Power-Efficient DRAM Speculation

Nidhi Aggarwal† Jason F. Cantin‡ Mikko H. Lipasti† James E. Smith†

†Electrical and Computer Engineering Dept.
University of Wisconsin, Madison
1415 Engineering Drive
Madison, WI 53705
{naggarwal, mikko, jes}@ece.wisc.edu

‡International Business Machines Corp.
11400 Burnet Road
Austin, TX 78758
jfcantin@us.ibm.com

Abstract

Power-Efficient DRAM Speculation (PEDS) is a power optimization targeted at broadcast-based shared-memory multiprocessor systems that speculatively access DRAM in parallel with the broadcast snoop. Although speculatively accessing DRAM has the potential performance advantage of overlapping DRAM latency with the snoop, it wastes power for memory requests that obtain data from other processors' caches. PEDS takes advantage of information provided by a Region Coherence Array to identify requests that have a high likelihood of obtaining data from another processor's cache, and does not access DRAM speculatively for those requests. By doing so, PEDS eliminates DRAM reads, reduces DRAM power consumption, reduces contention for DRAM resources, and increases the opportunity for DRAM power management. PEDS requires almost no additional hardware in systems that incorporate Region Coherence Arrays. Detailed simulation results show PEDS reduces average DRAM read traffic 28-32%, reduces average DRAM power dissipation 17-22%, and reduces average DRAM energy consumption 16-21%.

1. Introduction

Cache-coherent shared-memory multiprocessor systems have wide-ranging applications from commercial transaction processing and database services to large-scale scientific computing. They have become a critical component of internet-based services in general. As system architectures have grown to incorporate larger numbers of faster processors, power dissipation and energy consumption have become serious design constraints [1]. The power consumption of DRAM is now a first-class design consideration for shared-memory multiprocessor systems. For example, at a recent International Solid-State Circuits Conference (ISSCC), Sun Microsystems revealed that the power consumption of DRAM in the UltraSPARC T1 ("Niagara") systems running SPECjbb was approximately 60 watts [2]. This is approximately 22% of the total system power, nearly as much as all the processor cores consume.

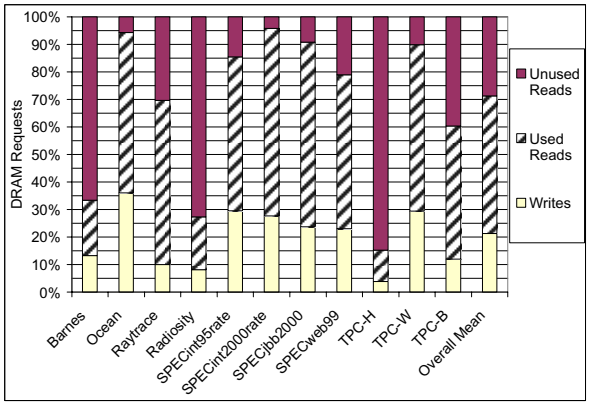

Figure 1: Breakdown of DRAM requests into Unused Reads, Used Reads, and Writes. Used Reads are read requests for which the DRAM data is used by the processor. Unused Reads are read requests for which the data was obtained from another processor's cache. Writes are DRAM write requests resulting from write-backs. See Table 2 for parameters.

Modern broadcast-based shared-memory multiprocessor systems commonly access DRAM speculatively to maximize performance [3, 4, 5]. The DRAM access is started after the memory controller receives the memory request but before the snoop response is available,

thereby overlapping the DRAM access with the remainder of the broadcast snoop. While many memory requests benefit from the lower latency of speculatively accessing DRAM, a significant fraction of requests obtain data from another processor's cache and do not use the data from DRAM. Simulation results for a four-processor system running a set of commercial, scientific, and multiprogrammed workloads indicate that 36.5% of the lines read from DRAM are not used because data is obtained from another processor's cache (See Figure 1). Approximately 28.7% of all DRAM accesses are unnecessary and waste power.

The data in Figure 1 is not surprising; other studies have shown that a significant fraction of memory requests in a broadcast-based shared-memory multiprocessor system are satisfied by cache-to-cache transfers [6, 7]. Barroso et al. observed that cache-to-cache transfers account for 55-62% of the requests in OLTP workloads running on a four-processor Alpha server with 4MB-8MB 2-way set-associative caches [6]. Karlsson et al. observed that "more than half of all second-level cache misses" in a large system running Java-based middleware result in cache-to-cache transfers [7]. Any increase in the count, size, or associativity of caches in the system will tend to increase the rate of cache-to-cache transfers, which in turn drives up the percentage of DRAM reads that go unused.

If the system could determine beforehand that a memory request will obtain data from another processor's cache, it could avoid accessing DRAM for that request. This would eliminate a DRAM read, reduce contention for DRAM resources, and more importantly reduce energy consumption and power dissipation. Performance would be unaffected because requests that do need data from DRAM would still access DRAM upon arrival at the memory controller.

This paper is the first to propose gating of speculative DRAM accesses by utilizing Region Coherence Arrays to detect memory requests that will likely obtain data from other processors' caches.

1.1. Region Coherence Arrays

Region Coherence Arrays (RCAs) are a recently proposed structure used for avoiding broadcast snoops and filtering snoop-induced cache tag lookups in broadcast-based shared-memory multiprocessor systems [8, 9]. RCAs monitor the coherence status of regions, where a region is a large, aligned area of memory that encompasses a power-of-two number of cache lines. RCAs identify regions of memory that are not shared by other processors, and exploit this information to avoid unnecessary broadcast snoops. Memory requests that do not require a broadcast snoop are sent directly to memory, sidestepping the broadcast interconnect. RCAs also identify regions from which the processor is caching lines, and thus can filter unnecessary snoop-induced cache tag lookups. RCAs are inexpensive to implement and have the potential to significantly decrease power consumption in the broadcast interconnect and cache tag arrays. This paper describes straightforward extensions to RCAs that robustly identify memory requests for which a speculative DRAM access would not be useful and would waste power.

1.2. Power-Efficient DRAM Speculation

Power-Efficient DRAM Speculation (PEDS) is a new optimization targeted at broadcast-based shared-memory multiprocessor systems that speculatively access DRAM before the broadcast snoop cycle completes. PEDS takes advantage of information provided by a Region Coherence Array to predict which memory requests are likely to be satisfied from other processors' caches, and reduces DRAM power by forgoing speculative DRAM accesses for those requests.

PEDS adds one bit to memory requests to inform the memory controller whether to fetch the requested line from DRAM speculatively. The memory controller buffers requests tagged as bad candidates for a speculative DRAM access until the snoop response arrives to validate the prediction. If the snoop response indicates that another processor will provide the data, the prediction was correct and the memory controller can drop the request. If the snoop response indicates no processor will provide the data, the prediction was incorrect and the line is fetched from DRAM. In this case, the request incurs a latency penalty.

1.3. Power-Saving Potential

First, PEDS eliminates DRAM reads, directly reducing DRAM activity and DRAM power consumption. Second, by eliminating DRAM reads there is less contention for DRAM resources, reducing queuing delays, bus turnarounds, and power consumption for the remaining DRAM requests. Third, by reducing DRAM activity there is increased opportunity for DRAM power management [10, 11, 12]. DRAM ranks may switch to low-power modes more quickly and remain in low-power modes longer. Though the RCAs consume some power, the elimination of 69% of the broadcast traffic and over 80% of the snoop-induced cache tag lookups more than compensates for it [9]. The additional power consumed by using an existing RCA for PEDS will be negligible.

1.4. Paper Overview

Related work is surveyed in the next section. This is followed by a discussion of the proposed PEDS implementations in Section 3. Sections 4 and 5 describe our methodology and present simulation results for a set of commercial, scientific, and multiprogrammed workloads. Section 6 describes avenues for future work. Finally, Section 7 concludes the paper.

2. Related Work

Coarse-Grain Coherence Tracking (CGCT) techniques have been proposed to reduce broadcast traffic and snoop-induced cache tag lookups in broadcast-based shared-memory multiprocessor systems [8, 9, 13]. Andreas Moshovos proposed the RegionScout Filter, a simple mechanism that uses a non-tagged hash table to track data cached by the processor, and a small, tagged set-associative array for buffering the addresses of non-shared regions currently being used by the processor [14]. Cantin, Lipasti, and Smith proposed Region Coherence Arrays. Region Coherence Arrays are tagged, set-associative arrays that use a region protocol to track the coherence status of regions used by the processor [8]. Zebchuk et al. recently described an elegant framework for maintaining region coherence information with very little overhead [14]. These techniques effectively avoid broadcast snoops with corresponding improvements in scalability and performance [9]. However, these initial investigations were focused only on the potential for avoiding unnecessary broadcast snoops and filtering unnecessary snoop-induced cache tag lookups. They did not exploit CGCT techniques to reduce DRAM power consumption. Rather, by reducing execution time these techniques increase DRAM power consumption.

Shen, Huh, and Sinharoy proposed Cache Residence Prediction (CRP), a set of techniques for predicting whether a read request will obtain data from another processor's cache [15]. One of these techniques uses the invalid cache frame state to predict whether other processors are caching a line. An invalid cache frame indicates that the frame was holding a cache line previously, but the frame was invalidated by another processor's request for the line it contained. If a processor finds such a frame in its cache, there is a high likelihood that another processor is still caching the line that it formerly held, and DRAM should not be accessed speculatively for that line. This technique is simple to implement and highly accurate but there is no quantitative evaluation available. We will quantitatively compare this technique to PEDS.

Dodson et al. proposed Adaptive Memory Access Speculation, a technique for using the early response in shared-memory multiprocessor systems with ring-based interconnects [16]. The early response is the snoop response of processors located between the requestor and the memory controller on the ring. Based on a saturating counter, the memory controller can either fetch data from DRAM speculatively when the request arrives, fetch data speculatively when the early response arrives, or fetch data after the snoop response arrives. In cases of little sharing, read requests speculatively access DRAM upon arrival at the memory controller. In cases of moderate sharing, the memory controller waits for the early response before performing the DRAM access. If the early response indicates that a processor is sharing the data, a DRAM access is not necessary. In cases of abundant sharing, the memory controller does not access DRAM until the snoop response arrives. In contrast, PEDS does not require early responses to detect requests that will result in a cache-to-cache transfer.

Fan, Ellis, and Lebeck investigated memory controller policies for utilizing DRAM power modes in cache-based systems [10]. This research focused on utilizing the low-power modes of modern DRAMs to power down chips when not in use. Analytical modeling was used to study the gap between clusters of memory requests and the threshold time after which the chip should switch to a different mode. Results indicated that the best solution was to power-down DRAM chips as soon as they become idle, and not try to predict how long they would remain idle. PEDS extends this work by reducing DRAM traffic and increasing the effective idle time of DRAM chips.

Delaluz et al. proposed Scheduler-Based DRAM Energy Management, in which the operating system switches DRAM modules to low-power modes [11]. The operating system scheduler keeps track of accesses to DRAM modules made by processes, and attempts to power down modules whenever possible without hurting performance. This technique benefits from the OS scheduler's global view of processes and requires little hardware support. The authors note that this technique can be used in concert with hardware techniques to optimize power consumption further.

Diniz et al. proposed a set of techniques for manipulating DRAM power states to keep DRAM power consumption within a predefined budget at all times, and for reducing power further when possible to do so without degrading performance beyond a predefined threshold [12]. By keeping the power consumption within a predefined budget, cooling and power provisioning costs can be reduced. The authors found that of the several techniques they evaluated, simple ones

such as Knapsack (offline optimization of DRAM power states) and LRU-Ordered (online optimization) performed best, and that these techniques are as effective at reducing energy consumption as performance-aware techniques, since they impose only minor performance degradation.

3. Implementation

PEDS is implemented with 1) an RCA, 2) a policy for deciding which speculative DRAM read requests should be inhibited, 3) a method of communicating this information to the memory controller, and 4) modifications to the memory controller to buffer such requests until the snoop response arrives.

Processor requests probe the RCA in parallel with the lowest-level cache. The region state indicates whether the request must be broadcast to the other processors in the system. If so, the RCA sets a single additional bit in the broadcast request to tag it as either a good or a bad candidate for a speculative DRAM access.

When a memory controller receives a read request tagged as a bad candidate for a speculative DRAM access, it buffers the request in its command queue until the snoop response arrives. If the snoop response indicates that another processor will provide the data, the memory controller drops the request. Otherwise, the DRAM read proceeds and the data is sent to the requesting processor. Reads incorrectly tagged as bad candidates for a speculative DRAM access incur a modest latency penalty.

Before going further, it is important to review the protocol RCAs use to track coherence information. RCAs utilize a protocol with seven states [8]. The first state is "Invalid", meaning that the processor is not caching any lines from the region, and the state of lines from the region in other processors' caches is unknown. The next two states are named "CI" and "DI", meaning that the processor may be caching clean or potentially modified copies of lines from the region (respectively), and that no other processor is caching lines from the region. By potentially modified, we mean that a processor may have modified copies of lines or exclusive copies of lines that may silently become modified. Next, we have "CC" and "DC", which mean that the processor may be caching clean or potentially modified copies of lines from the region (respectively), and other processors may be caching clean copies of lines from the region. These are also called "externally-clean" states. Finally, we have "CD" and "DD", for which other processors may be caching potentially modified copies of lines from the region, and the processor may be caching clean or potentially modified copies, respectively. These are the externally-dirty states.

We study four policy variations for PEDS. The first policy, PEDS-DKD (Delay-Known-Dirty), delays DRAM reads to regions that are in an externally-dirty state (i.e., a region state indicating that other processors may be modifying lines from the region, e. g., CD or DD) [8]. The second policy, PEDS-DLD (Delay-Likely-Dirty), adds an externally-dirty state (ID) to the region protocol to track externally-dirty regions that have been invalidated from the RCA, and delays DRAM reads for lines in regions that are in an externally-dirty state (i.e., ID, CD, and DD). The third policy, PEDS-DNC (Delay-Not-Clean), adds an externally-clean state (IC) to the region protocol to track externally-clean regions (i.e., regions that other processors may be caching read-only lines from, e.g., IC, CC, DC) that have been invalidated from the RCA, and delays DRAM reads for lines in regions that are in an externally-dirty or unknown region state. Finally, the fourth policy, PEDS-DAS (Delay-All-Snoops), forgoes speculative DRAM accesses for all requests that require a broadcast snoop (excluding requests sent directly to memory by the RCA).

Table 1 summarizes which region protocol states are used for each policy to identify requests that should speculatively access DRAM. These policies are described in detail in the pages that follow. A "y" indicates DRAM will be speculatively accessed for processor requests to a region with that state. An "n" indicates DRAM will not be speculatively accessed.

Table 1. Summary of PEDS policies.

	PEDS-DKD	PEDS-DLD	PEDS-DNC	PEDS-DAS
I	n	n	y	y
CI				
CC	n	n	n	y
CD	y	y	y	y
DI				
DC	n	n	n	y
DD	y	y	y	y
IC			n	
ID		y		

3.1. The Delay-Known-Dirty Policy (PEDS-DKD)

Figures 2 and 3 illustrate the potential accuracy of using the existing region protocol states to predict whether read requests are good or bad candidates for a speculative DRAM access. For each application and external region state, the percentage of read requests that obtain data from other processors' caches is shown (Figure 2). The left-hand bars show the percentage of all reads that obtain data from other proces-

sors' caches, 33.4% on average. The next set of bars shows the percentage of all broadcast reads that obtain data from other processors' caches for each application, 58.7% on average. By broadcast reads, we mean read requests that are not sent directly to memory by the RCA.

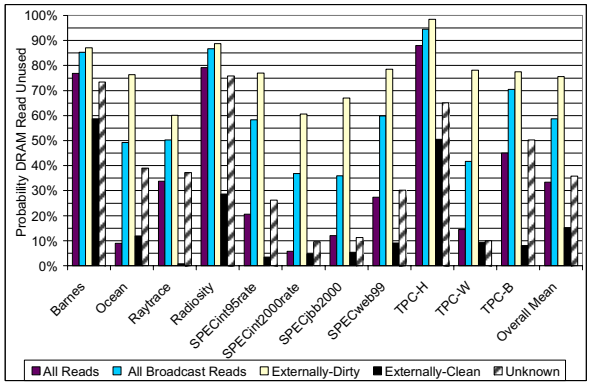

Figure 2: Probability that a read request will obtain data from another processor's cache.

Approximately 75.6% of the read requests that are broadcast while the region is in the RCA in an externally-dirty state (i.e., CD and DD) will obtain data from another processor's cache. These are typically reads to shared data, and these are the chief contributors to wasted DRAM power. On the other hand, only 15.2% of the read requests broadcast while the region is in an externally-clean state obtain data from another processor's cache. The requests to externally-clean regions tend to be instruction fetches, and the coherence protocol of the baseline system does not source clean data. The last set of bars in Figure 2, labeled "Unknown" corresponds to read requests that miss in the RCA. The external region state was unknown at the time the request was broadcast, and approximately 36% of such requests obtain data from other processors' caches.

Figure 3 shows the unused DRAM reads for each application, broken down by external region state. As we can see, reads to externally-dirty regions are the largest contributor to unused reads. The next largest contributor is reads to unknown reads. Reads to externally-clean regions make a very small contribution.

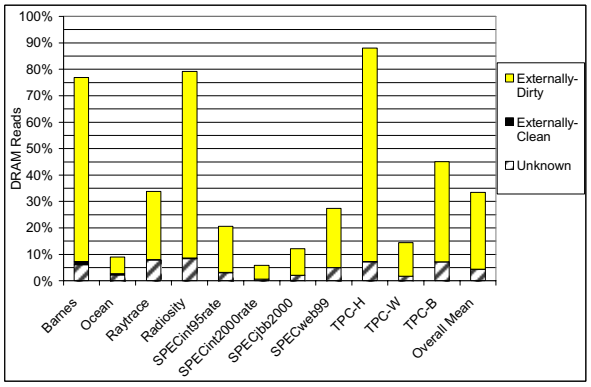

Figure 3: Unused Reads, broken down by region state. See Table 2 for parameters.

From Figures 2 and 3, one can estimate that by avoiding speculative DRAM accesses for reads from externally-dirty regions, DRAM read traffic will be reduced 29%, 36.5%-29% = 7.5% of DRAM reads will go unused, and 29% × ((1/75.6%)-1) = 9.4% of the DRAM reads will be delayed unnecessarily. However, by avoiding speculative DRAM accesses for requests with unknown external region state, DRAM read traffic will be reduced only 4% more while nearly doubling the number of DRAM reads delayed unnecessarily (Note that these are rough estimates; avoiding DRAM accesses will affect memory latency and system behavior). Based on this data, we propose the PEDS-DKD policy, which accesses DRAM speculatively for all read requests except those from regions known to be in an externally-dirty state. PEDS-DKD achieves the highest prediction accuracy using only the existing region protocol states.

3.2. The Delay-Likely-Dirty Policy (PEDS-DLD)

To reduce DRAM traffic further without harming performance, one must improve the prediction accuracy for read requests to regions with unknown external region state. Read requests that miss in a processor's RCA are the next largest contributor to the unused DRAM reads, but only 36% of such reads obtain data from another processor's cache. We observe that the cause of many misses in the RCA is dynamic self-invalidation; an optimization employed by RCAs to maximize their ability to inhibit broadcasts [8]. In response to external requests, an RCA self-invalidates the requested region if the processor is not caching any lines from that region. This increases the probability that the requesting processor will gain exclusive access to the region. Dynamic self-invalidation significantly improves the performance of RCAs, but also

throws away information about the external status of the region (information useful to PEDS).

To preserve the external region state after self-invalidation, we add a pseudo-invalid state to the region protocol: Invalid-Externally-Dirty (ID). This state is exclusively for PEDS, and does not affect the performance or correctness of the RCA. The ID state indicates that the processor is not caching any lines in the region, but other processors may be modifying lines in the region (hence, it is likely dirty). A broadcast snoop must be performed to promote the region to a valid state. The ID state is entered when a region is self-invalidated by an external request for a modifiable copy of a line in the region, or when a region in an externally-dirty state is self-invalidated (See diagram in Figure 4). Because other processors may silently replace regions in the RCA, this state is only a hint and can become stale without affecting correctness.

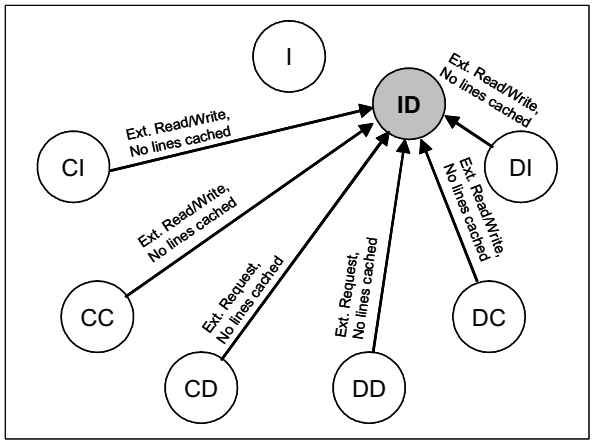

Figure 4: New region protocol state, ID, for PEDS-DLD. The state is entered on the self-invalidations of externally-dirty regions to preserve the external region state information.

Our second policy, PEDS-DLD (Delay-Likely-Dirty), uses this new state to predict whether speculatively accessed DRAM data will be used. If the region is in an externally-dirty state (including ID), DRAM accesses are delayed until the snoop response arrives.

3.3. The Delay-Not-Clean Policy (PEDS-DNC)

To reduce DRAM traffic even further, the system can avoid speculative DRAM accesses for all broadcast read requests except those to externally-clean regions (i.e., CC and DC). However, many reads to unknown regions do not result in a cache-to-cache transfer. Many of these are the result of externally-clean regions that were self-invalidated.

To preserve the external region state after self-invalidation, a second, alternative pseudo-invalid state is added to the region protocol: Invalid-Externally-Clean (IC). This state is exclusively for PEDS, and does not affect the performance or correctness of the RCA. The IC state indicates that the processor is not caching any lines in the region, but other processors may be caching clean copies of lines in the region. A broadcast snoop must be performed to promote the region to a valid state. The IC state is entered when an externally-clean region is self-invalidated by an instruction fetch (See state diagram in Figure 5). This state is also only a hint and can become stale.

Our third policy, PEDS-DNC (Delay-Not-Clean) uses the IC state to predict whether speculatively accessed DRAM data will be used. If the region is in an externally-clean state (i.e., CC, DC, and IC), DRAM reads are performed right away. All other broadcast reads are delayed until the snoop response arrives.

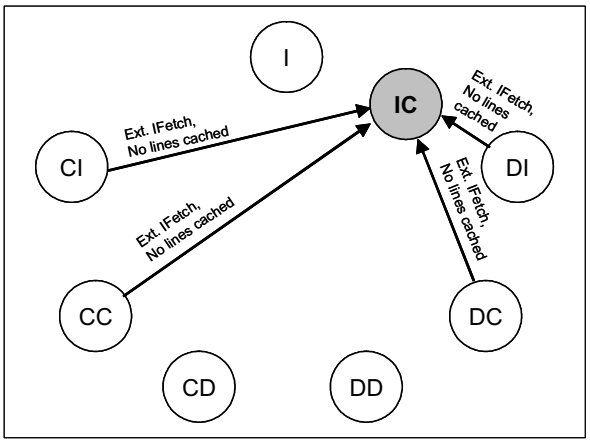

Figure 5: New region protocol state, IC, for PEDS-DNC. The state is entered on the self-invalidations of externally-clean regions to preserve the external region state information.

3.4. The Delay-All-Snoop Policy (PEDS-DAS)

To minimize DRAM traffic, we also propose a strict PEDS policy, which does not speculatively access DRAM for any broadcast reads (including reads to externally-dirty, externally-clean, and unknown regions). Only reads to known non-shared regions, which are sent directly to memory by the RCA, access DRAM immediately; all other reads are buffered until the snoop response arrives. Note that read requests sent directly to memory by an RCA are non-speculative; there is no broadcast snoop and no snoop response is pending. This PEDS policy is called PEDS-DAS (Delay-All-Snoop). PEDS-DAS can limit

DRAM reads to the same extent as a conventional (non-RCA-based) system that never speculatively accesses DRAM (requiring approximately one third fewer DRAM reads than our baseline, according to Figure 2). However, the PEDS-DAS policy will not have the performance degradation of a conventional system that completely avoids speculative accesses to DRAM. Provided the RCA is effectively detecting non-shared data and avoiding broadcast snoops, most read requests will still access DRAM upon arriving at the memory controller and will not incur a latency penalty.

3.5. Hardware Overhead

The additional overhead of implementing PEDS in a broadcast-based shared-memory multiprocessor system that already implements RCAs is quite small. However, even if an RCA is not already present in a design, an RCA with the same number of sets and associativity as the cache increases cache storage by only 6% [8]. If storage is a constraint, a smaller RCA [9] can be used instead. Although any such structure consumes power, the power it saves by avoiding broadcast snoops and filtering snoop-induced cache-tag lookups far outweighs its own power consumption.

No additional storage in the RCA is required for any of the proposed PEDS policies. The PEDS-DKD and PEDS-DAS policies require no additional region protocol states. The PEDS-DLD and PEDS-DNC policies require one additional region protocol state; however, there are only seven states in the original region protocol. No additional bits will be required to encode the region state.

To communicate with the memory controller, a bit needs to be added to the request packets sent to memory. This bit marks read requests as good/bad candidates for a speculative DRAM access. This bit may be encoded with the request type or other information, and constitutes a negligible overhead.

Finally, additional command queue capacity may be needed in the memory controller to buffer requests that do not speculatively access DRAM. Alternatively, a separate queue may be added to buffer these requests.

4. Evaluation Methodology

We performed detailed timing simulations with an execution-driven, shared-memory multiprocessor simulator [17] built on top of SimOS-PPC [18]. The simulator implements the PowerPC ISA and runs both user-level and system code. We modeled a four-processor system with a Fireplane-like interconnect [3] and 1.5GHz processors with resources similar to the UltraSPARC-IV [19]. Unlike the UltraSPARC-IV, the simulated processors feature out-of-order issue and each processor has a 1MB L2 cache. Each processor also has an RCA with the same organization as the L2-cache tags (8K sets, 2-way associative), and 512B regions (shown to be effective in [8, 9]). A more complete list of parameters is shown in Table 2.

Table 2. Simulation Parameters.

Processor	
Processor Clock	1.5GHz
Processor Pipeline	15 stages
Fetch Queue Size	16 instructions
BTB	4K sets, 4-way
Branch Predictor	16K-entry Gshare
Return Address Stack	8 entries
Decode/Issue/Commit Width	4/4/4
Issue Window Size	32 entries
ROB	64 entries
Load/Store Queue Size	32 entries
Int-ALU/Int-MULT	2/1
FP-ALU/FP-MULT	1/1
Memory Ports	1
Caches	
L1 I-Cache	32KB 4-way, 64B lines, 1-cycle
L1 D-Cache	64KB 4-way, 64B lines, 1-cycle
L2 Cache	1MB 2-way, 64B lines, 12-cycle
Prefetching	Power4-style, 8 stream, 5 line runahead R10000-style exclusive-prefetching
Cache Coherence Protocols	Write-Invalidate MOESI (L2), MSI (L1)
Memory Consistency Model	Sequential Consistency
Interconnect	
System Clock	150Mhz
Snoop Latency	106ns (16 cycles)
DRAM Latency	106ns (16 cycles)
DRAM Latency (Overlapped)	47ns (7 cycles)
Transfer Latency (Same Switch)	20ns (3 cycles)
Transfer Latency (Same Board)	47ns (7 cycles)
Transfer Latency (Remote)	80ns (12 cycles)
Memory	
Memory	8GB, Micron DDR200
Memory Channels	2
DMA Buffer Size	512B
Coarse-Grain Coherence Tracking	
Region Coherence Array	8K sets, 2-way set-assoc. 512B regions

To compute DRAM power dissipation and energy consumption, the execution-driven simulator worked in concert with the DRAMsim simulator from Maryland [20]. DRAMsim has detailed DRAM timing models and computes DRAM power consumption by tracking activities and power-down modes at the rank level. Our simulated system has 8GB of DDR-200 DRAMs running at 100MHz. The timing and electrical characteristics were taken from Micron's datasheet for 1Gb DDR-200 SDRAM DIMMs [21]. We use the high-performance SDRAM-close-page-map address mapping policy [20] and closed page row-buffer management policy that has been shown to work best with shared-memory multiprocessor systems [22]. We model the clock enable and disable feature for power

management in DDR systems where the clock can be disabled each cycle the rank is not servicing a command.

Table 3. Workloads Simulated.

Category	Benchmark	Comments
Scientific	Ocean	SPLASH-2 Ocean Simulation, 514 x 514 Grid
	Raytrace	SPLASH-2 Raytracing application, Car
	Barnes	SPLASH-2 Barnes-Hut N-body Simulation, 8K Particles
	Radiosity	SPLASH-2 Light Interaction Application (-room -ae 5000.0 -en 0.050 -bf 0.10)
Multiprogramming	SPECint2000Rate	Standard Performance Evaluation Corporation's 2000 CPU Integer Benchmarks, Combination of reduced-input runs
	SPECint95Rate	Standard Performance Evaluation Corporation's 1995 CPU Integer Benchmarks
Commercial	SPECweb99	Standard Performance Evaulation Corporation's World Wide Web Server, Zeus Web Server 3.3.7, 300 HTTP Requests
	SPECjbb2000	Standard Performance Evaulation Corporation's Java Business Benchmark, IBM jdk 1.1.8 w/ JIT, 20 Warehouses, 2400 Requests
	TPC-W	Transaction Processing Council's Web e-Commerce Benchmark, DB Tier, Browsing Mix, 25 Web Transactions
	TPC-B	Transaction Processing Council's Original OLTP Benchmark, IBM DB2 version 6.1, 20 clients, 1000 transactions
	TPC-H	Transaction Processing Council's Decision Support Benchmark, IBM DB2 version 6.1, Query 12 on a 512MB Database

For workloads, we use a combination of commercial, scientific, and multiprogrammed benchmarks (Table 3). Execution-driven simulations were started from checkpoints taken from a functional simulator used to boot AIX and warm up the workloads. Cache checkpoints were included to warm the caches prior to simulation. Due to workload variability, several runs of each benchmark were performed with small random delays added to memory requests to perturb the system, and the results were averaged [23]. The 95% confidence intervals for each benchmark are shown where appropriate. Overall means are shown for each graph. The overall means were computed by taking the arithmetic means of the results from the scientific, multiprogrammed, and commercial workloads separately, and then combining the resultant means to form an overall arithmetic mean that weights each category equally.

We simulated seven system configurations for comparison (Table 4). To compare conventional systems with and without speculative DRAM accesses, two baseline configurations were simulated, one that speculatively accesses DRAM for all read requests ("Base") and one that does no speculative DRAM accesses ("Base-NoSpec"). Next, we simulated a system with Cache Residence Prediction, which does not speculatively access DRAM for read requests to lines formerly held in invalid cache frames ("Shen-CRP") [15]. Four implementations of PEDS were simulated for comparison: PEDS-DKD, PEDS-DLD, and PEDS-DNC, PEDS-DAS.

Table 4. Configurations Simulated.

Abbreviation	Description
Base	Baseline, All Reads Access DRAM Speculatively
Base-NoSpec	Baseline, No Reads Access DRAM Speculatively
Shen-CRP	Baseline, No Invalid-Line Reads Access DRAM Speculatively
PEDS-DKD	RCA, No Ext-Dirty-Region Reads Access DRAM Speculatively
PEDS-DLD	RCA w/ ID, No Ext-Dirty-Region Reads Access DRAM Speculatively
PEDS-DNC	RCA w/ IC, Only Ext-Clean-Region Reads Access DRAM Speculatively
PEDS-DAS	RCA, No Broadcast Reads Access DRAM Speculatively

5. Results

5.1. Reduction in DRAM Reads Performed

Figure 6 shows the percentage of the DRAM reads that are performed by each of the seven different system configurations, normalized to a baseline with speculative DRAM accesses. The shaded portion of each bar is the percentage of DRAM read requests performed after the snoop response arrived. The lower the total height of the bar, the fewer DRAM reads performed; the smaller the shaded portion, the fewer reads that incur a latency penalty.

For the baseline with speculative DRAM accesses, all DRAM reads were performed, and no reads were delayed until the snoop response arrived. In contrast, the baseline system without speculative DRAM accesses performed the minimum number of DRAM reads (67%), delaying all read requests until the snoop response was available. The Shen-CRP system performs 85% of the DRAM reads with only 4% of the read requests being delayed unnecessarily. For its simplicity and low cost, Shen-CRP provides a respectable reduction in DRAM reads with a minimal performance impact. However, it does not exploit all of the available potential. Utilizing information from an RCA, PEDS achieves reductions in DRAM reads similar to that of a system that does not speculatively access DRAM, only performing 68-72% of the DRAM reads), with 6-15% of the reads delayed unnecessarily.

5.2. Opportunity for DRAM Power Management

Figure 7 illustrates the increased opportunity for DRAM power management. Along the Y-axis is the number of processor cycles between DRAM operations to a rank (read or write) on a logarithmic scale. DRAM ranks do not need to be powered-up for read

requests that obtain data from other processor's caches, allowing DRAM ranks to switch to low-power modes sooner and to remain in low-power modes longer. Compared to the baseline with speculative DRAM accesses, PEDS more than doubles the average time between DRAM operations to a rank. Note that the RCA reduces execution time, decreasing the time between DRAM operations.

The benchmarks that benefit most are Barnes, Radiosity, and TPC-H. For these benchmarks, a large percentage of requests result in cache-to-cache transfers, and removing these increases the time between requests 3-6 times.

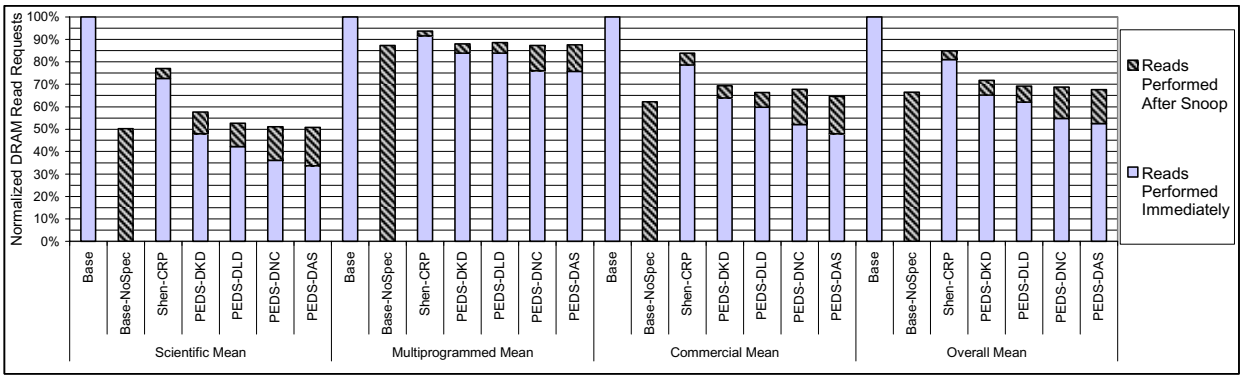

Figure 6: Average DRAM reads performed, normalized to that of the baseline, broken down into DRAM reads performed immediately and DRAM reads delayed until the snoop response arrives.

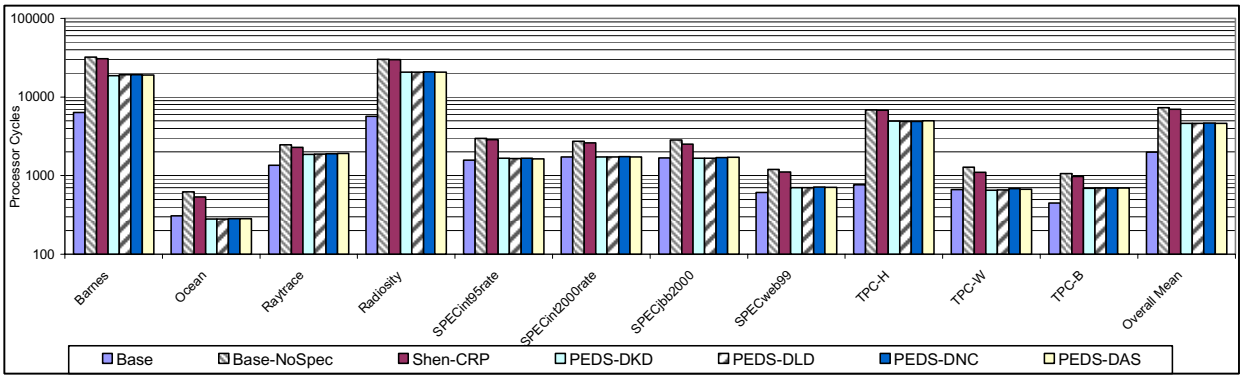

Figure 7: Average processor cycles between DRAM requests to a rank.

5.3. DRAM Power and Energy Savings

Figure 8 shows the average DRAM power consumption for the seven system configurations, normalized with respect to the baseline with speculative DRAM accesses. Not speculatively accessing DRAM reduces average DRAM power consumption 31% for the baseline. The Shen-CRP configuration achieves a 15% reduction in DRAM power consumption over the baseline. PEDS-DKD, PEDS-DLD, PEDS-DNC, and PEDS-DAS reduce average DRAM power consumption 17%, 20%, 21%, and 22%, respectively. PEDS can achieve 69% of the reduction in DRAM power consumption of a baseline system without speculative DRAM accesses.

Figure 9 shows the average DRAM energy consumption for the seven system configurations, normalized with respect to the baseline with speculative DRAM accesses. The average DRAM energy consumption is the product of the average DRAM power consumption and the execution time. Optimizations that reduce DRAM power consumption do not necessarily reduce the DRAM energy consumed in running an application (not to mention the system energy consumption). PEDS reduces average DRAM energy usage 16-21% --close to that of a baseline system without speculative DRAM accesses. Note that although the Base-NoSpec configuration in Figure 8 had nearly a 10% lower DRAM power consumption than PEDS-DAS, the difference in energy consumption is

negligible due to the increased execution time of Base-NoSpec. Interestingly, PEDS-DAS still provides reductions in energy compared to the less aggressive implementations despite delaying more DRAM reads unnecessarily. The RCA has identified most of the requests to non-shared data and not speculatively accessing DRAM for the remaining reads does not affect performance enough to offset the energy savings.

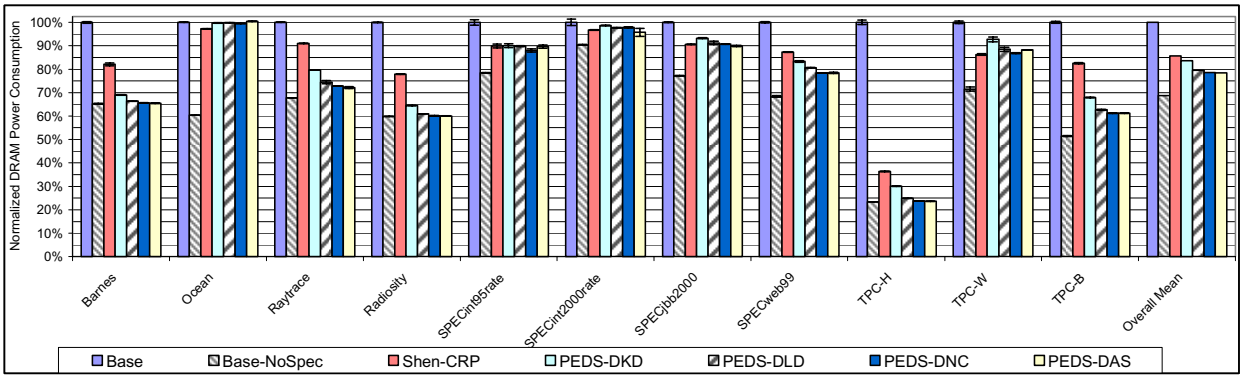

Figure 8: Normalized DRAM power consumption.

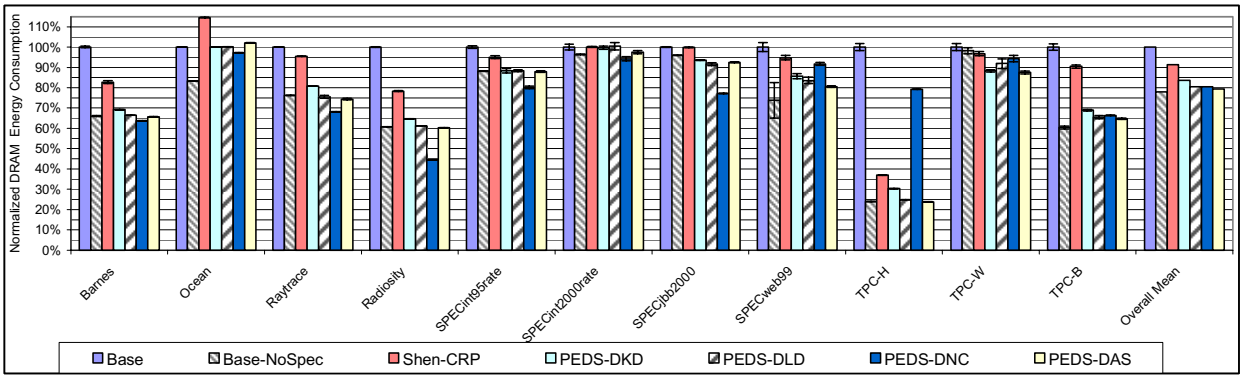

Figure 9: Normalized DRAM energy consumption.

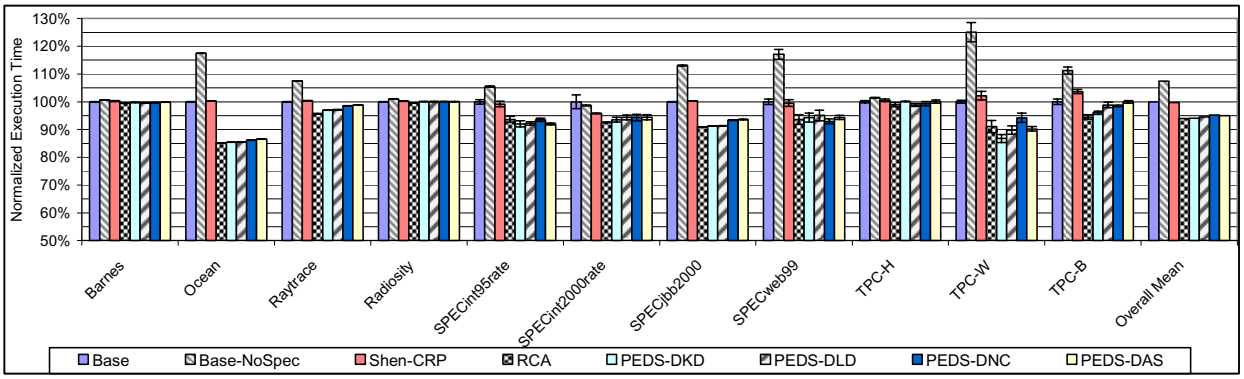

Figure 10: Normalized execution time.

5.4. Effect on Execution Time

Figure 10 shows the execution time of the seven system configurations, normalized with respect to the baseline with speculative DRAM accesses. In addition, Figure 10 contains bars for an RCA where all memory reads speculatively access DRAM. These bars are labeled "RCA", and are located to the left of the bars for the PEDS implementations to show how PEDS degrades performance over a system with an RCA alone. RCAs improve performance; hence, the performance of a system with PEDS cannot be directly compared to systems that implement other techniques such as Shen-CRP.

For each application, the execution time of PEDS is shorter due to the reduction in broadcast snoops achieved by the RCA. Throttling speculative DRAM accesses with PEDS degrades average execution time less than 1%, not enough to cancel out the execution time improvement of the RCA. The Shen-CRP configuration also does not affect performance noticeably. In contrast, not speculatively accessing DRAM in the baseline system increases average execution time 7%. Not speculatively accessing DRAM degrades performance significantly; however, utilizing an RCA PEDS can throttle DRAM accesses without the performance impact.

6. Future Work

In future work, there is potential to increase the effectiveness of PEDS with more precise information about data cached by other processors in the system. For example, state for multiple regions can be collected at once with each broadcast snoop, exploiting more spatial locality and minimizing misses in the RCA. Combined with a smaller region size, more precise tracking of coherence status can be achieved, resulting in better detection of reads that will go unused.

Another possibility is to add more bits to memory requests to give the memory controller more freedom to prioritize requests. For example, while requests for externally-dirty regions are very likely to obtain data from other processor's caches, requests to regions for which the external region state is unknown have a much lower probability. Communicating this information to the memory controller can enable it to prioritize requests accordingly. Read requests to non-shared regions are non-speculative and have the highest priority; requests to regions for which the state is unknown can have a lower priority, and requests to regions in an externally-dirty state can have the lowest priority.

Yet another avenue of future work is in combining PEDS with DRAM scheduling techniques to improve performance and bandwidth utilization [24, 25]. PEDS can improve memory access scheduling by removing unnecessary DRAM operations and the resource conflicts they cause. In addition, PEDS can enable memory schedulers to not only choose between memory operations based on hardware hazards and the program mix, but also based on whether operations are likely to fetch data that will be used.

Last, but not least, it would be interesting to see how well PEDS techniques apply to systems with other Coarse-Grain Coherence Tracking techniques, such as RegionScout Filters [13] or RegionTrackers [14]. Though this paper investigated only the use of PEDS in systems with Region Coherence Arrays, the same concepts may apply equally well to systems with such structures.

7. Conclusions

PEDS significantly reduces DRAM activity, achieves large reductions in DRAM power dissipation and energy consumption, and creates more opportunities for DRAM power management. Utilizing an RCA, PEDS achieves these benefits with negligible effect on performance. The hardware overhead of implementing PEDS in a multiprocessor system that incorporates RCAs is minimal.

The four PEDS policies investigated in this paper vary in aggressiveness, providing four alternatives that each make a different tradeoff between effectiveness and performance impact. PEDS-DKD is the least aggressive, and has the least performance impact. In order of increasing aggressiveness and performance impact, we have PEDS-DLD, PEDS-DNC, and PEDS-DAS.

We also find Cache Residence Prediction [15] to be effective with a minimal affect on performance despite its low cost. However, Cache Residence Prediction does not reduce DRAM read traffic and power consumption as much as PEDS.

Acknowledgements

We thank our many anonymous reviewers for their comments and suggestions. We also thank Prasun Agarwal, Candy Cantin, and Pattabi Seshadri for comments and help proofreading. This research was supported in part by the National Science Foundation under grants CCR-0133437, CCF-0429854, and CCF-0702272, as well as grants and equipment donations from IBM and Intel.

References

[1] Barroso, L., *The Price of Performance*, ACM Queue, Volume 3, Number 7, September 2005.

[2] Laudon, J., UltraSPARC *T1: Architecture and Physical Design of a 32-threaded General Purpose CPU*, Proceedings of the ISSCC Multi-Core Architectures, Designs, and Implementation Challenges Forum, 2006.

[3] Charlesworth, A., *The Sun Fireplane System Interconnect*, Proceedings of SC2001.

[4] Kalla, R., Sinharoy, B., and Tendler, J., *IBM Power5 Chip: A Dual-Core Multithreaded Processor,* IEEEMicro, 2004.

[5] Weber, F., *Opteron and AMD64, A Commodity 64 bit x86 SOC*, Presentation, Advanced Micro Devices, 2003.

[6] Barroso, L., Gharachorloo, K., and Bugnion, E., *Memory System Characterization of Commercial Workloads*. Proceedings of the 25th International Symposium on Computer Architecture, 1998.

[7] Karlsson, M., Moore, K., Hagersten, E., and Wood, D., *Memory System Behavior of Java-Based Middleware.* Proceedings of the 9th International Symposium on High-Performance Computer Architecture, 2003.

[8] Cantin, J., Lipasti, M., and Smith J., *Improving Multiprocessor Performance with Coarse-Grain Coherence Tracking*, Proceedings of the 32nd International Symposium on Computer Architecture, 2005.

[9] Cantin, J., Moshovos, A., Lipasti, M., Smith, J., and Falsafi, B., *Coarse-Grain Coherence Tracking: RegionScout and Region Coherence Arrays*, IEEE Micro Special Issue on Top Picks from 2005 Computer Architecture Conferences, 2006.

[10] Fan, X., Ellis, C., and Lebeck, A., *Memory Controller Policies for DRAM Power Management*, International Symposium on Low-Power Electronics Design, 2001.

[11] Delaluz, V., Sivasubramaniam, A., Kendemir, M., Vijaykrishnan N., and Irwin, M., *Scheduler-Based DRAM Energy Management*, Design Automation Conference, 2002.

[12] Diniz, B., Guedes, D., Meira, W., Bianchini, R., Limiting the Power Consumption of Main Memory. Proceedings of the 34th International Symposium on Computer Architecture, 2007.

[13] Moshovos, A., *RegionScout: Exploiting Coarse Grain Sharing in Snoop-Based Coherence*, Proc. of the 32nd International Symposium on Computer Architecture, 2005.

[14] Zebchuk, J., Safi, E., and Moshovos, A. *A Framework for Coarse-Grain Optimizations in the On-Chip Memory Hierarchy*. Proceedings of the 40th IEEE/ACM International Symposium on Microarchitecture, 2007.

[15] Shen, X., Huh, J., and Sinharoy, B., *Cache Residence Prediction*. United States Patent #7,266,642, IBM, September 2007.

[16] Dodson, S., Fields, S., Ghai, S., and Stuecheli, J., *Adaptive memory access speculation*. U.S. Patent #7058767, IBM, June 2006.

[17] Cain, H., Lepak, K., Schwartz, B., and Lipasti, M., *Precise and Accurate Processor Simulation*. Proceedings of the 5th Workshop on Computer Architecture Evaluation Using Commercial Workloads, 2002.

[18] Keller, T., Maynard, A., Simpson, R., and Bohrer, P., *Simos-ppc Full System Simulator.* http://www.cs.utexas.edu/users/cart/simOS.

[19] *UltraSPARC IV Processor, User's Manual Supplement*, Sun Microsystems Inc., 2004.

[20] Wang, D., Ganesh, B., Tuaycharoen, N., Baynes, K., Jaleel, A., and Jacob, B., *DRAMsim: A memory-system simulator.* SIGARCH Computer Architecture News, Volume 33, Number 4, September 2005.

[21] DDR-200 datasheet. http://www.micron.com/products/partdetail?part=MT46V128M8P-6T. Micron 2003.

[22] Natarajan, C., Christenson, B., and Briggs, F., *A study of performance impact of memory controller features in multi-processor server environment.* Proceedings of the 3rd Workshop on Memory Performance Issues, 2004.

[23] Alameldeen, A., Martin, M., Mauer, C., Moore, K., Xu, M., Hill, M., and Wood, D. *Simulating a $2M Commercial Server on a $2K PC.* IEEE Computer, 2003.

[24] Rixner, S., Dally, W., Kapasi, U., Mattson, P., and Owens, J., *Memory Access Scheduling*. Proceedings of the 27th International Symposium on Computer Architecture, 2000.

[25] Hur, I., and Lin, C., *Adaptive History-Based Memory Schedulers*, Proceedings of the 37th IEEE/ACM International Symposium on Microarchitecture, 2004.