Evaluating Heterogeneous Memory Model by Realistic Trace-Driven Hardware/Software Co-simulation

Wei Wang[1,2], Qigang Wang[1], Wei Wei[1], and Dong Liu[1]

[1] Intel Corporation
No.2 Kexueyuan south road, Haidian, Beijing, P.R. China, 100080
{wei.wang,qigang.wang,wei.a.wei,dong.liu}@intel.com
[2] Department of Electronic Engineering
Beijing Institute of Technology, Beijing, P.R. China, 100081
vanvane@gmail.com

Abstract. Traditional DRAM has faced more challenges in the memory subsystem. Meanwhile, more types of memories become available as new technologies have been developed in many areas. In this case, the unified memory architecture should be changed to a heterogeneous one to utilize the new memories and obtain optimal performance in terms of memory access latency and life time. In this paper, a hierarchical model is studied and compared with a flat model. To evaluate our designs, the system bus trace is collected for realistic trace-driven simulation. We use typical server benchmark SPEC jbb2005 and typical desktop benchmarks Quake 3 and SYSmark 2007 as our evaluation workloads. The experimental results show that the performance of proposed hierarchical model is very stable in writing access and its average reading access time is not sensitive to its associativity.

Keywords: Memory architecture, Performance model, Trace-driven simulation.

1 Introduction

The performance growth of CMP processors will result in DRAM disaster [4]. Meanwhile, Phase-Change RAM (PCRAM, PCM), Magneto Resistive RAM (MRAM) and Ferromagnetic RAM (FRAM) are three main contenders in the race for replacing flash, SRAM and DRAM. Twin Transistor RAM (TTRAM) and Zero capacitor RAM (Z-RAM) are considered to compete with SRAM and DRAM in the future. This paper focuses on exploring the potential of heterogeneous memories and evaluating the architectural options in detail.

Different from the predominantly unified main memory system, in this paper we propose and investigate a mixed memory model that may adopt two or more types of memory. Our proposed heterogeneous memory system consists of 2 or more parts, each of which is denoted as M_j, and part M_i has shorter latency than M_{i+1}. Such a memory subsystem brings new opportunities in cutting cost and shrinking size for main memory.

While access to M_1 can be carried out at the fastest speed, access to lower levels has longer latency. The problem addressed here is how to design heterogeneous

E. Luque, T. Margalef, and D. Benítez (Eds.): Euro-Par 2008, LNCS 5168, pp. 182–191, 2008.

memory subsystem to organize different memory types so that we get the optimal performance, shorter latency and longer life time.

To simplify the problem without loss of generality, in this study we consider the case when only two different types of memories are available. One type of memory is faster, yet smaller and more expensive, while the other is slower, yet larger and cheaper. We do realistic trace driven experiments to evaluate the performance of heterogeneous memory models.

The rest of the paper is organized as follows. In Section 2 we propose two different memory models, the hierarchical model and the flat one. In Section 3 we present details for our experimental system. We evaluate our models with benchmarks in Section 4. In Section 5 we conclude and present the future work.

2 Heterogeneous Memory Architectures

In this section, we introduce two heterogeneous memory architecture models. Hierarchical model is derived from the traditional cache/memory/storage infrastructure, while flat model was originally used in embedded system including most DSPs and MCU supporting uniform memory addressing and accessing. Fig. 1 illustrates the architectures for hierarchical and flat memory models. We use Average Memory Access Time (*AMAT*) as our performance measurement during our experiments.

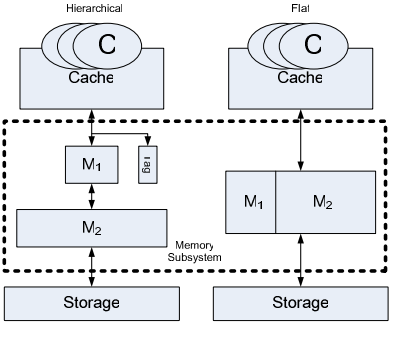


Fig. 1. Two memory models (Hierarchical and Flat)

2.1 Hierarchical Model

The idea of a hierarchical memory model is derived from the traditional memory hierarchy. In a hierarchical architecture, the upper level memory serves as an inclusive memory cache for lower level memory, where level M_i is included in level M_{i+1}. For a two level case, M_2 memory occupies the entire memory address space, while M_1 only occupies the hottest addresses. Tag is also a key consideration of memory hierarchy design. M_1 memory of larger line size and less associativity can reduce tag size. The time to find a victim tag for replacement also contributes to the miss penalty. But when compared with M_2 latency, as long as we keep this penalty low enough, the time to access and update tag could be neglected. In our experiment, SRAM tag is assumed with little overhead in tag comparison and refreshing, thus the tag time is not considered.

Table 1. Hierarchical memory model parameters

$L(atency)_{Mi\_rd}$	Time to read a page from M_i
$L(atency)_{Mi\_wr}$	Time to write a page to M_i
$missrate_{rd}$	Read misses / Read references
$missrate_{wr}$	Write misses / Write references
$dirtyrate_{rd}$	Writebacks occurred in read reference / Read misses
$dirtyrate_{wr}$	Writebacks occurred in write reference / Write misses

Some parameters are defined in Table 1 which will be used later to calculate *AMAT* for the proposed hierarchical model.

Then we can get *AMAT* by the following formula:

$$AMAT_{rd(wr)} = t_{hit\_rd(wr)} \times (1 - missrate_{rd(wr)}) + (t_{miss\_rd(wr)} + t_{dirty} \times dirtyrate_{rd(wr)}) \times missrate_{rd(wr)} \quad (1)$$

Where, $t_{hit\_rd} = L_{M1\_rd}$, $t_{miss\_rd} = \max\{L_{M2\_rd}, L_{M1\_wr}\}$, $t_{hit\_wr} = L_{M1\_wr}$, $t_{miss\_wr} = L_{M1\_wr}$, $t_{dirty} = \max\{L_{M1\_rd}, L_{M2\_wr}\}$. It is reasonable to assume L_{M2_rd} is longer than L_{M1_wr} and L_{M2_wr} is longer than L_{M1_rd}. When there is a reference from system bus to memory, the controller will first check its address in tag to see whether it hits in M_1 or not. A hit in M_1 will shorten the total access time to M_1, while a miss will result in longer reading access time due to loading data from M_2, finding a victim block and then further examining dirty bit in its tag. A dirty block will add extra time t_{dirty} to the access time as to backup the current content of the M_1 block to M_2. The dirty bit is cleared when the bock is brought in M_1 by a read reference and set after a write reference. In our first implementation of this hierarchical model, LRU algorithm is adopted for block replacement policy.

From formula (1), *AMAT* ratio of hierarchical model with different configurations (set associativity, M_1 size) can be written as below:

$$AMAT\frac{hierarchy}{hierarchy_{rd}} = \frac{(1 - missrate_{rd}) + \lambda(\theta \cdot dirtyrate_{rd} + 1) \cdot missrate_{rd}}{(1 - missrate'_{rd}) + \lambda(\theta \cdot dirtyrate'_{rd} + 1) \cdot missrate'_{rd}}; AMAT\frac{hierarchy}{hierarchy_{wr}} = \frac{(1 - missrate_{wr}) + (\eta \cdot dirtyrate_{wr} + 1) \cdot missrate_{wr}}{(1 - missrate'_{wr}) + (\eta \cdot dirtyrate'_{wr} + 1) \cdot missrate'_{wr}} \quad (2)$$

Different configurations will have different missrate and dirtyrate. An extreme case is when $missrate'_{rd(wr)} = 0$, we get *AMAT* ratio of hierarchical model to entire M_1 memory.

Where, $\lambda = L_{M2\_rd} / L_{M1\_rd}$, $\theta = L_{M2\_wr} / L_{M2\_rd}$, $\eta = L_{M2\_wr} / L_{M1\_wr}$. $\lambda(\eta)$ reflects the reading(writing) access time ratio of level-2 to level-1. θ reflects access time ratio of writing to reading in level-2.

2.2 Flat Model

In this architecture, M_1 and M_2 are both connected to the system bus and each occupies a fixed part of the total physical memory address space. A case for this architecture aims at memory resource QoS. As more threads/cores are enabled on die, the computing capability is exploited by simultaneously executing multiple tasks or applications. It is common that one application requires higher priority than the others, e.g., critical application running get higher priority than common business server application. Under such circumstance, QoS-aware architecture should be designed on

Table 2. Flat memory model parameters

$L(atency)_{Mi\_rd}$	Time to read a page from M_i
$L(atency)_{Mi\_wr}$	Time to write a page to M_i
Ref_{Mi_rd}	Read references to M_i
Ref_{Mi_wr}	Write references to M_i

die. Otherwise OS should manage and schedule the accesses to the two different types of memory. For example, a memory remapping layer may be added to support this. All memory references from the workload with higher priority will be redirected to M_1, and those from lower priority will be redirected to M_2. QoS policies or priority-based policies should be considered in this model [5].

The main issue is similar with hierarchical memory model. To better utilize the speed advantage of M_1, we need to prioritize system bus requests so that we get optimal system performance. As a first implementation of the studied system, we assume only high priority workloads can use M_1 memory and workloads with low priority can only use M_2 memory.

According to such assumption, the memory management module of OS is also the priority manager. It allocates memory in faster memory for higher priority tasks and allocates memory in slower memory for lower priority tasks. High priority tasks can also use slow memory if fast memory overflows, but low priority tasks can't use fast memory. *AMAT* can be calculated by the following formula and the variables are defined in Table 2.

$$AMAT_{rd(wr)} = \frac{Ref_{M1\_rd(wr)} \times L_{M1\_rd(wr)} + Ref_{M2\_rd(wr)} \times L_{M2\_rd(wr)}}{Ref_{M1\_rd(wr)} + Ref_{M2\_rd(wr)}} \tag{3}$$

From (1) and (3) *AMAT* ratio of hierarchical model to flat model can be written as below:

$$AMAT\frac{hierarchy}{flat}_{rd} = \frac{(1-missrate_{rd}) + \lambda(\theta \cdot dirtyrate_{rd}+1) \cdot missrate_{rd}}{(Ref_{M1\_rd} + \lambda \cdot Ref_{M2\_rd}) / (Ref_{M1\_rd} + Ref_{M2\_rd})}; AMAT\frac{hierarchy}{flat}_{wr} = \frac{(1-missrate_{wr}) + (\eta \cdot dirtyrate_{wr}+1) \cdot missrate_{wr}}{(Ref_{M1\_wr} + \eta \cdot Ref_{M2\_wr}) / (Ref_{M1\_wr} + Ref_{M2\_wr})} \tag{4}$$

From previous work [1], hierarchical model always performs much better than flat one in $AMAT_{wr}$ and write back ratio, and the $AMAT_{wr}$ of hierarchical one is shorter or longer than flat one (see Fig. 2) when λ, θ parameters change. The black region in Fig. 3 illustrates the scenario when hierarchical should be used to build a heterogeneous architecture. In this paper, we explore the hierarchical architecture in detail.

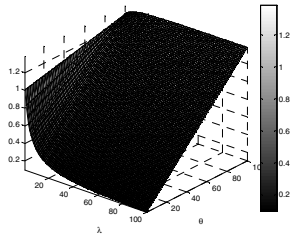


Fig. 2. $AMAT_{rd}$ Ratio

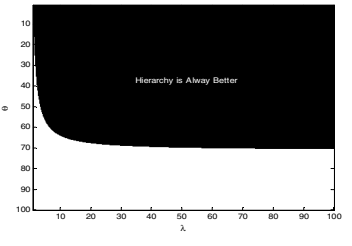


Fig. 3. Hierarchical VS. Flat

3 Research Methodology

To calculate *AMAT* with our formulas, lots of statistical results from trace-driven simulation are needed. In this section, we describe the trace-driven simulation for the proposed architectures.

There are two phases in trace-driven simulation: trace generation and simulation. In our experiment, the trace is collected from the system bus, Front Side Bus (FSB) of a machine which is running the realistic workloads and then analyzed by a simulator CASPER [3] which contains the detailed hierarchical and flat model implementations.

3.1 Physical Environment of the Simulation System

We developed an FSB transaction data collector called DragonHead (DH) 1.1. It is a system for cycle accurate long FSB trace collection and cache simulation. The experiment environment consists of eight main components, as is shown in Fig. 4.

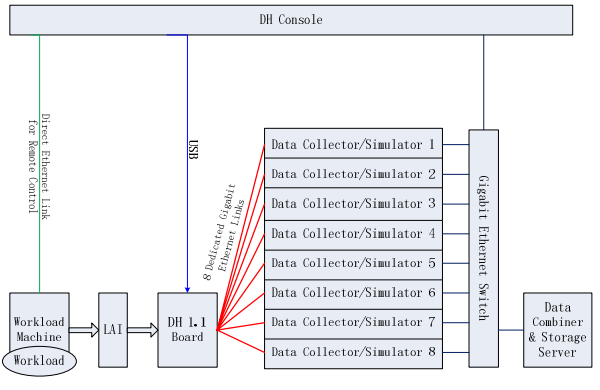


Fig. 4. Simulation System

- Workload Machine: A computer which hosts the workload and the Logic Analyzer Interposer (LAI) connectors. Its FSB traffic is captured by DH 1.1.
- Data Collector Cluster: A cluster employed to receive FSB traffic from DH 1.1 Board over Gigabit Ethernet cables.
- Data Combiner & Storage Server: A server used to receive the raw data from the Data Collection / Simulation Cluster, combine & refine the raw data to a trace file, and store the trace file. It also converts a trace file to CASPER format so that CASPER can do simulation on the converted file.
- Data Simulator Cluster: A cluster employed to run do simulations. It might be the same physical cluster of the DH data collector cluster.
- DH Console: A console which controls the whole system. It connects to the DH 1.1 Board through a USB interface, and connects to the Workload Machine, the Data Collector / Simulator Cluster and the Data Combiner & Storage Server with Ethernet link.

- DH 1.1 Board: A hardware board which collects FSB traffic and transports the collected traffic to DH data collection cluster over Gigabit Ethernet cables.
- Gigabit Ethernet Switch: It internally connects the DH Console, the Data Collection / Simulation Cluster, the Data Combiner & Storage Server, and optionally the Workload Machine;
- LAI: It connects the DH 1.1 board with the workload machine. It accepts the signals from the FSB of the workload machine.

With these components, our simulation system can generate traces from the memory references of workloads, and do simulations on these traces. In our experiment, the traces collected by DH1.1 are converted to a format which CASPER can handle.

3.2 Workload Machine

The workload machine is a dual-core Xeon server (detailed server configurations in Table 3).

Table 3. Server Configuration

CPU type	1.2GB Xeon Paxville (2 Cores with HT disabled)
Northbridge	Intel E7520
FSB frequency	100MHz
L1 Data cache	2 x 16 KBytes, 8-way set associative, 64-byte line size
Trace cache	2 x 12 Kuops, 8-way set associative
L2 cache	2 x 2048 KBytes, 8-way set associative, 64-byte line size
Page size	4096 Bytes
Memory Size	8192 MBytes

3.3 Workload

To evaluate the hierarchical model, we run standard server benchmark SPEC jbb2005 and two desktop benchmark workloads Quake 3 and SYSmark 2007. SPEC jbb2005 is a Java-based multi-threaded server benchmark that models a warehouse company with warehouses that serve a number of districts (much like TPC-C). SPEC jbb2005 is configured with 128 warehouses. Quake 3 is set to replay a DEMO, and SYSmark 2007 runs through all 4 parts: E-learning, Video Creation, Productivity and 3D.

An x86_64 kernel 2.6.18-8.el5 is installed in our workload machine as SPEC jbb2005 with our configuration consumes more than 4GB memory. For Quake 3 and SYSmark 2007, a 32bit Microsoft Windows XP with SP2 is also installed in our workload machine.

4 Experimental Result

In this section, we present the simulation result of the proposed heterogeneous memory architecture. The sensitivity of M_1 size and set associativity is shown with *AMAT* ratio and writeback to M_2 ratio.

4.1 Target Configurations

Various configurations of the studied models can be evaluated by altering parameters of the simulator. For hierarchical model, we evaluate the target systems with 8G M_2 memory and different M_1 memory sizes from 128MB to 4096MB with different associativity configurations, from 1-way to 64-way. The cache configurations of the targeted systems are the same with the workload machine (see Table 3) and configurations of the studied hierarchical memory system are shown in Table 4.

Table 4. Studied System Configurations

	Hierarchical
M_1 Size	128, 256, 512, 1024, 2048, 4096 MB
M_2 Size	8192 MB
Block Size	4096 B
Associativity	1, 2, 4, 8, 16, 32, 64 Way

4.2 Workload Characteristics

As our experimental data comes from the FSB, the 3 workloads have conspicuous differences in their memory reference behavior.

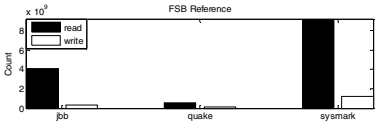
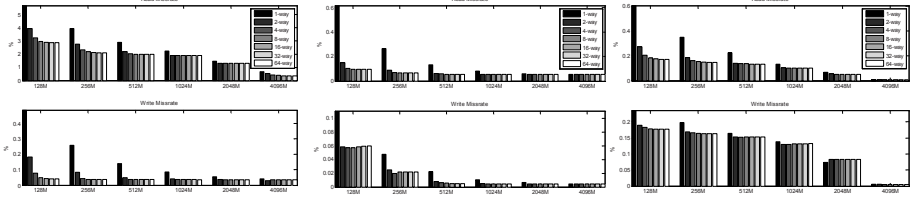


Fig. 5. Memory Reference

Missrate is a key factor in hierarchical *AMAT* model. The following figures show missrates for different M_1 sizes from 128MB to 4096MB while associativity scales from 1-way to 64-way. Read missrate is usually higher than write missrate according to Fig. 6. For most workloads, higher associativity brings much lower missrate than direct mapped when the size of M_1 memory is smaller than the working set.

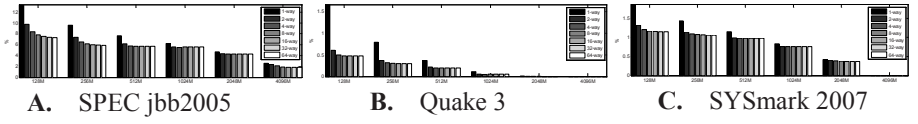


A. SPEC jbb2005 **B.** Quake 3 **C.** SYSmark 2007

Fig. 6. Missrate

4.3 Write Reference to M_2

Writeback is a vital indicator of the heterogeneous model when low speed memory has limited write times. From Fig. 7, we can conclude that higher associativity and bigger M_1 size of the hierarchical model help to reduce writebacks to M_2 for most workloads, thus give a longer life to M_2. But the performance will not go up with the growth of its set associativity when M_1 is bigger than 256M.

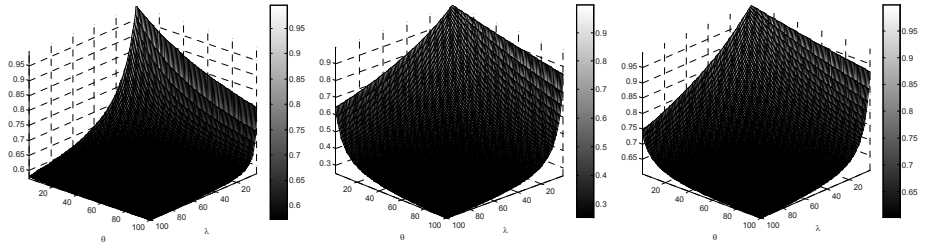


A. SPEC jbb2005 **B.** Quake 3 **C.** SYSmark 2007

Fig. 7. Writeback Ratio

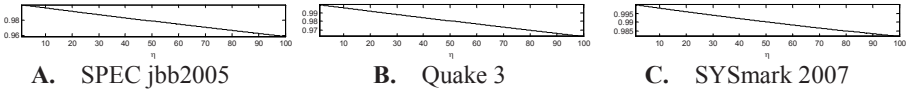
4.4 Sensitivity to M_1 Size

AMAT is our major performance measurement. We did experiments for hierarchical architecture with different configurations. It's obvious that larger M_1 will result in higher performance. However, 4x M_1 size doesn't bring back 4x performance. For the best case Quake 3, the minimal $AMAT_{rd}$ ratio is 0.3 according to Fig. 8. And for write access, the performance is not so sensitive to M_1 size.



A. SPEC jbb2005 **B.** Quake 3 **C.** SYSmark 2007

Fig. 8. $AMAT_{rd}$ Ratio of 512MB M_1 to 128MB M_1 of 1-way set associative



A. SPEC jbb2005 **B.** Quake 3 **C.** SYSmark 2007

Fig. 9. $AMAT_{wr}$ Ratio of 512MB M_1 to 128MB M_1 of 1-way set associative

4.5 Sensitivity to Associativity

Associativity helps to reduce the $AMAT_{rd}$. but similar to the performance of writeback ratio, the improvement grow little when M_1 size grows up according to Fig. 10 and Fig. 12.

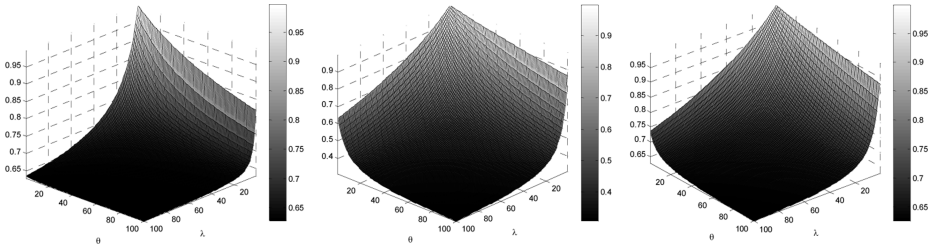


Fig. 10. $AMAT_{rd}$ Ratio of 4-way to 1-way Memory with 128M M_1

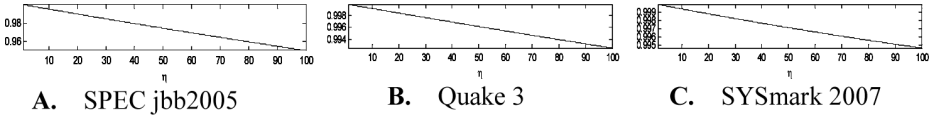


Fig. 11. $AMAT_{wr}$ Ratio of 4-way to 1-way Memory with 128M M_1

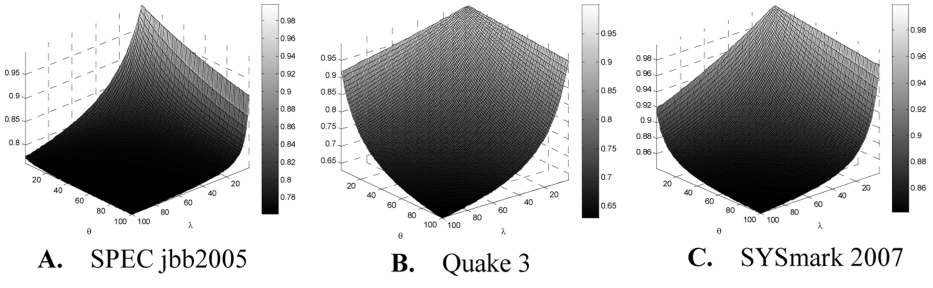


Fig. 12. $AMAT_{rd}$ Ratio of 4-way to 1-way Memory with 512M M_1

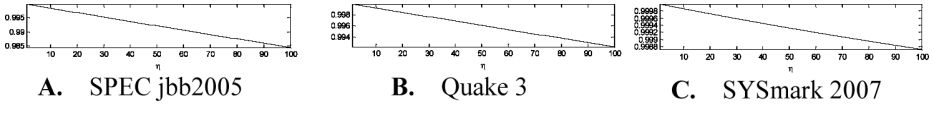


Fig. 13. $AMAT_{wr}$ Ratio of 4-way to 1-way Memory with 512M M_1

From Fig. 11 and Fig. 13, we can see that the write performance is nearly constant in the hierarchical model.

5 Conclusion and Future Work

In this paper, we have studied the performance a two-level hierarchical model where the first level is accessed at fast speed whereas the second level is a magnitude slower. By managing this, hierarchical model treats M_1 as a cache that uses LRU, write allocate and write back policy. We found the write performance is very stable.

And our experimental result suggests the performance of hierarchical model is not sensitive to its set associativity under most circumstances. So we should pay more attention to its size than its associativity when building a memory hierarchy.

Using a heterogeneous memory architecture opens up many opportunities due to cutting memory cost and reducing power consumption in future memory systems. We are doing research to further improve memory hierarchy on the following areas.

- Explore the potential impact of workload machine cache configurations.
- Do research in scheduling memory operation according to the hierarchical model for power saving.
- Consider other opportunities such as compression and intelligent prefetching schemes to optimize the hierarchical architecture.

References

[1] Wang, W., Wang, Q., Wei, W., Liu, D.: Modeling and Evaluating Heterogeneous Memory Architectures by Trace-driven Simulation. In: MAW 2008, Ischia, Italy, May 5 (2008)

[2] Hennessy, J.L., Patterson, D.A.: Computer Architecture – A Quantitative Approach, 4th edn. Morgan Kaufmann, San Francisco (2007)

[3] Iyer, R.: On Modeling and Analyzing Cache Hierarchies using CASPER. In: 11th IEEE/ACM Symposium on Modeling, Analysis and Simulation of Computer and Telecom Systems (October 2003)

[4] Ekman, M., Stenstrom, P.: A case for multi-level main memory. In: Proceedings of the 3rd Workshop on Memory Performance Issues: in Conjunction with the 31st international Symposium on Computer Architecture. WMPI 2004, Munich, Germany, June 20 - 20, 2004, vol. 68 (2004)

[5] Iyer, R., Zhao, L., Guo, F., Illikkal, R., Makineni, S., Newell, D., Solihin, Y., Hsu, L., Reinhardt, S.: QoS policies and architecture for cache/memory in CMP platforms. In: Proceedings of the 2007 ACM SIGMETRICS international Conference on Measurement and Modeling of Computer Systems (SIGMETRICS 2007), San Diego, California, USA, June 12 - 16 (2007)