Experimental Evaluation of Emerging Multi-core Architectures

Abdullah Kayi [1], Yiyi Yao [1], Tarek El-Ghazawi [1], Greg Newby [2]

[1]The George Washington University
Dept. of Electrical and Computer Engineering
Washington, DC 20052 USA
{apokayi, yyy, tarek}@gwu.edu

[2]Arctic Region Supercomputing Center
Fairbanks, AK 99775 USA

Abstract

The trend of increasing speed and complexity in the single-core processor as stated in the Moore's law is facing practical challenges. As a result, the multi-core processor architecture has emerged as the dominant architecture for both desktop and high-performance systems. Multi-core systems introduce many challenges that need to be addressed to achieve the best performance. Therefore, a new set of benchmarking techniques to study the impacts of the multi-core technologies is necessary. In this paper, multi-core specific performance metrics for cache coherency and memory bandwidth/latency/contention are investigated. This study also proposes a new benchmarking suite which includes cases extended from the High Performance Computing Challenge (HPCC) benchmark suite. Performance results are measured on a Sun Fire T1000 server with six cores and an AMD Opteron dual core system. Experimental analysis and observations in this paper provide for a better understanding of the emerging multi-core architectures.

1. Introduction

The emerging multi-core architectures provide a solution to increase the performance capability on a single chip without requiring a complex system and increasing the power requirements [1, 2, 3, 4]. However, these architectures have introduced many challenges in maximizing application performance. Thus, benchmarking of the multi-core architectures becomes very important to unveil the potentials of these systems. Most existing benchmarks are not targeting these multi-core architectures and thus are not able to exploit the multi-core specific low level features such as shared cache coherence overhead, memory resource contention and etc. To address this problem, this paper first proposes a set of multi-core specific performance metrics to be investigated in this research study. In addition, all the experiments focus on the sources of possible bottlenecks in these multi-core architectures to be able to evaluate the potentials of these systems. Some synthetic benchmarking cases, an extension from HPCC benchmarking suites [5] (STREAM [6] and RandomAccess [7]) and an FFTW [8] multi-threaded application featuring the proposed performance metrics are provided. Such benchmarking cases and application are applied to the UltraSPARC T1 processor [9] and the AMD Opteron processors [10]. The experimental methodology is explained in Section 2 whereas Section 3 shows the results and observations obtained during this study. And finally, Section 4 includes the conclusion and the future remarks.

2. Experimental Methodology

At the beginning of this research study, a set of multi-core specific performance metrics are identified to guide the benchmarking of multi-core architectures as illustrated in Table 1. Such metrics focus on the features that are different from previous single core architectures and exploit the potential sources of inefficiencies in the

1-4244-0910-1/07/$20.00 ©2007 IEEE.

Sources of Performance Inefficiency in Multi-core Architectures	**Aspects**	**Metrics**
CACHE	Cache coherency among the cores	*Cache Coherency Overhead*
MEMORY	Bandwidth sustained from the cores to the main memory	*Bandwidth*
	Latency observed from the cores to the main memory	*Latency*
	Contention among the cores while accessing the main memory	*Percentage of Memory Contention*

Table 1. Multi-core specific benchmarking metrics

multi-core architectures. As shown in Table 1, the performance metrics are developed in layers targeting the architectural aspects of multi-core systems. The Cache and the Memory are listed as the sources of performance inefficiency that we expect to affect a multi-core system performance. Within each of these sources, there are some aspects that are multi-core specific and these are listed in the second column of Table 1. Cache coherency among the cores is the aspect we identified to be the most important for the overall systems performance from cache architectural point of view mainly for the shared cache multi-core architectures. From the memory side, bandwidth sustained from the cores to the main memory, latency observed from the cores to the main memory and resource contention among the cores while accessing the main memory are selected as important aspects to be examined in this study. According to these aspects performance metrics are determined as the experimental goals of this study. For each of these metrics, we developed a synthetic benchmarking case to measure the potential performance or overhead to get a better understanding of the corresponding aspect in the experimented multi-core architectures.

In order to convey the experiments, UltraSPARCT1 (6 cores) and AMD Opteron single/dual core processors are utilized with Solaris and Linux operating systems respectively. To fully exploit the multi-core architectures, all the benchmarking cases were implemented using the POSIX [11] thread library if necessary. In addition, for both systems we used the gcc compiler. In order to interpret the results better, following sub-sections will describe the benchmarking schemes and methodology for each and every aspect stated earlier.

2.1 Cache Aspects

Different multi-core architectures use different ways of caching and cache sharing among the cores [12, 13, 14, 15]. For instance, each UltraSPARC T1 processor has a twelve-way associative (four banks) unified Level 2 (L2) on-chip cache, and each cache hardware strand shares the entire L2 cache [9, 16]. Besides this, each UltraSPARC T1 processor core has its own L1 instruction cache, L1 data cache, instruction TLB and data TLB. Thus, cache coherency effect is very important for such systems to be examined in detail. On the other hand, the AMD Opteron processor cores share neither the L1 cache nor the L2 cache [17]. However, cache coherency is still an important effect to be considered for these multi-core systems. In our benchmarking suite, we created cache trashing cases and accordingly tried to measure the cache coherency overhead in these scenarios. Further information will be given in Section 3 for this experiment with the results.

2.2 Memory aspects

It is quite interesting and important to examine how more than one core on a single chip will affect the overall system performance while accessing the main memory [18]. It is crucial to discover the challenges and limits of these multi-core systems for the earlier stated memory aspects. In order to examine the memory subsystem of these multi-core systems, we adapted two of the HPCC benchmarking tests, STREAM and RandomAccess [6, 7]. However, these tests were modified by POSIX threads library in order to be able to serve our needs. STREAM results were used mainly to obtain the memory bandwidth sustained by using various numbers of cores and RandomAccess results were utilized to investigate the memory latency and memory contention issues for the experimental test beds.

3. Experimental Results and Analysis

In this section, results from our experiments and corresponding analyses will be provided. Starting from the benchmarking efforts on cache, the experimental analysis will be presented in the same order they were described in the previous section with some additional benchmarking cases at the end. Figure 1 shows the cache coherency overhead observed in the UltraSPARC T1 processor. And Figure 2 represents the same analysis on the AMD Opteron dual core processor. Both of these figures include results which were normalized according to a single element inside given workloads. Results were obtained using different problem sizes starting with 10^5 elements to 10^8 elements in order to guarantee the cache trashing on both of the systems. As it is mentioned in section 2.1 both of these systems have cores which have dedicated L1 cache, but UltraSPARC T1 processor is a unique system in the sense that it provides hardware strands inside each core leading to four threads per given core, which is a part of chip multi threading (CMT) technology [19]. Thus, L1 cache is shared between the strands inside the core between four threads and on the upper level L2 cache is shared among all the available cores. Accordingly, caching becomes a very important issue for such processor in the sense that it may yield to bigger advantages and/or disadvantages. Considering the cache coherency effect since we are testing the corner cases, which includes aggressive cache trashing this type of shared L2 cache resulted in bigger overheads compared to the AMD Opteron processor (dual core) in which each core has its own L1 and L2 caches. We implemented the scheme used in this test as a part of our benchmarking suite.

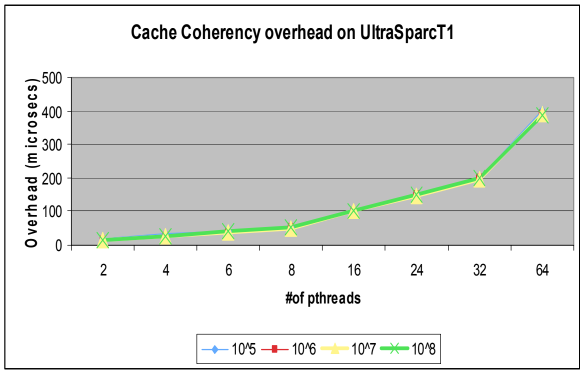

Figure 1: Cache Coherency Overhead on UltraSPARC T1 processor

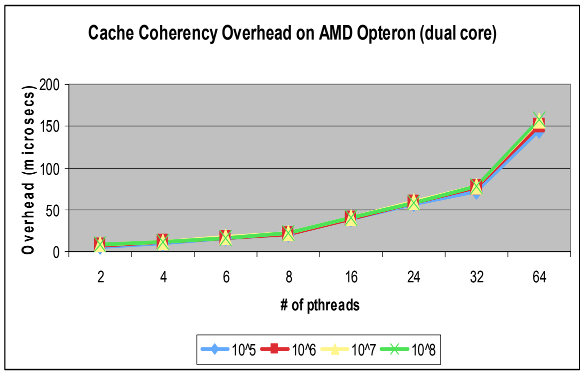

Figure 2: Cache Coherency Overhead on AMD Opteron processor

Figure 3 and Figure 4 demonstrate the results gathered from HPCC Stream benchmark on the UltraSPARC T1 processor and the AMD Opteron processor respectively. The figures include all different cases provided in the Stream benchmark by modifying the original implementation according to our needs using POSIX threads library. Although the original Stream benchmark is based on floating point computations the following results are based on integer operations. The reason for this change is to avoid any performance degrade that might have occurred due to the one floating point unit (FPU) for six cores on the UltraSparcT1 chip. In figure 3, we see a nice scaling up to 24 threads for the copy operation which actually is the physical limit since we have six cores and four threads per core. Scale and add operations also scale up to 24 threads but in a slower fashion resulting almost the half bandwidth compared to the copy operation. However, the triad operation which is a combination of both scaling and addition operations scales only up to 6 threads. The reason behind this behavior is most likely based on the performance of the Modular Arithmetic Unit (MAU) on the UltraSparcT1 chip where there is one MAU per core. Since copy operation is not affected by this unit it results in a bigger bandwidth but the scale, add and triad operations are directly affected from the capabilities of these units. Figure 4 represents consistent trend between different operations. It is easy to understand the physical limit of the system considering the curves scale up to 2 threads and start hanging in a steady state after that.

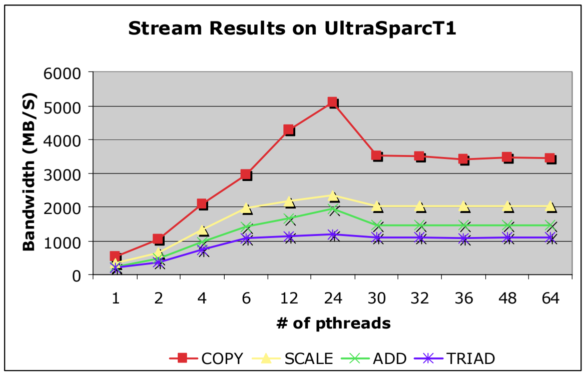

Figure 3: Memory Bandwidth Results on UltraSPARC T1 processor

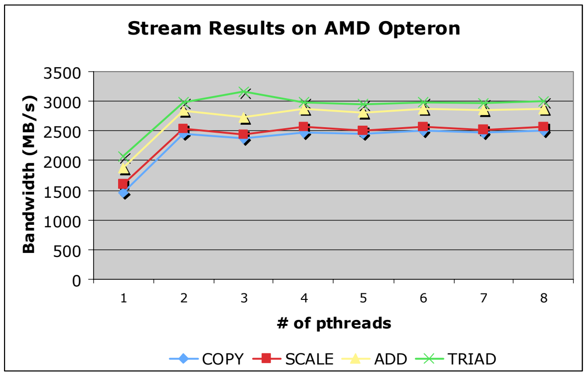

Figure 4: Memory Bandwidth Results on AMD Opteron dual core processor

Switching to the RandomAccess scheme from Stream, we used the dual core dual processor AMD Opteron and the same UltraSPARC T1 processor. Figure 5 illustrates the results for UltraSPARC T1 processor which actually presents a different result compared to the earlier analysis provided on Figure 1. In fact, this difference comes from the difference between Stream and RandomAccess benchmarking tests. RandomAccess provides its results in terms of Giga updates per Second (GUPS), which is a measurement that profiles the memory architecture of a system. UltraSPARC T1 processor used in our experiments has 6 cores and accordingly 12 memory banks which makes the actual physical limit for RandomAccess scheme. Figure 5 shows this limit and the actual GUPS values obtained on this processor. Figure 6 represents a clear indication of the physical limit in the test bed utilized which includes a dual core/dual processor (total of 4 processing cores) AMD Opteron processor as well as the GUPS values from this system.

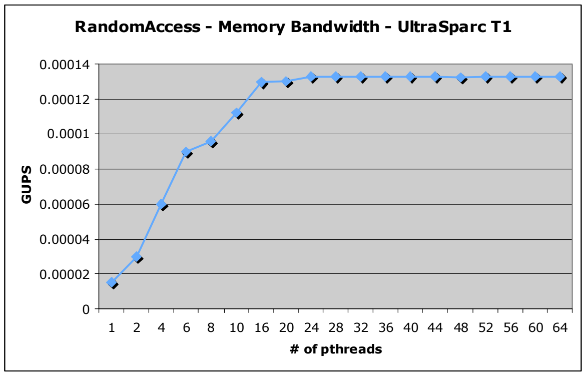

Figure 5: Memory Bandwidth analysis on UltraSPARC T1 processor using RandomAccess scheme

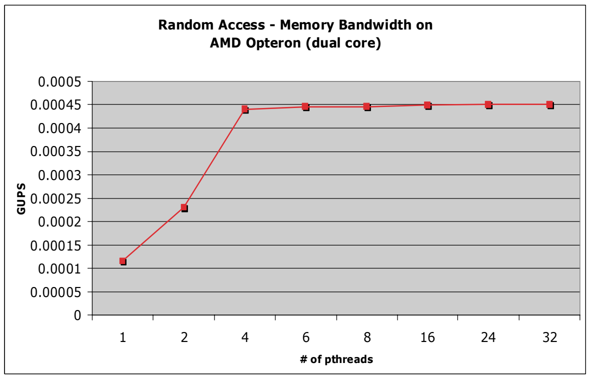

Figure 6: Memory Bandwidth analysis on AMD Opteron processor (dual core/dual processor) using RandomAccess scheme

Figure 7 and Figure 8 were also obtained from a similar test bed and the same test case which further shows the same indications as Figure 5 and Figure 6. However, these figures also present results for another performance metric which is the latency observed from the cores to the main memory.

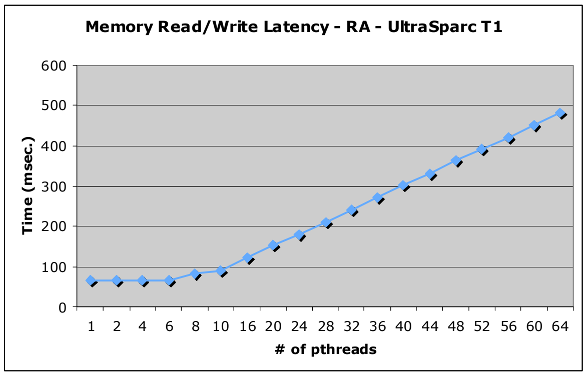

Figure 7: Memory Read/Write Latency on UltraSPARC T1 processor

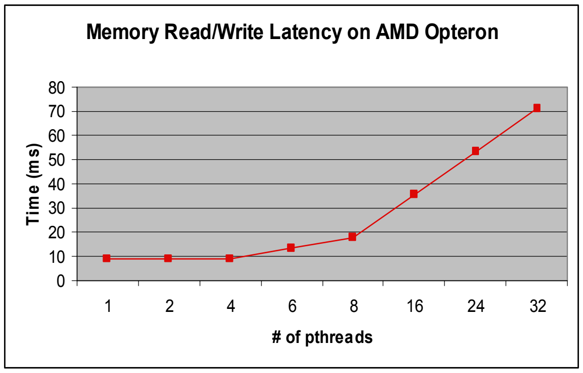

Figure 8: Memory Read/Write Latency on AMD Opteron dual core/dual processor

Furthermore, Figure 9 and Figure 10 demonstrate the memory contention experiments on both of these systems. In both of the figures, the y-axis represents the values for the metric that was stated earlier in Table 1 which is the percentage of memory contention between the cores while accessing the main memory.

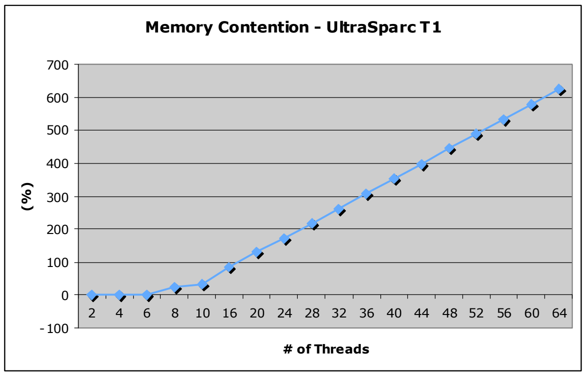

Figure 9: Memory Contention analysis on UltraSPARC T1 processor

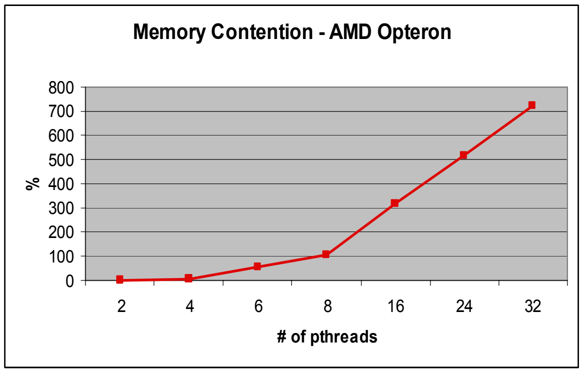

Figure 10: Memory Contention analysis on AMD Opteron processor

Finally, in addition to all the aspects and sub-features described in section 2 we used one application benchmark called "FFTW" in our benchmarking suite to further test these architectures for different type of workloads. Figure 11 represents the FFTW results on UltraSPARC T1 processor. It is not fair to compare these results with the ones from AMD Opteron as shown in Figure 12, since FFTW application is heavily based on floating point arithmetic and UltraSPARC T1 processor has only a single Floating Point Unit (FPU) for the whole chip, in other words all 6 cores and 24 hardware strands are using the same FPU. However, these values give us a better understanding of the FPU on the UltraSPARC T1 processor as it provides scalability up the number of cores as it is shown in Figure 11. On the other hand, Figure 12 represents the same experimental analysis on the AMD Opteron processor in which each core has its own dedicated FPU.

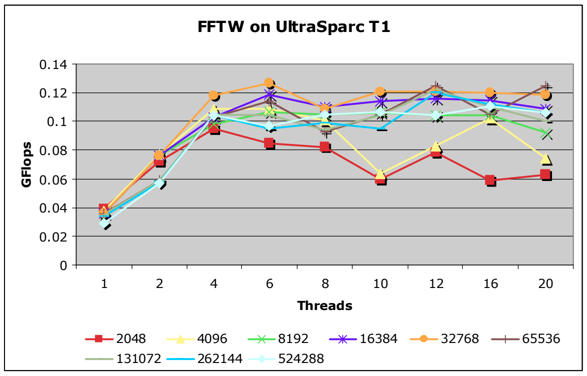

Figure 11: FFTW on UltraSPARC T1 processor

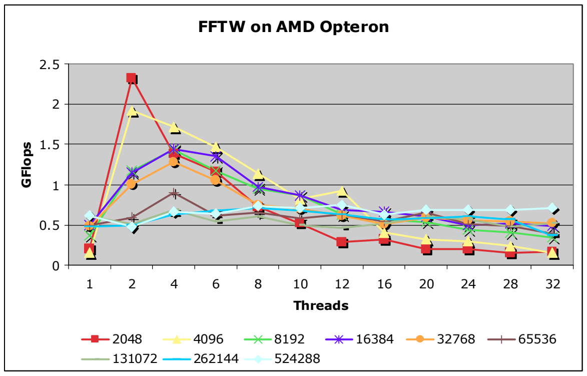

Figure 12: FFTW on AMD Opteron processor

4. Conclusion and Future Remarks

We presented a benchmarking methodology including some multi-core specific performance metrics as well as test cases for the emerging multi-core systems. As we see more and more of these multi-core systems are provided from the processor vendors, also considering the fact that the tendency of putting more processing cores inside a single chip grows it is crucial to understand the advantages and challenges come up with these systems. This study was intended as a case study to provide such information covering different features and layers of multi-core systems. Experimental analysis was conducted on the UltraSPARC T1 (six cores) and the AMD Opteron (dual core) processors as a proof of concept. The findings and observations from test cases were presented to give a better understanding of the multi-core systems in general. As a future study, utilizing hardware counters in the benchmarking suite might give a better understanding of the results and lead to more findings. Also as an extension to this study, in order to exploit the software aspect like the operating system (OS), we are planning to use a lightweight OS maybe just a scheduler to measure the OS overhead and efficiency while running jobs in these multi-core systems.

References

[1] http://multicore.amd.com/GLOBAL/WhitePapers/Multi-Core_Processors_WhitePaper.pdf

[2] Hofstee, H.P., "Future microprocessors and off-chip SOP interconnect," Advanced Packaging, IEEE Transactions on [see also Components, Packaging and Manufacturing Technology, Part B: Advanced Packaging, IEEE Transactions on], vol.27, no.2pp. 301- 303, May 2004

[3] Balakrishnan, S.; Ravi Rajwar; Upton, M.; Lai, K., "The impact of performance asymmetry in emerging multicore architectures," Computer Architecture, 2005. ISCA '05 Proceedings 32nd International Symposium on , vol., no.pp. 506- 517, 4-8 June 2005

[4] Kistler, M.; Perrone, M.; Petrini, F., "Cell Multiprocessor Communication Network: Built for Speed," Micro, IEEE , vol.26, no.3pp. 10- 23, May-June 2006

[5] http://icl.cs.utk.edu/hpcc/

[6] http://www.cs.virginia.edu/stream/

[7] http://icl.cs.utk.edu/projectsfiles/hpcc/RandomAccess/

[8] http://www.fftw.org/

[9] http://www.sun.com/processors/UltraSPARC-T1

[10] http://www.amd.com/usen/Processors/ProductInformation/0,,30_118_8825,00.html

[11] http://www.opengroup.org/austin/papers/posix_faq.html

[12] Kumar, R.; Zyuban, V.; Tullsen, D.M., "Interconnections in multi-core architectures: understanding mechanisms, overheads and scaling," Computer Architecture, 2005. ISCA '05 Proceedings 32nd International Symposium on, vol., no.pp. 408- 419, 4-8 June 2005

[13] Jichuan Chang; Sohi, G.S., "Cooperative Caching for Chip Multiprocessors," Computer Architecture, 2006 33rd International Symposium on , vol., no.pp. 264- 276, 17-21 June 2006

[14] Ying Zheng; Davis, B.T.; Jordan, M., "Performance evaluation of exclusive cache hierarchies," Performance Analysis of Systems and Software, 2004 IEEE International Symposium on - ISPASS, vol., no.pp. 89-96, 2004

[15] Yeh, P.C.C.; Patel, J.H.; Davidson, E.S., "Shared Cache for Multiple-Stream Computer Systems," Computers, IEEE Transactions on , vol.C-32, no.1pp. 38- 47, Jan 1983

[16] http://www.sun.com/servers/coolthreads/coolthreads_architecture_wp.pdf

[17] http://multicore.amd.com/en/

[18] http://www.embedded.com/showArticle.jhtml?articleID=183702075

[19] http://www.sun.com/processors/whitepapers/throughput_whitepaper.pdf