Advanced Computer Architecture

Cs622 Assignment 3: Report

Debanjan Chatterjee(20111016) and P J Leo Evenss(20111038)

Group:13

1.Introduction

Objective: Firstly, to collect the thread-wise memory access trace for each of the four binaries provided using a PIN instrumentation tool . It contains 4 fields namely ThreadId, MachineAccess, R/W, GlobalCount.

Secondly, using the obtained traces, implementing a simulator to model the multi-core hierarchy supporting directory level cache coherence using L1 and L2-shared caches.

2.Result Analysis

1)Number of Simulated Cycles:

Prog1	**Prog2**	**Prog3**	**Prog4**
140098902	2518123	9707119	1063679

2a) L1 Cache Access and Misses(core-wise):Prog1 and Prog2

	Prog1			**Prog2**		
Core	**Access**	**Hits**	**Miss**	**Access**	**Hits**	**Miss**
0	37764990	35689051	2075939	735781	602948	132833
1	14680248	14679028	1220	271341	271219	122
2	14680247	14679257	990	262299	262219	80
3	14680248	14679028	1220	262299	261696	603
4	14680248	14679358	890	262299	262220	79
5	14680248	14679358	890	267650	267573	77
6	14680248	14679358	890	262299	262222	77
7	14680248	14679257	991	198174	198113	61

2b) L1 Cache Access and Misses(core-wise):Prog3 and Prog4

	Prog3			**Prog4**		
Core	**Access**	**Hits**	**Miss**	**Access**	**Hits**	**Miss**
0	2160895	2024791	131505	604700	471885	132815
1	1101695	1101653	42	65690	65610	80
2	1066364	1066310	54	65690	65610	80
3	1054537	1054504	33	65690	65610	80
4	1101677	1101646	31	65690	65610	80
5	1069117	1069083	34	65690	65610	80
6	1082257	1082226	31	65690	65610	80
7	1076894	1076870	24	65690	65610	80

3) L2 misses

Prog1	**Prog2**	**Prog3**	**Prog4**
7295103	68399	73122	67278

4a) L1 Message Names and Counts (corewise):Prog1

Core	**Get**	**GetX**	**Put**	**PutX**	**PutE**	**InVal**	**InvalAck**	**Nack**	**WbInval**	**WbAck**
0	4556	197630	5242	521121	12302	221	31	32243512	3520	2564
1	300	2	4562	451	220	374	80	8858123	123	0
2	20	3	5230	564	456	178	944	8843125	123	0
3	35	3	2552	123	231	259	198	8827233	120	0
4	12	3	1250	252	123	1349	169	8512367	123	0
5	60	2	5235	252	123	13	302	7562155	125	0
6	10	1	2545	252	123	492	501	8231022	152	0
7	127	1	2352	252	123	169	830	8452526	125	0

4b) L1 Message Names and Counts (corewise):Prog2

Core	**Get**	**GetX**	**Put**	**PutX**	**PutE**	**InVal**	**InvalAck**	**Nack**	**WbInval**	**WbAck**
0	689	65669	702	131489	830	21	12	259550	231	122
1	29	1	35	11	30	3	5	266462	0	0
2	5	1	17	11	6	3	5	262073	0	0
3	266	3	276	10	267	0	4	260010	0	0
4	5	1	16	10	7	2	4	262058	0	0
5	4	1	15	10	6	2	4	264731	0	0
6	4	1	15	10	6	2	4	261985	0	0
7	11	1	8	9	13	8	3	197186	0	0

4c) L1 Message Names and Counts (corewise):Prog3

Core	**Get**	**GetX**	**Put**	**PutX**	**PutE**	**InVal**	**InvalAck**	**Nack**	**WbInval**	**WbAck**
0	3252	65686	3253	131506	3402	0	1	1617182	844	109
1	20	1	21	8	26	1	1	1075046	23	0
2	24	1	35	8	32	0	1	1057300	23	0
3	10	1	8	7	18	6	0	1051399	24	0
4	3	1	13	8	11	0	1	1074969	23	0
5	4	1	15	8	12	0	1	10586628	24	0
6	3	1	13	8	11	0	1	1065120	23	0
7	2	1	12	7	6	0	1	1062405	17	0

4d) L1 Message Names and Counts (corewise):Prog4

Core	**Get**	**GetX**	**Put**	**PutX**	**PutE**	**InVal**	**InvalAck**	**Nack**	**WbInval**	**WbAck**
0	424	65628	447	131418	573	0	23	128992	227	134
1	9	3	12	12	10	9	4	65505	0	0
2	5	2	16	12	6	2	4	65507	0	0
3	14	3	9	8	15	28	0	65503	0	0
4	4	3	16	12	6	2	4	65505	0	0
5	5	3	15	12	7	2	4	65505	0	0
6	4	3	16	12	6	2	4	65505	0	0
7	4	3	16	12	6	2	4	65505	0	0

5) L2 Message and Counts:

Messages	Prog1	Prog2	Prog3	Prog4
Get	63245120	1447391	4814573	523201
GetX	74125364	720560	4383779	197671
Upgrd	45254	370	2145	49
SWB	5120	1010	3318	469
Ack	197645	65676	65693	65648
WB	98130	65684	65618	65697

3.Considerations

- Nack counter is considered to be 1 (a number less than 20 as mentioned) just to maintain simplicity.
- WbInval is msg sent to invalidate the block in L1 level if the corresponding block is evicted from the L2 cache.
- Silent Eviction for the S state

4.Observations

1.Due to the consideration of waiting time of Nack as 1, the number of Get and GetX requests being sent has increased to L2 , as there is a back to back sending of request and this inturn led to more Nacks and eventually more traffic, which seemed to be the slowdown of Prog1 with large amount of traces.

2. The high number of hits indicate that , there is a large amount of spatial locality among the threads , so the sequence of machine accesses have been sent to a single core following little endian ordering.

3. The Coreid 0 has been overloaded with more machine access than others threads and this lead to more inconsistency when the shared cache is distributed among banks in NUCA architecture.

4. The GetX and Get Messages in the L1 level shows the request being forwarded by the home directory to the respective owner cores . So the amount of Get and GetX in L1 level equals the number of Shared write backs and Acks in L2 level.

5. The GetX and Get Messages in L2 level equals with the number of Put,PutX,PutE and also the request beings sent due to Nacks.

6.The number of Invalidations to the L1 equals with the number of InvalAcks.

7. The silent eviction for the S state have been used and so the number of write backs here mean only to the blocks replaced due to M state or E state

8. The Get and GetX request in L1 level is more among core 0 which shows that , it had most of the machine accesses in the M state or E state.

9. The Nacks arise due to the block requested being held by some other owner core and already a request to that block is in transit.

10. The values of Getx,Nack are expected to change with the changing the order of execution.