NC State University

Department of Electrical and Computer Engineering

ECE 463/521: SPRING 2017

Project #1: Cache Hierarchy

by

PRATHAMESH PRABHUDESAI

NCSU Honor Pledge: "I have neither given nor received unauthorized aid on this test or assignment."

Student’s electronic signature: \_\_Prathamesh Prabhudesai\_\_

(sign by typing your name)

Course number: \_\_ECE 521\_\_

(463 or 521?)

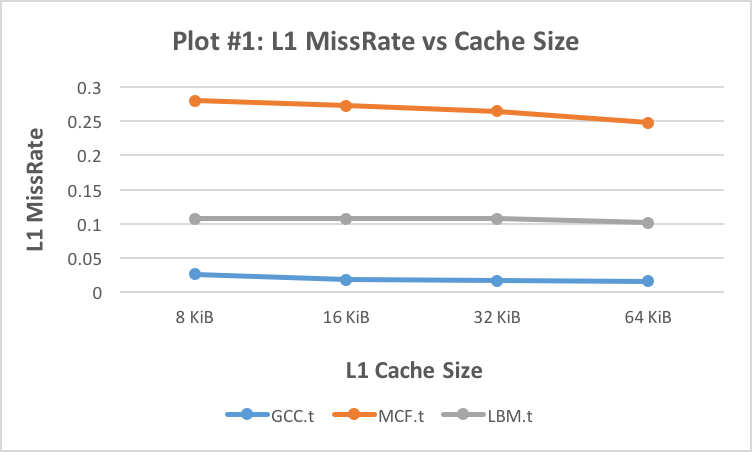
**Exploring the L1 Cache Design**

**(For simplicity, 1GB is considered to be 109 B)**

**Plot #1:**

Assume that there is no L2 cache. Assume that LRU replacement policy is used for the L1 cache. Fix the L1 block size at 64B and the L1 associativity at 4. Vary the cache size between 8KiB, 16KiB, 32KiB and 64KiB. Plot the L1 cache miss rates. The plot should have 4 data points per benchmark.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **L1 Cache Miss Rates** | | |
| **L1 Cache Size** | **GCC.t** | **MCF.t** | **LBM.t** |
| **8 KiB** | 0.026388 | 0.279413 | 0.107725 |
| **16 KiB** | 0.018553 | 0.272435 | 0.107721 |
| **32 KiB** | 0.0169005 | 0.264445 | 0.107721 |
| **64 KiB** | 0.016156 | 0.247528 | 0.101456 |



**Observations:**

1. As Cache size increases, the Cache Miss Rate decreases. For GCC and MCF traces, we can observe this trend. There is considerable decrease in miss rate as the size is increased.
2. However, in the case of LBM trace, increase in size from 8KiB to 16KiB and then 32KiB does not affect Miss Rate. But 64KiB of L1 does show the decrease in miss rate.

**Discussion:**

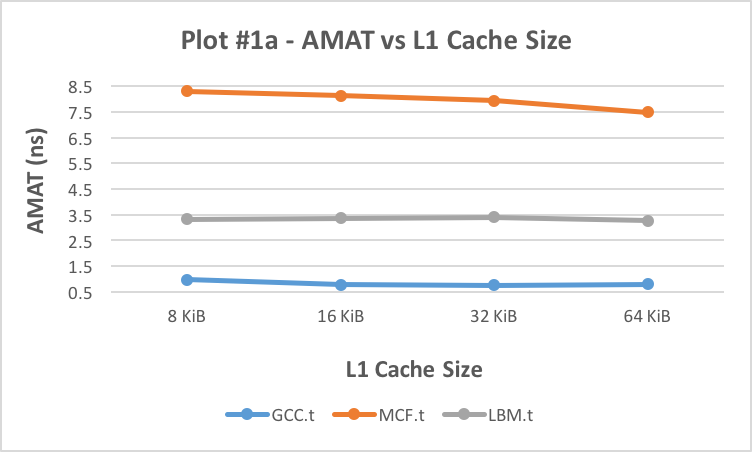
1. The most important reason for decrease in miss rate as the size increases is increasing the cache size decreases the capacity misses. So cache is large enough to hold the data and evicts from the previous cases.
2. The decrease in miss rates achieves saturation after some size. The decrease interval also decreases. This happens because, increase in size cannot change the cold and conflict misses any further.

**Miss Penalty = Memory Latency + (Block Size/Memory Bandwidth)**

**= 25ns + 64B/16GB/s = 25ns + 4ns = 29ns**

**AMAT = Hit Time + Miss Rate × Miss Penalty**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | **L1 AMAT (ns)** | | |
| **L1 Cache Size** | **Hit Time (ns)** | **GCC.t** | **MCF.t** | **LBM.t** |
| 8 KiB | 0.211173 | 0.976425 | 8.31415 | 3.335198 |
| 16 KiB | 0.233936 | 0.771973 | 8.134551 | 3.357845 |
| 32 KiB | 0.27125 | 0.7613645 | 7.940155 | 3.395159 |
| 64 KiB | 0.319481 | 0.788005 | 7.497793 | 3.261705 |

****

**Observations:**

1. As Cache size increases, the AMAT decreases approximately up to the size 32KiB. This trend can be seen in the GCC trace.
2. For MCF trace, this trend is not followed. The increase in size reduces AMAT.
3. The behavior for LBM trace is interesting. The AMAT increases as we go up in size till 32 KiB then the AMAT decreases.

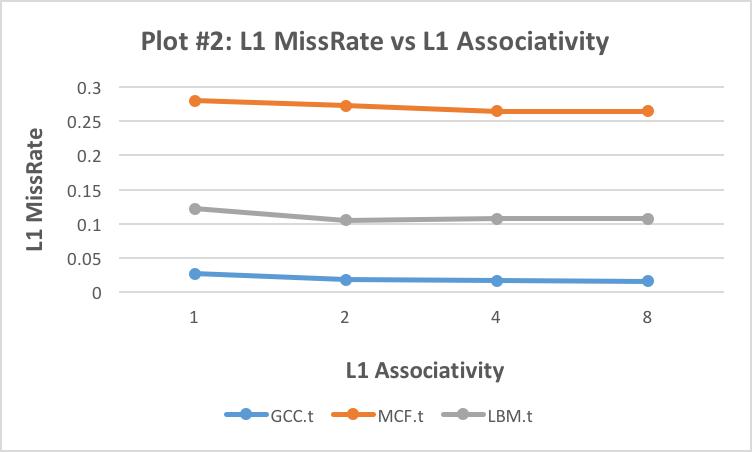
**Discussion:**

1. This happens because increasing the cache size increases the cache area. Thus increasing the hit times; which plays considerable role in deciding AMAT.

**Plot #2:**

Assume that there is no L2 cache. Assume that LRU replacement policy is used for the L1 cache. Fix the L1 block size at 64B and the L1 cache size at 32KiB. Vary the associativity between 1, 2, 4 and 8. Plot the L1 cache miss rates. The plot should have 4 data points per benchmark.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **L1 Cache Miss Rates** | | |
| **L1 Associativity** | **GCC.t** | **MCF.t** | **LBM.t** |
| **1** | 0.0269015 | 0.279413 | 0.121798 |
| **2** | 0.01819 | 0.272435 | 0.105314 |
| **4** | 0.0169005 | 0.264445 | 0.107721 |
| **8** | 0.0164355 | 0.264424 | 0.107721 |



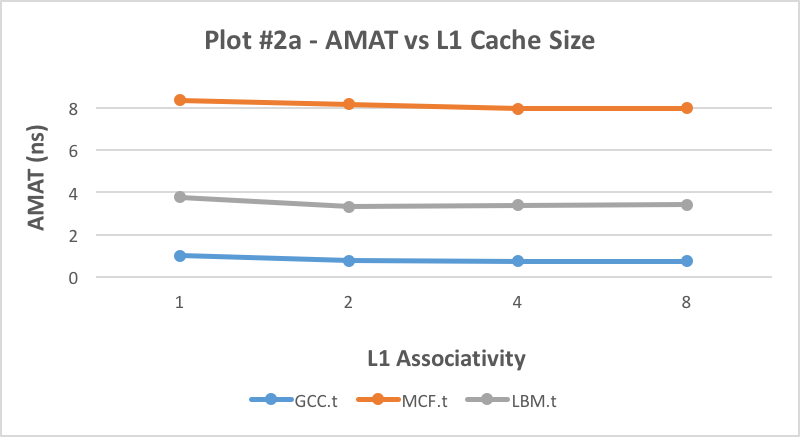
**Observations:**

1. As the associativity increases, the Miss Rate decreases. This trend is seen for all the traces.
2. The miss rate can achieve saturation after some point though. This can be seen in the LBM trace. For 4-way and 8-way it has same miss rate.

**Discussion:**

1. Increasing the associativity decreases the conflict misses that otherwise happens due to two cache-lines mapping to the same set, evicting the existing cache-line.
2. After 4-way associativity, we can also observe that increasing the associativity further doesn’t affect much of the performance because, cold misses & capacity misses are not affected.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | **L1 AMAT (ns)** | | |
| **L1 Associativity** | **Hit Time (ns)** | **GCC.t** | **MCF.t** | **LBM.t** |
| 1 | 0.233353 | 1.0134965 | 8.33633 | 3.765495 |
| 2 | 0.262446 | 0.789956 | 8.163061 | 3.316552 |
| 4 | 0.27125 | 0.7613645 | 7.940155 | 3.395159 |
| 8 | 0.288511 | 0.7651405 | 7.956807 | 3.41242 |



**Observations:**

1. As the associativity increases, the AMAT decreases. This trend is seen for all the traces till the 4-way associativity.
2. AMAT increases from 4-way to 8-way, since the cache area is increased.

**Discussion:**

1. Increasing the associativity decreases the conflict misses that otherwise happens due to two cache-lines mapping to the same set, evicting the existing cache-line.
2. After 4-way associativity, we can also observe that increasing the associativity further doesn’t affect much of the performance because, cold misses & capacity misses are not affected

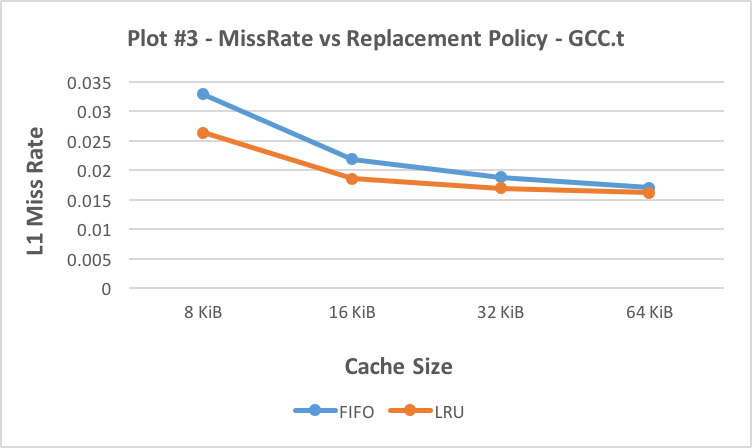
**Best Design:**

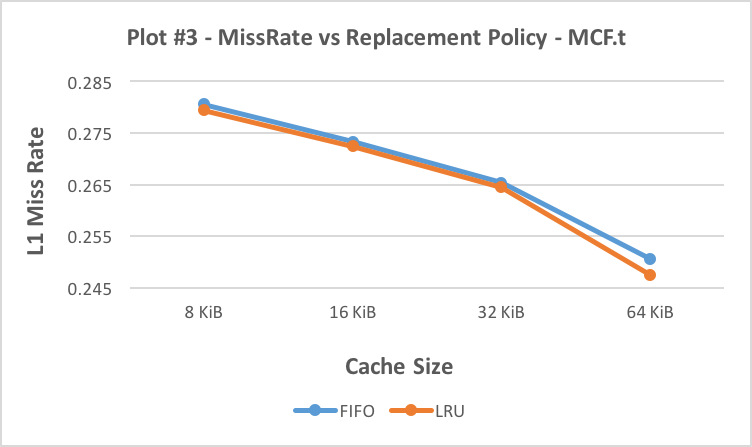
The area increases as we increase the size as well as the associativity of the cache. Taking this under account & considering the fact that smaller area (die area) results lower cost, we should consider cache structure such that it has as small area, less miss rate, less AMAT necessary for greater performance. Considering the hit times in CACTI table & the plots #1 & #2, we will recommend 32KB cache size with 2-way associativity.

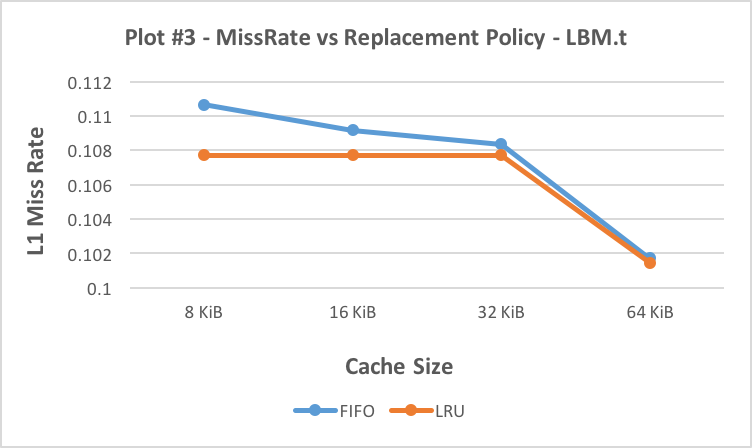
**Exploring the Replacement Policy**

**Plot #3:**

Assume that there is no L2 cache. Fix the L1 block size at 64B and the L1 associativity at 4. Vary cache size between 8KiB, 16KiB, 32KiB and 64KiB. For each configuration vary the replacement policy between LRU and FIFO. Plot the L1 Cache miss rates for each replacement policy. Note that for each benchmark this plot will have 2 data series (one per replacement policy) and each data series will have 4 data points (one per cache size). That is, 8 data points per benchmark.







|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **GCC.t** | | **MCF.t** | | **LBM.t** | |
| **L1 Cache Size** | **FIFO** | **LRU** | **FIFO** | **LRU** | **FIFO** | **LRU** |
| **8 KiB** | 0.0328635 | 0.026388 | 0.280598 | 0.279413 | 0.110667 | 0.107725 |
| **16 KiB** | 0.021876 | 0.018553 | 0.273329 | 0.272435 | 0.109162 | 0.107721 |
| **32 KiB** | 0.0187845 | 0.0169005 | 0.265338 | 0.264445 | 0.108364 | 0.107721 |
| **64 KiB** | 0.017017 | 0.016156 | 0.250573 | 0.247528 | 0.101733 | 0.101465 |

**Observations:**

1. We can see that Miss Rate is always high for FIFO replacement policy compared to LRU policy.
2. LRU gives significantly best performance for all the traces and cache sizes.

**Discussion:**

1. LRU evicts the block based on how recently the block is used. If it is recent it won’t be evicted and least recently used block will be evicted.
2. This check is not done in FIFO. The first block to enter the cache will be evicted first. Hence LRU performs better.

**Recommendation:**

We should choose the LRU as it’s the best replacement policy.

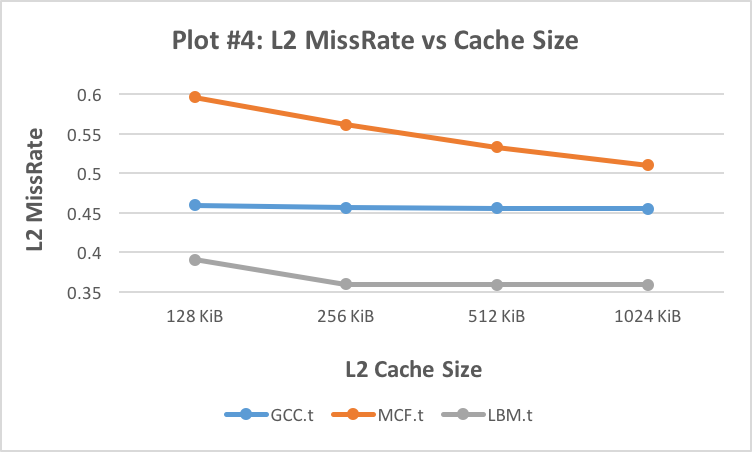
**Exploring the L2 Cache Design**

**(For simplicity, 1GB is considered to be 109 B)**

**Plot #4:**

Fix block size at 64B, L1 cache size at 16KiB, L1 associativity at 4 and the replacement policy to be LRU. Assume that the L2 Cache is always non-inclusive. Fix L2 associativity at 8. Vary the L2 cache size between 128KiB, 256KiB, 512KiB and 1MB. Plot the L2 cache miss rates. The plot should have 4 data points per benchmark.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **L2 Cache Miss Rates** | | |
| **L2 Cache Size** | **GCC.t** | **MCF.t** | **LBM.t** |
| **128 KiB** | 0.459704 | 0.59623 | 0.391004 |
| **256 KiB** | 0.456172 | 0.561573 | 0.359462 |
| **512 KiB** | 0.455574 | 0.533018 | 0.35941 |
| **1024 KiB** | 0.45546 | 0.509986 | 0.35941 |



AMAT = Hit TimeL1 + Miss RateL1 × (Hit TimeL2 + Miss RateL2 × Miss Penalty)

Hit Time for L1 (16 KiB, associativity = 4) = 0.233936

Miss Penalty = 29ns

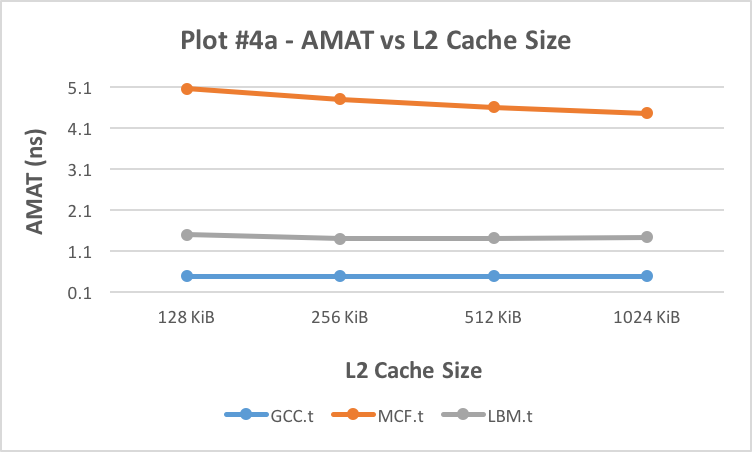
**Observations:**

1. As Cache size increases, the Cache Miss Rate decreases. For GCC and MCF traces, we can observe this trend. There is considerable decrease in miss rate as the size is increased.
2. However, in the case of LBM trace, increase in size from 512KiB to 1024KiB does not affect Miss Rate much.

**Discussion:**

1. The most important reason for decrease in miss rate as the size increases is increasing the cache size decreases the capacity misses. So cache is large enough to hold the data and evicts from the previous cases.
2. The decrease in miss rates achieves saturation after some size. The decrease interval also decreases. This happens because, increase in size cannot change the cold and conflict misses any further.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | **AMAT (ns)** | | |
| **L2 Cache Size** | **Hit Time L2 (ns)** | **GCC.t** | **MCF.t** | **LBM.t** |
| 128 KiB | 0.401236 | 0.488717893 | 5.053830411 | 1.498618458 |
| 256 KiB | 0.458925 | 0.48788785 | 4.7957353 | 1.406298437 |
| 512 KiB | 0.578177 | 0.489778586 | 4.602621657 | 1.418981938 |
| 1024 KiB | 0.705819 | 0.492085392 | 4.455428841 | 1.432731662 |
| L1 Miss Rate | | 0.018553 | 0.272435 | 0.107721 |



**Observations:**

1. As Cache size increases, the AMAT decreases approximately up to the size 256KiB. This trend can be seen in the LBM trace. And then it again starts increasing.
2. For MCF trace, this trend is not followed. The increase in size reduces AMAT.
3. The behavior for GCC trace is interesting. The AMAT increases as we go up in size.

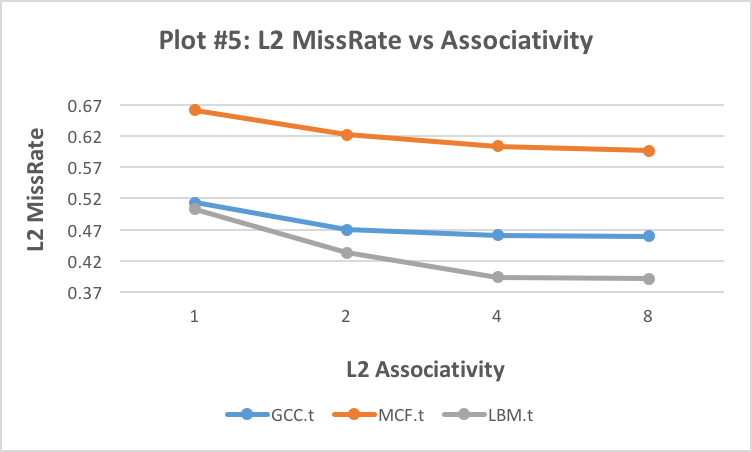
**Discussion:**

1. This happens because increasing the cache size increases the cache area. Thus increasing the hit times; which plays considerable role in deciding AMAT.

**Plot #5:**

Fix block size at 64B, L1 cache size at 16KiB, L1 associativity at 4 and the replacement policy to be LRU. Assume that the L2 Cache is always non-inclusive. Fix the L2 cache size at 128KiB. Vary L2 associativity between 1, 2, 4 and 8. Plot the L2 cache miss rates. The plot should have 4 data points per benchmark.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **L2 Cache Miss Rates** | | |
| **L2 Associativity** | **GCC.t** | **MCF.t** | **LBM.t** |
| **1** | 0.513062 | 0.661931 | 0.502475 |
| **2** | 0.469661 | 0.621907 | 0.432523 |
| **4** | 0.460858 | 0.603782 | 0.393176 |
| **8** | 0.459704 | 0.59623 | 0.391004 |



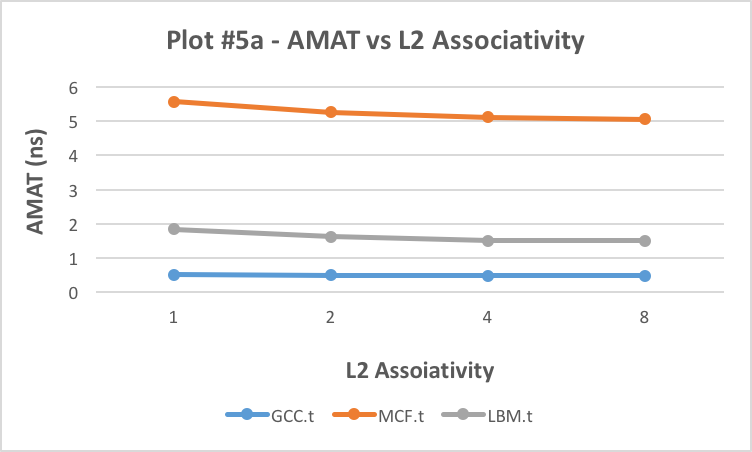
**Observations:**

1. As the associativity increases, the Miss Rate decreases. This trend is seen for all the traces.

**Discussion:**

1. Increasing the associativity decreases the conflict misses that otherwise happens due to two cache-lines mapping to the same set, evicting the existing cache-line.
2. After 4-way associativity, we can also observe that increasing the associativity further doesn’t affect much of the performance because, cold misses & capacity misses are not affected.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | **AMAT (ns)** | | |
| **L2 Assoc.** | **Hit Time L2 (ns)** | **GCC.t** | **MCF.t** | **LBM.t** |
| 1 | 0.3668 | 0.51678758 | 5.563527146 | 1.843134238 |
| 2 | 0.374603 | 0.493581005 | 5.249438741 | 1.625451102 |
| 4 | 0.38028 | 0.488949991 | 5.107786708 | 1.503146187 |
| 8 | 0.401236 | 0.488717893 | 5.053830411 | 1.498618458 |
| L1 Miss Rate | | 0.018553 | 0.272435 | 0.107721 |



**Observations:**

1. As the associativity increases, the AMAT decreases. This trend is seen for all the traces.

**Discussion:**

1. Increasing the associativity decreases the conflict misses that otherwise happens due to two cache-lines mapping to the same set, evicting the existing cache-line.

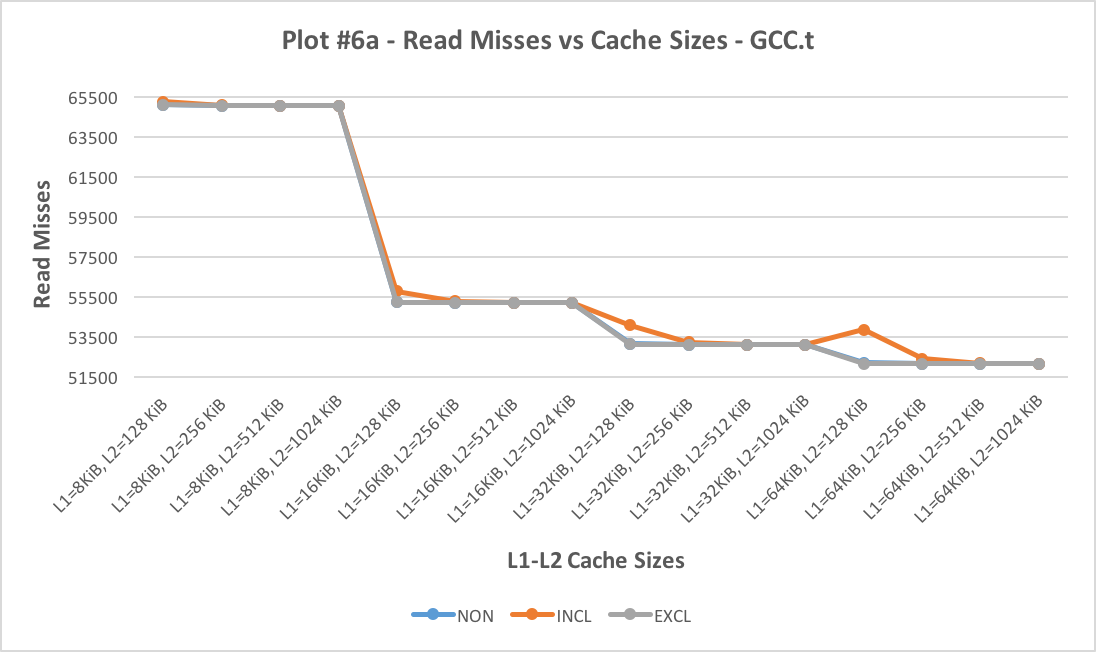
**Exploring the Inclusion Property Choices:**

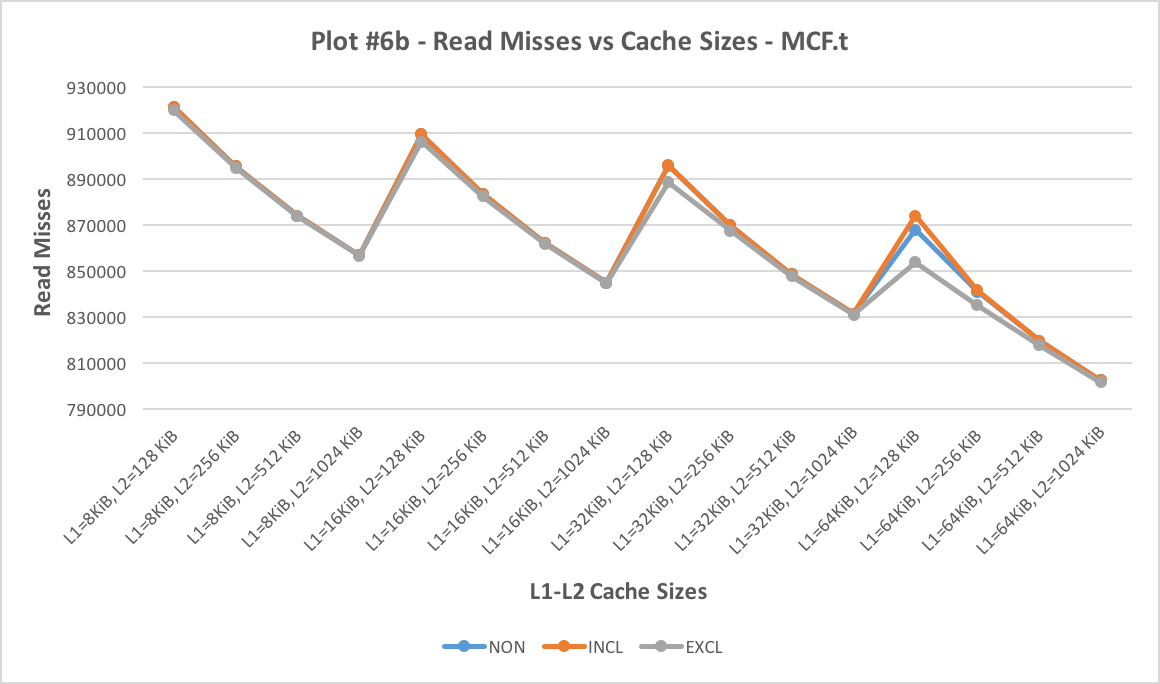
**(For simplicity, 1GB is considered to be 109 B)**

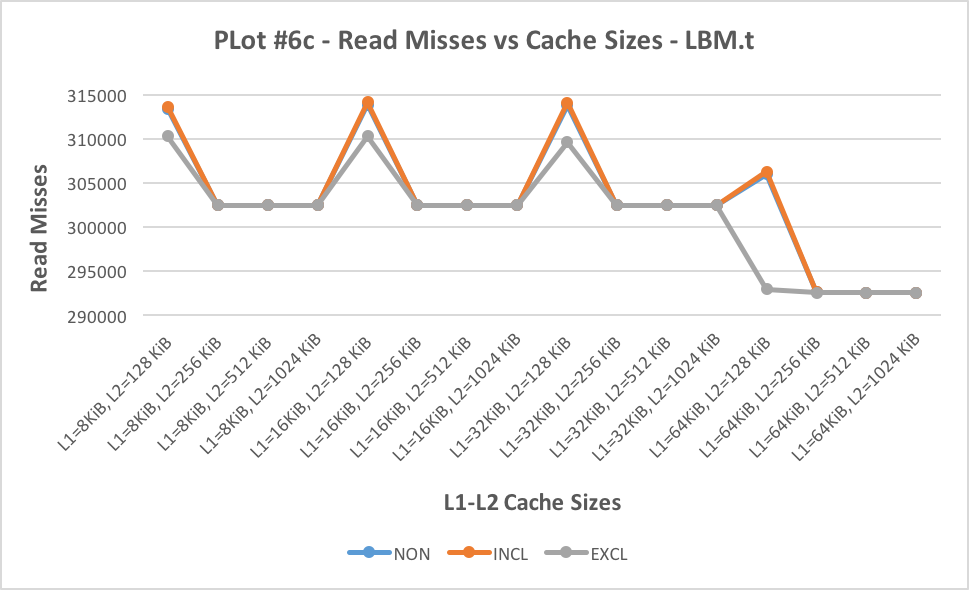
**Plot #6:**

Fix the block size at 64B, the L1 associativity at 4 and the L2 associativity at 8. Vary L1 cache size between 8KiB, 16KiB, 32KiB and 64KiB. For each resulting configuration, vary the L2 cache size between 128KiB, 256KiB, 512KiB and 1MB. There are a total of 16 configurations. Plot the number of cache misses resulting in a memory read for each configuration for each inclusion policy (inclusive, exclusive and non-inclusive). Note that for each benchmark this plot will have 3 data series (one per inclusion policy) and each data series will have 16 points (one per configuration).

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **GCC.t** | | | **MCF.t** | | | **LBM.t** | | |
| **Cache Sizes** | **NON** | **INCL** | **EXCL** | **NON** | **INCL** | **EXCL** | **NON** | **INCL** | **EXCL** |
| **L1=8KiB, L2=128 KiB** | 65092 | 65247 | 65088 | 921339 | 921391 | 919790 | 313461 | 313673 | 310353 |
| **L1=8KiB, L2=256 KiB** | 65049 | 65066 | 65049 | 895405 | 895413 | 894919 | 302528 | 302528 | 302522 |
| **L1=8KiB, L2=512 KiB** | 65045 | 65045 | 65045 | 874010 | 874010 | 873810 | 302517 | 302517 | 302517 |
| **L1=8KiB, L2=1024 KiB** | 65045 | 65045 | 65045 | 856786 | 856786 | 856676 | 302517 | 302517 | 302517 |
| **L1=16KiB, L2=128 KiB** | 55257 | 55791 | 55245 | 909403 | 909504 | 906118 | 313924 | 314172 | 310308 |
| **L1=16KiB, L2=256 KiB** | 55214 | 55298 | 55213 | 883493 | 883539 | 882428 | 302532 | 302532 | 302518 |
| **L1=16KiB, L2=512 KiB** | 55210 | 55222 | 55210 | 862150 | 862152 | 861758 | 302513 | 302513 | 302513 |
| **L1=16KiB, L2=1024 KiB** | 55210 | 55210 | 55210 | 844924 | 844924 | 844712 | 302513 | 302513 | 302513 |
| **L1=32KiB, L2=128 KiB** | 53166 | 54086 | 53142 | 895795 | 896031 | 888605 | 313855 | 314143 | 309637 |
| **L1=32KiB, L2=256 KiB** | 53129 | 53251 | 53120 | 869800 | 869860 | 867386 | 302532 | 302532 | 302517 |
| **L1=32KiB, L2=512 KiB** | 53118 | 53134 | 53118 | 848489 | 848503 | 847620 | 302513 | 302513 | 302513 |
| **L1=32KiB, L2=1024 KiB** | 53118 | 53118 | 53118 | 831267 | 831267 | 830848 | 302513 | 302513 | 302513 |
| **L1=64KiB, L2=128 KiB** | 52217 | 53881 | 52187 | 867973 | 873907 | 853657 | 305996 | 306305 | 292953 |
| **L1=64KiB, L2=256 KiB** | 52175 | 52424 | 52173 | 840858 | 841613 | 835127 | 292616 | 292616 | 292599 |
| **L1=64KiB, L2=512 KiB** | 52171 | 52195 | 52171 | 819600 | 819703 | 817695 | 292599 | 292599 | 292599 |
| **L1=64KiB, L2=1024 KiB** | 52171 | 52171 | 52171 | 802381 | 802396 | 801492 | 292599 | 292599 | 292599 |







**Observations:**

1. As any of the cache size increase, the number of read misses decrease.
2. This trend is observed for all the traces.
3. We have already deduced this conclusion from the plots 1 and 3 where we increase the cache size and there is a decrease in the cache miss rate. Cache miss rate is directly proportional to the read or write misses.
4. This is seen for all the inclusion properties too.
5. Initially the non-inclusion and inclusion gives better results than the exclusive cache implementation but ultimately the exclusive cache is a better implementation.

**Discussion:**

1. **There is more data stored in exclusive cache since the data in L1 and L2 is altogether different.**
2. **Inclusion and non-inclusion gives almost the same results.**

**Plot #7:**

Fix the block size at 64B, the L1 associativity at 4 and the L2 associativity at 8. Vary L1 cache size between 8KiB, 16KiB, 32KiB and 64KiB. For each resulting configuration, vary the L2 cache size between 128KiB, 256KiB, 512KiB and 1MB. There are a total of 16 configurations. Plot the number of cache misses resulting in a memory read for each configuration for each inclusion policy (inclusive, exclusive and non-inclusive). Note that for each benchmark this plot will have 3 data series (one per inclusion policy) and each data series will have 16 points (one per configuration).