Memory Behavior in Scientific vs. Commercial Applications

Taegyoun Kim, Heejung Wang and Kangwoo Lee
Department of Computer and Telecommunications
Dongguk University
Voice : (02) 2260-3843 / FAX : (02) 2285-3343
E-mail : klee@cakra.dongguk.ac.kr

Abstract

As the market size of multiprocessor systems for commercial applications, parallel systems, especially cache-coherent shared-memory multiprocessors that are conventionally designed for scientific applications need to be tuned in different fashion to achieve the best performance for new application area. In this paper, indepth investigation on the memory behavior which is the primary cause for performance changes were made. We chose representative benchmarks in scientific and commercial application areas. After running execution-driven simulation for bus-based cache-coherent shared-memory multiprocessors, we experienced significant differences and conclude that the systems must be carefully and differently designed to achieve the best performance when they are built for distinct applications.

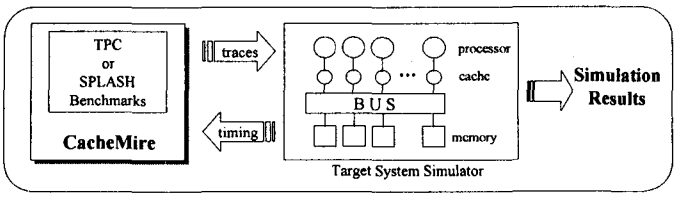
I. Introduction

Among diverse multiprocessor architecture models, cache-coherent shared-memory systems become cheap source of easy-to-program computing power. In addition to in scientific applications, another promising use of such machines is in commercial workloads such as OLTP (On-Line Transaction Processing) and DSS (Decision Support System) applications. In fact, many recently announced shared-memory systems including the Sequent STiNG [10] specifically target commercial markets.

But, even though hardware vendors normally present performance results of their machine for commercial workloads, there is no solid explanation supporting why the workloads perform the way they do. Particularly, just a little research effort have been made to understand the memory behavior of commercial workloads. This issue is quite essential since the performance of an application is mainly determined by how well the memory hierarchy is exploited. Furthermore, because of the increasing clock rates and computing powers of today's processors and continuous reduction in the price of memory, it will shortly be feasible for the databases to thoroughly reside in memory during the execution on such machines. Consequently, how well the memory hierarchy is exploited will determine the performance of the workload under consideration.

While there are too many literatures to mention for scientific application studies (and thus we do not enumerate excellent research works in this paper), relatively few results have been published for commercial applications. Suggs and Reynolds [14] presented the instruction fetch statistics of a uniprocessor for TPC-B [5] workloads. Torrellas et al [15] and Bhandarkar [4] evaluated commercial workload for the SGI multiprocessor and a DEC Alpha AXP system. In [11], the performance of shared-nothing architectures is investigated using synthetic workload and actual traces. Torrellas also presented his work in [16, 3] with data sharing analysis and related studies for DSS. The role of IO subsystems in shared-memory systems and a clustered system called SPAX [6] were investigated in [8], and [7], respectively. But, no paper dealt with in detail the difference of performances between scientific and commercial areas in a single literature.

In this paper, we compare the memory behavior of the applications in both areas. As workloads, we chose three benchmarks, FFT, LU and MP3D, from SPLASH [12] and SPLASH II [17] benchmark suites for scientific applications and a benchmark, TPC-B for commercial application. These benchmarks are coded in the C and parallelized using ANL macros [1]. Especially, to execute the transactions defined in TPC-B, we used a small parallelized DBMS engine called the EZDB [9]. The benchmarks are then run on a simulator called the CacheMire [2] for



[Fig. 1] Overview of simulation environment

execution-driven simulation. The target architecture is a bus-based shared-memory multiprocessor where cache coherency is maintained by Illinois protocol. Target model is diversely configured by changing cache block size and the number of processors.

The result of our simulation is sets of the number of cache misses. We made indepth analysis for the results and eventually found out that there are significant differences in curves plotted for the number of misses. To understand underlying causes that render the curves to appear so, the analysis of access patterns of processors to shared data objects are essential. What we found in consequence is that the architecture of shared-memory systems including cache memories and interconnects must be carefully designed when a system is built for commercial applications such as OLTP in different manner from the systems for scientific applications.

This paper is organized as follows. In Section II, the TPC-B with EZDB and SPLASH benchmarks are briefly introduced. Our simulation environment and simulation methodology is presented in Section III and simulation results and analysis are discussed in the subsequent section. Here, the comparison of memory behavior between the two categories of workloads are made in detail. Finally, this paper is closed with concluding remarks in Section V.

II. Scientific and Commercial Benchmarks

1. Scientific Benchmarks: SPLASH

1.1 FFT

FFT is a one-dimensional version of the radix-$\sqrt{n}$ six-step FFT algorithm. Two $\sqrt{n}$-by-$\sqrt{n}$ data arrays of complex numbers are x which contains the data points, and *trans* which contains the roots of unity. Two other data structures, umain and umain2, are only accessed by their home processors. FFT is not iterative and each element of x and *trans* is accessed constant times.

1.2 LU

LU application performs the LU decomposition of a dense matrix. There are two large matrices, A and L, of size N-by-N. A and L are stored column-wise in shared memory space. Columns i of A and L are statically allocated to the processor i *mod* P, where P is the number of processors. Each column of A is accessed by its Home only, whereas a column i of L is first modified once by the Home and then read by all the Homes of columns with indices greater than i.

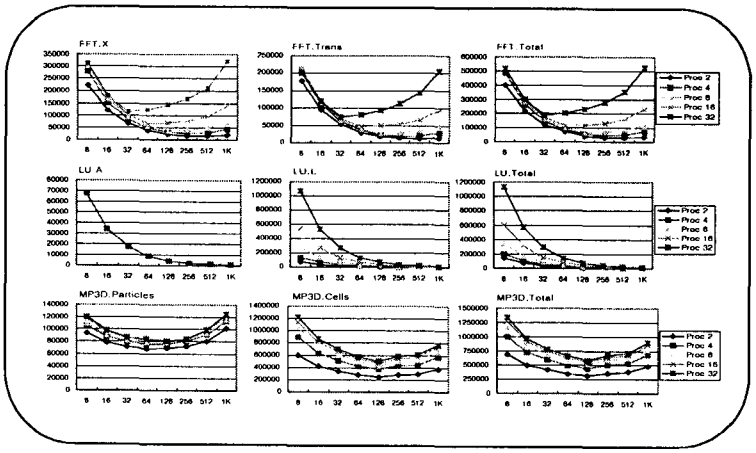
1.3 MP3D

MP3D repeatedly calculates the positions and velocities of particles during a preset number of time steps. The data set size, N, is the number of molecules in the one-dimensional array *Particles*. The size of each element of *Particles* is 36 bytes. 16 molecules in *Particles* array form a clump and the clumps are allocated to processors statically so that particle i is allocated to processor $(i/16)$ *mod* P, where P is the number of processors. *Cells* is a three-dimensional array of 48 byte elements, each of which is a location of the space where the molecules are moved. Unlike *Particles*, elements in *Cells* are shared by all processors. When a molecule is moved, the Cells component is modified.

2. Commercial Benchmarks: TPC-B

The TPC benchmarks are standard benchmarks widely used in industry to compare the performance of database systems. The simplest among them is TPC-B, in which a series of simple queries are executed on behalf of independent transactions.

There are four tables defined in TPC-B: *Teller*, *Branch*, *Account* and *History*. A *Teller* record is at least 100 bytes and contains fields *Tid*, *Bid* and *T_Balance*. A *Branch* record is at least 100 bytes and contains fields *Bid* and *B_Balance*. An *Account* record is at least 100 bytes and contains fields *Aid*, *Bid* and *A_Balance*. A *History* record is at least 50 bytes and contains fields *Aid*, *Tid*, *Bid*, *Delta* and *Time_stamp*. One of the restrictions is that the ratio between the numbers of records of types *Account*, *Teller* and *Branch* must be 100000:10:1.



[Fig. 2] Effect of Block size and number of processors in infinite cache for SPLASH benchmarks

III. Simulation Methodology

Fig. 1 illustrates the simulation environment. The simulation environment is based on the CacheMire testbench [17] which executes parallel applications to generate instructions and memory references. CacheMire executes codes of SPLASH benchmarks for scientific applications, while for commercial applications, it executes the code of EZBD running the script file and generates memory references. Instruction fetches, private and shared data accesses and synchronization operations (test-and-set) are monitored on the fly as execution progresses. Target architecture is a cache-coherent, single-bus, shared-memory multiprocessors with up to 32 processors. Caches are 4-way set associative with LRU (least-recently-used) replacement policy. Data coherence is enforced by the Illinois protocol.

To eliminate the effects of collision misses, we assume infinite caches. We have looked at cache block size between 8 and 1024 bytes.

VI. Simulation Results

1. Scientific Benchmarks

As is widely known, the curve of the number of cache misses over the changes of cache block size in shared-memory multiprocessors appears in the form of U. The simple reason is that, for small cache blocks, as the block size increases, the effect of prefetching dominates to reduce the number of misses caused by the accesses to contiguous memory locations. This is obvious in all graphs in Fig. 2. Especially, for LU where a whole column of an array of 1K bytes is accessed by a processor without intervention of other processors regardless of the number of processors, the number of cache misses is monotonously reduced.

For FFT, a block of data objects of size 1K/P bytes, where P is the number of processors, is accessed in burst manner also without the intervention of other processors. Therefore, for small number of processors, the shape of the curves is similar to the ones in LU while, for large number of processors, the number of misses start to increase. The main reason for the uprising curves is the false sharing misses that are caused by a large cache block contains some portion of data objects to be accessed by other processors but not by the local processor.

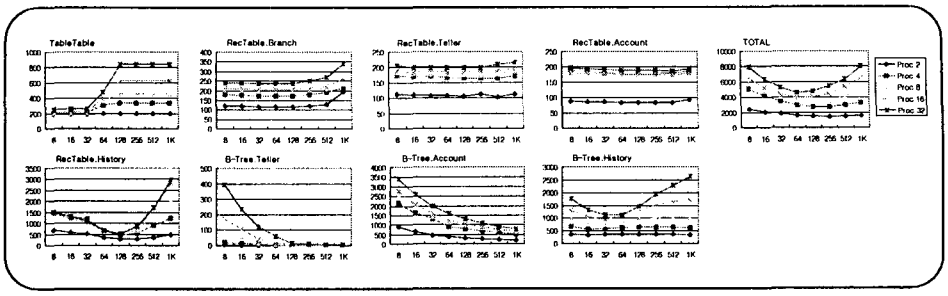
Unlike LU or FFT, MP3D presents its curves in the typical shape. It is because, each data element in two shared arrays are mostly accessed individually by randomly selected processors through entire execution. By the way, each element of two arrays occupies 36 bytes and 48 bytes, respectively, the number of misses reduces as the cache block size increases up to 64 or 128 bytes. For larger cache blocks, as mentioned earlier, false sharing effects start to appear and the curves rise up.

Finally, the effect of the number of processors is simple such that more processors enhance the degree of sharing and, therefore, increase the number of cache misses.

2. Commercial Benchmarks

2.1 Data Sharing (Infinite Caches)

Each graph in Fig. 3 shows the number of misses for one data structure with infinite caches as a



[Fig. 3] Effect of Block size and number of processors in infinite cache for TPC-B

function of the block size. The five curves in each graph correspond to systems with different number of processors. In the following we comment on the effects of the block size and number of processors.

Accesses to TableHead and RecordHead produce no coherence misses since they are read-only. Thus, we ignore these data structures along with the Branch B-Tree and private data.

2.1.1 Block Size

Looking at the curves for the total misses, we observe some gains due to spatial locality for smaller block sizes, but, for larger block sizes, the miss rate increases, a sign that false sharing dominates. False sharing is due to accesses to TableTable and to accesses to History and its B-Tree. Let's look at individual data structures.

The only interesting metadata is TableTable. Given a query and a database table name, the records in TableTable are searched to find the number of records and the number of fields in a record of the table. During a delete or insert operation the number of records in the table is updated in TableTable. In a TPC-B transaction, there are four queries and TableTable is searched four times per transaction. History is the only table where one record is inserted in a transaction and this causes cache misses on accesses to TableTable in consecutive queries. Because the size of TableTable is 104 bytes the number of false sharing misses grows dramatically for block sizes of 64 and 128 bytes, but remains constant for larger blocks.

Among the tables in the database, Branch and Teller are the smallest ones. One integer variable, B_id or T_id, of one record in Branch or Teller, respectively, is modified once in a transaction. Since the size of each record in these tables is 196 bytes, a block size smaller than 196 bytes does not affect the number of misses and all misses are true sharing misses. Larger blocks will cause false sharing misses. We observe more (false) sharing misses in Branch than in Teller for larger blocks, because there are only four records in Branch. Overall, the number of misses on accesses to Branch and Teller is small. Because there are 400,000 Account records and only one record is accessed (modified) in a transaction, the probability that an access to Account record hits in a cache is almost zero. Thus, most of times, one (cold) miss is counted in a transaction independently of the block size. In case of History, a new record is created and inserted into the table in a transaction and is never accessed again. On an insertion, a 196 byte record is filled, causing cold misses. Larger blocks reduce the number of cold misses. However, if the block is larger than 256 bytes, false sharing takes over.

During a transaction, multiple nodes in B-Trees are accessed to locate the record in a query. The B-Trees of Branch, Teller and Account are read-only. Each query in TPC-B updates the primary key field of a record of History, and thus its B-Tree is updated. The B-trees of Branch, Teller and Account have fewer misses for larger blocks. In particular, since the size of the B-Tree of Teller is only 800 Bytes, it fits into a 1024 byte cache block. Accesses to B-Tree of History, cause false sharing misses. In consequence, the number of misses ends up increasing for large block sizes.

2.1.2 Number of Processors

In scientific applications, shared data structures are partitioned and allocated among processors at the initial stage of a program. Careful data partition and allocation can minimize data communication. By contrast, in TPC-B, any processor can access any portion of the shared data. Thus, most write-shared

blocks are migratory. Hence the number of misses on every data structure is drastically affected by the number of processors.

The effect of the number of processors is less pronounced in Branch, Teller and Account than in TableTable, History and B-Trees. The reason is a single integer variable is modified and communicated in the first three data structures. On the other hand, even though one single integer variable is modified in TableTable and the B-Tree of History, the variables are communicated multiple times during a transaction. Finally, in History, a whole 196 byte record is modified causing more misses.

VI. Concluding Remarks

We have compared the memory behavior of both representative scientific and commercial benchmarks on cache-coherent shared-memory multiprocessors. and have presented the unique characteristics of each individual shared data structure including cache misses over various cache sizes, block sizes and numbers of processors. The result of our simulation were sets of the number of cache misses. We made indepth analysis for the results and eventually found out that there are significant differences in curves plotted for the number of misses. What we found is the architecture of shared-memory systems including cache memories and interconnects must be carefully designed when a system is built for commercial applications in different manner from the systems for scientific applications. In consequence, the cache block size under TPC-B is suggested to be larger than that used for scientific applications.

References

[1] Boyle, , J., et al, "Portable Programs for Parallel Processors, " Holt, Renihart and Winston Inc., 1987

[2] Brorsson, M., et al, "The CacheMire Testbench - A Flexible and Effective Approach for Simulation of Multiprocessors," Proc. of 26th Ann. Int'l Simulation Symp., pp. 41-49, Apr. 1993

[3] Cao, Q., et al, "Detailed Characterization of a Quad Pentium Pro Server Running TPC-D," Proc. 3rd Int'l Symp. on High-performance Computer Architecture, Feb. 1997

[4] Cvetanovic, Z and Bhandarkar, D., "Characterization of Alpha AXP Performance using TP and SPEC Workloads," Proc. of 21st Ann. Int'l Symp. on Computer Arch., pp. 60-70, Apr. 1994

[5] Gray, J., "The Benchmark handbook for Database and transaction Processing Systems," 2nd Ed. Morgan Kaufmann, 1993

[6] Hahn, W., et al, "SPAX: A New Parallel Processing System for Commercial Applications," Proc. of 11th Int'l Parallel Processing Symposium, pp. 744-749, Apr. 1997

[7] Hahn, W., Yoon, S., Lee, K. and Dubois, M., "Modeling and Evaluation of A New Cluster-Based System," Int'l Conf. on Algorithms and Arch. for Parallel Processing, pp. 177-184, Dec. 1997

[8] Lee, K., et al, "Bottleneck-Free Interconnect and IO Subsystem in SPAX," Int'l Conf. on Parallel and Distributed System," pp. 524-533, Dec. 1997

[9] Lee, K. and Kim, H., "Sharing Pattern Analysis of an OLTP Application," Proc. of Int'l Conf. on Computers, Communications and Systems, pp. 19-24, Nov. 6 1998

[10] Lovett, T. and Clapp, R., "STiNG: A CC-NUMA Computer System for the Commercial Market Place," Proc. of the 23rd Ann. Int'l Symp. on Computer Architecture, pp. 308-317, May 1996

[11] Marek, et al, "Performance Evaluation of Parallel Transaction Processing in Shared-Nothing Database Systems," Proc. of PARLE, pp. 295-310, May 1992

[12] Singh, J. P., et al, "SPLASH: Stanford Parallel Applications for Shared-Memory," Computer Architecture News, 20 (1): 5-44, Mar. 1992

[13] Stonebraker, M and Kemnits, G, "The POSTGRES Next Generation Database management System," Comm. of the ACM, Jun. 1986

[14] Suggs, D., et al, "Constructing Multiprocessor Workload Characterization," Proc. 33rd ACM Ann. Southeast Conf., Clemson, Mar. 1995

[15] Torrellas, J., et al, "Characterizing The Caching and Synchronization Performance of Multiprocessor Operating System," Proc. of the 5th ASPLOS, pp. 162-174, Oct. 1992

[16] Trancoso, P., et al, "The Memory Performance of DSS Commercial Workloads in Shared-Memory Multiprocessors," Proc. of 23rd Ann. Int'l Symp. on Computer Architecture, 1996

[17] Woo, S. C., et al, "The SPLASH II Programs: Characterization and Methodological Consideration," Proc. of 22nd Ann. Int'l Symp. on Computer Architecture," pp. 24-36, May 1995