to measure the bus and disk utilization. The maximum bus utilization was 70%, and the disk utilization was below 45%.

2) The investigation was divided into two parts. In the first study, the number of CPUs was fixed at 12, and the buffer size was varied for scales varying between 100 and 525. For each database scale, we started with a buffer size of 3.1GB. We then reduced the buffer size at intervals to 199MB at which point the bus utilization became moderately heavy for the larger databases. The system was tuned for best TPC-C throughput: new order transactions per minute (TPM). The characteristics of the TPM and data hit ratios with respect to buffer size and database scale were derived from these run results.
Because the point of diminishing returns for throughput was not reached even at the maximum buffer size for the larger scales, only smaller scales (100 to 250) were examined for *knee points* where the throughput plateaus begin in the graphs of Throughput vs. Buffer Size.

3) In the second study, we repeated the measurements for database scales 525 and 375 over the same range of buffer sizes for 4 and 8 CPUs. To confirm the trend we observed from these experiments, selected data points were measured for other scales. We investigated the change in the behavior observed in the first study as the number of CPUs changed.

4) All runs output (response time, transaction mix, etc.) were compliant to TPC-C specifications, except for the requirement that the TPM to warehouse ratio be within the range of 9 to 12.7. By intentionally allowing the TPM to exceed this range, we were able to gain more insight, and make equivalent comparisons during analysis.
Variations in measured peak throughputs between runs for each configuration are within 3%.

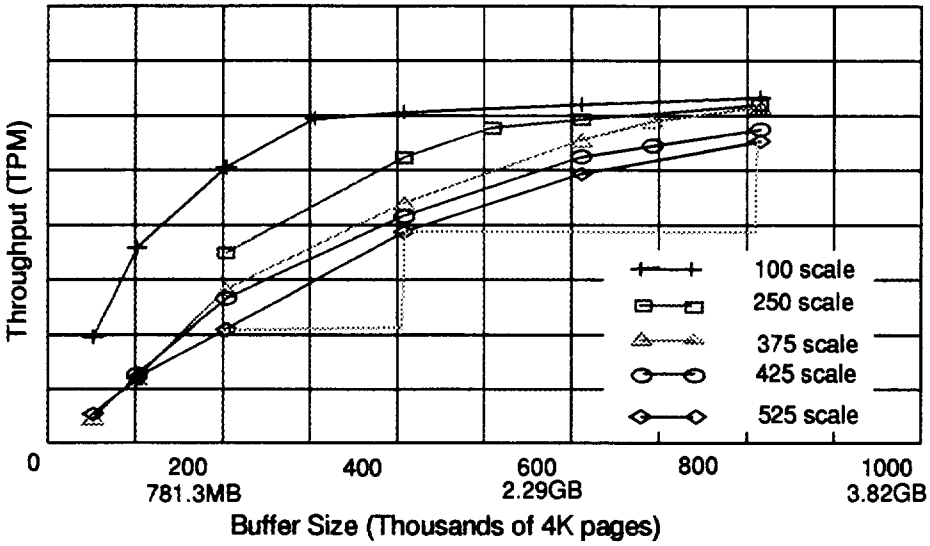
5) Since a fully configured TPC-C run with RTEs (remote terminal emulators) generating transactions requires a substantial amount of hardware for the database scales evaluated in this studies, a modified version of TPC-C was used in the experiments. In the modified version, all transaction requests were generated from a single client machine, and there was no think time between transactions. This stressed the server in the same way that a fully configured TPC-C hardware would. The throughput obtained using this method has been found to be very close to a fully configured TPC-C but at a fraction of the client hardware cost. The number of each type of database process (agents or backend processes) on the server for the modified version of TPC-C was about the same as those required for fully configured TPC-C runs.

The server used for the experiments was a shared memory multiprocessor system with 16 CPUs and 4 GB of memory. The database platform was IBM's DB2 for Solaris Version 2.1.1 [DB2Info]. A 525 warehouse database was built on the system across 136 disks. By default, DB2 distributes the data for each tablespace using a round robin algorithm across the available disks [DB2Adm]. Disk accesses were observed to be fairly balanced, and we did not have to re-layout the database as we varied the database scales. DB2 version 2.1.1 uses a single general buffer shared by the data and indices of all tables [MoHa92]. The page size of a buffer is 4KB. This study measured the overall buffer hit rate and the combined data and index hit rate of all tables.

4. Study I: Varying Buffer Size and Database Size on Fixed Number of CPUs

In this first study, the number of CPUs was fixed at 12. Measurements for database scales of 100, 250, 375, 425 and 525 were carried out. The buffer size was varied from 199MB to 3.1GB, corresponding to 51 to 815 thousand 4KB buffer pages. The bus

Figure 2. Throughput vs. Buffer Size
(Number of CPUs = 12)



utilization was below 60% at 3.1GB buffer size, and increased to about 70% at 199MB buffer size; a decreased buffer size and a larger database scale both increased the bus utilization. The maximum disk utilization was about 44%.

Results from smaller database scales, 100 to 250 at increments of 50 warehouses, were used to investigate the *knee points* in the plots of throughputs vs. buffer size. We found that for scales larger than 250, the curves did not reach a plateau (Figure 2).

The results of these experiments are summarized in section 4.1. Since we are constrained by TPC-C rules from presenting unaudited TPC-C numbers publicly, the TPM tick marks on the throughput axis in the graphs are not labelled. For the same reason, we do not present the coefficient values in the models for predicting the TPM. Nonetheless, absolute TPM is not required to analyze the trend and characteristics of performance, and a blank axis need not compromise our understanding of the behavior of OLTP workloads with respect to memory buffer size and database size.

While an understanding of performance characteristics is very helpful in predicting performance trends, it is even more useful to have models that quantify the way performance changes as database variables change. We apply regression analysis to the measured data. The regression models are described in section 4.1.4 and 4.1.6. Algorithms that use the models for performance predictions, and examples of their applications are discussed in section 4.2.

4.1. Observations and Modeling

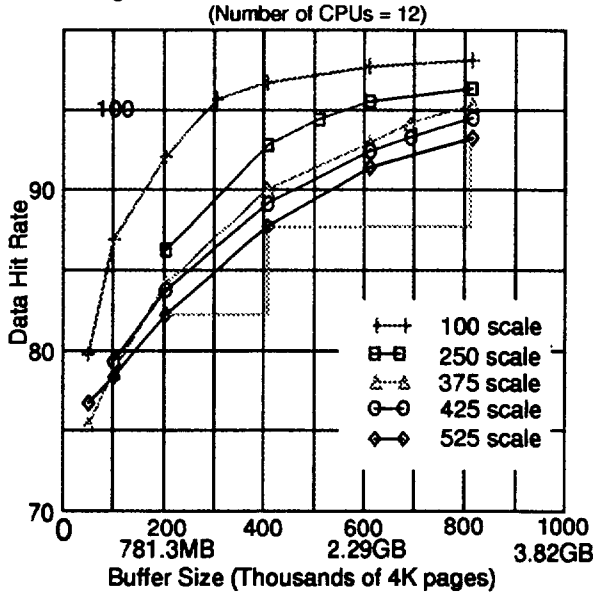
The throughput and buffer hit rates (overall, data and index) of various scales are compared and analyzed in this section. Major observations are summarized in the following subsections.

4.1.1. Given the same buffer size, a smaller database has a larger ratio of buffer size to database size, and the throughput benefit is greater from additional buffers until the plateau is reached.

Before the plateau, at any buffer size, the larger the ratio of the buffer size to database size the greater the benefit from adding the same amount of memory buffers. Figure 2 plots Throughput vs. Buffer Size for the scales of 100, 250, 375, 425 and 525. It shows that with a smaller database size, the throughput increases more quickly (a steeper slope) than with a larger scale, and reaches the plateau much sooner as the buffer size increases.

One might expect that for a given buffer size, a larger database will benefit more from increasing buffer size than a smaller database because of its higher miss rate. However, for a given buffer size, a larger portion of a small database can be captured in the buffer. 512MB of memory can ideally contain almost 6% of the database for a 100 scale database compared to only 1% for a 525 scale database. Thus, a smaller database size has a

Figure 3. Data Hit Rate vs. Buffer Size
(Number of CPUs = 12)



greater improvement in the buffer hit rate. This is shown in the graph for Data Hit Rate vs. Buffer Size in Figure 3. The curves show a similar trend as in Figure 2.

4.1.2. Index accesses occur at about twice the rate of data accesses, but increasing memory buffer size has a greater impact on data accesses.

Figure 4 compares the Overall Buffer Hit Rate, the Index Hit Rate and the Data Hit Rate as the buffer size increases. Only hit rates for 100 and 525 scale databases are shown in the figure but plots of other scale databases all show similar trends. The index hit rates are above 90%, and dominate the overall hit rates because of greater number of index accesses. Table 1 shows the ratio of index to data reads for logical (all buffer reads) and physical (buffer misses) reads. While the index shows twice the number of logical reads compared to data, there are more physical reads to the disks for data. It is the absolute number of physical I/Os that impacts the performance. Table 1 also shows the ratio of the number of I/Os reduced for index and data when the buffer size is increased from 102 to 815 thousand 4k pages. The reduction in physical I/Os for data reads is at least 38% greater than that for index. Hence increasing the buffer size has a greater impact on the data hit rate.

Database Scale	Buffer Size (thousands 4k pages)	Logical index/data read	Physical data/index reads	No. of Phy Read reduced data/index
100	815	2.3	9.2	1.9
	102	2.3	2.1	
525	815	2.0	4.9	1.38
	102	2.0	1.8	

Table 1. Comparison of Logical and Physical Index and Data Reads.

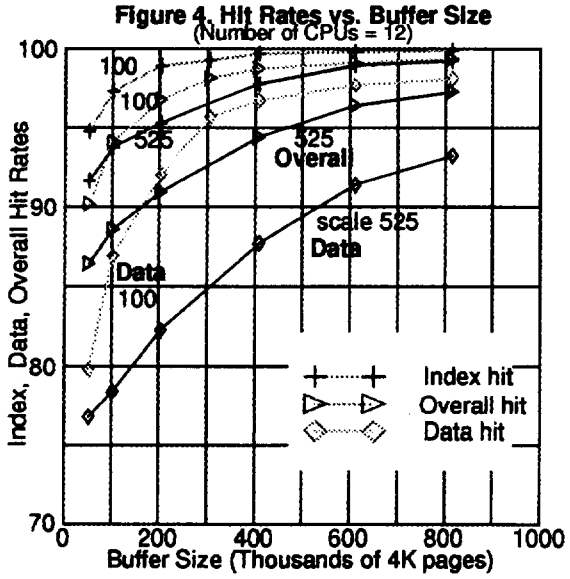
4.1.3. The optimal memory buffer size represents a relatively small fraction of the database size, but for large databases, this optimal memory buffer can be quite large.

The knee point where a plateau begins signifies the optimal memory buffer size for the given database scale. Figure 5 shows TPM vs. Buffer Size and the data hit rates at the knee points. We define a *knee point* in this experiment to be the point where the benefit in TPM becomes less than 2% peak TPM per thousand additional buffer pages. The benefit derived by increasing the buffer size is measured as the slope at a given point of a curve in Figure 5. (Our view is that using asymptotic bounds [LZGS84] for knee points is pessimistic for our graph. The 2% peak is based on the curve of scale 100 which levels off sharply after this slope.) Table 2 estimates the buffer size at which the knee points occur for scales of 100 to 250, their proportion compared to the database size and the data hit rates at the points. The buffer sizes for these knee points constitute a relatively small percentage of database size, and seem to indicate a reducing percentage for larger databases.

Database Scale	Buffer Size (GB)	%Database Size	Data Hit Rate
100	1.24	14.2%	95.7
150	1.59	12.2%	96.7
200	1.97	11.3%	96.3
250	2.19	10.1%	96.3
375 (not knee)	3.10	9.5%	95.3

Table 2. Buffer Size and Data Hit Rate at Knee Points

Note that the knee point has not been reached for scale 375, and the buffer size listed in Table 2 is the maximum buffer size used. We include 375 in the table for comparison of the amount of memory needed for these scales to obtain similar throughput.



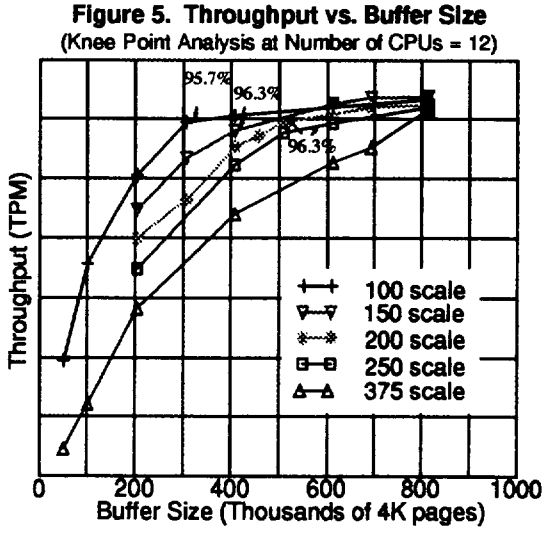


Fig. 6. Data Hit Rate vs. ln(Buffer Size/Database Scale)

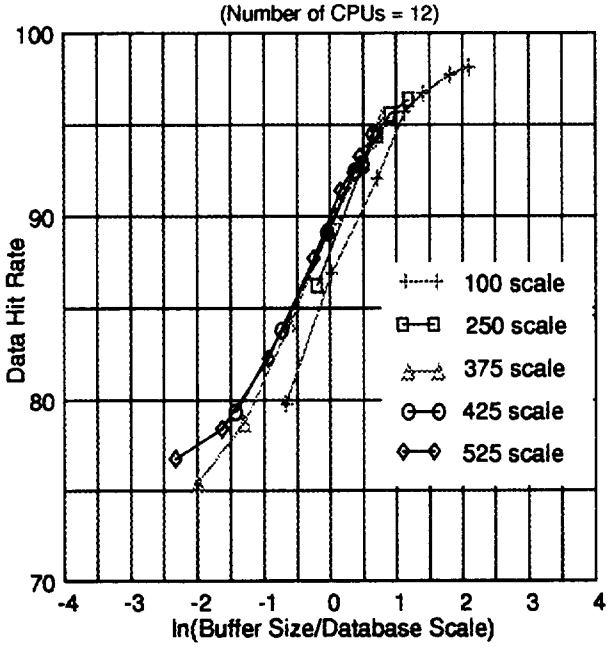
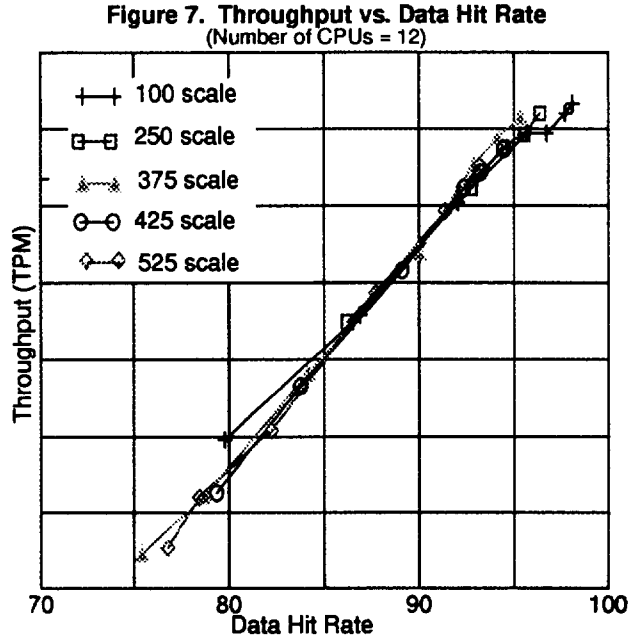


Figure 7. Throughput vs. Data Hit Rate



4.1.4. Linear relationship between Data Hit Rate and ln(Buffer Size/Database Scale).

We have observed in section 4.1.1 that the degree of benefit from a given memory buffer size is dependent on the ratio of the buffer size to the database size. Figure 6 plots Data Hit Rate vs. *ln(Buffer Size/ Database Scale)*. The graph shows that except for the very low-end at small buffer sizes and at the high-end where plateaus occurred, the rest of the curves are linear. The linear regions of the curves represent the greatest benefit obtainable from adding more memory buffer, and hence are the most important regions of the curves.

We fit the linear relationship between Data Hit Rate and *(Buffer Size/ DatabaseScale)* for 12 CPU to the regression model below:

$$DataHitRate = 87.65 + 0.0042 \times DatabaseScale + 7.91 \times \ln\left(\frac{BufferSize}{DatabaseScale}\right) \quad [1]$$

where *BufferSize* is in thousands of 4KB pages. This model fits well with R^2 equal to 99.9%. R^2 is the proportion of total variation about the mean of measured TPM explained by the deviations of the predicted values from the mean [DrSm81, DataD]. The F-ratio of the model, for test of significance of the regression, equal to 4701, well above the value at 1% point of F-distribution [DrSm81, DataD]. The ratios of the coefficients to its standard errors are, from left to right, 375, 7.63 and 87.1. The differences between the regression predictions to the measured values fall randomly in the range of *-1.92%* to *1.75%* .

4.1.5. As the buffer size increases, much more memory is required to obtain the same marginal benefit.

An important observation from Equation [1] is that as the buffer size increases for a given scale, much more memory is required to obtain a same amount of gain than at a lower buffer size. The gain is a function of the ratio of the new and old buffer sizes. The scale 525 curve in Figure 3, for example, shows that the data hit rate gained by increasing the buffer size from 203750 to 407500 buffer pages is about the same as that gained by increasing the buffer size from 407500 to 815000, but the latter uses twice the additional memory. Figure 2 also shows similar behavior for the throughput.

4.1.6. Linear relationship between Throughput (TPM) and Data Hit Rate.

Figure 7 plots Throughput vs. Data Hit Rate. As can be seen, there is a linear relationship for a major part of the curves. The slopes of these linear portions, although very similar, show a small dependency on the database scale. A larger database has a slightly steeper slope. This means that an improvement in data hit rate has a greater impact on TPM for larger databases. Remember, however, that it took much more memory to give the same improvement in data hit rate at a larger scale (Figure 3).

The linear portion of the curves in Figure 7 can be expressed in Equation [2] shown below for the throughput (TPM). Since most of the data points for scale 100 are around the plateau of the curve, we do not include scale 100 in the regression.

$$TPM = c + a \times DatabaseScale + b \times datahit \quad [2]$$

The model is a first order equation of two predictors: the DatabaseScale and Datahit rate, where c is the constant term, and a and b are the coefficients of the two predictors. This model has a R^2 equal to 99.4%. The F-ratio equal to 1133. The ratios of the

1. Given current database scale and Equations [1] and [2] for current number of CPUs.
2. For current scale, use equation[1] to estimate new data hit rate for target buffer size,

$$NewDataHitRate@TargetBufferSize = f(\ln(TargetBufferSize/CurrentScale))$$

and use equation [2] to estimate the new TPM,

$$NewTPM@TargetBufferSize = f(NewDataHitRate)$$

3. Check per warehouse ratio (NewTPM/CurrentScale), if within expectation and compliant to TPC-C specification, stop; otherwise, estimate new scale,

$$NewScale = NewTPM/perWarehouseRatio$$

4. Repeat Step 2 and 3, let *CurrentScale=NewScale*, until target TPM is found.

New Scale, Target Buffer Size → Target Hit Rate → Target TPM

Algorithm I: TPM Prediction for New Buffer size, Number of CPUs Unchanged

coefficient to its standard error for *a*, *b* and *c* are *-24.1* , *43.3* and *1.32* respectively. We compare the predicted TPM with the measured TPM. The differences fall randomly in the range of -2.46% to 1.44%. We further apply this model to predict the throughputs for scale 100. The errors are within 2% for those points in the linear region.

Since TPM is linearly dependent on the data hit rate, from Equation [1], TPM can directly be modeled as a function of *ln(Buffer Size/Database Scale)*. However, as we know that TPM is a function of number of CPUs while data hit rate may be a more generic behavior across number of CPUs, we prefer to evaluate the two relationships separately.

We will discuss the use of this model in the section 4.2 on prediction and examples. We will also revisit the model in Section 5.2 as we discuss how the characteristics change with the number of CPUs.

4.2 Predictions and Examples

It is especially useful to predict the throughput and the optimal buffer size. Prediction of the throughput is often required when:

1) only the memory buffer size changes,
2) the memory buffer size and number of CPUs change, and
3) some system architectural parameters change.

The goal here is to use the linear models to predict a TPM that is compliant with the TPC-C specification. In section 4.2.1, we discuss an iterative approach to prediction with different memory configurations. Section 4.2.2 presents model validation and an example application of the algorithm. In section 4.2.3, we show that if the database scale remains unchanged, a ball-park estimate for TPM at a different buffer size can be obtained. Prediction for the second case will be discussed in section 5.4 after we present the results of varying the number of CPUs. The third case is more complex and requires measurements not discussed in this paper; consequently we will not discuss prediction for the third case.

The other useful prediction is to estimate the optimal memory buffer size for a given configuration, i.e., the knee point in the Throughput vs. Buffer Size graph (Figures 2 and 5) beyond which further increases in the buffer size no longer produce significant performance gains. This is particularly important since the amount of memory needed to produce a performance gain increases greatly with buffer size (section 4.1.5). Not knowing the optimal amount of memory may result in adding memory with little or no performance improvement. We will discuss the knee points in section 4.2.4. Limited by our test configuration, we were only able to measure knee points for relatively small database scales, and the four data points obtained were not conclusive for developing a model.

4.2.1. Predicting TPM as buffer size changes while the number of CPUs remains unchanged.

We have already shown the linear relationship between Data Hit Rate and $\ln(BufferSize/DatabaseScale)$ (Equation [1]), and between TPM and Data Hit Rate (Equation [2]). In this section, we present an algorithm for predicting TPM when the number of CPUs is unchanged.

While it is possible to carry out regression directly with database scale and buffer size as parameters, more insight is gained by using a 2-step method. We know that TPM is a function of the number of CPUs. Data hit rate, however, as we shall show later in section 5.1, is fairly constant across number of CPUs for a fixed buffer size and database scale. By analyzing them separately, we have more flexibility in using each piece of information.

The algorithm is based on the two linear models shown in Equations [1] and [2]. The experiments are designed to be TPC-C compliant except for the TPM per warehouse (database scale) ratio. The new TPM obtained with equations [1] and [2] for a new buffer size may exceed the TPM per warehouse range permitted by the TPC-C specification. The database scale has to be increased/decreased to meet the requirement, and the TPM achievable from the new scale has to be re-estimated. The final TPM and the appropriate database scale to be used at the new buffer size can be found using the iterative algorithm I shown above.

Given the database scale and the TPM and Data Hit Rate

equations for current configuration, the algorithm first estimates the new data hit rate using equation [1] at the target buffer size and current database scale. The TPM is then derived from the new hit rate using equation [2] and the new TPM is checked against the TPC-C per warehouse rule. If it does not meet the specification, a new scale is estimated using a pre-defined TPM per warehouse ratio. This ratio can be derived from the current configuration, and often remains reasonably constant for the same platform. For the new scale and the target buffer size, repeat step 2 to 3 until a TPC-C compliant TPM is obtained.

4.2.2. Model Validation and Application Examples

For validation, we apply algorithm I to estimate a TPC-C compliant database scale and TPM at 12 CPUs for 3.1GB of memory buffer (815 thousands of 4KB pages). Starting with scale 375, and using Equations [1] and [2] and a per warehouse ratio based on past experience, a TPM and database scale conforming to the TPC-C per warehouse specification was obtained after four iterations. The iterations alternately over-estimate or under-estimate the per warehouse ratio until they eventually converge. We compared the estimated scale and TPM with the best TPC-C compliant measurements at 12 CPUs. The difference between the predicted database scale and the actual scale used is less than 1%, and that between the predicted and measured TPM is less than 5%.

While the algorithm is intended to estimate TPC-C performance, we may remove/modify step-3 to apply the model to real-world problems. One possible application is when the database of a company has grown beyond its initial system configuration and performance has begun to degrade. In such cases, the memory buffer could first be checked to see if it has already reached its plateau by using the percent database size in Table 2 as a guideline. If adding memory seems beneficial, then a few data points could be measured for the existing database size. Models equivalent to Equation [1] and [2], but simpler because database size is not a parameter in this example, may be developed from this data. Alternatively the quick estimation method discussed below can be used to estimate the new buffer size.

4.2.3. A quick estimation

It is possible to obtain a ball-park estimate on the TPM for the target buffer size at the same database scale. From equation [1], if the database scale remains unchanged, then for a change in buffer size, the change in data hit rate is

$$\Delta DataHitRate =$$

$$c \times \left[\ln\left(\frac{BufferSize2}{DatabaseScale}\right) - \ln\left(\frac{BufferSize1}{DatabaseScale}\right)\right]$$

$$= c \times \ln\left(\frac{BufferSize2}{BufferSize1}\right)$$

From Equation [2] and given that the database scale is a constant, the *change in TPM* is the *same* as long as the *ratio of BufferSize2/BufferSize1* is the *same*. To do a quick estimate without fitting a regression model, we only need to measure the TPM at two different buffer size configurations for the desired scales, and apply the above rule-of-thumb on these measurements.

The measurements we obtained agreed with this derivation. Shown below in Table 3 are the incremental percentage

Doubling Buffer Size	525 Scale		375 Scale	
	12-CPUs	8-CPUs	12-CPUs	8-CPUs
203.75 -> 407.5	22%	20%	18%	22%
407.5 -> 815	21%	20%	20%	22%

Table 3. Incremental Increase in TPM when Buffer Size Doubles.

increases in TPM (using TPM at 203.75 buffer size as base) as the memory buffer size doubles. The results show a high degree of consistency.

This relation and Equation [2] for TPM were applied to a different database platform with similar architecture to DB2 to estimate the buffer sizes for some target throughputs. The results were found to be accurate.

4.2.4. Rule-of-Thumb for Optimal Buffer Size

As we have discussed, more data points, particularly at larger scales, would be needed to develop a complete model. However, as we have shown in Table 2, all knee points occur at buffer sizes equal to about 10-15% of the database sizes, and this fraction seems to decrease with larger databases. If we assume that the knee point for a 375 scale database occurs at a data hit rate of 96.3% as for other scales, then from Equation [1] the buffer size will be 3.21GB corresponding to 9.82% of the database size. From this observation, we can establish a rule of thumb that in general a memory buffer size of about 10-15% of the database size is sufficient for good performance.

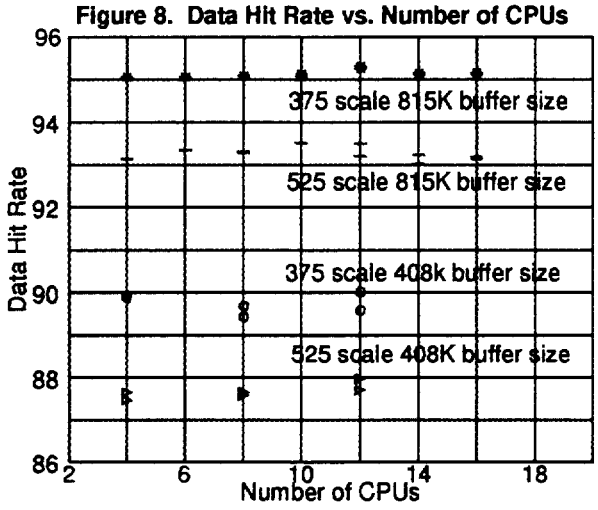
5. Study II: Varying Buffer Size, Database Size and Number of CPUs

In this section, we discuss how the characteristics presented in section 4 vary with the number of CPUs. In study I, we found that TPM has a linear relationship with data hit rate. In this study, we examine the data hit rate across number of CPUs in section 5.1, and evaluate this linear relationship as the number of CPUs changes in section 5.2. We know from past experience that TPM increases with the number of CPUs and have been using a *scalability formula* to model TPM as a function of number of processors. We discuss this scalability formula in section 5.3. Section 5.4 extends algorithm I to include this formula for predicting TPM as buffer size and number of CPUs are changed. Section 5.5 presents an example application.

5.1. Data Hit Rate across Number of CPUs

Figure 8 plots the data hit rate for scales 525 and 375 at 3.1GB (815 thousands of 4KB pages) of buffer for number of CPUs from 4 to 16. For each configuration of buffer size and number of CPU, we show a pair of data hit rates for TPM with less then 2% difference for that configuration. Also shown in the graph are the data hit rates for scales 525 and 375 at 1.55GB (407.5 thousands of 4KB pages).

The data hit rate remains fairly constant across the number of CPUs for a given database scale and buffer size, with an average



variation of about 0.25%. This means we can measure hit rate with a small CPU configuration for all database scales, and use them across a variable number of CPUs.

While TPM has increased with the number of CPUs, the buffer access pattern is about the same because of the pre-defined transaction mix and the randomness in TPC-C transactions. The growth in some database tables from executions of the transactions have little effect on the hit rates.

5.2. TPM vs. Data Hit Rate across Number of CPUs

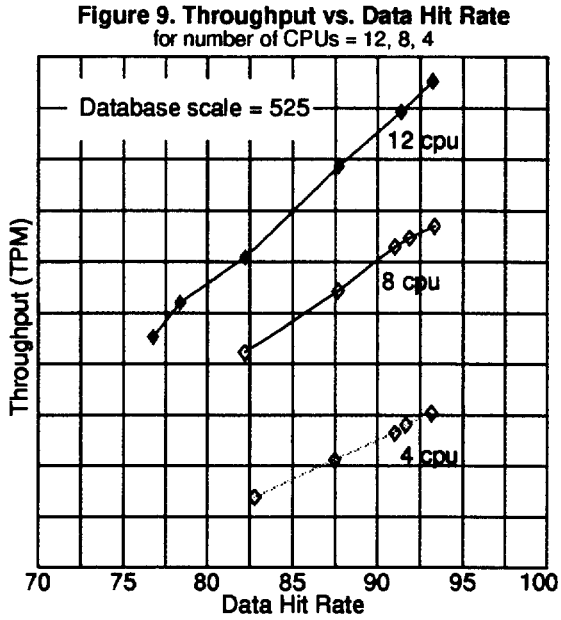
Figure 9 shows the graphs of TPM vs. Data Hit Rate for 4, 8 and 12 CPUs. For simplicity, we only plot for scale 525. Three observations can be drawn from the graph. First, the linear relationship between TPM and data hit rate (section 4.1.6) still holds. Second, the slope of the lines gets steeper with increasing number of CPUs. This means that the coefficient b in Equation [2] increases with number of CPU for a given database scale. The third observation is that the gap between 8 to 12 cpus is smaller than the gap from 4 to 8 cpus. We will discuss these gaps in terms of scalability factor in the section below.

5.3. Scalability Formula

The shape of the curve of TPM vs. number of CPUs is similar to the TPM vs. Buffer Size in Figure 2. The benefit of adding CPUs diminishes as the number of CPUs increases. In fact, it is common to calculate TPM per CPU to evaluate how TPM scales with additional CPUs. The ideal case will be a constant value per CPU. In practice, due to resource contention, each additional CPU contributes less to the throughput. The geometric scalability formula, shown in Equation [3] below, has been commonly used because of its simplicity [Arti91, McGa95].

$$\Lambda_{NCPU} = \Lambda_{1CPU} \times (1 + s + s^2 + s^3 + ... + s^{N-1})$$

$$= \Lambda_{1CPU} \times \left(\frac{1 - s^N}{1 - s}\right) \qquad [3]$$



where $0 < s < 1$. The formula starts with the throughput at 1 CPU as a base value. Subsequent CPUs each add a fraction of this base value to the overall throughput. The fraction is determined by a scalability factor, s, and it decreases exponentially with increasing number of CPUs. We have found this formula gives reasonable estimates of throughput over a wide range of CPUs.

We compare the gaps between the CPUs at different buffer sizes. Table 4 shows the ratio of TPM at 12-to-8 and 8-to-4 CPUs at buffer sizes of 815, 611.25, 407.5 and 203.75 thousand pages, for scales 525 and 375. The ratio of 12 to 8 CPUs are around 1.31 for both scales and all buffer sizes. The ratio of 8-to-4 CPUs are around 1.79 except for buffer sizes 815 and 204. The 4 CPU data points at these buffer size were not optimally tuned due to system availability.

#CPU compared	Buffer Size				Buffer Size			
	815	611	408	204	815	611	408	204
	TPM Ratio @525 scale				TPM Ratio @ 375 scale			
12/8	1.32	1.28	1.33	1.30	1.32	1.31	1.33	1.37
8/4	1.73	1.78	1.81	1.84	1.74	1.79	1.79	1.81

Table 4. Ratios of TPM at 12, 8 and 4 CPUs.

We compared these throughput ratios with those derived from past experience. The estimates were within 4% of the measurements.

From the scalability formula, we know that the gap between the lines of TPM vs. Data Hit Rate reduces with increasing number of CPUs since each additional CPU contributes less to the overall throughput. The measured TPM ratio for 12-to-8 CPU has dropped 27% from that at 8-to-4 CPU.

5.4. Prediction

Based on the above analysis, we can easily extend Algorithm I for a fixed number of CPU to include a change in number of CPUs. The modified algorithm, Algorithm II, is shown in Figure 10.

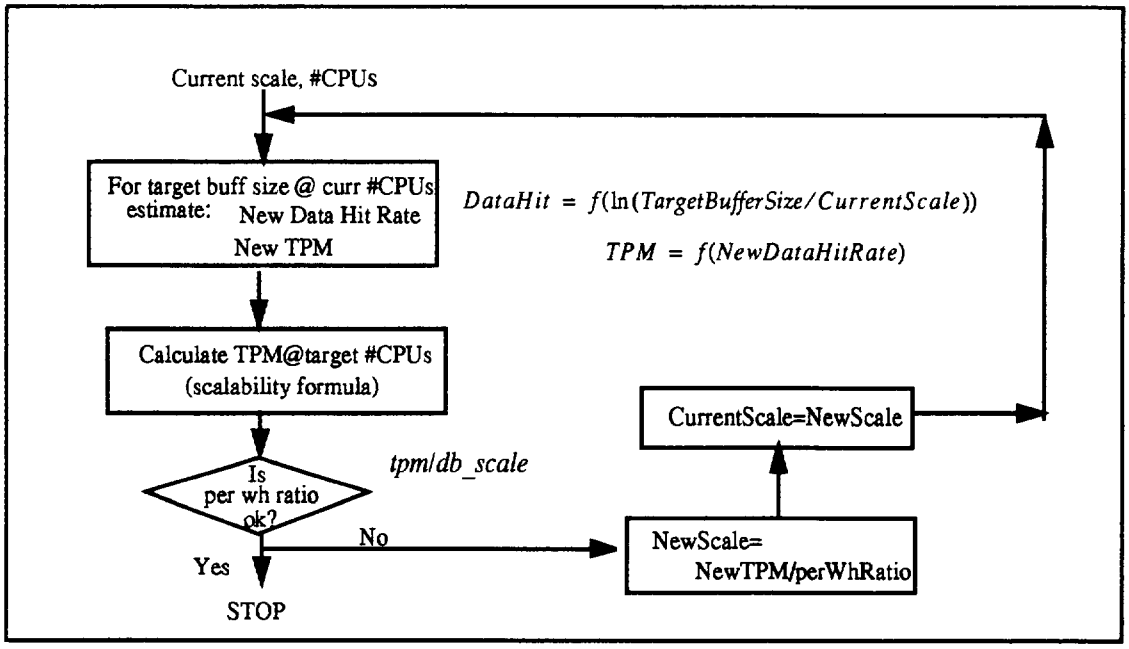


Figure 10. Algorithm II: TPM Prediction for Change in Buffer Size and Number of CPUs

As in algorithm I, use Equation [1] to estimate the new hit rate for the new buffer size, then use Equation [2] to predict the new TPM at current database scale and number of CPUs. Project this TPM, using the ratio estimated from the scalability formula, for the target number of CPUs. If the ratio of estimated TPM to the database scale is TPC-C compliant, stop, otherwise re-iterate the estimations of data hit rate and TPM for the new scale.

We applied this algorithm to predict the TPM for a similar hardware platform at several buffer sizes, and had obtained results within 5% error.

5.5. Applying to Memory and CPU Price/Performance Analysis

We have presented Algorithm I and II for estimating new throughput when considering a new system configuration. In addition, our analysis in the second study is very useful for considering the trade-offs between adding memory and CPUs.

By way of illustration, we plot TPM vs. Data Hit Rate in Figure 11 for 4 to 20 CPUs for scale 525 using scalability formula and measurements at 12-CPU as reference values. As we noted earlier, the lines converge as the number of CPUs increases. Let's say we currently have a data hit rate of 85%. At this data hit rate, the buffer size is about 1.28GB, far from the knee point size (section 4.1.3), and is on the linear region of the TPM and data hit rate graphs (Figures 6 and 7). Assume that the existing configuration has only 4 CPUs. To obtain a throughput marked at X, from Figure 3 it is estimated that 2GB of memory is required for the data hit rate at X (recall that for a given scale, data hit rate is fairly constant across number of CPUs). If the cost of 2GB of memory is comparable to adding four CPUs, it is obvious that adding four new CPUs is more appropriate. By adding four extra CPUs with the same memory configuration, the new throughput will increase to a throughput marked as X' (85% hit rate for 8 CPUs) that is much larger than X. In fact, for the database scale used in this example, the biggest benefit at small configurations will be adding the CPU first, then memory.

However, if the existing configuration has 12 CPUs and a data hit rate of 85%, to get to a throughput marked as Y, it is cheaper to add the memory. It requires about 1GB of memory as compared to four more CPUs. The trade-off between memory and CPUs is less obvious and our models are useful for analyzing the price/performance.

6. Conclusion

This paper addresses the potential benefits from the large memory configurations that are becoming available. We have investigated the impact of buffer size on performance for the TPC-C workload, a benchmark designed for complex OLTP systems. In particular, it is found that as the buffer size increases, much more memory is required to obtain the same improvement in throughput. We observed that the optimal memory buffer size is relatively small compared to the database size, and suggested a rule-of-thumb of 10-15% of the database size for a memory buffer. We showed that, for large databases, this optimal memory size can be greater than 4GB.

We have quantified the relationships: 1) between data hit rate, buffer size and database scale, 2) between TPM, data hit rate and database scale, and 3) between TPM and number of CPUs. Based on the observations and models developed, we presented algorithms for predicting throughput when buffer size and number of CPUs have changed.

Database vendors are developing buffer schemes that support separate index and data buffers, and separate buffers for the database tables. For database platforms supporting these features, a tentative approach is to first identify the buffers that have most impact on performance, and then evaluate the relationship of throughput with those buffer hit rates. While the TPC-C workload may not be representative of all real-world OLTP systems, this paper attempts to address practical performance issues which have received little attention in previously published research. The approach used in this paper can be used as a case study for similar workloads.

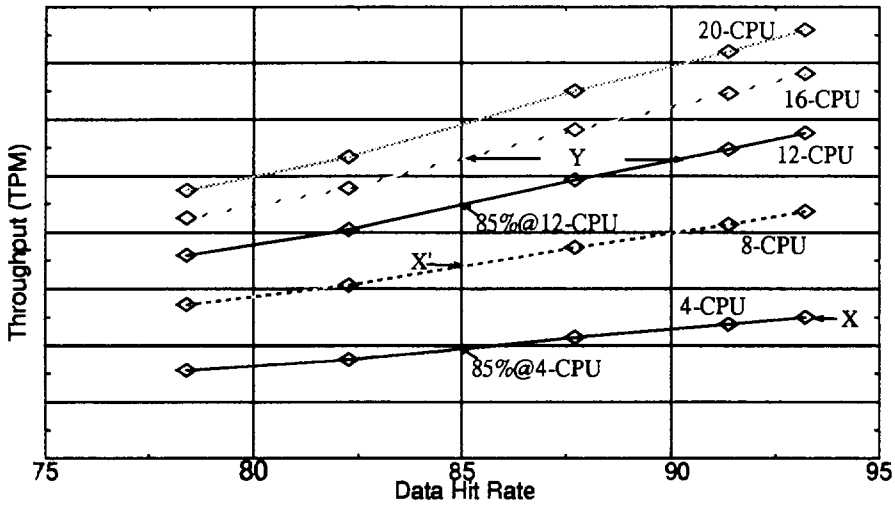


Figure 11. Estimated Throughput vs. Data Hit Rate
for scale=525 and number of CPUs = 20,16,12,8,4

References

[Amda67] Amdahl, G.,"Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities", AFIPS Conf. Proceedings, 1967

[Arti91] Artis, H.,"Quantifying Multiprocessor Overheads", Proceedings of CMG Conference, 1991.

[ChDe85] Chou, H. and DeWitt, D., "An Evaluation of Buffer Management Strategies for Relational Database Systems", Proceedings of VLDB, 1985.

[DB2Adm] "IBM DB2 Administration Guide for Common Servers", IBM Corp. 1995.

[DB2Info] "IBM DB2 Information and Concepts Guide for Common Servers", IBM Corp. 1995

[DaTo90] Dan, A. and Towsley, D., "An Approximate Analysis of the LRU and FIFO Buffer Replacement Schemes", Proceedings of Sigmetrics, 1990.

[DaYC93] Dan, A., Yu, P. and Chung, J., "Database Access Characterization for Buffer Hit Prediction", Database Engineering Conference, 1993.

[DataD] "DataDesk Statistic Guide", Data Description, Inc.

[DrSm81] Draper, N. and Smith, H., "Applied Regression Analysis", second edition, John Wiley & Sons, 1981.

[EfWH84] Effelsberg, Wolfgang and Haerder, "Principles of Database Buffer Management", ACM Transactions on Database Systems, Vol. 9, No. 4, December 1984.

[Gray93] Gray, J., Editor, "The Benchmark Handbook for Database and Transaction Processing Systems", Second Edition, Morgan Kaufmann, 1993.

[Gunt93] Gunther, N., "A Simple Capacity Model for Massively Parallel Transaction Systems", proceedings of CMG, 1993.

[Gunt96] Gunther, N., "Understanding The MP Effect: Multiprocessing in Pictures", to appear in Proceedings of CMG, 1996.

[LZGS84] Lazowska, E., Zahorjan, J., Graham, G., and Sevcik, K., "Quantitative System Performance", Prentice-Hall, 1984.

[LeDi93] Leutenegger, S. and Dias, D., "A Modeling Study of the TPC-C Benchmark", proceedings of SIGMOD, 1993.

[McGa95] McGalliard, J., "Case Study of Table-Top Sizing with Workload-Specific Estimates of the Multiprocessor Effect", proceedings of CMG Conference, 1995.

[MoHa92] Mohan, C., Haderle, D., Lindsay, B., and Schwarz, P., "Aries: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging", ACM Transaction on Database Systems, Vol. 17, No. 1, March 1992.

[Ston81] Stonebraker, M., "Operating System Support for Database Management", Communications of the ACM, Vol. 24, No. 7, July 1981.

[TPC95] TPC Benchmark C Standard Specification Revision 3.0", Transaction Processing Performance Council, February 15, 1995.