Project Report CDA5106- Fall 2015

Effects of Inclusions in Hierarchical Caches

Somnath Saha

Abstract

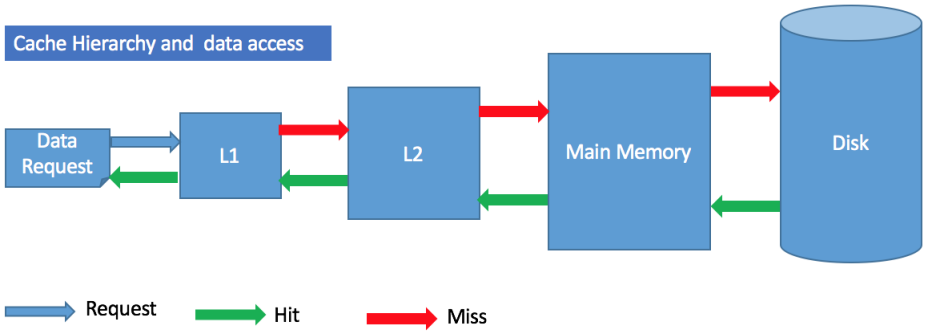
The word inclusion with reference to the hierarchical caches means, the content of lower level caches should be present at the higher level of caches. Enforcing inclusion in hierarchical caches means that under all circumstances this property should be preserved. In this project, we enforce inclusion property in Simple Scalar simulator and perform a comparative study for before inclusion result and after inclusion result.

1 Introduction

In this project, a comparative study has been performed based on the cache miss rate captured before inclusion and after enforcement of inclusion in the simple scalar simulator. Three benchmarks namely "GCC", "bzip", "VPR" have been considered for this study. We simulate the benchmarks and capture the results for analysis. The report is organized as follows. Section 2 provides the background of cache access and simulation using a simple scalar simulator. Section 3 provides the methodology. Section 4 provides experimental methods. Section 5 provides an evaluation. Section 6 provide a conclusion.

2 Background

A "memory hierarchy" in computer storage distinguishes each level in the "hierarchy" by response time. Hierarchical caches contain different level of caches like L1, L2 and main memory. Data access time from the caches increases in the hierarchy in an order L1-¿L2-¿Main Memory. Here we have concentrated on 2 level of caches L1 and L2. A cache access can be a hit or miss. The processor requests data from the lower level of caches. If it finds the data in L1, then it's called a hit otherwise a miss. If a miss occurs, request sends to the next level of caches which is L2. If the data is found, means to hit , it gets copied into L1 else L2 cache miss occurs. Further, request sends to the main memory to bring the data from. The following figure shows the hierarchy of caches. Enforcement is required each time L2 has a miss, cannot accommodate data and block replacement is required for the same.



3 Methodology

One of the inclusion property we have added in this project is to check if any block present into L1 data cache should be present into L2 data cache as well. If it does not find the same block into L2 cache, it removes the block from L1 data cache to keep data consistent. Inclusion property has been added based on following logic. During a L2 cache miss, missing block gets replaced from main memory. While, we replace block for L2 cache, we check if the same address is present into lower level L1 cache. If so we remove the block from L1 cache. A L2 block is larger in size compare to L1 block. One block in L2 cache would be equivalent to multiple rows in L1 cache. Algorithm of cache inclusion has been provided below. Following piece of code check the replaced block of L2 cache in L1 cache and remove it by calling cache_flush_addr(cp_l1,addr) methods.

Modifications have been done in the following files. cache.c, cache.h, sim_cache.c

```
if ( ! strcmp (cp->name,"dl2" ) && cp->lower_cp != NULL )
{
// Add the inclusion property
md_addr_t      l1_addr_to_be_removed;

// Check if l2 cache is pointing to L1 cache
if(cp->lower_cp !=NULL)
     {
```

```
      // There will be multiple rows in L1 for one row in L2 cache.
      int count_l1;
      for( count_l1=0;count_l1<(cp->bsize/cp->lower_cp->bsize);count_l1++)
      {
      //  extracting the block address of L1 cache to L2 cache
      r_add=CACHE_MK_BADDR(cp,repl->tag,set) +  (count_l1*(cp->lower_cp->bsize));

      // Check if L1 cache contain the address. Return non-zero value on success
      int isBlockPresent = cache_probe(cp->lower_cp, r_add );

      if(isBlockPresent)
      {
           // Flush the address in L1 cache
           lat += cache_flush_addr(cp->lower_cp, r_add ,now);
      }
    }
   }
}
```

4 Experimental Method

This project shows a comparative study between before inclusion and after inclusion of the property. There benchmarks namely GCC, bzip2, VPR have been considered for the experiment with the following three cache configurations.
CF-A: The L1D-cache and L1I-cache configuration: 16kB, 2-way with block size as 32 bytes, Unified L2 configuration: 512kB, 8-way with 128-Byte block size, all use LRU replacement policy CF-B: The L1D-cache and L1I-cache configuration: 16kB, 2-way with block size as 32 bytes, Unified L2 configuration: 128kB, 2-way with 128-Byte block size, all use LRU replacement policy CF-C: The L1D-cache and L1I-cache configuration: 8kB, 2-way with block size as 32 bytes, Unified L2 configuration: 128kB, 2-way with 128-Byte block size, all use LRU replacement policy

Simulator Run Command used:
./sim-cache -redir:sim SimResultFileName -cache:dl1 dl1:256:32:2:l -cache:il1 il1:256:32:2:l -cache:il2 dl2 -cache:dl2 dl2:512:128:8:l

./sim-cache -redir:sim SimResultFileName -cache:dl1 dl1:256:32:2:l -cache:il1 il1:256:32:2:l -cache:il2 dl2 -cache:dl2 dl2:512:128:2:l
./sim-cache -redir:sim SimResultFileName -cache:dl1 dl1:128:32:2:l -cache:il1 il1:128:32:2:l -cache:il2 dl2 -cache:dl2 dl2:512:128:2:l
Each bench mark has been simulated for each configuration three times to get the average data on cache miss_rate. Experimental data has been collected before inclusion and after inclusion to prepare the graphs which does the comparative studies of miss_rate.

5 Evaluation

The experimental result shows the miss_rate of different level of caches "Instruction Level cache 1" (IL1), Data cache level 1 (DL1) and Combined Cache(DL2) from three bench marks namely GCC, VPR, bzip2. Also a comparative study has been performed based on the miss_rate before inclusion and after inclusion.

A difference in miss_rate among configurations is visible as well. Configuration A (CF-A) and Configuration B(CF-B) is same for both IL1 and DL1 cache which has size of cache 256KB. Whereas cache size reduces to 128KB in Configuration C (CF-C). Cache in smaller size has more miss_rate compare to the other two configurations. This reflects in results from all the three bench marks for both IL1 and DL1. So, it indicates increasing cache size reduces miss_rate.

Miss rate and Total misses from simulation result

Before Inclusion

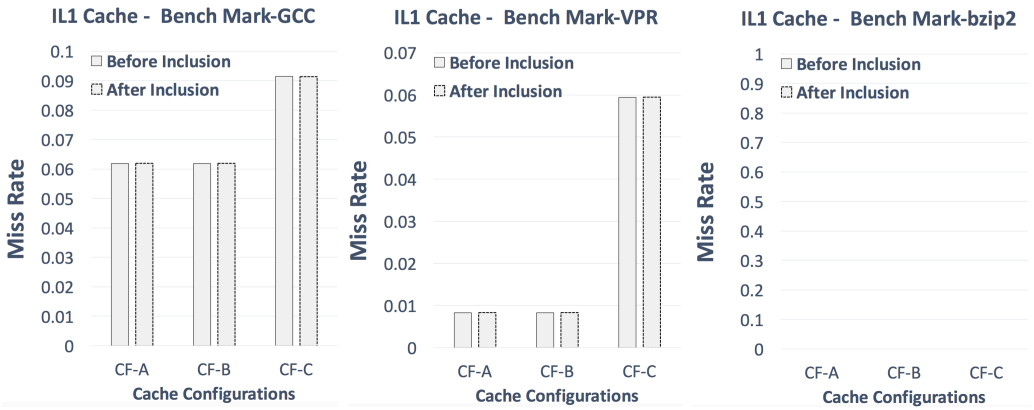
	Config - A		Config - B		Config - C	
	Miss Rate	Total Miss	Miss Rate	Total Miss	Miss Rate	Total Miss
IL1 Average Miss Rate and Total Misses						
GCC	0.0619	100668013	0.0619	100668028	0.0914	148664044
BZIP2	0	7534	0	7534	0	10056
VPR	0.0083	12983861	0.0083	12983861	0.0594	92533904
DL1 Average Miss Rate and Total Miss						
GCC	0.0235	15246766	0.0235	15246768	0.0375	24309972
BZIP2	0.0143	45782491	0.0143	45782491	0.0165	52771891
VPR	0.043	22095988	0.043	22095988	0.0703	36127629
DL2 Average Miss Rate and Total Misses						
GCC	0.0088	1063789	0.0789	9584220	0.0591	10772940
BZIP2	0.3331	22520795	0.5717	38654185	0.4528	35784733
VPR	0.0001	2524	0.0082	360505	0.0062	876460

After Inclusion

	Config - A		Config - B		Config - C	
	Miss Rate	Total Miss	Miss Rate	Total Miss	Miss Rate	Total Miss
IL1 Average Miss Rate and Total Misses						
GCC	0.0619	100668028	0.0619	100668028	0.0914	148664056
BZIP2	0	7534	0	7534	0	10056
VPR	0.0083	12983861	0.0083	12983861	0.0594	92533904
DL1 Average Miss Rate and Total Misses						
GCC	0.0235	15247715	0.0284	18389167	0.0408	26412212
BZIP2	0.0143	45810535	0.0146	46686237	0.0166	53218910
VPR	0.043	22095988	0.0435	22359663	0.0723	37141635
DL2 Average Miss Rate and Total Misses						
GCC	0.0088	1063761	0.0955	11828504	0.0665	12216319
BZIP2	0.333	22522968	0.5258	33816976	0.434	33719034
VPR	0.0001	2524	0.0106	467768	0.0138	1980366

In the above table sows the miss_rate for benchmarks. A zero miss_rate for IL1 cache in bzip benchmarks has appeared which is due to the very small number of cache miss against the total number of cache access.

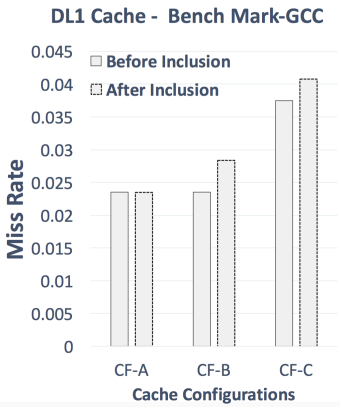
1. Experimental result of IL1 cache

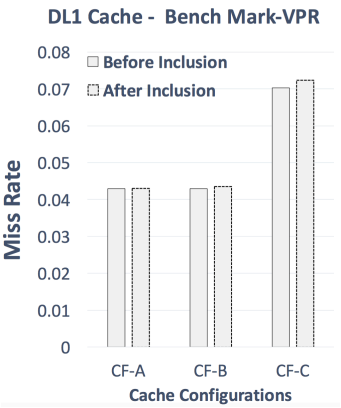


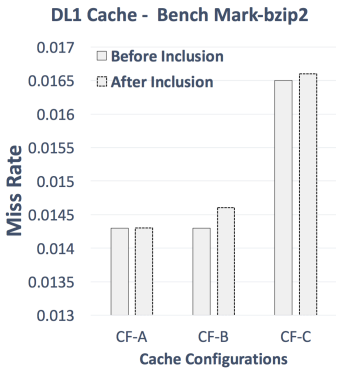
The above result for IL1 cache shows trend of no-change in miss_rate before inclusion and after inclusion for three benchmarks. As the inclusion doesn't impact IL1, so miss_rate should remain same and same reflects on the above result.

Also, a difference in miss rate is visible among configurations. Reducing cache size increases miss_rate. Same reflects in the experimental results for configuration C in all three benchmarks.

2. Experimental result of DL1 cache





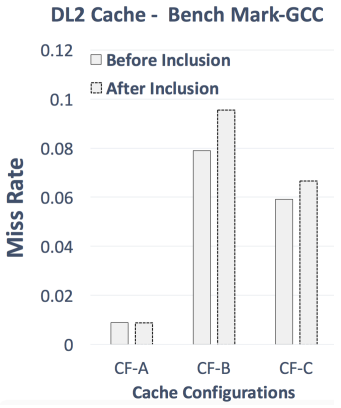


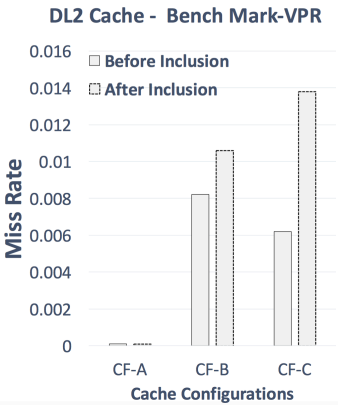
The above result for DL1 cache shows a trend of slight increase in miss rate after inclusion for all three benchmarks.

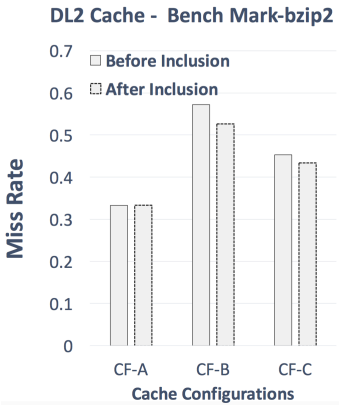
The changes of miss rate are due to the inclusion property that has been enforced in the DL1 cache. To hold the inclusion property, we clear the block from L1 cache if that is not present into the L2 cache. It might create extra miss rate as appeared in the results.

Also, a difference in miss rate is visible among configurations. Reducing cache size increases miss_rate. Same reflects in the experimental results for configuration C in all three benchmarks.

3. Experiemntal result from combined cache DL2







The above result shows miss_rate in combined data cache (DL2) for three benchmarks.

Changes in miss rate due to the enforcement of inclusion can be noticed as well in after inclusion miss_rate. It increases for GCC and VPR benchmarks.

To hold the inclusion property, we clear the block from L1 cache if that is not present into the L2 cache. This phenomenon might increase access into the L2 cache. Hence the miss_rate also differs in all benchmarks.

The change of miss rate also can be noticed from configuration A to configuration B and C. Cache size remain same for all three configurations. However, set-associativity has changed from 8 way set-associativity in configuration A to 2-way associativity in configuration B and C. Reducing the associativity increases the miss rate and the same is reflected in the result for all three benchmarks. Miss rate in configurations B and C has increased from configurations A.

6 Conclusion

The above experiment helps us to conclude that enforcing inclusion in cache impacts the data cache.However, it does not impact on instruction level cache (IL1). It increases the miss_rate in data cache level-1 (DL1) and combined cache (DL2). Also, the experiment for different cache configurations helps us to understand that increasing cache size and set-associativity reduces miss_rate.