

29. A Coherence Protocol for the Elimination of Passive Sharing in Single and Multiple Threaded Shared-Bus Shared-Memory Multiprocessors

Roberto Giorgi,
University of Alabama at Huntsville
and Cosimo Antonio Prete,
University of Pisa

Single-chip multiprocessors and multiple-thread architectures are becoming an affordable solution for high-performance general-purpose workstations and servers.

On these machines, the workload is typically constituted of both sequential and parallel applications. Shared-bus shared-memory multithreaded multiprocessor can be used to speed-up the execution of such workload. In this environment, the scheduler takes care of the load balancing by allocating a ready process on the first available processor, thus producing process migration.

Process migration and the persistence of private data into different caches produce an undesired sharing, named passive sharing. The copies due to passive sharing produce useless coherence traffic on the bus and coping with such a problem may represent a challenging design problem for these machines.

Many protocols use smart solutions to limit the overhead to maintain coherence among shared copies. None of these studies treats passive-sharing directly, although some indirect effect is present while dealing with the other kinds of sharing. Affinity scheduling can alleviate this problem, but this technique does not adapt to all load conditions, especially when the effects of migration are massive.

A simple coherence protocol is presented. This protocol eliminates passive sharing using information from the compiler that is normally available in operating system kernels. The performance of this protocol has been evaluated and compared against other solutions proposed in the literature by means of enhanced trace-driven simulation.

The performance of the proposed solution outperforms the other protocols, especially in the case of a multithreaded processor, thus demonstrating its effectiveness in this kind of hardware platform.

The complexity of the proposed approach has been evaluated in terms of the number of protocol states, additional bus lines and required software support.

The protocol further limits the coherence-maintaining overhead by using information about access patterns to shared data exhibited in parallel applications.

University of Pisa, ITALY



UAH

University of Alabama in Huntsville, USA

**A Coherence Protocol
For the Elimination of Passive Sharing
In Single and Multiple Threaded
Shared-Bus Shared-Memory Multiprocessors**

Roberto Giorgi and Cosimo Antonio Prete

contact address: giorgi@acm.org

Outline

- What is Passive Sharing?
- Techniques that reduce Passive Sharing
- Our solution at Protocol Level
- Performance Evaluation against other Protocols
- Effectiveness in Single-Context Multiprocessors and in Multiple-Context Multiprocessors
- Conclusions and References

Introduction

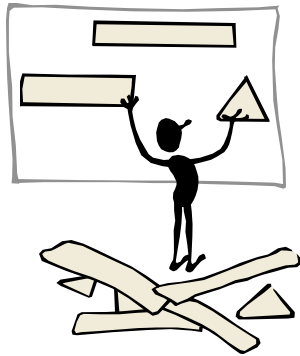
- Basic Multiprocessor Architecture:
N processors connected by a simple bus
- Our goal:
to evaluate the problems that limit the scalability of this machine while it runs commercial applications
- One source of problems is:
the presence of private-data copies in processor caches
- Why these copies are present:
the operating system allows process migration in order to achieve load balancing
- In current architectures those data appear as shared, generating unnecessary occupation of critical resources (the bus)
- We named this phenomenon “Passive Sharing” and we studied solutions to limit and reduce its effects

Our Reference Papers

- **Methodology:**
R. Giorgi, C.A. Prete, G. Prina, L. Ricciardi, "Trace Factory: Generating Workloads for Trace-Driven Simulation of Shared-Bus Multiprocessors", *IEEE Concurrency*, Vol. 5, No. 4, Oct. 1997.
- **Performance Evaluation:**
R. Giorgi, C.A. Prete, “PSCR: A Coherence Protocol for Eliminating Passive Sharing in Shared-Bus Shared-Memory Multiprocessors”, to appear on *IEEE Transactions on Parallel and Distributed Systems*

What is Passive Sharing?

TYPES OF SHARING
in multiprocessor systems

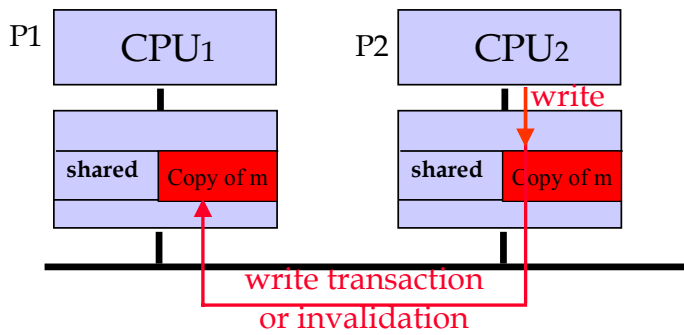


ACTIVE

FALSE

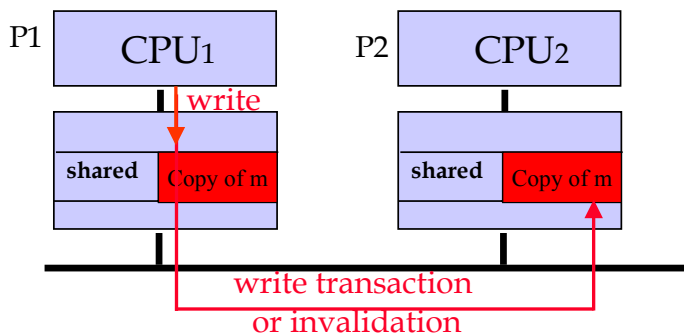
PASSIVE

Active (or True) Sharing



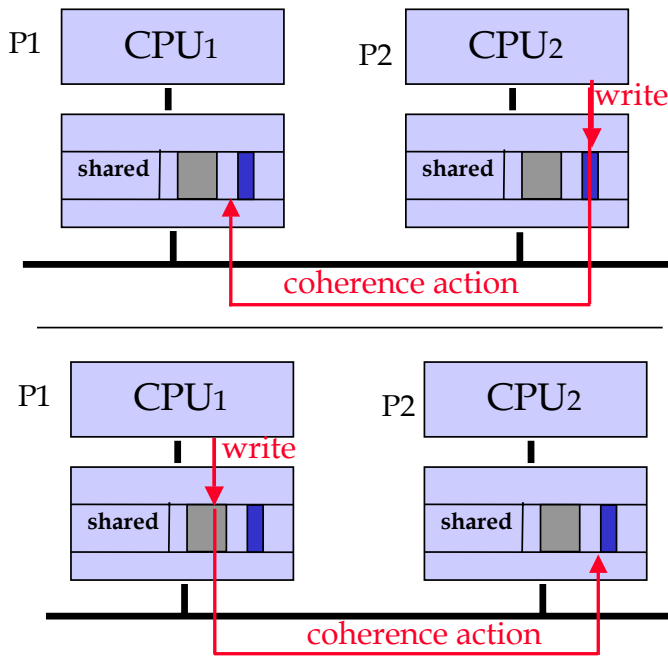
– The processes P1 e P2 really share the information m

– A bus action **IS NECESSARY** in order to keep coherent those copies



[Gupta92] [Adve91]

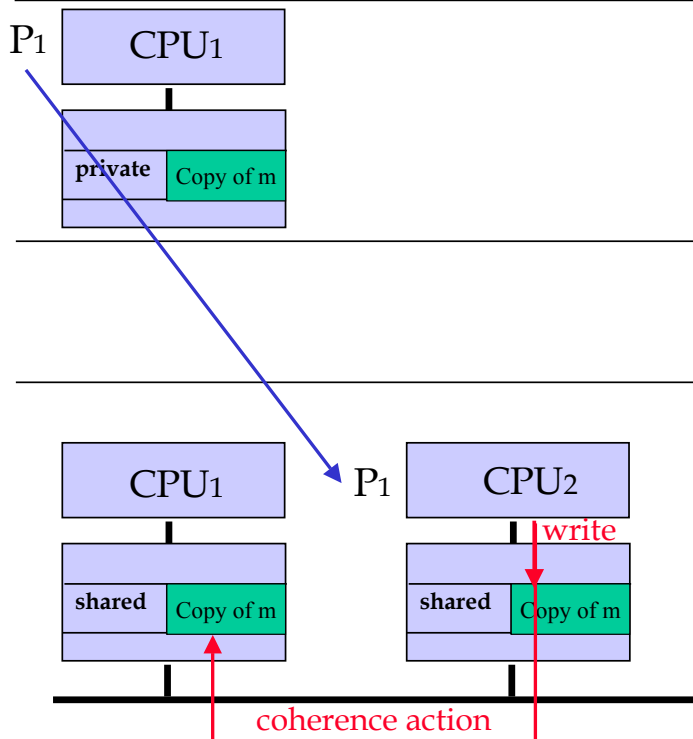
False Sharing



- The data block contains several variables always accessed separately
- The processes P1 and P2 DO NOT really share information
- Is it necessary a bus action in order to keep coherent those copies?

[Torrellas90] [Eggers91]

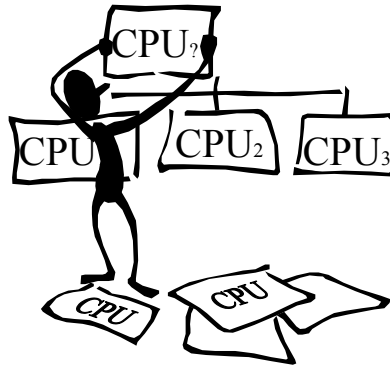
Passive Sharing



- Process P1 access the *private* data **m**
- Process P₁ is preempted ... and re-activated lately on a different processor
- When P1 access again its private data **m** both copies become *passively* shared
- Is it necessary a bus action in order to keep coherent those copies?

[Prete90] [Prete97] 

Techniques that Reduce Passive Sharing



Cache Flushing

- *Cache flushing*:
the cache is flushed at each context-switch
- **Advantages**
 - supported by almost all cache-controllers
- **Disadvantages**
 - loss of potentially useful information
 - bus-traffic peak due to dirty-copy updating
 - bus-traffic peak due to reload of working-set on new processor
 - this happens even if we selectively flush private data and private copies

Cache Affinity Scheduling

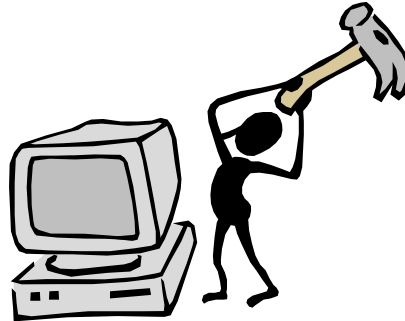
- *Cache-Affinity Scheduling:*
the process scheduler tries to resume a process on the processor where it last executed
- **Advantages**
 - the process can re-use part of its working-set eventually still present in that processor's cache
 - well-known technique currently implemented in many operating systems
- **Disadvantages**
 - cache-affinity is not always applicable!
The scheduler can be forced to schedule a certain process on another processor, especially when the number of “ready” processes is low

Write-Invalidate Protocols

- *Write-Invalidate Protocols:*
the remote data is invalidated at the first write access
- **Advantages**
 - passive copies are slowly and “spontaneously” eliminated without generating bus-traffic peaks
 - this kind of protocol is available in almost all high-performance CPUs (like Intel Pentium II and III, AMD K6, PowerPC, UltraSPARCII)
- **Disadvantages**
 - passive copies still persist and generate unnecessary bus-traffic

Our Solution to Eliminate Passive Sharing

- A Coherence Protocol for Eliminating Passive Sharing

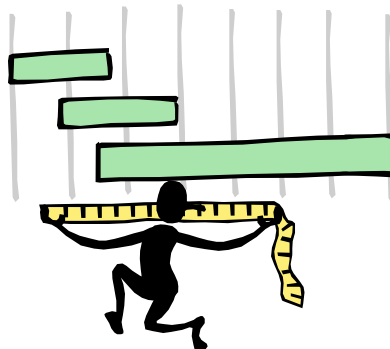


Our Proposed Solution



- By intervening at protocol level, it is possible to eliminate Passive Sharing (**PSCR, Passive-Shared Copy Removal** [Prete97])
- **Idea:** a private block is locally invalidated as soon as a different processor tries to fetch it
- **Advantages**
 - adaptivity to any situation imposed by the workload or operating system
 - compared to other protocols there's no additional cost in terms of bus transactions



Performance Evaluation



Workloads and Protocols

- The simulation are carried out by means of Trace Factory environment [Giorgi97e] 
 - simulation is “kind of” trace-driven, which also include the modelling of most influencing operating system activities (kernel references, process scheduling, virtual memory)
- Trace Factory produced the traces of real programs in a scenario of real multitasking. The generated workloads are:
 - UniP: 30 different sequential programs
 - Mix1/Mix2: like UniP plus a multithreaded application
 - OLTP, DSS (Database, benchmarck TPC-B, TPC-D), WEB Server
- Coherence protocols considered:
 - Write-Update class: Dragon [McCreight84]
 - Write-Invalidate class: Berkeley, MESI
 - Hybrid class: Competitive [Karlin86], Update-Once [Gee93] (Smith’s protocol)
 - Selective class: PSCR, AMSD (protocol for migratory sh.[Stenstrom93][Cox93]) 

Performance Evaluation

- Reference Machine

- Cache
 - cache size = 256 KBytes
 - block size = 64 Bytes
 - number of ways = 1 (direct access)
- Bus
 - width = 64 bit
 - read-block transaction cost = 32 cycles
 - cache-to-cache transaction cost = 24 cycles
 - write transaction cost = 5 cycles
 - update transaction cost = 18 cycles
 - invalidate transaction cost = 5 cycles

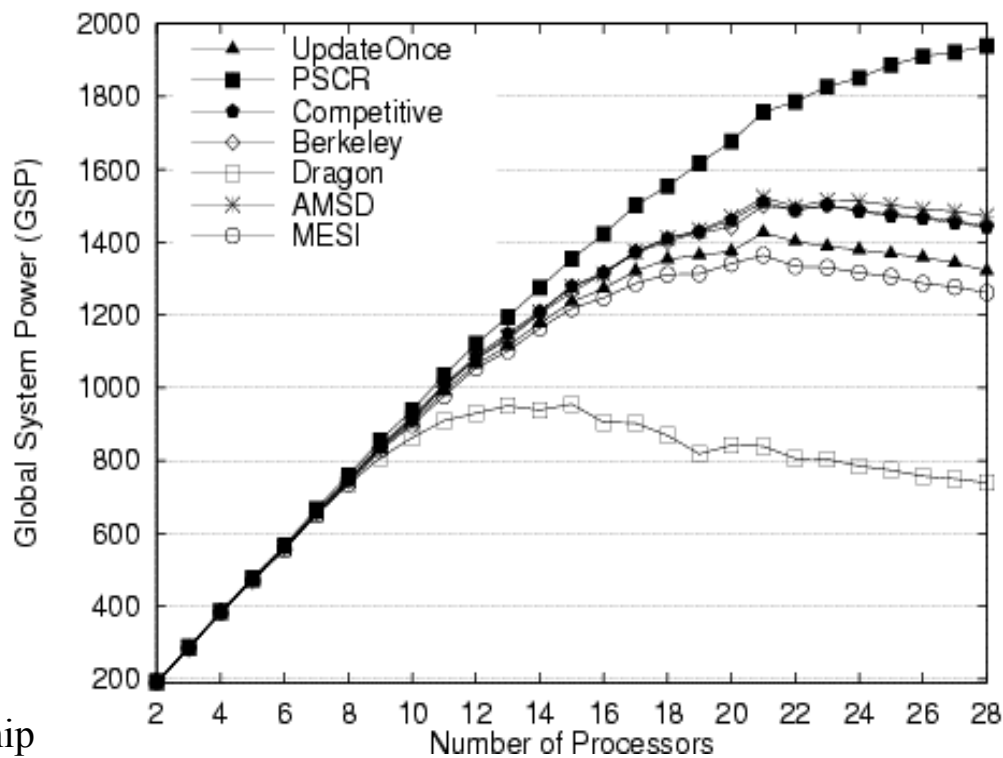
- Metric used in our comparisons

- GSP Global System Power [Archibald86]:

$$GSP = 100 \times \sum U_{cpu}$$

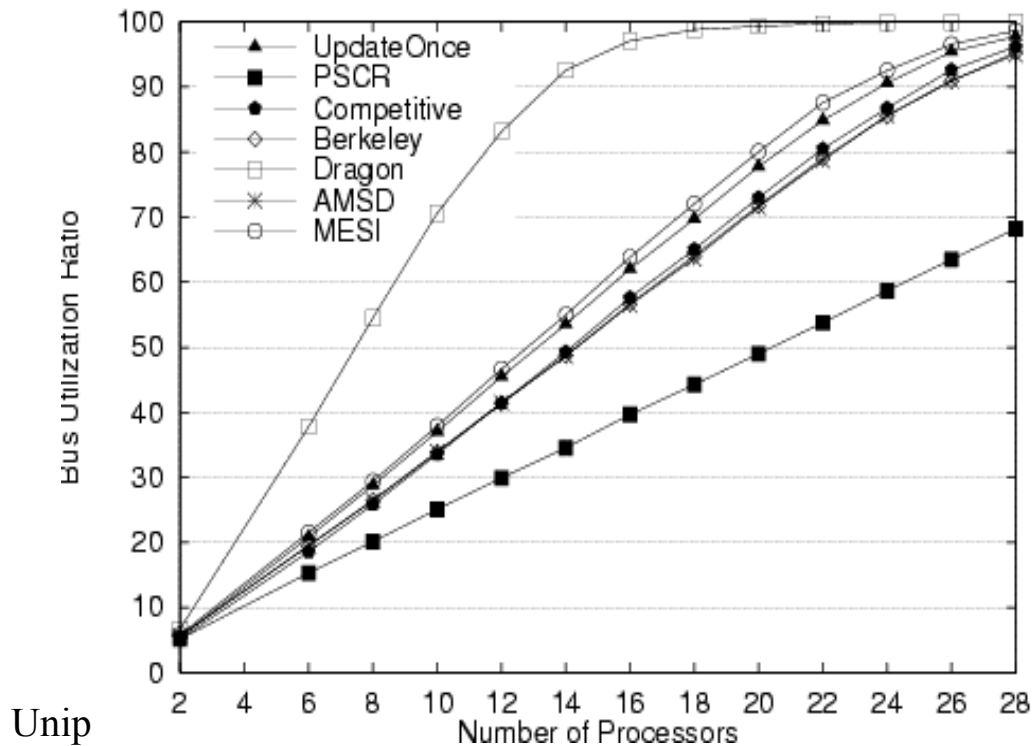
- $U_{cpu} = (T_{cpu} - T_{delay}) / T_{cpu}$
- T_{cpu} time to execute the whole workload
- T_{delay} time that the CPU has to wait for memory access completion

Comparison against other Protocols (1)

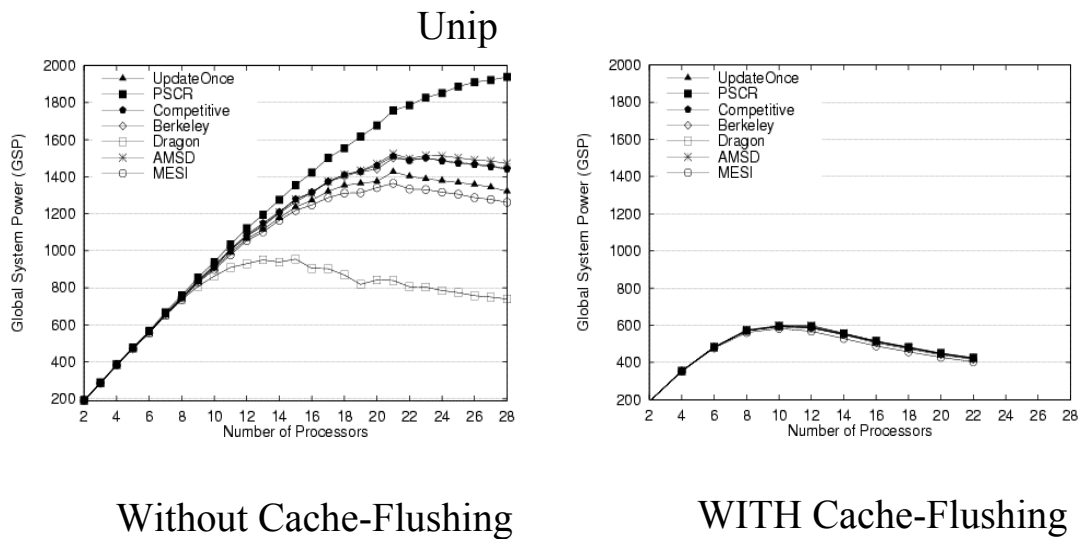


Unip

Comparison against other Protocols (2)

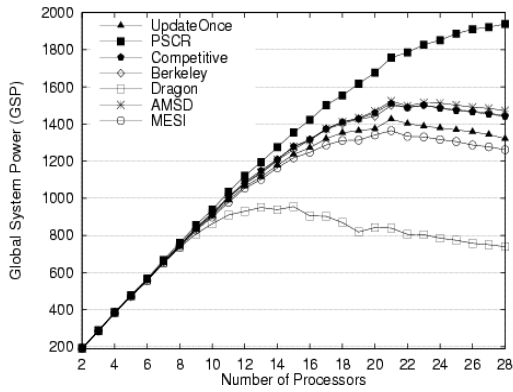


Effect of Cache-Flushing

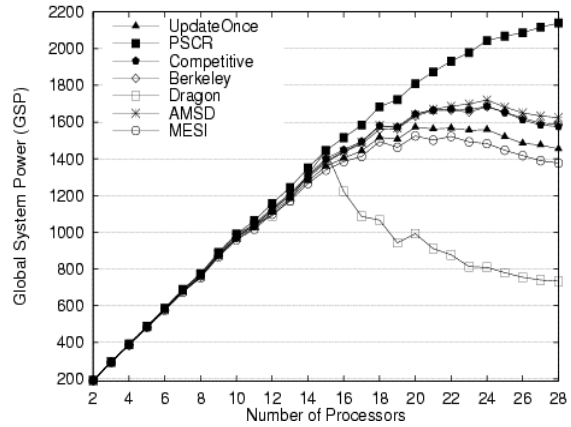


Effect of Affinity Scheduling

Unip



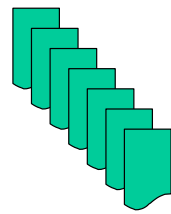
Generic Scheduling



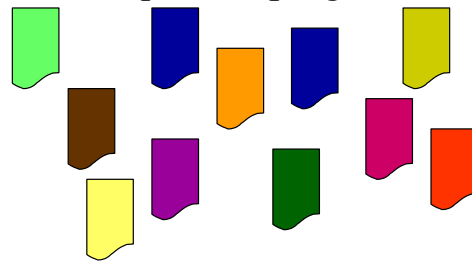
Cache-Affinity Scheduling

Multi-Context Multiprocessor Model

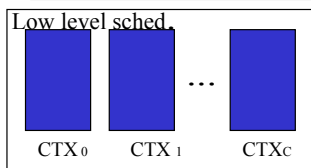
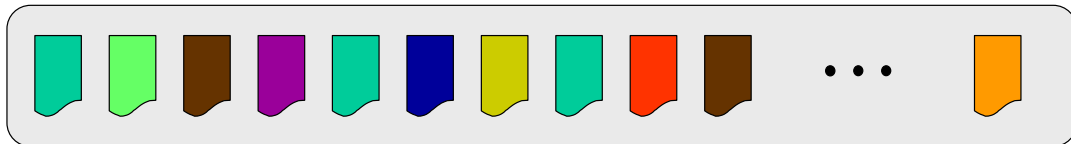
Multithreaded program



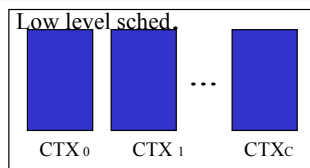
Sequential programs



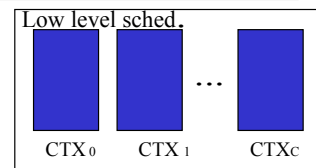
High level sched.



CPU₀

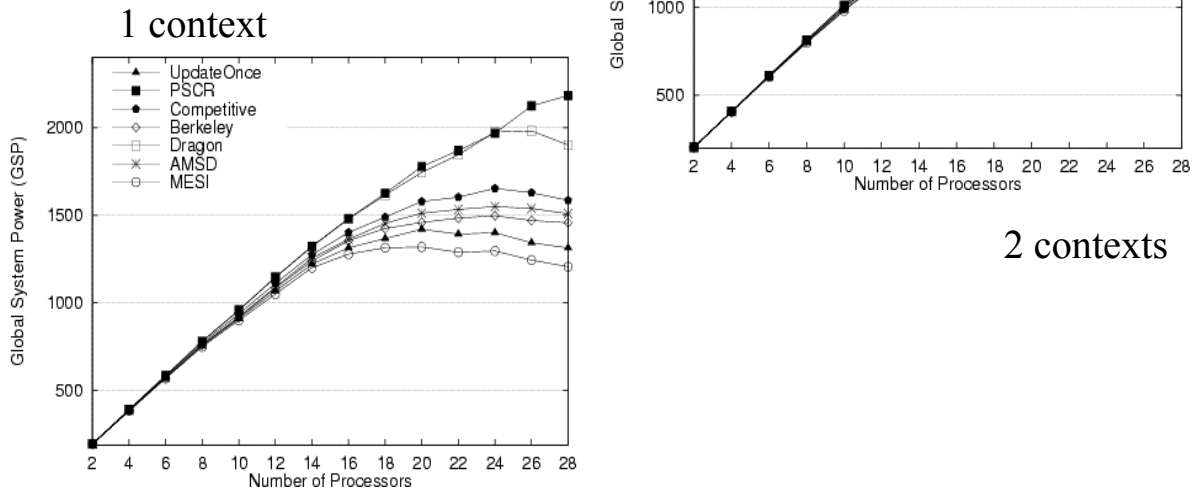


CPU₁



CPU_p

Multi-Context Case



Conclusions

- A solution to eliminate Passive Sharing has been evaluate against:
 - other techniques that may reduce this phenomenon
 - other coherence protocols that are currently know and implemented
 - in the case of single and multiple-context multiprocessors
- In our experiments on a shared-bus shared-memory multiprocessor, running a composite workload, PSCR protocol:
 - outperform the other coherence protocols
 - enhance the scalability of the machine

References

- [Adve91] S. V. Adve, M. D. Hill, and M. Vernon, "Comparison of Hardware and Software Cache Coherence Schemes," *Proc. of the 18th Int'l Symp. on Computer Architecture*, pp. 298-308, May 1991.
- [Archibald86] J. K. Archibald and J. L. Baer, "Cache Coherence Protocols: Evaluation Using a Multiprocessor Simulation Model," *ACM Trans. on Comp. Systems*, vol. 4, pp. 273-298, Apr. 1986.
- [Cox93] A. L. Cox and R. J. Fowler, "Adaptive Cache Coherency for Detecting Migratory Shared Data," *Proc. 20th Int'l Symp. on Computer Architecture*, San Diego, California, pp. 98-108, May 1993.
- [Eggers91] S. J. Eggers, "Simplicity versus Accuracy in a Model of Cache Coherency Overhead," *IEEE Trans. Computers*, vol. 40, no. 8, pp. 893-906, Aug. 1991.
- [Gee93] J. G. Gee and A. J. Smith, "Absolute and Comparative Performance of Cache Consistency Algorithms," Tech. Rep. UCB/CSD-93-753, EECS Computer Science Division, University of California, Berkeley, 1993.
- [Giorgi97e] R. Giorgi, C. Prete, G. Prina, and L. Ricciardi, "Trace Factory: Generating Workloads for Trace-Driven Simulation of Shared-Bus Multiprocessors," *IEEE Concurrency*, vol. 5, no. 4, pp. 54-68, Oct. 1997.
- [Gupta92] A. Gupta and W.-D. Weber, "Cache Invalidation Patterns in Shared-Memory Multiprocessors," *IEEE Trans. Computers*, vol. 41, no. 7, pp. 794-810, July 1992.
- [Karlin86] A. Karlin, M. Manasse, L. Rudolph, and D. Sleator, "Competitive Snoopy Caching," *Proc. 27th Symp. on Foundations of Computer Science*, pp. 244-254, Oct. 1986.
- [McCreight84] E. M. McCreight, "The Dragon computer system: an early overview," *NATO Advanced Study Institute on Microarchitecture of VLSI Computer*, Urbino, Italy, July 1984.
- [Prete90] C. A. Prete, "A new solution of coherence protocol for tightly coupled multiprocessor systems," *Microprocessing and Microprogramming*, vol. 30, no. 1-5, pp. 207-214, 1990.
- [Prete97] C. A. Prete, G. Prina, R. Giorgi, and L. Ricciardi, "Some Considerations About Passive Sharing in Shared-Memory Multiprocessors," *IEEE TCCA Newsletter*, pp. 34-40, Mar. 1997.
- [Squillante93] M. S. Squillante and D. E. Lazowska, "Using processor-cache affinity information in shared-memory multiprocessor scheduling," *IEEE Trans. Parallel Distributed Systems*, vol. 4, no. 2, pp. 131-143, Feb. 1993.
- [Stenström93] P. Stenström, M. Brorsson, and L. Sandberg, "An Adaptive Cache Coherence Protocol Optimized for Migratory Sharing," *20th Int'l Symp. on Computer Architecture*, pp. 109-118, May 1993.
- [Torrellas90] J. Torrellas, M. S. Lam, and J. L. Hennessy, "Share Data Placement Optimizations to Reduce Multiprocessor Cache Miss Rates," *Proc. 1990 Int'l Conf. on Parallel Processing. Vol. 2: Software*, Urbana-Champaign, IL, pp. 266-270,

Aug. 1999