Speculative Supplier Identification for Reducing Power of Interconnects in Snoopy Cache Coherence Protocols

Ehsan Atoofian
eatoofia@ece.uvic.ca

Amirali Baniasadi
amirali@ece.uvic.ca

Kaveh Aasaraai
aasaraai@ece.uvic.ca

Electrical and Computer Engineering Department
University of Victoria
3800 Finnerty Rd.
Victoria, BC, Canada

ABSTRACT
In this work we reduce interconnect power dissipation in Symmetric Multiprocessors or SMPs. We revisit snoopy cache coherence protocols and reduce unnecessary interconnect activity by speculating nodes expected to provide a missing data.

Conventional snoopy cache coherence protocols broadcast requests to all nodes, reducing the latency of cache to cache transfer misses at the expense of increasing interconnect power. We show that it is possible to reduce the associated power dissipation if such requests are broadcasted selectively and only to nodes more likely to provide the missing data.

We reduce power as we limit access only to the interconnect components between the requester and the supplier node. We evaluate our technique using shared memory applications and show that it is possible to reduce interconnect power by 21% in a 4-way multiprocessor without compromising performance. This comes with negligible hardware overhead.

Categories and Subject Descriptors
C.1.2 [**Multiple Data Stream Architectures**]: Multiprocessors

General Terms
Design, Performance.

Keywords
SMP, Cache Coherence Protocol, Power, Interconnect

1. INTRODUCTION
With ever increasing demand for higher performance, symmetric multiprocessors (or SMPs) offer an attractive solution as they achieve higher performance by exploiting thread-level parallelism.

CF'04, May 7–9, 2007, Ischia, Italy.

Multithread workloads which run simultaneously on multiprocessor nodes communicate through interconnect and dissipate significant power. As a result, inter-processor communication has become one of the bottlenecks in multiprocessor systems consuming a considerable share of the overall power [1]. Moreover, to improve performance of SMPs, it is expected that the number of processors in SMPs will increase. This in turn would result in higher interconnect complexity and power dissipation in future SMPs.

In a shared memory multiprocessor system, processors access interconnects to locate and provide data and instructions missing in local caches [3, 6, 8, 15-17, 20, 21]. Maintaining the correct state of the local data and responding to requests made by other processors is the responsibility of local caches. In the event of a cache miss and in a write-invalidate protocol, snoop requests, invalidate messages, and block writebacks have to take place. Such transactions contribute to the overall interconnect power dissipation.

Previous studies [12, 23] show that processors providing missing data show very high locality. In other words, if the required data missing in processor A is provided by processor B's local cache, chances are that next time A is missing a data, it will be provided by B again.

We introduce speculative supplier identification (or simply SSID) to reduce power of interconnects by sending snoop requests only to nodes more likely to provide the missing data. As we will explain in Section 4, we use processors' ID for speculation which results in much simpler hardware than that proposed in [12, 16, 23].

In SSID, the requesting node sends the request only to the previous supplier if there is high confidence that the previous supplier would provide the missing data. Therefore, we avoid broadcasting requests to every node. Consequently we reduce interconnect energy and only access links and switches connecting the requester and supplier. As a result we eliminate unnecessary activities not only in interconnects and internal switches but also in processor tag arrays.

In summary, we make the following contributions:

• We show supplier nodes tend to repeat their behavior. We show that for the configuration and applications used in this study, two consecutive cache misses are handled by the same remote node 84% of the time.

• We show that it is possible to identify the supplier processor for a local cache miss with high accuracy by using a small and simple predictor. We identify remote suppliers with an average accuracy up to 96% by exploiting a single entry 4-bit predictor.

• By limiting sending requests only to the predicted remote node, and while maintaining performance, we reduce link, switch and tag array energy by 21%, 22% and 17%% in a 4-way multiprocessor system respectively.

In this work, and as a case study, we use a binary tree interconnect network, similar to Sun Fireplane [2]. However, our method can also be used for other alternative non-bus interconnects (e.g., benes and fat-tree [22]).

The rest of the paper is organized as follows. In Section 2, we discuss our motivation. In Section 3, we review the related background. In Section 4, we discuss SSID and explain implementation details. In Section 5, we discuss methodology and evaluate SSID. In Section 6, we review related work. Finally, in Section 7, we offer concluding remarks.

2. MOTIVATION

In Figure 1, we show how an SMP system handles a snoop request. As presented, N processors sharing a single memory are connected through a network interconnect. Each processor has local L_1 and L_2. Assume that processor P_0 is about to read the elements of a shared array for the first time. Meantime P_{n-1} has already read the array, and all array elements are available in P_{n-1}'s local cache. A miss occurs as soon as P_0 reads the first array element. To find the missing data, P_0 broadcasts snoop request to all nodes (Figure 1.a). P_1, P_2, …, and P_{n-1} receive snoop requests and check their tag arrays. P_{n-1} finds the element and sends it to P_0. The system goes through the same procedure every time P_0 reads a new (missing) element.

This approach provides fast access but is inefficient from the energy point of view [6] as not all accesses made to interconnect components turn out to be useful. All nodes are snooped every time that an array element is accessed in P_0 but only one (P_{n-1} in the example) provides the data.

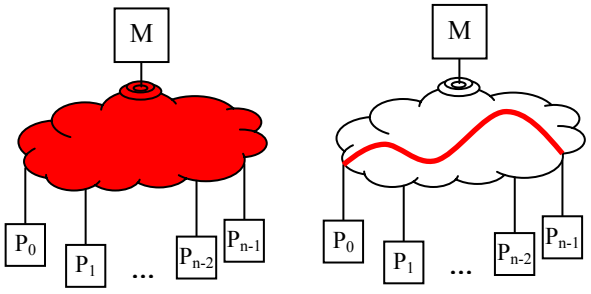


Figure 1. a) Conventional snooping. b) SSID snooping

In this work, we report how SSID uses supplier locality to address this design inefficiency and to reduce power in snoop-based symmetric multiprocessor systems. Supplier locality shows how often the current supplier of a missing data in a local node is the same as the previous supplier. In Figure 2 we report supplier locality for the SPLASH-2 benchmarks used in this study (see Section 5.1 for methodology). Except for *fft*, all benchmarks have a locality higher than 70%. On average, supplier locality is 84%.

SSID relies on this behavior to speculate the remote node providing the missing data. SSID sends the associated request only to the speculated node limiting the accesses to the path between the requester and supplier (Figure 1.b). This reduces power compared to a conventional snoop-based system where all nodes are accessed uniformly and regularly.

If SSID fails to accurately identify the supplier then snoop requests have to be broadcasted to every node, resulting in energy and latency penalty. This penalty can negate our savings if mispredictions occur too often. However, as we show later in this work, savings outweigh the associated overhead.

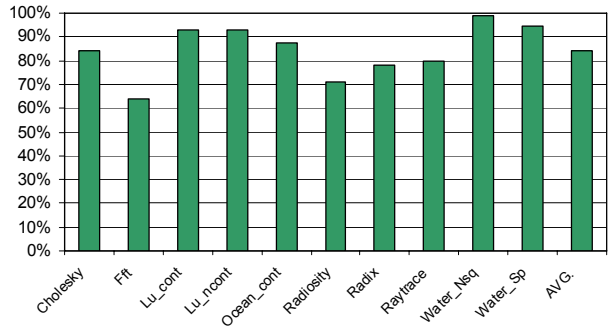


Figure 2. Supplier locality for the applications and the configuration used in this study.

3. BACKGROUND

In Section 3.1, we review a basic write-invalidate snoop protocol. We discuss the interconnect architecture used in this work in Section 3.2.

3.1 Write-Invalidate Snoop Protocol

The snoop protocol is used to maintain cache coherence in shared memory multiprocessors. A cache coherence protocol is a set of finite state machines that change their states in response to their local processors' requests and messages received on the bus. Each finite state machine is distributed over nodes with each local cache maintaining the state of its local data. Caches connected to the bus monitor bus transactions and update the state of their data and reply to requests when required.

On a cache read miss, the missing processor sends a request to everyone. All nodes check their tag arrays with the missing address and send the (valid) data to the requester if found.

In a write-invalidate protocol, upon a write miss, an invalidate request is sent over the bus. All processors having a copy of the address invalidate the corresponding entry in their caches.

3.2 Tree-base Interconnect Structure

The address interconnect used in this paper is similar to Sun Fireplane interconnect [2]. Figure 3 shows address interconnect structure. The structure is implemented using two level switches. Processors reside at the leaves of the tree, and the memory is connected at the root. At any moment, at most one message exists in the tree. From the processor viewpoint, the tree structure is similar to a bus [6].

A missing processor sends the request to the root switch. In the next step, the root switch sends request copies to other processors. Processors search their tag arrays and reply. If any of the processors has the data, the root switch selects the closest processor to the requester and forwards the processor's message. If none of the processors have the data, the root switch sends a request to the memory.

In SSID, we modify the baseline cache coherence protocol and reduce the number of steps involved. Instead of sending requests to the root and then having the request broadcasted by the root, the request is directly sent to other nodes. This reduces the number of accesses to the links by one and results in processors receiving snoops requests at different cycles. For example in Figure 3, under our system, a request sent by P_0 is received by P_1 earlier than P3.

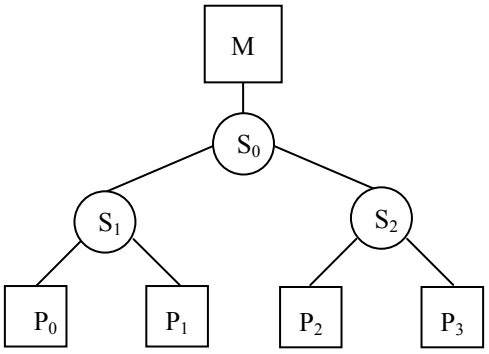


Figure 3. Address interconnect structure.

In our system, and similar to the conventional snoop-based system, processors reply to the root switch after tag lookup is performed. However, in our system, the root switch does not receive replies from processors at the same time. The root switch should wait to receive all replies and then select the closest supplier to the requester or send the request to the memory.

In this work we assume separate data and address interconnects. Data interconnect is similar to address interconnect and uses two levels of switches. When the supplier is determined by the cache coherence protocol, the supplier sends data to the requester through the data interconnect. In this work we focus on the address interconnect.

4. IMPLEMENTATION

In our proposed architecture, each node is equipped with a small single-entry predictor to speculate the supplier for missing data reads in the local cache. Using prediction for write commands has consistency model implications that are beyond the scope of this paper [5].

Figure 4 depicts a typical processor in our SMP configuration. Each processor includes a core, a private L_1 cache, a private L_2 cache, and a supplier predictor. Each predictor entry is equipped with two fields. The first field is a $\log_2 N$ bit field, where N is the number of processors, and is referred to as speculated supplier or SPL. SPL is used to record the last supplier. The second field is an n-bit saturating counter. We use saturating counters to achieve high accuracy. We increment the counter if the prediction is correct and reset it upon a misprediction. The predictor is trusted only if the saturating counter is more than a pre-decided threshold. The area and energy overhead associated with the predictor is negligible as the predictor only includes an n-bit counter and a $\log_2 N$ bit register. We refer to an SSID system using an n-bit counter as SSID-n (e.g., SSID-2 uses a single 2-bit counter).

Initially, there is no record of any previous supplier in the predictor. Therefore, no prediction is made when the first miss occurs. Under such circumstances, the processor broadcasts a snoop request to all nodes, basically following the conventional approach. When the supplier processor responds, the predictor is updated with the supplier number. For future cache misses, if the saturating counter exceeds the threshold, the request is only sent to the predicted node. The predicted node checks for the requested address, and replies. For accurate predictions and if the valid data is found in the speculated supplier, no additional step is needed. Consequently, instead of accessing all switches, links, and tag arrays, only the required components are accessed reducing power in both interconnect and tag arrays.

In the event of a supplier misprediction, the requester has to send a snoop request to all the other nodes. This results in extra accesses to interconnect and an increase in data communication latency. We show in Section 5, that the benefits of correct predictions outweigh the associated misprediction costs.

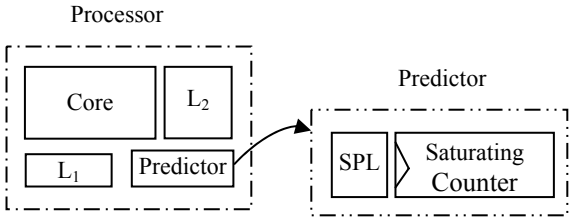


Figure 4. A processor using a node predictor: each predictor includes a speculative supplier (SPL) and a saturating counter.

SSID does not impose any changes to the state transitions of the underlying cache coherence protocol. In a MESI protocol [22], the requested cache block in the speculated supplier node could be in one of the following four states: modified, exclusive, shared, or invalid. If the state is modified, exclusive, or shared, and speculation turns out to be accurate, then both the supplier and requester will end up having the shared state. However, if the state of the requested cache block is invalid, a misprediction will occur and the requester will broadcast a snoop request. Consequently, whether the prediction is right or wrong, SSID

would not change any state transition in the MESI protocol. Also, SSID does not impose any limitation on software, and is completely transparent to the operating system.

To provide better understanding, in Figure 5, we show the actions taken under SSID for the example discussed earlier in Section 2. We assume SSID-1 with a prediction threshold equal to zero.

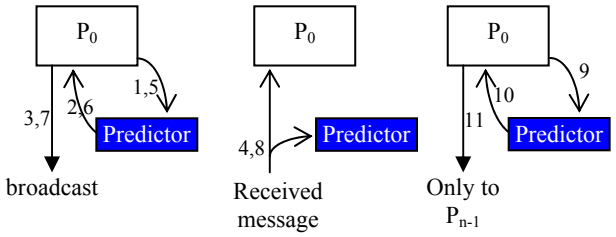


Figure 5. An SSID example: P_0 and P_{n-1} share an array. P_{n-1} has already read the elements and has them in its L_2 cache. P_0 starts reading the (missing) array elements. a.1) P_0 asks the predictor for the likely supplier. a.2) The predictor cannot make a prediction as there is no previous record. a.3) P_0 broadcasts snoop request to all nodes. b.4) P_{n-1} sends the data to P_0. Predictor is updated with the supplier processor number, and data is stored in P_0's local cache. a.5, a.6) Upon missing the array's second element, the predictor speculates P_{n-1} as the likely supplier. The predictor is not trusted since the saturating counter is not greater than the threshold. a.7) P_0 broadcasts snoop request to all nodes. b.8) The saturating counter is incremented as the predictor has made a correct prediction. c.9) For the third array element, P_0 probes the predictor. c.10) Predictor speculates that P_{n-1} is supplier. c.11) P_0 sends the request only to P_{n-1} (instead of broadcasting). P_{n-1} provides the array element.

5. EVALUATION

In this Section, we evaluate SSID. In Section 5.1 we present the methodology. In Section 5.2 we report the results.

5.1 Methodology

We used SPLASH-2 [4] benchmarks (details reported in Table 1) to evaluate our scheme. For simulation, we used the execution driven mode of SESC [7] modeling the out of order processors and the memory subsystem presented in Table 2. We used MESI protocol to maintain cache coherence in L_2 caches.

We used Orion [18] to estimate power of interconnects. We used CACTI [19] to measure tag arrays power dissipation.

5.2 Results

In Section 5.2.1, we report SSID coverage and accuracy. In Sections 5.2.2 and 5.2.3, we report how SSID impacts performance and energy of interconnects. We compare SSID to conventional cache coherence where snoop requests are broadcasted to all nodes upon any local cache miss.

Table 1. Splash2 benchmarks and input parameters

Benchmarks	Input Parameters
Cholesky	tk29.O
Fft	1M complex data points
Lu(contiguous, non-contiguous)	512×512 matrix, B=16
Ocean contiguous	258×258 grid
Radiosity	-batch –room
Radix	8M keys
Raytrace	Balls4.env
Water(nsquared, spatial)	4k molecules

Table 2. System parameters

Processor	Interconnect	Memory System
branch predictor:16K entry bimodal and gshare branch penalty: 17 Fetch/issue/commit: 6/4/4 RAS: 32 entries BTB: 2K entries, 2-way	bus clock cycle: 7 processor cycles switch latency: 1 cycle link latency: 1 cycle interconnect width: 64B	cache block size: 64B split I-L_1, D-L_1: 32KB, 4-way L_1 latency: 2 L_2: 512KB/8-way L_2 latency: 11 memory latency: 70 processor cycles

To make better evaluation of SSID possible, in Section 5.2.4., we compare SSID to serial snooping [6]. In serial snooping, a snoop request is initially sent only to the neighbor node. The neighbor node looks up its local cache and replies to the requester if the requested data is found, otherwise, it sends the snoop request to the next node. In both SSID and serial snooping, at any moment, at most one message exists in interconnect. As such, memory consistency is maintained accurately [24]. To the best of our knowledge, serial snooping is the only power aware snoop-based cache coherence in binary tree interconnects.

To provide better insight, in Section 5.2.4, we report how SSID and serial snooping affect performance and energy of interconnects in SMP systems with 4, 8, 16, and 32 nodes.

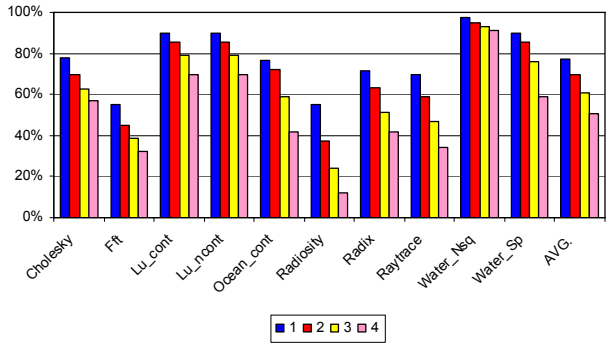
5.2.1 Coverage and Accuracy

In this Section, we report coverage and accuracy for SSID in a 4-way SMP system. Note that in a 4-way multiprocessor there are four predictors, one predictor per processor. We report average data for the four predictors.

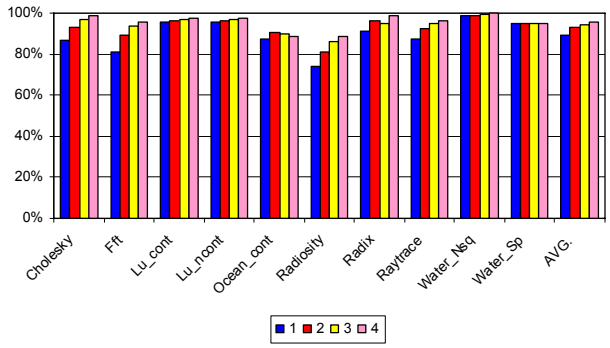
We define coverage as the percentage of all supplier nodes in cache to cache transfers that are accurately identified by predictors.

In figure 6.a, we report coverage for predictors with different sizes. We report for SSID-1, SSID-2, SSID-3 and SSID-4. We use thresholds values equal to zero, two, six, and 14, for SSID-1, SSID-2, SSID-3 and SSID-4 respectively. We picked these thresholds after testing different alternatives. In general, coverage is reduced as the counter size increases. On average, coverage varies from 51% to 77% for different counter sizes.

In Figure 6.b we report accuracy for predictors with different counter sizes. Accuracy shows how often the speculated supplier turns out to be the correct one. In general, accuracy improves as counter size increases. On average, accuracy changes from 89% to 96% for different counter sizes.



a) Coverage



b) Accuracy

Figure 6. Bars from left to right report coverage and accuracy for predictors equipped with 1-, 2-, 3- and 4-bit counters.

5.2.2 Performance

In this Section, we report performance for SSID-1, SSID-2, SSID-3, and SSID-4 compared to the baseline cache coherence protocol. Figure 7 reports performance for different benchmarks. Numbers less than one indicate a performance slowdown.

SSID has negligible impact on performance. For most benchmarks, the impact is less than 0.5%. Note that for some benchmarks (e.g., *raytrace*) SSID improves performance. This could be explained by the following: a) for these benchmarks, often the missing data is provided by local caches rather than the memory and b) SSID can speculate the supplier caches with high accuracy.

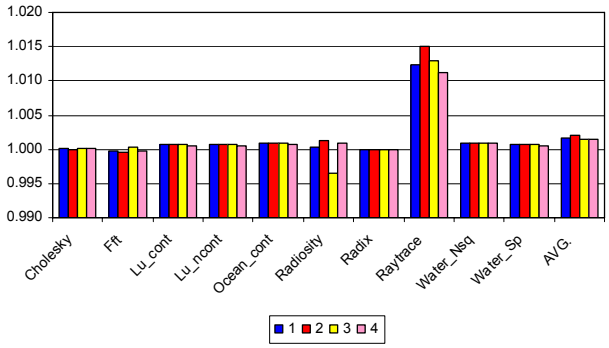


Figure 7. Performance for different predictor configurations. Bars from left to right report for predictors equipped with 1-, 2-, 3- and 4-bit counters.

5.2.3 Energy Reduction

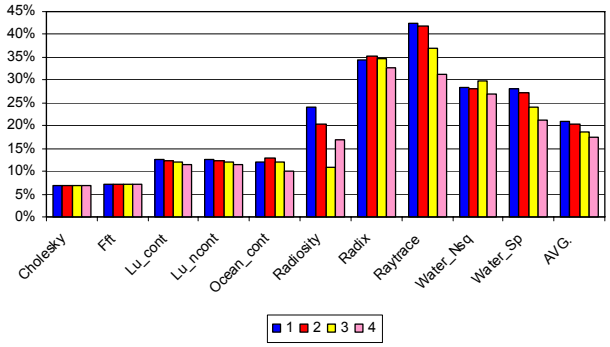
In Figure 8, we report energy reduction in links, switches, and tag arrays for SSID-1, SSID-2, SSID-3, and SSID-4 compared to the baseline cache coherence scenario. Generally, energy reduction is often lower for higher counter sizes. This is intuitive as smaller counters have higher coverage.

In Figure 8.a we report link energy reduction. On average, SSID-1, SSID-2, SSID-3 and SSID-4 reduce link energy by 21%, 20%, 18%, and 17% respectively. In Figure 8.b we report switch energy reduction. On average, SSID-1, SSID-2, SSID-3 and SSID-4 reduce switch energy by 22%, 21%, 21% and 20% respectively. In Figure 8.c we report energy reduction for tag arrays. SSID-1, SSID-2, SSID-3 and SSID-4 improve tag array energy by 17%, 15%, 13% and 11% respectively.

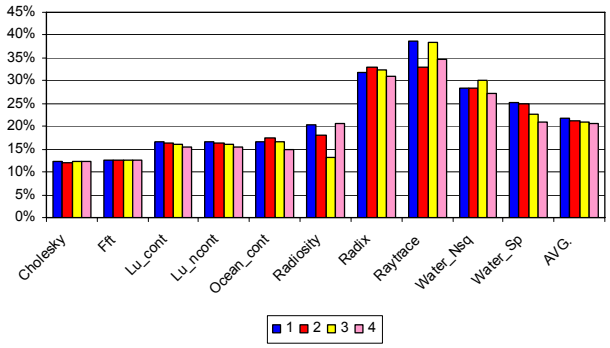
Our investigation shows that high performance and power improvement for *raytrace* is due to the fact that most of missed data are provided by peer caches. SSID speculates the supplier cache with high confidence and eliminates most of the unnecessary interconnect accesses.

Note that tag array energy reduction is close to zero for *cholesky* and *fft*. Our study shows that cache to cache transfers occur rarely in these benchmarks. As such, despite high accuracy, SSID does not improve tag array activity. However, the two benchmarks show energy reduction in links and switches as the result of the cache coherence step reduction explained in Section 3.2.

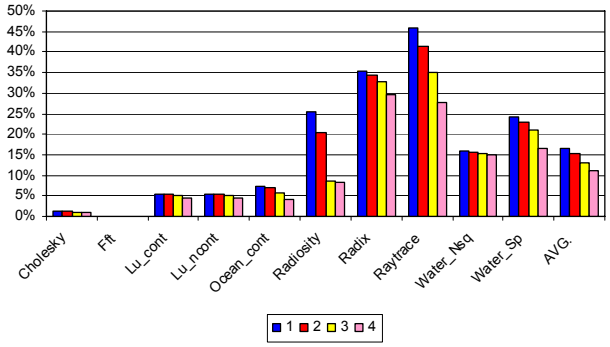
We conclude from Figure 8 that SSID-1 provides substantial power savings with negligible performance degradation, and minimal hardware overhead.



a) Link energy reduction



b) Switch energy reduction



c) Tag array energy reduction

Figure 8. Energy reduction in links, switches, and tag arrays. Bars from left to right report for predictors equipped with 1-, 2-, 3- and 4-bit counters.

5.2.4 Comparison with Serial Snooping

In this Section, we report performance and interconnect energy for SSID-1 and serial snooping in 4-, 8-, 16-, and 32-way SMP systems.

In Figure 9 we report performance relative to the baseline. Performance of SSID-1 is improved as higher numbers of nodes are exploited. On average, performance improves from 0.17% in a 4-way SMP to 1.7% in a 32-way SMP under SSID-1.

Serial snooping, on the other hand, shows much higher performance loss as the number of nodes is increased. In the 32-way SMP, serial snooping shows a performance loss of 19%. It should be noted that in serial snooping, cache to cache transfer latency depends on the number of links. In a binary tree with n leaves, the number of links is equal to 2n-2. As such, latency grows linearly with the number of processors in serial snooping. In SSID, however, latency depends on the height of tree which is equal to Log_2 n. Therefore, performance is less sensitive to the number of processor nodes under SSID.

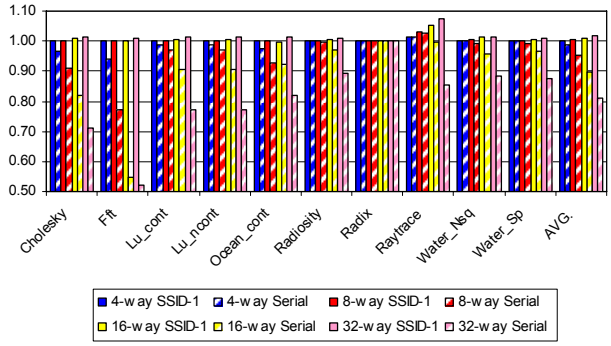


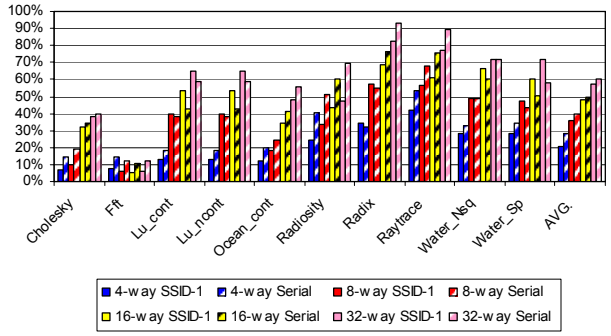
Figure 9. Performance for SSID-1 and serial snooping in 4-, 8-, 16-, and 32-way SMPs.

In some benchmarks, e.g. *fft*, serial snooping degrades performance considerably. In these benchmarks, quite often, memory is the supplier. However, in serial snooping, all remote caches are snooped one by one before a request is sent to the memory. This results in a considerable performance penalty.

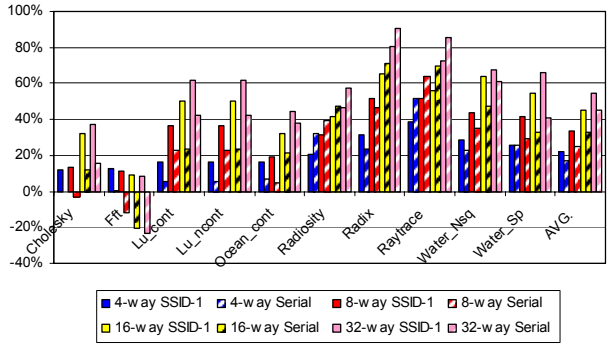
In Figure 10, we report energy reduction in interconnect and tag arrays for SSID-1 and serial snooping compared to baseline cache coherence scenario in 4-, 8-, 16-, and 32-way SMPs.

In Figure 10.a we report link energy. On average, SSID improves link energy by 21%, 36%, 48%, and 57% when the number of processors varies from 4 to 32. SSID falls slightly behind serial snooping in link energy. However, as Figure 9 shows, this comes with significant performance loss for serial snooping.

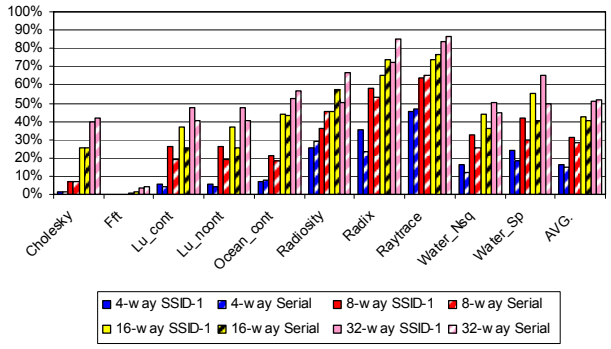
In Figure 10.b we report switch energy reduction. On average, SSID-1 reduces energy of switches from 22% to 55% when the number of processors increases from 4 to 32. In some benchmarks, e.g., *fft*, serial snooping increases switch energy up to 23%. On average, serial snooping falls behind SSID-1 for all SMP configurations. This is due to the fact that in serial snooping, whenever a cache lookup fails, the request is forwarded to the next node. Consequently, the closest switch to the processor is accessed at least twice. For example, in Figure 3, if P_1 receives a snoop request from P_0, and the requested address misses in P_1, the request is forwarded to P_2 through S_1 , and S_1 is accessed twice: once, when P_0 sends snoop request to P_1, and once when P_1 forwards snoop request to P_2. This is not the case under SSID.



a) Link energy reduction



b) Switch energy reduction



c) Tag array energy reduction

Figure 10. Energy reduction in links, switches, and tag arrays for serial snooping and SSID.

In Figure 10.c we report tag array energy reduction. Both methods improve tag array energy competitively. SSID-1 improves tag array energy from 17% to 51% for different number of processors.

We conclude from Figure 10 that SSID, while maintaining performance, improves interconnect and tag array energy as the number of ways increases in SMPs. On the other side, serial snooping harms performance considerably when the number of processors increases.

6. RELATED WORK

Acacio et al., exploited supplier (owner) locality to reduce latency of cache to cache transfer in cc-NUMA [12]. They used a two level predictor to convert 3-hop misses to 2-hop misses. The first level of predictor determined those misses that are satisfied by cache to cache transfers. The second level determined the list of nodes that have a valid copy of memory line. Requests were sent directly to the speculated nodes, removing the directory from the critical path. Martin et al. [16] proposed destination set prediction to achieve low latency of snoopy protocols and low bandwidth of directory protocols. They explored the design space by using different predictors. In all predictors, they used data or instruction address for indexing which requires exploiting large tables.

Our work is different from the above works, as we exploit supplier locality in snoopy cache coherence protocols and reduce energy (not latency). We report direct energy measurements and compare our technique to state-of-the-art power-aware cache coherence techniques. In addition, we report sensitivity analysis and study how changes in system architecture impact performance and energy improvements. Moreover, as energy is our major concern, we exploit predictors which are far simpler than that used in [12, 16].

Bjorkman et al. [23] proposed hints to reduce cache miss penalty. For each block in memory, they use one hint to identity the potential holder of the copy. When a cache miss can not be serviced in the local node, a request is sent to both home directory and to the hint node. If hint node has the copy, it sends it to the requestor and this reduces the cache miss delay by one hop. Otherwise, home directory provides data following convention method. While their method improves performance, sending two requests for each cache miss pollutes interconnect network and increases power dissipation in inter-node communication. In addition, they use hints for each memory block which increases hardware complexity dramatically. SSID, however, uses one SPL per node to reduce power dissipation in interconnects.

Saldanha and Lipasti [6] proposed serial snooping to reduce interconnects power. We introduce SSID as an alternative speculative approach and compare it to serial snooping.

Mukherjee and Hill [9] used prediction in distributed shared memory systems to speculate coherent messages in advance. Their work is based on the observation that memory blocks have a small number of repetitive sharing patterns. They used a general pattern-based predictor derived from two-level PAp branch predictor [10]. Memory Sharing Predictor (MSPs) [11] is a special type of general pattern-based predictor. MSP only predicts remote memory accesses and not the subsequent coherent messages. As such, it reduces predictor cost and improves accuracy.

While all works discussed above use speculation in directory-based cache coherence, we apply speculation in snoopy cache coherence to reduce power of interconnect.

In Jetty [13] snoops from remote nodes are filtered to reduce the number of L_2 cache accesses in SMPs. Each node has a filter on the bus side of the L_2 cache, checking the snoop requests sent from remote nodes. The filter identifies situations where the L_2 cache does not include the requested data and eliminates the associated extra L_2 tag arrays lookups. In RegionScout [14] a node determines in advance that a coarse grain region is not

available in none of the other nodes. As such, the request is sent directly to the memory, reducing both interconnect power and bandwidth. SSID can be used on top of Jetty and RegionScout possibly increasing power savings.

7. CONCLUSION

We introduced a power-aware speculative cache coherence protocol for SMPs. In our protocol we used a small and simple predictor to identify the node supplying local missing data. Consequently we limited sending requests only and directly to the speculated supplier. We saved power by avoiding broadcasting whenever there is high confidence in the prediction outcome. We identified and removed a large portion of unnecessary interconnect activities with high accuracy. We reduced energy of links, switches, and tag arrays by 21%, 22%, 17% respectively.

8. ACKNOWLEDGMENTS

This work was supported by the Natural Sciences and Engineering Research Council of Canada, Discovery Grants Program and Canada Foundation for Innovation, New Opportunities Fund.

9. REFERENCES

[1] S. Mukherjee et al., The Alpha 21364 network Architecture, IEEE Micro, Volume 22 Issue 1, pp.26-35, 2002.

[2] Alan E. Charlesworth. The Sun Fireplane System Interconnect, In Proceedings of the 2001 ACM/IEEE conference on Supercomputing, 2001.

[3] M. E. Acacio, J. Gonzalez, J. M. Garcia, and J. Duato, The Use of Prediction for Accelerating Upgrade Misses in CCNUMA Multiprocessors, In Proceedings of PACT-11, 2002.

[4] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The SPLASH-2 Programs: Characterization and Methodological Considerations. In International Symposium on Computer Architecture, June 1995.

[5] Robert C. Steinke, Gary J. Nutt, A unified theory of shared memory consistency, Journal of the ACM (JACM), v.51 n.5, p.800-849, September 2004.

[6] C. Saldanha and M. H. Lipasti, Power Efficient Cache Coherence, High Performance Memory Systems, Springer-Verlag, 2003.

[7] J. Renau, B. Fraguela, J. Tuck, W. Liu, M. Prvulovic, L. Ceze, K. Strauss, S. Sarangi, P. Sack, and P. Montesinos. SESC Simulator, Jan 2005. http://sesc.sourceforge.net.

[8] E. E. Bilir, R. M. Dickson, Y. Hu, M. Plakal, D. J. Sorin, M. D. Hill, and D. A. Wood, Multicast Snooping: A New Coherence Method using a Multicast Address Network, SIGARCH Comput. Architure News, pp. 294–304, 1999.

[9] Shubhendu S. Mukherjee and Mark D. Hill, Using prediction to accelerate coherence protocols, In Proceedings of the 25th Annual International Symposium on Computer Architecture, June 1998.

[10] Tse-Yuh Yeh and Yale Patt, Alternative implementations of two-level adaptive branch prediction, In Proceedings of the 19th Annual International Symposium on Computer Architecture, May 1992.

[11] A.-C. Lai and B. Falsafi, Memory sharing predictor: the key to a speculative coherent DSM, In Proceedings of the 26th annual international symposium on Computer architecture, pp. 172–183, 1999.

[12] M. E. Acacio, J. González, J. M. García, and J. Duato, Owner Prediction for Accelerating Cache-to-Cache Transfers in a cc-NUMA Architecture, In Proceedings of SC2002, Nov. 2002.

[13] A. Moshovos, B. Falsafi and A. Choudhary, JETTY: Filtering Snoops for Reduced Energy Consumption in SMP Servers, In Proceedings of the 7th International Symposium on High-Performance Computer Architecture, January 2001.

[14] J. Cantin, A. Moshovos, M. Lipasti, J. Smith, and B. Falsafi, Coarse-Grain Coherence Tracking: RegionScout and Region Coherence Arrays, IEEE Micro, v.26, n.1, pp. 70-79, Jan-Feb 2006.

[15] J. Huh, J. Chang, D. Burger, and G. S. Sohi, Coherence Decoupling: Making Use of Incoherence, In Proceedings of ASPLOS-XI, pp. 97-106, 2004.

[16] Milo M. K. Martin, Pacia J. Harper, Daniel J. Sorin, Mark D. Hill, and David A. Wood, Using Destination-Set Prediction to Improve the Latency/Bandwidth Tradeoff in Shared-Memory Multiprocessors, In Proceedings of the 30th Annual International Symposium on Computer Architecture, pages 206-217, 2003.

[17] K. M. Lepak and M. H. Lipasti, Temporally Silent Stores, In Proceedings of ASPLOS-X, pages 30–41, 2002.

[18] H. S. Wang, X. P. Zhu, L. S. Peh, and S. Malik. Orion: A Power-Performance Simulator for Interconnection Networks. In International Symposium on Microarchitecture, Nov. 002.

[19] P. Shivakumar and N. Jouppi. CACTI 3.0: An Integrated Cache Timing, Power and Area Model. Technical Report 2001/2, Compaq Computer Corporation, Aug. 2001.

[20] R. Iyer, L. N. Bhuyan and A. Nanda. "Using Switch Directories to Speed Up Cache-to-Cache Transfers in CCNUMA Multiprocessors", In Proc. of the 14th Int'l Parallel and Distributed Processing Symposium (IPDPS'00), pp. 721–728, May 2000.

[21] L. A. Barroso, K. Gharachorloo and E. Bugnion, "Memory System Characterization of Commercial Workloads", In Proc. of the 25th Int'l Symposium on Computer Architecture (ISCA'98), pp. 3–14, June 1998.

[22] D. E. Culler, J. Singh, A. Gupta, Parallel Computer Architecture: A Hardware/Software Approach, Morgan Kaufmann Publishers, San Francisco, Calif., 1998.

[23] M. Bjorkman, F. Dahlgren, and P. Stenstrom, Using Hints to Reduce the Read Miss Penalty for Flat COMA Protocols, In Proceedings of the 28th Annual Hawaii International Conference of System Sciences, pages 242-251, January 1995.

[24] A. Landin, E. Hagersten, and S. Haridi, Race-free interconnection networks and multiprocessor consistency, In Proc. of the 18th Intl. Symp. on Comp. Architecture, 1991.