Reference Behaviour of the TPC-C Benchmark

Wenguang Wang Richard B. Bunt
Department of Computer Science
University of Saskatchewan
57 Campus Drive
Saskatoon, SK S7N 5A9
Canada
Email: {wang, bunt}@cs.usask.ca

May 15, 2000*

Abstract

This report analyzes the reference behaviour of IBM DB2 running an On-line Transaction Processing benchmark TPC-C. Reference behaviour for different tables and different object types (data or index) is studied. The effect of the number of users is also examined. It is found that the correlated references exist in most tables. The period of correlated references is affected by the type of references to tables, type of objects (data or index), and number of users. It is also found that different tables exhibit different reference behaviour, which depends on how it is accessed by the applications. The data pages and index pages also exhibit different reference behaviour. The index pages of all tables have similar locality. Based on the reference behaviour of different tables and the data/index pages, the buffer pool can be effectively partitioned to group pages with similar properties into one partition.

1 Introduction

In a database system, the database resides in disks and is managed by the DBMS (database management system). Before a database item can be accessed by the DBMS, the disk page on which the item resides must be present in the main memory. Therefore, a *buffer pool* which caches all disk blocks in memory is maintained by the DBMS. Disk pages can be read into the buffer pool and written back to the disks from the buffer pool. Because the access time difference between memory and disks becomes larger and larger, a proper design of buffer pool management algorithm is very important to the performance of DBMS.

Key to the design of buffer pool management strategies is an understanding of how database applications refer to pages from the databases. Empirical studies of database reference behaviour over the past mainly focused on the Hierarchical [5, 6, 9, 8] and the network model (i.e. CODASYL) database systems [3, 13]. Many conflicting results were reported by them.

In this report, the reference behaviour of IBM DB2 (a relational DBMS) running the TPC-C benchmark (an OLTP workload) is studied. The implications of the reference behaviour on the design of the buffer pool management strategies are also discussed. Section 2 presents the related work on the reference behaviour of database systems. Section 3 describes the environment under study in this work, including a brief introduction of the OLTP workload and the TPC-C benchmark. Section 4 discusses the trace collection procedure. Section 5 presents the trace characteristics and

*Revised on January 2003

their implications on the design and tuning of buffer pool management strategies. Section 6 is the conclusion.

2 Related Work

Understanding how database applications reference pages is the key to the design of successful buffer pool management strategies. Empirical studies of database reference behaviour over the years focused mainly on hierarchical DBMS systems [5, 6, 9, 8] and on network model DBMS systems [3, 13]. Many conflicting results have been reported:

1. **Randomness**. A study of the reference behaviour in a network model database system [3] showed that some databases showed no locality at all. No physical sequentiality was found either. The application under study was a single-user retrieval-only on-line application. The size of the databases used was small.
2. **Sequentiality**. [8] studied an on-line control system running on IMS, a hierarchical DBMS. The results showed the lack of locality within one transaction, but strong sequentiality was found. Based on this sequentiality, some prefetching techniques were proposed to improve the performance [9].
3. **Weak locality**. [3] found that database references showed less locality than programs in general. Contrary to [8], this locality came from inter-transaction references. [13] studied the reference behaviour of some batch programs running on a network model DBMS and found locality in the database references. Sequentiality was not found.
4. **Strong Locality**. The study on reference behaviour of IMS [5] found clear evidence of locality in hierarchical database systems. Furthermore, the locality of database references was observed to be more regular, more predictable, and hence, more exploitable than found in programs in general. The workload studied was generated by batch programs which dealt with typically expensive and time-consuming applications. A later study [6] confirmed the strong locality.
5. **Diversity**. The work in [6] also showed a diversity of reference behaviour in relational DBMS. The locality of references was different depending on query types. In [1], the reference behaviour of a relational DBMS was analyzed based on the operation set of relational databases. The sequential references, random references, and hierarchical references are used to model the reference behaviour of database operations.

To summarize, previous studies have shown that the database reference behaviour had diversity depending on different applications and DBMS systems. The overall characteristics are a mix of different reference patterns.

3 The Environment Under Study

3.1 The OLTP Workload

After the relational model is proposed in [2], relational DBMS becomes the dominate DBMS versus hierarchical and network model DBMSs, because the simplicity of relational model and the high level query language SQL.

Among many large database applications for relational DBMSs, OLTP is an important type of workload. OLTP is a class of database software that manages applications typically for data entry and retrieval transactions in many industries. OLTP is the cornerstone by which most businesses do business.

Most transactions in OLTP are simple. The execution time of each transaction is short (several seconds). There are upper bounds requirements for the response time of the transactions. An OLTP application has many terminals connected to one or more central database servers through a network as shown in Figure 1. The client/server model is typically used for OLTP applications. Different terminals initiate various transactions to the server independently. The database on the server is updated frequently by these transactions. Due to the I/O intensive nature of OLTP applications, large buffer pools and many disks are used at the database server in an OLTP application.

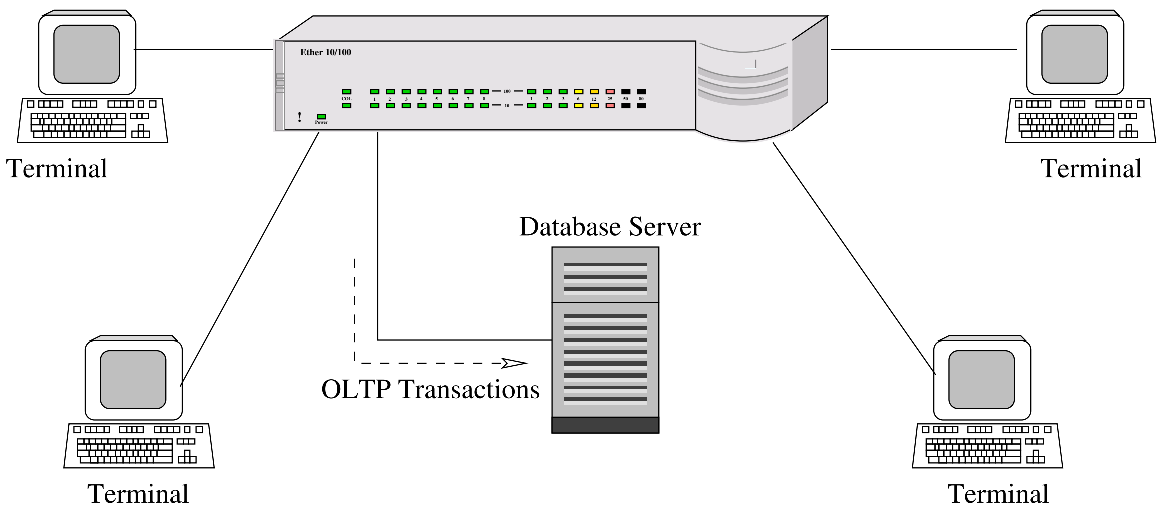

Figure 1: Architecture of an OLTP Application

3.2 The TPC-C Benchmark

The TPC benchmarks developed by the transaction processing performance council (TPC) [10] are widely accepted for testing the performance of database systems for transaction processing. TPC is a non-profit corporation founded to define transaction processing and database benchmarks. The TPC-C benchmark [11] represents an OLTP workload.

The TPC-C benchmark models the order processing operations of a wholesale supplier with some geographically distributed sales districts and associated warehouses. Each regional warehouse has 10 sales districts. Each district serves 3000 customers. The supplier has 100,000 items for sale. All warehouses maintain stocks for these items. Figure 2 shows the TPC-C business environment. Each warehouse is populated with about 100MB data before the benchmark is running. The scale of the TPC-C benchmark is expressed as the *number of warehouses*.

Five basic transactions that represent essential performance characteristics of the application are defined by the benchmark. These transactions are listed in Table 1. Performance of TPC-C is expressed in transactions-per-minute-C(tpmC), defined as the number of New Order transactions executed per minute.

Table 1: TPC-C Transactions

Name	Characteristic	Percentage
New-Order	read-write, mid-weight	45%
Payment	read-write, light-weight	43%
Order-Status	read-only, mid-weight	4%
Delivery	read-write	4%
Stock-Level	read-only	4%

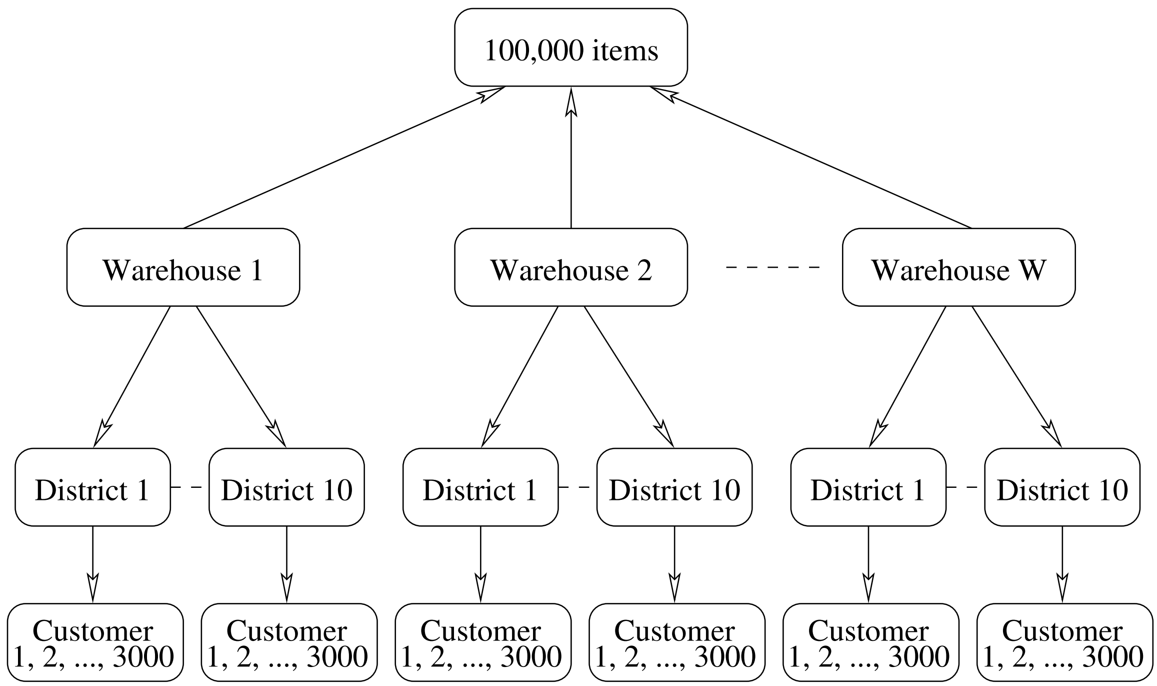

Figure 2: TPC-C Business Environment

The database of the TPC-C benchmark consists of nine tables. The relationships among these tables are defined in the entity-relationship diagram in Figure 3.

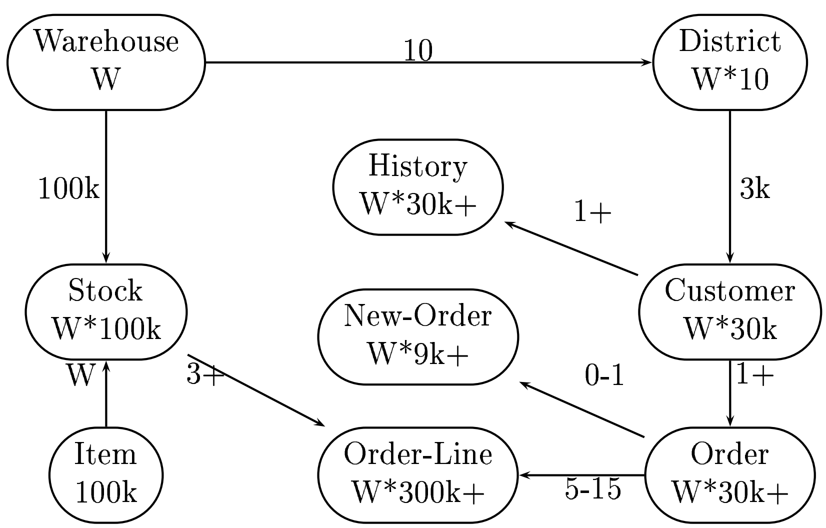

Figure 3: TPC-C Tables and their Relationships

The numbers in the entity blocks represent the number of rows of that table. These numbers are factored by W, the number of warehouses, to illustrate how the database scales. The numbers next to the relationship arrows represent the cardinality of the relationships (average number of children per parent). The plus (+) symbol indicates more rows will be added to the table as the benchmark runs.

In order to simulate the skewness of distribution of accesses in real applications, a non-uniform distributed random number generator is used to populate the database and generate transactions. This non-uniform random function $NURand_A(x, y)$ is defined as:

$$NURand_A(x, y) = ((random(0, A)|random(x, y)) + C)\%(y - x + 1) + x,$$

where:

1. $a|b$ stands for the bitwise logical OR operation between a and b.
2. $a\%b$ stands for a modulo b.
3. $random(a, b)$ stands for randomly selected uniformly distributed number within $[a, b]$.
4. C is a runtime random constant within $[0, A]$ that does not affect the distribution of the numbers but affects the "hot" values of the generated numbers. TPC-C benchmark has special rules for the way of selecting C so that it does not alter performance.
5. A is a constant that can affect the skewness of the distribution of the random numbers generated. $A = 2^n - 1$, where n is an integer, so that the lower n bits of the value returned by $random(x, y)$ can be affected. For these affected bits, the probability of value "1" is 75%, and the probability of value "0" is 25%, because this value is the logical OR operation from two uniformly distributed bits. The larger the value of A, the more skewed distribution the $NURand$ function generates.

When each district is populated with 3,000 customers, 1000 unique random last names are used for the first 1000 customers and the $NURand_{255}(0, 999)$ function is used to select a name from these 1000 names for the remaining 2000 customers. Figure 4 shows the cumulative density function of the popularity of the last names after the initial database population. The most popular last name is used by 60 customers. The cross in the figure shows that 21% last names are used by 60% customers.

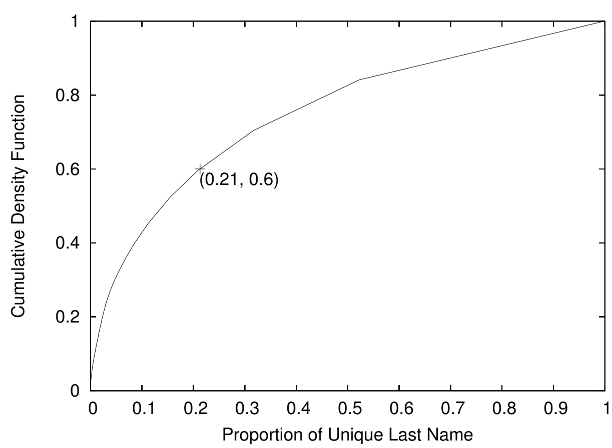

Figure 4: Distribution of Popularity of Last Name

During the execution of the benchmark, when the Item table is accessed by the New Order, Payment, and Order Status transactions, the item ID is selected using the $NURand_{8191}(1, 100000)$ function. When a customer is selected from the Customer table by the New Order transaction, the $NURand_{1023}(1, 3000)$ function is used to select a random customer ID. When a customer is selected by the Payment or Order Status transaction, it is selected by last name generated by the $NURand_{255}(0, 999)$ function 60% of the time, and selected by customer id generated by the $NURand_{1023}(1, 3000)$ function 40% of the time. The cumulative density functions of the popularity of customers and items referenced by the TPC-C transactions are shown in Figure 5. The item ID distribution is more skewed than the customer ID distribution. As shown in the figure, 16% item IDs attribute to 80% references, and 33% customer IDs attribute to 80% references.

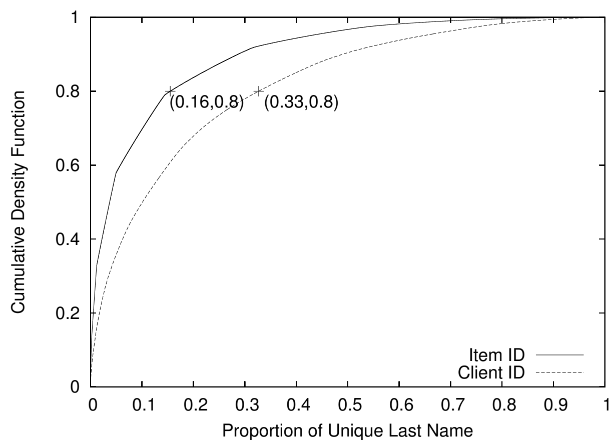

Figure 5: Distribution of Popularity of Item ID and Client ID

3.3 System Configuration and Experimental Setup

The DBMS studied in this research is IBM DB2 for Windows, version 7.1.0. The benchmark runs on an IBM PC Server 704 in the DISCUS lab. This machine is configured with 4 PentiumPro 200MHz processors, 512 MB RAM, 12 4.3GB hard disks attached to two PCI Wide Ultra SCSI-2 buses, running Windows NT Server 4.0. The data transfer rate of each SCSI bus is 40MB per second. 10 disks are IBM ST34571WC, and 2 disks are IBM DCHS04Y. All these disks are 7200RPM with an average seek time of about 9ms.

Since the focus of this study is on the server activities, remote terminal emulators required by TPC-C are not used to generate the TPC-C transactions. Instead, all transactions are generated on the DBMS server by a TPC-C driver program with no think time between transactions. The think time between transactions is removed to test the maximum throughput that the DBMS can achieve. The TPC-C driver program that implements the TPC-C benchmark was developed by IBM's Toronto Lab. When the TPC-C benchmark is running, 60 users send OLTP transactions to the DBMS.

A TPC-C database with 50 warehouses was created on the test machine. One disk of the machine is used for the log file of DB2. The buffer pool of DB2 can be configured up to 440MB, which is large compared to the size of the database used. The database can be created across from 3 to 11 physical disks, since at least three disks (4.3GB each) are needed to hold the database (10GB). To get the best throughput, software RAID-0 instead of RAID-5 is used to organize multiple disks.

4 The Traces

In order to trace the buffer pool activities to provide input for the simulator, the TPC-C benchmark is executed. During the execution of the benchmark, disk pages may need to be fixed/unfixed through the buffer pool. A trace point is placed between the upper layer of DBMS and the buffer pool as shown in Figure 6 to catch all fix and unfix requests. This is done by changing the source code of DB2 in the IBM Toronto Software Lab and utilize the db2trc tool coming with DB2.

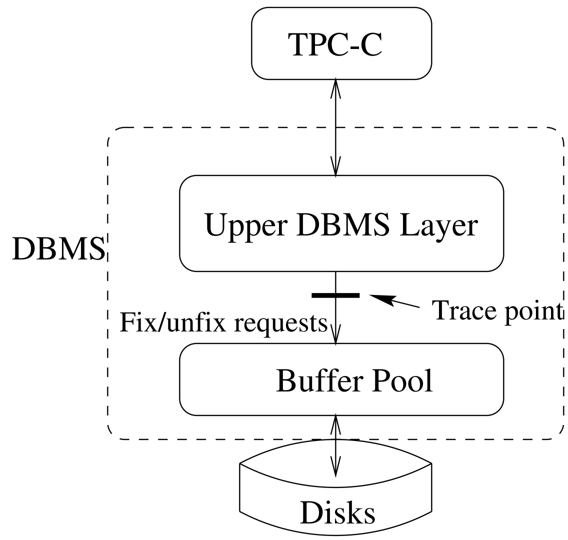

Figure 6: Trace Collection Point

Figure 7 shows how traces are collected. Once trace is turned on by db2trc, a shared in-memory trace buffer is created. Any requests passing the trace point will be recorded into the trace buffer. Because the trace buffer cannot hold all trace information needed by the simulator at one time, whenever the trace buffer is full, the TPC-C driver program suspends itself, notify db2trc to dump the trace into a file, and then resumes. This modification to the TPC-C driver program makes it possible to collect traces of any length.

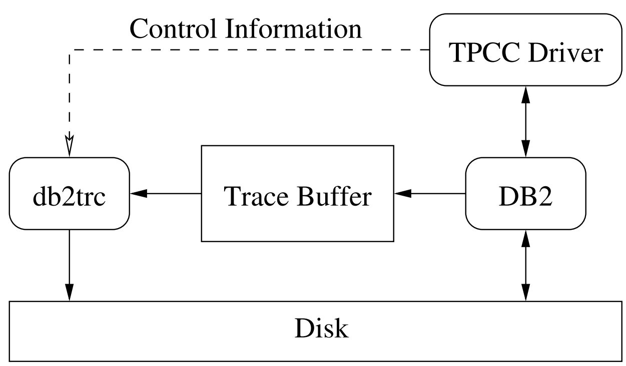

Figure 7: Trace Collection Structure

The trace records all necessary information related to fix and unfix requests. Table 2 presents the fields of a trace record. The fix mode defines two kinds of fix: *exclusive* and *shared*. If a page is fixed in the exclusive mode, it can be read/updated during the fix period. Any page can have at most one exclusive fix at any time. If a page is fixed in the shared mode, it can only be read (not updated) during the fix period. Multiple agents can fix the same page in shared mode at the same time.

Because of the overhead associated with the data collection process, the system runs much more slowly when recording a trace. For this reason, the orders of the aggregated fix/unfix requests from all agents to the buffer pool when the tracing tool is turned on may be different from that when the tracing tool is turned off. However, the orders of requests from each agent to the buffer pool

Table 2: Important Fields of the Trace

Field	Value
type	the type of the request, must be either *fix* or *unfix*
user id	the user who sends the request
object type	the type of the requested page, must be either *index* or *data*
table id	the table id where the requested page belongs to
page number	the logical page number of the requested page
fix mode (only for fix)	*exclusive* or *share*
modified (only for unfix)	whether or not this page has been modified

does not change. Each agent of DB2 has a unique agent ID, which is stored in every record of the trace. Therefore, the trace obtained can be separated into sub-traces by the agent IDs so that each sub-trace contains records only from one agent. These sub-traces can be used by the simulator. In order for the simulator to compute the throughput, that is, the number of transactions finished per minute, the begin and the end times of each transaction are recorded in the TPC-C program. This transaction information is then inserted into the trace based on the timestamp.

The trace used in the following analysis are collected from the PC Server 704 running TPC-C benchmark in the DISCUS lab. The number of warehouses in the TPC-C database is 50. The trace includes about 60 million buffer pool requests from 200 thousand transactions.

5 Trace Characterization

This section analyzes the reference behaviour of the traces collected. Since the presence of locality is the most common assumption when designing any replacement algorithm, it is the focus of this characterization. The locality of different tables and objects (data and indexes) is studied. Because there is no sequential access in the TPC-C benchmark (accesses are non-uniformly distributed random), the sequentiality of the traces is not discussed.

§ 5.1 explains the LRU stack depth that will be used to analyze the locality of the traces. § 5.2 analyzes the reference characteristics of a single user. § 5.3 analyze the reference characteristics of all users.

5.1 The LRU Stack Depth

The *LRU stack depth* is used to reflect the locality present in a particular trace. When measuring the LRU stack depth, all referenced pages are ordered by their recency of reference. For each reference, LRU stack is searched for the requested page. If it is found, the LRU stack depth of this reference is the number of pages that have smaller recency than this page. This page is then moved to the top of the stack. If the page is not found in the stack, the LRU stack depth of this reference is infinite.

If a page is referenced twice in a row, an *immediate re-reference* is noted. For any buffer pool replacement algorithm, immediate re-references are always buffer hit and do not change the behaviour of the buffer pool. The existence of immediate re-references may make the locality of the reference trace appear better than it actually is. Therefore, immediate re-references are removed in the following discussions.

The cumulative density function of the LRU stack-depth probability $F(x)$ is used to describe the locality of the reference, where x is the LRU stack depth. For a buffer pool of size x, $F(x)$ is also the buffer pool hit ratio if the buffer pool is managed by the LRU algorithm.

5.2 Reference Characteristics for a Single User

In the TPC-C benchmark, reference characteristics do not change over time, and all users have identical characteristics. There are about 1 million trace records for every user. Preliminary analyses of the traces showed that the first 250,000 records of a user had the same characteristics as the whole trace of this user. Thus only 250,000 records of user 0 are used for the analyses of this subsection, which is a long enough period to characterize the reference behaviour accurately.

5.2.1 Overall Characteristics

Figure 8 shows the LRU stack depth distribution of a single user. The buffer hit ratio is about 80% when the buffer pool size is only 1000. This implies good locality of the overall trace. The vertical line at the end of the curve means that 15% of the pages have infinite stack depth. These infinite stack depths are generated when every page is referenced the first time. Therefore, there are 15% distinct pages.

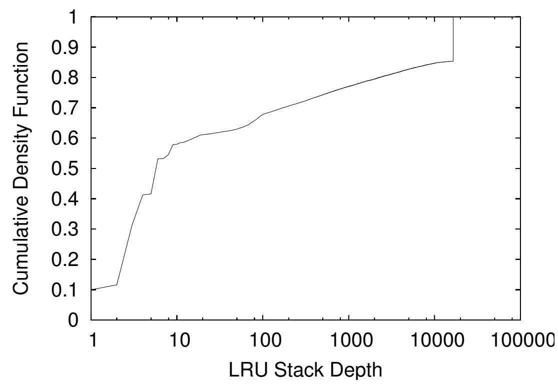

Figure 8: The LRU Stack Depth

5.2.2 Characteristics of Different Tables

There are nine tables in the TPC-C benchmark (see Figure 3). Experiments on the reference behaviour of every table show that they exhibit different locality. The nine tables are organized into three categories in Table 3 to focus the following discussions.

Table 3: Tables with Different Locality

Locality	Name of the Tables
Excellent	Warehouse, OrderLine
Good	District, Item, NewOrder, Order, Stock
Bad	History, Customer

Figure 9 shows the LRU stack depth of the tables Warehouse and OrderLine, which have excellent locality.

For Warehouse, no stack depth greater than 50 is found. Warehouse is the smallest table, with only 50 pages. Warehouse is accessed frequently in most transactions. Therefore, the good locality of this table is due in large part to the high popularity of pages of this table.

In OrderLine, more than 80% of the pages have stack depth less than 10. In one New-Order transaction, several new records are inserted into OrderLine. These records might be searched in several other transactions. As a result, good locality is observed in OrderLine.

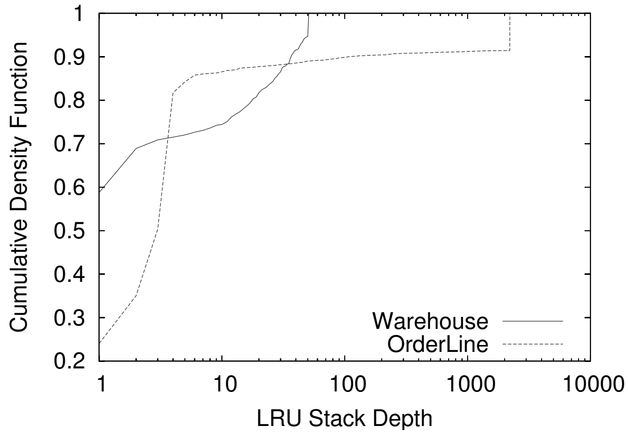

Figure 9: The LRU Stack Depth of the *Warehouse* and *OrderLine* Tables

Five tables (District, Item, NewOrder, Order, and Stock) have good locality. Their LRU stack depth plots have similar shape. For clarity, only the data for three tables, Item, Order and Stock, are plotted in Figure 10. The references to the Item table are non-uniformly distributed random access. Its good locality is due to the highly skewed distribution of the accesses as shown in Figure 5.

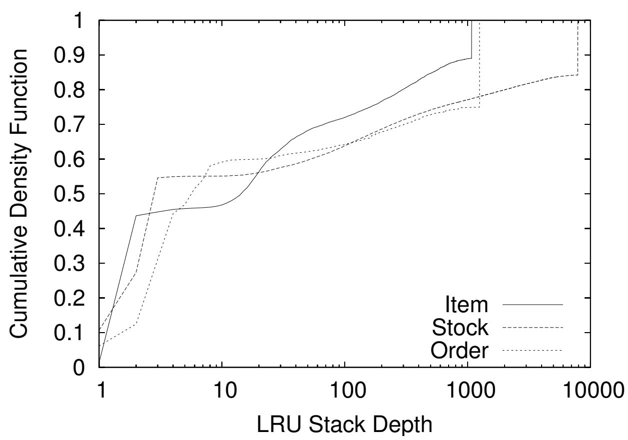

Figure 10: The LRU Stack Depth of the *Stock* and *Order* Tables

Figure 11 shows the locality of History and Customer, both of which have poor locality. This figure shows that History has almost no locality at all: about 85% of its pages have infinite stack depth. This is because data are appended at the end of the History Table and are not accessed again in the TPC-C benchmark. 50% of the pages in Customer have infinite stack depth. Customer is referenced by non-uniformly distributed references. It exhibits poor locality because the non-uniform random function used for customer table is less skewed (see Figure 5).

Figure 9, 10, and 11 show that there is a sharp increase in the buffer pool hit ratio for almost all tables when the buffer size is less than 10 pages, which is reflected as a "knee" in the figures. Since one page contains many table rows or index values, the upper layer of a DBMS typically accesses a database page for several times during a short interval. These references are called *correlated references*. The period that correlated references occur is called *period of correlated references*. The sharp increase of the buffer pool hit ratio is caused by correlated references. This sharp increase

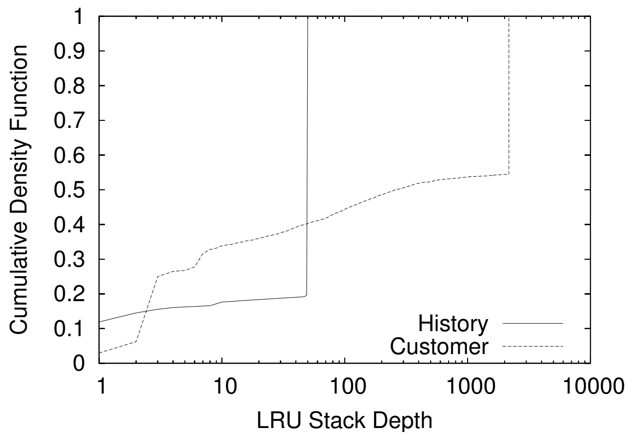

Figure 11: The LRU Stack Depth of the *History* and *Customer* Table

does not reflect the real locality of the page references in long term, therefore need to be considered when designing buffer pool replacement algorithms. In the LRU-K algorithm [7], a parameter is used to define the period of correlated references so that re-references occurring in this period are not counted as its k^{th} last references. In the 2Q algorithm [4], the length of the short term queue also reflects the estimation of the period of correlated references.

Most importantly, these figures show that different tables often have different reference characteristics. This is caused by the different ways the applications operate on the tables. If the data in a table are accessed in a highly skewed manner (as in Item), the high popularity of the hot data can cause high locality of the page references of this table. The reference behaviour of each table can guide the partitioning of the buffer pool. Tables with good locality can get high hit ratio with small buffer pool. The hit ratio will not increase much if more buffer pool space are given to these tables. On the other hand, the hit ratio of tables with poor locality keeps increasing even when the buffer pool is large. As a result, if the buffer pool need to be partitioned into several partitions, tables with good locality should be put into a small partition, and tables with poor locality should be put into a large partition.

5.2.3 Characteristics of Data And Indexes

The structures of data pages and index pages are different. Data pages have a linear structure, and are often stored sequentially on disks. Accesses to data pages can be random, sequential, or exhibiting locality. Index pages employ the B^+ tree structure to facilitate fast search by key values. Accesses to index pages always start from the root and go through all levels of the tree until either the search fails or the appropriate leaf is reached. Therefore, different reference behaviour is expected on data pages and index pages.

Figure 12 shows the stack depth distribution for data pages and index pages. 70% of data pages have infinite stack depth, while the locality of the index pages is much better.

The analysis of reference behaviour for data pages and index pages can guide the partitioning of the buffer pool. Because data and indexes have different reference behaviour, they could be put into different buffer pools and be managed by some dynamic partition algorithm.

Recall that Figure 11 shows that the Customer table has poor locality. The LRU stack depths of the data pages and the index pages of this table are shown in Figure 13. Nearly all the data pages have infinite stack depth while the index pages have locality similar to that of all index pages. In fact, all index pages have similar access patterns.

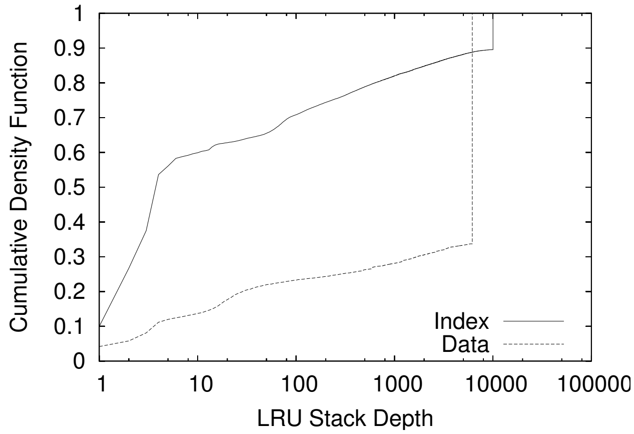

Figure 12: The LRU Stack Depth of Data and Indexes

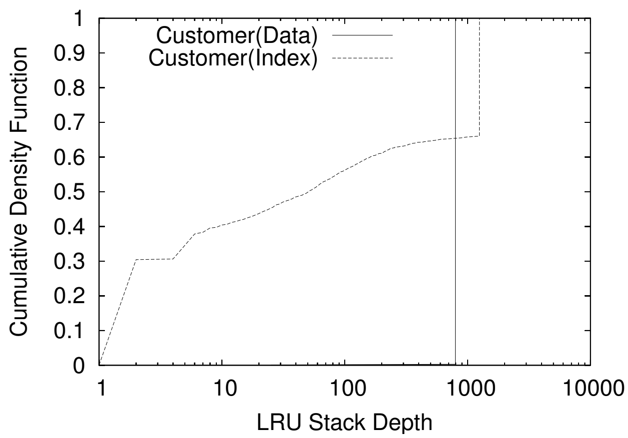

Figure 13: The LRU Stack Depth of the Table *Customer*

5.3 Characteristics of Multiple Users

All the above analyses are based on the requests generated by a single user. When many users use the DBMS at the same time, as in the case of the TPC-C benchmark, different users share pages of the same database. The aggregate reference behaviour is affected by this sharing. Since the workload does not change over time, the first 100,000 requests sent by each user are used in the following analyses.

Figure 14 shows the LRU stack depth of the overall reference trace for 1 user and 60 users. When the buffer pool is larger than 125 pages, the hit ratio of the trace with 60 users is similar to that of the trace with 1 user. This is because in the TPC-C benchmark, all users access the database in the same pattern as defined by the non-uniform random generator.

Similar to the single user case, the sharp increase in the buffer pool hit ratio at small buffer pool sizes (less than 125 pages) in Figure 14 indicates that there are correlated references. Since the correlated references from different users interleave with each other, the period of correlated references increases when the number of users increases. This can be seen in the figure that the knee occurs at a larger buffer pool size. If the buffer pool replacement algorithm needs to estimate the period of correlated references (e.g., LRU-K and 2Q), this parameter must be tuned according to the workload to get good performance.

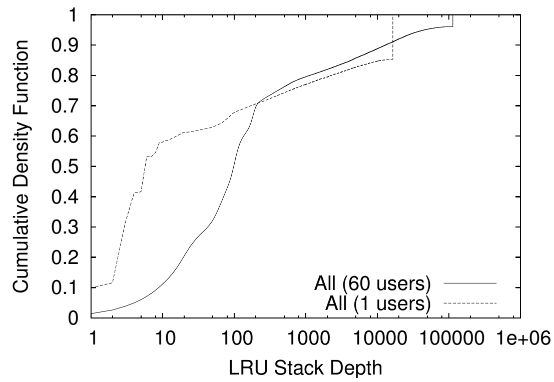

Figure 14: The LRU Stack Depth of 60 Users

6 Summary

This report analyzes the reference behaviour of the TPC-C benchmark. The correlated references exist in most tables. The period of correlated references is affected by the type of references to tables, type of objects (data or index), and number of users. If the period of correlated references is a parameter used in the buffer pool replacement algorithm as in LRU-K and 2Q, this parameter must be tuned according to these factors.

Different tables exhibit different reference behaviour, which depends on how it is accessed by the applications. The data pages and index pages also exhibit different reference behaviour. The index pages of all tables have similar locality. Based on the reference behaviour of different tables and the data/index pages, the buffer pool can be effectively partitioned, to group pages with similar properties into one partition.

References

[1] H.-T. Chou and D. J. DeWitt. An evaluation of buffer management strategies for relational database systems. In *Proceedings of the* 11^{th} *International Conference on Very Large Data Bases (VLDB'85)*, pages 174–188, Stockholm, Sweden, August 1985.

[2] E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.

[3] W. Effelsberg and M. E. S. Loomis. Logical, internal, and physical reference behavior in CODASYL database systems. *ACM Transactions on Database Systems*, 9(2):187–213, June 1984.

[4] T. Johnson and D. Shasha. 2Q: A low overhead high performance buffer management replacement algorithm. In *Proceedings of* 20^{th} *International Conference on Very Large Data Bases (VLDB'94)*, pages 439–450, Santiago de Chile, Chile, September 1994.

[5] J. P. Kearns and S. DeFazio. Locality of reference in hierarchical database systems. *IEEE Transactions on Software Engineering*, SE-9(2):128–134, March 1983.

[6] J. P. Kearns and S. Defazio. Diversity in database reference behavior. In *Proceedings of the 1989 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer*

Systems, volume 17(1) of *ACM SIGMETRICS Performance Evaluation Review*, pages 11–19, Berkeley, CA, May 1989.

[7] E. J. O'Neil, P. E. O'Neil, and G. Weikum. The lru-k page replacement algorithm for database disk buffering. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993*, pages 297–306, 1993.

[8] J. Rodriguez-Rosell. Empirical data reference behavior in data base systems. *Computer*, 9(11):9–13, November 1976.

[9] A. J. Smith. Sequentiality and prefetching in database systems. *ACM Transactions on Database Systems*, 3(3):223–247, September 1978.

[10] Transaction processing performance council. `http://www.tpc.org/`.

[11] TPC Benchmark^{TM} C. `http://www.tpc.org/tpcc/`.

[12] TPC Benchmark^{TM} W. `http://www.tpc.org/tpcw/`.

[13] A. I. Verkamo. Empirical results on locality in database referencing. In *Proceedings of the 1985 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, volume 13(2), pages 49–58, Austin, TX, August 1985.