Memory Communication (Evaluation)

CSCI 389: Computer Systems

Fall 2023

This assignment is designed to evaluate your understanding of the memory communication. Feel free to collaborate with others and use resources, but all code and writeups must be your own.

Due Date: Friday, October 27th at 11:00 am.

1. (4 points) **MESI Protocol States.**
   (a) (2 points) Explain what issues would arise if the Shared state was removed from the MESI protocol.

   Eliminating the Shared state from the cache protocol would result in multiple detrimental effects. Concurrent reads by various processors would encounter conflicts, thereby diminishing system parallelism and throughput. This change would also bring ambiguity to write-back procedures, making it challenging to determine modifications in data across different caches. Caches would become unsure of which among them possesses valid copies, escalating the frequency of broadcasted invalidations. Additionally, persistent cache invalidations among processors reading shared data would amplify bus traffic, consequently reducing system scalability. Processors would also face an increase in cache misses for each new read, compelling them to consistently retrieve data from the main memory. This frequent invalidation and fetching of cache lines would not only misuse memory bus bandwidth, potentially slowing down the system but would also surge power consumption in the cache subsystem and the memory bus. Furthermore, the absence of a Shared state might set the stage for deadlock scenarios, where processors indefinitely wait for exclusive access to a cache line.

   (b) (2 points) Explain what issues would arise if the Modified state was removed from the MESI protocol.

   Removing the Modified state from caching protocols would jeopardize cache coherence, as it becomes challenging to track which cache lines have been modified. This change would increase write latencies because of more frequent main memory accesses and pose dilemmas in choosing between write-through and write-back caching strategies. The absence of the Modified state would blur the distinction between cache lines that mirror the main memory and those altered by a processor. It could also escalate the likelihood of deadlocks due to cache line conflicts, amplify cache misses affecting system performance, and elevate power consumption from repeated main memory interactions. Furthermore, every processor's write action might mandate an immediate update to the main memory, heightening memory traffic, and eliminating the option for optimized direct cache-to-cache data transfers, thus introducing redundant bus activities.

2. (9 points) **Coherence Protocols.** Consider the following sequence of requests in a system containing two processors. Assume that neither variable is in any cache at the start of the sequence.

P0	P1
Read X	
Read Y	
	Read Y
Write X	
	Write Y
	Read X
Read Y	

If the system uses the MESI cache coherence protocol, show how the state transitions that the variables X and Y experience in processor P0 as well as the signals sent (if any) by that processor for each state transition.

Operation	**State (Before)**	**Signal Sent**	**State (After)**
Read X	I	BusRd	E
Read Y	I	BusRd	E
P1: Read Y	E	-	S
Write X	E	-	M
P1: Write Y	S	-	I
P1: Read X	M	BusRd/Flush	S
Read Y	I	BusRd	E

3. (12 points) **Consistency** Consider the following threads. Assume that all variables are set to zero before the threads begin.
Note: There is always the possibility that a variable never print. However, I didn't find this to ever be the case, so I will consider "0" and "1" to be the only possible values.

	Thread 1	Thread 2	Thread 3
L1	W = 1	X = 1	while (Y == 0) {}
L2	while (X == 0) {}	Z = 1	print X
L3	Y = 1	print Y	print W
L4	print Z		

(a) (4 points) Show the possible combinations of the values of the variables at the time they are printed if the system uses sequential consistency. For each combination of values, you should specify whether or not it is possible to observe that combination printed.

Here, I only found 3 possibilities.

W	X	Y	Z	Possible Output
0	0	0	0	No
0	0	0	1	No
0	0	1	0	No
0	0	1	1	No
0	1	0	0	No
0	1	0	1	No
0	1	1	0	No
0	1	1	1	No
1	0	0	0	No
1	0	0	1	No
1	0	1	0	No
1	0	1	1	No
1	1	0	0	No
1	1	0	1	Yes
1	1	1	0	Yes
1	1	1	1	Yes

(b) (4 points) Show the possible combinations of the values of the variables at the time they are printed if the system uses total store order. For each combination of values, you should specify whether or not it is possible to observe that combination printed.

Here, I only found the addition of 1100.

W	X	Y	Z	Possible Output
0	0	0	0	No
0	0	0	1	No
0	0	1	0	No
0	0	1	1	No
0	1	0	0	No
0	1	0	1	No
0	1	1	0	No
0	1	1	1	No
1	0	0	0	No
1	0	0	1	No
1	0	1	0	No
1	0	1	1	No
1	1	0	0	Yes
1	1	0	1	Yes
1	1	1	0	Yes
1	1	1	1	Yes

(c) (4 points) Show the possible combinations of the values of the variables at the time they are printed if the system uses processor consistency. For each combination of values, you should specify whether or not it is possible to observe that combination printed.

Here, I found four additional possibilities.

W	X	Y	Z	Possible Output
0	0	0	0	No
0	0	0	1	No
0	0	1	0	No
0	0	1	1	No
0	1	0	0	No
0	1	0	1	No
0	1	1	0	No
0	1	1	1	No
1	0	0	0	Yes
1	0	0	1	Yes
1	0	1	0	Yes
1	0	1	1	Yes
1	1	0	0	Yes
1	1	0	1	Yes
1	1	1	0	Yes
1	1	1	1	Yes