When the ARM weakly consistent memory model meets speculation: is it necessary?

Tao C. Lee and Marc-Alexandre Boéchat
CS, *EPFL*

***Abstract*—Aggressive memory-level-parallelism techniques have provided significant performance gain in Distributed Share Memory Designs. In this paper, we reevaluate speculative memory ordering in the context of Chip Multi-Processors (CMPs) and power-limited computation. We evaluate relative performance between Sequential Consistency, Total Store Order and Relaxed Memory Order on a selection of modern workloads to predict the performance of the ARM weakly consistent memory model.**

***Index Terms*—memory ordering, memory consistency models, the ARM weakly consistent memory model.**

I. PROBLEM DEFINITION AND MOTIVATION

Memory accesses are a key factor in performance in modern multiprocessor systems. Because of their inherent serial nature, store operations degrade parallelism. Hardware techniques have been proposed to alleviate this problem, however, the solutions largely concentrate on Distributed-Shared Memory (DSM) server systems, and research on embedded systems are rare, not mentioning the cross study in between.

Recent advances in embedded processors have made their energy efficiency and growing computing power attractive for a number of reasons [11, 12, 14, 15, 16]. We argue that it is worth pursuing to study the performance benchmarks of memory consistency models in embedded systems and compare them with those of sever systems. We are particularly interested in the ARM Weakly Consistent Memory Model (ARM WCMM) because the wide applications of ARM processors in embedded systems make it as a representative case.

Recent research results have shown that speculative memory ordering techniques could outperform Relaxed Consistency (RC) with sufficient hardware support [3]. Since it is widely regarded that ARM WCMM can be classified within RC, we are interested in the results of comparing the performance of speculative memory ordering techniques with ARM WCMM and investigate the deviations compared with [3]. In particular, we would like to extend the basis this work to a more general setting, and form the foundation of the study of the effectiveness of speculative memory ordering within the framework of ARM WCMM.

The remainder of this paper is divided as follows. In section 2, we cover background information about memory ordering and make a survey of the work already done in this area. We describe our methodology in section 3 and present the parameters of our design in section 4. We expose our results in section 5 and conclude in section 6.

T. C. Lee is a master student of Computer Science, EPFL.
M.-A. Boéchat is a master student of Computer Science, EPFL.

II. BACKGROUND AND RELATED WORK

Memory ordering stalls have induced significant performance limitation in modern multiprocessor systems. Because of their inherent serial nature, store operations degrade parallelism, wasting computing power by letting processors wait for data to process. Hardware techniques have been proposed to alleviate this problem, however, they all rely on the assumption that the correctness of computing operations is guaranteed.

Today, programmers can choose a wide spectrum of memory consistency models. Depending on the model selected, memory ordering poses different challenges both to programmers and to system performance optimization. Sequential Consistency (SC) requires all memory accesses to be in the order specified by the compiler. This model is the most intuitive for programmers but it is also the most restrictive for performance improving mechanisms; Processor Consistency (PC) relieves the ordering constraints of load operations, but requires that stores follow the sequential order in the programs; Relaxed Consistency (RC) relieves all the ordering constraints, but depends on specialized fence (memory barrier) instructions to enforce memory ordering when it is needed. Conventionally, RC exhibits the best performance but deferred the synchronization burdens to the programmers.

The advances of hardware speculations techniques have changed the conventional memory ordering landscape [2]. It has been shown with sufficient hardware support (e.g. cache, lookup table, store buffer, etc.) SC with hardware speculation could perform as well as RC.

Recent study in speculative memory ordering have shown that with high-performance hardware speculation, PC could even outperform RC [3], essentially breaking the conventional wisdom that RC is necessary and sufficient to reach the best performance. The physical interpretation behind this remarkable result is that effective hardware speculation could eliminate the performance overheads of fence instructions, making one step further toward store-wait-free mechanisms.

More generally, performance-transparent memory ordering techniques have been proposed to provide speculative memory ordering for all memory consistency models (SC, PC, RC) [4]. The results have shown that speculative techniques are not restricted to any kinds of memory consistency models but could be implemented whenever performance improvement is

concerned.

Unfortunately, the advances of speculative memory ordering have been largely restricted to server systems. One possible explanation is that embedded systems in the past emphasize on energy efficiency rather than pursuing performance. Processors developed for embedded systems rely on limited energy resources – batteries, so that a more conservative approach to speculation is expected. Nevertheless, with the spread of smart hand-held systems, the computing power of embedded systems are growing and catching up with the server systems [11, 12]. Even more interestingly, the energy-efficient benefits of embedded processors are increasingly valued in modern computer systems as conventional server systems hit unprecedented power walls [14, 15, 16], and proposals for running server systems with embedded architectures are thriving [11, 12].

In this paper, we investigate ARM WCMM, and whether it would be useful for performance improvement in server systems. In particular, we are interested in the performance of ARM WCMM running server workloads in comparison with reported leading speculative memory consistency models for server systems. The results of this study might lead to improved understanding of the prospects of running server workloads with embedded processors [11, 12]. The progress of multiprocessors systems have led to slight different implementation of memory consistency models within the same class. Processor vendors such as Alpha, Intel, and Sun all implement their memory consistency models. As representative cases in this paper, we study conventional SC, Total Store Order (TSO, the SPARC version of PC) and Relaxed Memory Order (RMO, the SPARC flavor of RC) memory consistency models, and ARM WCMM of ARM multiprocessor platforms.

This study turns out to be less straightforward as it seems because the disclosure of detailed implementations of ARM WCMM is limited to open access. However, recent interests in the correctness of memory consistency implementations in multiprocessors have spurred some open research literature [5, 6, 7, 8, 9]. Critical reasoning of ARM WCMM [5, 6] has disclosed some interesting nature of ARM WCMM that is worth studying. More specifically, the possible realization of the observer model [5] in multiprocessor systems, and its performance impacts on memory ordering is of particular interests.

A large number of papers [1, 2, 3, 4] are dedicated to improving the performance of speculative memory ordering for server systems. However, the studies of such mechanisms for embedded systems are rare, and cross studies are even less common. We argue that this study is worthwhile for bridging the gap between server systems and embedded systems – both for energy efficiency and performance.

III. EXPERIMENTAL METHODOLOGY

Using FLEXUS full-system simulator [13], we run timing simulation of CMPs for selected workloads with different memory consistency models, CPU classes and L2 access latency. The workloads are selected from representative server workloads (OLTP, Web, DSS) and emerging could workloads [18].

In the first stage, we simulate conventional memory consistency models (SC, TSO, RMO) and measure their performance in CMPs with aggressive cores. Particular analysis would be put on the comparative study of TSO and RMO, for this serves as one of our standpoints for further study. Then we increased shared L2 access latency to see if we could get results comparable to those published in [3] about DSM.

In the second stage, we focused on the behavior of simpler cores, both on server workloads to be compared to the results found in stage one and on emerging cloud workloads to see if there are fundamental differences induced by the memory-model used.

Implementation of ARM WCMM in FLEXUS simulator would require adding some code to the existing RMO mechanisms such that under ARM WCMM mode, the behavior is slightly different than the existing RMO model. This modification, however, will preserve all the benefits of the original implementations reported in [3] (correctness, RMW fence insertion), and only add necessary code for the exploration of ARM WCMM. For time reason, however, we did not implement this functionality.

The modification can be viewed as an extension to the existing FLEXUS 4.0 release, and would add an ARM WCMM model in the similar role of RMO in the FLEXUS simulator. For time reason, we left this to future work.

IV. DESIGN PARAMETERS

We ran timing simulation on FLEXUS to explore the impact of different parameters, in this section we expose the details of our system.

A. Workloads

We used two classes of workloads: modern server workloads and emerging cloud workloads.

Our server workloads configuration is given in Table 1. Those workloads are comparable to the one used in [3] and are a representative subset of their category. Those six workloads can be broken down in tree categories: Online Translation Processing using well-known benchmark TPC-C on DB2 and Oracle database. Apache and Zeus workloads are canonical web server workloads. We used another well-recognized benchmark for Decision Support System type workloads known as TPC-H, we also selected Query 2 (scan dominated) and Query 17 (join dominated).

Table 1. Server workloads parameters.

Online Transaction Processing (TPC-C)	
DB2	100 warehouses (10 GB), 64 clients, 450 MB buffer pool
Oracle	100 warehouses (10 GB), 16 clients, 1.4 GB SGA
Web Server	
Apache	16K connections, fastCGI, worker threading model
Zeus	16K connections, fastCGI
Decision Support (TPC-H on DB2)	
Qry 2	Scan-dominated, 450 MB buffer pool
Qry 17	Join-dominated, 450 MB buffer pool

The Cloud workloads we used are studied in [18]. The setup used for those workloads is given in Table 2. WordCount uses the hadoop framework to implement Map-Reduce. Cloud9 is a simulation tool to find bugs in software. Nutch is a Web Search engine fed with data crawled from the public internet. Cassandra is a Data Serving application while Web09_bank is fulfilling the classical task of Web Frontend.

Table 2. Cloud workloads.

WordCount	4GB set of Wikipedia pages
Coud9	Analyze of the command-line *printf* utility from the GNU CoreUtils 6.10
Nutch	Index size : 2GB data segment size: 23GB
Cassandra	30GB YCSB data-set
Web09_bank	Web frontend

B. System setup

We also used two different setups. Both are CMPs but while the first is using SUN SPARC III aggressive cores, the second is using more simple ARM Cortex-A15 cores.

System details are listed in Tables 3 and 4. The main differences are the size of the reorder buffer, the size of the store buffer and size of the caches. Also, the ARM cores are designed to run at a lower frequency than their SPARC counterparts. Both CMPs are using tiled mesh interconnect.

Both systems have 16 processors on a single chip.

Table 3. Aggressive system parameters.

Processing Nodes	UltraSPARC III V9 ISA 4 GHz 8-stage pipeline; out-of-order 4-wide dispatch / retirement 96-entry ROB, LSQ 32-entry conventional store buffer
L1 Caches	L1 Caches Split I/D, 64KB 2-way, 2-cycle load-to-use 3 ports, 32 MSHRs
L2 Cache	L2 Cache Unified, 8MB 8-way, variable hit latency 1 port, 32 MSHRs
Main Memory	Main Memory 3 GB total memory 22.5 ns access latency 64 banks per node 64-byte coherence unit
Protocol Controller	Protocol Controller 1 GHz microcoded controller 64 transaction contexts
Interconnect	Interconnect 16x3 mesh 25 ns latency per hop 128 GB/s peak bisection bandwidth

Table 3. Simple cores system parameters.

Processing Nodes	ARM Cortex-A15 2GHz 2-wide dispatch / retirement 60-entry ROB 16-entry store buffer
L1 Caches	L1 Caches Split I/D, 32KB 2-way
L2 Cache	L2 Cache Unified, 4MB 16-way, 25-cycle hit latency 1 port, 32 MSHRs
Main Memory	Main Memory 3 GB total memory 22.5 ns access latency 64 banks per node 64-byte coherence unit
Interconnect	Interconnect 16x3 mesh 25 ns latency per hop 128 GB/s peak bisection bandwidth

V. RESULTS

We developed our project in a two-step strategy, first we tried to reproduce the results published in [3] and more specifically the relationship in the performance between the different memory ordering mechanisms.

Then we tried to figure out how ARM WCMM would behave, based on the results obtained for RMO. We used both simple and more aggressive cores to see if WCMM is more suited for one class of processors.

A. Step one: reproducing ISCA 2007 results

Since the FLEXUS version we used, FLEXUS 4.0, did not support DSM systems simulation, we tried to simulate a CMPs system with increased latency to L2 cache. We ran our simulation with tree different lookup latencies: 10, 50 and 100 cycles, hoping that the bigger the latency the closer we would get to a DSM behavior. We encountered a problem with Zeus workload that could not be fixed before submitting this report, so the data for Zeus with latencies 50 and 100 are missing.

We show the graph published in [3] in figure 1 and the results we obtained in Figures 2 to 4. We were unable to get the same difference in the performance of SC, TSO and RMO. The results we obtained show a much smaller performance gain when using advanced memory ordering mechanism. Since we used similar workloads and system settings, we argue that there is a fundamental difference in using memory-level-parallelism in DSM and CMPs.

Our results show really little difference in performance between RMO and TSO. The advanced memory ordering schemes still provide better performance than SC, but again, the improvement here is not as big as the one shown in Figure 1.

Moreover, we have to notice that increasing L2 latency did not show the effect we were expecting, reducing the gap between SC, TSO and RMO even more.

According to these results, we draw the conclusion that ARM WCMM model would be unable to demonstrate significant performance improvement when using aggressive cores on CMPs.

Since ARM processors are less aggressive than the SPARC III we simulated in this phase, we were interested to see if using simpler cores in CMPs gives a better chance to ARM WCMM.

B. Step two: what would happen when using simpler cores?

Figure 5 shows the behavior of our server workloads when running on a simple core CMPs. We see right away that the performance gaps between SC, TSO, and RMO are more significant.

We also note that the improvement from RMO to TSO is bigger. We expect ARM WCMM to perform at least as well as RMO.

Those two observations give us a clue why ARM is providing a relaxed memory model. While negligible on aggressive cores, the gain of using a relaxed memory model becomes significant when applied to simpler cores.

We wanted to know if this trend was also true for emerging cloud workloads. We kept the same simulation environments and selected a set of cloud workloads from [18] to evaluate their performance under the three different memory ordering paradigms.

Figure 6 shows the results we obtained. The differences are even more significant in this type of workloads.

VI. CONCLUSION

In this study, we showed that RC brings little performance gain for aggressive CMPs but is non-negligible when using simpler CMPs.

Using relaxed memory models on current aggressive CMPs makes therefore little sense since it does not bring much performance gain. But the power wall is encouraging systems to adopt simpler cores for better power efficiency. So ARM WCMM might become an interesting option for performance improvement in power-efficient systems.

Nevertheless, we showed in our experiments that TSO only induced 3% ~ 6% performance penalties compared to RMO, making it an attractive choice for its simpler programming model and wide uses in existing software systems even without using complex and power-consuming speculation hardware.

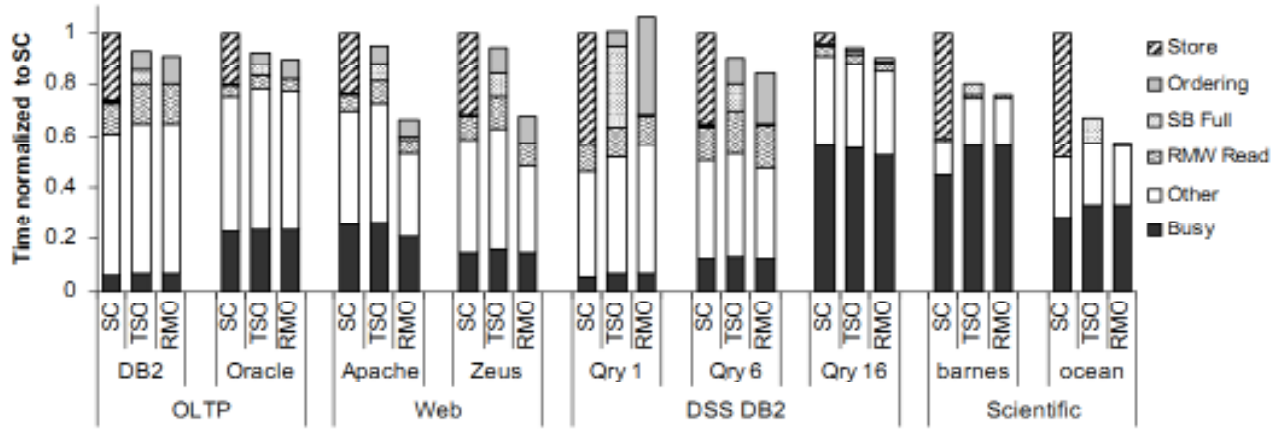

Figure 1: Execution time breakdown as published in [3] ISCA 2007

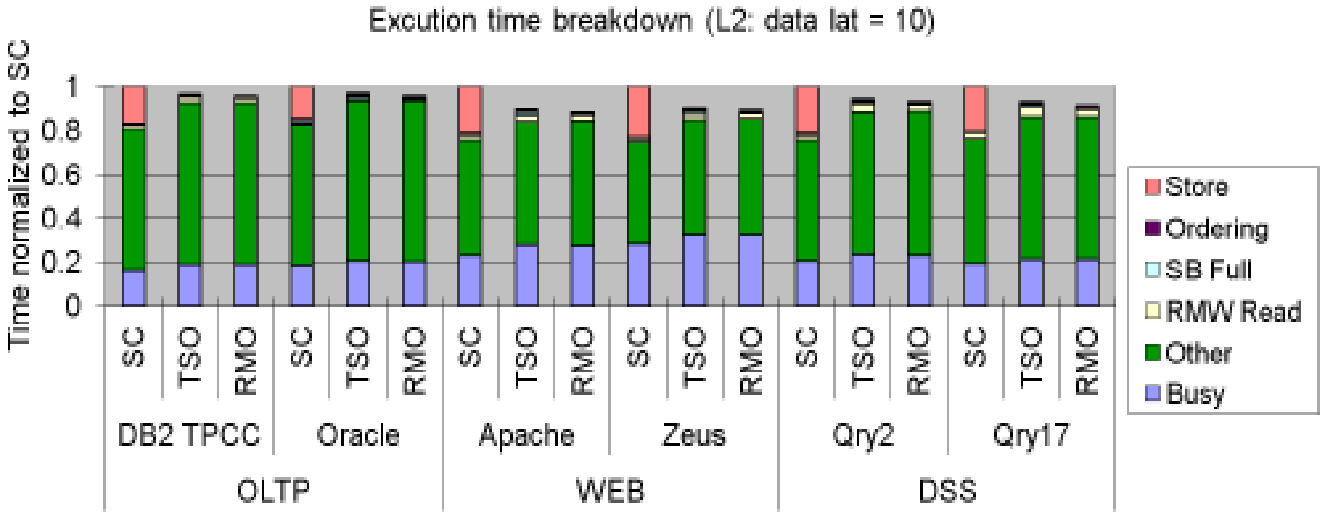

Figure 2: aggressive cores, server workloads, regular L2 latency

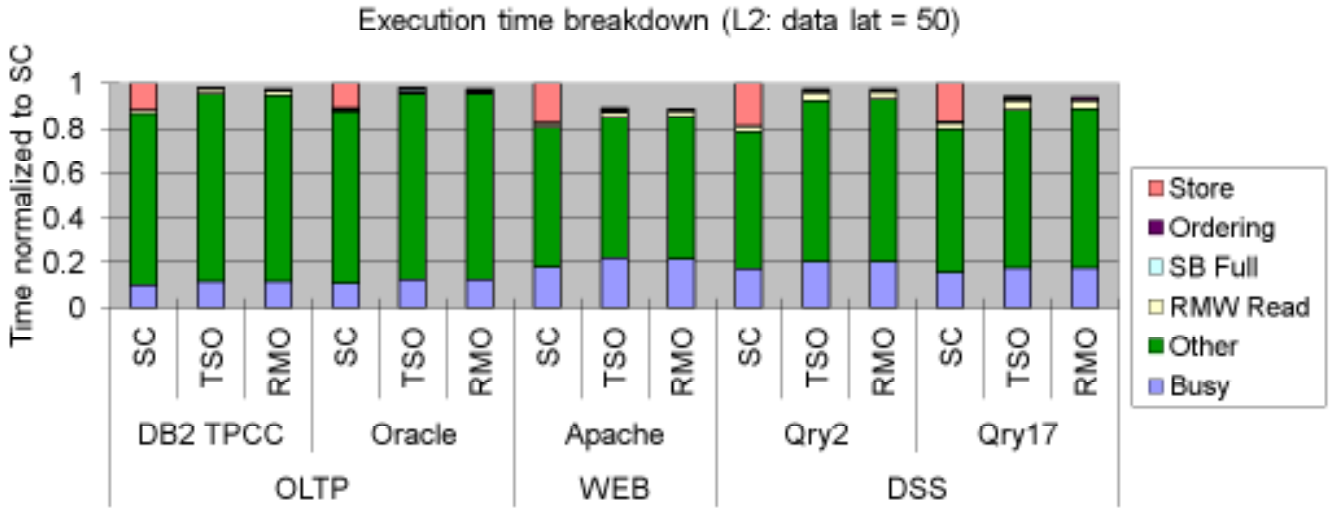

Figure 3: aggressive cores, server workloads, 5x L2 latency

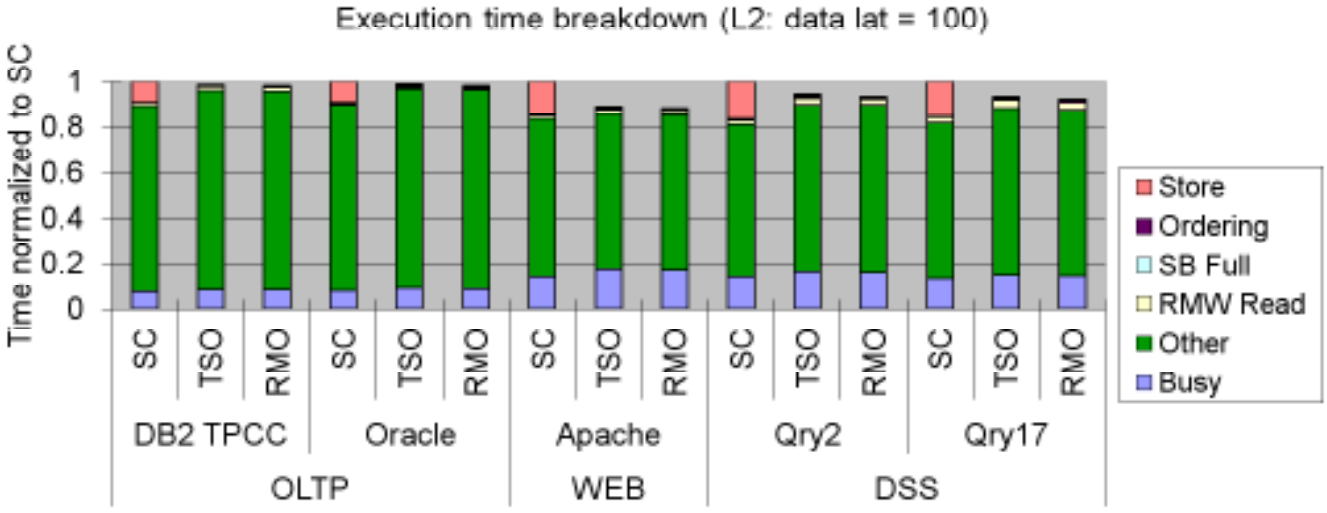

Figure 4: aggressive cores, server workloads, 10x L2 latency

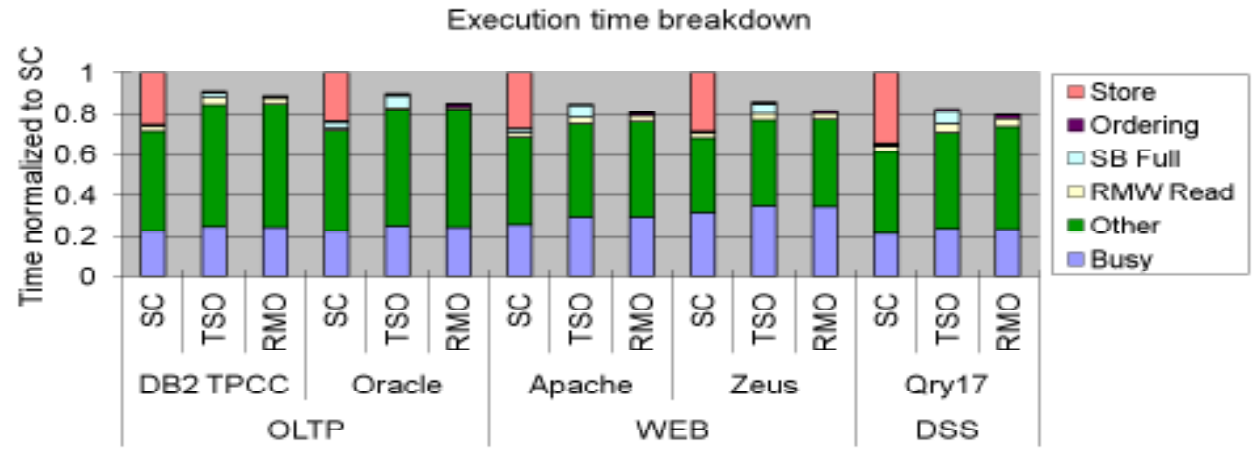

Figure 5: Timing breakdown for server workloads on simple cores

Figure 6: Timing breakdown for cloud workloads on simple cores

VII. References

[1] S. V. Adve and K. Gharachorloo. Shared memory consistency models: A tutorial. IEEE Computer, 29(12):66–76, Dec. 1996.

[2] C. Gniady, B. Falsafi, and T. N. Vijaykumar. Is SC + ILP = RC? Proc. of the 26th Int'l Symposium on Computer Archi-tecture, May 1999.

[3] Thomas F. Wenisch, Anastasia Ailamaki, Babak Falsafi and Andreas Moshovos. Mechanisms for Store-wait–free Multiprocessors. ISCA 2007.

[4] Colin Blundell. Milo M. K. Martin. Thomas F. Wenisch. INVISIFENCE: Performance-Transparent Memory Ordering in Conventional Multiprocessors. ISCA 2009.

[5] Nathan Chong, Samin Ishtiaq. Reasoning about the ARM weakly consistent memory model. MSPC 2008.

[6] Jade Alglave, Anthony Fox, Samin Ishtiaq, Magnus O. Myreen, Susmit Sarkar, Peter Sewell, Francesco Zappa Nardelli. The Semantics of Power and ARM Multiprocessor Machine Code. DAMP 2009.

[7] Jade Alglave, Luc Maranget, Susmit Sarkar, Peter Sewel. Litmus: Running Tests Against Hardware. TACAS 2011.

[8] Jade Alglave, Luc Maranget, Susmit Sarkar and Peter Sewell. Fences in Weak Memory Models. CAV 2010.

[9] Richard Grisenthwaite. ARM Barrier Litmus Tests and Cookbook. ARM Ltd. 2007.

[10] Paul E. McKenney. Memory Barriers: a Hardware View for Software Hackers. 2009.

[11] Emre Ozer. The State of the Future: ARM's Perspective. ARM Ltd. 2009.

[12] E. Özer, K. Flautner, S. Idgunji, A. Saidi, Y. Sazeides, B. Ahsan, N. Ladas, C. Nicopoulos, I. Sideris, B. Falsafi, A. Adileh, M. Ferdman, P. Lotfi-Kamran, M. Kuulusa, P. Marchal and N. Minas. EuroCloud: Energy-conscious 3D Server-on-Chip for Green Cloud Services. Workshop on Architectural Concerns in Large Datacenters in conjunction with ISCA-2010, June 2010.

[13] Thomas F. Wenisch and Roland E. Wunderlich. SimFlex: Fast, Accurate and Flexible Simulation of Computer Systems. Tutorial in the International Symposium on Microarchitecture (MICRO-38), November 2005

[14] Mark Horowitz, Elad Alon, Dinesh Patil, Samuel Naffziger, Rajesh Kumar, Kerry Bernstein. Scaling, Power, and the Future of CMOS. IEEE 2005.

[15] Rehan Hameed1, Wajahat Qadeer1, Megan Wachs1, Omid Azizi1, Alex Solomatnikov, Benjamin C. Lee, Stephen Richardson, Christos Kozyrakis, Mark Horowitz. Understanding Sources of Inefficiency in General-Purpose Chips. ISCA 2010.

[16] N. Hardavellas, M. Ferdman, B. Falsafi and A. Ailamaki. Toward Dark Silicon in Servers. IEEE Micro, July/August 2011.

[17] J. R. Platt. Strong Inference. Science 16 October 1964, Volume 146, Number 3642.

[18]M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A.D. Popescu, A. Ailamaki, and B. Falsafi, Clearing the Clouds: A Study of Emerging Workloads on Modern Hardware. 2011