IS2202 - Computer Systems Architecture

Multiprocessors and Memory Ordering

Bonus assignment 2

-Submitted by

Tamilselvan Shanmugam tamsha@kth.se

1. Cache Coherence

1.1 False Sharing

False Sharing miss:

False sharing miss happens when two threads access independent data that are happen to fall on the same cache line. Unnecessarily each invalidates other cache copy, forcing to get fresh copy. This adds more overhead and deteriorates performance.

To overcome false sharing

S/W Techniques:

Programmer can allocate variables to go to different cache lines using `__attribute__((align(64)))`.This is called padding, spacing technique[1]. Complexity increases as the program grows.

Compiler can detect this kind of false sharing and allocate that particular variable in register. It's hard to detect for complex programs.

Increase chunk value gives better result against false sharing[2]. This means the chance of two thread variables falling into same cache line is getting reduced.

H/W Technique

Implementing invalidation bits per word level will totally remove the false sharing[3] since it avoids cache line level invalidation checking. On the counter part this increases number of comparisons and invalidations sent via bus.

1.2 MSI-Coherence

Step	CPU0	CPU1	CPU2	CPU3	on bus	data source
0	WR 0				RTW	MEM
1			RD 0		RTS	Cache 0
2				WR 0	RTW	MEM
3			WR 0		RTW	Cache 3
4		WR 0			RTW	Cache 2
5			RD 0		RTS	Cache 1
6			WR 0		INV	Cache 2(Self)

Cache states:

step	Cache 0	Cache 1	Cache 2	Cache 3
0	M			
1	S		S	
2	I		I	M
3	I		M	I
4	I	M	I	I
5	I	S	S	I
6	I	I	M	I

1.3 MOSI Protocol

MOSI protocol introduces new state called "Owner" that has the privilege of not writing back the latest data (Owned state) to main memory, thereby reducing unwanted bus traffic.

2. Memory Consistency

5. Sequentially consistent: 2

It's equivalent to running all processor instructions in a single processor with interleaving. Those interleaving are arranged such a way that data dependency is achieved. Positioning all the instructions in order always prints the value 2. Additional info: Intel's 386-based Compaq SystemPro implements SC model[4].

Total Store Order: 2

TSO allows a read to complete before a write to different memory location in the same processor[5]. But it doesn't re-order the writes of same processor. It happens as in the program order. So TSO always prints 2. Additional info: IBM-370 implements TSO consistency[4].

Release Consistency: timing dependent

The code segment doesn't use any `Lock & Unlock` mechanisms. So operation `Flag=1;` may be completed faster than `A=2;` Another CPU reading `A` will not get the updated value. The result is not consistent.

6. Sequential consistency instead of release consistency

Sequential consistency always produces correct results nevertheless of program. It's easy for programmers to visualize sequential consistency. No need of synchronization mechanism. Preferred in simple applications, numbers of processors/cores are minimal.

On the other hand, release consistency doesn't guarantees the instructions are executed in the same order. Programmer has to write explicit acquire, release locks and accountable for how the data flows.

7. TSO instead of Sequential consistency

Total Store Order allows read to happen prior to writes. This hides memory latency and improves performance. The speedup gradually improves for weak consistency models as number of core increases[7].

Whereas sequential consistency has high latency, that degrades overall performance.

8. Why weaker memory model?

Though Strong memory model ensures correctness of the program, main purpose of multi processors (speedup) will not be achieved as expected because of the memory latency to transfer data across processors. Weaker memory model re-orders and overlaps memory access to improve speedup[8].

9. pthread and memory model

Considering the application uses pthreads and correctly synchronized, means the application is thread safe. No two memory access is allowed at same time if they are synchronous threads. In addition they synchronize memory with other threads[9]. This itself ensures data consistency. So, weak memory model doesn't affect the correctness of correctly synchronized applications.

References:

[1]"c - Aligning to cache line and knowing the cache line size - Stack Overflow." [Online]. Available: http://stackoverflow.com/questions/7281699/aligning-to-cache-line-and-knowing-the-cache-line-size. [Accessed: 10-Apr-2014].

[2]"5.4. Multithreading Problems." [Online]. Available: http://www.roguewave.com/portals/0/products/threadspotter/docs/2012.1/linux/manual_html/multithreading_problems.html. [Accessed: 10-Apr-2014].

[3]"6.2.2 Reducing False Sharing (Sun Studio 12: OpenMP API User's Guide)." [Online]. Available: http://docs.oracle.com/cd/E19205-01/819-5270/aewcz/index.html. [Accessed: 10-Apr-2014].

[4]"CSC/ECE 506 Spring 2013/10b ps - PG_Wiki." [Online]. Available: http://wiki.expertiza.ncsu.edu/index.php/CSC/ECE_506_Spring_2013/10b_ps. [Accessed: 12-Apr-2014].

[5]"Relaxed Consistency Models." [Online]. Available: http://www.cs.utah.edu/~rajeev/cs7820/pres/7820-12.pdf. [Accessed: 12-Apr-2014].

[6]"Release Consistency (2)." [Online]. Available: http://regal.csep.umflint.edu/~swturner/Classes/csc577/Online/Chapter06/img20.html. [Accessed: 12-Apr-2014].

[7]"Realization and Performance Comparison of Sequential and Weak Memory Consistency Models in Network-on-Chip based Multi-core Systems."

[8]"Lecture 14: Relaxed Memory Consistency." [Online]. Available: http://www.cs.cmu.edu/afs/cs/academic/class/15418-s12/www/lectures/14_relaxedReview.pdf. [Accessed: 12-Apr-2014].

[9]"General Concepts." [Online]. Available: http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap04.html#tag_04_11. [Accessed: 12-Apr-2014].