Weekly Assignment 4 - Report

Francesco Done'
qwg586@alumni.ku.dk
Erasmus student

October 8, 2020

Contents

1 Task 1

Assume that a shared-memory multiprocessor with a number of processor/private cache units connected by a shared single(atomic)-transaction bus. Our baseline cache-coherence protocol is an MSI protocol, but we want to investigate what performance gains can be achieved by using an exclusive state to make it an MESI protocol. The time to carry out various protocol actions is listed in the above table: for example, while a read and write hit takes only 1 cycle, a read request takes 40 cycles as it has to bring the block from the next level of the cache hierarchy. A bus upgrade request takes less time as it does not involve a memory-block transfer but rather it only invalidates other shared copies. We assume that a write-hit in the cache requires a Bus Upgrade (rather than read-exclusive) transaction because the memory-block copy is already valid in the cache (and the memory-block does not need to be transferred).
We write `Ri/B` *and* `Wi/B` *to denote a read and write operation, respectively, by processor/cache unit* `i` *to block* `B`*. Assuming block X is not in any of the private caches of processors 1 to 4, we want to determine the time it takes to execute the following sequence of accesses:* `R1/X, W1/X, W1/X, R2/X, W2/X, W2/X, R3/X, W3/X, W3/X, R4/X, W4/X, W4/X`

Request Type	**Time to carry out protocol action**	**Traffic**
Read hit	1 Cycle	N/A
Write hit	1 Cycle	N/A
Read request serviced by next level	40 Cycles	6 Bytes + B
Read-exclusive request serviced by next level	40 Cycles	6 Bytes + B
Bus Upgrade/update request	10 Cycles	10 Bytes

Table 1: Time and traffic per request type

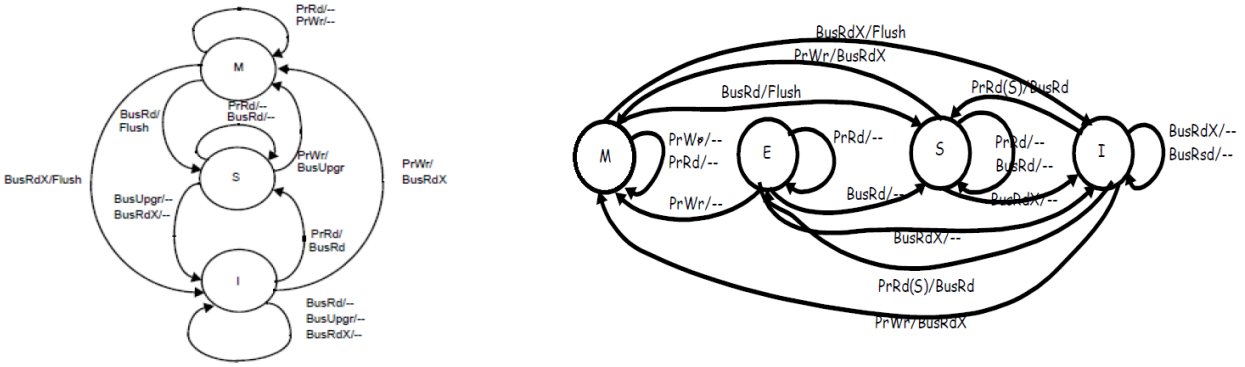

Figure 1: Overview: MSI vs MESI

1.1 Task 1.a

Assume that a transaction from state E to state M brings no access cost. How many cycles does it take to execute the access sequence under MSI vs. MESI?

1.1.1 Results

Access	# cycles MSI	# cycles MESI
`R1/X`	40	40
`W1/X`	10+1	1
`W1/X`	1	1
`R2/X`	40	40
`W2/X`	10+1	10+1
`W2/X`	1	1
`R3/X`	40	40
`W3/X`	10+1	10+1
`W3/X`	1	1
`R4/X`	40	40
`W4/X`	10+1	10+1
`W4/X`	1	1
TOTAL	**208**	**198**

Table 2: Number of cycles taken while executing the access sequence under MSI and MESI

MESI performs 10 cycles less than MSI.

1.1.2 Explaination

- **MSI**:
  - `R1/X`: P1 I→S, bus read request = 40 cycles
  - `W1/X`: P1 S→M, bus upgrade = 10 cycles + 1 cycle because of write hit
  - `W1/X`: P1 M↻, write hit = 1 cycle
  - `R2/X`: P2 I→S and P1 M→S, bus read request = 40 cycles
  - `W2/X`: P2 S→M and P1 S→I, bus upgrade = 10 cycles + 1 cycle because of write hit
  - `W2/X`: P2 M↻, write hit = 1 cycle
  - `R3/X`: P3 I→S and P2 M→S, bus read request = 40 cycles
  - `W3/X`: P3 S→M and P2 S→I, bus upgrade = 10 cycles + 1 cycle because of write hit
  - `W3/X`: P3 M↻, write hit = 1 cycle
  - `R4/X`: P4 I→S and P3 M→S, bus read request = 40 cycles
  - `W4/X`: P4 S→M and P3 S→I, bus upgrade = 10 cycles + 1 cycle because of write hit
  - `W4/X`: P4 M↻, write hit = 1 cycle
- **MESI**:
  - `R1/X`: P1 I→E, bus read request = 40 cycles
  - `W1/X`: P1 E→M, write hit = 1 cycle
  - `W1/X`: P1 M↻, write hit = 1 cycle
  - `R2/X`: P2 I→S and P1 M→S, bus read request = 40 cycles
  - `W2/X`: P2 S→M and P1 S→I, bus upgrade = 10 cycles + 1 cycle because of write hit

- `W2/X`: P2 M↻, write hit = 1 cycle
- `R3/X`: P3 I→S and P2 M→S, bus read request = 40 cycles
- `W3/X`: P3 S→M and P2 S→I, bus upgrade = 10 cycles + 1 cycle because of write hit
- `W3/X`: P3 M↻, write hit = 1 cycle
- `R4/X`: P4 I→S and P3 M→S, bus read request = 40 cycles
- `W4/X`: P4 S→M and P3 S→I, bus upgrade = 10 cycles + 1 cycle because of write hit
- `W4/X`: P4 M↻, write hit = 1 cycle

1.2 Task 1.b

Compare the traffic generated by the MSI and MESI protocols counted in bytes transferred, and assuming that the size of a memory block B is 32 bytes.

1.2.1 Results

Access	Bytes transferred MSI	Bytes transferred MESI
`R1/X`	6+32	6+32
`W1/X`	10	0
`W1/X`	0	0
`R2/X`	6+32	6+32
`W2/X`	10	10
`W2/X`	0	0
`R3/X`	6+32	6+32
`W3/X`	10	10
`W3/X`	0	0
`R4/X`	6+32	6+32
`W4/X`	10	10
`W4/X`	0	0
TOTAL	**192**	**182**

Table 3: Number of Bytes transferred while executing the access sequence under MSI and MESI

MESI transfers 10 Bytes less than MSI.

1.2.2 Explaination

- **MSI**:
  - `R1/X`: bus read request = 6+32 Bytes
  - `W1/X`: bus upgrade = 10 Bytes
  - `W1/X`: write hit = N/A
  - `R2/X`: bus read request = 6+32 Bytes
  - `W2/X`: bus upgrade = 10 Bytes
  - `W2/X`: write hit = N/A

  - `R3/X`: bus read request = 6+32 Bytes
  - `W3/X`: bus upgrade = 10 Bytes
  - `W3/X`: write hit = N/A
  - `R4/X`: bus read request = 6+32 Bytes
  - `W4/X`: bus upgrade = 10 Bytes
  - `W4/X`: write hit = N/A

- **MESI**:
  - `R1/X`: bus read request = 6+32 Bytes
  - `W1/X`: write hit = N/A
  - `W1/X`: write hit = N/A
  - `R2/X`: bus read request = 6+32 Bytes
  - `W2/X`: bus upgrade = 10 Bytes
  - `W2/X`: write hit = N/A
  - `R3/X`: bus read request = 6+32 Bytes
  - `W3/X`: bus upgrade = 10 Bytes
  - `W3/X`: write hit = N/A
  - `R4/X`: bus read request = 6+32 Bytes
  - `W4/X`: bus upgrade = 10 Bytes
  - `W4/X`: write hit = N/A

2 Task 2

Consider a shared-memory multiprocessor that consists of three processor/cache units and where cache coherence is maintained by an MSI protocol. The private caches are direct-mapped. The table below shows the access sequence taken by the three processors to four variables (A,B,C, and D), where A, B, and C belong to the same block and D belongs to a different block. The two blocks map to the same entry in the caches, and the cache is full initially and A, B, C, and D have not been accessed yet in the program.

Time	**Processor 1**	**Processor 2**	**Processor 3**
1	Read A		
2		Read B	
3			Read C
4	Write A		
5			Read D
6		Read B	
7	Write B		
8			Read C
9		Read B	

Table 4: Access sequence taken by the three processors to four variables

2.1 Task 2.a

Classify the misses with respect to cold, replacement, true sharing, and false sharing misses.

Time	**Processor 1**	**Processor 2**	**Processor 3**	**Miss type**
1	Read A			Cold
2		Read B		Cold
3			Read C	Cold
4	Write A			
5			Read D	Cold
6		Read B		False sharing
7	Write B			
8			Read C	Replacement
9		Read B		True sharing

Table 5: Misses classification

2.2 Task 2.b

Which of the misses could be ignored and still guarantee that the execution is correct?

The **false sharing** miss at time 6 could be ignored.

3 Task 3

We consider a scalable implementation of a shared-memory multiprocessor using a set of nodes that each contain a processor, a private cache, and a portion of the memory. Cache coherence is maintained using a directory cache protocol, where the directory uses a presence-flag vector associated with each memory block to keep track of which nodes have copies of that block. The time it takes to process a directory request at the home and a remote node is in both cases 50 cycles. (Also the time to lookup or install into the cache at remote/host/local node is also 50 cycles). The latency and traffic of all consistency-induced requests and responses are detailed in the table above, where the block size B is 32 bytes. Answer the following questions for each of the two MSI coherence protocols, i.e., the classical one that works in four hops and the optimized one (Standard DASH) that works in only three hops:

Request Type	Time to carry out protocol action	Traffic
Read hit	1 cycle	N/A
Write hit	1 cycle	N/A
BusRd	20 cycles	6 bytes
RemRd	20 cycles	6 bytes
RdAck	40 cycles	6 bytes
Flush	100 cycles	6 bytes + B
InvRq	20 cycles	6 bytes
InvAck	20 cycles	6 bytes
UpgrAck	20 cycles	6 bytes

Table 6: Time and traffic performed for each request type

3.1 Task 3.a

Determine the number of cycles needed to handle a read-cache miss when the home node is the same as the requesting (local) node and the memory copy is clean. Also determine the amount of traffic (in bytes) caused by the coherence transaction (across nodes).
$L = H$
$Dirty\ bit = 0$
$Cost = 50{+}1 = \mathbf{51}$
$Traffic = \mathbf{0}$

3.2 Task 3.b

Determine the number of cycles needed to handle a read-cache miss when the home node is the same as the requesting (local) node and the memory copy is dirty. Also determine the amount of traffic (in bytes) caused by the coherence transaction.
$L = H$
$Dirty\ bit = 1$
$Cost = 1{+}50{+}20{+}50{+}100{+}50 = \mathbf{271}$
$Traffic = 6{+}6{+}32 = \mathbf{44}$

3.3 Task 3.c

Determine the number of cycles needed to handle a read-cache miss when the home node is different from the requesting (local) node and the memory copy is clean. Also determine the amount of traffic (in bytes) caused by the coherence transaction.
$L \neq H$
$Dirty\ bit = 0$
$Cost = 1{+}20{+}50{+}100{+}50 = \mathbf{221}$
$Traffic = 6{+}6{+}32 = \mathbf{44}$

3.4 Task 3.d

Determine the number of cycles needed to handle a read-cache miss when the home node is not the same as the requesting (local) node, the memory copy is dirty, and the remote node is the same as the home node. Also determine the amount of traffic (in bytes) caused by the coherence transaction.
$L \neq H = D$
Dirty bit = 1
$Cost$ = 1+20+50+100+50 = **221**
$Traffic$ = 6+6+32 = **44**

3.5 Task 3.e

Determine the number of cycles needed to handle a read-cache miss when the home node is different from the requesting (local) node, the memory copy is dirty, and the remote node is different from the home node (and of course different from the requesting/local node). Also determine the amount of traffic (in bytes) caused by the coherence transaction.
$L \neq H \neq D$
Dirty bit = 1

- MSI
  $Cost$ = 1+20+50+20+50+100+50+100+50 = **441**
  $Traffic$ = 6+6+6+6+32+32 = **88**

- DASH
  $Cost$ = 1+20+50+20+50+100+150 = **391**
  $Traffic$ = 6+6+6+32+6+32 = **88**

4 Task 4

We Consider a 16-by-16 torus (tori) interconnection network, in which each link has a bandwidth of 100 Mbits/s. Determine the following interconnection network properties:

4.1 Task 4.a

Network diameter
for a n-by-n tori, the diameter is equal to n, in our case $(8x + 8y)$ = **16**.

4.2 Task 4.b

*Bisection bandwidth = bisection width * link bandwidth*
bisection bandwidth = $2n$ * 100 Mbits/s = 3200 Mbits/s = **3.2 Gbits/s**

4.3 Task 4.c

Bandwidth per node = total bandwidth / number of nodes
Network size = n^2 = 256

total bandwidth = total number of links * bandwidth = 512 * 100 Mbits/s = 51200 Mbits/s
bandwidth per node = 51200 Mbits/s / 256 = **200 Mbits/s**

5 Task 5

A design team is contemplating whether to use a n-by-n torus (tori) or an ndimensional hypercube to build an interconnect network that connects 4, 16, 64, and 256 nodes. In this exercise you will compare the network diameter and the bisection width for the two topologies to understand the trade-offs between them:

N	**n-by-n tori** n^2	**k-dim hypercube** 2^k
4	2^2	2^2
16	4^2	2^4
64	8^2	2^6
256	16^2	2^8

Table 7: Values for different nodes

5.1 Task 5.a

At what scale does the hypercube provide (strictly) higher bisection width than the torus?

N	**n-by-n tori** $2n$	**k-dim hypercube** 2^{k-1}
4	4	2
16	8	8
64	16	32
256	32	128

Table 8: Bisection

Hypercube provides (strictly) higher bisection from **N=64**.

5.2 Task 5.b

Determine the network diameter and the switch degree for the scale at which the hypercube provides a higher bisection width than the torus.

N	**n-by-n tori** n	**k-dim hypercube** k
4	2	2
16	4	4
64	8	6
256	16	8

Table 9: Diameter

N	**n-by-n tori** 4	**k-dim hypercube** k
4	4	2
16	4	4
64	4	6
256	4	8

Table 10: Switch degree

Having $N = 64$ means that the diameter of the Cube is **lower** than the diameter of the Toru, by the way from the same N we have that the Switch degree is **higher** in the Cube

5.3 Task 5.c

What can you say about the relative merits of the two topologies?

- *k-dim hypercube*: Having a higher bisection width implies that there is more parallellism and more bandwisth through many links.
- *n-by-n tori*: A lower switch degree permits more nodes without increasing the complexity, thus using less resources.