Available online at www.sciencedirect.com
ScienceDirect
Journal of Systems Architecture 54 (2008) 816–828
JOURNAL OF SYSTEMS ARCHITECTURE
www.elsevier.com/locate/sysarc

Memory hierarchy performance measurement of commercial dual-core desktop processors

Lu Peng [a,*], Jih-Kwon Peir [b], Tribuvan K. Prakash [a], Carl Staelin [c], Yen-Kuang Chen [d], David Koppelman [a]

[a] Electrical and Computer Engineering, Louisiana State University, Baton Rouge, LA 70803, United States
[b] Computer and Information Science and Engineering, University of Florida, Gainesville, FL 32611, United States
[c] Hewlett-Packard Laboratories, Technion City, Haifa 32000, Israel
[d] Architecture Research Laboratory, Intel Corporation, Santa Clara, CA 95052, United States

Received 6 April 2007; received in revised form 14 December 2007; accepted 15 February 2008
Available online 29 February 2008

Abstract

As chip multiprocessor (CMP) has become the mainstream in processor architectures, Intel and AMD have introduced their dual-core processors. In this paper, performance measurement on an Intel Core 2 Duo, an Intel Pentium D and an AMD Athlon 64 × 2 processor are reported. According to the design specifications, key derivations exist in the critical memory hierarchy architecture among these dual-core processors. In addition to the overall execution time and throughput measurement using both multi-program-med and multi-threaded workloads, this paper provides detailed analysis on the memory hierarchy performance and on the performance scalability between single and dual cores. Our results indicate that for better performance and scalability, it is important to have (1) fast cache-to-cache communication, (2) large L2 or shared capacity, (3) fast L2 to core latency, and (4) fair cache resource sharing. Three dual-core processors that we studied have shown benefits of some of these factors, but not all of them. Core 2 Duo has the best performance for most of the workloads because of its microarchitecture features such as the shared L2 cache. Pentium D shows the worst performance in many aspects due to its technology-remap of Pentium 4 without taking the advantage of on-chip communication.

Keywords: Multi-core processor; Core 2 Duo; Memory; Performance evaluation

1. Introduction

Due to advances in circuit integration technology and performance limitations in wide-issue, super-speculative processors, chip-multiprocessor (CMP) or multi-core technology has become the mainstream in CPU designs. It embeds multiple processor cores into a single die to exploit thread-level parallelism for achieving higher overall chip-level instruction-per-cycle (IPC) [4,10,11,31,32]. Combined with increased clock frequency, a multi-core, multi-threaded processor chip demands higher on- and off-chip memory bandwidth and suffers longer average memory access delays despite an increasing on-chip cache size. Tremendous pressures are put on memory hierarchy systems to supply the needed instructions and data timely.

In this paper, we report performance measurement results on three available dual-core desktop processors: Intel Core 2 Duo E6400 with 2.13 GHz [11], Intel Pentium D 830 with 3.0 GHz [15] and AMD Athlon 64 × 2 4400+ with 2.2 GHz [2]. The Core 2 Duo E6400 was manufactured using 65 nm technology with 291 million transistors [11], while the Pentium D 830 and the Athlon 64 × 2 4400+ were manufactured under 90 nm technology with about 230 million transistors [1,25]. In contrast to existing performance studies [9,23,24] that usually provide overall execution time and throughput, this paper emphasizes on the memory hierarchy performance. We measure memory

* Corresponding author. Tel.: +1 225 578 5535; fax: +1 225 578 5200.
E-mail address: lpeng@lsu.edu (L. Peng).

1383-7621/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.sysarc.2008.02.004

access latency and bandwidth as well as cache-to-cache communication delays. We also examine the performance scalability between single and dual cores on the three tested processors.

There are several key design choices for the memory subsystem of the three processors. All three have private L1 caches with different sizes. At the next level, the Intel Core 2 Duo processor adapts a shared L2 cache design, called *Intel Advanced Smart Cache* for the dual cores [13]. The shared L2 approach provides a larger cache capacity by eliminating data replications. It also permits naturally sharing of cache space among multiple cores. When only one core is active, the entire shared L2 can be allocated to the single active core. However, the downside for the shared L2 cache is that it suffers longer hit latency and may encounter unfair usage of the shared L2 cache. Both the Intel Pentium D and the AMD Athlon 64 × 2 have a private L2 cache for each core, enabling fast L2 accesses, but restricting any capacity sharing among the two cores.

The shared L2 cache in the Core 2 Duo eliminates on-chip L2-level cache coherence. Furthermore, it resolves coherence of the two core's L1 caches internally within the chip for fast access to the L1 cache of the other core [13]. The Pentium D uses an off-chip front-side bus (FSB) for inter-core communications. Basically, the Pentium D is basically a technology remap of the Pentium 4 symmetric multiprocessor (SMP) that requires to access the FSB for maintaining cache coherence [15]. AMD Athlon 64 × 2 uses a HyperTransport interconnect technology for faster inter-chip communication [2]. Given an additional ownership state in the Athlon 64 × 2, cache coherence between the two cores can be accomplished without off-chip traffic. In addition, the Athlon 64 × 2 has an on-die memory controller to reduce memory access latency.

It would be easier to compare memory performance for the three systems with a uniform measurement tool such as the Intel VTune analyzer [16]. However, VTune cannot run on AMD Athlon 64 × 2. Moreover, the performance counters on AMD present less functions compared with Intel's sophisticated performance counters. It is difficult to match the performance counters across the three processors. Therefore, we decided to use a macro memory benchmark suite, *lmbench* [30] to examine memory bandwidth and latency of the three processors. *Lmbench* attempts to measure the most commonly found performance bottlenecks in a wide range of system applications. These bottlenecks can be identified, isolated, and reproduced in a set of small microbenchmarks, which measure system latency and bandwidth of data movement among the processor, memory, network, file system, and disk. In addition, we ran *STREAM* [21] and *STREAM2* [22] recreated by using *lmbench*'s timing harness. These kernel benchmarks measures memory bandwidth and latency using several common vector operations such as matrix addition and copy of matrix.

To understand the data transfer among individual core's caches, we used a small *lockless* program [26]. This *lockless* program records the duration of ping–pong procedures of a small token bouncing between two caches to get the average cache-to-cache latency. Finally, we run a set of single- and multi-threaded workloads on the three systems to examine the dual-core speedups over a single core. For single-thread programs, we experiment a set of mixed SPEC CPU2000 and SPEC CPU2006 benchmarks [28]. For multi-threaded workloads, we select *blastp* and *hmmpfam* from the BioPerf suites [6], SPECjbb2005 [29], as well as a subset of SPLASH2 [34]. Based on the experiment results, we can summarize a few interesting findings.

(1) In general, Core 2 Duo and Athlon 64 × 2 have better overall memory bandwidth and lower latency than Pentium D. The Core 2 Duo processor handles cache coherence between L1 caches on chip and employs aggressive memory dependence speculation. Its shared L2 generates less off-chip traffic than the other two. Athlon 64 × 2 handles private L2 coherence through on-chip system interfaces. It benefits from its on-chip memory controller for lower memory latency.
(2) The cache-to-cache latency plays an important role in multi-threaded workload with heavy data sharing. The cache-to-cache latencies of the selected Core 2 Duo, Pentium D and Athlon 64 × 2 processors are measured at 33 ns, 133 ns and 68 ns, respectively. Core 2 Duo benefits from its on-chip access to the other L1 cache. Pentium D requires off-chip FSB for inter-core communications. Athlon 64 × 2 employs on-die communication through crossbar switch. The execution time of the selected dual-threaded programs range from 6.3 to 490, 8.7 to 526, and 7.3 to 621 in second for Core 2 Duo, Pentium D and Athlon 64 × 2, respectively.
(3) For single-thread benchmarks, Core 2 Duo shows the best performance for most of selected SPEC CPU2000 and CPU2006 workloads running on one core because the core can utilize the entire shared L2 cache. Execution time of single thread of all workloads range from 56 to 1500, 75 to 1703, and 73 to 1993 in second for Core 2 Duo, Pentium D, and Athlon 64 × 2, respectively. All three processors demonstrate limited performance scalability for dual-core, where Athlon 64 × 2 has the best. Core 2 Duo's speed-ups are constraint due to its fast single-thread performance in using the entire L2 cache.

This paper is organized as follows. Section 2 briefly introduces the architectures of the three processors. Section 3 describes the methodology and the workloads of our experiments. Section 4 reports the detailed measurement results and the comparison between the three processors. Section 5 describes related work. Finally, we give a brief conclusion in Section 6.

2. Architectures of dual-core processors

The Intel Core 2 Duo (Fig. 1a) E6400 emphasizes mainly on cache efficiency and does not stress on the clock frequency for high power efficiency. Although clocking at a slower rate than that of the Pentium D, a shorter and wider issuing pipeline compensates the performance with higher IPCs. In addition, the Core 2 Duo processor has more ALU units [9]. Core 2 Duo employs a shared L2 cache to increase the effective on-chip cache capacity. Upon a miss from the core's L1 cache, the shared L2 and the L1 of the other core are looked up in parallel before sending the request to the memory [14]. The cache block located in the other L1 cache can be fetched without off-chip traffic. Both memory controller and FSB are still located off-chip. The off-chip memory controller can adapt the new DRAM technology with the cost of longer memory access latency. Core 2 Duo employs aggressive memory dependence speculation for memory disambiguation. A load instruction is allowed to be executed before an early store instruction with an unknown address. It also implements a macro-fusion technology to combine multiple micro-operations. Other important features involve support for new SIMD instructions called Supplemental Streaming SIMD Extension 3, coupled with better power saving technologies.

The Pentium D 830 (Fig. 1b) glues two Pentium 4 cores together and connects them with the memory controller through the north-bridge. The off-chip memory controller provides flexibility to support the newest DRAM with the cost of longer memory access latency. The MESI coherence protocol from Pentium SMP is adapted in Pentium D that requires a memory update in order to change a modified block to a shared block. The system interconnect for processors remains through the front-side bus (FSB). To accommodate the memory update, the FSB is located off-chip that increases the latency for maintaining cache coherence.

The Athlon 64 × 2 is designed specifically for multiple cores in a single chip (Fig. 1c). Similar to the Pentium D processor, it also employs private L2 caches. However, both L2 caches share a system request queue, which connects with an on-die memory controller and a HyperTransport. The HyperTransport removes system bottlenecks by reducing the number of buses required in a system. It provides significantly more bandwidth than current PCI technology [3]. The system request queue serves as an internal interconnection between the two cores without involvements of an external bus. The Athlon 64 × 2 processor employs MOESI protocol, which adds an "Ownership" state to enable modified blocks to be shared on both cores without the need to keep the memory copy updated.

An important aspect to alleviate cache miss penalty is data prefetching. According to the hardware specifications, the Intel Core 2 Duo includes a stride prefetcher on its L1 data cache [13] and a next line prefetcher on its L2 cache [9]. The L2 prefetcher can be triggered after detecting consecutive line requests twice. The Pentium D's hardware prefetcher allows stride-based prefetches beyond the adjacent lines. In addition, it attempts to trigger multiple prefetches for staying 256 bytes ahead of current data access locations [12]. The advanced prefetching in Pentium D enables more overlapping of cache misses. The Athlon 64 × 2 has a next line hardware prefetcher. However, accessing data in increments larger than 64 bytes may fail to trigger the hardware prefetcher [5].

Table 1 lists the specifications of the three processors experimented in this paper. There are no hyper-threading settings on any of these processors. The Intel Core 2 Duo E6400 has separate 32 kB L1 instruction and data caches per core. A 2MB L2 cache is shared by two cores. Both L1 and L2 caches are 8-way set associative and have 64-byte lines. The Pentium D processor has a Trace Cache which stores 12Kuops. It is also equipped with a write-through, 8-way 16 kB L1 data cache with a private 8-way 1MB L2 cache. The Athlon 64 × 2 processor's L1 data and instruction cache are 2-way 64 kB with a private 16-way 1MB L2 cache for each core. Athlon 64 × 2's L1 and L2 caches in each core is exclusive. All three machines have the same size L2 caches and Memory. The Core 2 Duo and the Pentium D are equipped with DDR2 DRAM using advanced memory controllers in their chipsets. The Athlon 64 × 2 has a DDR on-die memory controller. All three machines have 2GB memory. The FSB of the Core 2 Duo is clocked at 1066 MHz with bandwidth up to 8.5 GB/s. The FSB of the Pentium D operates at

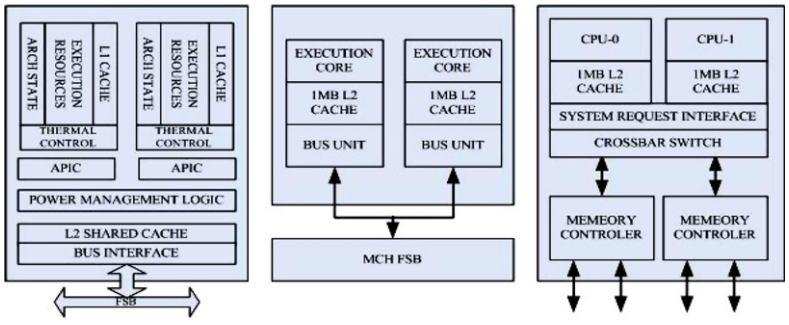

Fig. 1. (a) Intel Core 2 Duo; (b) Intel Pentium D; and (c) AMD Athlon 64 × 2.

Table 1
Specifications of the selected systems

CPU	Intel Core 2 Duo E6400 (2 × 2.13 GHz)	Intel Pentium D 830 (2 × 3.00 GHz)	AMD Athlon64 4400+ (2 × 2.20 GHz)
Technology	65 nm	90 nm	90 nm
Transistors	291 millions	230 millions	230 millions
Hyper-threading	No	No	No
L1 cache	Code and data: 32 kB × 2, 8 way, 64-byte cache line size, write-back	Trace cache: 12Kuops × 2, data: 16 kB × 2, 8-way, 64-byte line size, write-through	Code and data: 64 kB × 2, 2-way, 64-byte cache line size, write-back
L2 cache	2MB shared cache (2 MB × 1), 8-way, 64-byte line size, non-inclusive with L1 cache	2 MB private cache (1 MB × 2), 8-way, 64-byte line size, inclusive with L1 cache	2 MB private cache (1MB × 2), 16-way, 64-byte line size, exclusive with L1 cache
Memory	2 GB (1 GB × 2) DDR2 533 MHz	2 GB (512 MB × 4) DDR2 533 MHz	2 GB (1 GB × 2) DDR 400 MHz
FSB	1066 MHz data rate 64-bit	800 MHz data rate 64-bit	HyperTransport 16 bit up/down 2 GHz data rate (up + down)
FSB bandwidth	8.5 GB/s	6.4 GB/s	8 GB/s
HD interface	SATA 375 MB/s	SATA 150 MB/s	SATA 300 MB/s

800 MHz and provides up to 6.4 GB/s bandwidth. The Athlon 64 × 2 has a 2 GHz I/O HyperTransport with bandwidth up to 8 GB/s. Bandwidth of hard drive interface for the three machines are 375 MB/s, 150 MB/s and 300 MB/s, respectively. Because of our experiments are all in-memory benchmarks, difference in hard drives should have little impact.

3. Evaluation methodology

We installed SUSE linux 10.1 with kernel 2.6.16-smp on all three machines. We used O3 level GCC optimization to compile all the C/C++ benchmarks including *lmbench*, SPEC CPU2000, SPEC CPU2006, SPLASH2 and *blastp* and *hmmpfam* from BioPerf. SPECjbb2005 was compiled using SUN JDK 1.5.0.

We used *lmbench* suite running on the three machines to measure bandwidth and latency of memory hierarchy. *Lmbench* attempts to measure performance bottlenecks in a wide range of system applications. These bottlenecks have been identified, isolated, and reproduced in a set of small microbenchmarks, which measure system latency and bandwidth of data movement among the processor, memory, network, file system, and disk. In our experiments, we focus on the memory subsystem and measure bandwidth and latency with various memory operations listed in Table 2 [30]. Among them, we ran variable stride accesses to get average memory latency. In addition, we ran multi-copies *lmbench*, one on each core to test the memory hierarchy system. We also ran *STREAM* [21] and *STREAM2* [22] that are recreated by using *lmbench*'s timing harness. Each version has four common vector operations as listed in Table 3. During execution, a 24 MB array stream was allocated. Each vector operation processed array elements one by one. Average memory latencies for these vector operations were reported. Total data size processed in one second was reported as operation bandwidth.

Table 2
Memory operations from *Lmbench*

Operation	Description
Libc bcopy unaligned	Measuring how fast the processor can copy data blocks when data segments are not aligned with pages using a function bcopy()
Libc bcopy aligned	Measuring how fast the processor can copy data blocks when data segments are aligned with pages using a function bcopy()
Memory bzero	Measuring how fast the processor can reset memory blocks using a function bzero()
Unrolled bcopy unaligned	Measuring how fast the system can copy data blocks without using bcopy(), when data segments are not aligned with pages
Memory read	Measuring the time to read every 4 byte word from memory (stride 32 bytes)
Memory write	Measuring the time to write every 4 byte word to memory (stride 32 bytes)

We measured the cache-to-cache latency using a small *lockless* program [26]. It does not employ expensive read-modify-write atomic instructions. Instead, it maintains a *lockless* counter for each thread. The c-code of each thread is as follows:

```
*pPong = 0;
for (i = 0; i < NITER; ++i)
{
    while (*pPing < i);
    *pPong = i+1;
}
```

Each thread increases its own counter *pPong* and keeps reading the peer's counter by checking *pPing*. The counter *pPong* is in a different cache line from the counter *pPing*. A counter *pPong* can be increased by one only after verifying the update of the peer's counter. This pure software synchronization code generates a heavy read–write sharing between the two cores and produces a Ping–Pong procedure

Table 3
Kernel operations of the *STREAM* and *STREAM2* benchmarks

Set	Kernel	Operation
STREAM	copy	c[i] = a[i]
STREAM	scale	b[i] = scalar * c[i]
STREAM	add	c[i] = a[i] + b[i]
STREAM	triad	a[i] = b[i] + scalar * c[i]
STREAM2	fill	a[i] = q
STREAM2	copy	a[i] = b[i]
STREAM2	daxpy	a[i] = a[i] + q * b[i]
STREAM2	sum	sum = sum + a[i]

between the two caches to test processors in handling heavy cache-to-cache traffic.

For multiprogrammed workloads, the cross-product of mixed SPEC CPU2000/2006 benchmarks were run on the three machines to examine the dual-core speedups over a single core. All the SPEC CPU2000/2006 programs were run with their respective *ref* inputs. In our simulations, when two programs were run together, we guaranteed that each program was repeated at least four times. The shorter programs may run more than four iterations until the longer program completes its four full iterations. We discarded the results obtained in the first run and used the average execution time and other metrics from the remainder three runs. We calculated the dual-core speedup for multiprogrammed workloads similarly to that used in [25]. The single program's running time were collected individually. Afterwards, the average execution time of each workload when run simultaneously was recorded. The dual-core speedup of each workload is calculated by finding the ratio of average run time when run individually (single core) by the average runtime when run together (dual core). Finally, we add the speedups of the two programs run together to obtain the dual-core speedup. For example, if the speedups of two programs are 0.8 and 0.9 when run simultaneously, the respective dual-core speedup will be 1.7.

We used the same procedure for homogeneous multi-threaded workloads including *blastp* and *hmmpfam* from the BioPerf suites, a subset of SPLASH2, as well as SPECjbb2005. The BioPerf suite has emerging Bioinformatics programs. SPLASH2 is a widely used scientific workload suite. SPECjbb2005 is a java based business database program. Table 4 lists the input parameters of the multi-threaded workloads. We ran each of these workloads long enough to compensate overheads of sequential portions of the workloads.

Table 4
Input parameters of the selected multi-threaded workloads

Workload	Input parameters
blastp	Swissprot database, large input
hmmpfam	Large input
Barnes	104,8576 bodies
Fmm	524,288 particles
ocean-continuous	2050 × 2050 grid
fft	2^{24} total complex data points transformed
lu-continuous	4096 × 4096 node matrix
lu-non-continuous	4096 × 4096 node matrix
radix	134,217,728 keys to sort
SPECjbb2005	Default ramp up time 30 s, measurement time 240 s, from 1 to 8 warehouses

4. Measurement results

4.1. Memory bandwidth and latency

4.1.1. Lmbench

We first ran the bandwidth and latency test programs present in the *lmbench* suite. Fig. 2 shows memory bandwidth of several operations from *lmbench*. Fig. 2a, c and e shows the data collected while running one copy of *lmbench* on the three machines while Fig. 2b, d and f presents the 2-copy results. Several observations can be made:

(1) In general, Core 2 Duo and Athlon 64 × 2 have better bandwidth than that of Pentium D. Only exception is that Pentium D shows the best *memory read* bandwidth when the array size is less than 1MB. The shared cache of Core 2 Duo demands longer access latency though providing larger effective capacity. For Athlon 64 × 2, because the DRAM has lower bandwidth, its *memory read* bandwidth is lower than that of Pentium D when memory bus is not saturated. The *memory read* bandwidth for the three machines drops when the array size is larger than 32 kB, 16 kB and 64 kB, respectively. These reflect the sizes of their L1 cache. When the array size is larger than 2 MB, 1 MB and 1 MB for the respective three systems, we can see another dropping, reflecting their L2 cache sizes.
(2) The *memory bzero* operation shows different behaviors: when the array size is larger than their L1 data cache sizes, i.e., 32 kB for Core 2 Duo and 64 kB for Athlon 64 × 2, the memory bandwidth drops sharply. This is not true for Pentium D. The L1 cache of Core 2 Duo and Athlon 64 × 2 employ a write-back policy while the L1 cache of Pentium D uses a write-through policy. When the array size is smaller than their L1 data cache sizes, the write-back policy updates the L2 cache less frequently than the write-through policy, leading to higher bandwidth. However, when the array size is larger than their L1 data cache sizes, the write-back policy does not have any advantage as indicated by the sharp decline of the bandwidth.
(3) For Athlon 64 × 2, *libc bcopy unaligned* and *libc bcopy aligned* show a big difference while alignment does not have much difference for Core 2 Duo and Pentium D. 'Aligned' here means the memory segments are aligned to the page boundary. The operation *bcopy* could be optimized if the segments are

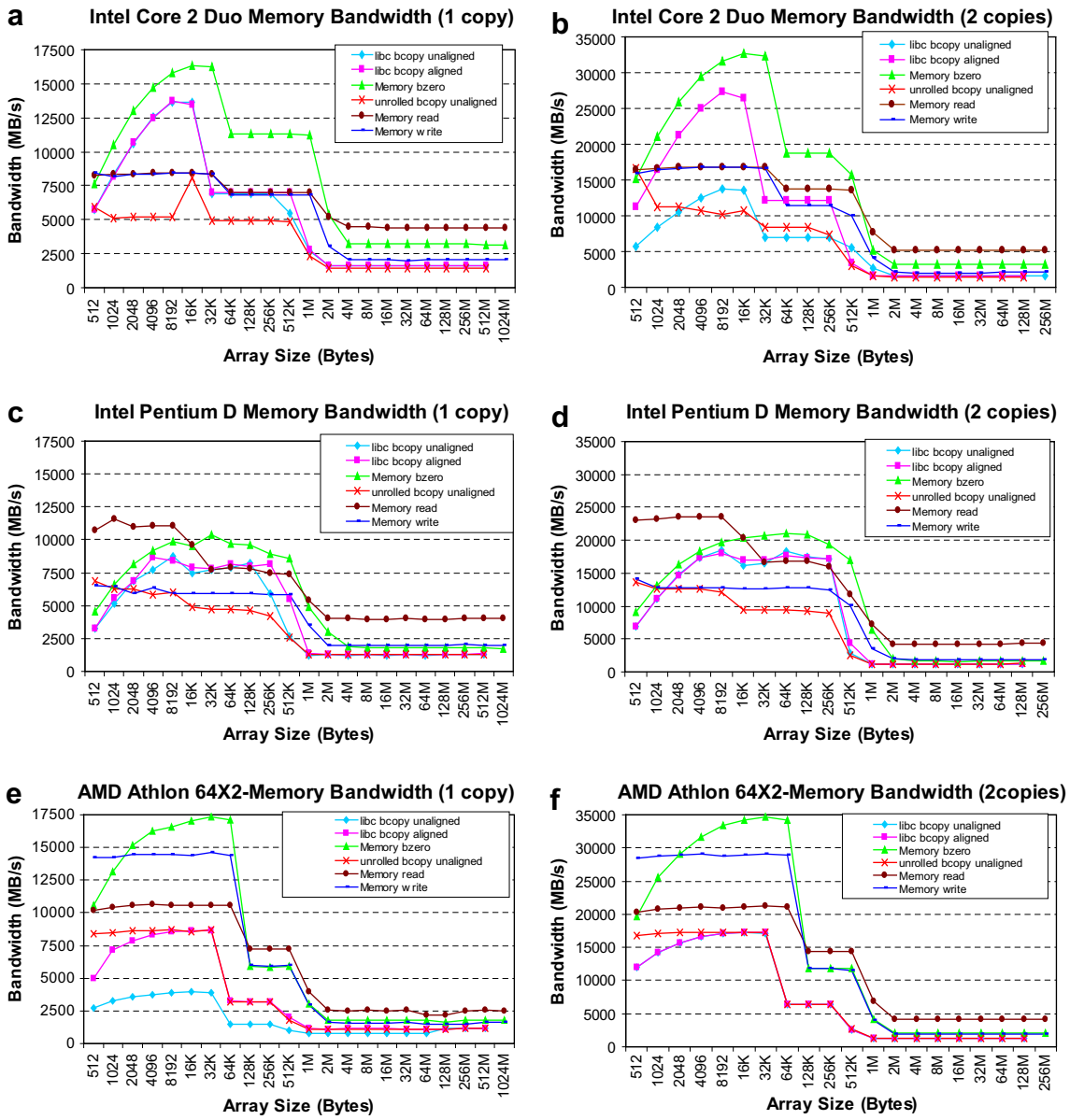

Fig. 2. Memory bandwidth collected from the *lmbench* suite with one or two copies.

page aligned. In Fig. 2a, c and e, Core 2 Duo and Pentium D show optimizations for *unaligned bcopy* access while Athlon 64 × 2 does not.

(4) Fig. 2b, d and f plots the bandwidth while running two copies of *lmbench*. The scale of the vertical axis of these three figures is doubled compared to their one-copy counterparts. We can observe that memory bandwidth of Pentium D and Athlon 64 × 2 are almost doubled for all operations. Core 2 Duo has increased bandwidth, but not doubled. This is because of the access contention when two *lmbench* copies compete with the shared cache. When the array size is larger than its L2 cache size 2MB, Athlon 64 × 2 provides almost doubled bandwidth for two-copy *lmbench memory read* operation compared with its one-copy counterpart. Athlon 64 × 2 benefits from its on-die memory controller and separate I/O HyperTransport. Intel Core 2 Duo and Pentium D processors suffer FSB bandwidth saturation when the array size exceeds the L2 capacity. Note that the line *libc bcopy unaligned* coincides with *libc bcopy aligned* in Fig. 2f.

Next, we examine memory load latency for multiple sizes of stride access and random access for all the three machines. Fig. 3a, c and e depicts the memory load latency lines of the three machines running with one copy of *lmbench*. Several observations can be made:

(1) For Core 2 Duo, latencies for all configurations jump after the array size is larger than 2 MB while for Pentium D and Athlon 64 × 2 latencies for all the

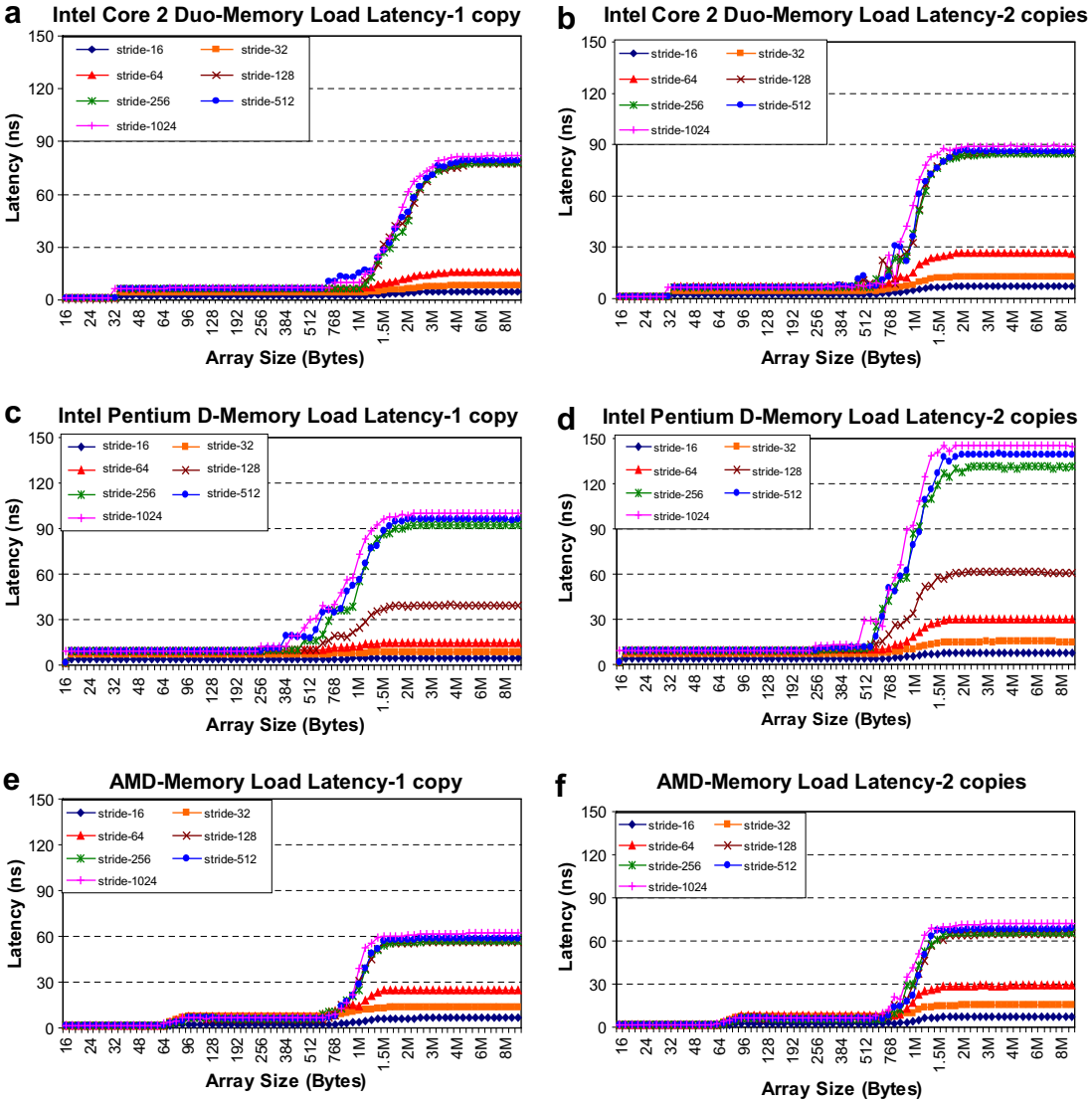

Fig. 3. Memory load latency collected from the *lmbench* suite with one or two copies.

configurations jump after the array size is larger than 1MB. This relates to the L2 cache sizes of the measured machines.

(2) As described in Section 2, when hardware prefetchers on all machines work, the memory bus bottleneck will not be reflected. When the stride size is equal to 128 bytes, Pentium D still benefits partially from its hardware prefetcher but the L2 prefetchers of Core 2 Duo and Athlon 64 × 2 is not triggered. This leads to better performance for Pentium D.

(3) When the stride size is larger than 128 bytes, all hardware prefetchers do not take effect. Multiple L2 cache misses put pressures onto the memory buses. Athlon 64 × 2's on-die memory controller and separate I/O HyperTransport show the advantage. Pentium D's memory latency has a large jump for these operations but Athlon 64 × 2's latency almost keeps unchanged.

(4) We increased pressure on memory hierarchy by running two copies of *lmbench* simultaneously shown in Fig. 3b, d and f. We found that Core 2 Duo and Athlon 64 × 2 have a slight increase in the latencies for stride sizes larger than 128 bytes while Pentium D's latencies increases a lot. Core 2 Duo benefits from its shared cache, which generates lower external traffic, while Athlon 64 × 2 take the advantage of on-chip memory controller and separate I/O Hyper-Transport. However, Pentium D's latencies jump due to suffering from memory bus saturation.

4.1.2. *STREAM*/*STREAM2*

We also ran eight kernel operations from *STREAM* and *STREAM2*. Fig. 4a shows memory bandwidth of *STREAM* and *STREAM2* operations when running with

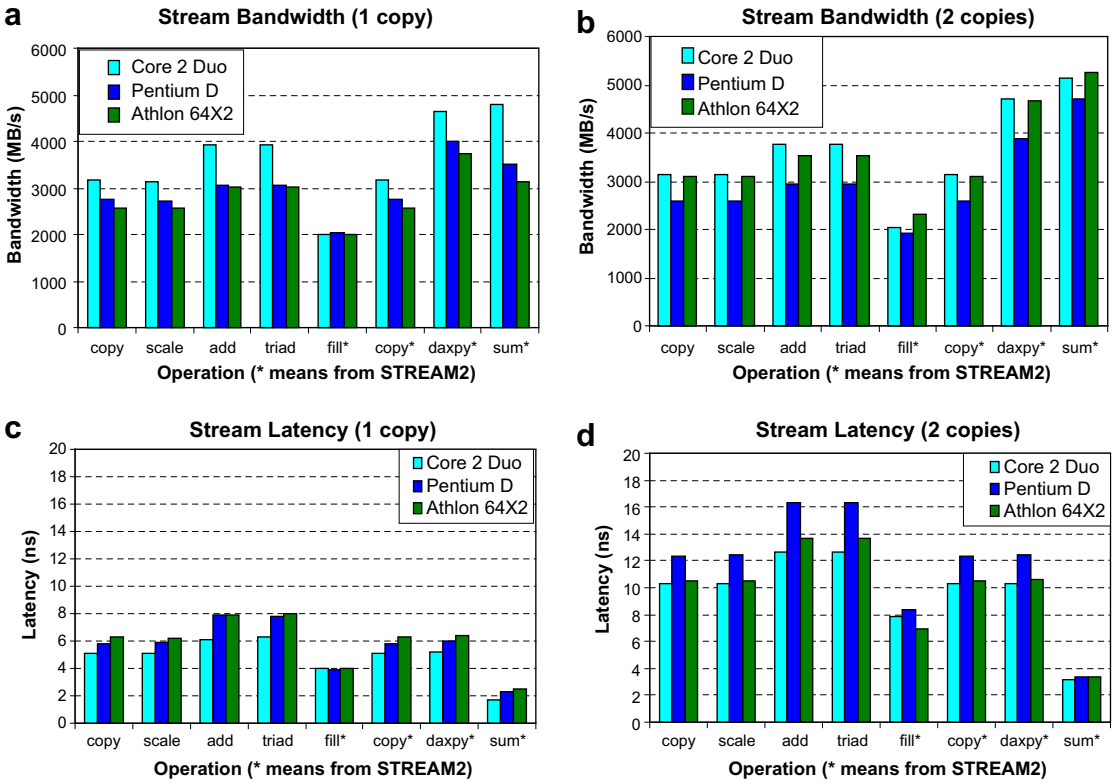

Fig. 4. Memory bandwidth and memory latency collected from the *STREAM* and *STREAM2* benchmarks. The benchmark runs with 1 or 2 copies.

a single copy of each operation. From this figure, we can see that Intel Core 2 Duo shows the best bandwidth for all operations because of L1 data prefetchers and the faster Front Side Bus. Pentium D has better bandwidth than that of Athlon 64 × 2. This is again because the Pentium D system is equipped with better DRAM than the Athlon 64 × 2 system.

Fig. 4b depicts memory bandwidth when running with two copies of each operation in *STREAM*/*STREAM2*, one on each core. From this figure, we can see that Core 2 Duo and Athlon 64 × 2 have better bandwidth than that of Pentium D. This is due to the fact that Pentium D's FSB is saturated when running two copies of each operation. Athlon 64 × 2 benefits from its on-die memory controller and separate HyperTransport for I/O although its main memory DDR bandwidth is worse than that of Pentium D. Core 2 duo benefits from the presence of its L1 data prefetchers and the faster FSB. Fig. 4c and d shows the memory latencies for the three machines. Similar to the bandwidth figures, memory latency of Core 2 Duo and Pentium D are shorter than that of Athlon 64 × 2 when a single copy of the *STREAM*/*STREAM2* benchmark is running. Apparently, the shorter latency from on-die memory controller does not pay off in comparison with an off-die controller with better DRAM technology. However, while running the 2-copy version, memory latency of Pentium D is higher than the other two.

4.2. Multiprogrammed workload measurements

We measured execution time of a subset of SPEC CPU2000 and CPU 2006 benchmarks. In Fig. 5a and c, the Core 2 Duo processor runs fastest for almost all workloads executed along, especially for memory intensive workloads *art* and *mcf*. Core 2 Duo has a wider pipeline, more functional units, and a shared L2 cache that provides bigger cache for single thread running along. Athlon 64 × 2 shows the best performance for *ammp*. *ammp* has large working set, resulting in high L2 cache misses. Athlon 64 × 2 benefits from its on-chip memory controller.

Fig. 5b and d depicts average execution time of each workload when mixed with another program in the same SPEC suite. There is an execution time increasing for each workload. For memory bounded programs *art*, *mcf* and *ammp*, execution time increasing is large while CPU bounded workloads such as *crafty*, *mesa*, *perl* and *sjeng* show little impact.

The multi-programmed speedup of the cross-product of mixed SPEC CPU2000 and CPU2006 programs for the three machines are given in Fig. 6, where C2D, PNT and ATH denote the measured Core 2 Duo, Pentium D, and Athlon 64 × 2, respectively. We can see that Athlon 64 × 2 achieves the best speedup for all workloads. *Crafty*, *eon*, *mesa* in CPU 2000 and *perl* in CPU2006 have the best speedup when run simultaneously with other programs because they are CPU-bound programs. On the other hand, in most cases, *art* shows

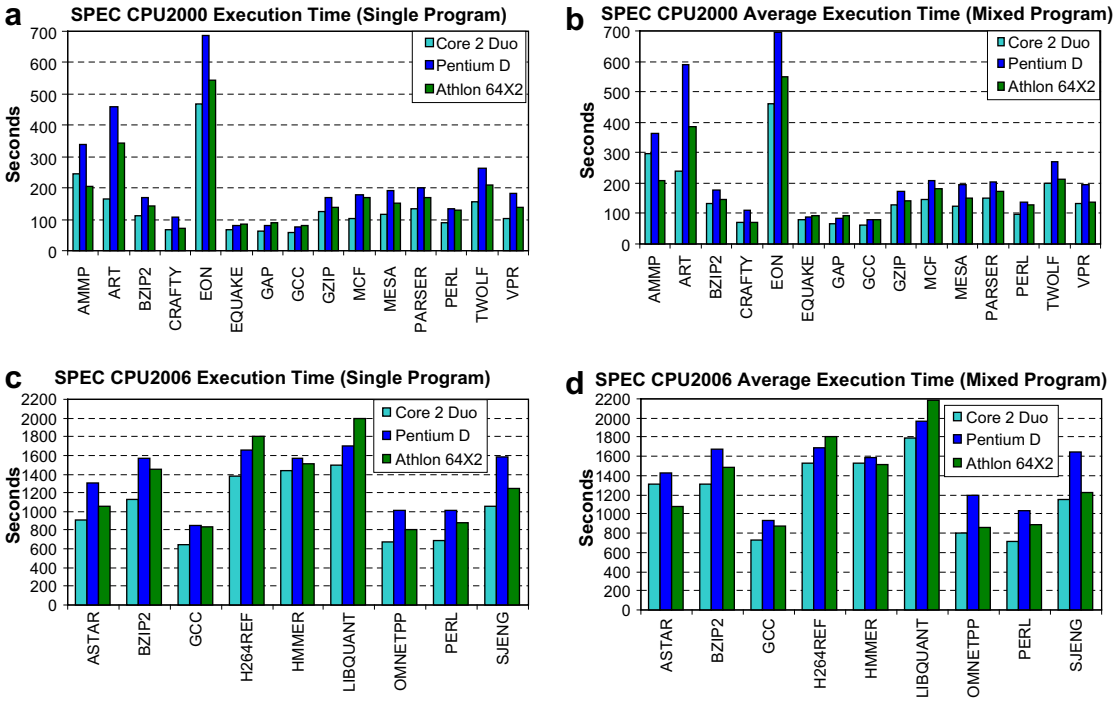

Fig. 5. SPEC CPU2000 and CPU2006 benchmarks execution time.

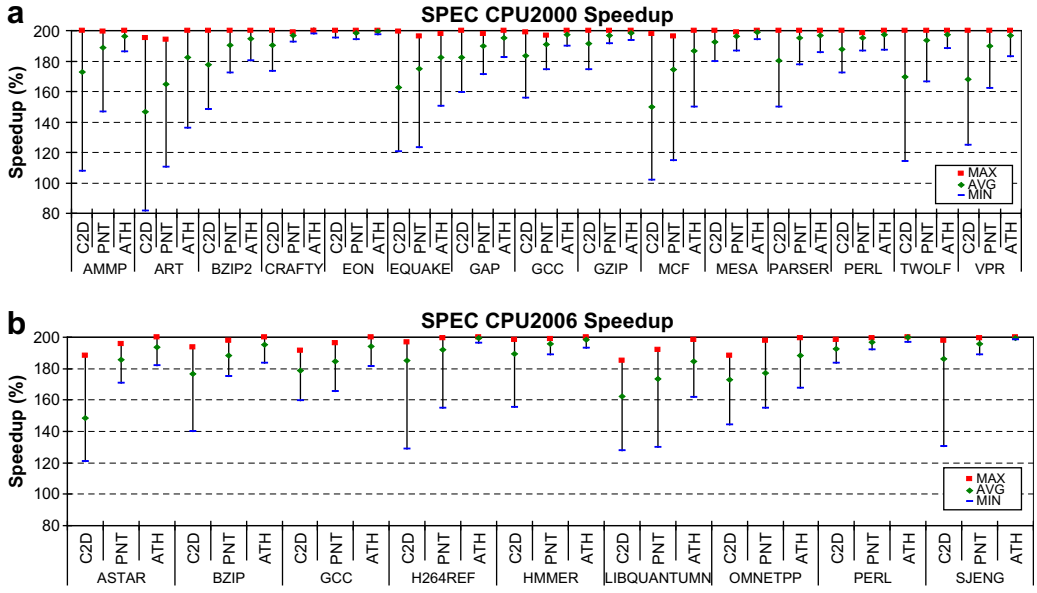

Fig. 6. Multi-programmed speedup of mixed SPEC CPU 2000/2006 benchmarks.

the worst speedup because it is a memory bounded program. Its intensive L2 cache misses occupy the shared memory bus and block another program's execution. In the extreme case, when an instance of *art* was run against another *art*, the speedups were 0.82, 1.11 and 1.36 for Core 2 Duo, Pentium D and Athlon 64 × 2. Other memory bounded programs, *ammp* and *mcf*, present similar behaviors.

Comparing the three machines, the multi-programmed Athlon 64 × 2 outperforms those of Core 2 Duo and Pentium D for almost all workload mixes. It is interesting to note that even though Core 2 Duo has better running time than the other two machines, the overall speedup is lesser. The reason again is due to its L2 shared cache that boosts single-core performance.

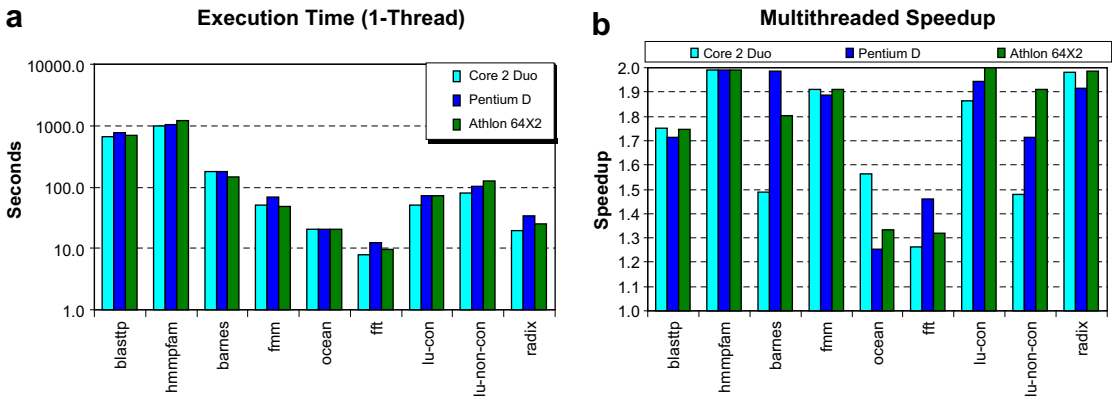

Fig. 7. (a) Execution time for 1-thread version of selected multi-threaded programs and (b) speedup for 2-thread version of selected multi-threaded programs.

4.3. Multi-threaded program behaviors

We used the *lockless* program to measure the dual-core cache-to-cache latency. The average cache-to-cache latencies are significantly different among Core 2 Duo, Pentium D, and Athlon 64 × 2, with 33 ns, 133 ns and 68 ns, respectively. This is again due to the fact that Core 2 Duo resolves L1 cache coherence within the chip, while Pentium D requires external FSB for cache-to-cache transfer. Athlon 64 × 2 uses on-chip system request interface and the MOESI protocol for cache-to-cache communication.

Next, we experiment with *blastp* and *hmmpfam* from the BioPerf suite and a set of the SPLASH2 workloads. Fig. 7a and b illustrates execution time of single thread version of the programs and the speedup when running with 2-thread version. In general, Core 2 Duo and Athlon 64 × 2 do not show performance advantages over Pentium D on bioinformatics and scientific workloads because of limited data communication between two cores. Similar results were also reported on Multimedia programs [9]. Among all applications, Core 2 Duo shows the best speedup over other processors for *ocean* due to its high cache-to-cache transfers [34]. We verified this behavior using Intel's VTune Performance Analyzer 8.0 [16]. Fig. 8 illustrates the average number of CMP_SNOOP.ANY events, which represents the remote cache access, per 1 K retired instructions on Core 2 Duo. Among all workloads. Ocean has the highest remote cache accesses per 1 K retired instructions. Pentium D shows the best speed up for *barnes* because of the low cache miss rate [34]. Recall that Pentium D processor also has the best *memory read* bandwidth when the array size is small. Bioinformatics workloads have high speedups for all three machines due to small working sets [6].

High data sharing workloads, such as SPECjbb2005 also benefit from fast cache-to-cache latency. Fig. 9 shows the transaction per second (TPS) throughput with one (denoted by 1w-1c) and two (denoted by 2w-1c) warehouses of SPECjbb2005 on the three systems. For a fair speedup comparison, we also run two copies of a single warehouse on two cores (denoted by 1w-2c). These two

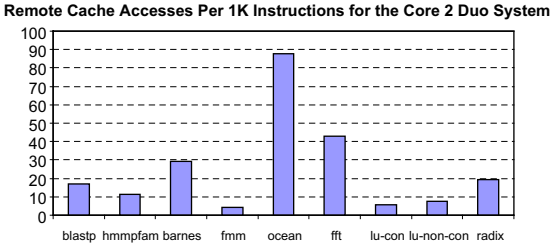

Fig. 8. Remote cache accesses per 1 K instructions for the Core 2 Duo system.

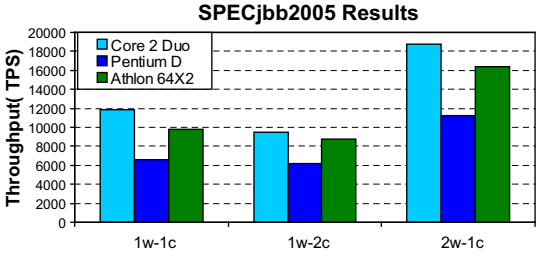

Fig. 9. SPECjbb2005 throughput running with one and two warehouses.

copies on the two cores compete with the shared L2 cache so that Core-2-Duo loses its unique advantage of taking the entire L2 capacity with a single warehouse. As can be observed, Core-2-Duo shows the worst performance degradation (about 20%) from 1w-1c to 1w-2c. Using 1w-2c as the basis, the 2w-1c speedups for the respective three systems are 1.97, 1.80, and 1.87 where Core-2-Duo is the winner.

5. Related work

The emergence of Intel and AMD dual-core processors intrigues hardware analysts. There are many online reports which compare performance of processors from both companies [9,23,24]. Most of them simply present the performance metrics such as running time and throughput

without detailed analysis. In this paper, we focus on the memory hierarchy performance analysis and understanding the underlying reasons.

Chip multiprocessor (CMP) or multi-core technology was first reported in [10]. Companies such as IBM and SUN applied it on their server processors [18,31,32]. In 2005, Intel announced to shelve its plan in pursuing higher frequency and instead switch to building multi-core processors [15]. Similarly, AMD also made the same decision about the same time [4].

Tuck and Tullsen [33] studied thread interactions on an Intel Pentium 4 hyper-threading processor. They used multi-programmed and multi-threaded workloads to measure speedup and synchronization and communication throughput. Bulpin and Pratt [7] measured an SMT processor with consideration about fairness between threads. They also showed the performance gap between SMP and Hyper-threaded SMT for multi-programmed workloads.

In [20], we did a case study on memory performance and scalability of the selected processors. In this journal version paper, we provide more detailed results and analysis.

There are several recent proposals to study the issues of CMP shared cache fairness and partitioning. In [19], the authors proposed and evaluated five different metrics such as shared cache miss rates, which can be correlated to execution time, used for CMP fairness and proposed static and dynamic caches partitioning algorithms that optimize fairness. This dynamic algorithm can help the operating system thread scheduling and to avoid thread thrashing. Other works proposed OS driven policy [27], cache management framework (CQoS) [17] and prediction models [8] for inter-thread cache contention in a shared CMP cache.

6. Conclusion

In this paper, we analyzed the memory hierarchy of selected Intel and AMD dual-core processors. We first measured the memory bandwidth and latency of Core 2 Duo, Pentium D and Athlon 64 × 2 using *lmbench*. In general, Core 2 Duo and Athlon 64 × 2 have better memory bandwidth than that of Pentium D.

We measured individual execution time of SPEC CPU2000 and CPU2006. We also measured the average execution time of each application when mixed with other programs on the dual cores. In general, Core 2 Duo runs fastest for all single and mixed applications except for *ammp*. We also observed that memory intensive workloads such as *art*, *mcf* and *ammp* have worse speedups. We measured the cache-to-cache latencies. Core 2 Duo has the shortest, while Pentium D has the longest. This generic memory performance behavior is consistent with the performance measurement results of multi-threaded workloads with heavy data sharing between the two cores.

The Core 2 Duo, with its shared L2, demonstrates distinct advantages when running a single program on one core. However, to manage the shared cache resource efficiently is a challenge especially when two cores have very different demands for caches. In summary, for the best performance and scalability, the following are important factors: (1) fast cache-to-cache communication, (2) large L2 or shared capacity, (3) fast L2 access latency, and (4) fair resource (cache) sharing. Three processors that we studied have shown benefits of some of them, but not all of them.

Acknowledgements

The comments from the second reviewer help a great deal to improve the content of this paper, especially leading to a bug found on the original STREAM benchmark. This work is supported in part by the Louisiana Board of Regents Grants NSF (2006)-Pfund-80 and LEQSF (2006-09)-RD-A-10, the Louisiana State University and an ORAU Ralph E. Powe Junior Faculty Enhancement Award.

References

[1] AMD, AMD Athlon 64 × 2 Dual-Core Processor Model Number and Feature Comparisons, <http://www.amd.com/us-en/Processors/ProductInformation/0,,30_118_948513041%5E13076,00.html>.
[2] AMD, AMD Athlon 64 × 2 Dual-Core Product Data Sheet, <http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/33425.pdf>.
[3] AMD, AMD HyperTransport Technology, <http://www.amd.com/us-en/Processors/DevelopWithAMD/0,,30_2252_2353,00.html>.
[4] AMD, Multi-core Processors: The Next Evolution in Computing, <http://multicore.amd.com/WhitePapers/Multi-Core_Processors_WhitePaper.pdf>, 2005.
[5] AMD, Software Optimization Guide for AMD64 Processors, p. 105 (Chapter 5), <www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/25112.PDF>.
[6] D. Bader, Y. Li, T. Li, V. Sachdeva, BioPerf: A benchmark suite to evaluate high-performance computer architecture on bioinformatics applications, in: Proceedings of the 2005 IEEE International Symposium on Workload Characterization, October 2005.
[7] J.R. Bulpin, I.A. Pratt, Multiprogramming performance of the Pentium 4 with hyper-threading, in: Proceedings of Third Annual Workshop on Duplicating, Deconstructing, and Debunking (WDDD), June 2004.
[8] D. Chandra, F. Guo, S. Kim, Y. Solihin, Predicting the inter-thread cache contention on a chip multiprocessor architecture, in: Proceedings of the 11th International Symposium on High Performance Computer Architecture, February 2005, pp. 340–351.
[9] F. Delattre, M. Prieur, Intel Core 2 Duo – Test, <http://www.behardware.com/articles/623-16/intel-core-2-duo-test.html>.
[10] L. Hammond, B.A. Nayfeh, K. Olukotun, A single-chip multiprocessor, IEEE Computer (September) (1997).
[11] Intel, Announcing Intel Core 2 Processor Family Brand, <http://www.intel.com/products/processor/core2/index.htm>.
[12] Intel, IA-32 Intel Architecture Optimization Reference Manual, pp. 6–4 (Chapter 6), <http://www.intel.com/design/pentium4/manuals/248966.htm>.
[13] Intel, Inside Intel Core Microarchitecture and Smart Memory Access, <http://download.intel.com/technology/architecture/sma.pdf>.
[14] Intel, CMP Implementation in Systems Based on the Intel Core Duo Processor, <http://www.intel.com/technology/itj/2006/voume10issue02/art02_CMP_Implementation/p03_implementation.htm>.
[15] Intel, Intel Pentium D Processor Product Information, <http://www.intel.com/products/processor/pentium_d/>.
[16] Intel, Intel VTune Performance Analyzers, <http://www.intel.com/cd/software/products/asmo-na/eng/vtune/239144.htm>.

[17] R. Iyer, CQOS: a framework for enabling Qos in shared caches of CMP platforms, in: Proceedings of the 18th International Conference on Supercomputing, 2004, pp. 257–266.
[18] R. Kalla, B. Sinharoy, J.M. Tendler, IBM Power5 chip: a dual core multithreaded processor, IEEE Micro. 24 (2) (2004) 40–47.
[19] S. Kim, D. Chandra, Y. Solihin, Fair cache sharing and partitioning in a chip multiprocessor architecture, in: Proceedings of the 13th International Conference on Parallel Architectures and Compilation Techniques, September 2004, pp. 111–122.
[20] L. Peng, J.-K. Peir, T.K. Prakash, Y.-K. Chen, D. Koppelman, Memory performance and scalability of Intel's and AMD's dual-core processors: a case study, in: Proceeding the 26th IEEE International Performance Computing and Communications Conference (IPCCC), New Orleans, LA, April 2007.
[21] J.D. McCalpin, Sustainable memory bandwidth in current high performance computers, Technical Report, Silicon Graphics, October 1995.
[22] J.D. McCalpin, The stream2 homepage, <http://www.cs.virginia.edu/stream/stream2>.
[23] A. Mitrofanov, Dual-core processors, <http://www.digital-daily.com/cpu/dualcore-cpu/index.htm>.
[24] www.motherboards.org, AMD Vesus Intel Battle of the Dual-Core CPUs, <http://www.motherboards.Org/reviews/hardware/1513_2.html>.
[25] V. Romanchenko, Intel processors today and tomorrow, <http://www.digital-daily.com/cpu/intel-roadmap/>.
[26] S. Michael, How can we measure cache-to-cache transfer speed? <http://www.aceshardware.com/forums/read_post.jsp?id = 20676&forumid = 2>.
[27] N. Rafique, W.-T. Lim, M. Thottethodi, Architectural support for operating system-driven CMP cache management, in: Proceedings of the 15th International Conference on Parallel Architectures and Compilation Techniques, September 2006, pp. 2–12.
[28] SPEC, SPEC CPU2000 and CPU2006, <http://www.spec.org/>.
[29] SPEC, SPECjbb2005, <http://www.spec.org/jbb2005/>.
[30] C. Staelin, lmbench – an extensible micro-benchmark suite, HPL-2004-213, December 2004, <http://www.hpl.hp.com/techreports/2004/HPL-2004-213.pdf>.
[31] Sun Microsystems, Sun's 64-bit Gemini Chip, Sunflash 66(4) (2003).
[32] J.M. Tendler, S. Dodson, S. Fields, H. Le, B. Sinharoy, IBM eserver Power4 System Microarchitecture, IBM White Paper, 2001.
[33] N. Tuck, D.M. Tullsen, Initial observations of the simultaneous multithreading Pentium 4 processor, in: Proceedings of the 12th International Conference on Parallel Architectures and Compilation Techniques (PACT), September 2003, pp. 26–34.
[34] S.C. Woo, M. Ohara, E. Torrie, J.P. Singh, A. Gupta, The SPLASH-2 programs: characterization and methodological considerations, in: Proceedings of the 22nd Annual International Symposium on Computer Architecture (ISCA), June 1995, pp. 24–36.

Lu Peng received his Bachelor and Master degrees in Computer Science and Engineering from Shanghai Jiaotong University, China. He obtained his Ph.D. degree in Computer Engineering from the University of Florida in Gainesville in April 2005. He joined the Electrical and Computer Engineering Department at Louisiana State University as an Assistant Professor in August, 2005. His research focus on memory hierarchy system, multi-core interconnection, power efficiency and other issues in CPU design. He also has interests in Network Processor. He received an ORAU Ralph E. Powe Junior Faculty Enhancement Awards in 2007 and a Best Paper Award from IEEE International Conference on Computer Design in 2001. He is a member of the ACM and the IEEE Computer Society.

Jih-Kwon Peir after receiving his Ph.D. degree from University of Illinois, He joined IBM T. J. Watson Research Center as a Research Staff Member. At IBM, he participated in the design and development of high-performance mainframe computers. During 1992–1993, he joined the Computer and Communication Lab in Taiwan as a Deputy Director of the Computer System Division, where he was in charge of the development of a Pentium-based, symmetric multiprocessor system. He is currently an Associate Professor at Computer and Information Science and Engineering Department, University of Florida. He also spent several summers visiting Intel's Microprocessor Research Lab and IBM Almaden Research Center. His major research and teaching focus is on the high-performance computer system architectures, microarchitectures and their memory hierarchy designs. He has published over 60 papers in international journals and conferences. He received an NSF Career Award and an IBM Research Partnership Award. He served on the Editorial Board of IEEE Transactions on Parallel and Distributed Systems. He also serves as a subject area editor for the Journal of Parallel and Distributed Computing.

Tribuvan Kumar Prakash was born in Bangalore, in the state of Karnataka, India. He graduated from High School in April 2000 with first class. In the Fall of 2000, he enrolled in the Department of Electronics and Telecommunications at the Vemanna Institute of Technology (affiliated to Visweswariah Technical University, Karnataka) and graduated with a first class distinction in spring 2004 with a Bachelor of Engineering degree. He then joined the Department of Electrical and Computer Engineering at Louisiana State University, Baton Rouge, to complete his master's, in the Fall of 2004. He worked with Unisys as an intern for the summer and fall semester of 2006. He graduated in August 2007 with the degree of Master of Science in Electrical Engineering and is currently working as IT Engineer at Realization Technology Inc, San Jose.

Carl Staelin is the project manager for the Systems and Software Project at HP Labs Israel. He received his B.Sc. in Electrical Engineering and B.A. Mathematics at the University of Rochester in 1985, and his M.A. and PhD. in Computer Science at Princeton University in the field of file system design in 1991. Upon graduation, he joined HP Labs with a joint appointment to the HP Berkeley Science Center, where he worked on hierarchical storage management, the 4.4BSD port of the Log Structured Filesystem (LFS), and distributed database systems, and to the Storage Systems Project, where he worked on the first HP AutoRaid product, focusing on the automated data migration and management algorithms. Starting in 1994, he prototyped numerous web-based services to demonstrate the power and flexibility of the web technologies. In 1997 he started the WebPaper project to bring paper into the digital world, which developed a number of technologies to automatically recognize document types and relevant metadata from scanned documents. He co-developed, and is a maintainer of, the lmbench micro-benchmark suite. More recently, he leads the development of the HP Indigo Photo Enhancement Server.

Yen-Kuang Chen received his Ph.D. degree from Princeton University, and is a Principal Engineer with the Corporate Technology Group, Intel Corporation. His research interests include developing innovative multimedia applications, studying the performance bottleneck in current computers, and designing next generation microprocessor/platform. Currently, he is analyzing emerging multimedia applications and providing inputs to the definition of the next-generational CPUs and GPUs with many cores. He has 10+ US patents, 25+ pending patent applications, and 75+ technical publications. He is one of the key contributors to Supplemental Streaming SIMD Extension 3 in Intel Core 2 Duo processor family. He is an associate editor of the Journal of VLSI Signal Processing Systems (including special issues on "System-on-a-Chip for Multimedia Systems", "Design and Programming of Signal Processors for Multimedia Communication", and "Multi-core Enabled Multimedia Applications & Architectures"), of IEEE Transactions on Circuit and System I, and IEEE Transactions on Circuit and System for Video Technology. He has served as a program committee member of 20+ international conferences and workshops on multimedia, video communication, image processing, VLSI circuits and systems, parallel processing, and software optimization. He has been Co-chairman of the MPEG-4 Intellectual Property Management and Protection Ad Hoc Group, and Chairman of the MPEG Study on Standard Digital Watermarking Technology Ad Hoc Group. He is an invited participant to 2002 Frontiers of Engineering Symposium (National Academy of Engineering) and to 2003 German–American Frontiers of Engineering Symposium (Alexander von Humboldt Foundation). He is an IEEE Senior Member and an ACM Senior Member.

David Koppelman received his Ph.D. in Computer Engineering from Rensselaer Polytechnic Institute. He is currently an Associate Professor in the Department of Electrical and Computer Engineering at Louisiana State University. His interests include parallel computation and computer architecture.