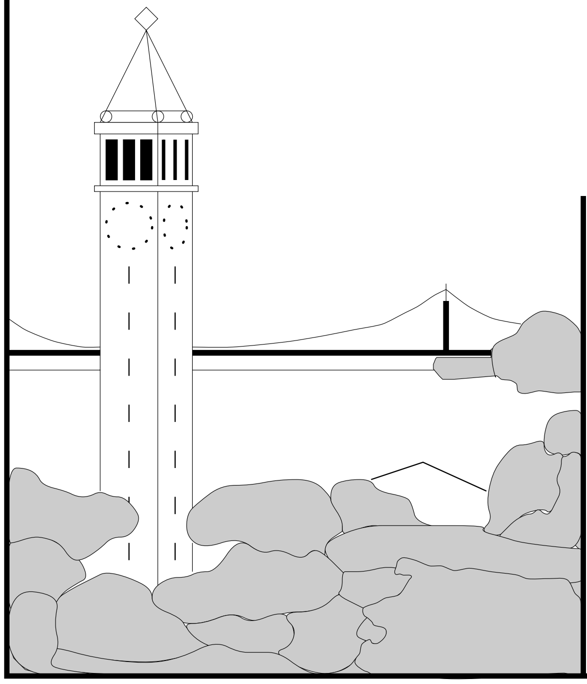
Characterization of Contention in Real Relational Databases

Vigyan Singhal *Alan Jay Smith*



Report No. UCB/CSD 94/801

March 21, 1994

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

Characterization of Contention in Real Relational Databases*

Vigyan Singhal Alan Jay Smith†

March 21, 1994

Abstract

Concurrency control is essential to the correct functioning of a database due to the need for correct, reproducible results. For this reason, and because concurrency control is a well formulated problem, there has developed an enormous body of literature studying the performance of concurrency control algorithms. Most of this literature uses either analytic modeling or random number driven simulation, and explicitly or implicitly makes certain assumptions about the behavior of transactions and the patterns by which they set and unset locks. Because of the difficulty of collecting suitable measurements, there have been only a few studies which use trace driven simulation, and still less study directed toward the characterizion of concurrency control behavior of real workloads.

In this report, we present a study of three database workloads, all taken from IBM DB2 relational database systems running commercial applications in a production environment. This study considers topics such as frequency of locking and unlocking, deadlock and blocking, duration of locks, types of locks, correlations between applications of lock types, two-phase vs. non-two-phase locking, when locks are held and released, etc. In each case, we evaluate the behavior of the workload relative to the assumptions commonly made in the research literature, and discuss the extent to which those assumptions may or may not lead to erroneous conclusions. We also present a simple mathematical model which predicts the frequency of blocking to be expected in these workloads, and compare those predictions to the observed frequency.

*This document is available as Technical Report No. UCB/CSD 94/801 from Computer Science Division (EECS), 571 Evans Hall, University of California, Berkeley, CA 94720.

†The authors' research is supported or has been supported in part by the National Science Foundation under grants MIP-9116578 and CCR-9117028, by NASA under Grant NCC 2-550, by the State of California under the MICRO program, and by the International Business Machines Corporation, Digital Equipment Corporation, Philips Laboratories/Signetics, Apple Computer Corporation, Mitsubishi Corporation, Sun Microsystems and Intel Corporation.

1 Introduction

Concurrency control is essential to the correct functioning of a database due to the need for correct, reproducible results. For this reason, and because concurrency control is a well formulated problem, there has developed an enormous body of literature studying the performance of concurrency control algorithms. Most of this literature uses either analytic modeling or random number driven simulation, and explicitly or implicitly makes certain assumptions about the behavior of transactions and the pattern by which they set and unset locks. Because of the difficulty of collecting suitable measurements, there have been only a few studies which use trace driven simulation, and we are aware of no studies which have been principally directed at characterizing the concurrency control behavior of real workloads.

There have been a few studies of database systems using traces. Some have addressed the issue of database buffer management e.g. [30, 27, 20] studied IMS, [34] studied a CODASYL system, and [17] has analyzed reference behavior in INGRES, a relational database system. These studies, however, have not looked at the issue of concurrency control. Studying concurrency control requires locking activity characteristics (both lock and unlock events), and information about transaction boundaries.

There has also been one group of researchers [38, 39, 37, 40] who have used locking traces from database systems for concurrency control studies. However, the thrust of their research has not been to do an extensive data characterization, and they have provided limited characterization of the transaction workloads they have used. Our main objective in this report is to provide such a characterization.

A large number of researchers have studied concurrency control algorithms from a performance point of view. The usual approach is to define a transaction model consisting of four sub-models: database model, transaction model, user model and system model. Given the model description, there are two popular methods of analysis— through analytical means or by stochastic simulations based on artificially constructed workload parameters (as opposed to trace-driven simulations). It is well known that analytic modeling is limited in the range of behavior assumptions that can employed if solutions are to be expected. Because of the absence of an agreed-upon model of transaction locking behavior, or even the availability of a variety of data and measurements, the simulation studies also make many assumptions. We hope to rectify this problem through this report.

Section 2 will briefly discuss some of the concepts and terminology we will need in this report. In Section 3, we discuss our traces, and how they were collected and analyzed. Section 4 describes the generic concurrency control model and describes the various assumptions which are present in most studies. A detailed look at the database related assumptions is provided in section 5, where we also consider their validity and the sensitivity of system performance to these assumptions. Section 6 does the same for transaction behavior assumptions. In Section 7, we present a model for estimating the inherent contention in the workloads. In Section 8, we will use the model that we develop in Section 7 to estimate the effect of correlation between the various lock types on the contention in the database system.

2 Terminology and Background

In this section, and in Appendix A, we provide an overview of the terminology and methodology used in concurrency control research, and introduce a few definitions which will be useful in our out analyses.

We discuss the concept of transactions (as defined by [16]) in Appendix A. In there, we also describe the various alternatives one has in implementing concurrency control in databases— locking (with blocking, and with restarts), optimistic concurrency control (OCC) and timestamping. In this report we will use the terms *Lock* and *Unlock* because the traced systems use locking to provide concurrency control. Our workloads, however, are equally applicable to the other approaches— OCC and timestamping. For these options the first access to a data item can be interpreted as a Lock request and the last access can be interpreted as an Unlock request.

2.1 Two phase locking vs non-two phase locking

If we choose locking to implement concurrency control, an important issue is *two-phase locking* (2PL). 2PL means that transactions do not acquire new locks after they have released one or more locks. Typically, only before a transaction is about to end does it know that it does not need any more locks. 2PL is typically ensured by releasing all locks only at the end of transactions.

None of the studies we are aware of have considered workloads where transactions release locks before the the end of each transaction. In real database systems, however, locks actually are released before end of transaction to provide better performance. In our traces, some of the transactions (especially the long ones) release Read locks soon after they are acquired and long before the transaction ends. Other transactions exhibit two-phase behavior— no locks are released before the transaction commits. In all cases, however, Write locks are never released before the end of transaction. This may be for two reasons— greater semantic consistency (otherwise transactions may see changes in data if they read the same data after they have written it) and also to avoid cascading aborts [15]. The locking behavior we have just described corresponds to isolation level 3[1] as described in [10]. The disadvantage of this

[1]Isolation levels have been defined in [10] as the degrees of interference transactions can tolerate; higher levels provide higher isolation but lower concurrency. Level 1 means that transactions cannot update any uncommitted changes by other transactions. Level 2 prevents transactions from reading any uncommitted changes. Level 3 does not allow a transaction to change a record on which another transaction holds a Read lock. Level 4 does not allow a transaction to change a record which another transaction has seen.

level is that it allows non-serializable behavior of transactions. As [13] has shown, two-phase locking (which corresponds to isolation level 4 of [10])) is a necessary and sufficient condition for serializability if transactions are allowed to access data in any arbitrary order. However, in most applications, the access order to the data has some restrictions and releasing some locks before end of transaction may also be sufficient for serializability. An example of this is access to B-tree locks where locks can be released and serializability can be maintained at the same time [2].

2.2 Cursor locks

We use the term *Cursor locks* to denote the Read locks which are unlocked before the end of transactions. We call them Cursor locks because usually the purpose of these locks is to guarantee the stability of the cursor pointer on the relation while data is being accessed from the relation [24]. These locks are held only while the object is being accessed. The non-Cursor locks are simply referred to as *Read* or *Write* locks. Although Cursor locks are typically held for a much shorter time than Read locks, they can still have a significant effect on performance, since the transaction acquiring the Cursor lock may have to wait a long time for that lock, even though it then holds it for only a short period.

2.3 Index locks

Index locks (used to provide fast access to data) also have to be locked to maintain transaction isolation; relational database systems often use B-trees to implement indexes. Most concurrency control performance studies have ignored the contention for index locks, although some recent studies have considered this issue specifically [19, 31]. We do not know of any study which has considered the performance of both index and data locking simultaneously. Because of the high use of index locks, they have the potential to cause significant performance problems. The fact that the systems we measured use subpages as the granularity of index locks (as opposed to pages for data) suggests that index contention may have been a problem in the past with some systems, although this has not been documented. In this report, locks on index subpages and data pages are simply referred to as index locks and data locks, respectively.

2.4 Locktimes and lockfractions

We define the term *locktime* of a lock as the length of time that the lock is held. All measured times in this report are real (not virtual) times on the traced database systems. The locktime of a transaction is defined as the sum of the locktimes for all the locks it acquires. Locktime is an important quantity— it indicates how long a transaction may have to wait to acquire a lock which conflicts with a lock already held by another transaction. Note that the ratio of the locktime of a transaction and the length of the transaction gives the average number of locks held by the transaction during its lifetime.

Unlike Cursor locks, all Read and Write locks are released only at the the end of the transaction. Locktimes for Read and Write locks should therefore be closely related to transaction length, unless most locks are acquired at the end of transactions. We define *lockfraction*, as the ratio of locktime to transaction length. For Read and Write locks,the lockfraction also indicates the fraction of the remaining lifetime when the lock is acquired.

3 Description of Traces

In this section, we describe our traces and the systems from which they were collected.

The data analyzed in this report has been collected from three different commercial installations: Security Pacific Bank (referred to as *bank*), Crowley Maritime Corp. (*transport*) and an anonymous telecommunications company (*phone*). The description of the relevant hardware and software configurations of the three sites appears in Table 1.

Two of the traces, *bank* and *transport* have been traced using the IBM DB2 GTF tracing facility [18]. The process of gathering these traces and a detailed description of the traced events appears in [35]. The *phone* trace has been gathered using a new and much more efficient tracing package [24], discussed further below. The DB2 tracing facility is able to activate tracing for any subset of fourteen different *event-classes*, for example *SQL events*, *I/O events*, and *Lock events*. Each event class causes trace records to be logged for a number of different system events.

The traces studied are trace segments containing contiguous trace information, without any breaks in the tracing period, collected from multiprocessor shared-memory (centralized) systems. The database management system at all three sites was DB2, IBM's relational database management system for IBM 3090 mainframes [12]. While these three traces may not completely represent the entire spectrum of transaction processing applications, our report still provides a good characterization of locking behavior in three very different transaction processing environments. To our knowledge, no other work in the published literature covers such a varied workload.

Locking entries

For our analysis, we used the *Lock* and *Unlock* entries from the traces. Data pages are locked in one page (4K Bytes) units. Indexes may be locked on a subpage partition; each index page is a B-tree node. The number of partitions is user-defined and variable per trace. We observed up to 16 partitions per page for some indexes in our traces.

The DB2 traces contain 3 kinds of locks— *Read*, *Write* and *Update*. Write locks conflict with each other and with Read locks. Update locks guarantee the same semantic consistency

Site	Phone	Bank	Transport
Company	(anonymous)	Security Pacific	Crowley Maritime
Trace Date	10/15/90	3/16/88	7/25/88
Hardware			
CPU	IBM 3090-600J (12)	IBM 3090-200 (2)	Hitachi/NAS XL80
Disks	IBM 3380/90 (35)	IBM 3380 (15)	IBM 3380 (20)
Software			
OS	MVS-XA 3.1	MVS-XA 2.1.7	MVS-XA 2.1
DB2 Release	Release 1 Version 2.1	Release 3 Version 1.3	Release 3 Version 1.3
Trace			
Size	216.91MB	307.69MB	567.58MB
Time	30min	1h 15min	3hr
Entries	8112768	830478	1834422
TPS	4.955	0.341	0.544

Table 1: Description of the trace sites and the traces. Entries refers to the total number of trace entries per traces (buffer manager and locking entries). TPS refers to transactions/second.

as Read locks; however, only Update locks are allowed to be upgraded to Write locks. Update locks do not conflict with Read locks, but conflict with each other and with Write locks. The reason for having Update locks, instead of directly upgrading Read locks, is to avoid possible future deadlocks at the cost of higher contention [10]. The performance tradeoffs of Read locks vs Upgrade locks are not obvious and, to our knowledge, such tradeoffs have never been studied in the literature. Since the functionality provided by Update locks is identical to Read locks, for this report we have mapped all Update locks to Read locks.

Transactions

Individual transactions in the trace are identified with a field called the *ACE address*, which identifies the address space of the process. There are no *Begin Transaction* or *End Transaction* entries in the traces. We have, therefore, chosen the time the first lock request is issued by a transaction as the beginning of the transaction lifetime. There is, however, a special unlock entry which says *Unlock all locks* owned by this ACE address. We use this to decide when a transaction ends. Our definition of transaction boundaries is smaller than the actual boundaries— we neglect the work done before the first lock request and after the last unlock request. However, we include all the work done while any locks were held. Since we are interested in concurrency control contention, this should be sufficient. Note that the actual system load and the transaction lengths will be greater than what is reported here. We refer to the number of active transactions at any moment as the multiprogramming level (*MPL*) of the system.

In this report, we measure various characteristics of the transactions. Using the method described above we have associated a transaction identifier with each trace entry. Then we unraveled the trace to separate the entries belonging to one transaction from the entries belonging to other interleaved transactions. Most measurements in this report have been done from this pool of transactions. This pool is what would be required, along with the ordering of transactions, if we were to use our traces for a trace-driven concurrency control model simulation.

Note that a transaction sometimes represents a smaller unit of work than a job, since a user job may spawn a number of transactions. A transaction is a complete unit of work, however, in that it releases all locks it owns when it finishes.

Timestamps

In two of the traces, *bank* and *transport*, all trace entries are tagged with a timestamp. In the *phone* trace, the timestamps have been synthesized by linearly interpolating time epochs from some entries which record timestamps. There is a special trace entry periodically records the time (every 32KB, or 1142 trace entries), and some of the trace entries (like *Commit* and a few others which we do not use in our analyses) do have timestamps.

Since our objective is to characterize the transaction workloads in real databases, we have removed lock-wait intervals from the traces for most of our analyses. This is done for a particular transaction by advancing the timestamps of the transaction entries that follow the lock-wait by the duration of the lock-wait interval. Only in Section 7, where we compare projected contention in the workload to the real measured contention, do we use the lock-wait intervals. The motivation to exclude existing lock-wait intervals is clear— we want to characterize the transaction workload that is loaded on the system, and not how the traced DB2 system processed that workload. Lock-waits represent one instantiation (one particular interleaving of transactions) of the workload; they are not part of the inherent workload. In fact, lock-waits represent a *performance index*, an output, for concurrency control studies.

We obtained *bank* and *transport* traces using the GTF tracing package, which imposes a substantial system overhead (of the order of 100%–200% is reported in [35]). GTF logs an enormous amount of information per trace entry, much of which (about 88% by volume) is not used for our analysis. This overhead may affect us in two ways. It may affect the validity of the durations of the transactions. However, in this report we are not interested in the absolute time values. All time values must be considered relative to each other. We make the reasonable assumption that all transactions are affected by the overhead equally. Another side-effect of the overhead is that the users' behavior may change due to this overhead. Although this may reflect on the system MPL, we have no reason to believe that it affects the mix of the transactions in the workload substantially. The *phone* trace has been logged using a very efficient new tracing package which incurs an overhead of about 2%–5% [24].

In fact we have been told that the users of the database system did not perceive any change in the system performance. Clearly this eliminates the possibility of the second side-effect we just discussed. Note that we also assume that all transactions are stretched equally due to multiprogramming effects.

System activity in the traced systems

In this subsection, we present the level of system activity in the database systems we traced. Figure 1 shows the transaction multiprogramming level and the average number of transactions waiting on lock requests. The figure represents a average number of transactions calculated using a continuous window of time (2min, 1min and 30sec for *transport*, *bank* and *phone*, respectively). The window size has been chosen for each trace to provide reasonable smoothing.

Because of the way we have defined transaction lifetimes earlier in this section, the figure represents a lower level of activity than the actual system— processes which do not lock any items are not counted; also portions of transaction lifetimes before the acquisition of first lock and after the release of all locks, are not counted.

The figure shows a relatively low multiprogramming level for the *transport* and *bank* traces. One of the reasons for this might be that the heavy overload of GTF tracing caused the transaction load to be less than the usual load, due both to the scheduler decreasing the MPL, and to users withdrawing from the system due to unresponsiveness. These two traces also show a near-zero level of lock contention. On the other hand, the *phone* plot shows a significant degree of contention. One interesting observation from the *phone* plot is the fact that the contention level mimics the MPL level so closely. This suggests that the system is trying to maintain the number of active transactions (those not blocked for locks) at a constant. This would suggest a load control policy which is based solely on the number of active transactions, and ignores the number of transactions blocked for locks. Such a policy increases the MPL under high contention levels and is likely to make things worse. See [26, 7] for a discussion of load control policies.

4 Concurrency control modeling

In this section we describe the generic concurrency control model used in most concurrency control analyses, and note also the various assumptions made in such modeling. Most assumptions have two flavors— behavioral assumptions (e.g. selection of exponential distributions) and assumptions about parameter settings (e.g. the value of the mean for that exponential distribution).

There are four main components of a concurrency control model: the database model, the user model, the transaction model and the system model.

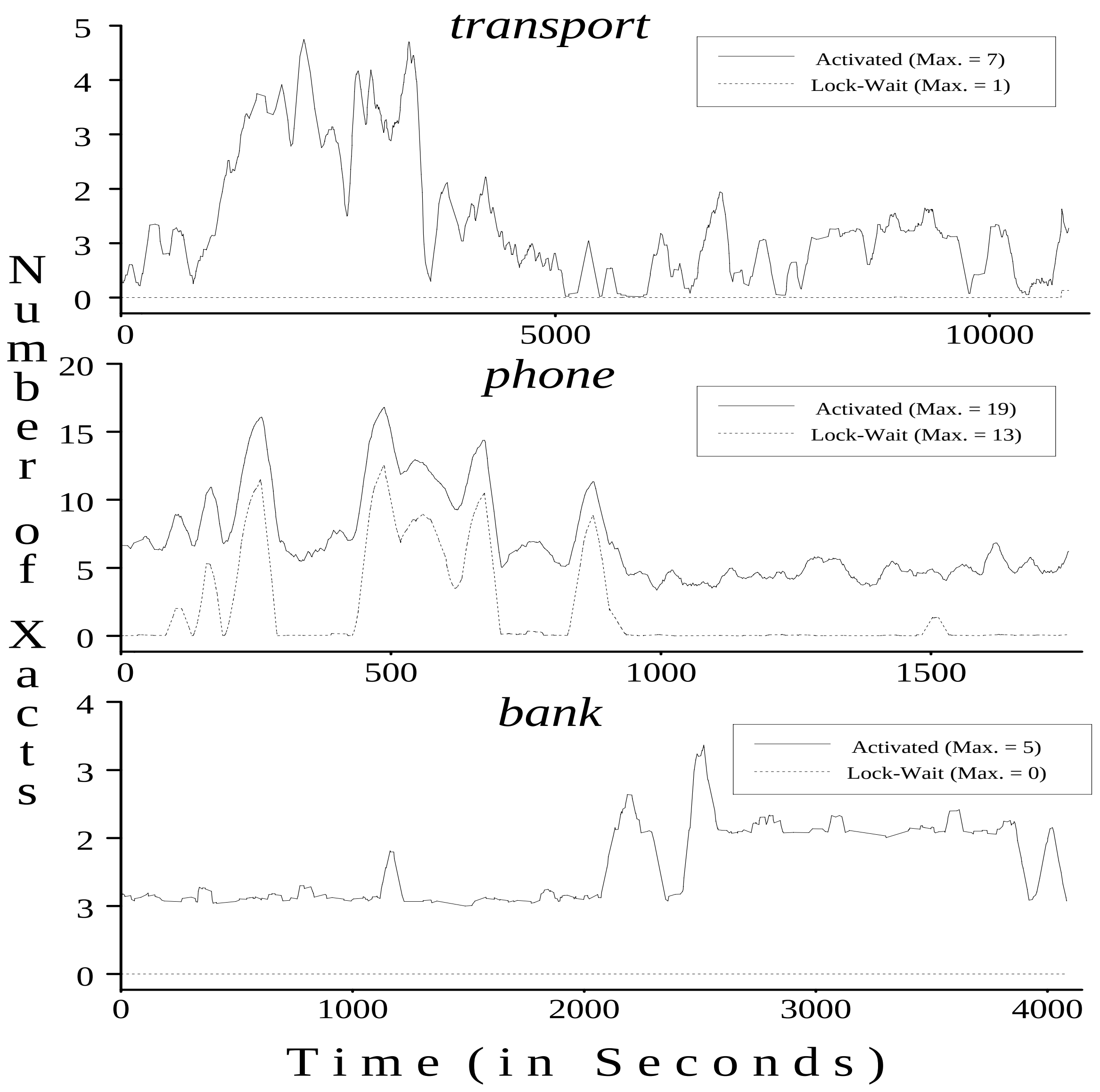


Figure 1: Transaction load for the traces. The number of transactions denotes an average over a time-period. The time-period is 120 sec, 15 sec and 30 sec, respectively, for the three traces. *Lock-wait* denotes the average number of transactions waiting because of lock conflicts.

Database model

The database model captures the characteristics of the database, like the database size, data distribution on multiple nodes, data replication, and the access pattern. A relational database, for example, is composed of several relations or tables. Each of the relations might be stored in one or more files. A relation is comprised of a set of tuples, such that each of these tuples is defined over the same attributes.

In concurrency control modeling studies, typically, a database is modeled as a collection of fixed size *data items*. Data items are grouped into *granules* to form units of access and concurrency control. With the exception of studies analyzing effects of granularity on concurrency control performance [9, 28], one granule usually comprises one data item, and is the unit of I/O access or concurrency control. In most commercial systems, granules are defined on a physical scale and not on a logical scale; locks are usually per page and are not related to the record sizes or the table sizes. In the context of this report, one lockable item or object can be interpreted to be one granule.

Clearly the number of the data items is an important parameter because it directly affects the amount of contention. The higher the number of granules, other parameters remaining identical, the lower is the contention. However, as we will see later, the number of lockable items in the database has been frequently underestimated in order to obtain non-negligible (and "interesting") levels of contention in the system.

The access pattern of the data items has often been modeled as being uniform over the entire database. To model non-uniform access some models have used a 80–20 or a 50–5 [11] access behavior. Note that a b–c access behavior means that b% of the accesses go to c% of the data items.

Although concurrency control performance for indexes has been studied in isolation [31], we don't know of a study that has analyzed the effects of index and data locking in the same framework. Since index pages have a B-tree structure, as opposed to the flat structure of the data items, it is reasonable to expect different conflict patterns for data and index locks.

Many, particularly early, studies, have assumed that all locks were Write locks to simplify the analyses. However, modeling both Read and Write locks requires only the Write locks to conflict with other Write locks and Read locks. Also, Read locks requests may have to wait while another Read lock request is active if a Write lock request lies in the queue. In real systems, most locks are Read locks. For realistic modeling, we need to find out the fraction of Write locks. Also, it has to be determined if the Read/Write ratio is dependent on how frequently an item is locked. Clearly if a more frequently accessed object is Write-locked less frequently than another object, this page would be less of a factor in the overall contention. This sort of behavior may certainly be expected in the index root pages which are accessed much more often than the leaf pages.

Another aspect of modeling Read and Write locks is Write-lock acquisition. The issue here is whether Write locks are set on items to be updated when they are first read, or whether such items are Read-locked and later the Read locks are upgraded to Write. [1] has studied this assumption and concluded that there is a significant difference in performance between the two cases. Also, different locking algorithms are affected to different degrees by this assumption.

User model

The user model describes the arrival process of the user transactions at the system. The arrival process is generally modeled either as an open system [9] (invariably as an exponential inter-arrival distribution) or as a closed system [33] where the users circulate through the system and resubmit transactions after the previous ones are executed. The users' transactions may be batch-type (non-interactive) or interactive. For a closed system, an external think time (deterministic or variable) may be modeled as the mean time between a transaction completion and the next submission from the same terminal. Open system models are invariably modeled using a Poisson arrival process. The transactions in our workloads come from both batch-mode submissions and interactive transactions. We do not have enough information in our traces to distinguish between the two.

Transaction Model

For our purposes, a transaction can be characterized by a string of concurrency control requests, CPU and I/O processing requests and intra-transaction think times. The transaction model defines how these various components combine to form a transaction. The various parameters are the total length of the transactions, order of the CPU, I/O, concurrency control and think time requests, total number of these requests per transaction, duration between requests, portion of database accessed by these requests, etc. A simple model would be to have one class of transactions, each transaction comprised of same number of concurrency control requests, identical duration between the requests without explicitly modeling CPU and I/O computation, and all data items being equally likely to be accessed on any access.

The modeling of transaction lengths is very important. While the mean of transaction lengths is an important parameter, the variation of the transaction lengths is also important, since it is well known in queueing systems that flow times increase with service time variance as well as mean [21].

System model

The system model captures the relevant characteristics of the system design (both hardware and software), including the physical resources (CPUs and disks), their performance parameters, and their associated schedules. The schedule refers to the both sequencing of CPU and

I/O requests and the amounts of service needed. The CPU and I/O time per logical service are specified as model parameters. The models also include the CPU service discipline, the number of CPUs for multiprocessors and the CPU configuration for distributed systems. With the growing interest in distributed database systems, varying the system model configuration and studying its effects on concurrency control performance has become a popular area of research [36, 8].

A few studies have not modeled the system resources at all. They assume that the rate of processing for an individual transaction remains constant independent of the multiprogramming level. This is equivalent to assuming infinite system resources. Although this approach might appear to limit the applicability of such studies to real systems, they do isolate the effects of data contention from resource contention on the system performance.

To the extent that the physical system configuration is sufficient for the offered workload, the issue of the system model is largely separable from modeling the transaction workload. This report addresses the latter issue, and we do not consider the system model further here.

Models in the Literature

We describe many concurrency models from the literature in Appendix B. There we discuss how each of these models address the various issues discussed above.

5 Modeling the database

In this section we will present the database-related characteristics of our workloads, and how the trace characteristics correlate with the assumptions in the literature. Specifically, we will look at the access distribution of locks over the database, the extent of index locking, the distribution of and correlation between Read, Write and Cursor locks, Write-locks, and the importance of Cursor locks.

5.1 Access distribution

The access distribution of locks has a significant impact on the contention. A uniform distribution would equally distribute the contention equally over all items and would be the best for system performance. With an increase in skew of the distribution, the items locked more often tend to become bottlenecks (similar to the *convoy phenomenon* in [5]). The knowledge of the skew in real database systems should be useful for future modeling studies; here we present the actual pattern of distribution of all locks over the data and index items in the database. In Figure 2, we show the skew in the distribution separately over the index and data items (the data for this figure is provided in Appendix C). Table 2 gives the complete statistics about number of locks, type of locks and locktimes.

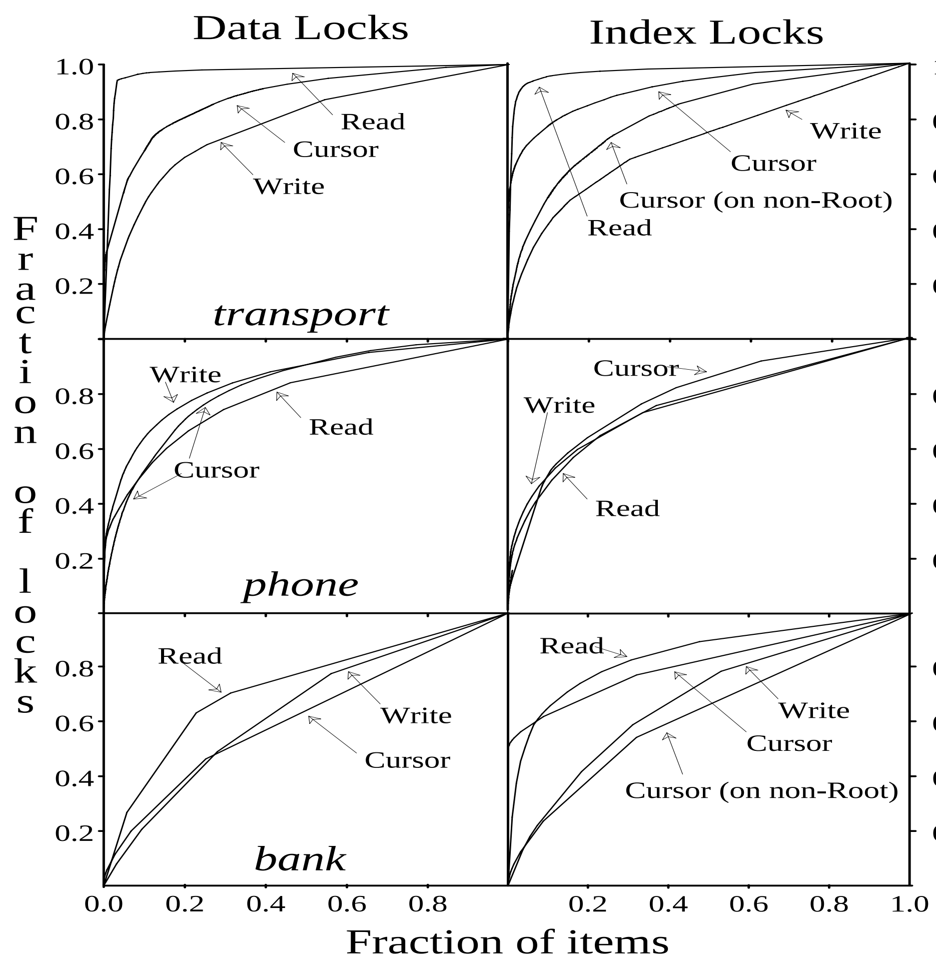


Figure 2: Distribution of locks over the index and database items. The three sets of plots are for *transport*, *phone* and *bank* respectively. Root refers to the root of B-tree indexes. Read and Write (and Cursor for *phone*) lock distributions for non-Root index items are almost identical to the ones in the figure because Root pages acquire these kinds of locks very rarely, as seen in Table 2.

The three traces show differing amounts of skew. *Transport* appears to have the highest skew and *bank* appears to have the lowest. These plots indicate that the common 80–20 and 50–5 estimates both appear to be reasonable. The Root pages of the B-tree indexes form a significant portion of the total Cursor locks for *transport* and *bank* (even though the number of Root items is insignificant compared to the non-Root ones, as seen in Table 2); so in Figure 2 we have also shown the skew for Cursor non-Root items. The figure shows that if we disregard the Root locks, the skew is much lower.

Table 2 provides data about the number of lockable items in the database, and the distribution of the three lock types over these items. It also shows the extent of index locking. Judging from the number and locktimes of index locks in this table, it would appear that index locking should indeed be a part of concurrency control analyses. In section 7, when we present a model of contention measurement, we will get more insight into the importance

	Total	Cursor Locks			Read Locks			Write Locks		
	items	Items	Locks	L-time	Items	Locks	L-time	Items	Locks	L-time
Transport:										
Data	37956	37291	541262	17235.5	780	29189	75522.1	416	1476	2511.1
	73.9%	75.8%	80.1%	62.8%	53.0%	58.0%	64.3%	17.8%	27.8%	41.8%
	100.0%	98.2%	94.6%	18.1%	2.1%	5.1%	79.3%	1.1%	0.3%	2.6%
Index	255	233	74772	990.1	12	126	147.8	21	59	22.4
Root	0.5%	0.5%	11.1%	3.6%	0.8%	0.3%	0.1%	0.9%	1.1%	0.4%
	100.0%	91.4%	99.8%	85.3%	4.7%	0.2%	12.8%	9.2%	0.1%	1.9%
Index	13157	11644	59821	9211.0	680	21036	41783.0	1899	3783	3469.9
Non-Root	25.6%	23.7%	8.9%	33.6%	46.2%	41.8%	35.6%	81.3%	71.1%	57.8%
	100.0%	88.5%	70.7%	16.9%	5.2%	24.9%	76.7%	14.4%	4.5%	6.4%
Phone:										
Data	82859	82260	873599	16049.6	1367	4609	446.7	1413	9787	2266.2
	20.2%	20.4%	38.6%	50.6%	30.6%	37.1%	34.0%	13.6%	31.1%	22.1%
	100.0%	99.3%	98.4%	85.5%	1.7%	0.5%	2.4%	1.7%	1.1%	12.1%
Index	76	73	10493	7.9	2	4	0.9	7	16	2.9
Root	0.0%	0.0%	0.5%	0.0%	0.0%	0.0%	0.1%	0.1%	0.1%	0.0%
	100.0%	96.1%	99.8%	67.5%	2.6%	0.0%	7.7%	9.2%	0.2%	24.8%
Index	327872	320966	1379964	15678.5	3093	7820	865.8	8956	21599	7968.5
Non-Root	79.8%	79.6%	60.9%	49.4%	69.4%	62.9%	65.9%	86.3%	68.8%	77.9%
	100.0%	97.9%	97.9%	64.0	0.9%	0.6%	3.5%	2.7%	1.5%	32.5%
Bank:										
Data	96009	95974	134052	10038.4	35	82	16362.1	32	63	139.3
	60.9%	61.0%	42.0%	47.7%	26.9%	14.6%	66.2%	36.8%	34.1%	34.3%
	100.0%	100.0%	99.9%	37.8%	0.0%	0.1%	61.7%	0.0%	0.0%	0.5%
Index	59	57	93088	1287.9	3	4	0.0	0	0	0.0
Root	0.0%	0.0%	29.2%	6.1%	2.3%	0.7%	0.0%	0.0%	0.0%	0.0%
	100.0%	96.6%	93.4%	100.0%	5.1%	6.6%	0.0%	0.0%	0.0%	0.0%
Index	61500	61399	91858	9730.8	92	474	8365.2	55	122	266.4
Non-Root	39.1%	39.0%	28.8%	46.2%	70.8%	84.6%	33.8%	63.2%	65.9%	65.7%
	100.0%	99.8%	99.4%	53.1%	0.1%	0.5%	45.7%	0.1%	0.1%	1.2%

Table 2: Distribution of Locks among various types of lockable items. Total items denotes the total number of distinct items in the trace; since some items are locked as more than one lock type, this total is less than the sum of item total for the three lock types. The upper percentages in each box represent the ratio of the statistic to the sum of statistics in that column. The lower percentages are the horizontal percentages reflecting the contribution of one lock type among the three lock types. L-time refers to locktime in seconds.

of index locking.

5.2 Cursor locks

Cursor locks have not generally been modeled because use of these locks can lead to semantic inconsistency, since they allow non two-phase locking and permit non-serializable behavior. In commercial systems, however, Cursor locks are indeed used, as is evident from Table 2. That table shows that Cursor locks dominate the total number of locks, and account for greater than 96% of all locks in all cases, except for Index Non-Root locks in *transport*, where they account for more than 70% of the total. One may argue that Cursor locks are locked for relative insignificant amounts of times compared to other locks, and hence they can cause little contention. Looking at the table, this is clearly not the case because the total locktimes of Cursor locks is not insignificant when compared to Read and Write locktimes, even though the individual locktimes for Cursor locks might be more than an order of magnitude lower than that of Read and Write locks. It is important to note that while a Cursor lock may be only held for a short period of time, there may be a significant delay in obtaining a Cursor lock, during which time the transaction waits.

5.3 Correlation between lock types

In this subsection we will study if for a certain lock item, the number of locks of one lock type is correlated with the number of lock types of another type. For example, a positive correlation between Cursor or Read locks and Write locks would indicate that we can expect more contention from Cursor-Write or Read-Write conflicts. The system performance is sensitive to the correlation between pairs of lock types; negative correlation is desirable for lower contention.

In the studies that have assumed a uniform access distribution of locks over all items, the item identifier for each lock request is chosen randomly and independently of anything else from the set of all items. Likewise, independently of the item, with a certain probability, the lock request is a Write request. Thus there is no correlation between Read and Write locks. Studies which have modeled a non-uniform b–c distribution have also chosen Write requests with a probability independent of anything else. Thus pages more likely to be Read locked are assumed to be more likely Write locked.

In real systems, for an arbitrary item, the number of locks belonging to the three types are not independent of one another. There will be objects which will almost always be read and rarely updated. On the other hand, for another set of objects, Write locks for that object may far outnumber the two other types.

To check if there is any correlation between the lock types we look at correlation between ranks of the lock types; rank correlation is measured using the standard Spearman rank

	Lock Type Pair	*# of objects*	*Test statistic $\hat{\rho}$*	*.95 test quantile*	Reject H_0 ?
Transport:					
Data	Cursor-Read	37956	0.1884	0.0101	Yes
	Cursor-Write	37956	0.2587	0.0101	Yes
	Read-Write	37956	0.1662	0.0101	Yes
Index	Cursor-Read	13412	–0.2094	0.0169	Yes
	Cursor-Write	13412	-0.3192	0.0169	Yes
	Read-Write	13412	0.2757	0.0169	Yes
Non-Root Index	Cursor-Read	13157	–0.2078	0.0170	Yes
	Cursor-Write	13157	–0.3185	0.0170	Yes
	Read-Write	13157	0.2778	0.0170	Yes
Phone:					
Data	Cursor-Read	82859	0.0053	0.0068	No
	Cursor-Write	82859	–0.0033	0.0068	No
	Read-Write	82859	0.3591	0.0068	Yes
Index	Cursor-Read	327948	–0.0458	0.0034	Yes
	Cursor-Write	327948	–0.1828	0.0034	Yes
	Read-Write	327948	0.3291	0.0034	Yes
Bank:					
Data	Cursor-Read	96009	–0.0110	0.0063	Yes
	Cursor-Write	96009	–0.0201	0.0063	Yes
	Read-Write	96009	–0.0003	0.0063	No
Index	Cursor-Read	61559	–0.0502	0.0079	Yes
	Cursor-Write	61559	–0.0269	0.0079	Yes
	Read-Write	61559	0.1789	0.0079	Yes
Non-Root Index	Cursor-Read	61500	–0.0501	0.0079	Yes
	Cursor-Write	61500	–0.0269	0.0079	Yes
	Read-Write	61500	0.1818	0.0079	Yes

Table 3: Spearman test for checking independence between two different lock types.

correlation test [6]. For each page we have measured three quantities— number of times it is Cursor locked, Read locked and Write locked. In Table 3 we show the measured correlation between all three pairs of these quantities.

The null hypothesis we want to test is that each pair of quantities is mutually independent versus the alternate hypothesis that the two variables are either positively or negatively correlated. We reject the null hypothesis at .05 level of significance. Since a significant fraction of Cursor index locks belong to Root pages for *transport* and *bank*, we have also included the test for non-Root pages for these traces. Note that even though the absolute correlation statistics are low, they are very significant statistically because of the very large number of objects. Statistical significance, however, does not imply that these low correlations significantly affect simulation results from experiments which do not take this correlation into account; in Section 8, we will study how the correlation between lock types directly affects the contention, using a contention model we will develop in Section 7.

The statistic for *Cursor-Read* correlation is not important for contention as these two types of locks do not conflict. The statistics for all index objects and non-root index objects

Trace	*Data locks*				*Index locks*			
	Upgrade Write		*Direct Write*		*Upgrade Write*		*Direct Write*	
	Locks	*L-time*	*Locks*	*L-time*	*Locks*	*L-time*	*Locks*	*L-time*
Transport	48	178.67	1428	2332.45	377	1537.19	3465	1955.16
Phone	1921	375.75	7866	1890.47	4023	1175.62	17592	6795.73
Bank	0	0.00	63	139.28	14	1.36	108	265.03

Table 4: Comparing Write locks acquired by upgrading Read locks to Write locks acquired directly. The latter is referred to as the no lock upgrades assumption. L-time refers to locktime in seconds.

are virtually the same, indicating that the large number of Cursor locks for root pages do not affect the overall correlation. The table shows very significant positive correlations between Read-Write locks for all sets of objects except data objects for *bank*. This would indicate that Read-Write contention would higher than if we assumed independence. However, since the correlation is much lower than 1, there will be less contention than the studies which use a *b–c* distribution of data and choose Write locks using independent probability values (implying a high positive rank correlation between Read and Write locks). In the table, most Cursor-Write pairs have a significant negative correlation, which makes Cursor-Write contention less probable than for the independence assumption.

5.4 Write-lock assumption

This "write lock" assumption addresses the issue of whether Write locks are acquired directly or are upgraded from Read locks. Some studies (for example, [29]) have assumed that Write locks are acquired directly. On the other hand, [1] has argued that assuming that Write locks are acquired directly (which they call *no lock upgrades* assumption) is incorrect, and that performance results are sensitive to this assumption. The no lock upgrades assumption leads to lower contention in both the blocking version and the restart version of locking algorithms. In the former, the assumptions prevents some deadlocks from happening; in the restart version, the transaction restarts which are inevitable restart sooner if the no lock upgrades assumption is made. However, the performance of the restart version improves more than the performance of the blocking version.

We compare the Write locks acquired directly with the Write locks upgraded from Read locks in Table 4. The table shows that most of the Write locks (approximately every 9 out of 10 Write locks) are acquired directly and the number of Write locks upgraded from Read locks is an order of magnitude lower. This is contrary to the assumption in many recent papers that Write locks are always upgraded from Read locks. As discussed in the previous paragraph, our statistics would imply that studies which model Write locks through upgrades not only underestimate the system performance but also relatively underestimate the performance of locking with restarts more than that of locking with blocking.

6 Transaction behavior modeling

In this section, we study the modeling of transaction behavior. The issues involved are modeling of transaction lengths, locktimes and the distribution of the three kinds of locks and transaction classes.

An issue that we will not address in this section is modeling of resource processing. This refers to the breakdown of resource requirements (e.g., CPU and I/O requests) during a transaction lifetime, the sequence of these request and the distribution of processing requirements. As noted earlier, this is outside the scope of this study, and cannot be done properly with the data we have available.

6.1 Transaction length

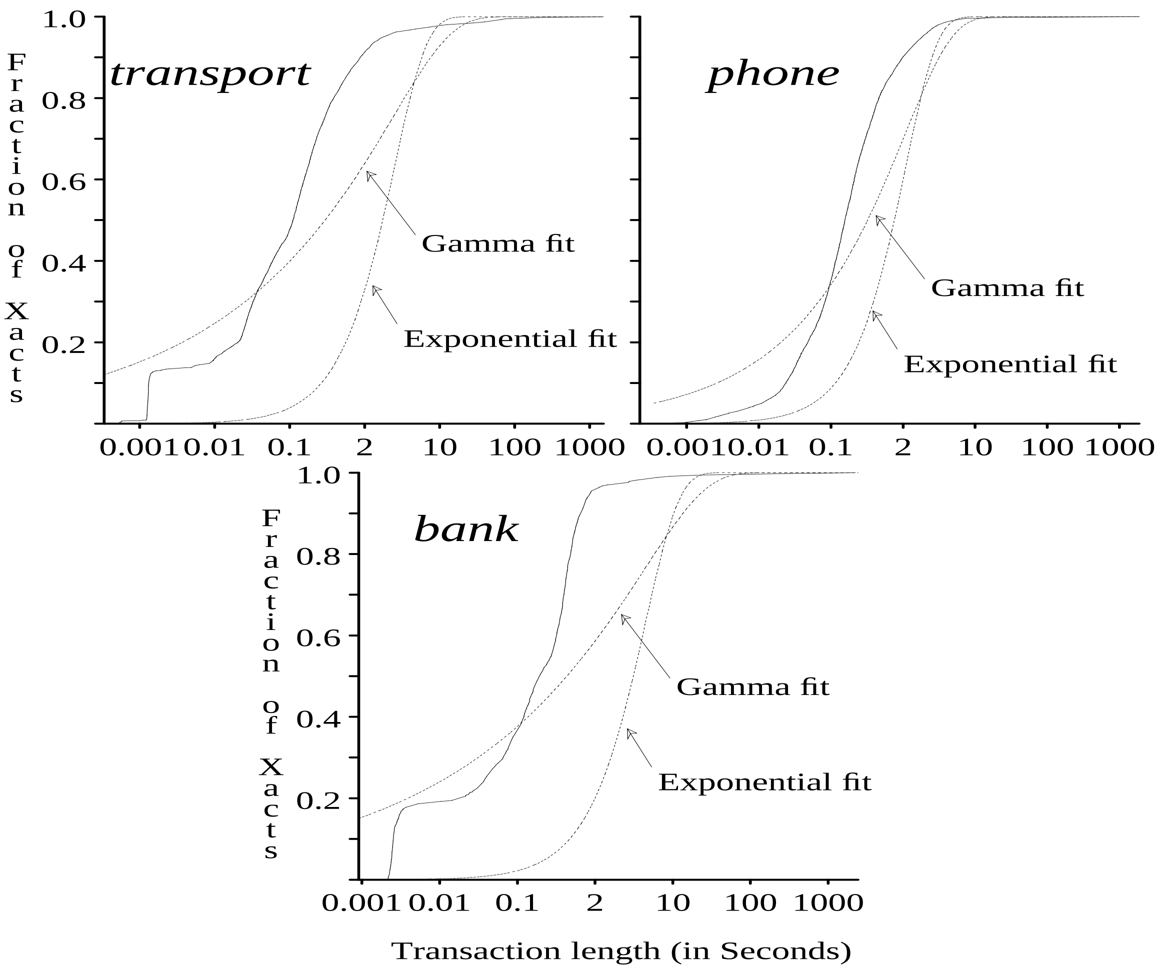
The length of transactions is a very important factor in the analysis of database contention, since long transactions usually imply long lock-waits because Read and Write locks are released only when transactions end. Further, even the distribution of transaction lengths is important, since the system performance is sensitive to the second and third moments of transaction lengths [32]. This is because short transactions can be blocked for long periods by long transactions holding necessary locks.

The transaction length distribution affects the relative performance of the various concurrency control schemes as well as their absolute performance. In locking with restarts, since transactions restart on all lock conflicts, the fact that locktimes may be large for long transactions will have a less negative effect on performance than locking with blocking.

Most concurrency control studies have used fixed length transactions. In Figure 3, we characterize transaction lengths for our traces (the data for the plot is provided in Appendix D). Since a few researchers have used exponential and gamma distributions to model transaction lengths, we have also plotted both the exponential fits and the gamma distribution function[2] fits. Both these fits[3] have been generated using the method of *Maximum Likelihood* [23], using the equations derived in Appendix H. Our analysis of the statistical goodness of fit (in Appendix D) shows that both fits are poor. In particular, both exponential and gamma distribution fits yield much lower figures for variance for all three traces than is measured.

[2] Gamma probability functions are a more general form of exponential probability functions. The density function of a gamma random variable X is defined as $f_X(x) = \frac{\lambda^n x^{n-1} e^{-\lambda x}}{\Gamma(n)}, x > 0$.

[3] The fits in Figure 3 are not as poor as they appear to be— this is because the plots use a log scale and the fits have been obtained before taking the logarithms. It is possible to get exponential fits (by just horizontally shifting the current fits) which will look much closer on this logscale, but are actually much poorer fits, as it becomes evident if we look at those fits on a normal plot.



Trace	*Min.*	*Max.*	*Median*	*Mean*	*Std. dev.*	$E(T^2)$	$E(T^3)$	$E(T^4)$
Transport	0.000504	1525.37	0.110703	2.50572	32.19	1042.5	1.062×10^6	1.341×10^9
Phone	0.000214	1778.91	0.149048	1.04052	23.33	545.4	7.530×10^5	1.218×10^9
Bank	0.002182	2226.72	0.194977	4.47598	82.60	6842.8	1.364×10^7	2.843×10^{10}

Figure 3: Transaction length distribution. The dotted lines show the fitted exponential and gamma distribution to the actual distribution. The last three columns in the table respectively provide the 2nd, 3rd and the 4th moments of transaction lengths.

6.2 Locktimes

As we have defined in Section 2.4, locktimes denote the length of time locks are held on the database items. For Cursor locks, both start and end points need to be known; for Read and Write locks, only start times are necessary, since both are released only at the end of the transaction.

In the literature, locktimes have not been explicitly modeled. In the static locking studies [25] all locks were acquired the beginning of transactions and thus the locktimes are identical to transaction lengths. However, many researchers have argued that static locking models are unrealistic because at the start of a transaction one may not know which pages are going to be locked. In most later studies, dynamic locking has been assumed, and in almost all those studies, Read/Write locks have been uniformly distributed over the entire transaction lengths. Static locking analyses underestimate performance in light workloads because of overestimating lock conflicts. In high-load workloads, dynamic locking may be at a disadvantage because of wasted work due to deadlocks; static locking can never have deadlocks.

There is no reason to believe that in real systems the locking epochs are uniformly distributed. We examine this distribution below.

Cursor locks

As we discussed in Section 2.2, Cursor locks are very common, especially in very long transactions where serializability is traded off for higher performance. Even if Cursor locktimes are short, however, they can't be ignored because of the possible waits to set cursor locks. It may, however, be reasonable to model Cursor lock times as zero; i.e. they are immediately unlocked after locking. We present the necessary measurements of Cursor locks in this subsection.

In Figure 4, we report the distribution of a statistically representative sample of locktimes of Cursor locks (the data for the plot is provided in Appendix E). The plot shows that the locktimes for Cursor locks are more than an order of magnitude lower than the transaction lengths. This can also be observed from Table 2, where we see that the ratio of Read and Write locktimes to the total locktimes is at least an order of magnitude more than the ratio of number of Read and Write locks to the number of all locks. It should also be noted that for Cursor locks, a very small fraction of the locktimes are abnormally high and these locktimes distort the mean and variance of Cursor locktime distributions.

We also tested for independence between transaction length and Cursor locktimes using Spearman's rank correlation test. Since Cursor locks are acquired only to keep the Cursor pointer on the data stable while data is being accessed, we did not expect to see any dependence between transaction length and Cursor locktimes. However, as shown by the results

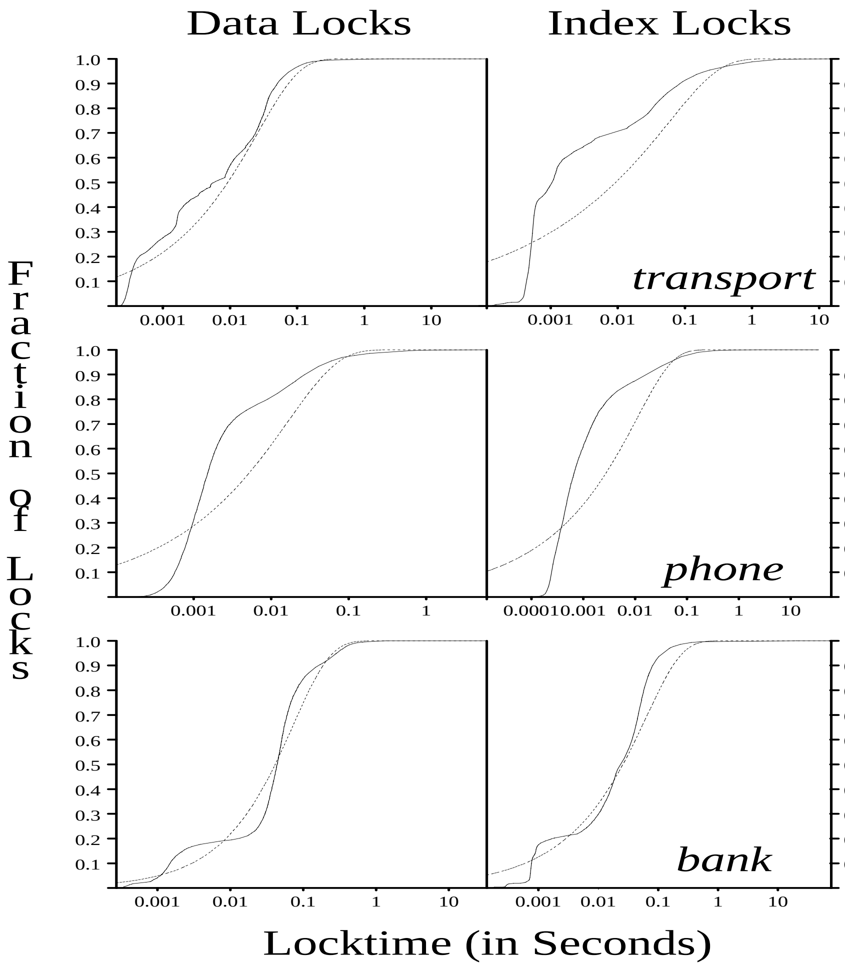


Figure 4: Distribution of locktimes for Cursor locks. The dotted lines are the gamma distribution fits to the actual distribution.

in Appendix F, we found significant positive correlation between the transaction length and locktimes for Cursor locks. This surprising observation could be due to the possibility that longer transactions are executed at lower priority.

If the long transactions do not lock a different set of Cursor objects than the short transactions, the effect of these significant correlations can be simulated in a model by parameterizing the distribution for Cursor locktime as a function of the length of the transaction that required it. However, if long transactions do lock a different set of Cursor objects, the lockable items have to be divided among the long and short transactions and the parameterization of Cursor locktimes has to be based on whether the object is more frequently locked by long transactions or short ones. This clearly requires more detailed database modeling—requiring us to model individual objects and associating them with long and short transactions. In real systems it is very reasonable to expect that long transactions lock a different set of Cursor objects than short transactions. Long transactions generally have a very different nature than short transactions— they access data in a sequential fashion, usually run at a low priority, and are usually batch submissions [24]. All this shows that it may be very difficult to accurately model Cursor locktimes, without resorting to using trace-driven simulations.

Since the locktimes for Cursor locks are much smaller than the transaction lengths and the locktimes for Read/Write locks, small errors in locktimes for Cursor locks should not affect the concurrency control performance analyses to a significant degree. In fact, as we discussed at the beginning of this section, a reasonable approximation would be to model Cursor locks with a locktime of 0— only the acquisition of Cursor locks is important. It would be interesting to investigate this further by experimentally determining the importance of accurate modeling of Cursor locktimes for the final performance results, but we do not do so in this report.

Read/Write locks

For reasons discussed above, we want to know the distribution of locktimes for Read and Write locks. Since Read and Write locks are held until the end of the transaction, we want to test the assumption that these locks are uniformly distributed over the transaction length.

For reasons discussed in Section 2.4, we will characterize lockfractions instead of locktimes for Read and Write locks. By definition, the values of lockfraction lie between 0 and 1. Static locking is equivalent to assuming a lockfraction of 1 for all locks. On the other hand, assuming that locks are acquired uniformly over the transaction length means that the lockfraction is uniformly distributed over the interval $[0, 1]$.

In Figure 5 we plot the distribution of lockfraction for Read and Write locks (the data for the plot appears in Appendix E). We also distinguish between index and data locks. Both the read and write lock acquisitions exhibit considerable skew relative to the uniform distribution assumption. This large variation in locktimes will cause more contention than would be the case for a uniform distribution of locktime, so we can expect that a naive model which assumes a uniform distribution to underestimate contention due to Read-Write lock conflicts. In Section 7, we will investigate if this is true by using a static model for measuring contention.

It is useful to check if the lockfractions are related to transaction lengths. Positive correlation between lockfractions and transaction lengths would indicate a high variance for locktimes. The results are reported in Appendix F; we observe a significant positive correlation in almost all cases for Cursor lockfractions; for Read and Write lockfractions, the results vary widely from positive to negative. In the literature, static locking models have assumed that all locks are acquired at the start of the transactions, implying a lockfraction of exactly 1. More realistic dynamic locking models have assumed that locks are acquired uniformly over the transaction duration. This would again mean that they have implicitly

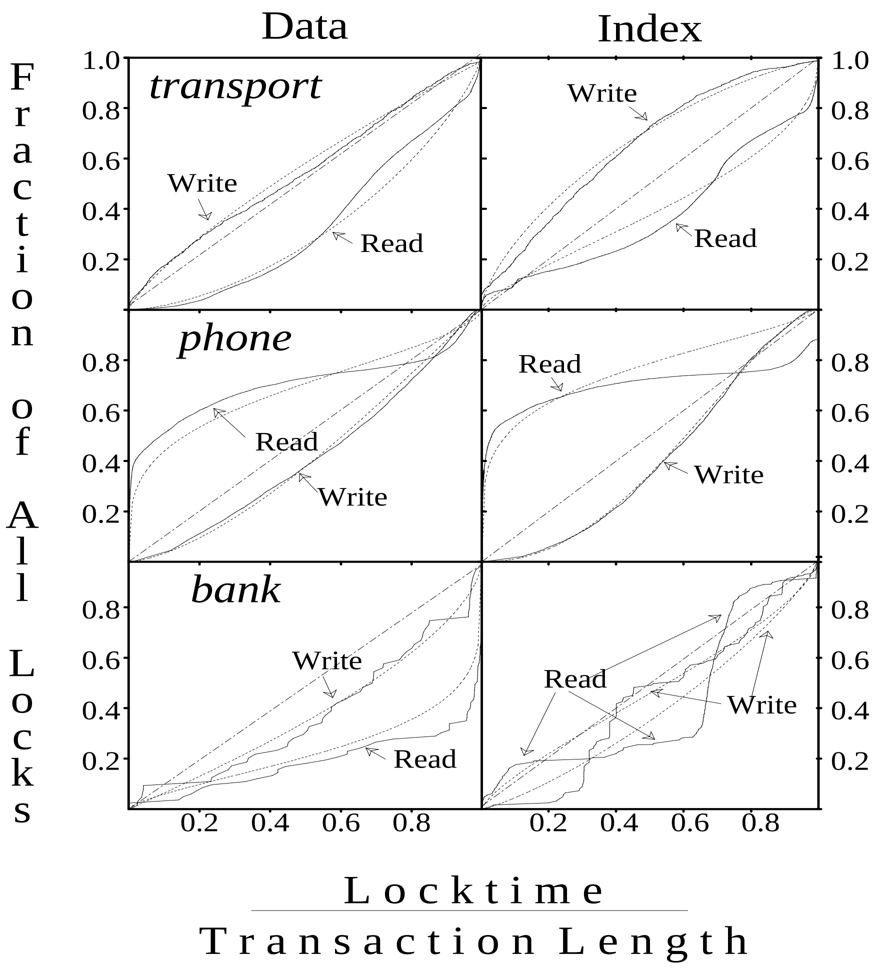


Figure 5: Distribution of locktimes for Read/Write locks. The dotted lines are the beta distribution fits to the actual distribution. The dashed line represents the line with slope 1.

assumed that lockfractions are independent of transaction lengths. Given the wide swings between positive and negative correlations, assuming no correlation is probably a reasonable assumption for most studies.

6.3 Transaction classes

Transaction classes refer to the different kinds of transaction applications running on the system. Most studies have used only one transaction class. [9] is one of the few studies that uses multiple transaction classes— a mix of short transactions with random access and higher probability of write with long transactions doing sequential access with a smaller probability of write. If real transaction behavior varies widely, it is important for many studies for the models to realistically represent this variation, since most performance measures decrease with increased variance in the workload. This can be done by incorporating multiple classes of transactions, each class with significantly different behavior. Using a low variance workload also gives an unfair advantage to the blocking version of locking because there is a shorter bound on the amount of lock-waits, due to low variance in transaction lengths. Another reason for modeling multiple transaction classes is that in the presence of multiple transaction classes, only a subset of them may become the contention bottleneck; modeling a single transaction class means that when contention becomes the bottleneck, no transactions can progress.

We have characterized each transaction present in our workload in tables in Appendix G. There are two classifications: the first is based on the application plan names as given by the system; the second classification is based on three factors: transaction length, Read/Write nature of transactions and two-phase vs non-two-phase transactions. This classification can be used to guide a realistic selection of parameters for concurrency control analysis.

Each transaction in our traces has a *plan name* attached to it. This plan name refers to the application type of the transaction. We first classified the various transactions based on their plan names, and is shown in Tables 13, 14 and 15 in Appendix G. The quantities measured for the transactions include transaction length, Write-locktimes and Read-locktimes. For each plan type, and for both index and data items, we tabulate the number of Read locks acquired, number of Write locks acquired, number of Read locks upgraded to Write, number of locks unlocked and the number of distinct items locked. It is important to distinguish the Write locks directly acquired from the Write locks upgraded through Read locks because, as discussed in Section 5.4, this distinction can significantly impact simulation results.

Not surprisingly, we find a very high variation between the characteristics of the different applications. Average transaction lengths vary a lot over different plans. Some transactions (e.g. plan names *ADCBPL* and *PDCBPL* for *transport*) exhibit two-phase locking behavior (no data locks are Cursor locks). Some plans (e.g. *PPYBXDBL* in *transport*) perform more index locking (both in number of locks and locktime) while others (e.g. *IQ303D* in *transport*) lock data more often.

An interesting observation is the fact that, for most plans, the number of locks acquired by transactions outnumbers the number of distinct items locked. Since Read and Write locks cannot be released before Commit, this fact can be attributed to Cursor locks being acquired on the same items repeatedly.

Another transaction grouping

The plan-based characterization described above exhibits high variability within many plans (i.e. groups). To get a better characterization for modeling, we can group the transactions on three mutually orthogonal axes:

1. Two-phaseness: Since two-phaseness of transactions is an important property, we use that as one of attributes for our grouping. If a transaction releases data locks before

Commit, locking is not two-phase and we group the transaction under *nPL*. If we do not see any data Read locks being unlocked, we assume it has two-phase locking and we label it *2PL*. Note that two-phase transactions may still unlock index pages.

2. Read-Write behavior: A large fraction of the transactions do not acquire Write locks. These transactions may have behavior dissimilar from the transactions that acquire Write locks. Therefore, we distinguish between these two classes of transactions.

3. Length of transactions: We have divided the transactions into 5 classes based on their length. To obtain a uniform logarithmic division over the range of transaction lengths, class i is defined to contain transactions with length l such that

$$i = \lfloor 5 * \frac{\ln l - \ln l_{min}}{\ln l_{max} - \ln l_{min}} \rfloor$$

   where l_{min} and l_{max} are, respectively, the lengths of the shortest and the longest transactions.

Tables 16, 17 and 18 in Appendix G show the characterization of transactions after grouping them by the three parameters discussed above. In these tables, we have also grouped together transactions to compare Read transactions vs Write transactions, and two-phase transactions with non-two-phase.

On the average, over all traces, Read transactions greatly outnumber the Write transactions. Read transactions have a greater variance of transaction lengths, number of locks and locktimes. Two-phase transactions outnumber the non-two-phase transactions. The non-two-phase transactions are much longer and acquire many more locks (mostly Cursor) than two-phase transactions.

Tables 16, 17 and 18 provide detailed statistics for a few workload classes for three different kinds of application areas. The statistics from these tables provide the parameters which one could use to realistically model a database model.

7 Model for Estimating Contention

In the previous sections we characterized the lock distributions over the lockable items (pages for data and subpages for index) for all the three lock types— Cursor, Read and Write. Coupled with our data about the correlation between the three lock types, it roughly indicates the amount of contention in the system and also whether the contention is going to be skewed over a few pages or more uniformly spread over a large number of pages. In this section, we present a simple model that we use to estimate the expected level of contention as a function of the locktime distribution. We also analyze how the contention is distributed over all lockable objects and differentiate between three sources for conflicts— *Cursor-Write*,

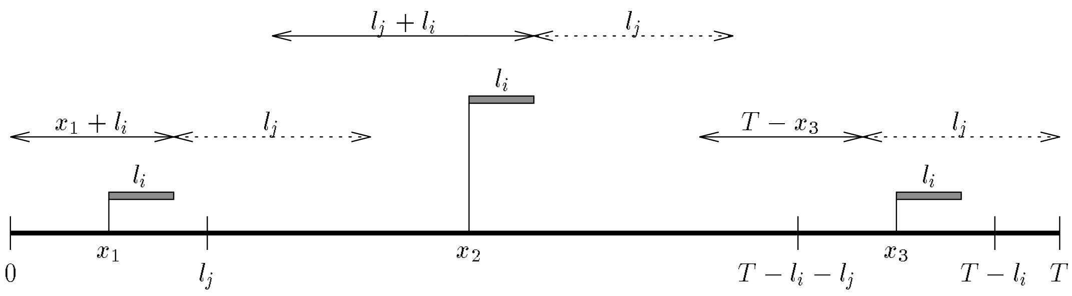


Figure 6: Conflicts between lock requests l_i and l_j. The figure shows the three cases where l_i can start in the intervals $[0, l_j]$, $[l_j, T - l_i - l_j]$ and $[T - l_i - l_j, T - l_i]$. The solid arrow indicates the interval where l_j must start in order for it to conflict with l_i. The dotted arrow along with the solid arrow indicates the interval in which l_j must lie for a conflict.

Read-Write and *Write-Write*. Our analysis indicates which items are likely to become a contention bottleneck as the MPL of the database system is increased.

For our contention model, we will assume that any particular lock request is equally likely to be present at any point of time during the trace period. (Note that our analysis does not use the various correlations that we have computed.) We will then compute the expected number of conflicts for any object by summing the probabilities of all possible pairs of conflicting lock requests for that object. Thus, for any pair of lock requests l_i and l_j for the same object, we are interested in finding the probability $p(i, j)$ that the pair conflicts during the trace execution. Now, obviously $p(i, j)$ is non-zero if and only if the two requests form one of the three conflicting lock type pairs described above. We solve for such a case. Let l_i and l_j also denote the locktimes for the locks, computed from our traces.

Let the trace period be T. We can calculate $p(i, j)$ as follows:

$$p(i,j) = \int_0^T \text{Prob}(l_i \text{ begins in } [x, x + dx])\text{Prob}(l_j \text{ conflicts with } l_i)$$

We can calculate the probability $p(i, j)$ as a sum of three cases: depending on whether l_i begins in the interval $[0, l_j]$,$[l_j, T - l_i - l_j]$ or $[T - l_i - l_j, T - l_i]$. Note that l_i cannot start after $T - l_i$. As Figure 6 illustrates, we have the following expression for $p(i, j)$:

$$\begin{aligned} p(i,j) &= \int_0^{l_j} \frac{dx}{T - l_i} \frac{x + l_i}{T - l_j} + \int_{l_j}^{T - l_i - l_j} \frac{dx}{T - l_i} \frac{l_j + l_i}{T - l_j} + \int_{T - l_i - l_j}^{T - l_i} \frac{dx}{T - l_i} \frac{T - x}{T - l_j} \\ &= \frac{(T - l_i - l_j)(l_i + l_j) + l_i l_j}{(T - l_i)(T - l_j)} \end{aligned}$$

If $T \gg l_i$ and $T \gg l_j$, we have,

$$p(i,j) = \frac{l_i + l_j}{T} \tag{1}$$

Now, for any lockable item X, let p_X denote the expected number of lock conflicts for X

if the trace workload is executed in time T. Hereafter we refer to p_X as the *contention index* for X.

$$p_X \equiv \sum_{i,j,i\neq j} p(i,j)$$

where the summation is over all conflictable locks for X. If we decompose p_X into the three different kinds of lock conflicts, namely Cursor-Write, Read-Write and Write-Write, we have

$$p_X = p_{X,cw} + p_{X,rw} + p_{X,ww}$$

where the three terms on the right side denote the contributions of the three pairs of lock types.

We now derive the expressions for the above three contention indices. Let $n_{X,c}$, $n_{X,r}$ and $n_{X,w}$ represent the total number of Cursor, Read and Write locks for X. Let $c_{X,1},\ldots,c_{X,n_{X,c}}$ denote the locktimes for the Cursor locks. Similarly, let $r_{X,1},\ldots,r_{X,n_{X,r}}$ and $w_{X,1},\ldots,w_{X,n_{X,w}}$ be defined for Read and Write locks. Using (1), the values of these three terms are:

$$\begin{aligned} p_{X,cw} &= \frac{\sum_{j=1}^{n_{X,c}}\sum_{i=1}^{n_{X,w}}(w_{X,i}+c_{X,j})}{T} = \frac{n_{X,c}t_{X,w}+n_{X,w}t_{X,c}}{T} && (2)\\ p_{X,rw} &= \frac{\sum_{j=1}^{n_{X,r}}\sum_{i=1}^{n_{X,w}}(w_{X,i}+r_{X,j})}{T} = \frac{n_{X,r}t_{X,w}+n_{X,w}t_{X,r}}{T} && (3)\\ p_{X,ww} &= \frac{\sum_{j=1}^{n_{X,w}}\sum_{i=j+1}^{n_{X,w}}(w_{X,i}+w_{X,j})}{T} = \frac{(n_{X,w}-1)t_{X,w}}{T} && (4) \end{aligned}$$

where $t_{X,c} = \sum_{i=1}^{n_{X,c}} c_{X,i}$; $t_{X,r} = \sum_{i=1}^{n_{X,r}} r_{X,i}$; $t_{X,w} = \sum_{i=1}^{n_{X,w}} w_{X,i}$ (summations over the individual locktimes for the item X).

Table 5 presents the amount of predicted contention in each of the workloads. The three data columns represent the expected number of lock conflicts due to Cursor-Write (E_{cw}), Read-Write (E_{rw}) and Write-Write (E_{ww}) conflicts, respectively. These quantities are calculated by adding up the respective quantities per page:

$$E_{cw} = \sum_X p_{X_i,cw};\ E_{rw} = \sum_X p_{X_i,rw};\ E_{ww} = \sum_X p_{X_i,ww} \qquad (5)$$

As noted earlier in this report, the actual, measured, number of lock conflicts for the *transport* and *bank* workloads was quite low. This is confirmed by our analysis, which in Table 5 shows that the expected number of conflicts is also very low. This very low level of conflict makes conclusions from our measurements and estimates for those workloads highly speculative, and that observation should be kept in mind in the remainder of this discussion.

We observe from the table that in *transport* and *phone* traces, data contention is about 75–80% of the total contention. The fact that index locks are locked on a finer granularity than data locks definitely contributes to the low contribution of index contention. In *bank*,

	No. of items	E_{cw}	E_{rw}	E_{ww}
Transport:				
Data	350	9.321047 (67.07%)	0.897341 (6.46%)	0.741479 (5.34%)
Index	1096	1.391543 (10.01%)	0.578239 (4.16%)	0.967619 (6.96%)
Phone:				
Data	1279	103.86584 (22.77%)	73.11964 (16.03%)	171.23769 (37.54%)
Index	4783	47.9453 (10.51%)	16.62089 (3.64%)	43.33041 (9.50%)
Bank:				
Data	24	0.012057 (1.67%)	0.000000 (0.00%)	0.074058 (10.25%)
Index	4783	0.360850 (49.96%)	0.053500 (7.41%)	0.221764 (30.71%)

Table 5: Distribution of expected number of lock conflicts over the three conflict type pairs. Percentages are over all conflict types for both index and data items. The number of items is the number of items which are locked more than once and Write-locked at least once, i.e. all items which can cause a conflict in a trace-driven simulation.

index contention dominates the total contention; however, the amount of contention in *bank* is orders of magnitude lower than other traces. In the *transport* and *bank* traces, most of the contention is due to *Cursor-Write* conflicts. In Section 2 we had suggested that, probably to gain a performance benefit, Cursor locks are used in place of Read locks, which provide greater semantic consistency. The fact that most of the contention in two of the traces is due to *Cursor-Write* conflicts emphasizes the importance of modeling Cursor locks instead of Read locks. Had Cursor locks been replaced by Read locks the performance would most likely have degraded considerably because Read locks cannot be unlocked before the transaction commits. In the *phone* trace, contention is more evenly split among the various kinds of conflicts.

For the data contention in *transport*, the contention is dominated by *Cursor-Write* contention. This is not surprising considering our analysis in Section 3.5 where we showed that for *transport*, the Write locks have a low skew over the database but they have a high positive correlation with Cursor locks. There is a negative correlation for *Cursor-Write* in case of *phone* and this partially explains the more even split among the various conflict types.

In Figure 7 we show the contention factor of the individual items with the highest predicted contention. Index contention seems to be much more uniformly spread over all items as compared to the data items. In fact, for the *phone* trace, two pages are responsible for 64.7% of the data contention and 49.4% of the total contention. These two pages are very likely to form a bottleneck (if there isn't one already) when the system load is increased.

The analysis in this section may be misleading in a few cases. First, if there is moderate to high contention for some items, the value of p_X for those items as calculated in this section

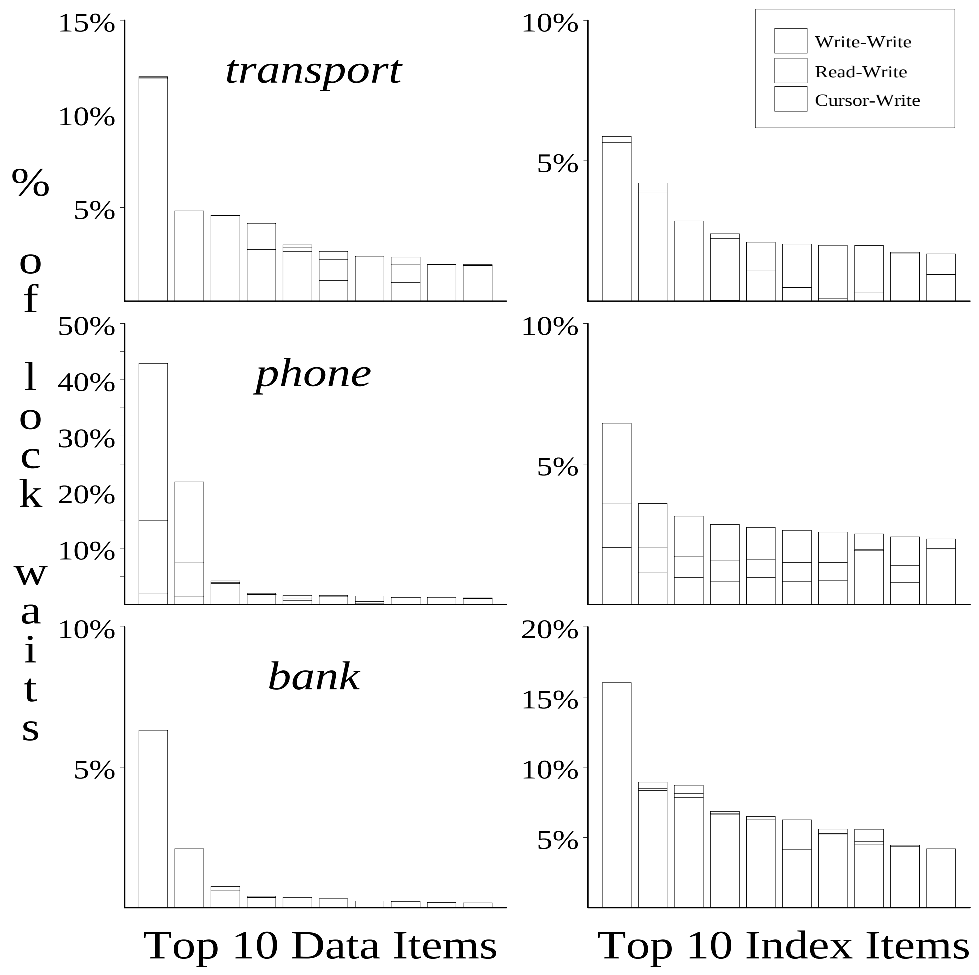


Figure 7: Contention index for individual lock items with maximum contention factor. The y-axis denotes the contention index for an object as a percentage of $E = E_{cw} + E_{rw} + E_{ww}$

may be an underestimate. This is because once we have frequent lock-waits on a particular item, the effective values of locktimes for the waiting lock request increases by the amount of time it has to wait. Thus some or all of $t_{X,w}, t_{X,r}, t_{X,c}$ may rise, causing our calculated values of the expected number of conflicts to be underestimates. A side-effect of higher contention also trickles down to the the items which are themselves *not* under high contention. This happens because of queueing effects— when lock-waits occur, not only does the locktime of a requesting lock request rise but also the locktimes for the locks being held by the requesting transaction rises.

Another source of inaccuracy would be if the jobs that submit the transactions inherently order the transactions such that there are fewer (or more!) chances of conflict than predicted by our model. In here we have assumed that at any moment, regardless of the state of the system, all of the remaining transactions have equal likelihood of being scheduled. This may not be the case in reality. Further, we have not incorporated the various positive and negative correlations that we observed between the occurrence of various lock types. The effect of positive and negative correlations was discussed earlier in this report.

Validation of contention model

In this section, we compare our predicted levels of lock contention with those measured in the real systems. Two of the three systems were running light workloads and exhibited little contention— *bank* had 0 lock-waits in the entire trace and *transport* had 6 lock-waits. In table 5, we predicted about 14 lock waits for *transport* and about 1 for *bank*, so the observed and predicted figures are reasonably close - certainly within the bounds of statistical error of the model.

As we can see from Figure 1, *phone* was a heavily loaded system— the reason the organization wanted their system to be traced was they were having performance problems due to high contention. This trace showed 271 lock-waits in the 30-minute trace. Since only this trace had a substantial number of lock-waits, we discuss only this system in detail.

If we look at our contention model in the previous subsection, we would expect the two data objects with the highest expected number of lock-waits to be the bottleneck in the worst case, or to at least dominate the total number of lock-waits in the best case. Most surprisingly, *none* of the 271 lock-waits is due to data objects; all of them have been caused by index objects. Presumably the ordering of transactions in the trace, and/or the various correlations discussed earlier are the cause of the discrepancy with our predictions from our simple model.

Using our contention model, we ordered the index objects based on the expected number of lock waits for each object. There were 4783 objects with a non-zero value. The 271 lock-waits in the trace were caused by 46 different objects. We ordered these 46 pages based on the number of traced lock-waits. Then, to validate our model we performed a rank correlation test to check the correlation between the measured ranks and the ranks predicted by our model in the previous section. The correlation statistic yielded a value of 0.3188 which is higher than the .95 test quantile of 0.2922. Thus we do have a weak correlation of the measured waits with our model. On the other hand, the correlation is not very strong, suggesting that one or more of the inaccuracies we mentioned above might be the cause. It must be noted that there were frequent deadlocks in the system during the trace. Since deadlock detection occurs only at (large) intervals of 30 seconds in this DB2 installation, when deadlocks occur the locktimes of objects would appear to be much larger than their values in the input workload. This would definitely cause many more spurious lock waits than in the case where deadlock detection occurred at smaller intervals.

8 Effect of correlation between lock types on contention

In Section 5.3, we measured the correlation between the three lock types and mentioned that this correlation is an important factor in causing contention. High values for correlation between Write locks and Cursor/Read locks can lead to high contention. In this section we will use the contention model we developed in the last section to analyze the effect of correlation between lock types.

Read and Cursor locks do not conflict with each other, so we will study only correlations among Read-Write and Cursor-Write lock types. We will determine the predicted contention in the workload assuming correlation values of –1, 0 and 1. We will then compare this with the actual contention values which we measured in the last section.

A (rank) correlation of 1 between two lock types (for example, Cursor and Write) means that if we order the items by the frequency of one lock type, and we obtain another order by using the frequency of the other lock type, the two orders are the same. We use the following procedure to determine the amount of contention that would be caused if the correlation between the Cursor and Write lock types were 1. We sort the N lockable items into two sorted lists— one sorted by the frequency of Cursor locks and the other by the frequency of Write locks. Then we create N imaginary items, and associate the ith imaginary item with the ith sorted (Cursor locktime, Cursor lock frequency) pair and the ith sorted (Write locktime, Write lock frequency) pair. Then we can determine the contention corresponding to a correlation of 1 by using the formulation we derived in the last section.

For a correlation of –1, we simply associate the ith imaginary item with the ith sorted (Cursor locktime, Cursor lock frequency) pair and the $(N - i + 1)$th sorted (Write locktime, Write lock frequency) pair.

To compute the contention for correlation of 0, we use the following formulation. A correlation of 0 corresponds to the fact that any of the N Cursor locktimes could be associated with any of the N Write locktimes with equal likelihood. Thus the ith sorted Cursor locktime and Cursor lock frequency is associated with the jth sorted Write locktime and Write lock frequency with a probability of $1/N$. Using this and equations (2)-(5), the expected contention for a locktype correlation of 0 corresponds to the following equations:

$$\begin{aligned}
E_{cw} &= \sum_{X=1}^{N}\sum_{Y=1}^{N} \text{Prob(item } X \text{ is associated with item } Y) \cdot \frac{n_{X,c}t_{Y,w} + n_{Y,w}t_{X,c}}{T} \\
&= \sum_{X=1}^{N}\sum_{Y=1}^{N} \frac{1}{N} \cdot \frac{n_{X,c}t_{Y,w} + n_{Y,w}t_{X,c}}{T} \\
&= \frac{\sum_{X=1}^{N} n_{X,c} \sum_{X=1}^{N} t_{Y,w} + \sum_{X=1}^{N} n_{Y,w} \sum_{X=1}^{N} t_{X,c}}{NT}
\end{aligned}$$

	Lock-type pair	*Contention with*		*Workload*	*Measured*	*Statistical*
		Correlation 0	*Correlation* 1	*Contention*	*Correlation*	*Significance*
Transport:						
Data	Cursor-Write	3.30746	137.88495	9.32105	0.2587	+
	Read-Write	0.44136	154.97754	0.89734	0.1662	+
Index	Cursor-Write	3.44249	85.06456	1.39154	–0.3192	—
	Read-Write	1.58865	103.89470	0.57824	0.2757	+
Phone:						
Data	Cursor-Write	14.45199	785.68190	103.86584	–0.0033	0
	Read-Write	0.10021	79.74778	73.11964	0.3591	+
Index	Cursor-Write	19.51881	535.81044	47.94530	–0.1828	—
	Read-Write	0.13858	26.32107	16.62089	0.3291	+
Bank:						
Data	Cursor-Write	0.05745	4.92254	0.01206	–0.0201	—
	Read-Write	0.00310	9.63043	0.00000	–0.0003	0
Index	Cursor-Write	0.23492	1497.18857	0.36085	–0.0269	—
	Read-Write	0.00533	4.29717	0.05350	0.1789	+

Table 6: Expected number of lock conflicts with correlation of 0 and 1 between the locktypes. The workload contention in the predicted contention if transactions occurred randomly in time (same as in Table 7). We have also shown the measured correlation (from Table 3). +, — and 0 indicate significant positive, significant negative and insignificant correlations, respectively.

Similarly,

$$E_{rw} = \frac{\sum_{X=1}^{N} n_{X,r} \sum_{X=1}^{N} t_{Y,w} + \sum_{X=1}^{N} n_{Y,w} \sum_{X=1}^{N} t_{X,r}}{NT}$$

We report the values of the expected contention for correlations of 0 and 1 for both Cursor-Write and Read-Write pairs in Table 6. The table does not contain values for correlation –1, because there will be *no* contention at correlation –1. This is because the pages which have a non-zero Write locks with be associated with pages with zero Read and Cursor locks, leading to zero contention.

Our first observation is that a few pages (even a single page) can affect the contention drastically; however, a few pages do not have any significant effect on the *rank* correlation, which is what was reported in Section 5.3. This effect is most clearly seen for Index contention in *bank* for a Cursor-Write correlation of 1. More than 99% of the contention is due to a single page which accounts for 45.2% of the total Cursor locks and 10.1% of the total Cursor locktime. In the workload this page has no Write locks and hence accounts for no actual contention.

The above observation is the reason why the statistical significance tests we performed in Section 5.3 may not be sufficiently indicative of the effect of correlation on the contention. From Table 8, we notice that in some cases even though the correlation is statistically significant, the actual contention is close to the contention for a 0 correlation. However, in other cases, the statistically significance test correctly predicts the effect of positive (or negative) significant correlation on contention— for example, contention due to Read-Write

Data locks in *phone* and Cursor-Write Index locks in *transport*. So, the statistical tests are modestly successful in determining the significance of the correlation for the purposes of predicting contention.

9 Conclusions

In this report we have looked at the locking behavior of three real relational database systems running three different kinds of real applications. We have used our traces to obtain a comprehensive characterization of transaction workloads in real commercial database systems. We believe that this data will be valuable not only to the researchers who can use it to create more accurate models, but also to database designers who can use this information for a deeper understanding of real world transaction workloads. It is important to note, however, that our characterization does not necessarily have predictive power - i.e. we have measured various parameters of the workload, but that does not allow us to be confident that we can predict the values of parameters that were not measured.

We have also looked at the various assumptions made in the concurrency control performance analysis literature, have measured their validity in the traces, and have analyzed the sensitivity of predicted system performance to these assumptions. A future direction of research would be to use trace-driven simulation to analyze the sensitivity of the assumptions more rigorously.

We have also constructed a very simple static model for contention, and have shown that it provides reasonably accurate estimates of observed contention. We believe that extending that model to include factors such as observed correlations should provide still better agreement with the measured data.

10 Acknowledgements

We would like to thank Ted Messenger of IBM Almaden Research Center for providing some of the traces, and Steve Luzmoor of Cray Research for his help in transfering these large traces to the tape readers on our machines.

References

[1] R. Agrawal, M. J. Carey, M. Livny. Concurrency Control Performance Modeling: Alternatives and Implications. *ACM Transactions on Database Systems*, Dec. 1987.

[2] R. Bayer, M. Scholnick. Concurrency of Operations on B-trees. *Acta Informatica*, 1977.

[3] P. A. Bernstein, V. Hadzilacos, N. Goodman. Concurrency Control and Recovery in Database Systems. Addison-Wesley, Reading, Massachusetts, 1987.

[4] P. A. Bernstein, N. Goodman. Concurrency Control in Distributed Database Systems. *Computing Surveys*, June 1981.

[5] M. Blasgen, J. Gray, M. Mitoma, T. Price. The Convoy Phenomenon. *Operating Systems Review*, 13(2), Apr. 1979.

[6] W. J. Conover. Practical Nonparametric Statistics. John Wiley & Sons, New York, 1980.

[7] M. J. Carey, S. Krishnamurthy, M. Livny. Load Control for Locking: the "Half and Half" Approach. In *Proc. of the 9th ACM Symp. on Principles of Database Systems*, Nashville, Tennessee, 1990.

[8] M. J. Carey, M. Livny. Distributed Concurrency Control Performance: A Study of Algorithms, Distribution and Replication. In *Proc. of the 14th Intl. Conf. on Very Large Database Systems*, Los Angeles, California, 1988.

[9] M. J. Carey, M. Stonebraker. The Performance of Concurrency Control Algorithms for Database Management Systems. In *Proc. of the 10th Intl. Conf. on Very Large Database Systems*, Singapore, 1984.

[10] C. J. Date. An Introduction to Database Systems, Vol. II. Addison-Wesley, Reading, Massachusetts, 1987.

[11] A. Dan, D. M. Dias, P. S. Yu. The Effect of Skewed Data Access on Buffer Hits and Data Contention in a Data Sharing Environment. In *Proc. of the 16th Intl. Conf. on Very Large Database Systems*, Brisbane, Australia, 1990.

[12] C. J. Date, C. J. White. A Guide to DB2: A User's Guide to the IBM product IBM DATABASE 2 and its Major Companion Products, 4th Ed. Addison-Wesley, Reading, Massachusetts, 1993.

[13] K. P. Eswaran, J. N. Gray, R. A. Lorie, I. L. Traiger. The Notions of Consistency and Predicate Locks in a Data Base System. *Communications of the ACM*, Nov. 1976.

[14] P. A. Franaszek, J. T. Robinson, A. Thomasian. Wait Depth Limited Concurrency Control. In *Proc. of the 7th Intl. Conf. on Data Engineering*, Kobe, Japan, 1991.

[15] J. N. Gray, R. A. Lorie, G. R. Putzolu, I. L. Traiger. Granularity of Locks and degrees of Consistency in a Large Shared Data Base. In *Modeling in Data Base Management Systems*, Elsevier North-Holland, New York, 1976.

[16] T. Haerder, A. Reuter. Principles of Transaction-Oriented Database Recovery. *Computing Surveys*, Dec. 1983.

[17] P. Hawthorn, M. Stonebraker. Performance Analysis of a Relational Data Base Management System. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data*, Boston, Massachusetts, 1979.

[18] IBM Corp. IBM Database 2: System Planning and Administration Guide, SC26-4085-3, IBM Corporation, White Plains, New York, May 1987.

[19] T. Johnson, D. Shasha. A Framework for the Performance Analysis of Concurrent B-Tree Algorithms. In *Proc. of 9th Symposium on Principles of Database Systems*, 1990.

[20] J. P. Kearns, S. DeFazio. Diversity in Database Reference Behavior. *Performance Evaluation Review*, May 1989.

[21] L. Kleinrock, Queueing Systems, Vols. I and II, Wiley, 1975.

[22] H. T. Kung, J. T. Robinson. On Optimistic Methods for Concurrency Control. *ACM Transactions on Database Systems*, June 1981.

[23] H. J. Larson. Introduction to Probability Theory and Statistical Inference. John Wiley & Sons, New York, 1982.

[24] Ted Messenger, IBM Almaden Research Center. Personal Communication, Nov. 1990.

[25] R. J. T. Morris, W. S. Wong. Performance Analysis of Locking and Optimistic Concurrency Control Algorithms. *Performance Evaluation*, May 1985.

[26] A. Moenkeberg, G. Weikum. Conflict-driven Load Control for the Avoidance of Data-Contention Thrashing. In *Proc. of the 7th Intl. Conf. on Data Engineering*, Kobe, Japan, 1991.

[27] J. Rodriguez-Rosell. Empirical Data Reference Behavior in Data Base Systems. *IEEE Computer*, Nov. 1976.

[28] D. R. Ries, M. R. Stonebraker. Locking Granularity Revisited. *ACM Transactions on Database Systems*, June 1979.

[29] I. K. Ryu, A. Thomasian. Analysis of Database Performance with Dynamic Locking. *Journal of the ACM*, July 1990.

[30] A. J. Smith. Sequentiality and Prefetching in Data Base Systems. *ACM Transactions on Database Systems*, September 1978.

[31] V. Srinivasan, M. J. Carey. Performance of B-Tree Concurrency Control Algorithms. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data*, Denver, Colorado, 1991.

[32] A. Thomasian. Performance Limits of Two-Phase Locking. In *Proc. of the 7th Intl. Conf. on Data Engineering*, Kobe, Japan, 1991.

[33] Y. C. Tay, N. Goodman, R. Suri. Locking Performance in Centralized Databases. *ACM Transactions on Database Systems*, Dec. 1985.

[34] A. I. Verkamo. Empirical Results on Locality in Database Referencing. In *Proc. of the ACM SIGMETRICS Conf.*, Texas, Austin, 1985.

[35] S. Viavant. Collection, Reduction and Analysis of DB2 Trace Data. *M. S. Report*, University of California, Berkeley, California, July 1989.

[36] Y. Wang, L. A. Rowe. Cache Consistency and Concurrency Control in a Client/server DBMS Architecture. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data*, Denver, Colorado, 1991.

[37] P. S. Yu, D. W. Cornell, D. M. Dias, A. Thomasian. On Coupling Partitioned Database Systems. In *Proc. of the 6th Ann. Symp. on Distributed Computing*, May 1986.

[38] P. S. Yu, D. M. Dias, J. T. Robinson, B. R. Iyer, D. W. Cornell. Distributed Concurrency Control for Data Sharing. In *Proc. of the Conf. on Management and Performance Evaluation of Computer Systems*, 1985.

[39] P. S. Yu, D. M. Dias, J. T. Robinson, B. R. Iyer, D. W. Cornell. Modeling of Centralized Concurrency Control in a Multi-System Environment. In *Proc. of the ACM SIGMETRICS Conf.*, Texas, Austin, 1985.

[40] P. S. Yu, D. M. Dias, J. T. Robinson, B. R. Iyer, D. W. Cornell. On Coupling Multi-Systems Through Data Sharing. *Proceedings of the IEEE*, May 1987.

A Terminology and Background

In here, we describe some background required for our paper.

Transaction – a unit of work

In database systems, a transaction represents a unit of work. Typically a transaction represents one meaningful activity in user's environment; for example, in a database for a financial institution, transferring money from one account to other would classify as a single transaction.. A transaction must satisfy the ACID property [16]: A for *atomic*, it must appear that either all or none of the operations have been executed; C for *consistency*, each transaction, by definition, preserves the consistency of the database; I for *isolation*, events within a transaction must appear to be isolated from other concurrent transactions; D for *durability*, once a transaction has ended (or *Commit*ed) the system must guarantee that the results will survive any malfunctions. Concurrency control is concerned with the problem of maintaining isolation. Logging and transaction recovery are responsible for maintaining atomicity and durability. *Serializability* of transactions has been widely accepted as the criterion for correctness for database systems [10]. An interleaved set of transactions is said to be serializable if and only if it produces the same results as some serial execution of the same transactions.

Concurrency control alternatives

There is a variety of ways in which concurrency control can be implemented in a database system. Most of these techniques fall under one of three broad categories— locking, optimistic concurrency control and timestamping. The reader is referred to [3] for a detailed survey of various schemes. In the following, we will give a brief introduction to these schemes.

1. *Locking.* The idea is very simple— each transaction acquires read locks before reading data and write locks before writing them. The lock manager guarantees that if an item is write-locked no other locks are permitted on it. This guarantee can be implemented in a variety of ways which span a spectrum; the two ends of this spectrum are *locking with blocking* and *locking with restarts.*

   Almost all the commercial systems that exist today use *locking with blocking* as the concurrency control option. This alternative requires that transactions obtain Read or Write locks before accessing and manipulating data. The transactions block until the lock they request is granted. As opposed to this, *locking with restarts* refers to the scheme where the transactions abort and restart if the lock they request cannot be granted immediately. Most performance studies have argued that, under most circumstances, locking with blocking performs better than other versions of locking and also optimistic concurrency control and timestamping [29, 1, 9]. Policies which limit the size of the of the lock-wait queue [14] lie somewhere in the middle of the spectrum.

2. *Optimistic concurrency control.* The basic idea is that transactions are allowed to execute unhindered until they reach their commit point, at which point they are validated [22]. Transactions maintain a list of records they have read or written in the form of read and write sets. A transaction fails validation if any granule in its readset has been written by a transaction that committed during its lifetime. Invalidated transactions are aborted and restarted.

3. *Timestamping.* Each transaction has a timestamp $ts(t_i)$, usually the time the transaction started. Each data granule x has a read timestamp $rts(x)$ and $wts(x)$ which denote the timestamps of the

latest reader and the latest writer respectively for x. A read request from t_i for x is granted only if $ts(t_i) >= wts(x)$, and a write-request from t_i for x is granted only if both $ts(t_i) >= rts(x)$ and $ts(t_i) >= wts(x)$. Transactions whose requests are not granted are aborted and restarted. For various versions of timestamping, the reader is referred to [4].

B Models in the literature

In this section we review a small sample of the most popular studies in concurrency control literature. In light of the discussion that preceded this subsection, we will focus on the modeling assumptions made in these studies. In the sections following this, we will examine the validity of these assumption using trace measurements.

[9] is among the more popular earlier studies. The objective of this paper is to compare various concurrency control algorithms in a variety of workloads. An open queueing model has been used to model the transactions. The system resources (CPU and I/O) have been explicitly modeled using CPU or I/O processing times per transaction step. Any transaction can belong to one of two classes— short (fewer lock requests) with random data access and long with sequential access. The number of transaction steps[4] is a constant or a uniform (over the interval 1 to $2 * mean$) random variable. For each lock request, the page is selected uniformly from the entire database, independent of anything else. The number of lockable objects in the database is varied from 1 to 10000. The short transactions lock an average of 2 objects with a write probability of 0.5 and the long one locks 30 objects with a write probability of 0.1. The resource requirements between lock request are identical. The main conclusion of this study was that, under most circumstances, locking with blocking gives the best performance.

A good example of an analytical study for concurrency control performance is the study by [33]. The objective of this study is to analytically model two kinds of locking strategies: locking with blocking and locking with restarts. This study uses a closed queueing model. System resources are not modeled and infinite resource conditions are assumed. This is to study the effects of data contention (in isolation from resource contention) as we discussed in section 4.1. All transactions request an identical number of locks, with the inter-request processing time a uniform random variable. Lock requests are chosen from a b–c data distribution (b% of the accesses go to c% of the data). However, each lock request is chosen independent of anything else. Under such conditions, they have proved that independently choosing a fixed fraction of lock requests as write requests is equivalent to a system with fewer lockable objects where all locks are Write. They have reported results with two sets of parameters— a 40 object database, with transactions consisting of 2 lock requests each, and a 10000 object database with transactions consisting of 20 lock requests each. Their results claim that, under infinite resource assumption, locking with blocking wins in low to moderate contention environments. Under higher contention environments, locking with restarts wins.

The only group of researchers who have used real database system traces for concurrency control analyses is [38, 39, 37, 40]. They have used a 15-minute trace from a hierarchical database system (IMS). However the focus of their study was not a detailed trace characterization. The only characteristics they have reported are transaction pathlength (in number of machine instructions), average number of locks and unlocks per transaction, average I/O requests per transaction and average I/O service time. They obtained an average probability of contention per lock request using one run of trace-driven simulation. For their concurrency control analysis, they used this value of lock contention probability to model concurrency control requests. All transactions in their model consist of identical number of locks and unlocks— the values are the average

[4]Each interval between two adjacent lock requests constitutes a transaction step.

values measured from the trace (15 locks and 9 unlocks per transaction). The resource requirements are also identical for each transaction. Lock contention probability is assumed to be proportional to the MPL, and the measured value from the trace-driven simulation is used as the base value. With this probability, and independent of any other event, each lock request conflicts. In case of a lock conflict, the time for the lock request wait is modeled as a random variable with mean one third of the transaction time.

In response to the results in [33], [1] have performed a study which investigates the implications of the infinite resource versus finite resources assumption for concurrency control performance analysis. Most of the other assumptions made in this model are similar to those made in [9]. They have shown that the infinite resource assumption biases the results in favor of locking with restarts because restarts do not incur any penalty. [1] began their analysis with a database of 10000 objects, each transaction acquiring an average of 8 objects, with a write-probability of 0.25. They discovered that these parameters cause little contention in the system and all algorithms performed equally well. So to study interesting performance effects, they reduce the database size to 1000. This is, in our opinion, an artificial and unrealistic way of analyzing contention. If there is little contention in the databases that exist today, we might expect more contention in the systems of the future not because they will be running smaller databases, but because more users will be running off the same database. Thus, [1] could have moved into a region of higher contention by increasing MPL, instead of reducing the database size.

The common theme that emerges from studying the literature in concurrency control performance is that all the models assume independence between all kinds of events. For example, write probability is independent of the object, transaction stepsize is independent of transaction length, etc. They also implicitly assume that each lock request is independent of all other requests. In real systems, we do not expect this independence. Also, in these models, each data object is equally likely to read locked as well as write locked. Clearly, in most transaction processing environments, this would not be the case. As we will elaborate in Sections 5 and 6, such assumptions will affect the system performance. Also, almost always, all transactions are assumed to be comprised of same number of lock requests. Recently, [32] has showed that database contention is very sensitive to the second moment of the distribution of transaction lengths. This makes it very important to realistically model transaction lengths.

Another aspect of concurrency control modeling that is lacking in the literature is modeling of index locks. Only a few studies have dealt with the issue of index contention [31]. However, we do not know of any study that has combined both index and data contention in the same framework and and has analyzed the relative effects of both on database system performance.

Also, unlocks have never been part of past models. The main reason for this is that if we allow unlocks, the database system loses the guarantee of serializability of transaction under some conditions [13]. However, real systems allow transactions to release locks before Commit time. There may be two reasons for this. The first reason is that there may be certain restrictions on the order of accessing objects which would ensure serializability even if certain locks are unlocked. This is the case for B-tree locks [2]. However, we suspect that the real reason is performance. Real systems are not willing to sacrifice performance to get serializability by disallowing unlocks. As we will report later, unlocks are quite prevalent in all the database system traces we have.

C Data for access distribution

In Section 5.1 we had shown plots for the distribution of locks over the database items. Here, in Table 7, we present the raw data values, which should be useful for a realistic modeling of the database.

Fraction of Items	*Fraction of locks*						
	Data			*Index*			
	Cursor	*Read*	*Write*	*Cursor (non-Root)*	*Cursor*	*Read*	*Write*
Transport:							
0.05	0.539345	0.950804	0.326694	0.373802	0.705997	0.927370	0.288912
0.10	0.684943	0.968345	0.490514	0.512392	0.771654	0.951423	0.413586
0.15	0.768534	0.974135	0.594444	0.604884	0.816688	0.962102	0.495314
0.20	0.806878	0.977492	0.661654	0.667438	0.846452	0.968018	0.548151
0.25	0.841327	0.979959	0.703929	0.725832	0.872925	0.972261	0.598125
0.30	0.872708	0.981295	0.733469	0.767821	0.893126	0.975531	0.648099
0.35	0.895795	0.982631	0.761653	0.806751	0.910775	0.978745	0.675169
0.40	0.913806	0.983967	0.789837	0.836391	0.924784	0.980380	0.700156
0.45	0.927586	0.985303	0.818022	0.861631	0.936805	0.982015	0.725143
0.50	0.938133	0.986639	0.846206	0.881095	0.945629	0.983650	0.750130
0.60	0.955620	0.989311	0.887263	0.920024	0.963278	0.986920	0.800104
0.70	0.969400	0.991983	0.915447	0.941605	0.973527	0.990190	0.850078
0.80	0.983179	0.994656	0.943631	0.961070	0.982351	0.993460	0.900052
0.90	0.993110	0.997328	0.971816	0.980535	0.991176	0.996730	0.950026
Phone:							
0.05	0.382851	0.411543	0.515985	0.288761	0.293663	0.347425	0.391575
0.10	0.519776	0.513603	0.639716	0.492132	0.495674	0.456607	0.490641
0.15	0.618753	0.593675	0.711822	0.574232	0.577289	0.539750	0.560599
0.20	0.705797	0.654285	0.761070	0.631939	0.634592	0.604295	0.611899
0.25	0.762856	0.701508	0.799556	0.678457	0.680770	0.655994	0.653366
0.30	0.804876	0.744673	0.829120	0.724974	0.726948	0.695552	0.694832
0.35	0.837691	0.774332	0.853341	0.767416	0.769095	0.735110	0.730467
0.40	0.864025	0.803992	0.874998	0.802305	0.803728	0.762653	0.751201
0.45	0.886794	0.833651	0.891050	0.830043	0.831274	0.782432	0.771934
0.50	0.905626	0.851703	0.905487	0.853302	0.854362	0.802211	0.792667
0.60	0.940777	0.881362	0.934362	0.899820	0.900539	0.841769	0.834134
0.70	0.965047	0.911022	0.956687	0.930223	0.930733	0.881327	0.875600
0.80	0.981168	0.940681	0.971125	0.953482	0.953822	0.920884	0.917067
0.90	0.990584	0.970341	0.985562	0.976741	0.976911	0.960442	0.958533
Bank:							
0.05	0.162266	0.234756	0.117460	0.160649	0.582339	0.539226	0.168033
0.10	0.246440	0.359756	0.215873	0.252638	0.628266	0.653766	0.270492
0.15	0.318035	0.466464	0.292063	0.319479	0.661495	0.713912	0.360656
0.20	0.389629	0.573171	0.368254	0.386320	0.694724	0.759415	0.442623
0.25	0.461224	0.652439	0.444444	0.453161	0.727953	0.795502	0.510246
0.30	0.498838	0.695122	0.511111	0.520002	0.761182	0.825314	0.577869
0.35	0.534635	0.722561	0.561904	0.565532	0.784011	0.846234	0.627049
0.40	0.570433	0.743902	0.612698	0.598952	0.800625	0.866109	0.672131
0.45	0.606230	0.765244	0.663492	0.632373	0.817240	0.885983	0.717213
0.50	0.642027	0.786585	0.714286	0.665794	0.833854	0.900627	0.762295
0.60	0.713622	0.829268	0.796826	0.732635	0.867084	0.920502	0.819672
0.70	0.785216	0.871951	0.847619	0.799476	0.900313	0.940376	0.864754
0.80	0.856811	0.914634	0.898413	0.866317	0.933542	0.960251	0.909836
0.90	0.928405	0.957317	0.949206	0.933159	0.966771	0.980125	0.954918

Table 7: Data for the access skew of lock types over all lock items. The total number of items and locks can be found in Table 2.

Fraction of Transactions	*Transaction length (in seconds)*		
	transport	*phone*	*bank*
0.05	0.001266	0.010391	0.002396
0.10	0.001312	0.022659	0.002518
0.15	0.008728	0.032166	0.002945
0.20	0.020355	0.045502	0.018692
0.25	0.026291	0.063004	0.039780
0.30	0.032471	0.078812	0.065460
0.35	0.043716	0.094543	0.087509
0.40	0.059250	0.110687	0.121399
0.45	0.085236	0.129517	0.150024
0.50	0.110703	0.148956	0.194977
0.55	0.131241	0.174789	0.272888
0.60	0.155930	0.200485	0.320816
0.65	0.187622	0.234955	0.363663
0.70	0.228027	0.285706	0.397659
0.75	0.289963	0.354187	0.433975
0.80	0.385300	0.440506	0.485245
0.85	0.551758	0.618256	0.541443
0.90	0.880707	0.939255	0.653809
0.95	1.778198	1.750076	0.869385
1.00	1525.373337	1778.907669	2226.724289

Table 8: Data for the distributions of transaction lengths for the three traces

D Transaction length modeling

In Section 6.1 we had provided the plots for the transaction length distributions for the three traces. Here, we provide the data for those plots in Table 8. We also fitted exponential and gamma distributions to the transaction length distributions, as shown in Figure 3. The computations for obtaining these fits are described in Appendix H. In table 9, we use the Kolmogorov Goodness of Fit test for continuous data [6] to check for the validity of fitted gamma distributions and the exponentials distribution to the actual distributions. The .95 test static rejects all fits[5]. Figure 3 (in the main body of the paper) gives the actual

[5] While the Method of Maximum Likelihood attempts to minimize a measure of the total distance of the the data-curve over all points (a measure which is imperceivable when viewing the log scale plot), the Kolmogorov Goodness of Fit test only looks at the one location on the x-axis where the distance between the fit and the data-curve is the maximum.

Fitted distribution				*Test statistic*	*.95 test quantile*	*Reject H_0*
Type	*Parameters*	*Mean*	*Std. dev.*			
Transport:						
gamma	$r = 0.21; \lambda = 0.0838$	2.51	5.47	0.281	0.018	Yes
exp	$\lambda = 0.399$	2.51	2.51	0.657	0.018	Yes
Phone:						
gamma	$r = 0.34; \lambda = 0.3268$	1.04	1.78	0.243	0.014	Yes
exp	$\lambda = 0.961$	1.04	1.04	0.465	0.014	Yes
Bank:						
gamma	$r = 0.195; \lambda = 0.0436$	4.48	10.12	0.372	0.035	Yes
exp	$\lambda = 0.223$	4.48	4.48	0.843	0.035	Yes

Table 9: Goodness of Fitness tests for distribution of transaction lengths

Fraction of Locks	*Locktime (in seconds)*					
	transport		*phone*		*bank*	
	Data	*Index*	*Data*	*Index*	*Data*	*Index*
0.05	0.000275	0.000397	0.000443	0.000214	0.001083	0.000717
0.10	0.000305	0.000427	0.000565	0.000229	0.001480	0.000763
0.15	0.000351	0.000458	0.000671	0.000259	0.002228	0.000900
0.20	0.000443	0.000488	0.000763	0.000305	0.012909	0.001694
0.25	0.000748	0.000504	0.000854	0.000336	0.024521	0.006790
0.30	0.001328	0.000534	0.000961	0.000381	0.029892	0.010056
0.35	0.001617	0.000549	0.001083	0.000427	0.033752	0.013138
0.40	0.001953	0.000580	0.001205	0.000488	0.037018	0.016266
0.45	0.003387	0.000778	0.001343	0.000565	0.040421	0.019028
0.50	0.005707	0.001038	0.001511	0.000641	0.043915	0.023575
0.55	0.009247	0.001221	0.001678	0.000763	0.047424	0.030411
0.60	0.011993	0.001648	0.001938	0.000946	0.050766	0.036530
0.65	0.017609	0.003296	0.002289	0.001175	0.054718	0.041672
0.70	0.024002	0.008682	0.002930	0.001495	0.060150	0.046478
0.75	0.028702	0.020538	0.004501	0.001984	0.067520	0.051559
0.80	0.033661	0.033096	0.008987	0.003036	0.080139	0.057587
0.85	0.038742	0.048767	0.016022	0.006027	0.102722	0.066422
0.90	0.050430	0.084351	0.027328	0.016632	0.161423	0.080963
0.95	0.077286	0.205750	0.053513	0.045013	0.286728	0.118332
1.00	61.118195	15.796967	6.016190	9.511536	31.514969	76.120483

Table 10: Data for the locktime distributions of Cursor locks for the three traces

values of standard deviation for the three traces; from Table 9, we see that exponential and gamma fits provide very inaccurate estimates for the standard deviation.

E Data for locktimes and lockfractions

In Section 6.2, we had plotted the locktime distributions for Cursor locks and the lockfraction distributions for Read/Write locks. In this section we present the corresponding data. Notice that, as explained in Section 6.2, this data for the Cursor locks in Table 10 has been compiled by randomly selecting a fraction of the actual data because of the enormously large number of Cursor locks (greater than a million) in the traces. However, this random selection does not produce any perceptible change in the plots. On the other hand, the data for the Read/Write locks in Table 11 is not a random sampling, but the entire data.

F Correlation tests between transaction lengths and locktimes

In Section 6.2 we had presented data about the locktimes for the Cursor and Read/Write locks for the three traces. Here we present the results of the statistical tests to check for correlation between lock durations and transaction lengths. For Cursor locks, we try to correlate locktimes with transaction lengths; for Read and Write locks we try to correlate lockfractions with transaction lengths.

Table 12 presents the results of these tests. For Cursor locks, there was significant positive correlation between locktimes and transaction lengths (we rejected the null hypothesis of independence at significance level .95 in favor of the alternate hypothesis of positive dependence). Note that even though the correlations in the table might appear to be really low, they are not insignificant because of the large sample size for

Fraction of Locks	*Ratio of Locktime to Transaction Length*					
	transport		*phone*		*bank*	
	Read	*Write*	*Read*	*Write*	*Read*	*Write*
Data:						
0.05	0.229133	0.013514	0.000732	0.125210	0.177928	0.036613
0.10	0.312004	0.048491	0.001217	0.186395	0.327585	0.230273
0.15	0.397360	0.075766	0.001774	0.249538	0.422469	0.279593
0.20	0.460607	0.121369	0.002654	0.307629	0.589007	0.337892
0.25	0.509314	0.167804	0.003895	0.362974	0.677694	0.447921
0.30	0.548788	0.208627	0.005980	0.415690	0.884510	0.493636
0.35	0.582777	0.265886	0.009564	0.474370	0.956210	0.548107
0.40	0.612303	0.334236	0.018358	0.525145	0.966523	0.573371
0.45	0.642499	0.390500	0.049231	0.577949	0.978297	0.632197
0.50	0.674292	0.448303	0.091797	0.624497	0.980401	0.687688
0.55	0.705845	0.499737	0.138100	0.667825	0.988938	0.693358
0.60	0.743220	0.555641	0.199926	0.711302	0.993979	0.766137
0.65	0.786680	0.602874	0.279363	0.748555	0.996598	0.801118
0.70	0.834076	0.659224	0.401885	0.786439	0.997823	0.833532
0.75	0.879183	0.702761	0.597291	0.822217	0.999160	0.850766
0.80	0.921288	0.748362	0.839060	0.857460	0.999406	0.963863
0.85	0.965054	0.802546	0.909088	0.890942	0.999596	0.965887
0.90	0.982705	0.857557	0.940286	0.920730	0.999771	0.969319
0.95	0.994278	0.909777	0.963222	0.949530	0.999911	0.977959
1.00	1.000000	0.997842	0.999987	1.000000	0.999998	0.999547
Index:						
0.05	0.012110	0.000764	0.000241	0.188515	0.007650	0.243959
0.10	0.098706	0.029075	0.000428	0.270481	0.046970	0.302886
0.15	0.191191	0.066370	0.000634	0.336214	0.077086	0.304804
0.20	0.326043	0.099461	0.001041	0.386620	0.303215	0.320005
0.25	0.433085	0.134071	0.001873	0.433621	0.436807	0.335084
0.30	0.506415	0.163101	0.003362	0.469880	0.637409	0.381501
0.35	0.561826	0.201434	0.005920	0.508426	0.659465	0.381857
0.40	0.610525	0.236224	0.010392	0.542723	0.666402	0.401098
0.45	0.646890	0.271726	0.017017	0.581186	0.671036	0.421895
0.50	0.682090	0.308223	0.029854	0.619764	0.676877	0.504118
0.55	0.707434	0.347662	0.062248	0.655453	0.682735	0.606800
0.60	0.733677	0.386467	0.129003	0.684835	0.692437	0.697100
0.65	0.776386	0.430282	0.225642	0.715827	0.704452	0.723899
0.70	0.836414	0.470639	0.372107	0.741900	0.718067	0.785612
0.75	0.903854	0.517287	0.768990	0.769104	0.730682	0.816505
0.80	0.967739	0.577389	0.933110	0.799586	0.745182	0.838876
0.85	0.984258	0.632381	0.968751	0.837032	0.764819	0.856392
0.90	0.992498	0.711871	1.000000	0.883365	0.820866	0.897385
0.95	0.997554	0.798762	1.000000	0.927732	0.981833	0.997217
1.00	1.000000	0.998785	1.000000	1.000000	0.999998	0.999952

Table 11: Data for the lockfraction distributions of Read and Write locks for the three traces

	Lock Type	*Sample size*	*Test statistic $\hat{\rho}$*	*.95 test quantile*	Reject H_0 ?
Transport:					
Data	Cursor	76208	0.3875	0.0071	Yes
	Read	29189	0.0124	0.0115	Yes
	Write	1476	0.3033	0.0511	Yes
Index	Cursor	19053	0.3218	0.0142	Yes
	Read	21162	0.2726	0.0136	Yes
	Write	3842	0.2011	0.0316	Yes
Phone:					
Data	Cursor	76143	0.3875	0.0071	Yes
	Read	4609	−0.5228	0.0289	Yes
	Write	9787	−0.5090	0.0198	Yes
Index	Cursor	19110	0.3218	0.0142	Yes
	Read	7824	−0.5961	0.0222	Yes
	Write	21615	−0.3516	0.0133	Yes
Bank:					
Data	Cursor	27157	0.4513	0.0119	Yes
	Read	82	0.4285	0.2178	Yes
	Write	63	−0.0633	0.2489	No
Index	Cursor	37039	0.0245	0.0102	Yes
	Read	478	−0.2666	0.0897	Yes
	Write	122	−0.4264	0.1782	Yes

Table 12: Spearman test for checking independence between transaction lengths and lock durations (locktimes for Cursor locks and lockfractions for Read/Write locks).

Cursor locks. For Read/Write locks, the table provides the results for the test for the null hypothesis of independence at significance level .95 in favor of the alternative hypothesis of a positive/negative dependence. For all cases (expect Write Data locks for *bank*) we found significant statistical correlation.

G Transaction Mix

Here we present the tables which characterize the transaction mix in our traced database systems. The discussion for these tables is presented in Section 6.3 in the main body of the paper.

Tables 13, 14 and 15 present the characterization of the the three workloads based on the application *plan names*. For each plan type, we tabulate average transaction length, Cursor locktimes, Read locktimes, Write locktimes, number of Cursor locks, number of Read locks, number of Write locks and number of distinct items locked.

Tables 16, 17 and 18 show the characterization of transactions after grouping them by the three parameters discussed in Section 6.3. In these tables, the group id identifies the classification of the transactions along the three axes. The first letter is determined by length of transaction, second letter by read-write behavior and the third letter by two-phaseness. For example, *3w2* comprises of transactions that belong to length class 3, acquire write locks and the locking is two-phase.

Plan Name	Number of xacts	Per transaction statistics								
		Xact	Locktime (locktime per lock)			Number of locks				Distinct
		length	cursor	read	write	cursor	read	write	upgrade	items
IQ303D	3107	1.632	2.344(.0240)	18.18(2.014)	.0000(.0000)	97.45	9.027	.0000	.0000	57.97
QMF220	219	18.67	22.05(.1026)	54.54(110.6)	.0522(.4766)	214.9	.4931	.0958	.0136	82.19
TDCBPL	2	13.92	.0000(.0000)	3081.(11.43)	289.5(7.934)	.0000	269.5	33.50	3.000	303.0
PDCB81	24	92.43	150.9(.0372)	.0000(.0000)	.0000(.0000)	4051.	.0000	.0000	.0000	350.6
PPYBXDBL	47	.6946	.0269(.0362)	1.085(4.639)	6.901(10.13)	.7446	.2340	.6595	.0212	1.468
PDC22	294	.7508	.5341(.0245)	.5069(4.028)	.9584(.6873)	21.76	.1258	1.384	.0102	16.79
PDC12	466	.6366	.3862(.0100)	.1935(9.018)	.3662(.5785)	38.56	.0214	.6330	.0000	15.39
PPYBXLXT	10	29.41	47.14(.0493)	.0000(.0002)	.0000(.0000)	955.8	.2000	.0000	.0000	930.6
ADCBPL	74	.1867	.0000(.0000)	.9762(1.416)	3.861(3.861)	.0000	.6891	.9189	.0810	1.608
TDCB51	1	8.124	.0000(.0000)	70.07(2.695)	270.2(6.284)	.0000	26.00	40.00	3.000	66.00
DSNTEP13	13	4.893	5.481(.0043)	10.95(10.95)	1.313(.9489)	1267.	1.000	1.230	.1538	847.3
PDC81	20	3.516	12.55(.0148)	.0000(.0000)	.0000(.0000)	846.4	.0000	.0000	.0000	263.7
PDC15	19	1.043	.5211(.0049)	2.046(1.296)	4.830(2.353)	105.5	1.578	2.052	.0000	108.7
TDC4A	21	50.43	.0968(.0423)	.6110(.9871)	8.962(3.764)	2.285	.6190	2.142	.2380	4.285
	103	.1853	.0000(.0000)	.9971(1.252)	1.263(2.829)	.0000	.7961	.4466	.0000	1.242
PDC31	422	.2647	.1599(.0264)	.0918(.2849)	.0255(.1146)	6.054	.3222	.2180	.0047	3.924
PDCBSM1	3	33.65	43.77(.0373)	.0000(.0000)	.0000(.0000)	1172.	.0000	.0000	.0000	799.0
ADB0011	14	1.580	.0065(.0459)	.8167(.4971)	3.797(1.772)	.1428	1.642	1.714	.4285	3.500
PDC25	584	.1135	.0543(.0086)	.0223(1.304)	.0399(.5697)	6.260	.0171	.0684	.0017	3.104
DSNTIA13	6	3.405	2.866(.0031)	17.96(6.736)	.0366(.0733)	915.8	2.666	.5000	.0000	919.0
PTOLINVO	6	4.610	.0068(.0058)	.0000(.0000)	4.166(2.777)	1.166	.0000	1.500	.0000	2.666
PTM10	198	.1219	.0062(.0194)	.0000(.0000)	.1245(.3081)	.3232	.0000	.4040	.0000	.6616
TTOLMAIN	38	23.89	.2687(.0391)	.0173(.0730)	.3703(.5026)	6.868	.2368	.7368	.0000	4.526
ATOST01	55	.1572	.0000(.0000)	.0744(.5847)	.0548(.1588)	.0000	.1272	.2909	.0545	.4181
PDC71	11	1.387	2.188(.0074)	.0000(.0000)	.0000(.0000)	294.3	.0000	.0000	.0000	294.3
PDC52	2	8.251	10.77(.0058)	.0068(.0068)	.0501(.0501)	1834.	1.000	1.000	.0000	580.5
PDCB81A	1	9.140	13.15(.0170)	.0000(.0000)	.0000(.0000)	772.0	.0000	.0000	.0000	646.0
PDC00	59	.1987	.3043(.0229)	.0000(.0000)	.0000(.0000)	13.28	.0000	.0000	.0000	13.28
ADC12	7	1.139	.4201(.0127)	.4155(2.908)	.0000(.0000)	32.85	.1428	.0000	.0000	10.28
ADC31	10	.6702	.3330(.0723)	.1534(1.534)	.0000(.0000)	4.600	.1000	.0000	.0000	3.500
PTOLMAIN	26	4.794	.1592(.0242)	.8076(.0002)	.0041(.0083)	6.576	.0384	.5000	.0000	5.923
PDC72	6	.3093	.6250(.0914)	.0000(.0000)	.0000(.0000)	6.833	.0000	.0000	.0000	6.333
TTOLALST	14	.1659	.0117(.0274)	.0000(.0000)	.1150(.1073)	.4285	.0000	1.071	.0000	1.357
TTOLLKPN	14	.7486	.0150(.0123)	.1119(.3133)	.1274(.1784)	1.214	.3571	.3571	.3571	1.428
ATOTST01	39	.0443	.0000(.0000)	.0102(.4005)	.0143(.1863)	.0000	.0256	.0769	.0000	.1025
TTOLINVO	6	.5680	.0288(.0157)	.0000(.0000)	.2086(1.252)	1.833	.0000	.1666	.0000	1.833
DSNESPCS	1	.4164	.1610(.1610)	.3959(.3959)	.0000(.0000)	1.000	1.000	.0000	.0000	2.000
TDC48	6	.2521	.1072(.0536)	.0001(.0009)	.0666(.0002)	2.000	.1666	.1666	.0000	2.333
TDC56	7	.3967	.0701(.0288)	.0000(.0000)	.5714(.0003)	2.428	.0000	.1428	.0000	2.000
PTOLALST	18	.0684	.0000(.0000)	.0000(.0000)	.0214(.0227)	.0000	.0000	.9444	.0000	.9444
PDC11	1	1.274	1.014(.0098)	.0000(.0000)	.0000(.0000)	103.0	.0000	.0000	.0000	53.00
PDC13	3	.4559	.1335(.0500)	.0265(.0796)	.0000(.0000)	2.666	.3333	.0000	.0000	3.000
TDC84	1	.4117	.6958(.0773)	.0000(.0000)	.0000(.0000)	9.000	.0000	.0000	.0000	8.000
PDC41	1	.6479	1.172(.0732)	.0000(.0000)	.0000(.0000)	16.00	.0000	.0000	.0000	16.00
ADC00	1	1.553	.7200(.0800)	.0000(.0000)	.0000(.0000)	9.000	.0000	.0000	.0000	9.000
PTOLLKPN	6	.3921	.0067(.0067)	.0283(.0567)	.0494(.0593)	1.000	.5000	.5000	.3333	1.500
TDCB84A	1	.3060	.2973(.0495)	.0000(.0000)	.0000(.0000)	6.000	.0000	.0000	.0000	6.000
TTOLLDEL	1	.7389	.1305(.0261)	.0000(.0000)	.0000(.0000)	5.000	.0000	.0000	.0000	4.000
TTOLGREX	1	.6781	.1710(.0342)	.0000(.0000)	.0000(.0000)	5.000	.0000	.0000	.0000	5.000
TDC00	4	.0514	.0317(.0317)	.0000(.0000)	.0000(.0000)	1.000	.0000	.0000	.0000	1.000
PDBB0020	2	.0551	.0424(.0424)	.0000(.0000)	.0000(.0000)	1.000	.0000	.0000	.0000	1.000

Table 13a. Data lock statistics for all plan types in "transport". The plans are sorted by the total locktime (data+index) for all transactions of the plan.

Plan Name	Number of xacts	Per transaction statistics								
		Xact	Locktime (locktime per lock)			Number of locks				Distinct
		length	cursor	read	write	cursor	read	write	upgrade	items
IQ303D	3107	1.632	1.200(.0911)	7.542(1.356)	.0000(.0000)	13.17	5.561	.0000	.0000	16.78
QMF220	219	18.67	13.22(.7335)	73.79(36.07)	.0098(.4297)	18.03	2.045	.0091	.0136	15.92
TDCBPL	2	13.92	1.613(.0077)	641.8(8.977)	137.1(10.55)	208.5	71.50	5.000	8.000	88.00
PDCB81	24	92.43	100.6(.0612)	.0353(.0292)	.0000(.0000)	1644.	1.208	.0000	.0000	172.0
PPYBXDBL	47	.6946	.1749(.0019)	1.104(.3223)	24.83(12.41)	88.72	3.425	.4893	1.510	5.425
PDC22	294	.7508	.6426(.0251)	.4788(.6672)	2.290(.5365)	25.52	.7176	4.074	.1938	24.96
PDC12	466	.6366	.1557(.0148)	.0588(.0556)	1.306(.4384)	10.49	1.057	2.881	.0965	9.778
PPYBXLXT	10	29.41	29.64(2.823)	.0556(.0327)	.0000(.0000)	10.50	1.700	.0000	.0000	11.30
ADCBPL	74	.1867	.0635(.0084)	1.197(.4843)	1.199(4.439)	7.554	2.472	.1081	.1621	3.851
TDCB51	1	8.124	2.406(.0051)	71.93(2.997)	72.63(6.052)	471.0	24.00	4.000	8.000	39.00
DSNTEP13	13	4.893	.1127(.0325)	16.06(6.962)	.0014(.0186)	3.461	2.307	.0000	.0769	4.461
PDC81	20	3.516	7.198(.9170)	.0051(.0093)	.0000(.0000)	7.850	.5500	.0000	.0000	5.300
PDC15	19	1.043	.4512(.0223)	3.842(.8295)	8.841(3.359)	20.21	4.631	.9473	1.684	17.78
TDC4A	21	50.43	.3327(.0100)	4.147(2.073)	3.346(3.698)	33.14	2.000	.4285	.4761	12.33
	103	.1853	.0530(.0135)	.5457(.2531)	.1431(1.340)	3.922	2.155	.0485	.0582	3.436
PDC31	422	.2647	.3018(.0164)	.0076(.0163)	.0926(.0946)	18.31	.4644	.9099	.0687	9.755
PDCBSM1	3	33.65	45.22(.0145)	.0000(.0000)	.0000(.0000)	3099.	.0000	.0000	.0000	276.6
ADB0011	14	1.580	.8001(.0476)	8.268(1.134)	4.047(1.231)	16.78	7.285	1.714	1.571	14.21
PDC25	584	.1135	.0867(.0062)	.0192(.0124)	.1729(.4613)	13.92	1.547	.3390	.0359	8.936
DSNTIA13	6	3.405	.3469(.0800)	12.42(4.658)	.0062(.0093)	4.333	2.666	.6666	.0000	6.333
PTOLINVO	6	4.610	.1191(.0029)	.2242(.2242)	14.25(2.949)	40.33	1.000	4.000	.8333	9.000
PTM10	198	.1219	.0842(.0210)	.0085(.0068)	.2546(.2639)	4.010	1.247	.9646	.0000	4.611
TTOLMAIN	38	23.89	.2444(.0171)	.2915(.2915)	.3548(.4494)	14.26	1.000	.1578	.6315	8.763
ATOST01	55	.1572	.1280(.0426)	.3282(.1388)	.0214(.1962)	3.000	2.363	.0000	.1090	4.036
PDC71	11	1.387	.6145(.0386)	.0000(.0000)	.0000(.0000)	15.90	.0000	.0000	.0000	6.454
PDC52	2	8.251	3.261(.0051)	.0000(.0000)	.0000(.0000)	638.5	.0000	.0000	.0000	24.50
PDCB81A	1	9.140	10.59(.0173)	.0000(.0000)	.0000(.0000)	612.0	.0000	.0000	.0000	48.00
PDC00	59	.1987	.0101(.0352)	.0025(.0087)	.0000(.0000)	.2881	.2881	.0000	.0000	.4576
ADC12	7	1.139	.7033(.0349)	.3543(2.480)	.3503(2.452)	20.14	.1428	.1428	.0000	10.85
ADC31	10	.6702	.6179(.0479)	.0000(.0000)	.0000(.0000)	12.90	.0000	.0000	.0000	5.400
PTOLMAIN	26	4.794	.1115(.0115)	.0373(.0485)	.0039(.0148)	9.692	.7692	.2692	.0000	5.115
PDC72	6	.3093	.7382(.0726)	.0000(.0000)	.0000(.0000)	10.16	.0000	.0000	.0000	6.666
TTOLALST	14	.1659	.1910(.0405)	.0520(.0607)	.0088(.0176)	4.714	.8571	.0000	.5000	4.500
TTOLLKPN	14	.7486	.0725(.0122)	.0063(.0126)	.0000(.0000)	5.928	.5000	.0000	.0000	3.214
ATOTST01	39	.0443	.0363(.0175)	.0382(.0196)	.0134(.1306)	2.076	1.948	.0769	.0256	3.128
TTOLINVO	6	.5680	.0878(.0142)	.0000(.0000)	.2180(1.308)	6.166	.0000	.1666	.0000	4.833
DSNESPCS	1	.4164	.3735(.0287)	1.237(.3094)	.0000(.0000)	13.00	4.000	.0000	.0000	16.00
TDC48	6	.2521	.2081(.0192)	.0001(.0011)	.0097(.0292)	10.83	.1666	.1666	.1666	10.00
TDC56	7	.3967	.1406(.0172)	.0000(.0000)	.0283(.1981)	8.142	.0000	.1428	.0000	6.571
PTOLALST	18	.0684	.0685(.0352)	.0000(.0001)	.0000(.0000)	1.944	.0555	.0000	.0000	2.000
PDC11	1	1.274	.5469(.0176)	.0000(.0000)	.0000(.0000)	31.00	.0000	.0000	.0000	11.00
PDC13	3	.4559	.2707(.0477)	.0397(.1191)	.0000(.0000)	5.666	.3333	.0000	.0000	5.333
TDC84	1	.4117	.6656(.0443)	.0000(.0000)	.0000(.0000)	15.00	.0000	.0000	.0000	9.000
PDC41	1	.6479	.0000(.0000)	.0000(.0000)	.0000(.0000)	.0000	.0000	.0000	.0000	.0000
ADC00	1	1.553	.1761(.0293)	.0000(.0000)	.0000(.0000)	6.000	.0000	.0000	.0000	2.000
PTOLLKPN	6	.3921	.0610(.0087)	.0000(.0000)	.0000(.0000)	7.000	.0000	.0000	.0000	3.000
TDCB84A	1	.3060	.2960(.0211)	.0000(.0000)	.0000(.0000)	14.00	.0000	.0000	.0000	7.000
TTOLLDEL	1	.7389	.2517(.0251)	.0000(.0000)	.0000(.0000)	10.00	.0000	.0000	.0000	4.000
TTOLGREX	1	.6781	.1721(.0172)	.0000(.0000)	.0000(.0000)	10.00	.0000	.0000	.0000	4.000
TDC00	4	.0514	.0527(.0058)	.0000(.0000)	.0000(.0000)	9.000	.0000	.0000	.0000	9.000
PDBB0020	2	.0551	.0623(.0207)	.0000(.0000)	.0000(.0000)	3.000	.0000	.0000	.0000	3.000

Table 13b. Index lock statistics for all plan types in "transport".

Xact Class	Number of xacts	Per transaction statistics								
		Xact	Locktime (locktime per lock)			Number of locks				Distinct
		length	cursor	read	write	cursor	read	write	upgrade	items
TKCE2305	8	185.6	580.9(.0361)	.0368(.0982)	.0000(.0000)	16079	.3750	.0000	.0000	9164.
TKCE2815	1	1778.	4099.(.0449)	29.80(9.934)	.0000(.0000)	91195	3.000	.0000	.0000	17114
TKAA5	1537	.3331	.0478(.0041)	.0307(.0418)	.7548(.2609)	11.52	.7345	2.187	.7052	8.886
TKAU5	800	1.251	2.874(.0231)	.3449(.2650)	.0000(.0000)	124.4	1.301	.0000	.0000	80.99
TKCE3055	4	141.5	428.1(.0330)	.1712(.1712)	.0000(.0000)	12946	1.000	.0000	.0000	7673.
TKCE0055	3	309.2	437.2(.0233)	.0711(.2134)	.0000(.0000)	18733	.3333	.0000	.0000	4661.
TKA25	198	.3820	.0268(.0059)	.0023(.0064)	1.245(.4542)	4.489	.3636	2.378	.3636	6.439
TKCE1255	1	639.9	692.8(.0030)	.0000(.0000)	.0000(.0000)	22530	.0000	.0000	.0000	11245
TKA75	106	.5101	.0696(.0049)	.8867(.0005)	1.965(.4948)	14.14	.0094	3.962	.0094	13.54
TKCIA745	14	1.518	.5161(.0013)	.0000(.0000)	12.57(1.189)	390.1	.0000	10.57	.0000	54.42
TKCE0065	853	.3291	.4295(.0401)	.0217(3.716)	.0000(.0000)	10.70	.0058	.0000	.0000	9.527
TKAC5	971	.2202	.0287(.0116)	.0014(.0028)	.2036(.1071)	2.476	.5077	1.421	.4788	4.402
TKBF5	72	1.846	3.086(.0028)	.0036(.2636)	.0000(.0000)	1097.	.0138	.0000	.0000	1069.
TKAY5	2	70.69	83.58(.0022)	.0000(.0000)	.0000(.0000)	37872	.0000	.0000	.0000	3706.
TKAW5	78	1.465	.9264(.0421)	.0599(.0632)	.0000(.0000)	21.97	.9487	.0000	.0000	19.10
TKA55	904	.2273	.1853(.0120)	.0001(.1338)	.0000(.0000)	15.42	.0011	.0000	.0000	11.20
TKAV5	14	9.957	1.822(.0815)	.1805(.0902)	.0000(.0000)	22.35	2.000	.0000	.0000	22.85
TKCIA755	46	.2083	.0197(.0027)	.2543(.0005)	1.269(.2212)	7.130	.0434	5.695	.0434	8.608
TKAR5	981	.1301	.0969(.0213)	.0237(.0545)	.0000(.0000)	4.539	.4362	.0000	.0000	4.975
TKCIA735	954	.0363	.0000(.0000)	.0341(.0341)	.0289(.0290)	.0000	.9979	.9979	.0000	1.995
TKCR0055	172	.1471	.0000(.0000)	.0009(.0009)	.4890(.1350)	.0000	1.000	2.622	1.000	3.622
TKA35	77	.4096	.0796(.0049)	.0244(.0247)	.3866(.1837)	16.18	.9870	1.155	.9480	12.81
TKSC5	193	.4546	.0169(.0041)	.0048(.2342)	.1432(.1946)	4.119	.0207	.7357	.0000	3.202
TKCIA815	2	.9677	.0799(.0228)	.0003(.0006)	2.830(.7075)	3.500	.5000	3.500	.5000	7.000
TKAH5	56	.2060	.0278(.0105)	.0238(.0371)	.2030(.1306)	2.642	.6428	1.071	.4821	4.035
TKAP5	112	.1173	.0880(.0108)	.0133(.0383)	.0000(.0000)	8.133	.3482	.0000	.0000	8.482
TKG55	33	.5245	.0870(.0159)	.0000(.0000)	.0000(.0000)	5.454	.0000	.0000	.0000	4.424
TKL25	28	.1855	.0086(.0060)	.0000(.0000)	.1587(.2778)	1.428	.0000	.5714	.0000	1.428
TKCR2305	7	4.730	.0472(.0026)	.0209(.0209)	1.508(.3406)	17.85	1.000	3.428	1.000	13.85
TKLA5	59	.1622	.0202(.0072)	.0000(.0000)	.0785(.1932)	2.796	.0000	.4067	.0000	2.644
TKAF5	23	.2355	.0324(.0067)	.0102(.0157)	.1874(.1002)	4.782	.6521	1.391	.4782	6.304
TKN55	4	1.020	.0736(.0184)	.0000(.0000)	.0000(.0000)	4.000	.0000	.0000	.0000	4.000
TKGA5	4	1.466	2.138(.0036)	.0000(.0000)	.0000(.0000)	585.0	.0000	.0000	.0000	585.0
TKL55	1	6.310	.2746(.0915)	.0000(.0000)	.0000(.0000)	3.000	.0000	.0000	.0000	3.000
TK005	97	.1176	.0084(.0083)	.0000(.0000)	.0000(.0000)	1.010	.0000	.0000	.0000	1.010
TKCR3055	3	.4531	.0000(.0000)	.0099(.0099)	1.356(.4069)	.0000	1.000	2.333	1.000	3.333
TKA65	3	.5195	.3680(.0139)	.5201(.2600)	.0000(.0000)	26.33	2.000	.0000	.0000	13.00
TKNU5	1	2.466	4.310(.0615)	.0000(.0000)	.0000(.0000)	70.00	.0000	.0000	.0000	70.00
TKD55	26	.1450	.0094(.0047)	.0000(.0000)	.0000(.0000)	2.000	.0000	.0000	.0000	2.000
TKCIA835	1	.7242	.0020(.0010)	.0425(.0425)	1.485(.4950)	2.000	1.000	2.000	1.000	5.000
TKL45	4	.2088	.0939(.0129)	.0000(.0000)	.1366(.2733)	7.250	.0000	.5000	.0000	5.500
TKCBZ055	4	.6271	.4259(.0013)	.0000(.0000)	.0000(.0000)	316.0	.0000	.0000	.0000	316.0
TKG65	6	.1378	.0707(.0223)	.0699(.0699)	.0000(.0000)	3.166	1.000	.0000	.0000	4.166
TKR55	202	2.091	.0021(.0010)	.0000(.0000)	.0000(.0000)	2.044	.0000	.0000	.0000	2.019
TKH25	6	.1439	.0124(.0124)	.0000(.0000)	.0814(.1629)	1.000	.0000	.5000	.0000	1.500
TKLY5	1	.5571	.4004(.0222)	.1342(.1342)	.0000(.0000)	18.00	1.000	.0000	.0000	9.000
TKAG5	2	.1891	.0495(.0247)	.0004(.0008)	.1957(.1304)	2.000	.5000	1.000	.5000	3.500
TKPB5	4	.3534	.0073(.0048)	.0000(.0000)	.0850(.3403)	1.500	.0000	.2500	.0000	1.750
TKAJ5	3	.1225	.0271(.0203)	.0002(.0007)	.1262(.0946)	1.333	.3333	1.000	.3333	2.666
TKP25	6	.1127	.0139(.0167)	.0000(.0000)	.0344(.2067)	.8333	.0000	.1666	.0000	1.000
Other 21 plans	139	.0813	.0039(.0035)	.0015(.1050)	.0062(.1729)	1.130	.0144	.0360	.0000	1.158

Table 14a. Data lock statistics for all plan types in "phone".

Xact Class	Number of xacts	Per transaction statistics								
		Xact	Locktime (locktime per lock)			Number of locks				Distinct
		length	cursor	read	write	cursor	read	write	upgrade	items
TKCE2305	8	185.6	529.5(.0187)	15.68(31.36)	.0000(.0000)	28175	.5000	.0000	.0000	25682
TKCE2815	1	1778.	4189.(.0320)	29.69(9.899)	.0000(.0000)	13057	3.000	.0000	.0000	67366
TKAA5	1537	.3331	.1352(.0079)	.0080(.0064)	2.571(.3734)	16.93	1.247	5.681	1.205	18.36
TKAU5	800	1.251	2.544(.0199)	.2686(.2016)	.0000(.0000)	127.4	1.332	.0000	.0000	113.6
TKCE3055	4	141.5	364.1(.0184)	.1452(.1936)	.0000(.0000)	19712	.7500	.0000	.0000	17960
TKCE0055	3	309.2	460.5(.0154)	92.49(277.4)	.0000(.0000)	29906	.3333	.0000	.0000	19980
TKA25	198	.3820	.0365(.0037)	.0219(.0214)	7.682(.5063)	9.747	1.025	14.18	.9898	23.37
TKCE1255	1	639.9	505.3(.0065)	.0000(.0000)	.0000(.0000)	77217	.0000	.0000	.0000	13316
TKA75	106	.5101	.1513(.0042)	.0097(.0034)	8.812(.5129)	35.86	2.849	14.33	2.839	44.57
TKCIA745	14	1.518	.5916(.0009)	.4014(.0419)	61.50(1.373)	600.1	9.571	35.42	9.357	93.07
TKCE0065	853	.3291	.6084(.0170)	.0215(6.119)	.0000(.0000)	35.73	.0035	.0000	.0000	31.59
TKAC5	971	.2202	.0884(.0112)	.0024(.0031)	.2917(.1254)	7.866	.7929	1.562	.7641	9.605
TKBF5	72	1.846	2.272(.0034)	.0000(.0000)	.0000(.0000)	650.6	.0000	.0000	.0000	650.6
TKAY5	2	70.69	95.92(.0004)	.0000(.0000)	.0000(.0000)	21626	.0000	.0000	.0000	21629
TKAW5	78	1.465	2.277(.0025)	.8956(.3776)	.0000(.0000)	882.6	2.371	.0000	.0000	883.7
TKA55	904	.2273	.1444(.0114)	.0000(.0000)	.0000(.0000)	12.60	.0000	.0000	.0000	10.94
TKAV5	14	9.957	15.75(.0080)	.2098(.1049)	.0000(.0000)	1959.	2.000	.0000	.0000	1958.
TKCIA755	46	.2083	.0449(.0039)	.0053(.0011)	4.095(.1498)	11.41	4.695	22.65	4.673	35.69
TKAR5	981	.1301	.0729(.0141)	.0245(.0563)	.0000(.0000)	5.149	.4362	.0000	.0000	5.586
TKCIA735	954	.0363	.0002(.0027)	.0707(.0340)	.0384(.0325)	.0880	2.077	1.099	.0817	3.244
TKCR0055	172	.1471	.0005(.0005)	.0060(.0029)	.2757(.1343)	1.000	2.052	.0000	2.052	3.052
TKA35	77	.4096	.1905(.0066)	.0009(.0015)	.3654(.2024)	28.85	.6493	1.155	.6493	20.50
TKSC5	193	.4546	.0196(.0032)	.0047(.0455)	.1724(.1993)	6.036	.1036	.7823	.0829	4.756
TKCIA815	2	.9677	.0950(.0061)	.0001(.0001)	20.93(.8210)	15.50	.5000	25.00	.5000	37.50
TKAH5	56	.2060	.0684(.0080)	.0252(.0344)	.1387(.1317)	8.553	.7321	.4821	.5714	8.000
TKAP5	112	.1173	.0594(.0122)	.0210(.0605)	.0000(.0000)	4.848	.3482	.0000	.0000	5.196
TKG55	33	.5245	.4721(.0049)	.0000(.0000)	.0000(.0000)	94.66	.0000	.0000	.0000	48.69
TKL25	28	.1855	.0114(.0031)	.0037(.0117)	.4466(.2605)	3.642	.3214	1.428	.2857	4.535
TKCR2305	7	4.730	.0679(.0036)	.0399(.0174)	.8304(.3633)	18.85	2.285	.0000	2.285	7.285
TKLA5	59	.1622	.0250(.0036)	.0000(.0000)	.1185(.1943)	6.813	.0000	.6101	.0000	5.593
TKAF5	23	.2355	.1132(.0121)	.0149(.0181)	.1262(.1001)	9.347	.8260	.6086	.6521	8.869
TKN55	4	1.020	2.124(.0153)	.0000(.0000)	.0000(.0000)	138.5	.0000	.0000	.0000	118.7
TKGA5	4	1.466	.0017(.0008)	.0000(.0000)	.0000(.0000)	2.000	.0000	.0000	.0000	2.000
TKL55	1	6.310	7.033(.0036)	.0000(.0000)	.0000(.0000)	1901.	.0000	.0000	.0000	952.0
TK005	97	.1176	.0646(.0022)	.0000(.0000)	.0000(.0000)	28.85	.0000	.0000	.0000	28.85
TKCR3055	3	.4531	.0004(.0004)	.0390(.0195)	.8154(.4077)	1.000	2.000	.0000	2.000	3.000
TKA65	3	.5195	.3383(.0116)	.5217(.2608)	.0000(.0000)	29.00	2.000	.0000	.0000	12.66
TKNU5	1	2.466	.0012(.0006)	.0000(.0000)	.0000(.0000)	2.000	.0000	.0000	.0000	2.000
TKD55	26	.1450	.1037(.0051)	.0000(.0000)	.0000(.0000)	20.15	.0000	.0000	.0000	10.57
TKCIA835	1	.7242	.0692(.0062)	.0503(.0251)	.8431(.4215)	11.00	2.000	.0000	2.000	12.00
TKL45	4	.2088	.0831(.0095)	.0410(.0820)	.1320(.1320)	8.750	.5000	.5000	.5000	5.250
TKCBZ055	4	.6271	.0002(.0002)	.0000(.0000)	.0000(.0000)	1.000	.0000	.0000	.0000	1.000
TKG65	6	.1378	.0092(.0036)	.1190(.1190)	.0000(.0000)	2.500	1.000	.0000	.0000	3.500
TKR55	202	2.091	.0056(.0018)	.0000(.0000)	.0000(.0000)	3.049	.0000	.0000	.0000	3.024
TKH25	6	.1439	.0138(.0046)	.0001(.0003)	.1387(.1189)	3.000	.1666	1.000	.1666	3.666
TKLY5	1	.5571	.2168(.0065)	.4295(.4295)	.0000(.0000)	33.00	1.000	.0000	.0000	20.00
TKAG5	2	.1891	.0671(.0111)	.0012(.0025)	.1443(.1443)	6.000	.5000	.5000	.5000	6.500
TKPB5	4	.3534	.0095(.0029)	.0000(.0000)	.1153(.2306)	3.250	.0000	.5000	.0000	3.750
TKAJ5	3	.1225	.0024(.0012)	.0006(.0009)	.1225(.0919)	2.000	.6666	.6666	.6666	3.333
TKP25	6	.1127	.0157(.0062)	.0000(.0000)	.0655(.1967)	2.500	.0000	.3333	.0000	2.833
Other 21 plans	139	.0813	.0091(.0025)	.0018(.0620)	.0082(.1433)	3.640	.0288	.0432	.0144	3.669

Table 14b. Index lock statistics for all plan types in "phone".

Xact Class	Number of xacts	Per transaction statistics								
		Xact	Locktime (locktime per lock)			Number of locks				Distinct
		length	cursor	read	write	cursor	read	write	upgrade	items
CIFM1001	162	26.55	53.05(.0885)	41.72(307.2)	.0000(.0000)	599.2	.1358	.0000	.0000	526.6
QMF220	31	54.13	33.81(.0407)	305.8(206.1)	.0815(.1487)	829.7	1.483	.5483	.0000	644.2
FBSPLAN	130	.9426	.0831(.0101)	.0000(.0000)	1.270(3.589)	8.161	.0000	.3538	.0000	4.130
CIFOI001	386	.3533	.3429(.1082)	.0765(14.77)	.0000(.0000)	3.168	.0051	.0000	.0000	2.772
CIFOI003	313	.3882	.4478(.0620)	.0000(.0000)	.0000(.0000)	7.214	.0000	.0000	.0000	4.498
	190	.2633	.0438(.0196)	.4719(22.41)	.0000(.0000)	2.231	.0210	.0000	.0000	.8421
CIFOI004	127	.2915	.2297(.1076)	.0000(.0000)	.0000(.0000)	2.133	.0000	.0000	.0000	1.866
CCCPLAN	20	2.746	1.439(.0177)	.0000(.0000)	.0000(.0000)	81.05	.0000	.0000	.0000	56.75
CIFOI007	73	.3456	.3836(.1081)	.0000(.0000)	.0000(.0000)	3.547	.0000	.0000	.0000	3.041
CIFOI005	20	.2503	.1944(.0720)	.0407(.1018)	.0000(.0000)	2.700	.4000	.0000	.0000	2.750
CCASDB24	1	19.43	10.95(.0027)	.0000(.0000)	.0000(.0000)	4020.	.0000	.0000	.0000	1709.
CIFOI006	5	.5720	.2485(.0355)	.0000(.0000)	.0000(.0000)	7.000	.0000	.0000	.0000	3.400
CIFOI002	7	.2536	.1687(.0562)	.0000(.0000)	.0000(.0000)	3.000	.0000	.0000	.0000	3.000

Table 15a. Data lock statistics for all plan types in "bank".

Xact Class	Number of xacts	Per transaction statistics								
		Xact	Locktime (locktime per lock)			Number of locks				Distinct
		length	cursor	read	write	cursor	read	write	upgrade	items
CIFM1001	162	26.55	58.89(.0579)	22.66(18.82)	.0000(.0000)	1016.	1.203	.0000	.0000	388.8
QMF220	31	54.13	29.48(.1773)	145.8(90.44)	.0645(.1818)	166.2	1.612	.2903	.0645	115.3
FBSPLAN	130	.9426	.1474(.0083)	.9737(5.754)	2.682(3.141)	17.63	.1692	.7615	.0923	11.71
CIFOI001	386	.3533	.5268(.0506)	.1145(14.73)	.0000(.0000)	10.41	.0077	.0000	.0000	6.911
CIFOI003	313	.3882	.5981(.0427)	.0000(.0000)	.0000(.0000)	13.98	.0000	.0000	.0000	8.469
	190	.2633	.0062(.0061)	.0025(.0024)	.0000(.0000)	1.026	1.052	.0000	.0000	2.052
CIFOI004	127	.2915	.3841(.0437)	.0000(.0000)	.0000(.0000)	8.787	.0000	.0000	.0000	6.598
CCCPLAN	20	2.746	2.425(.0227)	.0000(.0000)	.0000(.0000)	106.4	.0000	.0000	.0000	32.15
CIFOI007	73	.3456	.6021(.0586)	.0000(.0000)	.0000(.0000)	10.27	.0000	.0000	.0000	7.027
CIFOI005	20	.2503	.2912(.0469)	.0410(.1025)	.0000(.0000)	6.200	.4000	.0000	.0000	5.000
CCASDB24	1	19.43	.0436(.0109)	.0000(.0000)	.0000(.0000)	4.000	.0000	.0000	.0000	4.000
CIFOI006	5	.5720	.6351(.0387)	.0000(.0000)	.0000(.0000)	16.40	.0000	.0000	.0000	11.40
CIFOI002	7	.2536	.2797(.0466)	.0000(.0000)	.0000(.0000)	6.000	.0000	.0000	.0000	6.000

Table 15b. Index lock statistics for all plan types in "bank".

Xact Class	Number of xacts	Per transaction statistics								
		Xact	Locktime (locktime per lock)			Number of locks				Distinct
		length	cursor	read	write	cursor	read	write	upgrade	items
Data lock stats:										
0R2	878	.0019	.0000(.0000)	.0006(.0082)	.0000(.0000)	.0000	.0797	.0000	.0000	.0797
1R2	2680	.0782	.0000(.0000)	.4414(.0739)	.0000(.0000)	.0000	5.969	.0000	.0000	5.969
2R2	893	.4951	.0000(.0000)	3.843(.3184)	.0000(.0000)	.0000	12.06	.0000	.0000	12.06
3R2	10	8.317	.0000(.0000)	48.79(4.280)	.0000(.0000)	.0000	11.40	.0000	.0000	11.40
4R2	2	219.6	.0000(.0000)	3281.(218.7)	.0000(.0000)	.0000	15.00	.0000	.0000	15.00
0W2	8	.0020	.0000(.0000)	.0000(.0000)	.0009(.0009)	.0000	.0000	1.000	.0000	1.000
1W2	22	.0884	.0000(.0000)	.0219(.0690)	.0582(.0533)	.0000	.3181	1.000	.0909	1.318
2W2	45	.7714	.0000(.0000)	.7602(.6579)	1.703(.5397)	.0000	1.155	2.911	.2444	4.066
3W2	11	7.104	.0000(.0000)	582.1(9.254)	132.6(5.229)	.0000	62.90	23.36	2.000	86.27
0RN	74	.0053	.0007(.0006)	.0000(.0000)	.0000(.0000)	1.283	.0000	.0000	.0000	1.148
1RN	285	.1001	.0587(.0096)	.0006(.0190)	.0000(.0000)	6.105	.0315	.0000	.0000	4.526
2RN	623	.7185	.6626(.0150)	.6310(1.917)	.0000(.0000)	44.07	.3290	.0000	.0000	27.64
3RN	104	31.45	59.90(.0206)	306.5(34.62)	.0000(.0000)	2901.	8.855	.0000	.0000	1405.
4RN	25	338.6	414.2(.0536)	990.2(260.5)	.0000(.0000)	7718.	3.800	.0000	.0000	2496.
1WN	40	.1082	.0213(.0055)	.0469(.0587)	.0622(.0383)	3.875	.8000	1.600	.0250	5.300
2WN	251	1.057	.1814(.0080)	.2515(.4073)	1.664(.4779)	22.60	.6175	3.438	.0438	11.55
3WN	40	19.28	4.300(.0151)	7.928(10.93)	13.84(6.672)	284.2	.7250	2.050	.0250	175.3
4WN	4	118.9	.4842(.0365)	.0767(.1023)	.0116(.0466)	13.25	.7500	.2500	.0000	7.000
?R?	5574	2.402	3.052(.0325)	12.32(2.434)	.0000(.0000)	94.00	5.062	.0000	.0000	45.60
?W?	421	3.875	.5233(.0128)	16.20(7.031)	5.963(1.701)	40.97	2.304	3.392	.1140	26.88
??2	4549	.2837	.0000(.0000)	3.979(.6526)	.3378(3.392)	.0000	6.097	.0919	.0077	6.189
??N	1446	9.495	11.92(.0318)	39.70(39.62)	.6734(.9513)	374.3	1.002	.6984	.0090	164.1
Index lock stats:										
0R2	878	.0019	.0010(.0005)	.0024(.0011)	.0000(.0000)	2.108	2.119	.0000	.0000	3.293
1R2	2680	.0782	.0257(.0041)	.2659(.0569)	.0000(.0000)	6.147	4.671	.0000	.0000	9.223
2R2	893	.4951	.1452(.0093)	1.993(.3488)	.0000(.0000)	15.59	5.715	.0000	.0000	19.38
3R2	10	8.317	2.456(.0172)	25.93(5.084)	.0000(.0000)	142.1	5.100	.0000	.0000	145.3
4R2	2	219.6	234.2(.1316)	1314.(219.0)	.0000(.0000)	1780.	6.000	.0000	.0000	1779.
0W2	8	.0020	.0022(.0011)	.0000(.0000)	.0000(.0000)	2.000	.0000	.0000	.0000	2.000
1W2	22	.0884	.0762(.0119)	.0343(.0419)	.0583(.0583)	6.409	.8181	.8636	.1363	4.772
2W2	45	.7714	.3689(.0284)	2.189(.7522)	2.038(.3902)	12.95	2.911	4.577	.6444	13.15
3W2	11	7.104	1.720(.0095)	147.4(5.591)	48.86(5.906)	179.9	26.36	3.454	4.818	39.90
0RN	74	.0053	.0032(.0008)	.0000(.0000)	.0000(.0000)	3.837	.0000	.0000	.0000	3.486
1RN	285	.1001	.0821(.0070)	.0004(.0234)	.0000(.0000)	11.62	.0175	.0000	.0000	6.701
2RN	623	.7185	.5052(.0197)	.3198(1.633)	.0000(.0000)	25.63	.1958	.0000	.0000	13.40
3RN	104	31.45	25.38(.1584)	124.7(34.88)	.0000(.0000)	160.2	3.576	.0000	.0000	58.03
4RN	25	338.6	250.1(.1488)	852.6(343.8)	.0000(.0000)	1681.	2.480	.0000	.0000	325.6
1WN	40	.1082	.0676(.0045)	.0322(.0252)	.0481(.0223)	14.77	1.275	1.500	.6500	11.07
2WN	251	1.057	.4976(.0187)	.5944(.3375)	5.433(.4242)	26.54	1.760	12.12	.6852	27.40
3WN	40	19.28	2.719(.0120)	4.511(1.555)	37.40(7.874)	225.6	2.900	2.475	2.275	86.40
4WN	4	118.9	.6006(.0224)	.0321(.0214)	.0010(.0014)	26.75	1.500	.0000	.7500	12.75
?R?	5574	2.402	1.780(.0859)	7.154(1.983)	.0000(.0000)	20.72	3.607	.0000	.0000	13.38
?W?	421	3.875	.6556(.0144)	4.874(1.947)	8.295(.9093)	45.38	2.503	8.227	.8954	28.45
??2	4549	.2837	.1604(.0183)	1.561(.3553)	.1386(1.812)	8.781	4.393	.0578	.0187	11.22
??N	1446	9.495	6.549(.1001)	24.08(29.62)	1.979(.8195)	65.44	.8131	2.213	.2019	24.56

Table 16. Transaction mix for "transport". The first, second and third letters in the class name respectively define the transaction size, read/write and two-phaseness of the transactions. For instance, 3W2 represents write two-phase transactions belonging to size class 2. ?R? and ?W? respectively denote Read and Write transactions. ??2 and ??N respectively denote two-phase and non-two-phase transactions.

Xact Class	Number of xacts	Per transaction statistics								
		Xact	Locktime (locktime per lock)			Number of locks				Distinct
		length	cursor	read	write	cursor	read	write	upgrade	items
Index lock stats:										
0R2	79	.0017	.0005(.0004)	.0000(.0000)	.0000(.0000)	1.215	.0000	.0000	.0000	1.215
1R2	437	.0377	.0014(.0005)	.0000(.0000)	.0000(.0000)	2.457	.0000	.0000	.0000	2.457
2R2	26	.2775	.0072(.0006)	.0000(.0000)	.0000(.0000)	11.57	.0000	.0000	.0000	11.57
3R2	1	18.15	29.19(.0076)	18.15(18.15)	.0000(.0000)	3808.	1.000	.0000	.0000	3809.
0W2	4	.0047	.0000(.0000)	.0084(.0037)	.0028(.0023)	.0000	2.250	1.000	.2500	3.250
1W2	1132	.0455	.0005(.0015)	.0577(.0298)	.0475(.0404)	.3436	1.937	.9001	.2765	3.165
2W2	170	.1767	.0049(.0033)	.0206(.0129)	.2645(.1583)	1.452	1.600	.1647	1.505	3.205
0RN	195	.0026	.0021(.0016)	.0000(.0000)	.0000(.0000)	1.261	.0000	.0000	.0000	1.261
1RN	1697	.0686	.0314(.0041)	.0042(.0363)	.0000(.0000)	7.573	.1172	.0000	.0000	7.134
2RN	2459	.6606	.7780(.0114)	.1012(.1659)	.0000(.0000)	67.66	.6100	.0000	.0000	60.54
3RN	149	7.068	12.23(.0031)	.4306(.4550)	.0000(.0000)	3928.	.9463	.0000	.0000	1268.
4RN	13	405.8	887.0(.0200)	33.31(54.13)	.0000(.0000)	44188	.6153	.0000	.0000	30243
1WN	331	.0990	.0080(.0017)	.0023(.0017)	.2277(.0720)	4.561	1.350	1.815	1.347	7.244
2WN	2125	.3910	.1331(.0066)	.0086(.0062)	3.346(.3855)	19.90	1.387	7.312	1.367	22.08
3WN	8	15.61	5.931(.0312)	.8963(.0682)	85.76(1.369)	189.8	13.12	50.00	12.62	91.37
?R?	5056	1.604	3.036(.0114)	.1526(.4172)	.0000(.0000)	265.9	.3657	.0000	.0000	148.0
?W?	3770	.2839	.0887(.0072)	.0252(.0159)	2.114(.3687)	12.19	1.584	4.666	1.067	14.37
??2	1849	.0668	.0170(.0053)	.0471(.0352)	.0534(.0609)	3.199	1.338	.5684	.3082	5.095
??N	6977	1.298	2.243(.0113)	.1117(.1457)	1.128(.3937)	198.4	.7666	2.370	.4949	113.7
Data lock stats:										
0R2	79	.0017	.0000(.0000)	.0000(.0000)	.0000(.0000)	.0000	.0000	.0000	.0000	.0000
1R2	437	.0377	.0000(.0000)	.0000(.0000)	.0000(.0000)	.0000	.0000	.0000	.0000	.0000
2R2	26	.2775	.0000(.0000)	.0000(.0000)	.0000(.0000)	.0000	.0000	.0000	.0000	.0000
3R2	1	18.15	.0000(.0000)	18.22(9.112)	.0000(.0000)	.0000	2.000	.0000	.0000	2.000
0W2	4	.0047	.0000(.0000)	.0034(.0034)	.0021(.0021)	.0000	1.000	1.000	.0000	2.000
1W2	1132	.0455	.0000(.0000)	.0278(.0279)	.0570(.0467)	.0000	.9964	1.054	.1660	2.051
2W2	170	.1767	.0000(.0000)	.0104(.0104)	.4581(.1632)	.0000	1.000	1.852	.9529	2.852
0RN	195	.0026	.0017(.0010)	.0000(.0000)	.0000(.0000)	1.630	.0000	.0000	.0000	1.630
1RN	1697	.0686	.0314(.0049)	.0045(.0388)	.0000(.0000)	6.421	.1172	.0000	.0000	5.771
2RN	2459	.6606	.8055(.0120)	.1019(.1822)	.0000(.0000)	66.73	.5595	.0000	.0000	54.21
3RN	149	7.068	11.01(.0132)	.3747(.4166)	.0000(.0000)	831.5	.8993	.0000	.0000	288.1
4RN	13	405.8	941.8(.0224)	2.361(3.837)	.0000(.0000)	41973	.6153	.0000	.0000	10829
1WN	331	.0990	.0024(.0013)	.0006(.0008)	.1866(.0712)	1.788	.7613	1.858	.7613	4.145
2WN	2125	.3910	.0585(.0046)	.0037(.0059)	.9008(.2755)	12.72	.6254	2.651	.6174	9.961
3WN	8	15.61	.5643(.0042)	5.258(6.009)	18.46(1.330)	134.1	.8750	13.00	.8750	37.12
?R?	5056	1.604	3.148(.0188)	.0718(.2112)	.0000(.0000)	167.1	.3399	.0000	.0000	64.70
?W?	3770	.2839	.0344(.0045)	.0221(.0288)	.6011(.2315)	7.616	.7665	2.086	.5095	6.804
??2	1849	.0668	.0000(.0000)	.0278(.0395)	.0770(.0764)	.0000	.7052	.8182	.1892	1.523
??N	6977	1.298	2.300(.0183)	.0566(.1195)	.3043(.2680)	125.2	.4736	.9105	.2251	50.16

Table 17. Transaction mix for "phone".

Xact Class	Number of xacts	Per transaction statistics								
		Xact	Locktime (locktime per lock)			Number of locks				Distinct
		length	cursor	read	write	cursor	read	write	upgrade	items
Index lock stats:										
0R2	332	.0059	.0034(.0031)	.0023(.0022)	.0000(.0000)	1.075	1.045	.0000	.0000	2.075
1R2	230	.1043	.0560(.0194)	.0160(.0636)	.0000(.0000)	2.882	.2521	.0000	.0000	2.800
2R2	2	.6589	.1995(.1330)	.1680(.3361)	.0000(.0000)	1.500	.5000	.0000	.0000	2.000
1W2	3	.3407	.1198(.0102)	.0337(.1011)	.6050(.2016)	11.66	.3333	3.000	.0000	10.33
2W2	5	1.365	.0150(.0005)	.0016(.0042)	5.297(1.203)	28.40	.4000	4.000	.4000	9.600
3W2	1	61.36	.0517(.0010)	122.3(61.19)	244.9(61.23)	49.00	2.000	4.000	.0000	15.00
0RN	7	.0300	.0105(.0046)	.0000(.0000)	.0000(.0000)	2.285	.0000	.0000	.0000	2.285
1RN	674	.3096	.4403(.0407)	.0055(.2881)	.0000(.0000)	10.80	.0192	.0000	.0000	7.424
2RN	182	1.098	1.357(.0462)	.7873(.0017)	.0000(.0000)	29.36	.0274	.0000	.0000	14.11
3RN	8	25.22	3.493(.0925)	8.807(7.046)	.0000(.0000)	37.75	1.250	.0000	.0000	15.50
4RN	6	970.2	1737.(.0615)	1359.(429.4)	.0000(.0000)	28206	3.166	.0000	.0000	10997
1WN	1	.5024	.2634(.0042)	.4243(.2121)	1.084(.3614)	62.00	2.000	2.000	1.000	25.00
2WN	14	2.005	.4756(.0045)	.2699(.2099)	5.453(.9088)	104.0	1.285	5.214	.7857	60.42
?R?	1441	4.482	7.641(.0601)	5.717(18.18)	.0000(.0000)	127.1	.3143	.0000	.0000	52.07
?W?	24	4.074	.3087(.0042)	5.278(5.067)	14.61(2.874)	72.70	1.041	4.500	.5833	40.20
??2	573	.1684	.0260(.0119)	.2221(.3097)	.4768(7.806)	2.179	.7172	.0575	.0034	2.497
??N	892	7.243	12.33(.0599)	9.235(122.9)	.0868(.8900)	205.9	.0751	.0840	.0134	83.59
Data lock stats:										
0R2	332	.0059	.0000(.0000)	.6439(.0187)	.0000(.0000)	.0000	.0030	.0000	.0000	.0030
1R2	230	.1043	.0000(.0000)	.0092(.1336)	.0000(.0000)	.0000	.0695	.0000	.0000	.0695
2R2	2	.6589	.0000(.0000)	.1676(.3353)	.0000(.0000)	.0000	.5000	.0000	.0000	.5000
1W2	3	.3407	.0000(.0000)	.0000(.0000)	.6355(.1733)	.0000	.0000	3.666	.0000	3.666
2W2	5	1.365	.0000(.0000)	.0000(.0000)	2.945(1.227)	.0000	.0000	2.400	.0000	2.400
3W2	1	61.36	.0000(.0000)	.0000(.0000)	122.5(61.25)	.0000	.0000	2.000	.0000	2.000
0RN	7	.0300	.0059(.0041)	.0000(.0000)	.0000(.0000)	1.428	.0000	.0000	.0000	1.428
1RN	674	.3096	.2929(.0564)	.0019(.3266)	.0000(.0000)	5.192	.0059	.0000	.0000	3.577
2RN	182	1.098	1.023(.0428)	.2785(5.068)	.0000(.0000)	23.90	.0549	.0000	.0000	14.47
3RN	8	25.22	6.078(.0094)	34.14(15.17)	.0000(.0000)	642.1	2.250	.0000	.0000	288.3
4RN	6	970.2	1600.(.0793)	2672.(501.0)	.0000(.0000)	20171	5.333	.0000	.0000	17400
1WN	1	.5024	.0023(.0011)	.0000(.0000)	1.126(.3754)	2.000	.0000	3.000	.0000	4.000
2WN	14	2.005	.0022(.0015)	.0000(.0000)	1.955(.7822)	1.500	.0000	2.500	.0000	3.357
?R?	1441	4.482	6.966(.0748)	11.35(199.5)	.0000(.0000)	93.01	.0569	.0000	.0000	77.57
?W?	24	4.074	.0014(.0014)	.0000(.0000)	6.985(2.661)	.9583	.0000	2.625	.0000	3.166
??2	573	.1684	.0000(.0000)	.0043(.1385)	.2428(5.566)	.0000	.0314	.0436	.0000	.0750
??N	892	7.243	11.25(.0748)	18.34(255.6)	.0319(.7501)	150.2	.0717	.0426	.0000	125.3

Table 18. Transaction mix for "bank".

H Method of Maximum Likelihood

Here, we present some calculations we had to do to apply the method of maximum likelihood in order to estimate parameters of a fitted distribution. The method is described in detail in [23].

We denote the samples from the data distribution by $x_1, x_2, \ldots, x_n$. We assume that the data is sampled from a random variable X whose probability law depends on some unknown parameters. In this paper, we deal with three probability functions:

- Exponential distribution (parameter is λ):

$$f_X(x) = \lambda e^{-\lambda x}, x > 0$$

- Gamma distribution (parameters are r and λ):

$$f_X(x) = \frac{\lambda^r x^{r-1} e^{-\lambda x}}{,\,(r)}, x > 0$$

- Beta distribution (parameters are α and β):

$$f_X(x) = \frac{,\,(\alpha+\beta)}{,\,(\alpha),\,(\beta)} x^{\alpha-1}(1-x)^{\beta-1}, x > 0$$

The maximum likelihood method says that the unknown parameters should be chosen such that the *likelihood function* is maximized. The likelihood function is defined as

$$L_X(\Theta) = f_X(x_1) f_X(x_2) \cdots f_X(x_n)$$

For exponential probability law, the value of the parameter λ equals $\frac{n}{\sum x_i}$. For gamma probability law, the following equations determine r and λ:

$$\lambda = \frac{nr}{\sum x_i}$$

$$n \ln \lambda + \sum \ln x_i = n\Psi(r),$$

where $\Psi(r) \equiv \frac{1}{\Gamma(r)} \frac{d\Gamma(x)}{dx}|_{x=r}$

The parameters r and λ can be determined by numerical methods or from mathematical tables.

We now apply the method to determine the parameters α and β for the beta probability law.

Since $L_X(\Theta)$ is non-negative, we examine $K(\Theta) = \ln L_X(\Theta)$, the natural logarithm of the likelihood function. Because the natural logarithm is a monotonic increasing function, the value of Θ that maximizes $K(\Theta)$ is identical with the value that maximizes $L_X(\Theta)$.

$$\begin{aligned}
L_X(\alpha,\beta) &= \prod_{i=1}^{n} \frac{,\,(\alpha+\beta)}{,\,(\alpha),\,(\beta)} x_i^{\alpha-1}(1-x_i)^{\beta-1} \\
&= (\frac{,\,(\alpha+\beta)}{,\,(\alpha),\,(\beta)})^n (\prod x_i)^{\alpha-1} (\prod (1-x_i))^{\beta-1} \\
K(\alpha,\beta) &= \ln L_X(\alpha,\beta) \\
&= n \ln ,\,(\alpha+\beta) - n \ln ,\,(\alpha) - n \ln ,\,(\beta) + (\alpha-1)\sum \ln x_i + (\beta-1)\sum \ln(1-x_i)
\end{aligned}$$

Then

$$\begin{aligned}
\frac{\partial K}{\partial \alpha} &= n\Psi(\alpha+\beta) - n\Psi(\alpha) + \sum \ln x_i \\
\frac{\partial K}{\partial \beta} &= n\Psi(\alpha+\beta) - n\Psi(\beta) + \sum \ln x_i
\end{aligned}$$

Setting the partial derivatives to zero, we get

$$\begin{aligned}\Psi(\alpha+\beta)-\Psi(\alpha) &= -\frac{\sum \ln x_i}{n}\\ \Psi(\alpha+\beta)-\Psi(\beta) &= -\frac{\sum \ln(1-x_i)}{n}\end{aligned}$$

These equations determine the value of α and β.