CS622: Advance Computer Architecture #1

Aditya Jain (20111004) | Mahi Agrawal (20111029)
adityaj20@iitk.ac.in | maahi20@iitk.ac.in
Group Id: 23

Indian Institute of Technology, Kanpur— October 6, 2020

Problem 1

Given the trace file of L1 cache misses for six SPEC CPU 2006 applications, calculate the number of L2 and L3 cache misses for different cache hierarchy policies - inclusion, NINE, and exclusion.

Table 1: Number of L2 Misses

Trace File	Number of References	Number of L2 Misses			L2 Misses Rate (PKR)		
		Inclusive	NINE	Exclusive	Inclusive	NINE	Exclusive
bzip2	1,06,57,627	53,98,166	53,97,576	53,97,576	506.51	506.45	506.45
gcc	1,46,10,811	30,36,461	30,29,809	30,29,809	207.82	207.37	207.37
gromacs	34,31,511	3,36,851	3,36,724	3,36,724	98.16	98.13	98.13
h264ref	23,48,573	9,69,678	9,65,624	9,65,624	412.88	411.15	411.15
hmmer	35,09,765	17,43,421	17,35,322	17,35,322	496.73	494.43	494.43
sphinx3	1,07,53,447	88,20,349	88,15,130	88,15,130	820.23	819.75	819.75

PKR - Number of L2 misses per 1K references

Observation 1.1: For all six applications, inclusion cache policy has the highest number of both L2 and L3 cache misses.

Inclusion cache policy performance is worse as it suffers from:

- **Inclusion Victims:** When a line is evicted from the L3 cache, inclusion policy enforces the eviction of that block from L2 cache also. When access hits in the L2 cache, the LRU order of the block is updated only in the target L2 cache set, and the LRU order of the block remains unchanged in the L3 cache. As a result, a block that is receiving a lot of hits in the L2 cache may become LRU in its L3 cache set and may get evicted. So this frequently hit block on the next reference will be miss in both the caches and would be brought from memory.
- **Small effective cache capacity:** In inclusion policy, L3 cache contains all the blocks of L2 cache. This duplication of blocks results in the decrease in the effective cache capacity. In our experiment, the ratio of the size of L2 cache to the L3 cache is 1:4. So effectively only 75% of L3 cache capacity is available for storing blocks that are not present in the L2 cache.

NINE and exclusion policy suffered less L2 misses as they are immune to inclusion victims

Observation 1.2 : The number of L2 Misses in the case of NINE and Exclusion policy are same, where as the number of misses for L3 cache are not same in NINE and exclusion policy.

- The L2 misses are same in both as an eviction of block in L3 cache does not invalidates the block in L2 cache, where as the L3 misses are different because the effective capacity of L3 cache in case of exclusive policy is greater than the effective capacity in case of NINE policy.

Trace File	Total References	Number of L3 Misses			L3 Miss Rate (PKR)		
		Inclusive	NINE	Exclusive	Inclusive	NINE	Exclusive
bzip2	1,06,57,627	14,46,388	14,45,846	8,89,221	135.71	135.66	83.44
gcc	1,46,10,811	13,73,402	13,66,248	12,42,825	94.00	93.51	85.06
gromacs	34,31,511	1,70,531	1,70,459	1,59,306	49.70	49.67	46.42
h264ref	23,48,573	3,42,146	3,33,583	1,43,686	145.68	142.04	61.18
hmmer	35,09,765	3,91,226	3,76,344	3,00,047	111.47	107.23	85.49
sphinx3	1,07,53,447	82,07,362	82,05,144	72,20,777	763.23	763.02	671.48

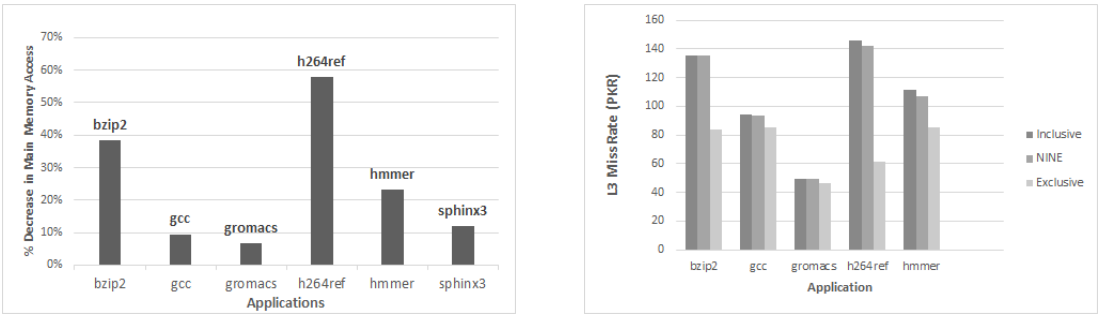
PKR - Number of L2 misses per 1K references

Table 2: Number of L3 Misses

Observation 1.3: In the case of L3 caches, we observed the number of misses is least in the exclusive and most in the inclusive cache.

Exclusive L3 misses < NINE L3 misses < Inclusive L3 misses

Miss rate of the NINE L3 cache is marginally lower than that of inclusive L3 cache. Since NINE cache also suffers from some decrease in effective cache capacity, this marginal difference is due to the absence of inclusion victims.



(a) % decrease in M.M. access in exclusion policy

(b) L3 misses for different cache policies

Figure 1

Figure (1.a) shows the percentage decrease in the number of main memory access for blocks on changing the hierarchy policy from inclusion to exclusion. It shows significant fall for all applications experimented.

Observation 1.4: The number of misses in the exclusive L3 cache is significantly lower than that of both inclusive and NINE caches.

Exclusive cache has the largest cache capacity as it does not allows duplication of blocks among the levels of the cache hierarchy. In our experiment:

- Effective exclusive policy capacity = 2560 KB
- Effective inclusive policy capacity = 2048 KB

Exclusion policy effective cache capacity is **1.25 times** of the inclusion policy cache capacity. Due to this increase in cache capacity and no inclusion victims in exclusion cache policy enjoys significantly low L3 cache misses.

Problem 2

Observation 2.1 : LRU performance is far worse than the Belady's Min policy

Explanation

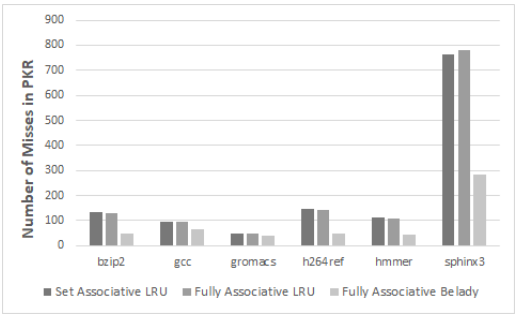
- **Lru policy** performs worse than the **Belady's Min policy** as the LRU looks into the past references and **assumes that the block which is least recently used is unlikely to be requested in near future**,so the block which is least recently used is thrown out of the cache,but this **assumption does not hold always**, due to which it might replace a block which is going to be used in near future,incurring more number of misses. where as **Belady's Min policy** is an optimal policy,it **looks into the future references and replaces the block in the cache which is not going to be used for the longest time in future** and hence decreasing the number of misses.
- Considering the above points we can say that the assumptions made by LRU were not correct for the large number of evictions from L3 cache, this is also depicted by the table and the graph which shows the total number of misses in the case of belady's Min policy and LRU policy.
- This also explains the huge difference between the number of misses and the performance gap between the LRU and belady's Min policy.

Trace File	Number of References	Number of L3 Misses		
		Set Associative LRU	Fully Associative LRU	Fully Associative Belady
bzip2	1,06,57,627	14,46,388	13,61,401	5,36,836
gcc	1,46,10,811	13,73,402	13,69,924	9,39,289
gromacs	34,31,511	1,70,531	1,69,368	1,43,254
h264ref	23,48,573	3,42,146	3,35,880	1,11,605
hmmer	35,09,765	3,91,226	3,77,024	1,53,447
sphinx3	1,07,53,447	82,07,362	83,87,248	30,68,580

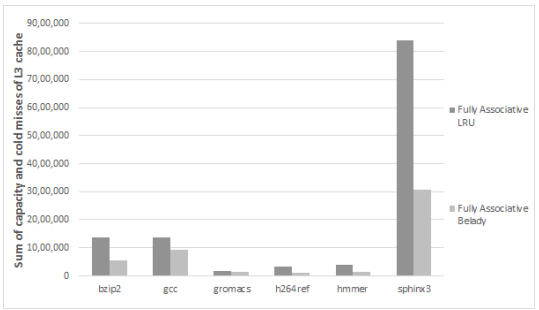
Table 3: Number of L3 Misses

Trace File	Number of References	L3 Miss Rate (PKR)		
		Set Associative LRU	Fully Associative LRU	Fully Associative Belady
bzip2	1,06,57,627	135.714	127.74	50.371
gcc	1,46,10,811	93.999	93.761	64.287
gromacs	34,31,511	49.696	49.357	41.747
h264ref	23,48,573	145.682	143.014	47.52
hmmer	35,09,765	111.468	107.421	43.72
sphinx3	1,07,53,447	763.231	779.959	285.358

Table 4: Number of L3 Misses per Thousand Access (PKR)



(a) Number of L3 Misses per thousand Access



(b) L3 misses for different cache policies

Figure 2

Observation 2.2 : Belady's Min policy cannot be implemented in real.
Explanation

- **Belady's Min policy** requires the references to future block request therefore it is **not possible to implement it in real**, it is significant only for the purpose that one should not scratch his head to find a policy which will give better performance than Belady's Min policy.

Observation 2.3 : Fully associative cache does not always perform better than the set associative cache
Explanation: There are two possible reasons for set associative cache performing much better than the fully associative cache, for the same cache size and block size.

- One reason is that, there might be a possibility that the working set of the blocks accessed is in a such manner that the working set exceeds the size of the cache,in such case there will always be a miss in fully associative cache on each access, where as if the working set is such that although it exceeds the size of the cache, but they are mapped in different sets in a manner, that some sets are not completely occupied by the working set, in this case set associative cache will incur less number of misses than the fully associative cache.

- For example consider there is a cache with 4 lines, and one implementation is set associative cache with 2 sets,and 2 ways per set,the other implementation is of fully associative with 4 lines, consider the block accesses as 1,2,4,6,8,1,2,4,6,8... in this fully associative will have 100% miss rate, where as in set associative there will be a hit on 1, after the initial cold miss.

- The other reason can be that the block replaced in fully associative cache and set associative cache need not be same. The replaced block in fully associative cache is LRU to the whole cache, where as the replaced block in set associative cache may or may not be LRU to the whole cache, the block is LRU to a specific set. If these two blocks are not same then there is the possibility that the block evicted by the fully associative cache may be accessed in near future, and if it is accessed in near future, then it will incur a miss, where as it will be a hit for set associative cache.

- These two points explains why set associative can have better performance than fully associative cache, and **this is also shown by the trace file sphinx3**(Table 5)

Trace File	Number of L3 Misses				
	Set Associative LRU	Fully Associative LRU	Cold	Conflict	Capacity
bzip2	14,46,388	13,61,401	1,19,753	84,987	12,41,648
gcc	13,73,402	13,69,924	7,73,053	3,478	5,96,871
gromacs	1,70,531	1,69,368	7,73,053	1,163	61,406
h264ref	3,42,146	3,35,880	63,703	6,266	2,72,177
hmmer	3,91,226	3,77,024	75,884	14,202	3,01,140
sphinx3	82,07,362	83,87,248	1,22,069	-1,79,886	82,65,179

Table 5: L3 Misses Classification, Considering FA LRU

Trace File	Number of L3 Misses				
	Set Associative LRU	Fully Associative Belady	Cold	Conflict	Capacity
bzip2	1446388	5,36,836	119753	9,09,522	4,17,083
gcc	1373402	9,39,289	7,73,053	4,34,113	1,66,236
gromacs	170531	1,43,254	7,73,053	27,277	35,292
h264ref	342146	1,11,605	63,703	2,30,541	47,902
hmmer	391226	1,53,447	75,884	2,37,779	77,563
sphinx3	8207362	30,68,580	1,22,069	51,38,782	29,46,511

Table 6: L3 Misses Classification,Considering FA Belady