INDIAN INSTITUTE OF TECHNOLOGY KANPUR

CS 622A Semester 2020–2021-I

Assignment 3

GROUP-17

Members
Mayank Bansal
Roll-20111032

Hirak Mondal
Roll-20111022

Professor:
Dr. Mainak Chaudhury
Department of Computer Science

INTRODUCTION

In this assignment we have modeled an array of L1 private caches and an array of shared L2 cache banks. The number of L1 caches is equal to the number of L2 cache banks. The L2 cache bank id is derived by taking the least significant few bits of the L2 cache set index. For example, if a 2 MB 16-way L2 cache with 64-byte blocks has eight banks, the bank id is obtained from the three bits right after the block offset. The modeled L2 cache is inclusive. Each L1 cache has two unbounded input queues. One queue accepts memory operations from a machine access trace. The other queue accepts messages from the L2 cache banks and other L1 caches. Both queues are unbounded to avoid deadlocks arising from shortage of queue space and therefore, can be implemented using a linked list. In fact, the queue that inputs memory operations from an access trace can be thought of as a trace file. Each L2 cache bank has one input queue accepting messages from L1 caches.Neither the L1 caches nor the L2 cache banks have any outgoing queues. This is to simplify the simulator. Whenever an L1 cache needs to send a message to an L2 cache bank, it computes the bank id and directly enqueues the message at the tail of the input queue of the destination L2 cache bank. Similarly, if an L1 cache needs to send a message to another L1 cache, it directly enqueues the message at the tail of the input queue of the destination L1 cache. Also, if an L2 cache bank needs to send a message to an L1 cache, it directly enqueues the message at the tail of the input queue of the destination L1 cache.

This simulator models the behavior of a directory-based cache coherence protocol.We have started with the solution for the second assignment and generated a thread-wise machine access trace files from the four programs you used in the second assignment. The additional information that we have added in trace is the distinction between a load and a store operation. The protocol that we have followed in this assignment is the MESI protocol.

We have collected the result for the following system specification.

• Number of cores: 8

• Cache coherence protocol: directory-based MESI.

• L1 cache: 32 KB, 8-way, 64-byte block size, LRU (S state blocks are replaced silently from L1 caches), 8 in number

• L2 cache: 4 MB, 16-way, 64-byte block size, LRU, 8 banks (each bank is 512 KB), inclusive

General Instructions for Compiling and Running the Codes

Step 1: Download Pin 3.15 and set up.

Step 2: Paste the Directory named **CS622** that is being provided by us, under "pin-3.15-98253-gb56e429b1-gcc-linux/source/tools/"

Step 3: Set the path of the terminal to this folder using the **cd** command. If your "pin-3.15-98253-gb56e429b1-gcc-linux"file is at the HOME directory then you can use the command →
cd pin-3.15-98253-gb56e429b1-gcc-linux/source/tools/CS622

Compilation and Runtime Instruction To Calculate the Trace

• A file named **trace.cpp** is provided inside the directory named **"CS622"** to record the total number of machine accesses along with its nature whether read/write as well as the thread number.

• Compile the program whose machine access you want to compute. The command for that is → **gcc -O3 -static -pthread filename.c -o filename.**

• Then enter the command → **make obj-intel64/trace.so**

• Then finally run it using →

../../../pin -t obj-intel64/trace.so – ./filename number_of_threads

• To know the line count, we can use → **wc -l trace_filename.out**

Output :It outputs a file named **trace.out** which contains the **thread_id**, **Read or Write information** and **Address of Machine Accesses** and **wc -l trace.out** gives us the total number of machine accesses.

We have also attached a **Global Count** to each machine access irrespective of thread id.

Compilation and Runtime Instruction For the Simulator

• A C++ file named **simulator.cpp** is provided. Paste it in the directory which contains the trace file which we have generated using the pintool. The simulator program assumes the name of the trace file is **trace.out**. If the name is something else, then one must change it editing the C++ code and change the file name under the function **FILE *fp = fopen("filename.out", "r");** which is present inside the main function

• Compile the code using **g++ simulator.cpp**

• Run the code using **./a.out**

Output :It outputs total number of machine cycles, total L1 accesses, total number of machine accesses, total miss in L1 and L2 cache, total invalidation requests and acknowledgement received by the L1 cache, total GET, GETX, PUT, PUTX messages received by the respective private L1 caches, total GET and GETX message,Upgrade request, Upgrade Acknowledgement,Writeback messages, Sharing writeback messages received by the L2 cache banks.

Names and Explanation of the Messages

Messages received by the L1 cache

I. Invalidate Request

This is a message received by private L1 caches to invalidate a block that is present inside them. This message is received by a private L1 cache when another processor wants to write some data in a block which is already present in the private cache of this processor. So the block (and in turn the data) becomes invalid.

The invalidation message is sent by a L2 cache bank (the home bank of the block).

II. Invalidation Acknowledgement

This is the acknowledgement in response to the invalidation request that the L2 cache bank sent to some processor's (say P1's) private L1 cache (upon receiving some specific message like a GETX message from some other processor's (say P0's) private L1 cache. Note that this L1 cache of P0's is the one that receives the invalidation acknowledgement).

III. GET Message

A L1 cache receives a GET message when it has a block in the modified (M) state and another processor wants to read the block. So, the L2 cache bank forwards a GET request to this L1 cache (request originally sent by the L1 cache of the processor that wants to read the aforementioned block) so that the L1 cache can send the updated copy back to the requester directly.

IV. GETX Message

A L1 cache receives a GETX message when it has a block in the modified (M) state and another processor wants to write to the block. So, the L2 cache bank forwards a GETX request to this L1 cache (request originally sent by the L1 cache of the

processor that wants to write the aforementioned block) so that the L1 cache can send the updated copy back to the requester directly.

V. PUTE and PUT Message

PUTE and PUT are the replies that a processor receives in response to their GET request.

A L1 cache receives the PUTE message when a block is placed inside the cache in exclusive (E) state, and the PUT message when a block is placed inside the cache in shared (S) state.

VI. PUTX Message

PUTX is the reply that a L1 cache receives in response to its GETX request.

VII. Upgrade Acknowledgement Message

Upgrade requests are send to a L2 cache bank when a L1 cache has a block in shared (S) state but it wants to change or upgrade the block to modify (M) state. Thus the L1 cache sends an Upgrade message to the L2 cache bank (i.e. to the home bank of the block) and the L2 cache bank replies back with an Upgrade Acknowledgement message to the L1 cache indicating that the upgrade request has been acknowledged by the L2 cache bank.

Messages received by the L2 cache banks

I. GET Message

A L2 cache bank receives a GET message when a processor requests to read a block from that L2 cache bank (i.e. the home bank of the block).

II. GETX Message

A L2 cache bank receives a GETX message when a processor requests for a block from that L2 cache bank (i.e. the home bank of the block) on which it wants to perform the Write operation.

III. Sharing Write-Back(SWB) Message

When a processor has a block in modified (M) state which still has not been written back to the L2 cache bank (the home bank of the block) and another processor wants to access it (either to read or write in the block), then the processor which has the block in modified (M) state writes back the block in the L2 cache bank (and thus updates the block in the L2 cache bank) and supplies the block as well to the processor which needs it. This writing back of the modified block in L2 cache bank when it is requested by another processor is termed as Sharing Write-Back(SWB).

IV. UPGRADE Message

Upgrade requests are send to a L2 cache bank when a L1 cache has a block in shared (S) state but it wants to change or upgrade the block to modify (M) state. So it sends an Upgrade message to the L2 cache bank (i.e. the home bank of the block).

V. Write-back Message

When a block is in modified (M) state in some L1 cache and then it is evicted from the cache, then the L2 cache bank (i.e. the home bank of the block) receives a write-back message (from the L1 cache) which tells the L2 cache bank to update the modified block in itself.

Results for Program 1

- Number of Simulated Cycles → 139555415
- Number of L1 misses →

Processor	Misses
Processor 0	1137145
Processor 1	714929
Processor 2	748334
Processor 3	905802
Processor 4	809033
Processor 5	844702
Processor 6	877884
Processor 7	780411

- Number of L1 access →

Processor	Number of L1 Access
Processor 0	37765122
Processor 1	14680208
Processor 2	14680198
Processor 3	14680248
Processor 4	14680228
Processor 5	14680228
Processor 6	14680198
Processor 7	14680248

Total Number of L1 access → 140526678

- Number of L2 cache misses → 3407806

- **Names and count of messages received by L1 cache**

I. Invalidate Requests

Processor	Number of Messages
Processor 0	2976
Processor 1	2406
Processor 2	2582
Processor 3	6230
Processor 4	3934
Processor 5	4909
Processor 6	1603
Processor 7	5623

So, Total Invalidation requests received→ 30263

II. Invalidation Acknowledgements

Processor	Number of Messages
Processor 0	2193
Processor 1	2787
Processor 2	2820
Processor 3	5229
Processor 4	6305
Processor 5	3729
Processor 6	3314
Processor 7	3886

So, Total Invalidation Acknowledgement received→ 30263

III. GET Messages

Processor	Number of Messages
Processor 0	2424
Processor 1	974
Processor 2	1424
Processor 3	3329
Processor 4	2556
Processor 5	2251
Processor 6	889
Processor 7	2178

So, Total Get Message received→ 16025

IV. GETX Messages

Processor	Number of Messages
Processor 0	14
Processor 1	0
Processor 2	1
Processor 3	0
Processor 4	1
Processor 5	0
Processor 6	0
Processor 7	0

So, Total GETX Message received→ 16

V. PUT and PUT-E Message Count

Processor	Number of Messages
Processor 0	406885
Processor 1	618699
Processor 2	635792
Processor 3	714629
Processor 4	666637
Processor 5	684176
Processor 6	700961
Processor 7	651682

So, Total PUT and PUT-E Message received→ 5079461

VI. PUTX Messages

Processor	Misses
Processor 0	730171
Processor 1	96188
Processor 2	112574
Processor 3	191135
Processor 4	142561
Processor 5	160475
Processor 6	176978
Processor 7	128694

So, Total PUTX Message received→ 1738776

VII. Upgrade Acknowledgement

Processor	Misses
Processor 0	2903
Processor 1	4128
Processor 2	3662
Processor 3	5347
Processor 4	5912
Processor 5	4801
Processor 6	4396
Processor 7	5383

So, Total Upgrade Acknowledgement received→ 36532

• **Names and count of messages received by L2 cache**

I. Total GET Messages → 5079461

II. Total GETX Messages → 1738776

III.Total Sharing Writeback Messages (SWB) → 16041

IV. Total UPGRADE Messages → 36532

V. Total Writeback Messages → 6112405

Results for Program 2

- Number of Simulated Cycles → 2616950
- Number of L1 misses →

Processor	Misses
Processor 0	78132
Processor 1	24515
Processor 2	24504
Processor 3	24504
Processor 4	24505
Processor 5	24037
Processor 6	24501
Processor 7	15863

- Number of L1 access →

Processor	Number of L1 Access
Processor 0	737187
Processor 1	284174
Processor 2	273433
Processor 3	300486
Processor 4	262289
Processor 5	301645
Processor 6	262289
Processor 7	198561

Total Number of L1 access → 2620064

- Number of L2 cache misses → 122300

- **Names and count of messages received by L1 cache**

I. Invalidate Requests

Processor	Number of Messages
Processor 0	35
Processor 1	10
Processor 2	6
Processor 3	7
Processor 4	4
Processor 5	7
Processor 6	4
Processor 7	6

So, Total Invalidation requests received→ 79

II. Invalidation Acknowledgements

Processor	Number of Messages
Processor 0	19
Processor 1	7
Processor 2	8
Processor 3	8
Processor 4	10
Processor 5	9
Processor 6	9
Processor 7	9

So, Total Invalidation Acknowledgement received→ 79

III. GET Messages

Processor	Number of Messages
Processor 0	32
Processor 1	3
Processor 2	4
Processor 3	3
Processor 4	4
Processor 5	2
Processor 6	2
Processor 7	2

So, Total Get Message received→ 52

IV. GETX Messages

Processor	Number of Messages
Processor 0	14
Processor 1	0
Processor 2	0
Processor 3	0
Processor 4	0
Processor 5	0
Processor 6	1
Processor 7	0

So, Total GETX Message received→ 15

V. PUT and PUT-E Message Count

Processor	Number of Messages
Processor 0	16541
Processor 1	24506
Processor 2	24498
Processor 3	24495
Processor 4	24497
Processor 5	24028
Processor 6	24495
Processor 7	15855

So, Total PUT and PUT-E Message received→ 178915

VI. PUTX Messages

Processor	Misses
Processor 0	61591
Processor 1	8
Processor 2	8
Processor 3	8
Processor 4	9
Processor 5	7
Processor 6	7
Processor 7	7

So, Total PUTX Message received→ 61645

VII. Upgrade Acknowledgement

Processor	Misses
Processor 0	56
Processor 1	21
Processor 2	33
Processor 3	42
Processor 4	27
Processor 5	31
Processor 6	31
Processor 7	49

So, Total Upgrade Acknowledgement received→ 290

• **Names and count of messages received by L2 cache**

I. Total GET Messages → 178915

II. Total GETX Messages → 61645

III.Total Sharing Writeback Messages (SWB) → 67

IV. Total UPGRADE Messages → 290

V. Total Writeback Messages → 133128

Results for Program 3

- Number of Simulated Cycles → 9896185
- Number of L1 misses →

Processor	Misses
Processor 0	192387
Processor 1	65313
Processor 2	65316
Processor 3	65310
Processor 4	65320
Processor 5	65304
Processor 6	65313
Processor 7	65316

- Number of L1 access →

Processor	Number of L1 Access
Processor 0	2151732
Processor 1	1079972
Processor 2	1108735
Processor 3	1098137
Processor 4	1158009
Processor 5	1111644
Processor 6	1103691
Processor 7	1094979

Total Number of L1 access → 9906899

- Number of L2 cache misses → 382063

- **Names and count of messages received by L1 cache**

I. Invalidate Requests

Processor	Number of Messages
Processor 0	567
Processor 1	526
Processor 2	18
Processor 3	11
Processor 4	25
Processor 5	498
Processor 6	24
Processor 7	527

So, Total Invalidation requests received→ 2196

II. Invalidation Acknowledgements

Processor	Number of Messages
Processor 0	541
Processor 1	23
Processor 2	24
Processor 3	20
Processor 4	508
Processor 5	20
Processor 6	526
Processor 7	534

So, Total Invalidation Acknowledgement received→ 2196

III. GET Messages

Processor	Number of Messages
Processor 0	542
Processor 1	2
Processor 2	2
Processor 3	3
Processor 4	3
Processor 5	2
Processor 6	2
Processor 7	2

So, Total Get Message received→ 558

IV. GETX Messages

Processor	Number of Messages
Processor 0	14
Processor 1	0
Processor 2	0
Processor 3	0
Processor 4	0
Processor 5	0
Processor 6	0
Processor 7	0

So, Total GETX Message received→ 14

V. PUT and PUT-E Message Count

Processor	Number of Messages
Processor 0	130790
Processor 1	65312
Processor 2	65304
Processor 3	65308
Processor 4	65315
Processor 5	65295
Processor 6	65307
Processor 7	65309

So, Total PUT Message received→ 587940

VI. PUTX Messages

Processor	Misses
Processor 0	61598
Processor 1	6
Processor 2	5
Processor 3	6
Processor 4	6
Processor 5	5
Processor 6	6
Processor 7	6

So, Total PUTX Message received→ 61638

VII. Upgrade Acknowledgement

Processor	Misses
Processor 0	308
Processor 1	269
Processor 2	267
Processor 3	250
Processor 4	754
Processor 5	262
Processor 6	263
Processor 7	317

So, Total Upgrade Acknowledgement received→ 2690

• **Names and count of messages received by L2 cache**

I. Total GET Messages → 587940

II. Total GETX Messages → 61638

III.Total Sharing Writeback Messages (SWB) → 572

IV. Total UPGRADE Messages → 2690

V. Total Writeback Messages → 617182

Results for Program 4

- Number of Simulated Cycles → 1064481
- Number of L1 misses →

Processor	Misses
Processor 0	69963
Processor 1	8186
Processor 2	8186
Processor 3	8183
Processor 4	8185
Processor 5	8183
Processor 6	8184
Processor 7	8186

- Number of L1 access →

Processor	Number of L1 Access
Processor 0	605091
Processor 1	65670
Processor 2	65670
Processor 3	65670
Processor 4	65670
Processor 5	65678
Processor 6	65670
Processor 7	65670

Total Number of L1 access → 1064789

- Number of L2 cache misses → 65913

- **Names and count of messages received by L1 cache**

I. Invalidate Requests

Processor	Number of Messages
Processor 0	36
Processor 1	5
Processor 2	5
Processor 3	6
Processor 4	9
Processor 5	9
Processor 6	7
Processor 7	6

So, Total Invalidation requests received→ 83

II. Invalidation Acknowledgements

Processor	Number of Messages
Processor 0	28
Processor 1	5
Processor 2	9
Processor 3	8
Processor 4	10
Processor 5	9
Processor 6	6
Processor 7	8

So, Total Invalidation Acknowledgement received→ 83

III. GET Messages

Processor	Number of Messages
Processor 0	40
Processor 1	3
Processor 2	2
Processor 3	3
Processor 4	2
Processor 5	2
Processor 6	2
Processor 7	2

So, Total Get Message received→ 56

IV. GETX Messages

Processor	Number of Messages
Processor 0	14
Processor 1	1
Processor 2	1
Processor 3	1
Processor 4	1
Processor 5	1
Processor 6	1
Processor 7	3

So, Total GETX Message received→ 23

V. PUT and PUT-E Message Count

Processor	Number of Messages
Processor 0	8372
Processor 1	8178
Processor 2	8178
Processor 3	8175
Processor 4	8178
Processor 5	8177
Processor 6	8177
Processor 7	8178

So, Total PUT Message received→ 65613

VI. PUTX Messages

Processor	Misses
Processor 0	61591
Processor 1	9
Processor 2	8
Processor 3	6
Processor 4	7
Processor 5	6
Processor 6	7
Processor 7	7

So, Total PUTX Message received→ 61641

VII. Upgrade Acknowledgement

Processor	Misses
Processor 0	29
Processor 1	4
Processor 2	4
Processor 3	4
Processor 4	7
Processor 5	7
Processor 6	5
Processor 7	5

So, Total Upgrade Acknowledgement received→ 65

• **Names and count of messages received by L2 cache**

I. Total GET Messages → 65613

II. Total GETX Messages → 61641

III.Total Sharing Writeback Messages (SWB) → 79

IV. Total UPGRADE Messages → 65

V. Total Writeback Messages → 65339

• **Observations**

Some of the observations that we found to be common across all of the four programs are -

- The number of simulated cycles is less than the total number of machine accesses for each of the programs, which means that we are able to utilize the multi-core architecture and perform some amount of computation in parallel. The comparison is shown below

Program Name	Total Machine Accesses	Simulated Cycles
prog1.c	140526678	139555415
prog2.c	2620064	2616950
prog3.c	9906899	9896185
Prog4.c	1064789	1064481

- The number of accesses of the L1 cache of processor 0 is the highest. So, processor 0 is performing the highest number of read/write operations among all of the processors.
- The PUTX message count for L1 cache of the processor 0 is very high. This means that the cache suffers from a large number of write misses (as receiving PUTX means that the cache sent GETX message earlier). This in turn explain the large number of L1 cache misses of the processor 0 (as compared to the remaining processors).
- The invalidation requests received by the L1 caches are very less as compared to the number of L1 cache accesses (for all of the processors). This means that ,there are few cases (or even rare in case of program 2,3 and 4) where a processor is trying to modify a block that is already present in shared (S) state in some other processor's private cache
- We know that a L1 cache receives GET message from L2 cache bank when it has a block in modified (M) state and some other processor wants to read the block. In our simulation, we observed that such a case is very rare as the message count for GET is very less as compared to the total number of L1 cache accesses for each of the processor.
- Similarly, we know that a L1 cache receives a GETX message when it has a block in the modified (M) state and another processor wants to write to the block. As the GETX message count is also very low, we can conclude that this

case occurs rarely.

- The low GET and GETX message counts also explains the low SWB message count.

We also found some observations specific to some of the programs like -

- For program 2,3 and 4, the count of Upgrade messages received by L2 cache banks is very low. This means that either the block is getting evicted from the cache before it can be upgraded or few write requests comes in after read requests for the same block.