ECE 4100 Project 3 Report

Multi-Processor Cache Coherence

Ruoyang Xu

Last Edited: April 19, 2019

Introduction

This is the report for ECE 4100 Advanced Computer Architecture Project 3: Cache Coherence. The simulator will implement four protocols on a bus-based broadcast system to ensure cache coherence across different processor caches: MSI, MOSI, MESI, MOESIF.

The simulations would be conducted on 4, 8 and 16 core system. Each core in the system has one level of cache. This cache is associative and of infinite size. The protocols will be tested on 8 test environment settings.

Individual Trace Experiment

This section investigates the best protocol for each experiment.

Experiment 1

The result for each configuration is shown in Table 1. Experiment 1 is a 4-processor trace where each processor only has 3 memory requests. Shown in the table, MOESIF and MOSI protocols performs better than MSI and MESI. This is largely due to the fact that MOESIF and MOSI were able to provide cache-to-cache transfer even when data is being modified. This has led to the elimination of one less 100-cycle response from memory and produce a smaller runtime.

The recommended protocol would be **MOSI** since MOESIF and MOSI have the same performance and it is easier to implement MOSI than MOESIF.

Table 1: Experiment 1 Result

	MSI	MOSI	MESI	MOESIF
Runtime	317	217	317	217
Cache Miss	7	7	7	7
Cache Access	12	12	12	12
Silent Upgrade	0	0	0	0
$-to-$ Transfer	4	5	4	5

Experiment 2

The result for each configuration is shown in Table 2. Experiment 2 is a 4-processor where the majority of the requests are read and the few writes are concentrated around a number of times. This has led to the fact that the ability to share on top of modified data really valuable. MOSI and MOESIF again were able to provide more cache to cache transfer and save more time.

The recommended protocol would be **MOESIF**. Because it has the lowest runtime and the second best option is not fast enough to amend for decreased complexity.

Table 2: Experiment 2 Result

	MSI	MOSI	MESI	MOESIF
Runtime	2367	1167	2367	987
Cache Miss	30	30	30	30
Cache Access	104	104	104	104
Silent Upgrade	0	0	1	1
$-to-$ Transfer	7	19	7	24

Experiment 3

The result for each configuration is shown in Table 3. Experiment 3 is a 8-processor simulation that every four cores operates around a similar range of memory location. Processors has intense read and write operations yet rarely share the same memory location. This has cause the Exclusive state to be very useful thus the outstanding performance of MESI and MOESIF.

The optimal configuration is **MOESIF**, given that it takes the minimal runtime and is 1.685 times faster than the second best.

Table 3: Experiment 3 Result

	MSI	MOSI	MESI	MOESIF
Runtime	3723	3723	2907	1725
Cache Miss	56	56	48	48
Cache Access	200	200	200	200
Silent Upgrade	0	0	8	8
$-to-$ Transfer	20	20	20	32

Experiment 4

The result for each configuration is shown in Table 4. Experiment 4 is a 4-processor simulation where one core writes and reads to memory locations and all three other cores have identical memory access traces and only do read operations. Since there is write operation to memory locations, both Owned and Exclusive state would be very useful thus MOESIF, the combination of both would be even more optimal than MOSI and MESI. **MOESIF** is undoubtedly the best configuration since it has the smallest number of cache miss and largest number of cache-to-cache transfer, leading to minimal runtime.

Table 4: Experiment 4 Result

	MSI	MOSI	MESI	MOESIF
Runtime	2265	1869	1647	751
Cache Miss	27	29	19	19
Cache Access	60	60	60	60
Silent Upgrade	0	0	3	3
$-to-$ Transfer	5	11	3	12

Experiment 5

The result for each configuration is shown in Table 5. Experiment 5 is a 8-processor simulation. Processors compete around memory location 0x000BEEF0 while other memory locations are not really shared-accessed. The same location being accessed by all defeats the purpose of an Exclusive state and it is quite apparent that MESI have the exact same performance as MSI. **MOESIF** wins over MOSI by having a Forward State and providing more cache-to-cache transfer.

Table 5: Experiment 5 Result

	MSI	MOSI	MESI	MOESIF
Runtime	1661	1261	1661	561
Cache Miss	21	21	21	21
Cache Access	37	37	37	37
Silent Upgrade	0	0	0	0
$-to-$ Transfer	5	9	5	16

Experiment 6

The result for each configuration is shown in Table 6. Experiment 6 is a 16-processor simulation. 8 cores are only reading memory addresses and four of the eight reads 0x5000000 for 70+ times. The rest four of eight writes to memory locations that is only read by itself. Of the remaining 8 cores, four cores does the exact same operation and only reads, leaving the last four competing cache coherece aroudn 0x000BEEF0. This is a similar scenario with experiment 5 and **MOESIF** undoubtedly wins.

Table 6: Experiment 6 Result

	MSI	MOSI	MESI	MOESIF
Runtime	7775	6975	5225	3425
Cache Miss	87	87	62	62
Cache Access	747	747	747	747
Silent Upgrade	0	0	25	25
$-to-$ Transfer	12	20	12	30

Experiment 7

The result for each configuration is shown in Table 7. Experiment 7 is a 16-processor simulation with four processor cores doing intense read and write operation to the exact same memory address. There's another four cores doing read mixed with write operation to the same address. Four of the remaining cores solely conducts read operation and the last four does read and write operation to similar yet non-conflicting memory addresses.

The intense intermittent r/w operation would cause both Exclusive and Owned state to underperform, since they would quickly become invalid. A combination of both is the better solution and **MOESIF** again outperforms everyone.

Table 7: Experiment 7 Result

	MSI	MOSI	MESI	MOESIF
Runtime	6459	5359	3993	2909
Cache Miss	79	79	55	55
Cache Access	952	952	952	952
Silent Upgrade	0	0	24	24
$-to-$ Transfer	17	28	17	28

Experiment 8

The result for each configuration is shown in Table 8. Experiment 8 is a 16-processor simulation with mixed operations. Having both Owned state and Exclusive State helps reducing the number of queries to the memory, but having both, **MOESIF** is definitely the best in terms of runtime.

Table 8: Experiment 8 Result

	MSI	MOSI	MESI	MOESIF
Runtime	9477	8477	7139	4939
Cache Miss	110	110	91	91
Cache Access	800	800	800	800
Silent Upgrade	0	0	19	19
$-to-$ Transfer	18	28	22	44

System Architecture Experiment

To architect a system where all the provided programs were equally important, I would choose **MOESIF** as the protocol in the system. This is because MOESIF yields the smallest runtime across all processor count and traces. It also outperforms the second place by a non-negligible margin. There is simply no reason to not choose it as the protocol for the system.

Limitation to the Simulator

The cache coherence simulator made certain assumptions that are unlikely in real world and may affect actual performance. There is a rather low possibility for a real world computer to have a fully associative and **infinite** size L1 cache. Having no eviction renders certain protocol less effective than they actually are. To equip the simulator with cache eviction would increase the number of cache share requests and is likely to cause performance change. Furthermore, there is usually than one level of cache in CPU cores, adding this hiearchy of cache can make this even more realistic.

The simulator limits the number of memory requests onto the bus. This limits the number of load/store FU to be only one, which is probably not the case in superscalar processor model.

Furthermore, the traces appears to have causality between different cores that is not modeled by this simulator. Experiment 2 p0 and p1 read and write along the same cache addresses, which could imply the cores are doing vectorized array access or queue access. In some of those situations, read write operation order across different cores need to be maintained and this simulator does not seem to be modeling this causality. Having the order modeled can cause difference in performance and is considered a limitation to this simulator.