Understanding the Memory Performance of Data-Mining Workloads on Small, Medium, and Large-Scale CMPs Using Hardware-Software Co-simulation

Wenlong Li, Eric Li, Aamer Jaleel, Jiulong Shan, Yurong Chen, Qigang Wang, Ravi Iyer,
Ramesh Illikkal, Yimin Zhang, Dong Liu, Michael Liao, Wei Wei, John Du

Intel Corporation
{ wenlong.li, eric.q.li, aamer.jaleel, yurong.chen, yimin.zhang }@intel.com

Abstract

With the amount of data continuing to grow, extracting "data of interest" is becoming popular, pervasive, and more important than ever. Data mining, as this process is known as, seeks to draw meaningful conclusions, extract knowledge, and acquire models from vast amounts of data. These compute-intensive data-mining applications, where thread-level parallelism can be effectively exploited, are the design targets of future multi-core systems. As a result, future multi-core systems will be required to process tera-byte-level workloads. To understand the memory system performance of data-mining applications, this paper presents the use of hardware-software co-simulation to explore the cache design space of several multi-threaded data mining applications. Our study reveals that the workloads are memory intensive, have large working-set sizes, and exhibit good data locality. We find that large DRAM caches can be useful to address their large working-set sizes.

1. Introduction

Increasing power densities and diminishing returns from deeper pipelines have eliminated increasing clock frequency as means of achieving higher performance. As a result, the design space of future high-performance processors has shifted to chip multiprocessors (CMPs) [2, 12, 22]. Incorporating multiple cores on the same die presents many memory hierarchy design choices to processor architects; making the already difficult design decision even more difficult. Understanding the memory system requirements of emerging applications is crucial to pick the appropriate design choice. This paper presents the memory system requirements of emerging parallel data-mining applications on small, medium, and large-scale CMPs.

Data mining is an emerging software technology that can extract useful information from massive amounts of raw data. The wide spread use of data-mining techniques in fields such as medicine, finance, and entertainment require future computing platforms to be able to process tera-byte-level workloads. Identifying the memory system requirements and the suitable memory system organization requires extensive simulation studies. However, current software based simulation techniques are slow and do not allow for fast full-run exploratory studies.

In response to the slow speeds of existing software-based simulators, FPGA-based hardware simulation has been proposed [21]. However, resource constraints on FPGAs and FPGA programming can sometimes create inflexibility when using the FPGA simulation infrastructure. Alternatively, both the flexibility of software and the speed of hardware can be combined to improve simulation speeds. For example, combining a full system simulator with VMX support and an FPGA-based simulator can enable full system co-simulation that is several orders of magnitude faster than traditional simulators (e.g. 30-50 MIPS). As a result, rather than only simulating small regions of an application, hardware-software co-simulation can now support simulating applications when run to completion. Simulation of applications run to completions is valuable as it supports changing application phase behavior and also helps choose representative regions for detailed simulation. This is extremely important now as emerging applications execute trillions of instructions.

For purposes of this paper, we present the use of hardware-software co-simulation to characterize the memory behavior of parallel data-mining applications when run to completion. Using a full-system simulator that uses VMX to directly execute simulated applications on the native machine and an FPGA-based cache simulator that directly connects to the processor front-side bus, we characterize the memory performance of parallel data-mining workloads on small, medium, and large-scale CMPs. Our study makes the following contributions:

- A detailed cache sensitivity study (by varying the cache size from 4MB to 256MB) reveals that the workloads have working-set sizes of 32MB or more (e.g. 256MB). Such large working set sizes indicates the need for large caches. Since large SRAM based cache organizations can be expensive to build, alternative cache organizations using DRAM (e.g. embedded DRAM, off-die DRAM caches, or 3D die-stacking) can be useful to reduce the latency and bandwidth to main memory.
- A cache line sensitivity study (by varying the line size from 64B to 4KB) reveals that cache performance improves with increasing cache line size. This behavior indicates that data-mining workloads exhibit good spatial locality and benefit from caches with large line sizes.
- A prefetching study revealed that the data-mining workloads benefit from hardware prefetching. We observed a 33% performance improvement by enabling the hardware prefetcher on an Intel® Xeon® machine.

Table 1: Input parameters and datasets

Workloads	Parameters	Size of Data Input
SNP	600k sequences, each with length 50	30MB, real datasets from HGBASE
SVM-RFE	253 tissue samples, each with 15k genes	30MB, real micro-array dataset on Cancer
RSEARCH	100MB database, search sequence size 100	100MB, real datasets from Gene bank
FIMI	990k transactions and mini-support=800	30MB, real dataset Kosarak: http://fimi.cs.helsinki.fi/data/
PLSA	two sequences in 30k length	60KB, real DNA sequences from Gene bank
MDS	220 pages with 25k sequences	4.1M, synthetic dataset from web search document
SHOT	10-min MPEG-2 video	200MB, 720x576 resolution
VIEWTYPE	10-min MPEG-2 video	200MB, 720x576 resolution

The rest of this paper is organized as follows. Section 2 describes the parallel data-mining workloads used in this study. Section 3 describes the hardware-software co-simulation methodology, Section 4 presents the experiment results, Section 5 presents related work and finally Section 6 provides conclusions.

2. Data Mining Workloads Overview

Data mining is an emerging software technology used to extract meaningful information and relationships from a large amount of raw data. Data mining takes advantage of advances in the fields of artificial intelligence (AI) and statistics by using algorithms such as pattern recognition, machine learning, decision making, and statistical modeling. This section briefly introduces some popular data mining algorithms and the associated workloads used in this study. Table 1 summarizes the workloads and their input sets.

2.1. Bayesian Network Workloads

A Bayesian Network (BN) is a probabilistic model that encodes probabilistic relationships between variables of interest. Learning the structure of a BN from data is the most important task of BN applications [19]. The goal of BN is to identify the statistic relationship between variables, and usually at the same time, the conditional probability distribution of each variable can also be determined.

Single nucleotide polymorphisms (SNPs) are DNA sequence variations that occur when a single nucleotide is altered in a genome sequence. The SNP workload [4] uses the hill climbing search method, which selects an initial starting point and searches that point's nearest neighbors. The neighbor that has the highest score is then made the new current point. This procedure iterates until reaching a local maximum score.

2.2. Classification and Prediction Workloads

Classification means assigning an object a pre-defined class/category/label. Prediction can be viewed as the construction and use of a model to access the class of an unlabeled sample. Classification and prediction have numerous applications including credit approval, medical diagnosis, performance prediction, and selective marketing.

A classifier training problem has an input dataset called the training set that consists of example records with a number of attributes. The objective of a classifier training algorithm is to use this training dataset to build a model such that the model can be used to assign unclassified records into one of the defined classes [10]. Support Vector Machines-Recursive Feature Elimination (SVM-RFE) [4] is one of feature selection method, which is extensively used in disease finding (gene expression). The selection is obtained by a recursive feature elimination process: at each RFE step, a gene is discarded from the active variables of a SVM classification model, according to some prior criteria.

A Cocke-Younger-Kasami (CYK) algorithm is a basic parsing algorithm for context-free language. RSEARCH [4] uses it for RNA secondary structure homolog searches. It decodes the Stochastic Context-Free Grammar (SCFG) to search a single RNA sequence against the database to find its homologous RNAs.

2.3. ARM Workloads

Association Rule Mining (ARM) is a commonly used data-mining problem. It is the process of analyzing a set of transactions to extract association rules. Frequent Itemsets Mining (FIM) is the basis of ARM. It tries to discover groups of items or values that co-occur frequently in a transactional data set. Many FIMI (FIM Implementation) algorithms have been proposed in literature, including FP-growth and Apriori-based algorithms, where FP-growth is proved to be much faster than the other FIM implementations. The FIMI workload [5] in use is based on the FP-Zhu package, which includes three stages: first-scan, FP-tree construction, and mining.

2.4. Optimization Workloads

Sequence alignment is an important tool in bioinformatics which is used to identify the similar and divergent regions between two sequences. PLSA [4, 15] uses a dynamic programming approach to solve sequence matching problem. It is based on the algorithm proposed by Smith and Waterman, which uses local alignment to find the longest common substring in sequences.

2.5. Text Mining Workloads

Text mining is the nontrivial extraction of implicit, previously unknown, and potentially useful information from a large amount of textual data. Web search engines find and rank documents based on maximizing relevance to the user's query, yet these systems still require users to read hundreds of closely-ranked documents to locate the relevant sections of text they are looking for. By synthesizing information common to retrieved documents, multi-document summarization can help users of information retrieval systems to find relevant documents with a minimal amount of reading. Multi-Document Summarization (MDS) workload [6] combines the advantages of the previous two methods, the graph-based ranking algorithm and the Maximum Marginal Relevance (MMR) algorithm, not only considering the similarities between a user's query and the main topic of the documents, but also minimizing the possible redundancy in the summary result.

2.6. Video Mining Workloads

Rapid advances in the technology of media capture and storage have contributed to an amazing growth of digital video content. As the growth of content generation and dissemination explodes, mining information from large video data becomes increasingly important, e.g., video surveillance, sport highlights detection, and home video retrieval, to name a few.

Parsing a video can be analyzed on four levels: frame, shot, scene, and the whole video sequence. To analyze the video content semantically, shot boundary detection is a prerequisite step. In the shot detection workload [6, 16], a color histogram of 48 bins in RGB space, 16 bins for each channel, and a pixel-wise difference feature, as a supplement to the color histogram, are used to introduce spatial information and infer the final shot information.

View type plays a critical role in video understanding. The View type workload [6, 17] uses playfield area and player size to determine four kinds of view type: global, medium, close-up, and out of view with each key frame [8]. The corresponding low-level processing includes playfield segmentation by the HSV dominant color of playfield and connect-component analysis. The dominant color of the playfield is adaptively trained by the accumulation of the HSV color histogram on a lot of frames.

We choose application parameters and datasets such that they represent realistic and time consuming executions. Table 1 shows the parameters and dataset sizes. Note that for some applications, like FIMI, only a limited number of real large datasets exist. For such workloads moderate or widely cited synthetic datasets are used.

3. Hardware-Software Co-Simulation

Simulation speed has always been a concern in the field of computer architecture. Complex performance simulators executing at speeds in the order of KIPS are quite common today. The speed issue is exacerbated when it comes to multi-core platform simulation.

FPGA-based emulation [3, 20] is now capable of emulating complex micro architectures within a single FPGA. However, we are still faced with the non-trivial task of interconnecting multiple FPGAs and bringing up the BIOS and operating system (OS) on top of that platform in order to emulate large-scale multi-core architectures.

Our simulation methodology combines the speed of FPGA emulation with the flexibility of software simulation. The basic idea is to use an execution driven simulation framework integrated with FPGA emulators to create a co-simulation framework. A combination of the native execution capabilities of simulators and fast emulation speeds provides a fast and flexible co-simulation environment delivering MIPS of system simulation throughput. Our platform is based on Intel's configurable cache emulator Dragonhead and its full system simulator SoftSDV [24].

3.1. Dragonhead

Dragonhead is an FPGA-based passive cache emulator. It snoops memory transactions from FSB and emulates a pre-defined cache algorithm. It can provide cache performance data in real time.

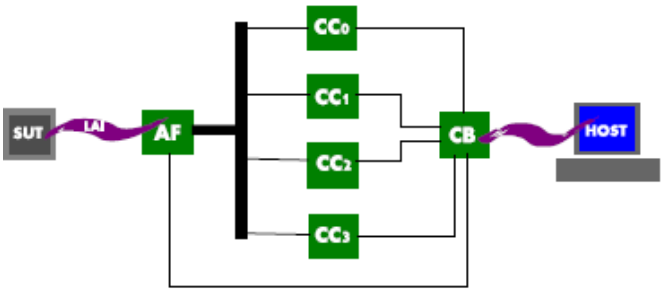

Figure 1: Dragonhead cache emulator

Figure 1 shows the block diagram of Dragonhead. Dragonhead has 6 FPGAs: AF, CC0, CC1, CC2, CC3, and CB. AF gets FSB transactions from LAI and sends them to CC after regulation. CC processes the coming requests and generates cache performance data. CB is responsible for configuring AF, CC, and collecting cache performance data. A host computer reads performance data from CB every 500 microseconds.

Since Dragonhead is based on an FPGA, different kinds of cache algorithms can be implemented. Currently, Dragonhead emulates cache sizes from 1M to 256M, cache line size from 64B to 4096B, with LRU replacement policy. Dragonhead emulates a shared LLC at a speed of 100MHz.

3.2 SoftSDV

SoftSDV is an execution-driven full system simulator. It provides functional models that can boot real BIOS, unmodified versions of an OS, and performance models that

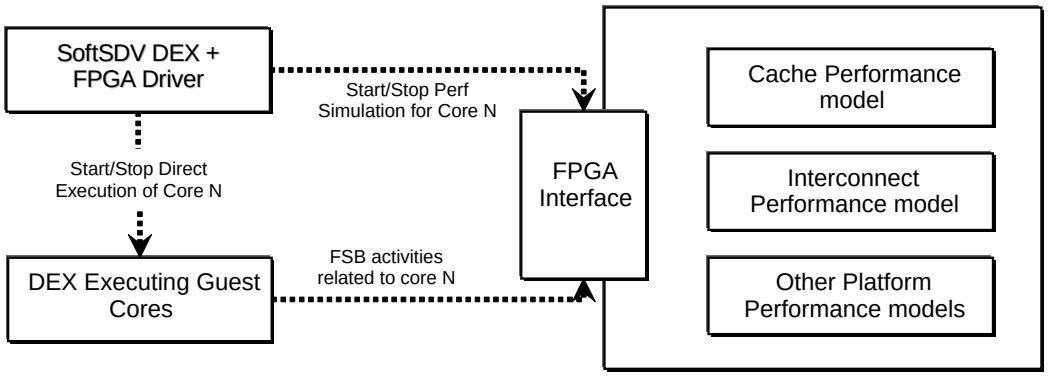

Figure 2: SoftSDV DEX-Dragonhead Interface

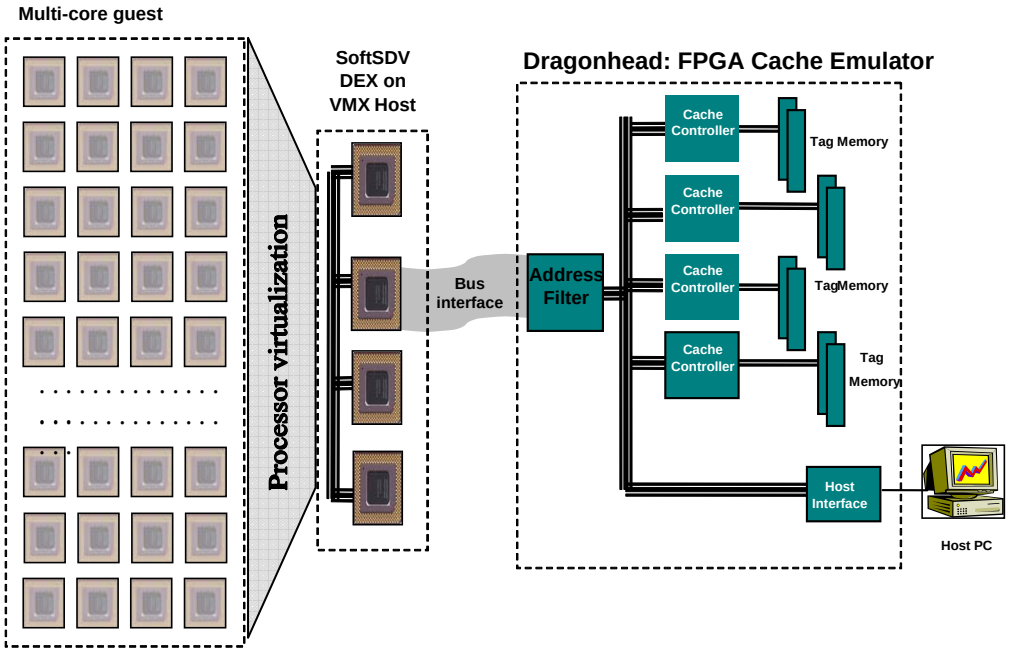

Figure 3: Hardware Software Co-simulation Platform

enable users to conduct various studies. One of the recent enhancements to SoftSDV execution -driven simulation is the introduction of native execution capabilities called DEX or Direct Execution. SoftSDV makes use of the VMX capabilities of Intel processors and directly executes simulated code on the underlying CPU, which offers a several magnitude speedup over traditional functional simulation. With the help of VMX, SoftSDV allows workloads to run on the processor natively for a specific duration; it can get the control back, save the processor state, and restart it from the saved state at a later time. This makes it possible for SoftSDV to schedule MP workloads on a UP system by time slicing the processor execution and exposing it as an MP system to the OS running on top of the virtual platform. This helps simulate multiple processors using a single processor or with fewer processors than the simulated guest.

However, DEX only supports the functional simulation part of the execution-driven simulation. In order for us to get the performance impact of future platform architectures that are being investigated, there needs to be performance models exercised during this direct execution. Due to the direct execution nature of DEX, however, performance simulations are not supported in DEX mode. Performance is done by switching back to binary-translation-based software simulation mode, which slows the simulation speed down into KIPS. Today DEX is used to fast forward to the point of interest before switching to the detailed execution-driven simulation with performance models. This enables the OS to be booted and workloads to be run in multi-core environments with up to 64 HW threads. Therefore, while DEX direct execution mode is highly effective in fast forwarding the functional simulation, it cannot be used to drive a performance model for performance evaluation.

Table 2: Workload characteristics

Workloads	IPC	Instruction Count (Billions)	% Memory Instructions	% Memory Read Instructions	DL1 Accesses / 1000 Inst	DL1 Misses / 1000 Inst	DL2 Misses / 1000 Inst
SNP	0.12	71.26B	50.75%	37.41%	508	12.01	7.77
SVM-RFE	0.87	37.02B	45.14%	43.64%	451	61.40	2.96
MDS	0.06	217.8B	49.34%	43.46%	493	51.00	18.95
SHOT	0.61	15.01B	53.85%	30.66%	538	18.86	4.07
FIMI	0.51	50.28B	47.10%	35.74%	471	15.99	3.76
VIEWTYPE	0.49	33.61B	49.02%	36.86%	490	31.77	3.56
PLSA	1.08	356.8B	83.10%	46.66%	831	4.60	0.18
RSEARCH	0.62	53.9B	42.3%	33.2%	423	10.65	0.72

3.3 HW-SW Co-simulation: Dragonhead + SoftSDV

As we described in the last two sections, Dragonhead is capable of running emulation at very high speeds, but is limited by the platform it is attached to. SoftSDV DEX provides a fast and highly flexible simulation environment, but lacks performance modeling support. We use a new co-simulation methodology to run SoftSDV in DEX mode while enabling it to drive a performance model through integrated Dragonhead emulation. By doing so, we exploit the flexibility and multi-core support of SoftSDV and the performance simulation speed of FPGA emulation. Three technologies play roles in this co-simulation environment: a flexible full system simulation environment provided by SoftSDV, DEX Native execution supported by VMX, and the fast performance emulation provided by Dragonhead.

The basic idea behind co-simulation is making the simulation and emulation environments communicate. Since Dragonhead is already snooping on FSB, we use the same FSB interface for this communication purpose (Figure 2). Some memory transactions are predefined as messages from SoftSDV to Dragonhead. SoftSDV can send the following information to Dragonhead via these memory transaction messages.

1. Start emulation
2. Stop emulation
3. Core-ID
4. Instruction retired
5. Cycles completed

While SoftSDV simulates several virtual cores on a single core by processor virtualization through scheduling, Dragonhead connected to the FSB performs cache emulation for these virtual processors. This means that a physical processor will execute the work for multiple logical cores in a sequential manner, scheduled by the DEX driver. During this direct execution time, Dragonhead sitting on the bus could continue to emulate the cache, and it is aware of the core ID that is being run natively in that time slot. This enables multi-core cache emulation on the FPGA platform.

Start and stop emulation allows the emulator to avoid memory accesses outside of the simulated workload. For example, the SoftSDV code and the host OS will also execute during the simulation, and by restricting the emulation to the window between start and stop, these accesses are excluded. Core ID allows the cache emulator to identify the memory accesses originating from different simulated cores and emulate accordingly. Instructions retired and cycles completed are useful in computing instruction and time synchronized statistics such as MPI and miss rate. The time and instruction synchronization are important, since the simulation and emulation run in two separate time domains.

A pictorial representation of the integration of SoftSDV and Dragonhead is shown in Figure 3. The real platform we are using includes a DP system integrated with a Dragonhead emulator. SoftSDV DEX runs on this system to provide a virtual platform of cores scaled from 1 to 32. The whole system can run at 50MIPS and can provide cache performance data in real time.

4. Memory Characterization Results

We use hardware-software co-simulation to characterize the memory system performance of data-mining workloads on CMPs of different sizes. We present application instruction profile, cache size and cache line size sensitivity studies, and the sensitivity of the workloads to prefetching.

4.1. Experimental Methodology

To explore the memory system performance, we run the data-mining workloads on three simulated CMP systems: a small-scale CMP (8 cores, SCMP), a medium-scale CMP (16 cores, MCMP), and a large-scale CMP (32 cores, LCMP). All cores of the CMP are assumed to be single-threaded. The workloads are all compiled with aggressive compiler optimizations and are run to completion using representative data input sets (see Table 1).

4.2. Workload Characteristics

Table 2 presents the application characteristics of the workloads run to completion in single-threaded mode. The data was collected on a Pentium 4 machine (8KB L1 cache, 512KB L2 cache) using the Intel VTune® analyzer [13]. For each application, we present the total number of instruc-

tions executed and the distribution of instructions that reference memory. The instruction profile indicates that roughly half of the instructions (as many as 83% for the PLSA) reference memory. We also observe that memory read instructions constitute 56-96% of total memory instructions. Since these workloads intensively read and analyze data to discover meaningful patterns or relationships, the large share of memory instructions, especially memory read instructions, is to be expected.

In addition to the instruction profile, Table 2 also provides the L1 and L2 cache performance. For each level of the cache, we present the number of accesses and misses per 1000 instructions. Except for PLSA and RSEARCH, which have relatively low L2 cache miss rates, all other applications experience high L2 cache miss rates. Comparing the IPC numbers in Table 2, we observe that applications with high L2 cache miss rates have low IPC performance: for example, 0.06 for MDS, and 0.12 for SNP. To gain better insight on the memory system requirements of these applications, we conduct a cache sensitivity study to determine the suitable cache size.

4.3. LLC Performance of SCMP, MCMP, and LCMP

We evaluate the cache performance of multi-threaded parallel workloads on different sized CMPs. Figure 4 presents the LLC performance (in terms of cache misses per 1000 instructions) with a 64 byte line size on an 8-core small-scale CMP. The x-axis presents the cache size and the y-axis presents the misses per 1000 instructions.

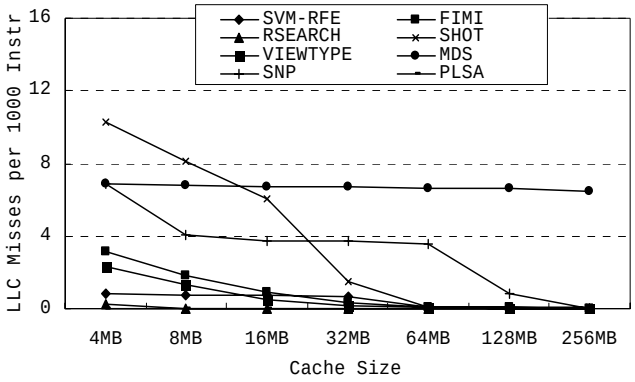

Figure 4: LLC miss per 1000 instructions on SCMP (8 cores)

The figure shows that most applications (besides MDS) benefit from increasing the cache size. MDS receives no benefit with the simulated cache sizes because one of its frequently referenced data structures is a sparse matrix of 300MB. On the other hand, of the workloads that benefit from increasing the size of the cache, we observe that SNP has two working set sizes, first at 16MB and the next at 128MB, SHOT has a 32MB working set size, VEWTYPE and FIMI have 16MB working set sizes, while SVM-RFE, PLSA, and RSEARCH each have 4MB working set sizes[1]. The large working-set size for some of the workloads indicates the need for large on-chip caches. Since large SRAM cache organizations can be expensive to build, alternative cache organizations using DRAM (e.g. embedded DRAM (eDRAM), off-die DRAM-based large last-level caches, 3D die-stacking) are essential to reduce the latency and bandwidth to main memory.

Figures 5 and 6 illustrate the cache performance of scaling the number of threads per workload to simulate an MCMP (16 cores) and an LCMP (32 cores) respectively. The cache behavior of these workloads based on thread-scaling groups them into two possible categories: (a) all threads share a primary data structure (b) all threads mostly work on private data structures and a small shared data structure is used for book keeping.

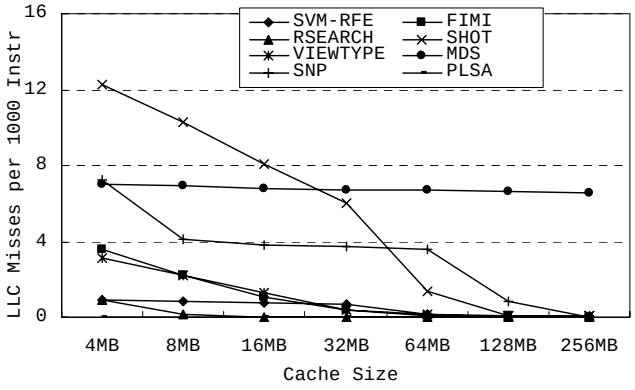

Figure 5: LLC miss per 1000 instructions on MCMP (16 cores)

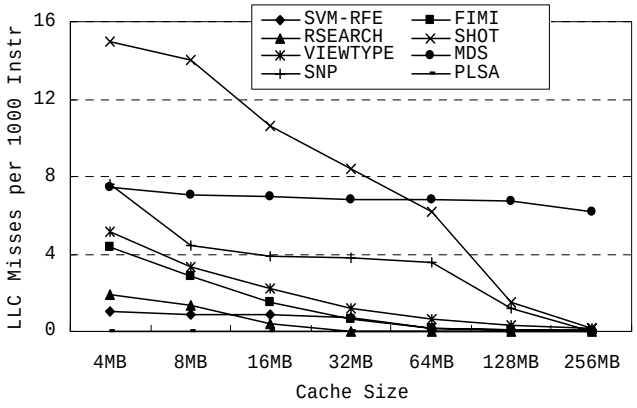

Figure 6: LLC miss per 1000 instructions on LCMP (32 cores)

From the figure, MDS, SVM-RFE, and SNP fall into the first category as their cache performance does not vary with increasing thread count. Such behavior indicates that the workloads operate on a global working set and additional threads contribute minimal private data.

The working set size of FIMI, RSEARCH, and PLSA increases with increasing thread count. This is because each additional thread adds additional private data to the work-

[1]SVM-RFE behaves different from [14] due to data blocking optimizations.

load data footprint, hence increasing the overall working-set of the workload. The increase in footprint translates into decreased cache performance. For example, all threads in FIMI share a read-only global tree structure, and each thread operates on a portion of the tree. Additionally, each thread also allocates private data to compute and store the temporary mining results. For these workloads, the footprint of the global working set is much larger than that of the additional private per-thread data. As a result, the total footprint does not scale linearly with the number of threads, and the additional per-thread private data footprint causes 20-30% increase in the number of cache misses.

On the other hand, for SHOT and VIEWTYPE applications, each thread of the workload operates on a very small shared data structure and relatively large private working set (about 4MB per thread for SHOT and 1MB per thread for VIEWTYPE). As a result, the total working set increases almost linearly with the number of threads, and the cache performance worsens when scaling the number of cores. For example, with a 32MB cache, the cache miss rate is increased by about 50-60% for SHOT and VIEWTYPE when scaling the core count from 8 to 16 cores. On MCMP, the working set for FIMI and RSEARCH is about 16MB and 8MB (similar to SCMP), while for SHOT and VIEWTYPE applications, the working set is twice that of SCMP, that is about 64MB and 32MB respectively.

Figure 6 further confirms our observations and analysis in comparing Figure 4 and 5. Like in Figure 4, PLSA, MDS, SVM-RFE, and SNP do not show obvious changes in cache performance when scaled to LCMP. Based on our understanding of these workloads, we believe that the cache performance of these workloads will not scale on a large number of cores, even on 128 cores. For these workloads, a small LLC, such as 8MB, will deliver a good memory subsystem performance. On the other hand, for FIMI and RSEARCH, cache misses increase when scaling the core count from 16 to 32, due to an increased working set. On LCMP (32 cores), the working set for FIMI and RSEARCH is about 32MB and 16MB, respectively, compared to MCMP. The working set for these two workloads is increased nearly linearly with the number of cores. Based on these observations and the knowledge of these workloads, we project that the working set of these two workloads will further increase as core numbers increase, and their working set will exceed 32MB on 128 cores. Thus, a large DRAM cache can provide good memory subsystem performance. As for SHOT and VIEWTYPE, the working set scales linearly with the number of cores as each thread maintains a large private working set and little data is shared among threads. On LCMP, the working set of SHOT and VIEWTYPE is about 128MB and 64MB; therefore, they are certain to be good candidates for large DRAM caches.

4.3. Cache Line Size Scaling Impact

Figure 7 presents the cache performance of the workloads with different cache line sizes on LCMP with a 32MB LLC. We observe that all workloads achieve better cache performance when the cache line size is increased. Particularly, we observe that SHOT, MDS, SNP, and SVM-RFE almost get linear miss reductions (around 1/3 to 1/4) from 64B to 256B, but the trend diminishes quickly from 256 to 1024 bytes. For other workloads, the improvement is not that significant. The workloads that favor large cache line sizes are those that exhibit a streaming constant stride access pattern. For example, SHOT iterates on a large array with a constant stride, and the MDS application iterates through a large compressed matrix with constant stride. The figure shows that a 256 byte cache line provides the maximum benefit, hence we expect that a 256-byte line size is sufficient for large DRAM caches.

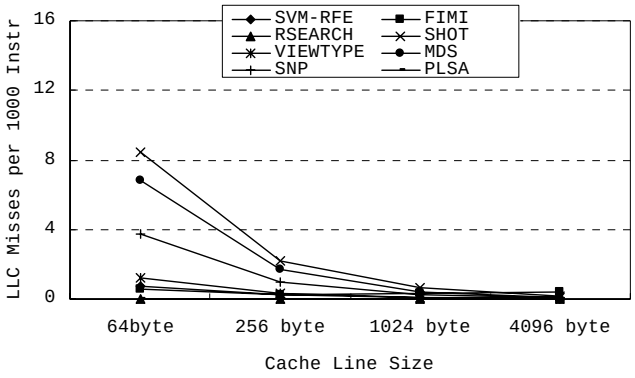

Figure 7 Line size sensitivity on LCMP with 32MB LLC

Based on the cache line size sensitivity study and our understanding of these workloads, we expect the large amount of spatial locality exhibited by the workloads to be captured by a hardware stride prefetcher. The next section, evaluates the performance of hardware prefetching for the workloads on a real machine.

4.4. Effects of Hardware Prefetching

Figure 8 presents the performance benefit of hardware prefetching measured on a 16-way Unisys machine. The Unisys machine is an IA-32 3.0 GHz Intel Xeon® shared-memory multiprocessor system that supports stride-based hardware prefetcher. We now present the benefits of hardware prefetching for the workloads when run in single-threaded and 16-threaded mode when compared to the same system with the hardware prefetcher turned off.

Since these workloads have large working-set sizes with algorithms that exhibit linear access patterns (in forward and backward directions), a hardware prefetcher can effectively prefetch the required data into the processor cache, hide memory latency and improve performance. Figure 8 shows that the performance of all applications is considerably improved. For some workloads, such as VIEWTYPE, FIMI, PLSA, RSEARCH, SHOT, and SVM-RFE, the parallel version benefits more from hardware prefetching than the serial version. This is most likely due to multiple data streams recognized by the prefetcher and the sufficient

bandwidth available for the issue of prefetches. For other workloads, such as SNP and MDS, parallel versions of these workloads impose higher contention on the bandwidth than serial versions due to high cache miss rates. As a result, little bandwidth is available for hardware prefetching. Thus, for such workloads, the performance benefit with hardware prefetching is better for serial versions of the workload.

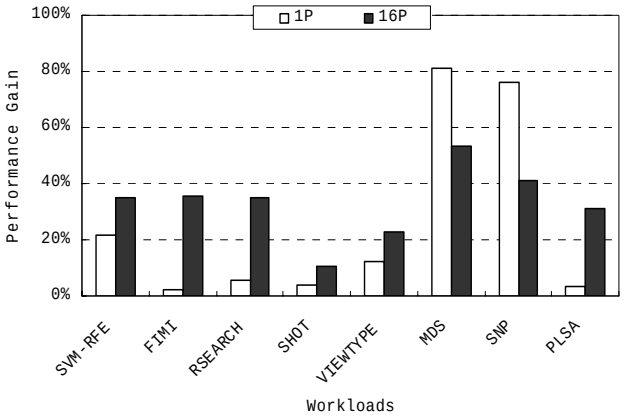

Figure 8: Performance gain of hardware prefetch

In summary, the workloads evaluated in this study are a representative set of memory-intensive data-mining workloads. A cache size sensitivity study revealed that most workloads have large working set sizes and can benefit from large DRAM caches. When increasing the thread count of each workload, we observe two behaviors. First, workload threads that reference only global shared data structures experience marginal increases in cache misses. On the other hand, workload threads that reference mostly private data structures show significant increases in cache misses. Based on our knowledge of the algorithm used by the workloads, we believe that 5 of the 8 workloads will benefit from a large DRAM cache when scaled to a 128-core CMP. Finally, since workloads exhibit good spatial locality, large cache line sizes and hardware prefetching can be helpful in improving cache performance.

5. Related Work

As data-mining workloads become mainstream applications running on future processors, it is critical to understand the performance characteristics of these workloads on future CMPs. Both industry and academia have invested resources in characterizing the scalability and performance of emerging data-mining applications. Chen et al. investigated the scalability and performance of data-mining bioinformatics applications [4]. Zambreno et al. composed the data-mining benchmark suite, Minebench, and analyzed important performance characteristics on an 8-way shared memory machine [26]. Jaleel et al. [14] characterized the LLC performance and data-sharing behavior of parallel bioinformatics workloads.

Recent studies have also investigated the design policies for the cache hierarchy of CMPs. Liu et al. discussed the tradeoffs of implementing shared/private caches and proposed a mechanism of allocating multiple last-level private caches to one core of a CMP [18]. Chishti et al. presented mechanisms for optimizing replication, coherence communication, and for exploiting unused cache space in CMPs [7]. Zhang et al. proposed victim replication to achieve the benefits of private caches with shared caches [27]. Hsu et al. [11] explored the cache design space for large-scale CMPs.

Existing work has also focused on understanding the memory behavior and performance of parallel workloads. Abandah et al. proposed a configuration-independent approach to analyze the working set, concurrency, and communication patterns, as well as the sharing behavior of shared memory applications [1]. Jaleel et al. characterized the memory and sharing behavior of parallel workloads using the binary instrumentation tool, Pin [14]. Barroso et al. characterized the memory system behavior using ATOM [23], performance counters on an Alpha 21164, and the SimOS simulation environment. Woo et al. characterized several aspects of the SPLASH-2 benchmark suite [25] using an execution-driven simulation with the Tango Lite [9] tracing tool. Nurvitedhi et al. used an FPGA-based cache model (PHA$E) that connects directly to the front-side bus to analyze the shared vs. private L3 cache behavior of SPECjAppServer and TPC-C [21].

Our work differs from prior work in that we use hardware-software co-simulation to conduct full-run memory performance studies of parallel workloads on small, medium, and large-scale CMPs. We use a full-system simulator that directly executes simulated applications on the native machine using VMX support. Cache performance analysis is done via an FPGA-based cache simulator that is directly connected to the front-side bus. Unlike existing software based simulation techniques, the hardware-software co-simulation methodology can simulate workloads at speeds of 30-50 MIPS.

6. Conclusions

The widespread use of data-mining techniques in emerging fields such as medicine, finance, and entertainment requires future computing platforms to be able to process terabyte-level workloads. This paper presents the memory system requirements of parallel data mining workloads on small, medium, and large scale CMPs. We use a hardware-software co-simulation infrastructure that combines the flexibility of full-system software simulation with an FPGA-based cache emulator. The co-simulation infrastructure simulates applications to completion at speeds of 30-50 MIPS. Our studies with the data-mining applications reveal that they have large working sets and can benefit from large DRAM based last-level caches. Additionally, the workloads benefit from large cache lines due to the presence of high spatial locality. Prefetching studies demonstrated up to 33% improvements in performance on real machines.

7. References

[1] GA Abandah, and ES Davidson, "Configuration Independent Analysis for Characterizing Shared-Memory Applications," in Proceedings of the 12th. International Parallel Processing Symposium (IPPS), Orlando, Florida, 1998.

[2] AMD Inc., "AMD Multi-core Processors," http://multicore.amd.com/en

[3] N. Chalainanont, E. Nurvitadhi, R. Morrison, L. Su, K. Chow, SL Lu, and K. Lai, "Real-time L3 Cache Simulations Using the Programmable Hardware-Assisted Cache Emulator (PHA$E)," Sixth Annual Workshop on Workload Characterization, October 27, 2003.

[4] Y. Chen, Q. Diao, C. Dulong, C. Lai, W. Hu, E. Li, W. Li, T. Wang, and Y. Zhang, "Performance Scalability of Data-Mining Workloads in Bioinformatics," Intel Technology Journal, Volume 09, Issue 02, May 19, 2005.

[5] Y. Chen, C. Dulong, C. Lai, W. Hu, E. Li, W. Li, J. Shan, T. Wang, and Y. Zhang, "Performance Scalability of Parallel Game-Tree Search and Frequent Itemsets Mining Applications," Technical Report.

[6] Y. Chen, Q. Diao, C. Dulong, W. Hu, E. Li, W. Li, J. Shan, T. Wang, and Y. Zhang, "Performance Scalability of Media Mining Applications," Technical Report.

[7] Z. Chishti, M. Powell, and TN. Vijaykumar, "Optimizing Replication, Communication, and Capacity Allocation in CMPs," in Proceedings of the 32nd International Symposium on Computer Architecture (ISCA), Wisconsin, Madison, 2005.

[8] A. Ekin, A. M. Tekalp, and R. Mehrotr, "Automatic soccer video analysis and summarization," IEEE Trans. on Image processing, 12(7):796-807, 2003.

[9] S. Goldschmidt and J. Hennessey. "The Accuracy of Trace-Driven Simulations of Multiprocessors." Tech Rep. CSL-TR-92-546, Stanford University, Sept. 1992.

[10] J. Han and M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers, August 2000.

[11] L. Hsu, R. Iyer, et al., "Exploring the Cache Design Space for Large-Scale CMPs," 1st Workshop on Design, Architecture and Simulation of CMP (dasCMP), Nov 2005.

[12] Intel Inc., "Intel Multi-core Platform," http://www.intel.com/technology/computing/multi-core/

[13] Intel Corp. VTune performance analyzer. Available at http://www.intel.com/software/products/VTune

[14] A. Jaleel, M. Mattina, and B. Jacob, "Last-level cache (LLC) performance of data-mining workloads on a CMP-A case study of parallel bioinformatics workloads", in Proceedings 12th International Symposium on High Performance Computer Architecture (HPCA 2006), Austin TX, February 2006.

[15] E. Li, C. Xu, T. Wang, L. Jin, and Y. Zhang, "Parallel Linear Space Algorithm for Large-scale Sequence Alignment," in Europar'05, Lisbon, Portugal, September 2005.

[16] E. Li, W. Li, T. Wang, N. Di, C. Dulong, and Y. Zhang, "Towards the Parallelization of Shot Detection-A Typical Video Mining Application Study," ICPP 2006, Columbus, Ohio, USA, August 14-18, 2006.

[17] W. Li, E. Li, C. Dulong, Y.K. Chen, T. Wang and Y. Zhang, "Workload Characterization of a Parallel Video Mining Application on a 16-Way Shared-Memory Multiprocessor System," to appear at IISWC 2006.

[18] C. Liu, A. Sivasubramaniam, M. Kandemir, "Organizing the Last Line of Defense before Hitting the Memory Wall for CMPs," in Proceedings of the 6th International Conference on High Performance Computer Architecture (HPCA), Madrid, Spain, 2004.

[19] K. Murphy, "The Bayes Net Toolbox for Matlab," Computing Science and Statistics, vol. 33, 2001.

[20] AK. Nanda, KK. Mak, K. Sugavanam, R. Sahoo, V. Soundararajan, T. B. Smith, "MemorIES: A Programmable, Real-Time Hardware Emulation Tool for Multiprocessor Server Design," ASPLOS 2000, pp 37-48.

[21] E. Nurvitadhi, N. Chalainanont, and SL. Lu, "Characterization of L3 Cache Behavior of SPECjAppServer2002 and TPC-C," in Proceedings of the 19th International Conference on Supercomputing (ICS), Boston, Massachusetts, 2005.

[22] Platform 2015: Intel Processor and Platform Evolution for the Next Decade: ftp://download.intel.com/technology/computing/archinnov/platform2015/download/Platform_2015.pdf

[23] A. Srivastava and A. Eustace, "ATOM: A System for Building Customized Program Analysis Tools", Programming Language Design and Implementation (PLDI), 1994, pp. 196-205.

[24] R. Uhlig, R. Fishtein, O. Gershon, H. Wang, "SoftSDV: A Presilicon Software Development Environment for the IA-64 Architecture," Intel Technology Journal, Q4 1999.

[25] S. Woo, M. Ohara, E. Torrie, J. Singh, and A. Gupta, "The SPLASH-2 Programs: Characterization and Methodology Considerations," in Proceedings of the 22nd International Symposium on Computer Architecture (ISCA), Santa Margherita Ligure, Italy, 1995.

[26] J. Zambreno, B. Ozisikyilmaz, J. Pisharath, G. Memik, and A. Choudhary "Performance Characterization of Data Mining Applications using MineBench," in Proceedings of the 9th Workshop on Computer Architecture Evaluation using Commercial Workloads (CAECW-9), February 2006.

[27] M. Zhang and K. Asanovic, "Victim Replication: Maximizing capacity while Hiding Wire Delay in Tiled CMPs," in Proceedings of the 32nd International Symposium on Computer Architecture (ISCA), Wisconsin, Madison, 2005.