Comparing the Memory System Performance of DSS Workloads on the HP V-Class and SGI Origin 2000 *

Rong Yu
Dept of Computer Science
Texas A&M University
College Station, TX
r0y6011@cs.tamu.edu

Laxmi Bhuyan
Dept of Computer Science
University of California
Riverside, CA
bhuyan@cs.ucr.edu

Ravi Iyer
Enterprise Architecture Lab
Intel Labs
Hillsboro, OR
ravishankar.iyer@intel.com

Abstract

In this paper, we present an in-depth analysis of the memory system performance of the DSS commercial workloads on two state-of-the-art multiprocessors: the SGI Origin 2000 and the HP V-Class. Our results show that a single query process takes almost the same amount of cycles in both machines. However, when multiple query processes run simultaneously on the system, the execution time tends to increase more in SGI Origin 2000 than in HP V-Class due to the more expensive communication overhead in SGI Origin 2000. We also show how the rate at which number of data cache misses, context switches and the overall execution time increases when more query processes run simultaneously.

1 Introduction

Commercial workloads such as DSS and OLTP workloads have surpassed the traditional workloads (scientific/technical applications) and now dominate the server market. However, system architecture and design has been primarily influenced by scientific benchmarks such as SPLASH-2 [13]. Recent studies [1, 2, 5, 6, 8, 11] have shown that commercial database workloads have different system characteristics from scientific workloads. All of this research focuses on understanding the characteristics of DSS or OLTP workloads on a single platform. The contribution of our research is the characterization as well as comparison of the memory system performance of DSS workloads across different multiprocessor designs. The two state-of-the-art platforms used in our research are the HP V-Class and the SGI Origin 2000. Moreover, we not only examine the performance characteristics of a single query process running on these systems, but also evaluate the performance characteristics with varied number of query processes simultaneously running in the system. Our previous effort [4] on understanding the memory system performance of the HP V-Class and the SGI Origin 2000 used microbenchmarks and scientific applications. In this paper, we focus on the performance of a commercial decision support system workload based on the TPC-H benchmark [12]. Due to the long running time of TPC-H queries and resource constraints, we selected 3 representative TPC-H queries (Q6, Q21 and Q12) and ran them on an open source DBMS (PostgreSQL) that we instrumented using hardware counter APIs to collect performance data on the two systems.

Our results show several aspects of system characteristics of TPC-H queries. We show that the DSS data cache performance differs depending on whether the query mainly uses sequential scan to access tables or mainly uses indices to access tables. We also show that the two level cache hierarchy in the SGI Origin 2000 provides better performance than the single cache hierarchy in HP V-Class. The larger cache size and larger line size has a bigger effect on index queries than on sequential queries. A performance comparison between SGI Origin 2000 and HP V-Class shows that when only one query runs on the system it takes almost the same amount of cycles on both machines. However when more processes (each running the same type of query) run simultaneously on the system, the execution time on SGI Origin 2000 tends to increase more than that on HP V-Class due to the more expensive communication overhead.

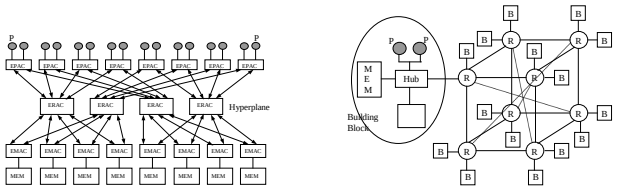
2 Experimental Setup & Methodology

In this section, we describe the hardware system, the DBMS, the TPC-H benchmark and the methodology of our experiments.

2.1 System Architectures

The two shared memory multiprocessors in our study are HP V-Class [3] and SGI Origin 2000 [7]. A conceptual block diagram of the 16-processor HP V-Class server [3] is shown in Fig. 1. The symmetric multiprocessor design utilizes 8 Exemplar Processor Agent Controllers (EPAC),

*This work was done while the authors were at Texas A&M University



(a) HP V-Class (b) SGI Origin 2000

Figure 1. System Architecture Overview

4 Exemplar Routing Attachment Controllers (ERAC), and 8 Exemplar Memory Access Controllers (EMAC). Each EPAC consists of two HP PA-8200 processors. Connecting EPACs and EMACs is a hyperplane crossbar, that provides high-bandwidth, low-latency nonblocking access from processors and I/O to the memory. The PA-8200 is a 64-bit, 200MHz, 4-way out-of-order superscalar processor based on the RISC Precision Architecture (PA-2.0). Its one-level cache system has separate 2MB instruction cache and 2MB data cache. As we see from Fig. 1, the memory system follows the uniform memory access (UMA) principle.

A conceptual block diagram of the 32-processor SGI Origin 2000 is shown in Fig. 1. The SGI Origin 2000 is a cache-coherent non-uniform memory access (ccNUMA) multiprocessor [7]. The basic building block is a dual processor node. Within each node, there are two MIPS R10000 processors, 1-8 memory banks, each with 512MB memory and its corresponding directory memory, and a portion of the I/O subsystem. The processors, memory and I/O are connected to a hub chip in the node, which has a crossbar, a network interface and interfaces to above modules. The nodes are connected together via a bristled hypercube interconnect network. The R10000 is a 64 bits, 250MHz, 4-way out-of-order superscalar processor. It has a 32KB L1 instruction cache, a 32 KB L1 data cache and a 4MB unified L2 cache. The cache line sizes for L1 and L2 cache are 32 bytes and 128 bytes respectively.

2.2 Software Overview (PostgreSQL & TPC-H)

PostgreSQL [9, 10] is the most advanced open-source database management system. It was originally developed at the University of California at Berkeley. The reason we chose PostgreSQL is mainly because it is an open source DBMS which enables us to instrument the source code with hardware counter references to collect various statistics. There are some differences though, between commercial DBMS such as Oracle and PostgreSQL. One major difference between a commercial DBMS and PostgreSQL can be the granularity of locking system. Currently PostgreSQL fully supports only relation level locking. This may become a bottleneck in multiple parallel queries. Fortunately, since we only test read-only queries, this bottleneck is not applied in our case because multiple query processes can get multiple read locks for the same table.

The TPC-H benchmark [12] simulates a complex business analysis system for a wholesale supplier to manage, sell and distribute products worldwide. TPC-H benchmark includes 22 read-only queries (Q1-Q22) and 2 refreshment functions (RF1,RF2). Our research just focuses on read-only queries because these queries are more complicated than the refreshment functions and more frequently used to make critical business decisions. We chose Q6, Q21 and Q12 as the ones that we examine in detail in our study. We believe that these 3 queries can represent the important memory access characteristics of most other queries. Q6 is a simple query in TPC-H benchmark. Q6 gives the amount of revenue that would have been added if the company had eliminated certain discounts for small quantity orders in a certain year. One sequential scan of table Lineitem is enough for Q6. Q21 lists the top 100 infamous suppliers that failed to meet the committed delivery dates for multi-supplier orders in which they were the only ones that failed. Q21 has one sequential scan of table Order and five index scans, including three on table Lineitem. Thus Q21 should intrinsically exhibit the characteristics of index scans. Q12 determines whether selecting less expensive modes of shipping is negatively affecting critical-priority orders by causing more parts to be received by customers after the committed date. Q12 scans the table Lineitem sequentially and for each tuple that satisfies the selection condition, it uses index scans to find the matching ones in table Order. Thus it will exhibit characteristics of both: the sequential scan and the index scan.

2.3 Evaluation Methodology

Our evaluation methodology is dependent on the use of performance counters that are available on the PA-8200 and R10000 microprocessors. We use a software library developed in the PARASOL research group to access the PA-8200 hardware counters on the HP V-Class. On the SGI Origin 2000, we make ioctl() system calls to directly access the MIPS R10000 hardware counters. In order to do the tests and get performance data via hardware counters, we also modified the PostgreSQL source code to one single executable. Due to the system configuration limitation in HP V-Class, we configured the buffer pool in shared memory to 512MB. Because of our limited resources and the required length of time, we set the database size for our test to 200MB. Note that 200MB is the size of flat raw data files for populating the database. The actual database size including the index file for primary keys would be roughly 400MB. On the whole, there are three orthogonal dimensions in our tests. The first dimension is type of TPC-H query (query number). The second dimension is the number of parallel queries simultaneously running. We vary the number of parallel query processes from 1 to 8. When more than one query process are configured, different query processes are assigned to different processors. The third dimension is the platform: either the HP V-Class or the SGI Origin 2000. For each configuration, we perform the same test four times and use the average values.

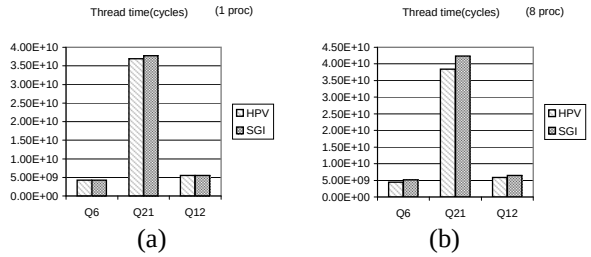


Figure 2. Thread Time in Cycles

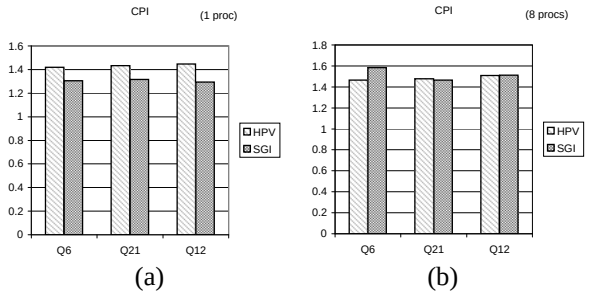


Figure 3. Cycles Per Instruction

3 Single (Same) Query Execution Results

In this section, we compare the performance of three queries on both HP V-Class and SGI Origin 2000. We examine execution time, CPI and data cache misses. For each part, we compare the performance data between the two platforms when running a single query process of Q6, Q21 and Q12 and also compare the performance between the two platforms when running eight query processes (all running the same type of query) simultaneously for each Q6, Q21 and Q12.

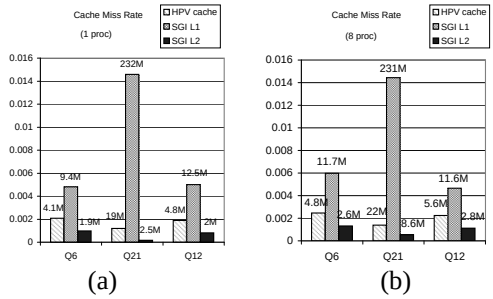


Figure 4. Data Cache Misses

3.1 Execution Time

Fig. 2 shows the thread execution time of Q6, Q21 and Q12 on HP V-class and SGI Origin 2000 in cycles. Thread time measures the total time that the thread of a process runs on the CPUs. It doesn't include the time when the process waits in the ready state to acquire a CPU. So it should be less than or equal to the wall-clock time. Fig. 2(a) shows that actually when one query runs on the system, the number of running cycles on both machines are very close. However, since the SGI Origin 2000 runs at a higher clock rate, the overall execution time on the SGI Origin 2000 is lower than on the HP V-Class. From Fig. 2(b), we see that when 8 query processes run on the system, SGI Origin 2000 actually uses much more cycles to finish the query. The reason is that as the number of query processes increases, communication to keep the metadata consistent between different processors goes up. From Iyer et al. [4], we know the communication overhead is more expensive in SGI Origin 2000 than that in HP V-Class.

3.2 Cycles Per Instruction (CPI)

Fig. 3 shows the CPI on both platforms under the two situations. On the whole, CPI for these 3 queries are not high, ranging from 1.3 to 1.6. The difference of CPI between the two machines in Fig. 3 (a) is mainly due to the little difference of the instruction event counters in the two machines, as explained above. By comparing Fig. 3 (a) and (b) we see that as the number of query processes increases, CPI on both machines increase. However CPI increases little on HP V-Class while more significant on SGI Origin. For example, on HP V-Class CPI for Q6 increases from 1.418 to 1.466 while on SGI Origin 2000 it increases from 1.305 to 1.585. This is consistent with the previous section. As more query processes are running, more synchronization activities between different processors introduce higher penalty in SGI Origin than in HP V-Class.

3.3 Data Cache Misses

Fig. 4 shows the data cache miss rate on both platforms under the two configurations. Just for convenience, we also mark the value of data cache misses on each bar. Let us look at Fig. 4 (a) first. For Q6, the L1 Dcache misses (9.4M) on SGI are only a little more than twice the Dcache misses (4.1M) on HP V-Class. Knowing that SGI L1 Dcache is only 32KB while HP V-Class Dcache is 2MB and that both have the same cache line size, the L1 Dcache in SGI Origin is quite efficient for Q6. The reason for this is that Q6 is a sequential query, which has very good spatial locality but poor temporal locality. As we explained in previous sections, there is record data, index data, metadata and private data in a DBMS. In a pure sequential query like Q6, no index data is used. The record data in a sequential query does not have temporary locality because of the optimizer. Private data and metadata both have temporal locality, which is the reason why the misses of L1 Dcache in SGI Origin are double the cache misses in HP V-Class. The locality characteristic also explains why the L2 Dcache in SGI Origin does not improve the performance as much for Q6 as for Q21. The longer cache lines (128-bytes) decrease the

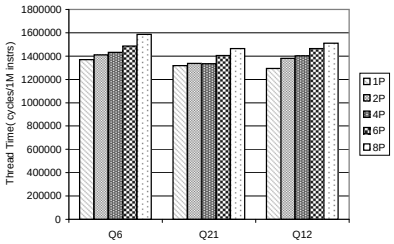


Figure 5. Thread Time on Origin 2000

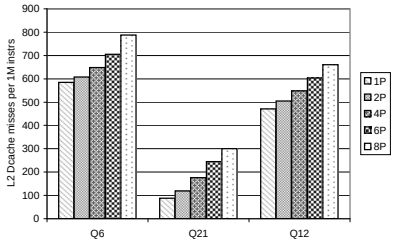


Figure 6. L2 Data Cache Misses on Origin 2000

cache misses for both Q6 and Q21 while the larger size of L2 cache has a smaller effect on cache misses for Q6 than for Q21. For Q21, the L1 Dcache misses in SGI Origin are roughly 12 times more than the Dcache misses in the HP V-Class. The reason for this is that Q21 is an index query, which has quite good spatial locality as well as moderate temporal locality. For example, the nodes close to the root in the index tree are likely to be reused later. In Q21 the L2 cache in SGI Origin greatly reduces the cache misses down to 2.5M, which is also much less than the corresponding Dcache misses in HP V-Class. The longer cache line and bigger cache size are obviously the major reason. For Q12, the situation is very similar to Q6, so it is more like a sequential query.

When comparing Fig. 4(a) and (b), we see that when 8 query processes are running in the systems the miss rates on HP V-Class and on SGI Origin increase. This is mainly due to the communication / coherence overhead which affects miss ratios of large caches (L1 miss ratio in SGI Origin remains unaffected) that tend to hold data for a long period of time.

4 Multiple (Diff) Query Execution Results

In this section we analyze the performance variation of both HP V-Class and SGI Origin 2000 as number of (different) query processes running in the system increases from 1 to 8.

4.1 Performance Analysis on the SGI Origin 2000

4.1.1 Thread time

Fig. 5 presents the thread time of 3 queries on SGI Origin with varied number of query processes. We report the thread time in number of cycles per million instructions. From Fig. 5, we see a clear trend that as number of query processes increases, the thread time increases for Q6, Q21 and Q12. The reason is that as more query processes run in the system, more communication, coherence and synchronization activities occur. Thus there are more dirty misses and stores to shared lines. As the number of processes (and processors) increases to 6 and 8, the increase in thread time becomes larger. The reason could be attributed to the interconnection network topology and the fact that shared memory requests from different processors are routed to the same node or a couple of different nodes which hold the shared memory for the DBMS.

4.1.2 L2 data cache misses

Of the cache misses, we only analyze L2 data cache misses here because this could make a significant difference in the execution time as the number of query processes increases. Fig. 6 presents the L2 data cache misses per 1M instructions for 3 queries. We see that as number of query processes increases from 1 to 8, L2 data cache misses increase significantly. Fig. 6 also shows that L2 data cache misses per 1M instructions of Q21 is much less than that of Q6 and Q12. As mentioned earlier, this is because Q21 is an index query and therefore has better temporal locality. We also like to point out that for the sequential queries (Q6 and Q12), the major contributor of L2 Dcache misses remains the normal cold start and capacity misses even when query processes increase from 1 to 8. However for the index query Q21, as query processes increase from 1 to 8, misses caused by communication becomes the major component of L2 Dcache misses.

4.2 Performance Analysis on the HP V-Class

4.2.1 Thread Time

Fig. 7 presents the thread times of 3 queries on HP V-Class with varied number of query processes. As for Fig. 5, we report the thread time as the number of cycles per million instructions. From Fig. 7, we see an overall trend of a very slow increase in the thread time as the number of query processes increase. Comparing with SGI Origin 2000 results (Fig. 5), we find that the lower communication overhead in the HP V-Class helps in keeping the increase in thread time to a minimum. We also notice that the largest increase in thread time results from an increase in the number of query processors from 1P to 2P. From 2P to 4P, the thread time even decreases for all three queries. This peculiarity can be attributed to the migratory enhancement in HP V-Class cache coherence protocol, which we will elaborate on in following subsections.

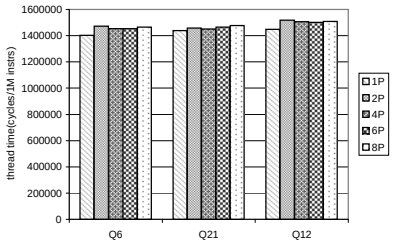


Figure 7. Thread Time on V-Class

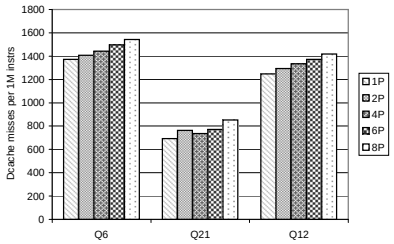


Figure 8. Data Cache Misses on V-Class

4.2.2 Data Cache Misses

Fig. 8 shows that the data cache misses in HP V-Class moderately increase as the number of query processes increases, which is consistent with results obtained on the SGI Origin 2000 (Fig. 6). Actually the increase of data cache misses in HP V-Class and SGI Origin 2000 are close, if we take into account the difference of two cache hierarchies and the event counter error. The moderate increase in Dcache misses shows that cold start and capacity issues still remain the major contributor to Dcache misses for all three queries even as the number of query processes increase from 1 to 8.

4.2.3 Memory Latency

Fig. 9 shows the memory latency of 3 queries on HP V-Class with varied number of query processes. Note that the memory latency reported here is the total time taken in completing a memory access without considering latency hiding (overlap of two memory accesses). That is, the event

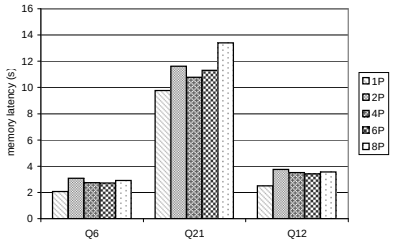


Figure 9. Memory Latency on V-Class

counter increments based on the number of open (waiting) memory requests at each system bus clock tick. In PA 8200 processor, there could be at most 10 simultaneously open requests.

From Fig. 9, we see that there is a big increase in memory latency as the number of query processes increases from 1 to 2. From 2P to 4P, the memory latency however decreases. This behavior is due to the migratory enhancement in the cache coherence protocol of HP V-Class. When the first query process (running on p1) reads a line within a database data page or index page from the memory, the cache line is in exclusive state in p1's cache. When a second query process (running on p2) sends out a read request for the same memory line, the control information has to be sent back from p1 to the home directory. Then the home directory sends the data line to p2 and converts the memory block to shared state. Compared to the speculative enhancement in SGI, this process takes longer time. When a third query process wants the memory line, however, the home memory directly sends the block to that processor since the memory line is in shared state now. Thus we see an increase in memory latency from 1P to 2P and a slight decrease in memory latency from 2P to 4P.

Although the migratory optimization in the HP V-Class seems to have a negative effect on the memory latency of the 3 TPC-H queries, the query processes can benefit from it for lock accesses as follows. When one query process needs to obtain a read lock, it first reads the lock information saved in the lock and transaction hash tables to check if it can acquire the lock. If it gets a positive response, it then updates the lock information. Without the migratory optimization, in the first step when the query process reads the lock information that is dirty in other processor, the data would be shared by the two processors. Then in the second step, if the process is able to get the lock (in our case this should be no problem), it needs to invalidate the shared copy and then update the lock information. With the migratory enhancement, however, in the first step the data block is already invalidated from the owner processor. Thus there is no write cache miss in the second step and the home directory will not be visited for a second time. The reason why we do not see this positive effect in Fig. 9 is that all runs ($>$ 1P) benefit from it, not just the 2P case. Moreover as we mentioned earlier, communication between different processes is just a small contributor to the overall cache misses anyway.

4.2.4 Context Switches

Fig. 10 shows both voluntary and involuntary context switches per 10M instructions during the query execution. An involuntary context switch occurs when the time slot assigned to the process is used up or another process with higher priority gets ready. A voluntary context switch is initiated by the process itself when it does I/O, synchronization, or issues system calls such as sleep() and select(). From Fig. 10, we see that when only one query process runs in the system, almost all the context switches are involuntary context switches. When two query processes run

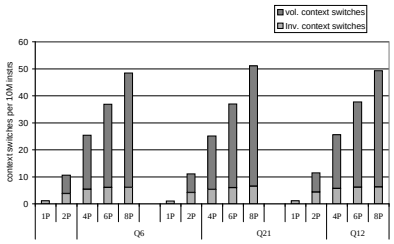


Figure 10. Context Switches on V-Class

in the machine, the total number of context switches increases abruptly and voluntary context switches are more than involuntary context switches. From 2P to 8P, the context switches increase rapidly and almost linearly. The majority of context switches beyond 2P are voluntary context switches. This behavior indicates that the lock management system in the DBMS is the reason. After studying the PostgreSQL code, we found that it adopts a backoff technique in acquiring a spinlock in both HP V-Class ported code and SGI Origin 2000 ported code. If a query process cannot get a spinlock, the process would delay some time, using the select() system call, and try again later. A select() call causes a voluntary context switch. When more query processes run in the system, it is more difficult to get a spinlock and thus more delays occur. This is the reason why voluntary context switches increase rapidly as the number of query processes increases. While backoff using the select() call is perfect for uniprocessor systems, it is not so efficient in multiprocessors because query processes do not share the same processor. This increases the wall time (response time) significantly and is detrimental to the service quality. For involuntary context switches, we also see a slow increase as number of query processes increases. Surprisingly, it seems that the number of context switches per 10M instructions is not a function of the type of query.

5 Conclusions

In this paper, we presented a detailed and in-depth memory system performance analysis and comparison of TPC-H benchmark characteristics on SGI Origin 2000 and HP V-Class multiprocessors. We not only examined the performance of single query process, but also examined the performance variation as multiple query processes simultaneously run in the system.

Our results show that the data cache performance of TPC-H queries differs depending on whether the query is more like a sequential query or more like an index query. Index queries express a somewhat bigger footprint but have better locality than sequential queries. Also, the two level data cache hierarchy in the SGI Origin 2000 performs better than the single cache level in the HP V-Class. The larger L2 cache size and longer line size have a more significant effect on index queries than on sequential queries. When only one query runs on the system it takes almost the same amount of cycles in HP V-Class and SGI Origin 2000. However when more query processes run simultaneously on the system, although the running cycles in both machines increase, it tends to increase more in SGI Origin 2000 than in HP V-Class. This indicates that the communication overhead is more expensive in SGI Origin 2000. Also, when multiple query processes were run on the two machines, voluntary context switches become predominant.

References

[1] L. Barroso, et al. "Memory System Characterization of Commercial Workloads", *Proceedings of the 25th Annual International Symposium on Computer Architecture*, pp3-14, June 1998.

[2] Q. Cao, et al., "Detailed Characterization of a Quad Pentium Pro Server Running TPC-D", *International Conference on Computer Design (ICCD)*, October 1999.

[3] HP 9000 V-Class Server Architecture, Second Edition, Hewlett-Packard Corporation, http://docs.hp.com/hpux/systems/#vclass. Accessed: September 15, 2000.

[4] R. Iyer, et al., "Comparing the Memory System Performance of the HP V-Class and SGI Origin 2000 Multiprocessors Using Microbenchmarks and Scientific Applications", *Proc. of the 13th ACM-SIGARCH International Conference on Supercomputing*, June 1999.

[5] R. Iyer, et al., "A Trace-driven Analysis of Sharing Behavior in TPC-C", *2nd Workshop on Computer Architecture Evaluation using Commercial Workloads*, 1999.

[6] K. Keeton, D. Patterson, Y. He, R. Raphael, and W. Baker, "Performance Characterization of a Quad Pentium Pro SMP Using OLTP Workloads", *Proc. Of the 25th Annual International Symposium on Computer Architecture (ISCA-98)*, pages 15-26, June 1998.

[7] J. Laudon and D. Lenoski, "The SGI Origin: A ccNUMA Highly Scalable Server", *Proceedings of the 24th International Symposium on Computer Architecture*, pp. 241-251, May 1996.

[8] A. Maynard, C. Donnelly and B. Olszewski, "Contrasting Characteristics and Cache Performance of Technical and Multi-user Commercial Workloads", *Proc. of the 6th Intl. Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pg 145-156, Oct. 1994.

[9] PostgreSQL User's and Developer's Guide, Postgres Global Development Group, Sep. 1996. http://www.postgresql.org/users-lounge/docs/6.5/postgres/

[10] M. Stonebraker and G. Kemnitz, "The POSTGRES Next Generation Database Management System", *Communications of the ACM*, October 1991.

[11] P. Trancoso, et al., "The Memory Performance of DSS Commercial Workloads in Shared-Memory Multiprocessors", *Proc. of HPCA-3*, pp 250-260, Feb. 1997.

[12] Transaction Processing Performance Council, TPC BENCHMARKTM H Standard Specification, Rev. 1.1.0, http://www.tpc.org/benchmark_specifications/Tpc-h/tpch110.pdf, Jan. 2000.

[13] S. C. Woo, et al., "The SPLASH-2 Programs: Characterization and Methodological Considerations", *Proceedings of the 22nd International Symposium on Computer Architecture*, pages 24-36, June 1995.