Scaling and Characterizing Database Workloads: Bridging the Gap between Research and Practice

Richard Hankins°*, Trung Diep°, Murali Annavaram°, Brian Hirano[1], Harald Eri[1], Hubert Nueckel[2], and John P. Shen°

°*Microprocessor Research Labs (MRL)*
[2]Software Solutions Group
Intel® Corporation

[1]Server Technologies
Oracle® Corporation

Abstract

On-Line Transaction Processing (OLTP) workloads are crucial benchmarks for the design and analysis of server processors. Typical cached configurations used by researchers to simulate OLTP workloads are orders of magnitude smaller than the fully scaled configurations used by OEM vendors to achieve world-record transaction processing throughput. The objective of this study is to discover the underlying relationships that characterize OLTP performance over a wide range of configurations. To this end, we have derived the "iron law" of database performance. Using our iron law, we show that both the average instructions executed per transaction (IPX) and the average cycles per instruction (CPI) are critical to the transaction-throughput performance. We use an extensive, empirical examination of an Oracle® based commercial OLTP workload on an Intel® Xeon™ multiprocessor system to characterize the scaling behavior of both the IPX and the CPI. We demonstrate that across a wide range of configurations the IPX and CPI behavior follows predictable trends, which can be accurately characterized by simple linear or piece-wise linear approximations. Based on our data, we propose a method for selecting a minimal, representative workload configuration from which behaviors of much larger OLTP configurations can be accurately extrapolated.

1. Introduction

On-Line Transaction Processing (OLTP) is a lucrative market for shared-memory multiprocessors (SMPs) and naturally, OLTP workloads are crucial benchmarks for the design and performance analysis of server processors used in SMPs. Unlike other standard benchmark suites, such as SPEC [20], OLTP workloads are difficult to configure for a number of reasons. First, setting up a physical OLTP system requires the fine-tuning of a myriad of configuration parameters. The setup complexity is further compounded by the inaccessibility of proprietary database source code. Finally, scaled OLTP setups are prohibitively expensive for most researchers; hence, the typical OLTP setups used by researchers in simulations and those used in production can differ by up to three orders of magnitude.

Researchers, using full-system simulators and binary instrumentation tools, can only simulate small-scale OLTP setups, known as *cached setups*. Cached setups significantly scale down the database size so that the working set fits within system memory, resulting in negligible disk I/O. Cached setups, while still difficult to configure, are relatively inexpensive to set up as they use small amount of system memory and only a few disks. These setups allow researchers to focus on the interactions between CPU and the memory subsystem, without worrying about the I/O interactions.

On the other hand, production environments have much larger OLTP setups, known as *scaled setups*. OEM system vendors configure scaled setups to showcase world-record transaction processing throughputs. These setups use a large number of disks, are dominated by disk I/O, and their working sets far exceed the memory capacity. These setups are tuned carefully for optimal performance using each processor's own performance-monitoring counters. These setups are infeasible to simulate; therefore, using these setups to explore new microarchitecture ideas is impractical.

Despite the importance of the OLTP workloads in server processor design, we are not aware of any definitive published study that characterized the workload behavior ranging from cached to scaled setups. Such a characterization is essential to understanding how simulation based design decisions perform in real-world production environments. Therefore, the primary objective of this study is to understand the differences in system behavior between cached and scaled setups, and to discover the underlying relationships that characterize OLTP performance over a wide range of configurations.

Based on one of the most comprehensive experimental analysis of a commercial OLTP workload on a physical IA-32 multiprocessor system, this paper makes several contributions. First, it proposes a simple extension to the well known iron law of processor performance to characterize OLTP performance. Using the *new* iron law

* Richard Hankins is a Ph.D. student at the University of Michigan; this research was performed while he was an intern in Intel-MRL.

Proceedings of the 36th International Symposium on Microarchitecture (MICRO-36 2003)
0-7695-2043-X/03 $17.00 © 2003 IEEE

of database performance we show that transaction processing throughput is primarily determined by both the instructions executed per transaction (IPX) and cycles per instruction (CPI). Second, using detailed performance measurements from performance counters, we show that the contribution of the branch and compute instructions to the overall CPI is nearly unchanged across a wide range of OLTP configurations. Expectedly, L3 cache misses are the single largest bottleneck and contribute nearly 60% to the overall CPI. Unexpectedly, coherence traffic in our small-scale multiprocessor system has minimal impact on performance, indicating that OLTP workloads would scale well on future CMP designs. Third, we demonstrate that across a wide range of configurations the IPX and CPI behavior follows predictable trends, which can be accurately characterized by linear approximation models. We propose a method for selecting a minimal, representative workload configuration from which behaviors of much larger OLTP configurations can be accurately extrapolated.

The rest of this paper is organized as follows. Section 2 briefly describes relevant prior work on OLTP workload analysis. Section 3 describes our OLTP workload, the Oracle Database Benchmark (ODB), and the experimental methodology used in this study. Section 4 presents an analysis of the system-level behavior of ODB across a wide range of configurations. Section 5 relates the system-level behavior to the processor-level behavior. Section 6 characterizes the scaling behavior of ODB using linear approximation models. We conclude in Section 7.

2. Previous OLTP Workload Analysis

Most of the recent microarchitecture research on OLTP workload was based on simulations using cached setups [2][3][5][7][9][18][19]. These studies focused on analyzing the performance impact of architectural features on database workloads. Most of these studies show that database applications have large instruction and data memory footprints, suffer from frequent context switches, and have significant cache miss rates.

Several previous studies [1][8][10][12][21] analyzed the behavior of OLTP workloads by monitoring physical system behavior using embedded performance counters. These studies primarily focused on the characterization of the memory and disk I/O subsystems. Although these studies were based on data measurements from a physical system using a scaled setup, they are limited to one OLTP configuration. In particular, they did not focus on how the behavior varies with changing OLTP configuration. The emphasis of this study is to examine the workload behavior ranging from cached to scaled setups and to establish fundamental workload characteristics that span the entire range.

In [21], they used a scaled OLTP workload and studied the workload behavior by varying the amount of memory and the number of warehouses. Using examples, they demonstrated that the configuration of the OLTP system could impact several key architectural and operating system characteristics, such as the breakdown of user and kernel time, the disk I/O rates, and the cycles per instruction (CPI) of the processors. They concluded that departures from a well-balanced scaled system can adversely affect the workload behavior and can mislead designers down the wrong path. However, their study did not focus on the reasons for the observed changes in system behavior. Our paper differs from their study in that while we also observe changes in the system behavior, our in-depth study shows that the situation is not dire, as the behavior across a wide range of OLTP configurations can be accurately characterized.

3. Experimental Methods

For this research, we use an OLTP workload that uses Oracle® as the underlying database server. We run this workload on a small-scale multiprocessor system consisting of four Intel® Xeon™ processors [15]. The performance counters [14] in the Xeon processors are used to collect our experimental data.

3.1 Oracle Database Benchmark (ODB)

In this study, we use the Oracle Database Benchmark (ODB)[1], which is an OLTP workload derived from the Oracle 9i Release 2 RDBMS, as the underlying database server. ODB simulates an order-entry business where clients execute transactions against a database. ODB database comprises of a collection of *warehouses*, each warehouse supplies data to ten *sales districts*, and each sales district serves three thousand *customers*. Typical transactions include entering and delivering customer orders, recording payments received from customers, checking the status of a previously placed order, and checking inventory levels at a warehouse.

Figure 1 presents a graphical overview of the ODB workload and shows the interactions between various ODB processes. When ODB starts execution, the underlying Oracle database spawns two types of processes: user processes and Oracle processes. A user process executes a database client's application code. An Oracle process can be either a server process that performs the actual database queries on behalf of the user or a background process that performs maintenance tasks. Two background processes of note are the database writer and the log writer. The database writer searches the pool of database blocks that are cached in the main memory and writes modified blocks back to disk. The log writer process records to disk all changes made to the database.

[1] ODB is not a compliant TPC-C Benchmark™, even though there may be similarities in the database schema and the transactions in the workload. Any results presented here should not be interpreted as or compared to any published TPC-C Benchmark results. TPC-C Benchmark is a trademark of Transaction Processing Performance Council (TPC).

All Oracle processes share a large, shared-memory segment called the System Global Area (SGA). SGA is a region of shared memory that is allocated when an Oracle database is started. SGA is shared by the server processes as well as the background processes. The largest area in SGA is devoted to the *database buffer cache*, which tracks the usage of the database blocks to keep the most recently and frequently used blocks in memory.

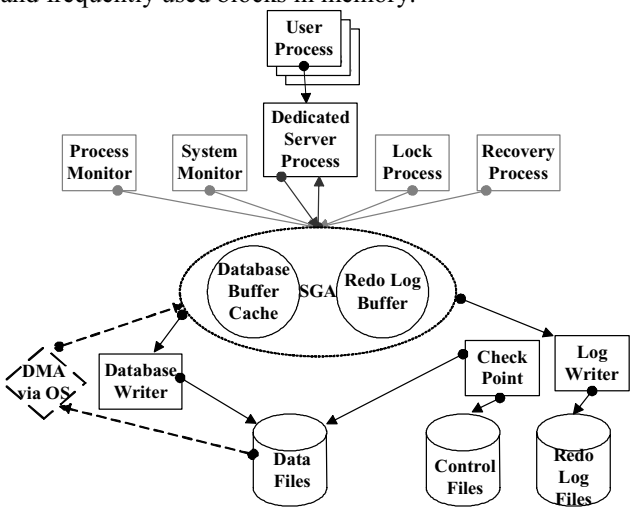

Figure 1: Overview of the ODB workload

The server processes operate by accessing control data blocks and database buffers held in the SGA. If a server process tries to access a database block not in SGA, the process initiates a DMA transfer to bring the block from disk to SGA and then relinquishes control of the CPU so that another server process can execute. When the disk transfer has completed, the operating system awakens the initiating server process at the next scheduled opportunity. This concurrent transaction execution allows the server process's idle time to be masked with useful work.

In our setup, the size of an ODB warehouse is about 100 MB, which includes the database tables and the indices. There are two log files each 25 GB that are shared by all warehouses.

3.2 OLTP Configuration Space

ODB has numerous configuration parameters that can influence ODB execution behavior. One important parameter is the SGA size. SGA is intended to hold as much of the database working set as possible in memory. In this study, we do not use the Intel physical addressing extensions [14] to go beyond 4 GB of virtual memory space. In the 4 GB physical system used in this study we allocate the minimum 1 GB required for the Linux OS and use the remaining memory for the SGA.

To reduce the configuration complexity to a manageable level, we focus on the following parameters to represent an OLTP configuration: number of warehouses (W), number of concurrent clients (C), number of processors (P), and number of disks (D). The first two parameters describe the workload configuration, while the latter two describe the system configuration.

3.2.1 Workload Scaling

Scaling the workload involves spanning the spectrum from a cached setup to a scaled setup. The parameter that dictates whether a setup is cached or scaled is the number of warehouses. The number of warehouses can span from tens of warehouses for a cached setup to thousands of warehouses for a scaled setup. Our goal is to formulate empirical relationships between a cached setup and a scaled setup.

In order to perform a valid performance comparison between two different ODB configurations, we try to maintain a consistent CPU utilization for all configurations. In this study we strive to keep CPU utilization above 90% across all configurations. To achieve such high CPU utilization, the number of clients needs to be increased as we scale the number of warehouses. This relationship exists because an increase in the number of warehouses results in an increase in the disk I/O rate. To mask the increasing disk I/O latency, the number of concurrent clients needs to be increased. In our experimental evaluation, we achieve our goal of 90+% CPU utilization at each configuration by adjusting the number of clients as appropriate in a range from 8 to 64. Consequently, the number of warehouses and the number of clients are not necessarily independent parameters. Table 1 shows the number of clients used for the wide range of OLTP configurations. As expected, the number of clients required to saturate the CPU grows relatively slowly for small number of warehouses and also for fewer processors. As we increase the number of warehouses far beyond the size of the SGA the disk I/O rate increases disproportionately requiring us to use more clients to hide the disk latency.

	Clients		
Warehouses	1P	2P	4P
10	8	10	10
50	8	16	32
100	6	16	48
500	12	25	56
800	13	36	64

Table 1: Number of Clients at 90% CPU Utilization

3.2.2 System Scaling

The two system parameters for an OLTP configuration are processors and disks. One objective of this study is to look at the design of chip multiprocessors (CMP) for OLTP workloads. Typical CMP designs [6][11][17] allow a modest number of processors (2 to 8

cores) to be integrated on a single silicon die. Our interest in CMP designs, coupled with the constraint of our Quad SMP system, limits the scaling of P to four processors. Furthermore, for this study, we did not scale D; instead, we limited the number of warehouses to 1200 to fit within the 26 disks in our experimental system.

3.3 IA-32 Multiprocessor System

At the beginning of this study we chose the best performing Intel Xeon 4-way SMP server at that time as our experimental machine, with Red Hat® Linux Advanced Server 2.1 OS and 2.4.9-e.10smp kernel. The Intel Xeon processors in our system operate at 1.6 GHz and have three levels of caches. The first level has an execution trace cache, while the second and third levels have unified instruction and data caches of 256 KB and 1 MB, respectively. The Xeon MP processors are based on the NetBurst™ microarchitecture and are capable of running with Hyper-Threading technology [15]. For the purpose of this study, we do not enable the Hyper-Threading technology. Our system is populated with 4 GB of PC200 DDR memory using the ServerWorks™ Grand Champion™ HE chipset and has 26 Ultra320 SCSI drives, each with 73 GB of capacity.

The Xeon MP processor provides a comprehensive set of embedded counters for performance-monitoring events [14]. There are 18 performance counters grouped into 9 pairs, with each pair associated to a particular subset of events. The particular counters can be selected by setting the counter configuration control registers. Performance monitoring on Intel processors is completely noninvasive and does not affect the execution of the Oracle software. To sample the performance-monitoring counters, we use the EMON program and the corresponding kernel-level driver. EMON is a software tool used internally at Intel that is able to sample the processor's performance-monitoring counters at a user-determined frequency. A number of publicly available software products provide similar access to the performance-monitoring counters, including the Intel VTune™ Performance Analyzer [15].

For all experiments, we ran ODB for twenty minutes prior to taking measurements. This warm up period allows the database to fill its buffer cache in SGA with data and enter into steady-state I/O behavior. A ten minute measurement period follows the warm up period. During the measurement period, each event is measured for ten seconds in a round-robin fashion. The event measurements are repeated six times.

3.4 "Iron Law" of Database Performance

We have adapted the classic "iron law" of processor performance [13] to provide a model for database performance and a framework for analyzing ODB workload behavior. The classic iron law states the processor performance, *S*, as a function of three terms: $S = \frac{F}{PL \times CPI}$, where *F* is the clock frequency, *PL* is the path length of the program, and *CPI* is the average cycles per instruction. Performance can be improved either by increasing *F*, or by decreasing *PL* or *CPI*. Although the three parameters can be independently improved, in practice they are inter-dependent on each other. For instance, increasing the frequency usually comes at the expense of increasing the CPI.

For database workloads, performance is determined by the transaction-processing throughput, which can be measured in terms of number of *transactions per second* (*TPS*). We can model the TPS of each processor in an OLTP system as follows: $TPS_{cpu} = \frac{F}{PL \times CPI}$, where the path length, *PL,* is now the average number of instructions executed per transaction (IPX) and *CPI* is the average number of cycles executed per instruction.

To get the overall throughput of a multiprocessor system, the TPS of each processor can be multiplied by the number of processors in the system. We define a new *iron law of database performance* for a multiprocessor system as follows:

$$TPS_{mp} = \frac{P \times F}{IPX \times CPI}.$$

Database performance can be increased by increasing *F* or *P*, or by decreasing *IPX* or *CPI*. The CPI in this equation is the average CPI measured at each processor, taking into account the effects of inter-processor communication. For our study, *F* is fixed, and *P* is varied. Our intent is to understand how IPX and CPI vary over a wide range of processor and warehouse configurations and the next two sections present experimental data to aid this understanding.

4. System-level Behavior of ODB

The performance of an OLTP system is typically measured in terms of transaction processing throughput. In this section, we first show how TPS varies with the number of warehouses and the number of processors. We then examine the two primary system characteristics that affect TPS: IPX and CPI. The CPI component is further analyzed in Section 5.

4.1 Transaction Processing Throughput

Figure 2 shows how TPS varies as we scale the number of warehouses from 10 to 1200, and as we scale the number of processors from 1 to 4. The maximum TPS is achieved at about 10W for all three processor configurations. The TPS then decreases as the number of warehouses is increased.

To better explain the TPS trends, we first identify three distinct regions of ODB operation: *CPU bound,*

balanced, and *I/O bound*. We describe a system as being *CPU bound* when the working set of ODB fits almost entirely in main memory. As previously mentioned, researchers typically call such a system a cached setup. The performance of a CPU bound system is limited by the available computing power. These systems have negligible disk I/O; hence, there is very little need to overlap processing and I/O. As shown in Table 1 fewer than 10 concurrent clients are needed to saturate the CPU. Based on the disk I/O measurements, in our setup, we classified a CPU bound system to have fewer than 50 warehouses.

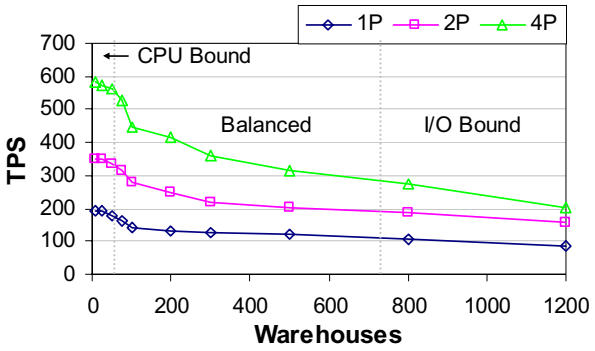

Figure 2: Variance of ODB TPS with P and W Scaling

The ODB working set increases almost linearly with the number of warehouses. As the working set increases beyond the size of the database buffer cache (2.8 GB in our setup), fewer blocks remain cached in the database buffer cache and, therefore, more disk I/O is required. The database system tries to mask this increasing disk latency with concurrent, independent threads of execution from the different database clients. We define a *balanced* system as one having sufficient compute and disk bandwidth in which the disk read latency can be completely overlapped by using an ample number of independent threads of execution. We maintain the CPU utilization of a balanced system above 90%, representing a system that overlaps nearly all of the I/O latency. However, to achieve such a high level of CPU utilization these configurations need more concurrent clients as shown in Table 1. A balanced system in out setup has fewer than 800 warehouses.

As the number of warehouses continues to increase, the I/O subsystem eventually reaches its maximum throughput. We define such a system as being *I/O bound*. Adding CPUs or increasing the number of database clients cannot increase the performance of an I/O bound system. In our setup, the 1200-warehouse setup can be categorized as an I/O bound system since our I/O subsystem reached its maximum throughput, and the CPU utilization would not go higher than 63% for the 4P system.

In order to compare various ODB configurations fairly, we chose only those configurations that could sustain a high level of CPU utilization. For the remainder of the paper, we only discuss our results for configurations that range from 10 to 800 warehouses in which the CPU utilization is consistently kept above 90%. We do not include the 1200W setup in which the CPU utilization is much less than 90%. Note that in Figure 2, the demarcation of the three regions is done based on empirical measurements on our machine and the boundaries of these regions are dependent on the system configuration, such as memory size, and disk bandwidth.

Figure 3 shows how the overall CPU utilization is split between OS and user code. In ODB, most of the OS execution overhead comes from disk I/O. Since the disk I/O grows with the number of warehouses the OS overhead increases from less than 10% to just above 20% at 800 warehouses.

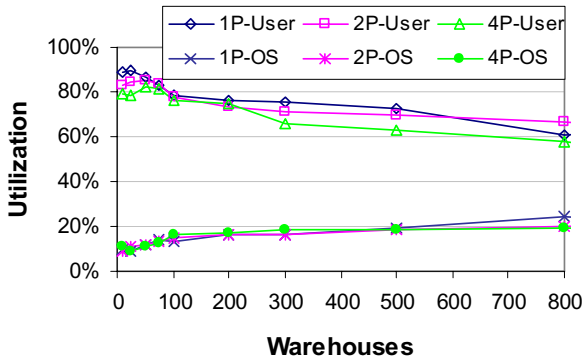

Figure 3: CPU Utilization Split: OS and User

4.2 Transaction Path Length

In the previous subsection, we showed how the overall TPS performance varies with W and P. In this section, we show how IPX contributes to this TPS trend. As stated in Section 3.4, TPS decreases with increasing IPX, if all other factors remain constant.

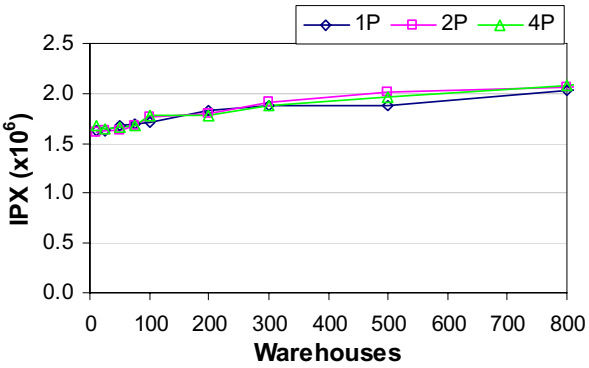

Figure 4: Millions of Instructions per ODB Transaction

In Figure 4, the IPX trend (in millions of instructions) is shown to increase roughly linearly with W. To explain this trend, we split the IPX into two components: user-space IPX and OS-space IPX, shown in Figure 5 and Figure 6, respectively. The user-space IPX is

flat, indicating that the number of instructions executed in the database does not vary significantly with W. However, the OS code path is increasing with W. This trend is due to the increasing amount of I/O that must be serviced by the OS, along with an increasing amount of time spent in the OS scheduler to perform context switching. This trend is further analyzed in the next subsection.

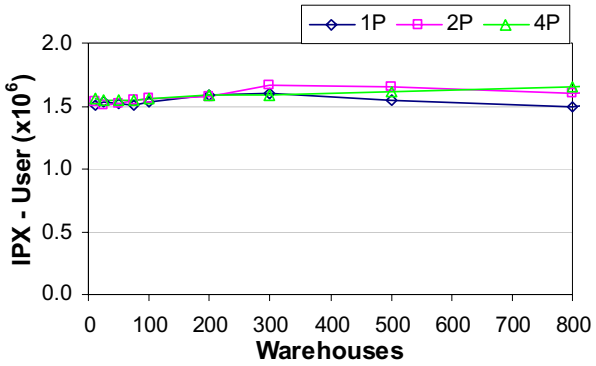

Figure 5: User-Space IPX

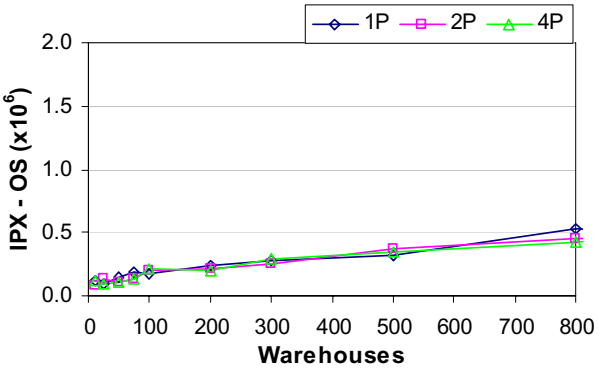

Figure 6: OS-Space IPX

4.3 Major System Events Affecting IPX

Figure 7 shows the total number of disk I/Os (reads and writes) per transaction, in 1K blocks. As explained in Section 2.1, the size of the working set increases linearly with the number of warehouses. Once the working set size surpasses the SGA size, the disk I/O per transaction increases with warehouses. From 10 to 25 warehouses, the disk read traffic is very small, as all of the working set resides in the main memory. As the number of warehouses increases, more disk I/O occurs to read in database pages from disk that no longer fit in the database buffer cache.

The amount of data written to disk also increases with the number of warehouses. The write traffic can be categorized into two types. One is the log data written to the disk for the purpose of maintaining consistency and durability of database updates. The amount of log data generated depends on the transaction type and how the database executes that transaction. In particular, it does not depend on the number of processors or on the number of warehouses. ODB, on average, generates 6 KB of log data per transaction. For up to 25 warehouses, the disk write traffic is almost entirely log traffic, as there is almost no need to swap pages out of the database buffer cache.

The second type of disk write traffic is due to writing back the dirty pages that are evicted from the database buffer cache. The working set increases with the number of warehouses, and as a result, a growing number of dirty pages will have to be removed from the database buffer cache and written back to disk.

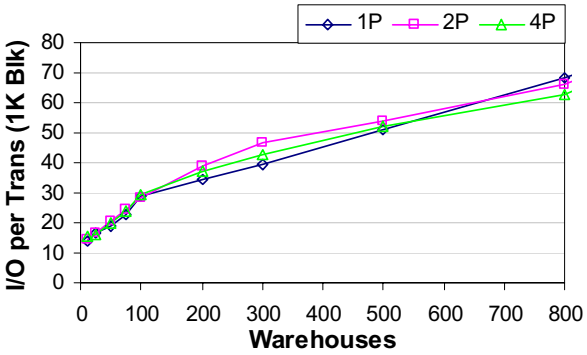

Figure 7: Total disk I/Os per ODB Transaction

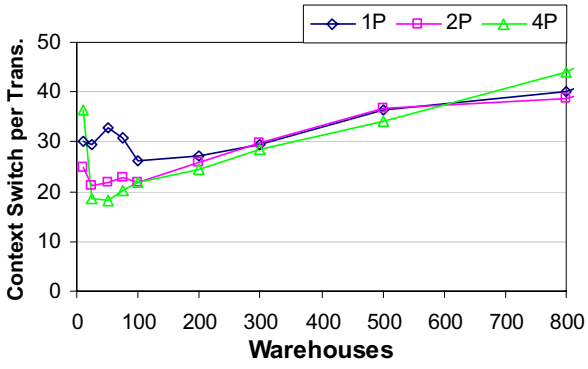

Figure 8: Context Switches per ODB Transaction

Closely related to the amount of I/O processed by the system is the amount of context switching that occurs per transaction. The operating system uses context switching to continue computation across multiple, independent threads of execution, one or more of which may be stalled due to disk reads or blocked on a synchronization structure, such as a semaphore. The number of context switches should be proportional to the disk reads, except at the smallest warehouse configurations where data contention is at its highest. Note that disk writes are typically non-critical and are handled asynchronously by the OS.

Figure 8 shows the number of context switches per ODB transaction executed. The amount of context switching is very high for the 10-warehouse configuration, primarily due to database block contention that results from multiple processes sharing a very small data set. As the number of warehouses increases, the amount of context switching sharply decreases as the data contention

is reduced. As the workload size continues to increase, increasing disk I/O causes the context switching to increase. Based on the trend in Figure 8, apart from the cached configurations, all other configurations exhibit context-switching trends that correlate well with the disk I/O reads.

5. Processor-level Behavior of ODB

In this section, we analyze the average cycles per instruction (CPI) as W is varied. We decompose CPI to understand how various microarchitectural components contribute to the overall CPI. Our results indicate that L3 cache misses are the single largest bottleneck to CPI performance. Hence, we further analyze the number of L3 misses per instruction executed (MPI), and the impact of bus utilization on the L3 miss penalty. As stated in Section 3.4, TPS decreases with increasing CPI, if all other factors remain constant.

5.1 Processor CPI Trends

Figure 9 shows the CPI trends as the number of warehouses is increased. The overall CPI increases with the number of warehouses and with the number of processors. From 10 to 100 warehouses, the slopes are steep, but as W increases beyond 100, the slopes begin to level off, indicating small CPI increase with increasing W. The CPI trends in Figure 9 are further separated into the CPI for user-space instructions and the CPI for OS-space instructions, as shown in Figure 10 and Figure 11, respectively. The user-space CPI, shown in Figure 10, correlates well with the overall CPI. This correlation is to be expected as the user-space code is between 70%-80% of all code executed.

The OS-space CPI trend, shown in Figure 11, slightly decreases with the increase in W. The high variance in the OS-space CPI trend for a small number of warehouses can be attributed to the small percentage of time that is spent in the operating system code and the resulting sampling errors in EMON. As the amount of time spent executing operating system code increases, the CPI decreases due to the improved locality and due to idle spin loops.

5.1.1 CPI Breakdown

To further understand how various microarchitectural components of the Xeon processor contribute to the overall CPI, we decompose the CPI into microarchitectural events and measure these events using the performance counters. After extensive analysis, we found that 10 events are satisfactory to characterize the microarchitectural behavior of our system. We list these performance-monitoring events in Table 2, along with aliases that we use for the remainder of this section.

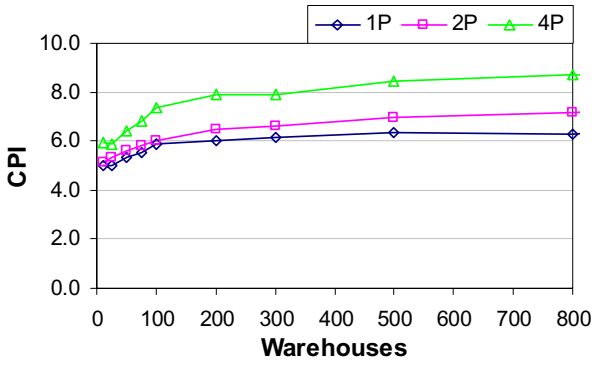

Figure 9: Overall CPI Trends

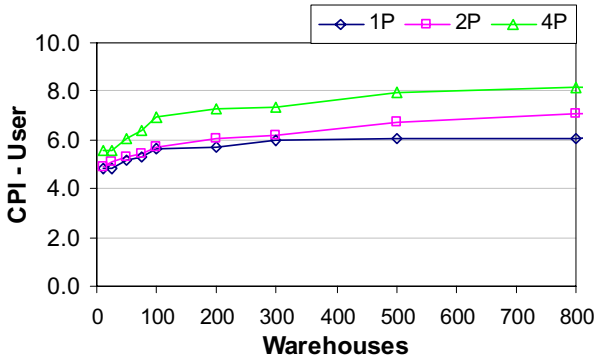

Figure 10: User Space CPI

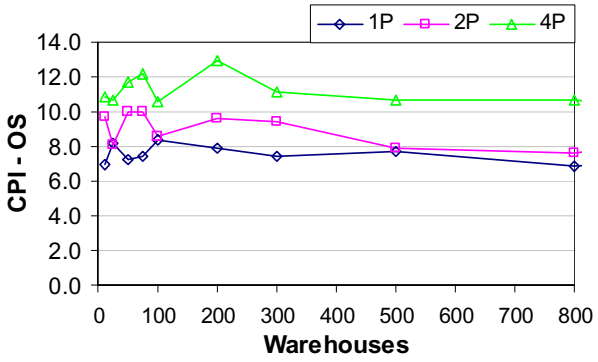

Figure 11: OS-Space CPI

To attribute the microarchitectural components to the CPI, we use the simple approach of assigning a fixed number of CPU stall cycles to each microarchitectural event, as shown in Table 3. The contribution of each event to the overall CPI is then calculated by multiplying the fixed stall cycles with the corresponding event count, as shown in Table 4. After summing the resulting contributions, we arrive at a computed CPI. The difference between the measured CPI and the computed CPI is recorded as the "Other" component.

Event Alias	EMOM Events Used	Description
Instructions	instr_retired	The number of instructions retired
Branch Mispredictions	mispred_branch_retired	The number of mispredicted branches
TLB Miss	page_walk_type	The number of misses in the TLB
TC Miss	BPU_fetch_request	The number of misses in the Trace Cache
L2 Miss	BSU_cache_reference	The number of misses in the L2 cache
L3 Miss	BSU_cache_reference	The number of misses in the L3 cache
Clock Cycles	Global_power_events	The number of unhalted clock cycles
Bus Utilization	FSB_data_activity	The percentage of time the processor bus is transferring data
Bus-Transaction Time	IOQ_active_entries & IOQ_allocation	The average amount of time to complete a bus transactions once it enters the IOQ

Table 2: Performance-Monitoring Events Used in CPI Analysis

Event Alias	Cycles per Event
Instruction	0.5
Branch Misprediction	20
TLB Miss	20
TC Miss	20
L2 Miss	16 (measured)
L3 Miss	300 (measured)
Bus-Transaction Time for 1P	102 (measured)

Table 3: Clock Cycle Cost for Each Component

CPI Component	Contribution Formula
Inst	Instructions * 0.5
Branch	Branch Mispredictions * 20
TLB	TLB Miss * 20
TC	TC Miss * 20
L2	(L2 Miss – L3 Miss) * 16
L3	L3 Miss * (300 + Bus-Transaction Time – Bus-Transaction Time for 1P)
Other	Clock Cycles / Instructions – sum (computed components)

Table 4: Description of CPI Component Costs

Figure 12 shows the breakdown of the CPI into its components for increasing workload sizes. From this breakdown, it appears that the primary contributor to CPI is the main memory latency, labeled as "L3 Miss" in the figure. As the figure shows, surprisingly, the compute and branch components of the CPI vary little as the workload scales; however, the memory component increases with the number of warehouses and with the number of processors. This fact motivates further examination of the L3 cache miss behavior in the following subsection.

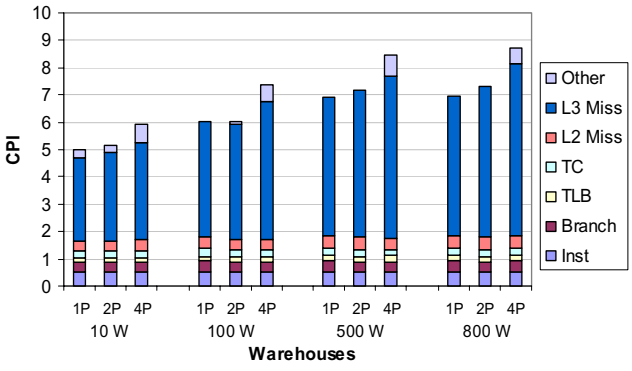

Figure 12: CPI Breakdown by Event

5.2 Memory System Contributions

In Section 5.1, we show that the L3 cache-miss latency is the primary bottleneck to performance, contributing nearly 60% to the CPI. We now investigate how L3 cache behavior changes with W in order to gain insight into how the memory-system behavior affects CPI. The metric that we use is the average number of L3 cache misses per instruction executed (MPI).

Figure 13 shows the L3 MPI as W is varied. Starting with 10 warehouses, the MPI increases sharply until about 100 warehouses, as the working set exceeds L3 capacity. At approximately 100 warehouses, the trend in MPI becomes less steep, as the L3 miss rate reaches saturation at 60%. As the number of warehouses continues to increase, the L3 MPI also continues to increase due to the increasing context switching, which results in cache flushes.

The overall MPI increases with W but, surprisingly, MPI does not increase with the number of processors. These results show that cache coherence misses are not noticeably affecting the miss rate. This observation seems to contradict the findings in previous research [5][6][7]. We believe that the discrepancy is due to two reasons. First, we are using Oracle9i as opposed to Oracle7i, which

is used in [6]. Oracle9i has better scaling performance than Oracle7i. Second, low coherence misses may be an indirect effect of having a relatively small (1MB) L3 cache, in which the coherence misses are overshadowed by the magnitude of capacity misses.

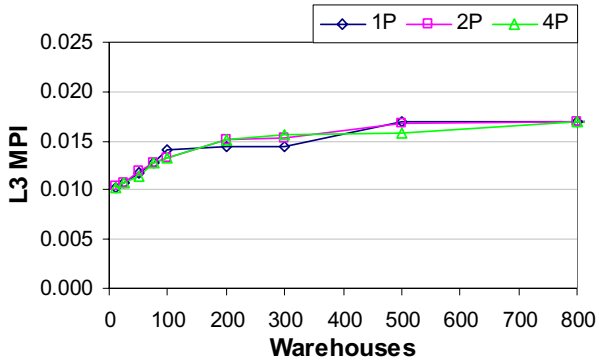

Figure 13: L3 Misses per Instruction Executed

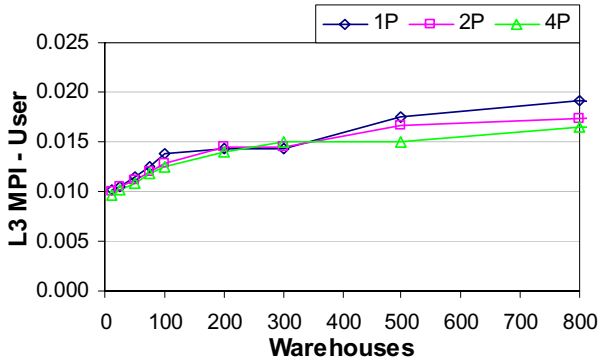

Figure 14: User-Space L3 Misses per Instruction

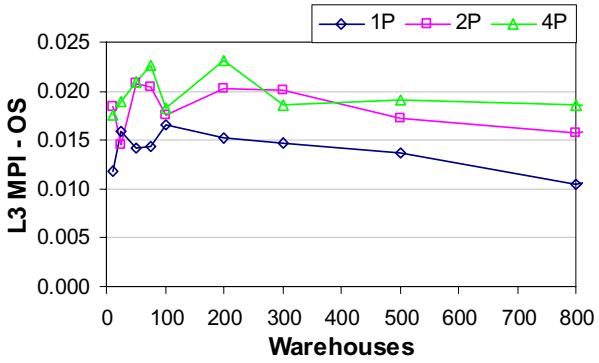

Figure 15: OS-Space L3 Misses per Instruction

The overall L3 MPI has been further split into two components: user-space and OS-space components, as shown in Figure 14 and Figure 15, respectively. In Figure 14, the user-space component of MPI correlates well with the overall MPI. In Figure 15, the OS-space component of MPI decreases with the increasing workload. This trend is due to the increasing amount of time spent in kernel code, which results in increased temporal locality of kernel code and data structures.

The trend in the overall MPI (Figure 13) correlates well with the trend in CPI (Figure 9), except that CPI increases with the number of processors while MPI does not. Further investigation revealed that CPI increases with P due to growing bus-traffic overhead; as the number of processors increases, so does the total number of transactions on the bus. As the bus utilization increases, the time to execute a bus transaction also increases. Figure 16 shows the average time, in CPU clocks, to execute a bus transaction once it enters the IOQ. The IOQ latency remains nearly constant with increasing W on 1P. However, there is a dramatic increase in IOQ latency in 4P due to the increased bus utilization. The bus utilization approaches 45% for the 4P configurations but is less than 30% for the 2P configurations.

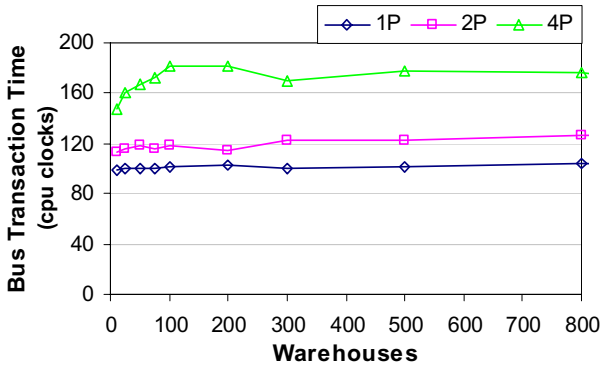

Figure 16: Bus-Transaction Time (in the IOQ)

6. Modeling Behavior

In the previous section, our results show that both the CPI behavior and the MPI behavior follow predictable trends across a wide range of ODB configurations. We now show that trends in CPI and MPI can be accurately approximated by two linear regions of behavior. We then propose a method for selecting the minimal ODB configuration that can be used as a basis for analyzing large-scale ODB configurations.

6.1 CPI and MPI Trending

In Section 5.2, we analyzed the L3 MPI trends and concluded that the MPI trends have two regions of behavior. In the first region, which includes configurations up to 100 warehouses, the sharp increase in MPI is due to L3 caching effects. We call this region the *cached region*. In the second region, which includes configurations having more than 100 warehouses, the L3 cache miss rate reaches near saturation, and the MPI trend becomes relatively flat. We call this region the *scaled region*. Our results also show that L3 cache miss stalls are the single largest determinant of CPI; not surprisingly, the CPI trends also reflect the same two regions of behavior. We propose to use simple, linear models to approximate the two regions of behavior.

Using the CPI trend of the 4P configuration, Figure 17 illustrates the linear approximation models for the two behavioral regions. Using linear least-squares regression on the CPI data, we derive the two linear equations. We consider the intersection of the two linear equations to be the boundary between the cached region and the scaled region. We call the intersection of the two regions the *pivot point*. The pivot point represents a transition point where the execution behavior of the workload on a given system changes from being a cached setup to behaving like a scaled setup.

Similar to the CPI trend, the MPI trends can also be separated into two regions that can be modeled by two linear equations. Using the MPI trend for the 4P configuration, Figure 18 shows the two linear equations representing the cached and scaled regions for the L3 MPI.

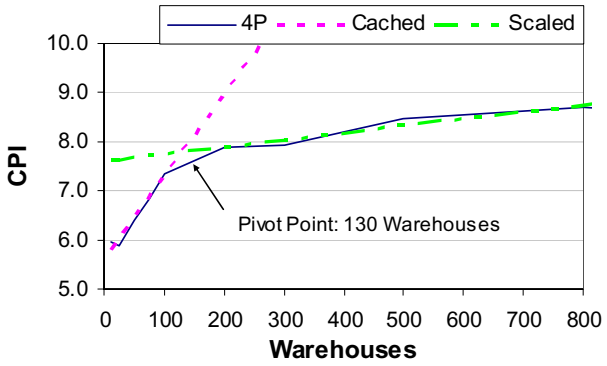

Figure 17: Linear Model of CPI

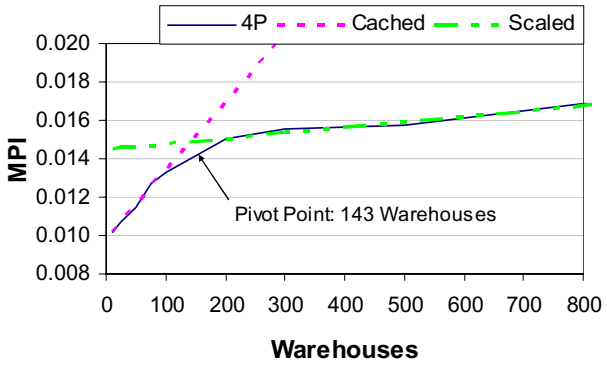

Figure 18: Linear Model of MPI

6.2 Pivot Point for Workload Scaling

The pivot point can be used as a lower bound to represent an OLTP workload with sufficient execution behavior to look like a scaled setup. In other words, for a configuration to exhibit scaled setup behavior, its size must be greater than the pivot point. For example, based on Figure 17 and Figure 18, the 200W setup can be viewed as a representative scaled setup, based on which the behavior of even larger setups can be accurately extrapolated using the linear approximation equation. Hence, simulation results based on the 200W setup may be used to accurately project the behaviors of fully scaled setups, and there is no need to simulate larger setups.

	CPI	MPI
1P	119	102
2P	142	147
4P	130	144

Table 5: Number of Warehouses for Pivot Points

The numbers of warehouses for the CPI and MPI pivot points for all three processor configurations are shown in Table 5. All the pivot points are below 150 warehouses across all processor configurations. Except for 1P, the CPI pivot point is smaller than the MPI pivot point. This discrepancy is due to the increased bus latency of 2P and 4P configurations over 1P (Figure 16). The MPI trends account only for the number of L3 misses, while the CPI trends account for the variation in the latency of an L3 miss. The increased bus latency makes the CPI trends steeper than the MPI trends; hence, the CPI trend reaches the scaled region earlier than the MPI trend. Since the CPI accounts for the latency effects more accurately than MPI, we believe that the CPI pivot point is a more accurate representative of the scaled region. Since the pivot points shown in Table 5 are derived from a single machine configuration, one natural question is how does the pivot point change with varying system attributes?

6.3 System Effects on the Pivot Point

The slope of the cached region is determined by the L3 cache size. We expect the slope of the CPI curve in the cached region to decrease with increasing L3 cache size. Hence, the pivot point will shift to the right with increasing L3 cache size. On the other hand, the slope of the scaled region is determined by the context switch frequency. Context switch frequency is, in turn, dependent on the number of concurrent clients and the I/O bandwidth. As the I/O bandwidth increases with the addition of disk drives, the I/O latency will decrease (until it reaches some minimum threshold), and the number of concurrent processes necessary to overlap I/O latency will also decrease. As a result, the number of context switches, and hence the OS overhead, will decrease. The pivot point may then shift towards the left, indicating that a smaller configuration can be used as the representative workload. Thus, increasing the L3 cache size and disk bandwidth has opposing effects in determining the pivot point. For systems with larger L3 caches, more bus bandwidth, and more disk I/O bandwidth than our system, we also expect a smoother transition between the cached and the scaled regions than what we observed.

In order to validate these conjectures, we recently conducted similar scaling experiments on a Quad Itanium2 server with 16GB memory and 34 disks (8 more than what we used in our Xeon server). The Itanium2 processor has a 3MB L3 cache and has about 50% higher bus bandwidth.

Due to timing and space constraints we present only the CPI scaling data, as shown in Figure 19. More details on our Itanium2 experiments are available in [22]. As shown in the figure, the 3MB L3 cache in Itanium2 processor reduces the slope of the CPI curve in the cached region. Similarly, the increased bus and disk bandwidth causes a more gradual increase in the CPI in the scaled region. Hence, the resulting CPI pivot point of 118 warehouses is close to the CPI pivot point of the Quad Xeon server.

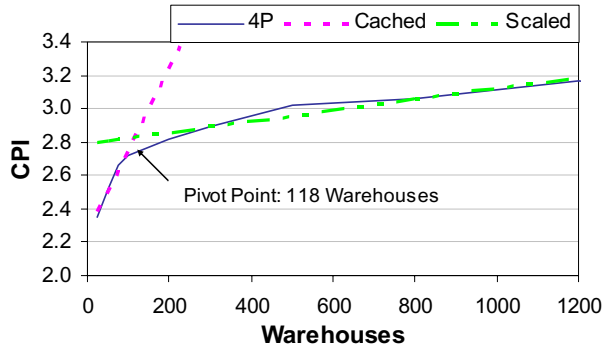

Figure 19 CPI Scaling on an Itanium2 Quad Server

7. Conclusions

The extensive experimental study presented in this paper has yielded a number of interesting observations, some not surprising and others unexpected. For database workloads running on small-scale multiprocessors, the primary impediment to processor performance is the penalty due to L3 cache misses. Contrary to the belief among researchers, coherence misses from a small-scale multiprocessor system are not a crucial bottleneck to overall performance. Instead, having adequate bus and I/O bandwidths along with cache capacity is essential to achieving high transaction throughput.

Overall database performance can be modeled by an adaptation of the classic iron law for processor performance. Based on our new iron law of database performance, we show that both the IPX (average instructions per transaction) and CPI (average cycles per instruction) are critical to performance. As the database workload size is scaled, we see corresponding increases in IPX and CPI. A linear increase in IPX is observed, mainly due to the increase of I/O activities and the associated increase in context switches. The increase in CPI can be accurately modeled by two linear regions. In the cached regions (small number of warehouses), the CPI increase is quite steep, reflecting the inability of the L3 to capture the working set. However, in the scaled region (large number of warehouses), L3 cache misses reach near saturation, and the CPI increase is quite gradual for increasing workload size. To increase overall database workload performance, the primary targets for optimization are the reduction of IPX and CPI.

While our experimental setup is representative of a typical database server, it has a few limitations, and the experimental results from this study must be interpreted with these limitations in mind. First, the 1 MB L3 cache is not large enough for ODB; larger L3 capacity is definitely needed for OLTP workloads. The 26 SCSI disk drives in our experimental setup are insufficient to provide adequate I/O bandwidth when the number of warehouses is increased beyond 800, which in turn can increase the I/O latency and induce additional stalled times in the CPUs waiting for disk I/O reads. We also see the limitation of the bus bandwidth that prevents us from scaling the number of processors beyond 4. For the 4P system with less than 1000 warehouses, we see the bus utilization approaching 45%, which is quite high for a production OLTP system. Of course, the 32-bit addressing limits our physical memory capacity. This in turn limits the SGA size and the critical database buffer cache size. For our immediate follow-on work, we plan on alleviating some of these limitations in a new set of experimental studies.

Regardless of these limitations, this study has given us a great deal of insights into database (OLTP) workloads and how they behave on a small-scale multiprocessor system. One aim of this study is to demystify the database workloads for architects who are designing the next-generation server processors and platforms. We see that there is no mysterious chasm between small cached setups and large scaled setups, in which unpredictable behaviors may emerge. While the IPX and CPI at the two extremes are different, there are predictable trends that span from one end to the other, at least across the two orders of magnitude of workload sizes (10W to 1000W).

The experimental study of this paper has yielded new "food for thought" concerning future research directions. In terms of research methodology, in order to capture scaled setup behavior, the database workload configuration must be greater than the pivot point size. In terms of novel techniques for improving database workload performance, they must ultimately reduce either IPX or CPI, or both. The reduction of IPX should focus on reducing the overhead of context switches due to I/O activities. For CPI reduction, techniques targeting coherence misses may not yield much performance gain. Other than increasing the capacity of L3, more efficient use of the limited L3 capacity, through more judicious and specialized caching schemes, should be explored. We plan to address some of these issues in our future research.

8. Acknowledgements

We would like to thank Nhon Quach (Oracle), Seckin Unlu (Intel) and his group, and Yong-Fong Lee (Intel) for reviewing the paper and providing constructive feedback. We also thank the anonymous reviewers for their thorough and thoughtful comments and suggestions.

9. References

[1] A. Ailamaki, D. DeWitt, M. Hill, and D. Wood. DBMSs on a Modern Processor: Where Does Time Go? In *Proceedings of the 25th International Conference on Very Large Data Bases*, pages 266–277, September 1999.

[2] A.R. Alameldeen and D.A. Wood. Variability in Architectural Simulations of Multi-threaded Workloads, In *Proceedings of the 9th International Symposium on High-Performance Computer Architecture*, pages 7-18, February 2003.

[3] M. Annavaram, T. Diep and J.P. Shen. Branch Behavior of a Commercial OLTP Workload on Intel IA32 Processors. In *Proceedings of the International Conference on Computer Design*, pages 242-248, January 2001.

[4] T. Diep, M. Annavaram, H. Nueckel, B. Hirano, and J. P. Shen. Analyzing Performance Characteristics of OLTP Cached Workloads by Linear Interpolation. In *Proceedings of the 6th Workshop on Computer Architecture Evaluation using Commercial Workloads*, pages 51-59, February 2003.

[5] L.A. Barroso, K. Gharachorloo, and E. Bugnion. Memory System Characterization of Commercial Workloads. In *Proceedings of the 25th International Symposium on Computer Architecture*, pages 3–14, June 1998.

[6] L.A. Barroso, K. Gharachorloo, R. McNamara, A. Nowatzyk, S. Qadeer, B. Sano, S. Smith, R. Stets, and B. Verghese. Piranha: A Scalable Architecture Based on Single-Chip Multiprocessing. In *Proceedings of the 27th International Symposium on Computer Architecture*, pages 282-293, June 2000.

[7] L.A. Barroso, K. Gharachorloo, A. Nowatzyk, and B. Verghese. Impact of Chip-Level Integration on Performance of OLTP Workloads. In *Proceedings of the 6th International Symposium on High-Performance Computer Architecture*, pages 3-14, January 2000.

[8] Z. Cvetanovic and D. Bhandarkar. Characterization of Alpha-Axp Performance using TP and SPEC Workloads. In *Proceedings of the 21st International Symposium on Computer Architecture*, pages 60–70, April 1994.

[9] J. Lo, L. A. Barroso, S. Eggers, K. Gharachorloo, H. Levy, and S. Parekh. An Analysis of Database Workload Performance on Simultaneous Multithreaded Processors. In *Proceedings of the 25th Annual International Symposium on Computer Architecture*, pages 39-50, June 1998.

[10] M. Franklin, W.P. Alexander, R. Jauhari, A.M.G. Maynard, B.R. Olszewski. Commercial Workload Performance in the IBM Power2 Risc System/6000 Processor. IBM J. of Research and Development, 38(5): 555–561, 1994.

[11] J. Kahle. Power4: A Dual-CPU Processor Chip. *Microprocessor Forum '99*, October 1999.

[12] K. Keeton, D.A. Patterson, Y.Q. He, R.C. Raphael, and W.E. Baker. Performance Characterization of a Quad Pentium Pro SMP Using OLTP Workloads. In *Proceedings of the 25th International Symposium on Computer Architecture*, pages 15–26, June 1998.

[13] J. Shen and M. Lipasti, Modern Processor Design: Fundamentals of Superscalar Processors, McGraw Hill, 2002.

[14] The IA-32 Intel® Architecture Software Developer's Manual, Volume 3: System Programming Guide.

[15] The Intel VTune Performance Analyzer. http://www.intel.com/software/products/vtune/.

[16] The Intel Xeon Processor MP Product Overview. http://developer.intel.com/design/Xeon/xeonmp/prodbref/.

[17] K. Olukotun, B.A. Nayfeh, L. Hammond, K. Wilson and K. Chang. The Case for a Single-Chip Multiprocessor. In *Proceedings of the 7th International Symposium on Architectural Support for Parallel Languages and Operating Systems*, pages 2-11, Oct 1996.

[18] P. Ranganathan and K. Gharachorloo and S.V. Adve and L.A. Barroso. Performance of Database Workloads on Shared-Memory Systems with Out-of-Order Processors. In *Proceedings of the 8th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 307–318, October 1998.

[19] M. Rosenblum, E. Bugnion, S. Herrod, E. Witchel, and A. Gupta. The Impact of Architectural Trends on Operating System Performance. In *Proceedings of the 15th Symposium on Operating Systems Principles*, pages 285–298, December 1995.

[20] Standard Performance Council. The SPEC95 CPU Benchmark Suite. http://www.spec.org/cpu2000.

[21] K. Keeton, D.A. Patterson. The impact of Hardware and Software Configuration on Computer Architecture Performance Evaluation. In *the* first *Workshop on Computer Architecture Evaluation using Commercial Workloads*.

[22] R. Hankins, M. Annavaram, T. Diep, H. Eri, B. Hirano, H. Nueckel, and J. P. Shen. *Comparing and Contrasting OLTP Workload Scaling on IA32 and IPF*. October 2003. http://www.intel.com/research.