Analysis of the Characteristics of Production Database Workloads and Comparison with the TPC Benchmarks

Windsor W. Hsu[†⋆]
Alan Jay Smith[⋆]
Honesty C. Young[†]

[†]*IBM Research Division*
IBM Almaden Research Center
San Jose, CA 95120
{*windsor,young*}@*almaden.ibm.com*

[⋆]*Computer Science Division*
University of California
Berkeley, CA 94720
{*windsorh,smith*}@*cs.berkeley.edu*

Report No. UCB/CSD-99-1070

November 1999

Computer Science Division (EECS)
University of California
Berkeley, California 94720

Analysis of the Characteristics of Production Database Workloads and Comparison with the TPC Benchmarks

Windsor W. Hsu[†⋆]
Alan Jay Smith[⋆]
Honesty C. Young[†]

[†]IBM Research Division
IBM Almaden Research Center
San Jose, CA 95120
{windsor,young}@almaden.ibm.com

[⋆]Computer Science Division
University of California
Berkeley, CA 94720
{windsorh,smith}@cs.berkeley.edu

Abstract

There has been very little empirical analysis of any real production database workloads. Although The Transaction Processing Performance Council benchmarks C (TPC-C) and D (TPC-D) have become the standard benchmarks for online transaction processing and decision support systems respectively, there has also not been any major effort to systematically analyze their workload characteristics, especially in relation to those of real production database workloads. In this paper, we examine the characteristics of the production database workloads of ten of the world's largest corporations and we also compare them to TPC-C and TPC-D. We find that the production workloads exhibit a wide range of behavior; in some cases, the TPC benchmarks fall reasonably within the range of real workload behavior, and in other cases, the TPC benchmarks are not representative of the real workloads. While the two TPC benchmarks generally complement one another in reflecting the characteristics of the production workloads but there are still some aspects of the real workloads that are not represented by either of the benchmarks. Specifically, our analysis suggests that the TPC benchmarks tend to exercise the following aspects of the system differently than the production workloads: concurrency control mechanism (TPC-C tends to have longer transactions and fewer read-only transactions than the production workloads while some of TPC-D's transactions are much longer but are read-only and are run serially), workload-adaptive techniques (the production workloads have I/O demands that are much more bursty), scheduling and resource allocation policies (unlike TPC-C whose transactions are very regular and TPC-D where the queries are run serially, the production workloads tend have many concurrent and diverse transactions), and I/O optimizations for temporary and index files (TPC-C has no I/O activity to temporary objects while most of TPC-D's references are directed at index objects). In this paper, we also reexamine Amdahl's rule of thumb for a typical data processing system (one bit of I/O for every instruction) and discover that both the TPC benchmarks and the production workloads generate on the order of 0.5 to 1.0 bit of logical I/O per instruction, surprisingly close to the much earlier figure.

This work has been supported by the State of California under the MICRO program, and by IBM, Cisco Corporation, Fujitsu Microelectronics, Intel Corporation, Microsoft Corporation, Quantum Corporation, Sun Microsystems and Toshiba Corporation.

This report is also available as Research Report RJ 10165, IBM Almaden Research Center, San Jose, CA.

1 Introduction

The Transaction Processing Performance Council (TPC) benchmarks C (TPC-C) [51] and D (TPC-D) [52] have emerged as the de facto standard benchmarks for on-line transaction processing (OLTP) systems and decision support systems (DSS) respectively. While such standard benchmarks are important for progress in the field in that they define the playing field by establishing objectives that are easily measurable and repeatable, the real utility of the benchmarks is whether they represent the workloads of interest. To effectively use a benchmark, therefore, we have to carefully evaluate its characteristics against those of the target workloads to understand how closely they correspond. Although the TPC-C and TPC-D benchmarks have become widely accepted and as a result are heavily used for both systems design and marketing, there has not been any major effort to empirically determine their workload characteristics, let alone to establish how representative their characteristics are of real workloads.

In fact, there has been very little empirical analysis of any real production database workloads. This reflects the fact that production systems are by definition critical to the proper functioning of an organization so that it is very difficult to get access to them for the purpose of conducting a scientific study, especially if the study requires any software changes or if data is to be collected and removed from the system. Therefore, although the hallmark of a good benchmark is that it should capture all the essential characteristics of the workload of interest without undue complexity, we often do not have a clear picture of the characteristics of the target workload. This is highly undesirable because a poorly designed benchmark may impede real progress in the field if it is not realistic and end up focusing energy and attention on issues that do not often arise in production environments.

In this research, we use trace-driven simulations [46, 54] to empirically examine the characteristics of the peak production database workloads of ten of the world's largest corporations as well as workloads similar to the TPC-C and TPC-D benchmarks[1]. Our main focus in this paper is on what we call descriptive system-level characteristics. These are the logical properties of a workload that a user or system administrator can readily understand and relate to without requiring detailed knowledge of the internals of the system. We compare and contrast such characteristics of the production workloads with those of the TPC benchmarks, paying special attention to any performance implications. In a companion paper [20], we examine in detail the I/O reference behavior of the workloads.

The traces used in this study were collected on systems running IBM's industrial-strength DB2 relational database management system (DBMS) and to the best of our knowledge, represents by far the most complete and diverse set of production workloads ever reported on in the literature. We cannot overemphasize the amount of time, effort and cost that these traces represent. This research would not have been possible without the support and help of many.

The rest of this paper is organized as follows. Section 2 contains a brief overview of previous work in the area of workload characterization and analysis. Section 3 discusses our methodology and describes the traces that we use. The characteristics of our workloads are presented in Section 4. Concluding remarks appear in Section 5 and acknowledgements in Section 6. Because of space constraints, we can only highlight some of our analysis results in this paper. More detailed graphs and data are available from our web site [22].

2 Related Work

There have been several published studies of the reference behavior of database workloads, especially that of hierarchical and network databases. See for instance [7, 12, 17, 26, 27, 41, 44, 55, 58]. For a more complete bibliography, the reader is referred to [20]. Unfortunately, most of these studies do not provide descriptive characteristics of the workloads being analyzed even though the reference behavior clearly depends on the workload imposed on the database. Without knowing the kinds of workload that are being analyzed, interpreting the results of the studies is very difficult. Consequently, there seems to be conflicting conclusions as to whether locality or sequentiality is present in the database reference stream. [58], which investigated design issues in disk caches using data from several commercial installations including both IMS and DB2 customer sites, is one of the notable exceptions that provides some characteristics of the workloads analyzed. In addition, a recent study of lock contention in database systems contains some transaction statistics based on traces taken from three commercial DB2 installations [43].

Though the TPC-C [51] and TPC-D [52] benchmarks have clearly been extensively studied and optimized by both database and system vendors, there has not been any systematic attempt to characterize these workloads empirically and to compare their characteristics with those of production database workloads. Based on static analysis of accesses to tables, [31] looked at the skewness in the data access of TPC-C. [53] contains an empirical study of how the database size, buffer size and the number of CPUs affect the throughput and buffer hit rate of TPC-C on symmetric multiprocessors (SMPs). Recently, [21] analyzed the query plans taken from a recently certified TPC-D setup and considered the potential

[1] Because our TPC benchmark setups have not been audited per the benchmark specifications, our workloads are technically not TPC benchmark workloads and should only be referred to as TPC-like. In the rest of this paper, when the terms TPC-C and TPC-D are used to refer to our benchmark workloads, they should be taken to mean TPC-C-like and TPC-D-like respectively.

benefit of offloading TPC-D operations to storage systems with embedded processors.

[35, 50, 57] studied file reference characteristics on time-shared VAX-11/780s in an academic environment. The measurements show that most of the accessed files are small, though large files account for a large fraction of the bytes moved. Similar file usage patterns are reported in a subsequent study conducted on a collection of about forty 10-MIPS workstations running the Sprite operating system in a comparable academic environment [5]. There have been several other studies that focused on the effectiveness of caching in the filesystem [14, 56]. An analysis of the file usage patterns in commercial computing environments is presented in [40]. Unlike most other studies that were based on data from academic or research environments, this study was based on traces collected at eight different and relatively large VAX/VMS customer sites. The workload at these sites included program development, scientific computing, office applications, transaction processing and batch environments. The analysis reveals that a relatively small percentage of the files are active and that a very small number of files account for most of the operations.

There is a large body of work on characterizing scientific workloads in parallel and supercomputing environments. See for instance [6, 9, 33, 34, 36, 37, 39]. In general, scientific vector applications tend to have large I/O request sizes and large files. Parallel scientific workloads tend to have smaller I/O request sizes.

3 Methodology

The methodology used in this paper is trace-driven simulation [46, 54]. In trace-driven simulation, relevant information about a system is collected while the system is handling the workload of interest. This is referred to as tracing the system and is usually achieved by using hardware probes or by instrumenting the software. In the second phase, the resulting trace of the system is played back to drive a model of the system under study. In other words, trace-driven simulation is a form of event-driven simulation where the events are taken from a real system operating under conditions similar to the ones being simulated. More comprehensive discussions of this technique and its strengths and weaknesses can be found in [46, 54].

The traces used in this study were collected by instrumenting commercial DBMSs. Instrumenting the DBMS allows the trace information to be collected at a logical level. This reduces dependencies on the system being traced and allows the trace to be used in a wider variety of studies, including those in which the models are somewhat different from the original system. In this study, we examined a total of 14 traces representing both industry standard benchmarks (TPC-C and TPC-D [51, 52]) and the production workloads of ten of the world's largest corporations. The benchmark traces were collected on a multiprocessor Personal Computer (PC) Server running DB2/Universal Database (DB2/UDB) V5 [25] on Windows NT 4.0. The production traces were collected on IBM mainframes running various versions of DB2/MVS, now known as DB2/390 [23].

In order to make our characterization more useful for subsequent mathematical analyses and modeling by others, we fitted our data to various functional forms through non-linear regression which we solved by using the Levenberg-Marquardt method [38]. When appropriate, we also fitted standard probability distributions to our data by using the method of maximum likelihood to obtain parameter estimates and then optimizing these estimates by the Levenberg-Marquardt algorithm [38].

3.1 Trace Collection

We instrumented DB2/UDB at the source level to collect relevant trace information for the TPC benchmarks. Because the act of tracing a system may affect its behavior, we paid special attention to minimizing any such disturbances. For instance, our tracing facility collects the trace records in shared memory before batch writing them asynchronously to disk. The shared memory buffer is double buffered so that trace collection is not blocked during write-backs. Each trace record is time-stamped with minimal overhead by directly accessing the processor cycle counter. At certain trace points, it is expensive to collate all the interesting information. In such cases, enough data is written to the trace so that an off-line post-processing step can be used to reconstruct the information. We collected trace records for both logical and physical reads and writes, prefetch requests initiated by DB2, references to the database log and transaction starts and ends. By comparing the TPC-C throughput results when trace collection is enabled and disabled, we estimate that this tracing mechanism imposes an overhead of less than 5%. This figure is dramatically lower than tracing overheads that have been previously observed; GTF tracing can require over 50% of the CPU time.

The production traces were collected using a custom DB2/390 tracing package developed at IBM's Almaden Research Center. This tracing package is designed to collect trace data with a minimum amount of overhead so that it can be run on customer production systems with little throughput impact. It is built upon the existing DB2 Instrumentation Facility and its performance trace [24]. The basic approach is to use a DB2 exit routine to collect the required data from a specially instrumented DB2 build. The collected data are assembled into trace records and stored in large memory buffers that are tracked by a task operating asynchronously in another address space. When a buffer becomes full, this other task batch writes the trace records to disk or more typically, to tape cartridges that are either stacked or housed in multiple tape units. This tracing package collects trace

Transaction	Min. %	Profile	Description
New Order	-	Mid-weight, read-write, online response time requirement	Initiates an order for an average of 10 items.
Payment	43	Light-weight, read-write, online response time requirement	Updates the customer's balance and reflects the payment on the district and warehouse sales statistics.
Order Status	4	Mid-weight, read-only, online response time requirement	Queries the status of a customer's last order.
Delivery	4	Mid-weight, read-write, relaxed response time requirement	Processes a batch of 10 new orders, one for each district for a given warehouse.
Stock Level	4	Heavy, read-only, relaxed response time requirement	Counts the number of items in the last 20 orders in a district that fall below the stock threshold.

Table 1: Summary of TPC-C's Transactions. The benchmark specifies the minimum percentage of transactions that are Payment, Order Status, Delivery and Stock Level transactions. New Order transactions, whose completion rate determines the TPC-C performance metric, make up the remainder.

records for buffer manager requests, transaction boundaries, and locking events. In tests conducted on an IBM 4381-T92 when handling a DB2 transaction oriented workload at 70% CPU utilization, the trace collection added only about 4% to the CPU utilization.

The buffer pool interface in both DB2/UDB and DB2/390 allows pages to be "fixed" or pinned in memory [11, 48]. Once a page is fixed, the buffer pool interface can be bypassed so that data within the page can be directly manipulated by the various DBMS components. This allows the DBMS components to use the buffer pool as working storage, thereby eliminating the need for the components to make local copies of the data. Consequently, there are references within the pinned pages that result from the direct manipulations by the DBMS components that are using the buffer pool as working storage. Since our traces were collected at the level of the buffer pool interface, they do not contain such references reflecting direct use of buffer pool storage as working storage.

3.2 Workload Description

The TPC-C benchmark is designed to model the workload of complex OLTP application environments [51]. To help users relate to the components of the benchmark, the benchmark has been given the life-like context of a wholesale supplier. The workload is centered around the order processing operations of this supplier, operations that are typical of companies that manage, sell or distribute products/services.

The supplier portrayed by the benchmark has a number of geographically distributed sales districts and associated warehouses. Each regional warehouse covers 10 districts each of which serves 3000 customers. Each district handles its customer information and the new orders placed by the customers. In addition, each district maintains the status of its orders and the history of customer orders. The status of each item ordered is tracked through an order-line table. All the warehouses maintain stocks for the 100,000 items sold by the company. As the supplier's business expands, new warehouses and associated sales districts are created.

A direct translation of this business context into a database design results in the nine tables specified in the benchmark: WAREHOUSE, DISTRICT, CUSTOMER, ORDER, NEW-ORDER, ORDER-LINE, STOCK, HISTORY and ITEM. Following the assumed business expansion path, the number of warehouses is the base unit of scaling for the TPC-C database. All other tables, except ITEM, scale with the number of warehouses according to the above-mentioned ratios. Our TPC-C trace was collected on a setup with 800 warehouses. The TPC-C transaction mix is designed to represent a complete business cycle. It consists of business transactions that enter new orders, query the status of existing orders, deliver outstanding orders, enter payments from customers and monitor warehouse stock levels. These five transaction types are summarized in Table 1. The TPC-C performance metric is the number of orders processed per minute.

To increase its realism, the TPC-C benchmark attempts to model several real world concepts such as distributed systems, data entry errors and access skew. The benchmark specifies that 1% of all items ordered are not in-stock at the regional warehouse and must be supplied by another warehouse. In addition, 1% of the New-Order transactions are chosen at random to simulate data entry errors to exercise the performance of rolling back update transactions. However, by disallowing the use of the input data from a rolled back transaction for a subsequent transaction, the benchmark does not attempt to model the resubmission of failed transactions.

Perhaps the most important aspect of the benchmark is that it is designed to model non-uniform data access. In particular, accesses to the CUSTOMER, ITEM and STOCK tables are skewed according to pre-specified non-uniform distributions. As reported in [31], the largest skew is seen in the STOCK table, where with proper tuple arrangement permitted by the benchmark, 84% of the accesses go to about 20% of the hottest pages. This is similar to the familiar "90/10" or "80/20" locality rule. The entire TPC-C benchmark is motivated and described in detail in [51].

While the TPC-C benchmark models the operational end of the business environment where real-time transactions are processed, the TPC-D benchmark models the analysis end of the business environment where trends are analyzed and refined to support sound business decisions. As is the case for the TPC-C benchmark, the TPC-D benchmark is given the realistic context of a wholesale supplier to help the user relate intuitively to the components of the benchmark. The TPC-D benchmark has eight required tables: REGION, NATION, SUPPLIER, PART, PARTSUPP, CUSTOMER, ORDER, and LINEITEM. The benchmark takes a scaling factor as parameter to determine the size of the tables. Our trace was taken on a setup of scale 30, which means that the two largest tables, ORDER and LINEITEM, contained 45 million and 180 million tuples respectively.

The TPC-D benchmark is comprised of a set of 17 business queries chosen to have broad industry-wide relevance. The TPC-D queries are far more complex than most OLTP transactions and typically examine large volumes of data using a rich set of operators and selectivity constraints. The TPC-D database is neither a one-time snapshot of a business operations database nor a database where OLTP applications are running concurrently. Rather, it is a decision support database that tracks, possibly with some delay, the OLTP database through batch updates. To exercise the update functionality of the DBMS, the TPC-D benchmark includes 2 update functions that modify a small percentage of the database. Table 2 summarizes the 17 queries and 2 update functions that make up the TPC-D benchmark.

The TPC-D benchmark defines a power test to measure the raw query execution power of a system with a single active user. It also defines a throughput test that a user may elect to omit. Our trace captures the entire run of a power test. The test starts off with the first update function (UF1). Next, the 17 queries are processed in a sequence specified by the benchmark. Finally, the second update function (UF2) is executed. Note that unlike the TPC-C benchmark, the TPC-D benchmark does not attempt to model data skew. More details about the TPC-D benchmark can be found in [52].

Both our TPC-C and TPC-D traces were collected in general compliance with the benchmark specifications. One notable exception was the TPC-C requirement that the transaction per minute to warehouse ratio be within the range of 9 to 12.7. We intentionally violated this requirement in order to trace a larger database. In addition, since we are primarily interested in the server workload, we did not go to the expense of setting up remote terminal emulators to generate the transactions for TPC-C. Instead, all our transaction requests were generated from a single client machine with no think time between transactions. Because the TPC benchmark rules prohibit publicly disclosing TPC performance figures that have not been independently audited, we withhold from this paper any data that may be used to derive our TPC metrics. This omission of absolute TPC performance numbers should not compromise our understanding of the logical characteristics of the benchmarks.

Our other traces were collected in the day-to-day production environments of a diverse group of very large corporations. The industries represented include aerospace, banking, consumer goods, direct mail marketing, financial services, insurance, retail, telecommunications and utilities. In all cases, our traces include the peak production database workload as identified by the system managers. This is typically a combination of transaction processing and long-running queries. The trace referred to as Telecom in [58] and Phone in [43] is the first 30 minutes of the trace we call TelecomB1.

3.3 Trace Description

Table 3 summarizes the characteristics of the various traces that are used in this paper. Because of the large number of production workloads, we often also present the arithmetic mean of their results. This is denoted as "Prod. Ave." In the table, the term *object* refers to a logical collection of data, such as a database table or an index, that is managed as an entity in much the same way as a file. *Data size* represents the total size of all the objects in the system and was obtained from the catalog dumps that were taken when the systems were traced. The *footprint* of a trace is defined as the amount of data referenced at least once in the trace. The traces record information from the perspective of the DBMS. Therefore, the object count includes DBMS system objects such as catalogs, views and query plans. In addition, the transactions recorded are database transactions, several of which may be needed to perform a single business transaction. The production traces were taken off the primary systems in use at some of the world's largest corporations in the early nineties. Though these databases were considered very large a few years ago, they are comparable in size to the TPC benchmark databases that can be supported on a high-end multiprocessor PC server today.

Figure 1 plots the trace footprint as a function of the number of references, which is a measure of the trace length. Because there is a wide variation in the footprint of our traces, we plot the footprint as a percentage of the total data size of the workload and use two different scales in Figures 1(a) and 1(b) to facilitate comparison among the workloads. From the figures, only the TPC-D, Bank, ConsGds and TelecomA traces approach steady state in the sense that they do not appear to be actively referencing new data. Though the ar-

Query	Name	Description
UF1	New sales update	Adds new rows representing 0.1% of the initial population to the ORDER and LINEITEM tables to emulate the addition of new sales information.
Q1	Pricing summary report	Provides summary pricing report for all parts shipped as of a given date which is within 60-120 days of the greatest ship date in the database.
Q2	Minimum cost supplier	Finds the supplier who can supply a given part in a given region at the lowest cost. If several suppliers tie, lists the suppliers with the 100 highest account balances.
Q3	Shipping priority	Retrieves the shipping priority and potential revenue of orders that have the largest revenue among those that have not been shipped as of a given date. Orders are listed in decreasing order of revenue up to a maximum of ten orders.
Q4	Order priority check	For each order priority, counts the number of orders entered in a given quarter in which at least one lineitem was received by the customer later than its committed date. Counts are listed in ascending priority order.
Q5	Local supplier volume	Lists for each nation in a region the revenue that was received from orders in which both the customer and supplier were within that nation.
Q6	Forecasting revenue change	Quantifies the increase in revenue that would have resulted from eliminating certain discounts on items that are below a given quantity in a given year.
Q7	Volume shipping	Finds, for 2 given nations, the gross discounted revenues derived from parts that were shipped between the nations during 1995 and 1996.
Q8	National market share	Determines how the market share of a given nation within a given region has changed from 1995 to 1996 for a given part.
Q9	Product type profit measure	Determines how much profit is made on a given line of parts, broken out by supplier nation and year. Lists the nations in alphabetical order and, for each nation, the year and profit in reverse chronological order.
Q10	Returned item reporting	Finds the customers who have returned parts that were ordered in a given quarter. Lists 20 customers in descending order of lost revenues.
Q11	Important stock identification	Finds the most valuable subset of suppliers' stock in a given nation. Displays the part number and the value in descending order of value.
Q12	Shipping modes and order priority	Counts, for 2 different shipping modes, the lineitems that were shipped before the commit date but were received by customers after the commit date in a given year. Partitions the late lineitems into two groups depending on their priority.
Q13	Sales clerk performance	Computes the loss of revenue on orders placed by a given clerk due to parts being returned by customers. Groups and orders the results by the year in which the parts were ordered.
Q14	Promotion effect	Determines the percentage of revenue in a given month that was derived from parts on promotion.
Q15	Top supplier	Finds the supplier(s) who contributed the most to the overall revenue for parts shipped during a given quarter.
Q16	Parts/supplier relationship	Counts the number of suppliers who have not had complaints registered at the Better Business Bureau and who can supply parts that are not of a given type and brand in 8 different sizes. Results are listed in descending order of count and ascending order of brand, type and size.
Q17	Small-quantity-order revenue	Looks at parts of a given brand and container type to determine the average lineitem quantity. Calculates the average yearly gross loss in revenue if orders for these parts with a quantity of less than 20% of this average were no longer taken.
UF2	Old sales update	Removes rows representing 0.1% of the initial population from the ORDER and LINEITEM tables to emulate the removal of stale or obsolete information.

Table 2: Summary of TPC-D's Queries and Update Functions. In the TPC-D power test, UF1 is executed, followed by queries 1 to 17 in an order defined by the benchmark, and then by UF2.

Trace	Aerospace	Bank	ConsGds	DirMktg1	DirMktg2	FinSvcs	Insurance	Retail	TelecomA	TelecomB1	TelecomB2	Utility	Prod. Ave.	TPC-C	TPC-D
Source	Aerospace company	Banking corp.	Consumer goods company	Direct mail marketing firm	Direct mail marketing firm	Financial services firm	Insurance company	Discount store	Telecom. Company A	Telecom. company B	Telecom. company B	Utility company	-	TPC benchmark C	TPC benchmark D
Platform	MVS on IBM S/370	MVS on IBM S/370	MVS on IBM S/370	MVS on IBM S/370	MVS on IBM S/370	MVS on IBM S/370	MVS on IBM S/370	MVS on IBM S/370	MVS on IBM S/370	MVS on IBM S/370	MVS on IBM S/370	MVS on IBM S/370	MVS on IBM S/370	WinNT on Intel X86	WinNT on Intel X86
DBMS	DB2/MVS	DB2/MVS	DB2/MVS	DB2/MVS	DB2/MVS	DB2/MVS	DB2/MVS	DB2/MVS	DB2/MVS	DB2/MVS	DB2/MVS	DB2/MVS	DB2/MVS	DB2/UDB	DB2/UDB
Date Collected	2/3/1992	5/13/1991	9/8/1992	9/18/1991	9/19/1991	6/6/1991	10/7/1992	7/1/1992	4/15/1992	10/8/1990	10/9/1990	5/14/1991	-	2/10/1998	3/8/1998
Duration (h:m)	2:29	22:57	1:59	1:03	2:02	3:54	2:41	4:52	1:40	2:27	1:42	3:16	4:15	(withheld)	(withheld)
# Objects	2203	1281	626	1446	1446	3124	1953	434	521	255	255	1139	1224	101	192
Data Size (MB)	33558	53079	3423	18191	18191	10064	38095	72188	197422	15114	15114	39070	42792	70246	77824
Footprint (MB)	1397	9600	726	1137	1362	2127	1732	6769	2986	947	976	5727	2957	13267	51580
# References	7779007	35916414	7133845	6401880	14396125	15664004	20648874	38646360	13072916	11531195	13757374	37653369	18550114	196067649	218130354
# Xacts	98931	85173	66102	11892	14906	20956	70242	797637	84378	36508	25899	118191	119235	890885	230
Read Ratio (%)	93.8	90.6	86.9	95.4	95.6	90.9	84.8	86.9	85.9	93.0	98.1	89.3	90.9	87.4	97.8

Table 3: Summary of Trace Characteristics. The term *object* refers to a logical unit of data, such as a database table or an index, that is managed like a file.

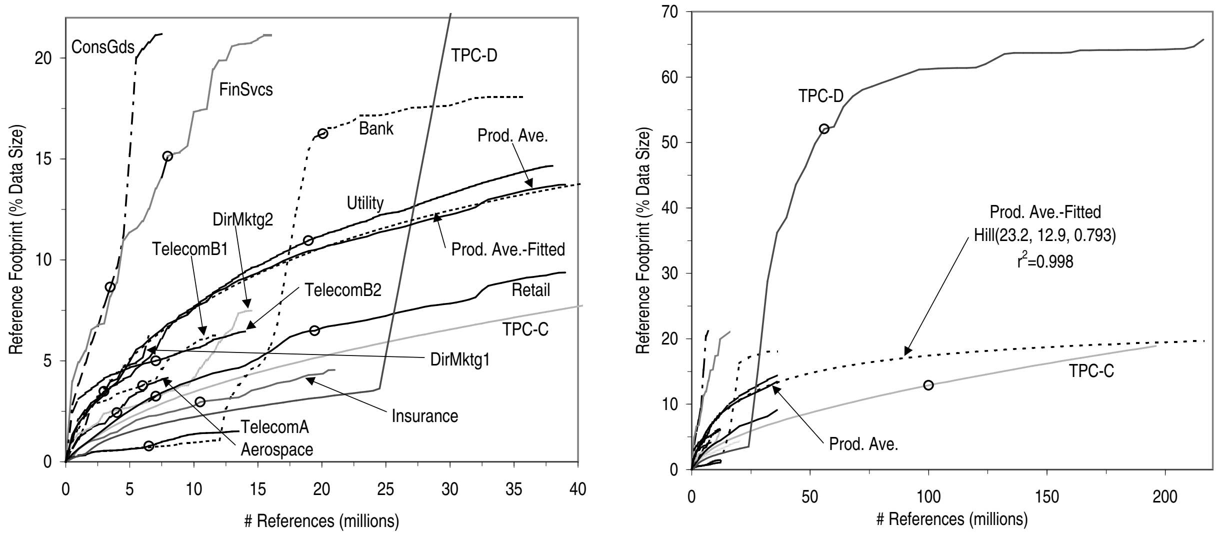
tificial nature of TPC-C is apparent in the smoothness of its footprint profile, the rate at which it references new pages is within the spectrum defined by the other traces. The write footprint profiles are presented in Figure 2. These profiles show how the percentage of pages written increases with the number of references. Compared to most of the production traces, the TPC traces generate modified pages at a much higher rate. We will examine the write behavior of the various workloads in greater detail in [20].

An important issue in using trace-driven simulations to study memory hierarchy design is that the traces must have a sufficiently large footprint for the memory configurations of interest. However, estimating the length of trace required is difficult because the relationship between the trace length and footprint is not well-understood. In this paper, we empirically determine this relationship by looking at the average footprint of our production traces. Because the traces are of different lengths, if we simply average the footprints, the number of traces being averaged will decrease with the trace length so that the resulting curve will contain discontinuities. Therefore, we take the average of the rate of increase of the footprint and then integrate the resulting expression. More formally, we define the average footprint after X references as $\int_0^X \overline{\frac{d}{dx}(f_i(x))}\,dx$, where $f_i(x)$ denotes the footprint of trace i after x references. This is plotted as the lines labeled "Prod. Ave." in Figures 1 and 2. Note that we omit Bank in plotting the average because its footprint profile is clearly unlike any of the other production workloads.

We find that the relationship between trace length and footprint can be accurately described by the Hill equation which was originally proposed for modeling the absorption of oxygen by hæmoglobin [19]. The Hill model, $Hill(f_{max}, k, n)$, represents a family of sigmoidal saturation curves defined by $f(x) = \frac{f_{max} \cdot x^n}{k + x^n}$ where f_{max} is the asymptotic value of $f(x)$ and k and n are parameters that determine the shape and slope of the curve. In our current context, the value of f_{max} represents the percentage of data that is predicted to be in active use. For instance, from Figure 1, the model predicts that 23.2% of the data will be referenced if the trace is infinitely long. From Figure 2, the model predicts that only 9.84% of the data will be written to.

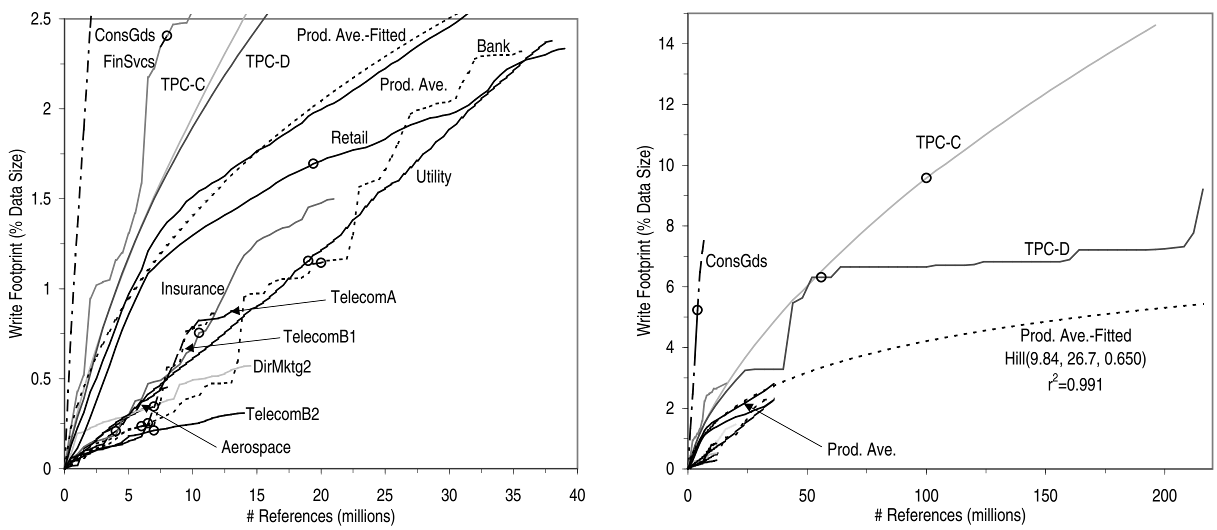
In the course of this research, there were situations where the state of various simulators had to be established before meaningful statistics could be collected. This is often referred to as warming up the simulator. For instance, the buffer pool in a real system is seldom empty, except during start up. Therefore, if we simulate the buffer pool miss ratio starting with an empty buffer pool, the results will be skewed by the extra misses that are needed to fill the buffer pool. A more meaningful approach is to collect the statistics after the buffer pool has been filled or warmed up. Such statistics are known as *warm* statistics. Unless otherwise stated, we used half of the trace for such warm-up purposes for most of the traces. Because the footprint of Bank increases abruptly around the middle the trace, we prolonged the warm-up period for Bank to slightly beyond the halfway mark. For the TPC-D trace, we used only a quarter of the trace to warm up our simulators because this already achieved a large enough footprint. The various warm-start points are presented in Table 4 and are also circled in Figures 1 and 2.

Since our traces were taken from two different kinds of systems and with different tracing mechanisms, we need to examine some of the underlying assumptions and compare the features that are present in the traces to make sure that we have a compatible set of data for our analysis. First, the TPC traces contain references to the database log while the production traces do not. Table 5 shows how significant log activity is in the TPC traces. Since the 17 queries in TPC-D



(a) (b)

Figure 1: Reference Footprint of the Traces as a Function of Trace Length. The default warm-start points for the simulations in [20] are circled.



(a) (b)

Figure 2: Write Footprint of the Traces as a Function of Trace Length. The default warm-start points for the simulations in [20] are circled.

Trace	Aerospace	Bank	ConsGds	DirMktg1	DirMktg2	FinSvcs	Insurance	Retail	TelecomA	TelecomB1	TelecomB2	Utility	TPC-C	TPC-D
# References	3889504	20000000	3566923	3200940	7198063	7832002	10324437	19323180	6536458	5765598	6878687	18826685	98033825	54532589
% References	50.0	55.7	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	25.0
% Trace Time	42.6	65.8	49.1	45.0	40.1	51.4	50.4	50.5	64.6	45.3	50.3	50.3	51.9	42.0

Table 4: Warm-Start Point.

Trace	Aerospace	Bank	ConsGds	DirMktg1	DirMktg2	FinSvcs	Insurance	Retail	TelecomA	Utility	Average	TPC-C	TPC-D
Blind Writes (%)	6.08	12.63	10.92	9.13	11.98	20.48	4.76	3.10	16.44	10.81	10.63	0.49	29.82

Table 6: Percent of Writes that are not Preceded by Reads of the Same Page.

Trace	TPC-C	TPC-D
Log Refs. (%)	11.63	1.31
Log Forces (%)	0.37	0.000072
Log Footprint (%)	0.000029	0.0000076
Overhead Refs. (%)	5.78	2.90

Table 5: Significance of Log Activity and References to Overhead Pages.

are read-only, there is very little log activity in the TPC-D trace. For the TPC-C trace, one out of every nine references is a log reference. Though the number of log references is significant, because of the use of group transaction commits and the fact that the log is usually used in an append mode where all the references are to the tail of the log, the number of physical log I/O (log force) as well as the log footprint are both insignificant when compared to the non-log activity. Moreover, since the I/O activity corresponding to the log is already well understood to consist primarily of sequential writes, we decided to delete the log references from the TPC traces so that they would contain data comparable to that collected from the real workloads.

Second, when write access is requested for a page, the buffer manager generally has to ensure that the page is present in the buffer pool. In special cases where the entire page is to be written, a "no-read" indication can be given to the buffer manager to indicate that it is not necessary to fetch the page from disk. We refer to such writes as blind writes. The TelecomB traces were collected with an early version of the DB2/390 tracing package that did not distinguish blind writes. In other words, all the writes in the TelecomB traces are preceded by reads of the same page. Table 6 shows how significant blind writes are in the various other traces. Since writes are a small fraction of the total references to begin with, the lack of blind write information in the TelecomB traces is not expected to have a significant impact on our analysis.

Third, all our traces record logical references to the database objects. In reality, these logical references have to be mapped to some physical space. Depending on whether the DBMS is set up to use raw partitions or the filesystem for storage, either the DBMS or the filesystem has to maintain overhead pages to store the logical address to physical address mapping and to keep track of the free space. The TPC traces include references to these overhead pages. As shown in Table 5, the overhead pages account for a small percentage of the total number of references. Therefore, to be consistent, we have also chosen to delete the references to the overhead pages in the TPC traces.

Finally, some of the traces contain references to large pages, *i.e.,* those with sizes that are multiples of the 4KB base page size. For consistency, we converted these to refer to 4KB pages.

4 Workload Characteristics

4.1 Transaction Characteristics

Transactions are the building blocks of a database workload. The characteristics of transactions are therefore good reflections of the nature of the workload. Table 7 summarizes the transaction characteristics of our workloads. In this table, we consider both the logical and physical read ratio. The former is defined in terms of references to permanent objects only while the latter accounts for references to both temporary and permanent objects. The terminology stems from the fact that references to temporary objects are not intrinsic to the transaction but are a function of physical constraints such as memory size. Table 7 also contains data on the *page reuse* of transactions. This is defined as the ratio of the number of references to the number of pages referenced and is an indication of the locality of reference exhibited by the transactions.

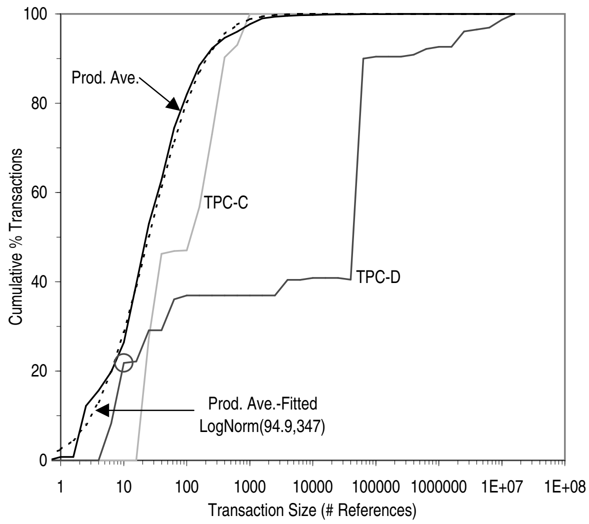
The table shows that the production workloads are very diverse in their transaction characteristics. In certain cases, however, TPC-C and TPC-D still fall outside the broad range of behavior exhibited by the production workloads. For instance, the proportion of logically read-only transactions in the production workloads varies from 19% in TelecomA to 90% in Utility with an average of about 60%. On the other hand, only 8% of TPC-C's transactions are logically read-only. Since read-only transactions are easier to isolate from one another, this suggests that TPC-C stresses the concurrency control mechanism more than the production workloads. Notice also that the TPC-D transactions have a lot more references than those of the production workloads but they involve fewer objects and have much better locality.

Figure 3 plots the distribution of *transaction size*, which is the number of references in a transaction. The transactions in the TPC benchmarks, especially those in TPC-D, tend to be larger than those of the production workloads. In

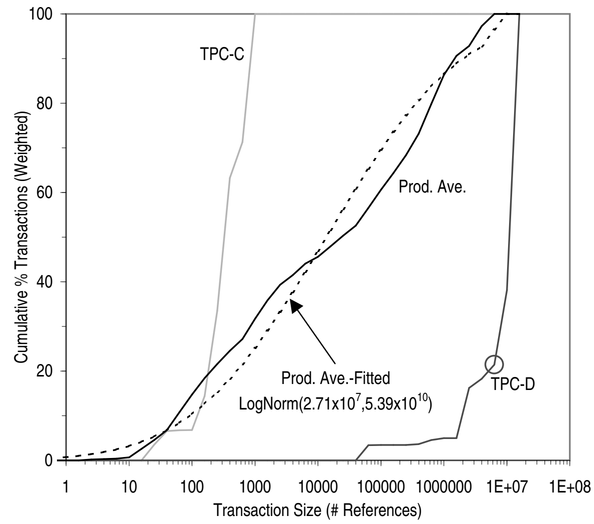
	Trace	Aerospace	Bank	ConsGds	DirMktg1	DirMktg2	FinSvcs	Insurance	Retail	TelecomA	TelecomB1	TelecomB2	Utility	Prod. Ave.	TPC-C	TPC-D
Xacts	#	98931	85173	66102	11892	14906	20956	70242	797637	84378	36508	25899	118191	119235	890885	230
	% Read-only[1]	76.4	43.4	27.6	74.3	78.2	71.2	81.2	32.3	19.3	60.8	59.7	89.8	59.5	7.96	37.4
	% Read-only[2]	74.3	41.1	21.1	60.0	60.1	59.4	54.3	32.1	11.7	56.1	54.3	38.2	46.9	7.96	32.6
Read Ratio (%)[1]	Ave. (%-tile)	95.6 (22.9)	86.0 (48.7)	90.3 (54.7)	96.2 (21.9)	97.0 (19.3)	94.7 (24.8)	96.9 (15.6)	93.1 (39.0)	88.3 (38.5)	95.3 (36.1)	95.1 (35.5)	98.8 (10.1)	93.9 (30.6)	86.5 (54.5)	89.1 (59.1)
	Median	100	92.3	88.3	100	100	100	100	94.5	92.6	100	100	100	97.3	84.6	82.2
	Std. Dev.	8.74	13.97	7.99	7.82	7.04	10.1	9.47	7.06	14.6	8.25	8.45	4.59	9.01	4.84	8.74
	90%-tile	100	100	100	100	100	100	100	100	100	100	100	100	100	91.4	100
	10%-tile	80	69.2	80	82.3	85.7	78.0	90.2	83.3	50	85.7	85.2	98.3	80.7	82.8	82.1
Read Ratio (%)[2]	Ave. (%-tile)	95.1 (24.9)	85.6 (50.0)	88.2 (57.6)	94.7 (29.3)	95.0 (27.6)	92.6 (32.1)	93.2 (38.2)	91.8 (45.9)	83.4 (39.5)	94.0 (34.4)	93.8 (34.6)	89.8 (45.7)	91.4 (38.3)	86.5 (54.5)	88.9 (59.6)
	Median	100	85.5	87.5	100	100	100	100	93.3	86.4	100	100	91.4	95.4	84.6	82.2
	Std. Dev.	9.33	13.8	8.56	8.73	8.50	11.5	10.8	7.51	14.1	9.38	9.46	10.4	10.2	4.84	8.61
	90%-tile	100	100	100	100	100	100	100	100	100	100	100	100	100	91.4	100
	10%-tile	80	69.2	77.3	80.8	80.6	73.7	82.2	80	50	80.8	80.8	75	75.9	82.8	82.1
# References	Ave. (%-tile)	78.6 (90.6)	422 (96.1)	108 (78.4)	538 (91.1)	966 (90.3)	747 (97.0)	294 (90.7)	48.5 (86.0)	155 (95.2)	316 (88.2)	531 (90.7)	319 (93.8)	376.9 (90.7)	220 (58.1)	948393 (92.2)
	Median	4	29	39	24	29	25	32	19	51	49	51	28	31.7	153	63892
	Std. Dev.	4306	15739	381	8889	15968	17602	11708	7951	5439	8948	28885	24724	12545	247	6273027
	90%-tile	68	162	197	433	895	319	257	63	115	400	459	232	300	492	117929
	10%-tile	2	6	7	3	3	4	3	13	4	8	8	3	5.3	26	10
# Pages Refed	Ave. (%-tile)	22.3 (84.6)	119 (96.3)	61.6 (77.2)	120 (89.7)	144 (88.5)	152 (89.8)	56.4 (78.9)	16.6 (64.7)	56.6 (90.7)	83.4 (82.5)	74.5 (80.0)	57.7 (80.6)	80.3 (83.6)	73.0 (57.9)	87509 (90.9)
	Median	3	28	21	14	15	14	21	13	26	35	36	18	20.3	61	12501.5
	Std. Dev.	439	2715	351	903	1123	1833	366	328	1350	547	743	1517	1018	81.3	484698
	90%-tile	33	90	90	124	242	159	114	29	53	115	117	76	104	161	24776
	10%-tile	2	3	6	3	3	3	3	5	2	7	7	3	3.9	12	4
# Pages Written	Ave. (%-tile)	1.81 (81.5)	12.4 (87.6)	5.93 (63.4)	3.77 (79.7)	6.98 (90.0)	16.0 (94.0)	6.37 (82.8)	2.11 (65.5)	9.88 (84.8)	5.45 (72.5)	4.79 (64.8)	5.66 (82.1)	6.76 (79.1)	16.5 (51.0)	14068 (92.6)
	Median	0	3	2	0	0	0	0	1	4	0	0	3	1.1	4	6003
	Std. Dev.	16.2	454	8.77	63.0	178	290	178	141	938	86.5	18.7	306	223	14.7	59347
	90%-tile	3	16	13	6	7	10	13	4	13	8	8	8	9.1	36	6298
	10%-tile	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0
# Objs. Refed	Ave. (%-tile)	4.76 (69.9)	11.8 (56.4)	15.6 (65.9)	8.19 (65.1)	8.70 (62.4)	8.40 (66.6)	13.0 (62.9)	7.14 (60.8)	11.4 (46.8)	11.8 (51.7)	12.0 (56.5)	12.9 (61.9)	10.5 (60.6)	10.0 (55.0)	4.6 (80.4)
	Median	2	8	8	6	7	6	7	6	12	11	11	7	7.6	7	4
	Std. Dev.	5.43	10.9	14.0	7.29	7.88	8.21	16.2	4.55	7.73	8.42	8.60	12.9	9.34	3.65	3.43
	90%-tile	13	35	37	17	18	18	31	14	21	22	26	29	23.4	14	6
	10%-tile	1	2	2	2	2	2	1	2	2	4	4	2	2.2	7	2.1
Page Reuse	Ave. (%-tile)	1.55 (74.4)	2.12 (80.2)	1.85 (60.8)	2.62 (74.9)	3.18 (77.5)	2.38 (73.2)	1.88 (70.0)	1.73 (60.3)	2.97 (92.3)	3.01 (87.2)	3.26 (87.3)	2.04 (65.8)	2.38 (75.3)	2.75 (63.3)	5.79 (86.1)
	Median	1	1.47	1.71	1.45	1.53	1.55	1.36	1.58	2	1.34	1.4	1.79	1.52	2.5	5.11
	Std. Dev.	2.43	5.75	1.08	5.04	9.56	11.6	2.68	1.09	211	5.57	6.09	1.76	22.0	1.17	5.14
	90%-tile	2.05	3.33	2.33	4.6	5.55	3.21	3.34	2.47	2.67	3.81	4.12	3	3.37	3.43	9.17
	10%-tile	1	1	1	1	1	1	1	1	1.25	1	1	1	1.02	1.94	2.75

[1] Logical read ratio. Only references to permanent objects are considered.
[2] Physical read ratio. References to both temporary and permanent objects are considered.

Table 7: Transaction Characteristics. We use %-tile to denote the percentile at which the average value occurs.



(a) Distribution of Transaction Size. The circled point shows that 22% of TPC-D's transactions contain not more than 10 references.



(b) Distribution of Transaction Size Weighted by Transaction Size. The distribution is weighted in the sense that a transaction of size s is counted s times. The circled point indicates that 21% of TPC-D's references are caused by transactions that contain fewer than 6,300,000 references.

Figure 3: Number of References Per Transaction.

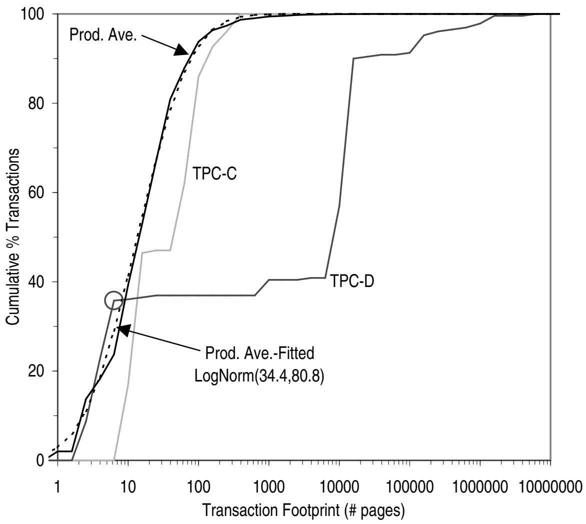
	Aerospace	Bank	ConsGds	DirMktg1	DirMktg2	FinSvcs	Insurance	Retail	TelecomA	TelecomB1	TelecomB2	Utility	Prod. Ave.	TPC-C	TPC-D
E[S]	78.6	422	108	538	966	747	294	48.5	155	316	531	319	377	220	9.48×10^5
E[S²]	1.85×10^7	2.48×10^8	1.57×10^5	7.93×10^7	2.56×10^8	3.10×10^8	1.37×10^8	6.32×10^7	2.96×10^7	8.02×10^7	8.35×10^8	6.11×10^8	2.22×10^8	1.09×10^5	4.03×10^{13}
E[S³]	1.65×10^{13}	5.23×10^{14}	5.46×10^9	3.85×10^{13}	1.79×10^{14}	3.98×10^{14}	1.97×10^{14}	3.93×10^{14}	2.12×10^{13}	8.04×10^{13}	2.69×10^{15}	2.48×10^{15}	5.85×10^{14}	7.51×10^7	3.04×10^{21}

Table 8: First, Second and Third Moments of the Number of References in a Transaction (S).

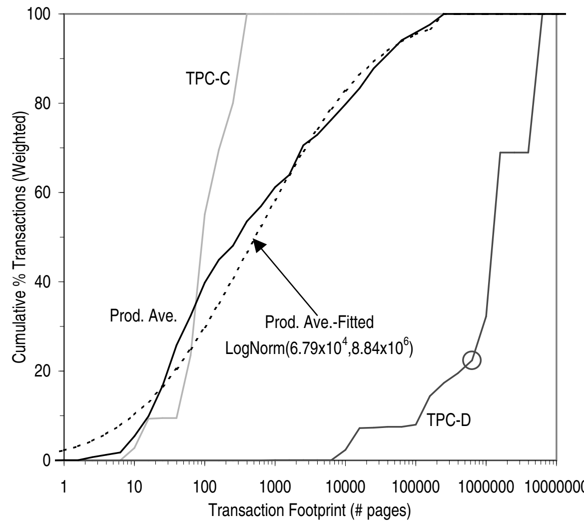
addition, TPC-D's transactions have a wide range in sizes. When two-phase locking is used to ensure that transactions are serializable [13], locks tend to be released only when transactions end so that long transactions typically imply long lock waits. Therefore the transaction size, which can be considered the virtual transaction length or duration, is a very important factor in analyzing concurrency control mechanisms. To make our data more useful for mathematical modeling, we fitted it with standard probability distributions. As shown in Figure 3, the lognormal distribution (denoted LogNorm(μ,σ) where μ is the mean and σ is the standard deviation) turns out to be a very good fit.

Since short transactions can be blocked for long periods by long transactions holding the necessary locks, system performance is sensitive to the second and third moments of the transaction size [49]. In addition, the distribution of transaction size affects not only the absolute but also the relative performance of different concurrency control schemes [43]. Therefore, we also present the average and higher moments of the transaction size for our various workloads in Table 8.

In Figure 4, we plot the distribution of *transaction footprint* or the number of pages referenced by a transaction. The lognormal distribution is again a very good fit. Figures 3 and 4 show that most of the transactions are small but large transactions account for most of the references and most of the pages referenced. In contrast to TPC-C, the production workloads are made up of transactions with a wide range of sizes and footprints. That there is such a mixture of large and small transactions complicates the task of scheduling and allocating resources to satisfy the different performance requirements of the transactions. For instance, a suitable balance has to be found between allowing large transactions to make good forward progress and preventing them from monopolizing the buffer pool. Regrettably, this issue is beyond the scope of the current study, which only considers the characteristics of workloads as they have been scheduled and tuned in production environments.



(a) Distribution of Transaction Footprint. The circled point indicates that 36% of TPC-D's transactions reference fewer than 7 pages.



(b) Distribution of Transaction Footprint Weighted by Transaction Footprint. The distribution is weighted in the sense that a transaction with footprint f is counted f times. The circled point shows that 22% of the pages referenced by TPC-D are due to transactions that reference fewer than 630,000 pages.

Figure 4: Number of Pages Referenced Per Transaction.

4.2 Degree of Concurrency

In order to effectively utilize system resources, database systems allow the concurrent execution of multiple transactions through concurrency control mechanisms, such as locking, that provide each transaction with an isolated view of the system. The degree of concurrency, *i.e.,* the number of concurrently active transactions, in a workload directly affects issues such as lock contention and deadlocks. Furthermore, for each active transaction in the system, the DBMS has to maintain a database agent and its associated context, which is non-trivial and includes various control blocks and private memory. The time-averaged number of transactions that are active in the various workloads at any one time is summarized in the last row of Table 9. The production workloads again exhibit very diverse characteristics with the time-averaged degree of concurrency ranging from slightly below 5 in ConsGds to nearly 80 in Aerospace.

Dynamically creating a database agent can be a significant part of the cost in short and medium size transactions. In situations where the degree of concurrency is rather constant, the agents and private resources can be held and reused. Figure 5 shows how the degree of concurrency in the various workloads vary over time. The very static profiles for both TPC-C and TPC-D stand in stark contrast to those of the production workloads and imply that the TPC benchmarks will not exercise the agent creation process of the DBMS. For a more quantitative characterization of the extent to which the degree of concurrency fluctuates over time, we time-averaged the degree of concurrency over intervals ranging from 100 milliseconds to the trace length. The maximum values observed for each of these interval sizes are presented in Table 9. We also plot the distribution of the degree of concurrency time-averaged over one-second periods in Figure 6. As shown in the figure, the lognormal distribution is a reasonably good fit for the average of the production workloads.

4.3 Object Characteristics

For performance reasons, most DBMSs offer an option to bypass the filesystem provided by the operating system to directly access the raw storage devices. In this case, the DBMS provides its own basic filesystem functionality such as allocating storage and tracking free space. In this section, we look at the characteristics of the objects in the various workloads to better understand what is required of the underlying filesystem, whether it is provided by the operating system or the DBMS.

The total number of objects and the fraction of them that are referenced or modified are presented in Table 10. The total object count was obtained from the catalog dumps that were taken when the systems were traced. Notice that the production workloads have significantly more objects than

Trace	Aerospace	Bank	ConsGds	DirMktg1	DirMktg2	FinSvcs	Insurance	Retail	TelecomA	TelecomB1	TelecomB2	Utility	Prod. Ave.	TPC-C	TPC-D
Inst.	132	27.0	16.0	25.0	24.0	48.0	27.0	138	17.0	20.0	16.0	40.0	44.2	60.0	8.00
100-ms	131	27.0	15.0	25.0	23.0	48.0	26.0	137	17.0	19.6	16.0	38.9	43.6	60.0	8.00
1-s	128	27.0	12.8	24.3	22.6	48.0	26.0	134	15.4	19.1	16.0	36.0	42.4	60.0	8.00
10-s	125	26.8	11.0	20.9	20.4	47.5	24.3	130	13.8	18.2	14.2	27.7	40.0	60.0	8.00
1-min	124	25.7	8.8	19.7	18.1	33.9	20.7	122	11.6	15.5	12.3	15.9	35.7	60.0	8.00
10-min	119	21.2	7.3	14.0	13.4	29.6	17.2	79.1	8.9	9.4	8.0	13.2	28.4	60.0	8.00
100-min	108	11.0	5.0	12.2	11.9	25.9	13.2	53.5	5.7	6.4	6.4	12.3	22.7	60.0	5.59
Trace Len.	77.7	5.39	4.91	12.1	11.6	20.5	12.8	42.7	6.50	5.06	6.32	11.3	18.1	60.0	3.07

Table 9: Degree of Concurrency Averaged over Various Time Intervals. The table shows the peak or maximum value observed for each interval size. The instantaneous maximum is denoted by "Inst.". The row labeled "Trace Len." is essentially the degree of concurrency time-averaged over the entire trace.

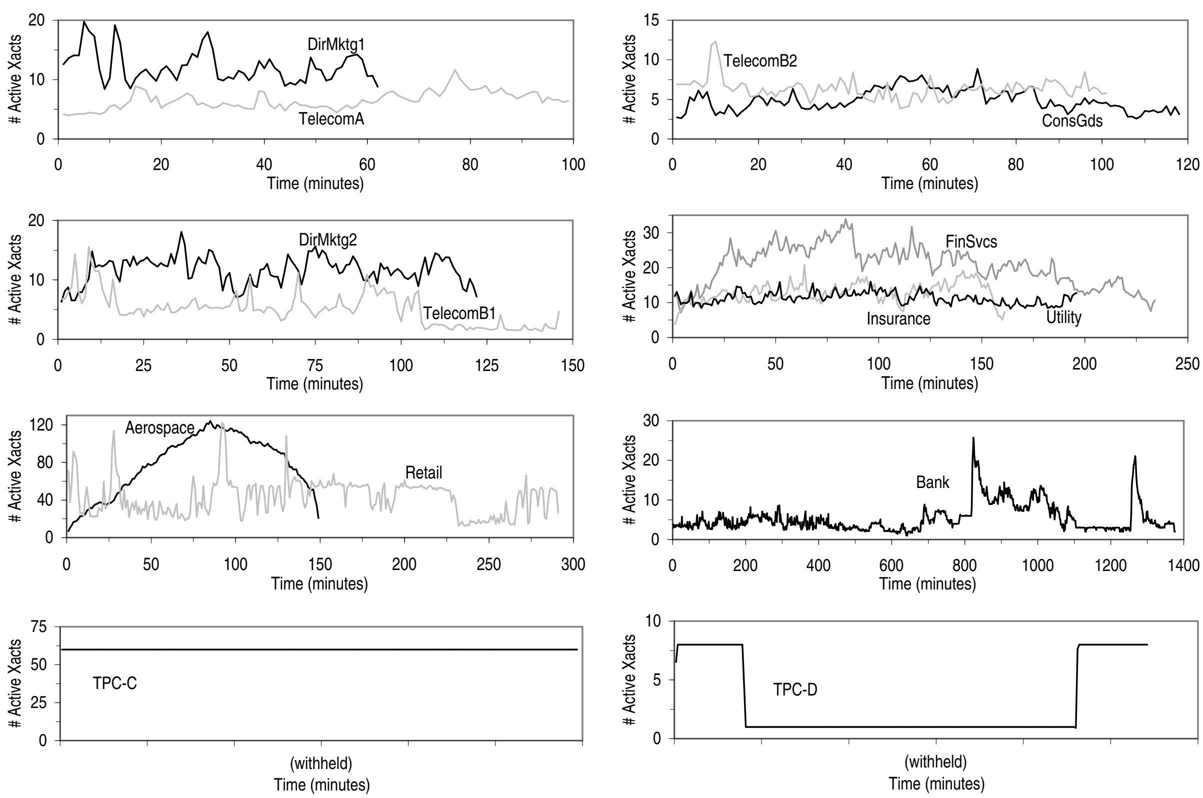


Figure 5: Profile of Degree of Concurrency Over Time. The data in this figure have been smoothed by averaging over one-minute intervals.

	Trace	Aerospace	Bank	ConsGds	DirMktg1	DirMktg2	FinSvcs	Insurance	Retail	TelecomA	TelecomB1	TelecomB2	Utility	Prod. Ave.	TPC-C	TPC-D
Objects	Total #	2203	1281	626	1446	1446	3124	1953	434	521	255	255	1139	1224	101	192
	% Refed	42.9	79.5	39.6	32.3	36.3	35.2	39.7	43.8	48.6	60	62.0	54.3	47.9	15.8	37.5
	% Modified	22.0	37.9	20.6	15.3	16.0	14.5	21.3	24.4	26.9	36.1	38.4	36.3	25.8	11.9	22.9
Pages	Total #	8590909	13588236	876401	4656812	4656812	2576270	9752447	18480252	50539937	3869199	3869199	10002028	10954875	17982935	19922913
	% Refed	4.16	18.1	21.2	6.25	7.49	21.1	4.55	9.38	1.51	6.27	6.46	14.7	10.1	18.9	66.3
	% Modified	0.45	2.32	7.78	0.34	0.57	2.83	1.50	2.33	0.882	0.864	0.310	2.38	1.88	14.6	9.93

Table 10: Reference Activity on Object and Page Bases.

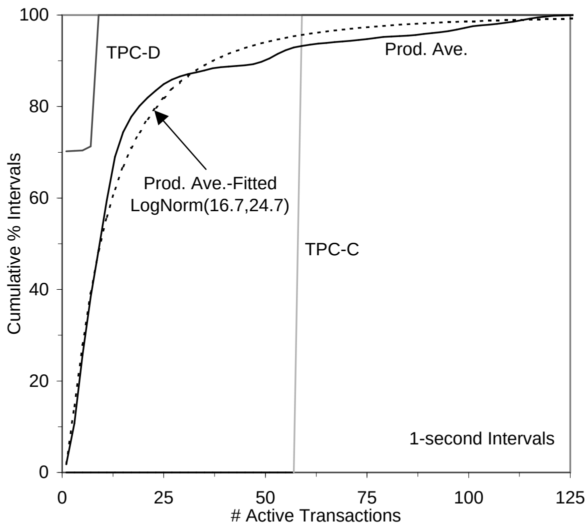


Figure 6: Distribution of Degree of Concurrency Time-Averaged over One-Second Intervals. The distribution for one-minute intervals is virtually identical.

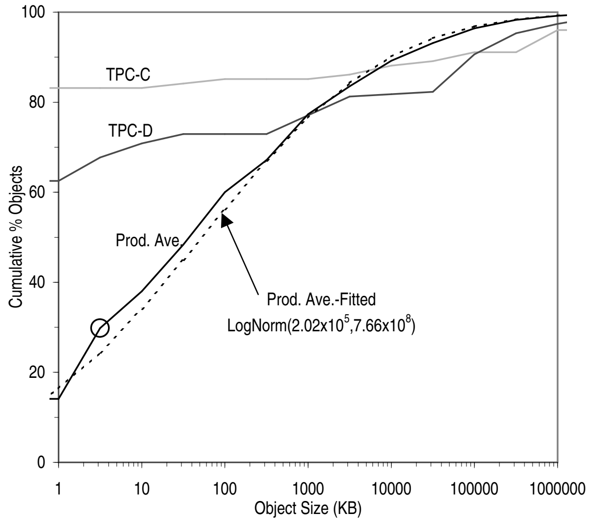
the two benchmarks. This is not surprising because the benchmarks are supposed to be distillations of real environments and should therefore contain only the core portions of the real workloads. Furthermore, the benchmark traces were collected on DB2/UDB which considers the various indices of a table as a single object instead of individual objects. Figure 7 presents the distribution of object size. Observe that the object size, like the transaction size, tends to approximately follow a lognormal distribution. In addition, most of the objects are small but the very large objects account for most of the bytes. This is similar to what has been observed in a general UNIX filesystem although the scale there is much smaller [32]. Interestingly, the distribution of file sizes in PCs running Windows in an office enivironment has also been recently reported to follow a lognormal distribution but the files are again much smaller than the objects in the database workloads [10].

Notice from Table 10 that, for most of the workloads, less than half of the objects are referenced for the duration of the trace. In general, a common approach to improving computer system performance is to place the items that are likely to be used in faster storage. At the system level or in other words, external to the DBMS, we can statically allocate to faster storage (*e.g.,* solid-state disks) the hottest objects, *i.e.,* those with the highest density (rate per byte) of reference. This approach reflects what has been referred to as the "$\lambda_{i,j}$" model [45] in which a transaction stream i references object j as a Poisson process with rate $\lambda_{i,j}$. Under such a model, an optimal static allocation should give non-lookahead optimal results, as with the A_o algorithm for the independent reference model for program behavior [1]. We consider the performance potential of such an approach in Figure 8.

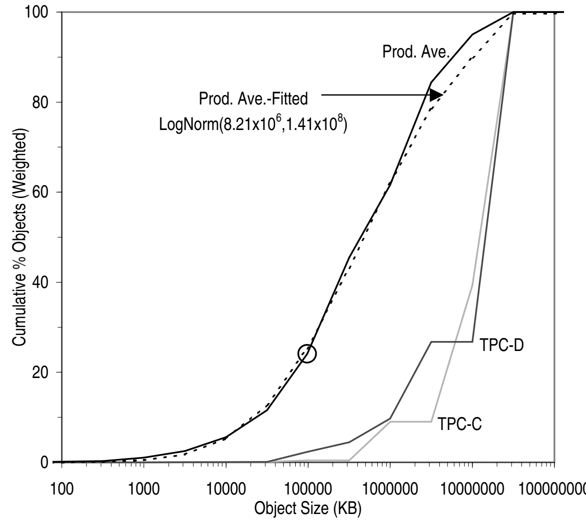
Figure 8(a) shows that a small number of objects account for most of the references. This skew in the access pattern is common in computer systems and has been expressed as the "90/10" or "80/20" locality rule. For instance, in 1971, Knuth observed that the nth most important statement in a set of FORTRAN programs accounts for $(\alpha - 1)\alpha^{-n}$ of running time, where α is a parameter [30]. We refer to this model of reference skew as the Knuth(α) model. As shown in Figure 8(a), we fitted the data for our production workloads with this function. Since the fit is not very good, we also experimented with more complicated functions. It turned out that the Hill equation [19] discussed in Section 3.3 is a much better fit.

Since the objects are of different sizes, we need to account for their sizes to fully understand the potential benefit of allocating hot objects to faster storage. This is done in Figure 8(b). Notice that the production workloads on average have a much higher reference skew than the two TPC benchmarks. This suggests that the production workloads will generally be more amenable to strategies that attempt to statically optimize data placement on an object basis. In [20], we further consider the static management of faster storage on a page basis and the results indicate that dynamic management offers a dramatically better hit ratio. This is in line with conclusions in [45] and indicates that reference probabilities are clearly time varying and the $\lambda_{i,j}$ model, like the independent reference model for programs, is not valid.

Handling write operations, especially those that involve a small number of pages, is the Achilles heel of certain classes of storage systems such as those based on RAID-5 [8]. In

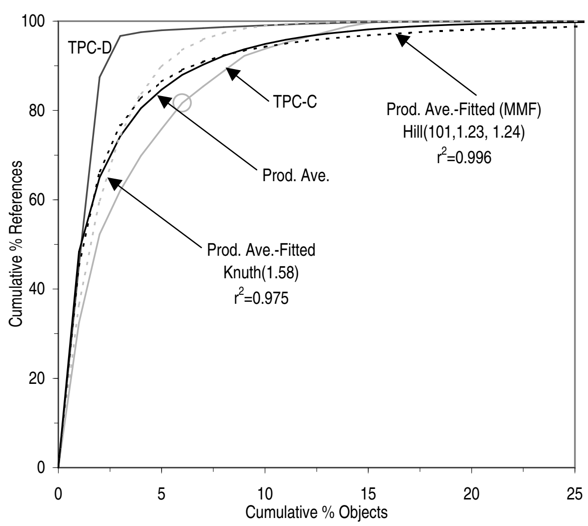


(a) Distribution of Object Size. The circled point shows that on average, 30% of the objects in the production environments are 3KB or smaller.

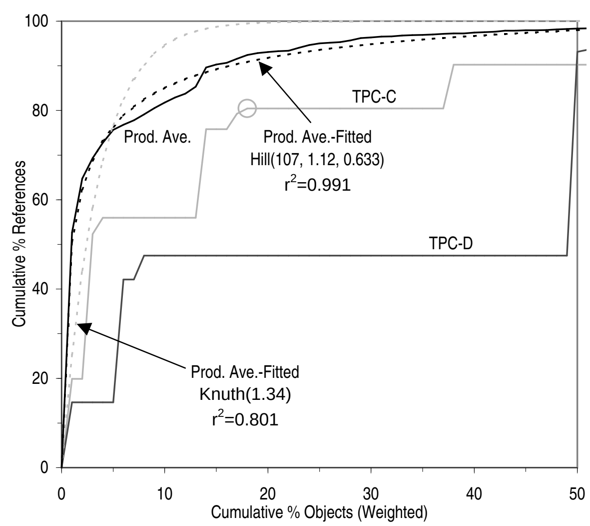


(b) Distribution of Object Size Weighted by Object Size. The distribution is weighted in the sense that an object of size s is counted s times. The circled point indicates that on average, objects smaller than 100MB account for only 24% of the total data size in the production environments.

Figure 7: Size of the Objects in the Various Workloads.

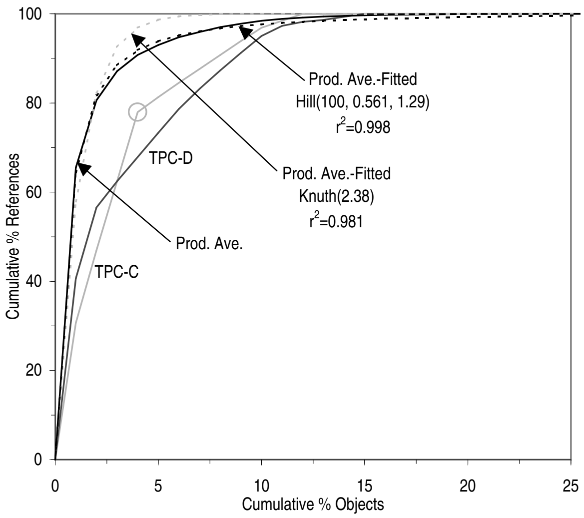


(a) Pareto Plot of the Number of References Per Object. The circled point indicates that for TPC-C, 6% of the hottest objects account for 82% of the references.

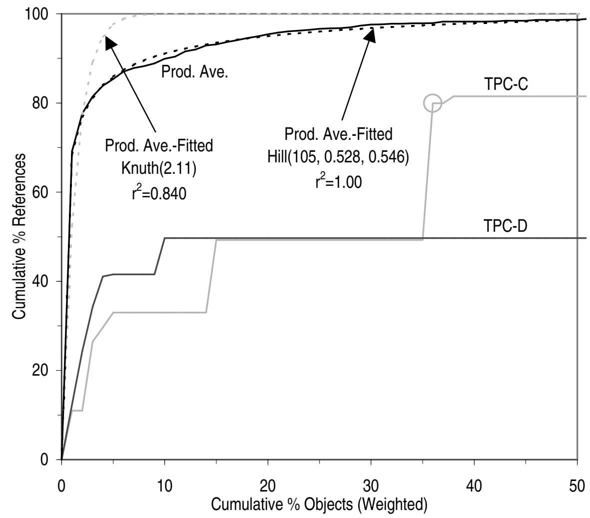


(b) Pareto Plot of the Number of References Per Object Weighted by the Object Size. In this plot, an object of size s is counted s times. The circled point shows that for TPC-C, 80% of the references are directed at objects that constitute 18% of the total data size.

Figure 8: Reference Skew on an Object Basis.



(a) Pareto Plot of the Number of Writes Per Object. The circled point shows that for TPC-C, 4% of the hottest objects account for 78% of the writes.



(b) Pareto Plot of the Number of Writes Per Object Weighted by the Object Size. In this plot, an object of size s is counted s times. The circled point indicates that for TPC-C, 80% of the writes are directed at objects that constitute 36% of the total data size.

Figure 9: Write Skew on an Object Basis.

such systems, data are stripped across multiple disks and are protected from individual disk failures by parity blocks that are scattered among the disks. Each parity block protects a stripe of data and has to be updated whenever any page within the stripe is written. For write operations that do not involve all the pages in a stripe, generating the new parity will involve reading the old parity and either the old data or the data that is not being updated. For such small write operations, mirroring, also known as RAID-1, offers better performance because no reads are required. However, RAID-1 tends to be more expensive than RAID-5 in that it uses more disks. This suggests that it may make sense to locate the frequently modified objects in RAID-1 and the more static objects in RAID-5. We consider the effectiveness of such an approach by plotting the write skew in Figure 9. As is the case for the reference skew, we find that the write skew can be accurately described by the Hill equation. From Figure 9(b), the write skew taking into account the size of the objects is generally less pronounced than the reference skew but is still very significant for the production workloads. Again, the two TPC benchmarks show a lot less skew at the object level than do the production workloads.

In Table 11, we break down the objects into data objects, index objects and temporary or work file objects. Observe that although the data pages account for the majority of pages in most of the workloads, index objects account for the largest chunk of references. This suggests that studies that do not consider index references, such as [31], may not give the complete picture. Notice further that most of the objects in the production workloads are index objects but this is not the case in the TPC benchmarks. Part of the reason is that, as mentioned above, the various indices of a table are considered a single object in DB2/UDB. Another observation from Table 11 is that the temporary objects may account for up to 80% of the write traffic and must therefore be considered when characterizing the write behavior of the workloads. Furthermore, except for TPC-C, which has no activity to temporary objects, the temporary objects account for a very significant portion of the modified pages.

Note that TPC-D tends to stand out among the workloads. In particular, the ratio of index references to data references in TPC-D is a high 17. Part of the reason is that in TPC-D's Update Function 1 (UF1), we append the records to be inserted so that it is possible to insert one whole page of records with only one data reference. Perhaps the bigger reason is that so much effort has gone into optimizing TPC-D that we can create indices that contain all the data needed by the queries. This allows "index-only" access where there is no need to probe the base table after an index lookup. In some sense, data is replicated in the indices, which partly explains why our TPC-D setup contains more index pages than data pages. Notice also that less than 1% of the modified pages in TPC-D are data pages and that less than 10% of the writes update data pages. Instead, most of TPC-D's

	Trace	Aerospace	Bank	ConsGds	DirMktg1	DirMktg2	FinSvcs	Insurance	Retail	TelecomA	TelecomB1	TelecomB2	Utility	Prod. Ave.	TPC-C	TPC-D
Objects	Total #	2203	1281	626	1446	1446	3124	1953	434	521	255	255	1139	1224	101	192
	% Data	44.5	42.2	39.6	35.1	35.1	48.2	48.9	46.1	45.1	43.9	43.9	40.2	42.7	50.5	60.4
	% Index	55.2	57.3	52.2	64.7	64.7	51.6	50.6	52.8	54.1	54.1	54.1	59.4	55.9	49.5	26.0
	% Temp	0.2	0.5	8.1	0.2	0.2	0.2	0.5	1.2	0.8	2.0	2.0	0.4	1.4	0	12.5
Refed Objs	Total #	945	1019	248	467	525	1101	775	190	253	153	158	619	538	16	72
	% Data	40.3	51.8	41.5	39.2	38.1	42.6	43.6	48.9	43.5	37.9	36.7	42.6	42.2	50	44.4
	% Index	59.3	47.5	56.9	60.4	61.5	55.9	55.2	48.9	54.9	59.5	60.8	56.7	56.5	50	19.4
	% Temp	0.4	0.7	1.6	0.4	0.4	0.5	1.2	2.1	1.6	2.6	2.5	0.6	1.2	0	33.3
Modified Objs	Total #	484	485	129	221	231	453	416	106	140	92	98	413	272	12	44
	% Data	38.8	38.4	42.6	43.0	43.3	48.8	42.3	50	46.4	32.6	32.7	45.3	42.0	66.7	40.9
	% Index	60.3	60.2	54.3	56.1	55.8	50.1	55.5	46.2	50.7	63.0	63.3	53.8	55.8	33.3	4.5
	% Temp	0.8	1.4	3.1	0.9	0.9	1.1	2.2	3.8	2.9	4.3	4.1	1.0	2.2	0	54.5
Pages	Total #	8590909	13588236	876401	4656812	4656812	2576270	9752447	18480252	50539937	3869220	3869220	10002028	10954879	17982935	19922913
	% Data	75.9	57.8	80.8	66.6	66.6	87.3	72.2	82.2	68.5	69.2	69.2	78.4	72.9	84.1	45.2
	% Index	24.0	41.0	19.2	33.2	33.2	11.2	27.8	17.7	31.5	30.2	30.2	21.6	26.7	15.9	50.8
	% Temp	0	1.2	0	0.2	0.2	1.5	0.1	0	0	0.6	0.6	0	0.4	0	4.1
Refed Pages	Total #	357605	2457640	185803	291094	348582	544496	443480	1732833	764358	242472	249878	1466054	757025	3396262	13204481
	% Data	75.0	86.2	73.8	69.8	78.8	84.0	68.7	62.8	70.0	60.5	65.5	73.8	72.4	74.0	50.8
	% Index	24.6	6.9	26.1	28.7	18.0	8.8	29.8	36.9	29.5	30.0	33.3	26.1	24.9	26.0	43.1
	% Temp	0.4	6.8	0.1	1.5	3.2	7.3	1.6	0.3	0.5	9.4	1.1	0.2	2.7	0	6.1
Modified Pgs	Total #	38964	315518	68192	15834	26654	72912	146213	431483	445787	33423	12006	237996	153749	2627637	1979231
	% Data	39.5	22.2	63.8	32.9	29.1	32.5	69.8	66.9	77.4	10.9	26.7	63.2	44.6	83.5	0.8
	% Index	56.8	24.6	35.9	38.8	28.7	13.3	25.5	31.8	21.7	20.7	49.7	35.8	31.9	16.5	58.4
	% Temp	3.7	53.1	0.4	28.3	42.2	54.2	4.7	1.2	0.9	68.4	23.6	1.0	23.5	0	40.9
References	Total #	7779007	35916414	7133845	6401880	14396125	15664004	20648874	38646360	13072916	11531195	13757374	37653369	18550114	1.96E+08	2.18E+08
	% Data	38.9	37.1	21.4	38.8	33.7	36.5	43.5	30.2	34.4	30.2	16.2	31.1	32.7	28.9	5.4
	% Index	57.4	48.6	66.6	56.6	61.4	37.6	48.8	65.1	54.9	39.0	68.3	53.7	54.8	71.1	92.7
	% Temp	3.7	14.3	12.0	4.6	4.9	25.9	7.7	4.7	10.7	30.8	15.6	15.2	12.5	0	1.8
Writes	Total #	484786	3374269	935996	297787	630881	1428314	3135002	5080919	1842585	811884	262996	4021942	1858947	24723791	4737739
	% Data	31.2	19.2	30.8	23.1	24.3	17.4	46.7	58.4	45.9	8.5	17.5	23.6	28.9	61.7	9.7
	% Index	42.8	28.3	24.2	31.1	24.2	6.8	36.7	25.3	20.1	11.0	21.1	10.8	23.5	38.3	48.8
	% Temp	26.0	52.5	44.9	45.8	51.5	75.8	16.5	16.3	34.0	80.5	61.5	65.6	47.6	0	41.5

Table 11: Relative Significance of Data, Index and Temporary Objects.

updates are directed at index and temporary objects. Such behavior is a reflection of the fact that TPC-D is a query processing workload that is predominantly read-only and that has been well-tuned to use indices effectively. It implies that optimizations for handling index and temporary objects are disproportionately important for TPC-D.

4.4 I/O Intensity and Burstiness

A major consideration in designing a computer system is that it should be able to sustain I/O activity that is commensurate with its processing power. When designing the IBM System/360, Amdahl observed that the amount of I/O generated per instruction tends to be relatively constant [3]. More specifically, Amdahls' rule of thumb states that a typical data processing system generates approximately 1Mb/s of I/O bandwidth for every MIPS of processing power [18]. This rule of thumb dates back to the sixties and major changes in both hardware and software have since occurred. Therefore, in this section, we revalidate it by empirically estimating the ratio of I/O activity to processing power required for our workloads.

We use the term *bPI* (bits Per Instruction) to denote the number of bits of I/O generated per instruction. We emphasize the logical bPI which is defined in terms of the logical I/O generated per instruction. This is an intrinsic characteristic of the workload that is relatively independent of system configuration such as memory size. Note, however, that dramatic differences in memory size can result in algorithmic changes that affect the logical bPI. For instance, the amount of memory available for sorting determines whether external sorting techniques are required and if so, the number of merge phases needed [29]. Similarly, as more memory is available, fewer passes are needed to perform hash joins [42] and this translates into less I/O and therefore lower bPI. Conversely, with larger and cheaper memories, previously advantageous tradeoffs of additional computation for less memory use no longer apply.

The physical bPI for a given system configuration can be obtained from the logical bPI by multiplication with the buffer pool miss ratio. However, the physical bPI so obtained reflects only the physical I/O generated by the database. For instance, it does not account for system generated I/O which may constitute a significant portion of the total I/O in certain environments. For example, in [45], it was found that over 80% of the I/O was I/O that was not "visible" to the user, *i.e.,* it was not I/O by a user process to a user defined file. Our trace data reflects only database system I/O and not whatever I/O may have been generated by the operating system or other applications.

Unfortunately, we do not have information regarding the system configurations for our production workloads. We do know, however, that the installations from which our traces were taken tend to have some of the highest-end systems available at the time. So we assume that these systems had about 100 MIPS of processing power, which is roughly half the processing power of the most powerful mainframe systems that IBM began shipping in late 1992. For the TPC benchmarks, the processing power of the systems is determined by the following formula:

$$MIPS = \frac{\textit{\# processors} \cdot \textit{processor clock speed}}{\textit{estimated CPI}}$$

We estimate that the CPI (Cycles Per Instruction) is about 3 for TPC-C and 1.5 for TPC-D, in view of the results presented in [2, 28].

The average amount of logical I/O generated per instruction for the various workloads is summarized in the last row of Table 12. The corresponding numbers for the write I/O activity are shown in Table 13. On average, the production workloads have a logical bPI of about 0.6, approximately one tenth of which is due to writes. TPC-C's bPI is about 3 times higher while TPC-D's bPI is about twice as high. Note, however, that mainframe and x86 MIPS are not equivalent and cannot be directly compared. Our primary interest in this exercise is merely to determine an order-of-magnitude estimate for bPI. We find the figure of 0.6 to be surprisingly high - almost as high as the earlier noted figure of 1.0, based on systems of the 1960s, despite all of the changes suggesting much lower I/O rates.

Results presented in [20] show that a buffer pool that is 1% of the total data size can achieve an average hit ratio of about 90% for the production workloads. With such a hit ratio, the average physical bPI value for the production workloads appears to be around 0.06, which is much lower than Amdahl's rule of thumb. In the 1960s, of course, physical and logical I/O were the same thing. The corresponding hit ratio for the TPC benchmarks is around 95%, meaning that the physical bPI for TPC-C and TPC-D is comparable to the average of the production workloads (0.08 for TPC-C and 0.05 for TPC-D).

The burstiness of the I/O traffic is a very important characteristic of a workload and has implications on the techniques that can be applied to improve I/O performance. For instance, a bursty traffic pattern suggests that buffering mechanisms that smooth out the traffic will be useful. More generally, it indicates that there are opportunities to use the relatively idle periods to do some useful work. One common approach is to defer or offload some work from the busy periods to the relative lulls. Write buffering with subsequent destage and parity-logging disk arrays [47] can be viewed as examples of such an approach. Another frequently used approach is to eagerly or speculatively perform some work in the hope that such work will help improve performance during the next busy period. Examples of such techniques include prefetching, reorganizing data based on access patterns, and garbage collection. A bursty traffic pattern may also be more amenable to techniques that adjust and adapt to

Trace	Aerospace	Bank	ConsGds	DirMktg1	DirMktg2	FinSvcs	Insurance	Retail	TelecomA	TelecomB1	TelecomB2	Utility	Prod. Ave.	TPC-C	TPC-D
100-ms	2.51	3.46	2.23	3.20	3.73	3.28	5.37	3.72	5.27	3.74	3.97	3.66	3.68	6.09	7.54
1-s	1.76	2.85	1.16	2.09	2.45	2.65	3.19	2.60	3.23	1.89	2.16	2.87	2.41	2.66	6.95
10-s	1.40	1.92	0.632	1.69	1.67	1.65	2.12	1.91	2.81	1.09	1.59	2.40	1.74	2.02	5.66
1-min	0.976	0.878	0.464	1.21	1.42	1.14	1.69	1.29	2.14	0.854	1.32	2.18	1.30	1.76	5.59
10-min	0.575	0.586	0.365	0.848	0.945	0.528	1.25	1.05	1.14	0.683	0.989	1.44	0.867	1.70	3.94
100-min	0.333	0.412	0.335	0.566	0.751	0.437	0.837	0.799	0.540	0.478	0.747	1.08	0.609	1.69	2.22
Trace Len.	0.285	0.142	0.328	0.559	0.643	0.365	0.700	0.724	0.715	0.430	0.739	1.047	0.556	1.62	0.991

Table 12: Number of Logical I/O Bits Per Instruction Averaged over Various Time Intervals. The table shows the peak or maximum value observed for each interval size. The row labeled "Trace Len." is essentially the average logical bPI over the entire trace.

Trace	Aerospace	Bank	ConsGds	DirMktg1	DirMktg2	FinSvcs	Insurance	Retail	TelecomA	TelecomB1	TelecomB2	Utility	Prod. Ave.	TPC-C	TPC-D
100-ms	1.14	1.06	0.492	0.963	0.796	2.01	1.25	0.816	1.41	0.800	0.731	2.22	1.14	0.677	1.16
1-s	0.559	0.664	0.193	0.583	0.494	1.15	0.820	0.623	1.05	0.684	0.634	1.83	0.774	0.355	0.767
10-s	0.311	0.383	0.109	0.305	0.307	0.394	0.592	0.270	0.499	0.146	0.143	1.38	0.403	0.261	0.208
1-min	0.0821	0.332	0.0785	0.128	0.157	0.277	0.409	0.196	0.305	0.110	0.0562	0.726	0.238	0.225	0.173
10-min	0.0289	0.189	0.0525	0.0386	0.0468	0.0648	0.299	0.154	0.192	0.0743	0.0248	0.233	0.117	0.214	0.0578
100-min	0.0212	0.051	0.0421	0.0261	0.0358	0.0554	0.154	0.0985	0.0548	0.0444	0.0146	0.137	0.0612	0.213	0.0301
Trace Len.	0.0178	0.0134	0.0430	0.0260	0.0282	0.0333	0.106	0.0951	0.101	0.0303	0.0141	0.112	0.0517	0.205	0.0215

Table 13: Number of Logical I/O Bits Written Per Instruction Averaged over Various Time Intervals. The table shows the peak or maximum value observed for each interval size. The row labeled "Trace Len." is essentially the average number of logical I/O bits written per instruction over the entire trace.

the traffic. For instance, if the write traffic is bursty, setting aside a fixed portion of the buffer pool as the write cache will probably not perform as well as letting the write cache grow and dynamically deciding when and what pages to destage.

In this paper, we briefly consider how the workloads vary in the burstiness of their I/O traffic. Readers who are interested in the detection of idle periods and the prediction of their lengths are referred to [15]. Figure 10 shows the profile of logical bPI over time. Observe that the I/O traffic of the production workloads tends to be rather bursty in nature. Because of time-of-day effects, the fluctuation in bPI is especially pronounced for Bank, which was observed for 23 hours. The I/O traffic for TPC-D is also very bursty. In contrast, TPC-C's I/O traffic stands out as being very regular, suggesting that TPC-C, unlike the production workloads, will not discriminate against systems that do not exploit the idle periods.

For a more quantitative characterization of the burstiness, we time-averaged the logical bPI for the various workloads over intervals ranging from 100 milliseconds to the trace length. The maximum values observed for each of these intervals are presented in Tables 12 and 13. The fact that the bPI drops significantly when averaged over longer time periods indicates that the I/O traffic of the workloads tends to be very bursty in nature. When designing systems, we have to take this burstiness into consideration and design not just for the average case. Notice further that the writes account for a larger fraction of the bPI for smaller interval sizes. This suggests that the write activity is more bursty than the read activity.

Figure 11 plots the distribution of the number of logical I/O bits per instruction averaged over one-second and one-minute periods. As shown in the figure, the data can be modeled reasonably well by the beta distribution (denoted Beta($\alpha 1$, $\alpha 2$) where $\alpha 1$ and $\alpha 2$ are the standard parameters) and the exponential distribution (denoted Exp(μ) where μ is the mean). For instance, the distribution of bPI averaged over one-second periods tends to follow the beta distribution with parameters 1.69 and 38.7 that is scaled by 10.3 and translated by 0.000259. This is denoted as Beta(1.69, 38.7)$\times$10.3+0.000259 in Figure 11.

5 Conclusions

In this paper, we empirically examine the workload characteristics of the peak production database workloads of ten of the world's largest corporations as well as those of the industry-standard benchmarks for on-line transaction processing and decision support systems, namely TPC-C and TPC-D respectively. Even though the production workloads were run on similar systems at around the same point in time, they turned out to be very diverse. Nevertheless, in certain

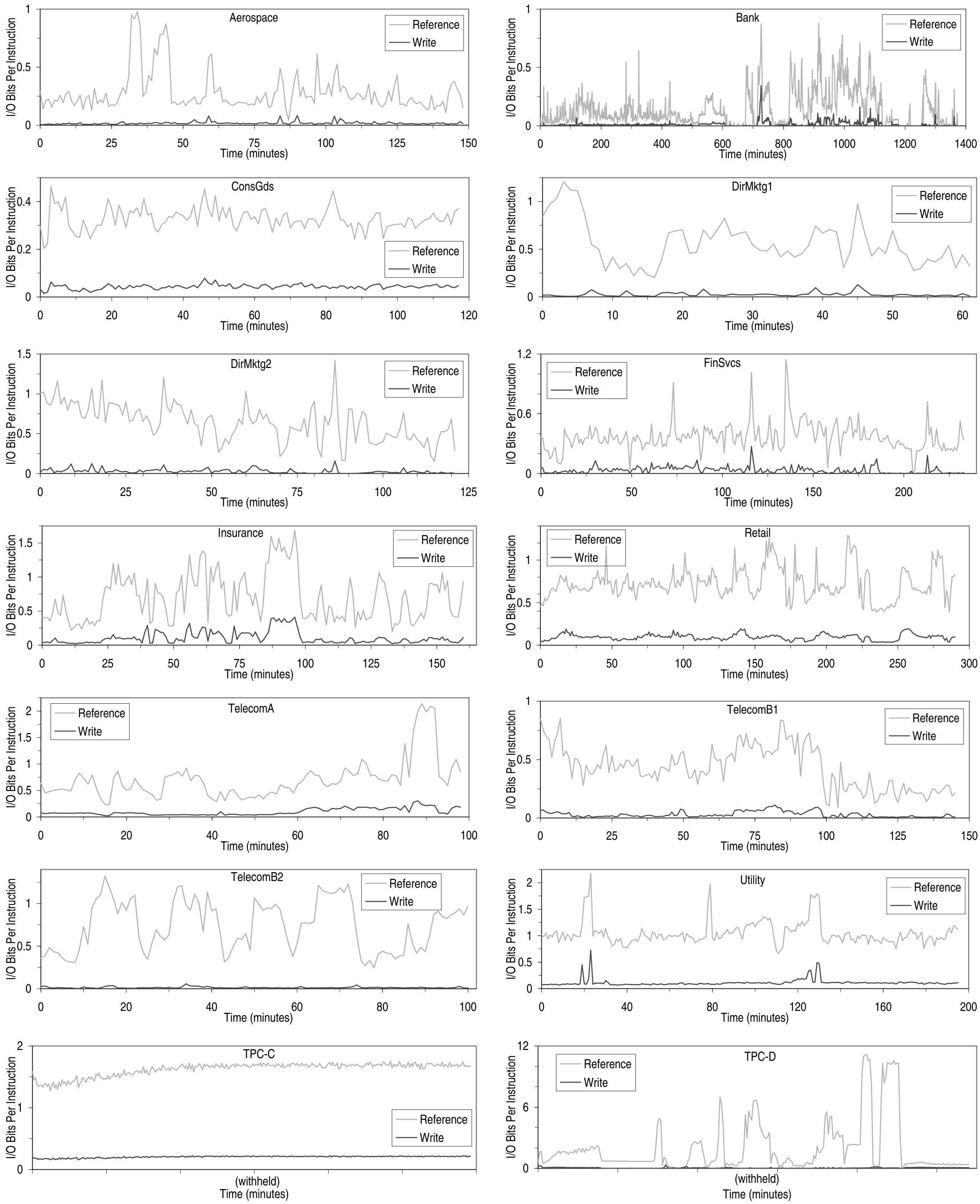
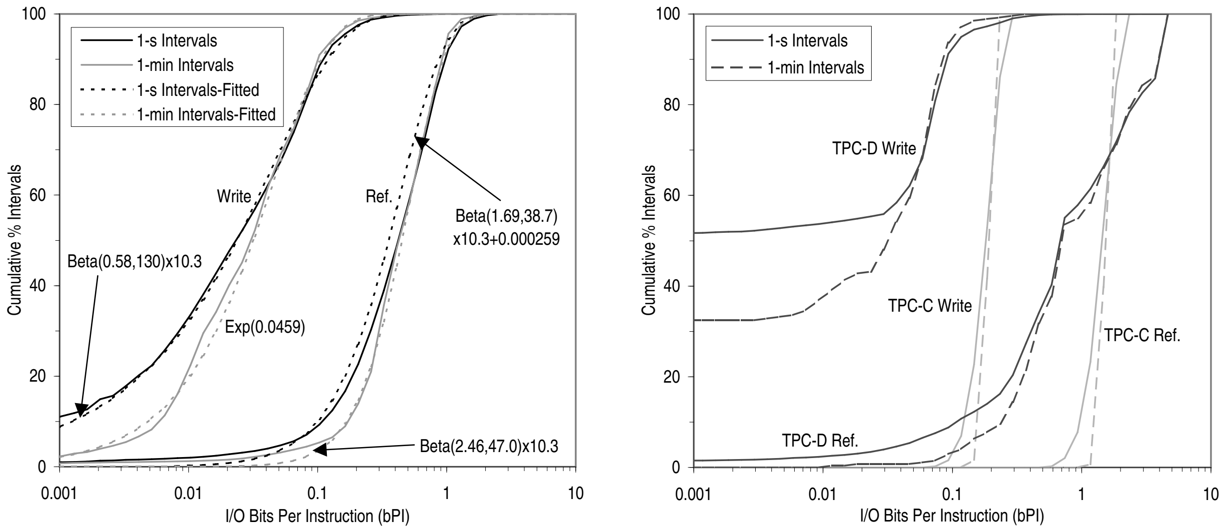


Figure 10: Profile of Number of Logical I/O Bits Per Instruction over Time. The data in this figure have been smoothed by averaging over one-minute intervals.



(a) Average of Production Workloads. (b) TPC Benchmarks.

Figure 11: Distribution of Number of Logical I/O Bits Per Instruction Averaged over One-Second and One-Minute Time Intervals.

cases, TPC-C and TPC-D still fall outside the broad spectrum of behavior exhibited by the production workloads. In general, the two TPC benchmarks tend to complement one another in the sense that they are representative of different aspects of the production workloads. However, there are still some characteristics of the real workloads that are not reflected by either of the benchmarks.

We find that the production workloads are dynamic in that their characteristics are time-varying. For instance, their I/O demands are very bursty, suggesting that adaptive techniques for smoothening the load and for intelligently exploiting idle periods will be useful in a production setting. In stark contrast, TPC-C is very static and predictable, implying that TPC-C primarily evaluates peak performance, which though definitely important, does not translate exactly into effective performance in a production environment with bursty workload characteristics. TPC-D is better in this regard but it shares with TPC-C the characteristic of having a rather stable degree of concurrency. This means that these benchmarks will tend not to measure the overheads for setting up and destroying database agents, which can be significant in a production environment.

Another aspect of the regularity of TPC-C is manifested in the size of its transactions. Unlike the production workloads which contain transactions with a wide variety of sizes, TPC-C's transactions are very uniform in size. In other words, TPC-C will not test techniques for scheduling and allocating resources among transactions with different resource and performance requirements even though these are common in the production environments. TPC-D appears to be similar to the production workloads in that it contains transactions with a wide variety of sizes. However, the very long transactions in TPC-D are due to the read-only queries that are run serially in the power test. Therefore, it too does not evaluate scheduling and resource allocation among diverse transactions.

When two-phase locking is used to ensure that transactions are serializable [13], locks tend to be released only when transactions end. Therefore the distribution of transaction size is a very important factor in determining lock contention. As we have seen, TPC-C's transactions are rather uniform in size. Furthermore, when compared to the production workloads. TPC-C tends to have longer transactions and relatively few read-only transactions. All these suggest that TPC-C stresses the concurrency control mechanism differently than the production workloads analyzed in this paper. Some of TPC-D's transactions are much longer than those of the production workloads but since they are read-only, they can be run at a lower isolation level, *i.e.,* under more relaxed consistency requirements [16]. Furthermore, in the TPC-D power test, the long transactions are run serially. In other words, TPC-D tends not to load the concurrency control mechanism.

While temporary objects account for a significant portion of the write traffic in the production workloads, TPC-C does not have any activity to temporary objects. TPC-D is more in line with the production workloads in this regard but it stands out in that practically all of its references are directed at the index objects. All these mean that TPC-C does not assess the handling of temporary objects whilst TPC-D disproportionately rewards index optimizations. Our analysis also suggests that on an object basis, the production workloads exhibit significantly higher reference and write skew than do the two benchmarks. In other words, statically allocating hotter objects to faster storage will be more beneficial to the production workloads than to the TPC benchmarks.

As part of our analysis, we also reexamine Amdahl's rule of thumb from the sixties, which states that a typical data processing system generates about 1Mb/s of I/O bandwidth for every MIPS of processing power. We discover that both the TPC benchmarks and the production workloads generate logical I/O rates within a factor of two of the earlier figure, despite the passage of 20-30 years. Physical I/O rates, of course, are about 90% lower due to the use of buffering and caching techniques not used in the earlier period.

6 Acknowledgments

Hsu gratefully acknowledges the support of IBM during this research. This work was initiated as a joint project through an IBM Fellowship. Hsu is now supported by IBM Research. Smith's research in this and related areas is also funded by the State of California under the MICRO program, IBM, Cisco Corporation, Fujitsu Microelectronics, Intel Corporation, Microsoft Corporation, Quantum Corporation, Sun Microsystems and Toshiba Corporation.

Collecting traces and getting them into a form suitable for analysis constitute a very long and involved process. This research would not have been possible without the help of many. In particular, the authors would like to thank Eric Louie and Alex Hazlitt for their prompt answers and solutions to the many questions and problems that arose during the setting up of the TPC-C and TPC-D benchmarks. The authors would also like to thank Jyh-Herng Chow, Amit Somani and Surrendra Verma for helping the authors to navigate through the DBMS source code to determine the appropriate instrumentation points. In addition, the authors are much indebted to the many people who contributed to getting the production traces. Special mention is made of Ted Messinger who, even in retirement, provided the authors with help on processing and interpreting the production traces. Barbara Tockey and James Teng also provided invaluable help in this regard. The authors would also like to express their gratitude to Chuck Tribolet who offered to take on the arduous task of getting the raw production traces off more than 600 tape cartridges and onto the authors' computer system. Finally, the authors would like to recognize Joseph M. Hellerstein for his helpful comments on early versions of this paper.

References

[1] A. V. Aho, P. J. Denning, and J. D. Ullman, "Principles of optimal page replacement," *Journal of the ACM*, vol. 18, no. 1, pp. 80–93, Jan. 1971.

[2] A. Ailamaki, D. J. DeWitt, M. D. Hill, and D. A. Wood, "DBMSs on a modern processor: Where does time go," in *Proceedings of International Conference on Very Large Data Bases (VLDB)*, (Edinburgh, Scotland), Sept. 1999.

[3] G. M. Amdahl, "Storage and I/O parameters and systems potential," in *Proceedings of IEEE International Computer Group Conference (Memories, Terminals, and Peripherals)*, (Washington, DC), pp. 371–372, June 1970.

[4] Anonymous, "Errata," *ACM Computing Surveys*, vol. 29, no. 4, pp. 428–428, Dec. 1997. See [54].

[5] M. G. Baker, J. H. Hartman, M. D. Kupfer, K. W. Shirriff, and J. K. Ousterhout, "Measurements of a distributed file system," in *Proceedings of ACM Symposium on Operating Systems Principles*, (Pacific Grove, CA), pp. 198–212, Oct. 1991.

[6] S. J. Baylor and C. E. Wu, "Parallel I/O workload characteristics using Vesta," in *Input/Output in Parallel and Distributed Computer Systems* (R. Jain, J. Werth, and J. C. Browne, eds.), vol. 362 of *The Kluwer International Series in Engineering and Computer Science*, ch. 7, pp. 167–185, Kluwer Academic Publishers, 1996.

[7] I. R. Casas and K. C. Sevcik, "A buffer management model for use in predicting overall database system performance," in *Proceedings of IEEE International Conference on Data Engineering.*, (Los Angeles, CA), pp. 463–469, Feb. 1989.

[8] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson, "RAID: high-performance, reliable secondary storage," *ACM Computing Surveys*, vol. 26, no. 2, pp. 145–185, June 1994.

[9] P. E. Crandall, R. A. Aydt, A. A. Chien, and D. A. Reed, "Input/output characteristics of scalable parallel applications," in *Proceedings of Supercomputing '95*, (San Diego, CA), Dec. 1995.

[10] J. Douceur and W. Bolosky, "A large-scale study of file-system contents," in *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computing Systems*, (Atlanta, GA), pp. 59–70, May 1999.

[11] W. Effelsberg and T. Haerder, "Principles of database buffer management," *ACM Transactions on Database Systems*, vol. 9, no. 4, pp. 560–595, Dec. 1984.

[12] W. Effelsberg and M. E. S. Loomis, "Logical, internal, and physical reference behavior in CODASYL database systems," *ACM Transactions on Database Systems*, vol. 9, no. 2, pp. 187–213, June 1984.

[13] K. P. Eswaran, J. N. Gray, R. A. Lorie, and I. L. Traiger, "The notions of consistency and predicate locks in a database system," *Communications of the ACM*, vol. 19, no. 11, pp. 624–633, Nov. 1976. Also available as Research Report RJ 1487, IBM Research Laboratory, San Jose, CA, Dec. 1974.

[14] R. A. Floyd and C. S. Ellis, "Directory reference patterns in hierarchical file systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 1, no. 2, pp. 238–247, June 1989.

[15] R. Golding, P. Bosch, C. Staelin, T. Sullivan, and J. Wilkes, "Idleness is not sloth," in *Proceedings of USENIX Technical Conference*, (New Orleans, LA), pp. 201–212, USENIX, Jan. 1995.

[16] J. N. Gray, R. A. Lorie, G. F. Putzolu, and I. L. Traiger, "Granularity of locks and degrees of consistency in a shared data base," in *Modeling in Data Base Management Systems, IFIP TC-2*, pp. 365–394, 1976. Also available as Research Report RJ 1606, IBM Research Laboratory, San Jose, CA, June 1975.

[17] P. Hawthorn and M. Stonebraker, "Performance analysis of a relational data base management system," in *Proceedings of*

ACM SIGMOD International Conference on Management of Data, (Boston, Massachusetts), pp. 1–12, May 1979.

[18] J. L. Hennessy and D. A. Patterson, *Computer Architecture A Quantitative Approach*. Morgan Kaufmann Publishers, Inc. San Francisco, CA, second ed., 1996.

[19] A. V. Hill, "The combinations of hæmoglobin with oxygen and carbon monoxide," *Biochemistry Journal*, vol. 7, pp. 471–480, 1913.

[20] W. W. Hsu, A. J. Smith, and H. C. Young, "I/O reference behavior of production database workloads and the TPC benchmarks - an analysis at the logical level." Technical Report CSD-99-1071, Computer Science Division, University of California, Berkeley, Nov. 1999. Also available as Research Report RJ 10166, IBM Almaden Research Center, San Jose, CA, Nov. 1999.

[21] W. W. Hsu, A. J. Smith, and H. C. Young, "Projecting the performance of decision support workloads on systems with smart storage (SmartSTOR)." Research Report RJ 10145, IBM Almaden Research Center, San Jose, CA, July 1999. Also available as Technical Report CSD-99-1057, Computer Science Division, University of California, Berkeley, CA, Aug. 1999.

[22] W. W. Hsu, A. J. Smith, and H. C. Young, "Results and data for 'Analysis of the I/O characteristics of production database workloads and the TPC benchmarks'," 1999. http://www.cs.berkeley.edu/~windsorh/DBChar.

[23] IBM Corporation, *DB2 for OS/390 V5 Installation Guide*. 1997.

[24] IBM Corporation, *DB2 PM for OS/390 V5 Batch User's Guide*. 1997.

[25] IBM Corporation, *DB2 UDB V5 Administration Guide*. 1997.

[26] J. P. Kearns and S. DeFazio, "Locality of reference in hierarchical database systems," *IEEE Transactions on Software Engineering (SE)*, vol. 19, no. 2, Mar. 1983.

[27] J. P. Kearns and S. DeFazio, "Diversity in database reference behavior," *Performance Evaluation Review*, vol. 17, no. 1, pp. 11–19, May 1989.

[28] K. Keeton, D. Patterson, Y. He, R. Raphael, and W. Baker, "Performance characterization of a quad Pentium Pro SMP using OLTP workloads," in *Proceedings of ACM SIGARCH International Symposium on Computer Architecture (ISCA)*, (Barcelona, Spain), pp. 15–26, June 1998.

[29] D. E. Knuth, *The Art of Computer Programming: Sorting and Searching*, vol. 3. Addison-Wesley Publishing Company, second ed., 1998.

[30] D. E. Knuth, "An empirical study of FORTRAN programs," *Software - Practice and Experience*, vol. 1, no. 2, pp. 105–133, Apr./June 1971.

[31] S. T. Leutenegger and D. M. Dias, "A modeling study of the TPC-C benchmark," in *Proceedings of ACM SIGMOD International Conference on Management of Data*, (Washington, D.C.), pp. 22–31, May 1993.

[32] M. K. McKusick, W. N. Joy, S. J. Leffler, and R. S. Fabry, "A fast file system for UNIX," *ACM Transactions on Computer Systems*, vol. 2, no. 3, pp. 181–197, Aug. 1984.

[33] E. L. Miller and R. H. Katz, "Input/output behavior of supercomputing applications," in *Proceedings Supercomputing '91*, (Albuquerque, NM), pp. 567–576, Nov. 1991.

[34] N. Nieuwejaar, D. Kotz, A. Purakayastha, C. Schlatter Ellis, and M. Best, "File-access characteristics of parallel scientific workloads," *IEEE Transactions on Parallel and Distributed Systems*, vol. 7, no. 10, pp. 1075–1089, Oct. 1996.

[35] J. Ousterhout, H. Da Costa, D. Harrison, J. Kunze, M. Kupfer, and J. Thompson, "A trace-driven analysis of the UNIX 4.2 BSD file system," in *Proceedings ACM Symposium on Operating System Principles*, (Orcas Island, WA), pp. 15–24, Dec. 1985.

[36] B. K. Pasquale and G. C. Polyzos, "A case study of scientific application I/O behavior," in *Proceedings of the Second International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, (Durham, NC), pp. 101–106, Jan. 1994.

[37] B. K. Pasquale and G. C. Polyzos, "Dynamic I/O characterization of I/O intensive scientific applications," in *Proceedings of Supercomputing '94*, (Washington, DC), pp. 660–669, Nov. 1994.

[38] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1990.

[39] A. Purakayastha, C. Schlatter Ellis, D. Kotz, N. Nieuwejaar, and M. Best, "Characterizing parallel file-access patterns on a large-scale multiprocessor," in *Proceedings of IEEE International Parallel Processing Symposium*, (Santa Barbara, CA), pp. 165–172, Apr. 1995.

[40] K. K. Ramakrishnan, P. Biswas, and R. Karedla, "Analysis of file I/O traces in commercial computing environments," in *Proceedings of ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, (Newport, RI), pp. 78–90, June 1992.

[41] J. Rodriguez-Rosell, "Empirical data reference behavior in data base systems," *Computer Magazine of the Computer Group News of the IEEE Computer Group Society*, vol. 9, no. 11, Nov. 1976.

[42] L. D. Shapiro, "Join processing in database systems with large main memories," *ACM Transactions on Database Systems*, vol. 11, no. 3, pp. 239–264, Sept. 1986.

[43] V. Singhal and A. J. Smith, "Analysis of locking behavior in three real database systems," *The VLDB Journal*, vol. 6, no. 1, pp. 40–52, Jan. 1997. Extended version available as Technical Report CSD-94-801, Computer Science Division, University of California, Berkeley, CA, Apr. 1994.

[44] A. J. Smith, "Sequentiality and prefetching in database systems," *ACM Transactions on Database Systems*, vol. 3, no. 3, pp. 223–247, Sept. 1978. Also available as Research Report RJ 1743, IBM Research Laboratory, San Jose, CA, Mar. 1976.

[45] A. J. Smith, "Disk cache — miss ratio analysis and design considerations," *ACM Transactions on Computer Systems*, vol. 3, no. 3, pp. 161–203, Aug. 1985.

[46] A. J. Smith, "Trace driven simulation in research on computer architecture and operating systems," in *Proceedings of the Conference on New Directions in Simulation for Manufacturing and Communications*, (Tokyo, Japan), pp. 43–49, Aug. 1994.

[47] D. Stodolsky, M. Holland, W. V. Courtright II, and G. A. Gibson, "Parity-logging disk arrays," *ACM Transactions on Computer Systems*, vol. 12, no. 3, pp. 206–235, Aug. 1994.

[48] J. Z. Teng and R. A. Gumaer, "Managing IBM Database 2 buffers to maximize performance," *IBM Systems Journal*, vol. 23, no. 2, pp. 211–218, 1984.

[49] A. Thomasian, "Performance limits of two-phase locking," in *International Conference on Data Engineering*, (Los Alamitos, CA), pp. 426–435, Apr. 1991.

[50] J. G. Thompson, *Efficient Analysis of Caching Systems*. PhD thesis, University of California, Berkeley, 1987. Available as Technical Report CSD 87/374, Computer Science Division, University of California, Berkeley, CA, Oct. 1987.

[51] Transaction Processing Performance Council, *TPC Benchmark^{TM} C Standard Specification Revision 3.3.2*. June 1997.

[52] Transaction Processing Performance Council, *TPC Benchmark^{TM} D Standard Specification Revision 1.3.1*. Dec. 1997.

[53] T.-F. Tsuei, A. N. Packer, and K.-T. Ko, "Database buffer size investigation for OLTP workloads," in *Proceedings of ACM SIGMOD International Conference on Management of Data*, (Tucson, AZ), pp. 112–122, May 1997.

[54] R. A. Uhlig and T. N. Mudge, "Trace-driven memory simulation: A survey," *ACM Computing Surveys*, vol. 29, no. 2, pp. 128–170, June 1997. See erratum [4].

[55] A. I. Verkamo, "Empirical results on locality in database referencing," in *Proceedings of ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, (Austin, TX), pp. 49–58, 1985.

[56] B. B. Welch, "Measured performance of caching in the Sprite network file system," *Computing Systems*, vol. 4, no. 3, pp. 315–342, Summer 1991.

[57] S. Zhou, H. Da Costa, and A. J. Smith, "A file system tracing package for Berkeley UNIX," in *Proceedings of the 10th* USENIX *Conference*, (Portland, OR), pp. 407–419, June 1985.

[58] B. T. Zivkov and A. J. Smith, "Disk cache design and performance as evaluated in large timesharing and database systems," in *Proceedings of the CMG (Computer Measurement Group) Conference*, (Orlando, FL), pp. 639–658, Dec. 1997. Abridged version published as "Disk Caching in Large Database and Timeshared Systems", Proceedings of the Fifth International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, (Haifa, Israel), pp. 184-195, Jan. 1997. Extended version available as Technical Report CSD-96-913, Computer Science Division, University of California, Berkeley, CA, Sep. 1996.