**PROJECT : 2**
**CSE 590 : Computer Architecture**

# ANALYSIS OF CACHE PARAMETERS AND TRADE-OFFS ON X86 PROCESSOR USING GEM5

**Venugopal Shah: 50291126**

**Swati Sajee Kumar : 50289560**

# INDEX

## 5. L1 I Associativity



## 6. L2 Associativity



## 7. Block Size

## AIM:

A project to analyse how varying the cache parameters can influence the performance of a processor to great extents. The architecture used for analysis is X86 architecture. Also to note the best suited parameters for each benchmark through analysis of hit rates, miss rates and CPIs.

## SOFTWARE INITIALIZATION:

The software used for this project is gem5. The gem5 simulator is a modular platform for computer-system architecture research, encompassing system-level architecture as well as processor microarchitecture

- Step 1: Install gem5 using following commands:

```
$ sudo apt install build-essential git m4 scons zlib1g zlib1g-dev libprotobuf-dev protobuf-compiler libprotoc-dev libgoogle-perftools-dev python-dev python
```

- Step 2: To build gem5 on Virtual Machine use following command:

```
$ sudo scons build/X86/gem5.opt -j <number of CPUs+1>
```

## BENCHMARK DESCRIPTION

1. 401.bzip2:
   - Benchmark Description : 401.bzip2 is based on Julian Seward's bzip2 version 1.0.3. All compression and decompression happens entirely in memory. This is to help isolate the work done to only the CPU and memory subsystem.
   - Input Description: 401.bzip2's reference workload has six components: two small JPEG images, a program binary, some program source code in a tar file, an HTML file, and a "combined" file, which is representative of an archive that contains both highly compressible and not very compressible files. Each input set is compressed and decompressed at three different blocking factors ("compression levels"), with the end result of the process being compared to the original data after each decompression step.
   - Output Description : The output files provide a brief outline of what the benchmark is doing as it runs. Output sizes for each compression and decompression are printed to facilitate validation, and the results of decompression are compared with the input data to ensure that they match.
2. 429.mcf:
   - Benchmark Description : 429.mcf is a benchmark which is derived from MCF, a program used for single-depot vehicle scheduling in public mass transportation. The program is written in C. The benchmark version uses almost exclusively integer arithmetic.
   - Input Description : The input file contains line by line the number of timetabled and dead-head trips (first line),for each timetabled trip its starting and ending time,for each dead-head trip its starting and ending timetabled trip and its cost.Worst case execution time is pseudo-polynomial in the number timetabled and dead-head trips and in the amount of the maximal cost coefficient. The expected execution time, however, is in the order of a low-order polynomial.
   - Output Description : The benchmark writes to two output files, inp.out and mcf.out. inp.out contains log information and a checksum. mcf.out contains check output values describing an optimal schedule computed by the program.

3. 456.hmmer:
    - Benchmark Description : Profile Hidden Markov Models (profile HMMs) are statistical models of multiple sequence alignments, which are used in computational biology to search for patterns in DNA sequences.The technique is used to do sensitive database searching, using statistical descriptions of a sequence family's consensus. It is used for protein sequence analysis
    - Input Description: A database (sprot41.dat) is used in the reference workloads. An input workload (nph3.hmm) is used to find a ranked list of best sorting sequences from the sprot41.dat file using the hmmsearch function.For test, train, and 1 of the 2 reference workloads, 3 different hmm files are used to with the hmmcalibrate function to calibrate HMM search statistics. This function scores a large number of synthesized random sequences from the input file and fits an extreme value distribution (EVD) to the histogram of those scores.
    - Output Description: Four output files contains a ranked list of matches.
4. 458.sjeng :
    - For this benchmark according to the output file we observed that there were few histograms and matrices which were given as input. Also the execution time for this benchmark was the longest compared to other benchmarks.
5. 470.lbm :

- There were two input files for this benchmark. The execution time for this benchmark was also long, but lesser than that of 458.sjeng

# ANALYSIS AND RESULTS:

## 1 L1 D Cache Size:

***Commands used are as follows:***

***Benchmark-1: 401.bzip2***

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/swati/Project1_SPEC-master/401.bzip2/src/benchmark -I 100000000 -o /home/swati/Project1_SPEC-master/401.bzip2/data/input.program --caches --l2cache –l1d_size =16kB

***Benchmark-2: 429.mcf***

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/swati/Project1_SPEC-master/429.mcfsrc/benchmark -I 100000000 -o /home/swati/Project1_SPEC-master/429.mcf/data/inp.in --caches --l2cache –l1d_size =16kB

***Benchmark-3: 456.hmmer***

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/vshah3/Project1_SPEC-master/456.hmmer/src/benchmark -I 100000000 -o /home/vshah3/Project1_SPEC-master/456.hmmer/data/bombesin.hmm --caches --l2cache --l1d_size=16kB

***Benchmark-4: 458.sjeng***

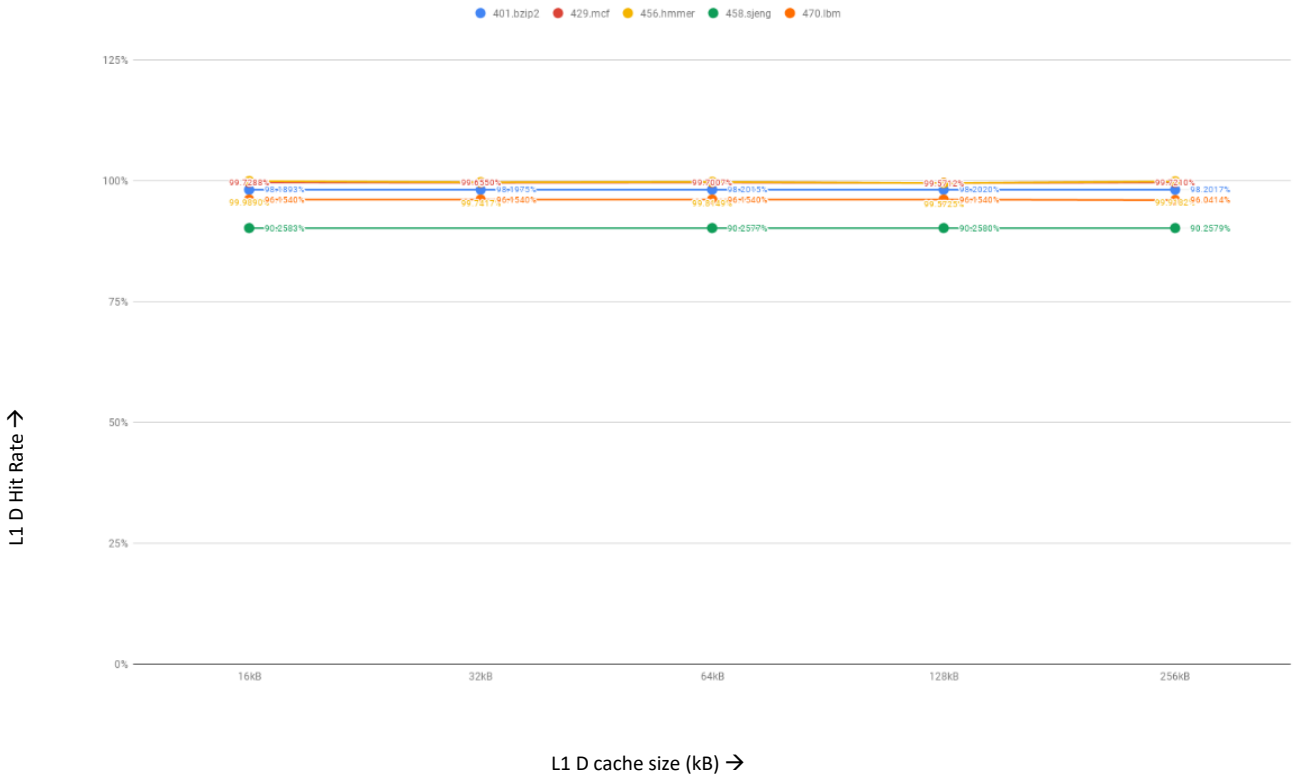
sudo ./build/X86/gem5.opt configs/example/se.py -c /home/swati/Project1_SPEC-master/458.sjeng/src/benchmark -I 100000000 -o /home/swati/Project1_SPEC-master/458.sjeng/data/test.txt --caches --l2cache --l1d_size=16kB

***Benchmark-5: 470.lbm***

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/vshah3/Project1_SPEC-master/470.lbm/src/benchmark -I 100000000 -o “20 reference.dat 0 1 /home/vshah3/Project1_SPEC-master/470.lbm/data/100_100_130_cf_a.of” --caches --l2cache --l1d_size=16kB

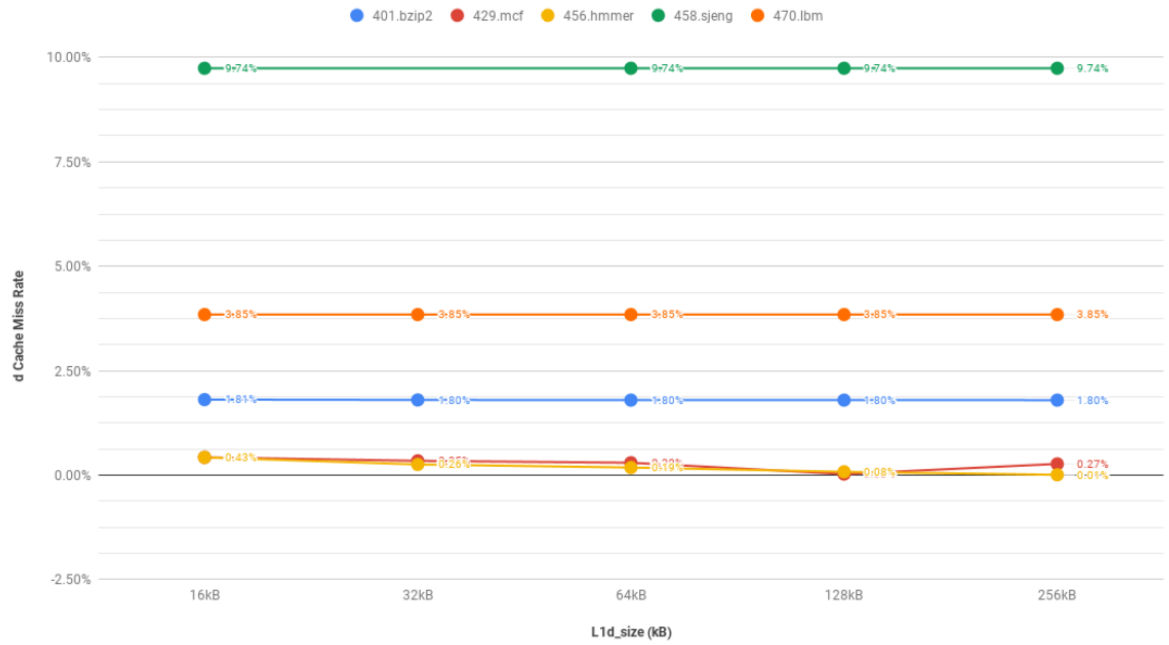
***Tested Values of l1d_size :*** 16kB, 32kB, 64kB, 128kB, 256kB

1.1 L1 D Cache Size vs Hit Rate of L1 D Cache



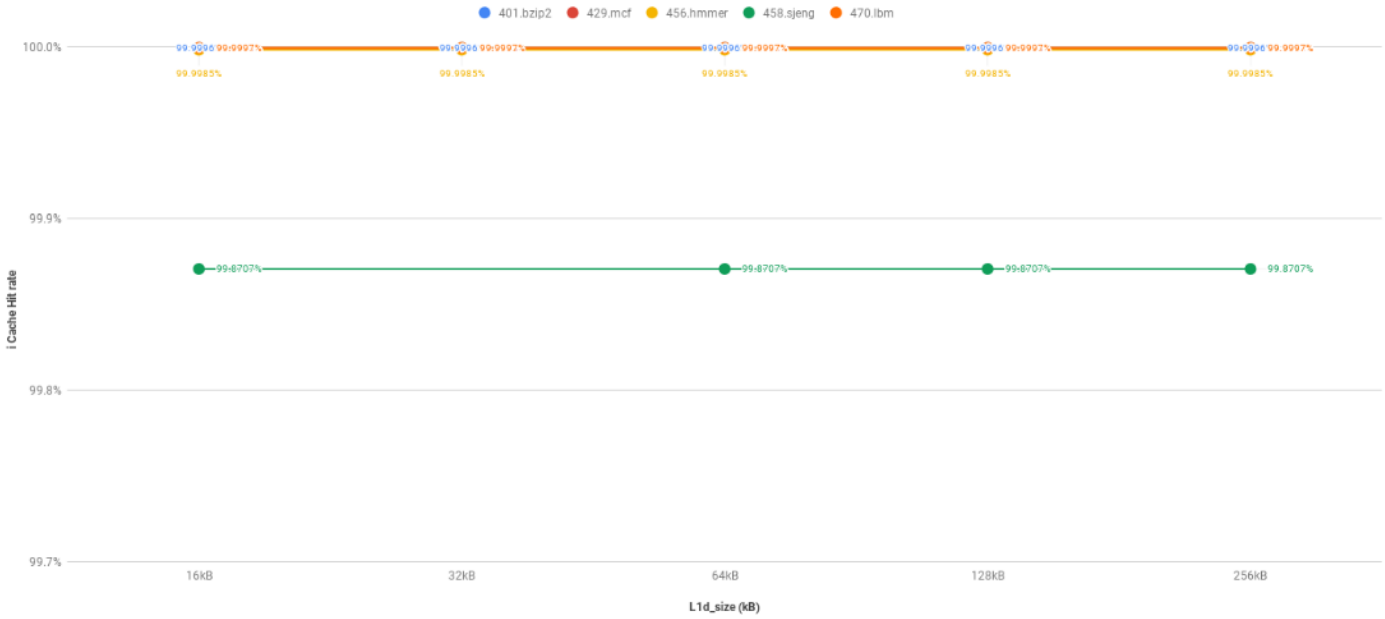

From the graph we can observe that the L1 D (data) hit rate of benchmarks 1,2,3 and 5 lie very close to 98 to 99 percentage. That is the fraction of memory access found in L1 D cache is pretty high. Only for benchmark 4 , the hit rate is slightly low (around 96%) compared to other benchmarks. This could be because of the factor that some amount of data of benchmark 4 is not stored in L1 D cache. Also if we notice there is only a micro point increase in hit rate when we increase the L1 D cache size, this increase is negligible, hence we find almost a linear graph.

1.2 L1 D Cache Size vs Miss Rate of L1 D Cache




Miss Rate is (1-hit rate), thus it is just the inverse of what we discussed above. That is, for benchmarks 1,2,3 and 5 the L1 D miss rate is less. Benchmark 2 and 3 have lowest miss rates of around 0.43 %, followed by Benchmark 1 and 5 having 1.80 % and 3.85% respectively. The highest miss rate can be seen in Benchmark-4.

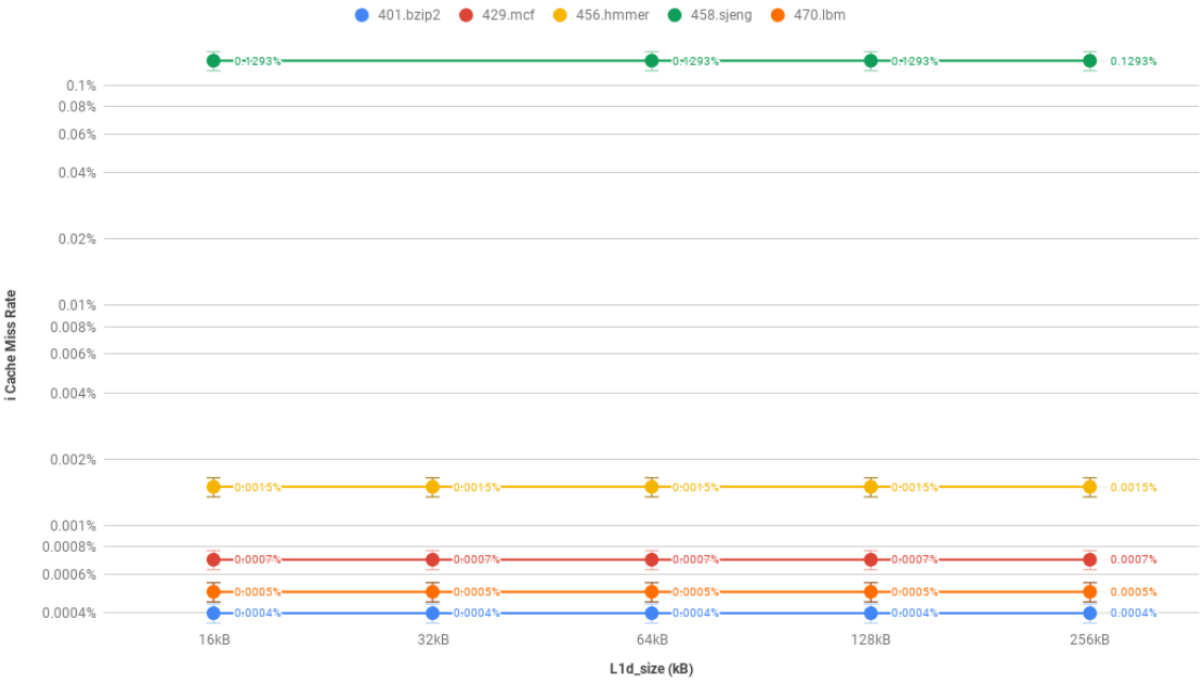
1.3 L1 D Cache Size vs Hit Rate of L1 I Cache




From the graph we observe the instruction cache hit rate for all the benchmarks when the L1 D (data) is varied. We can see that for Benchmarks 1,2,3,5 the hit rates are almost 99.9999% (approx. 100%), however for benchmark-4 , the L1 Instruction hit rate is around 99.87%. This result and above results, implies that the data supply of benchmark

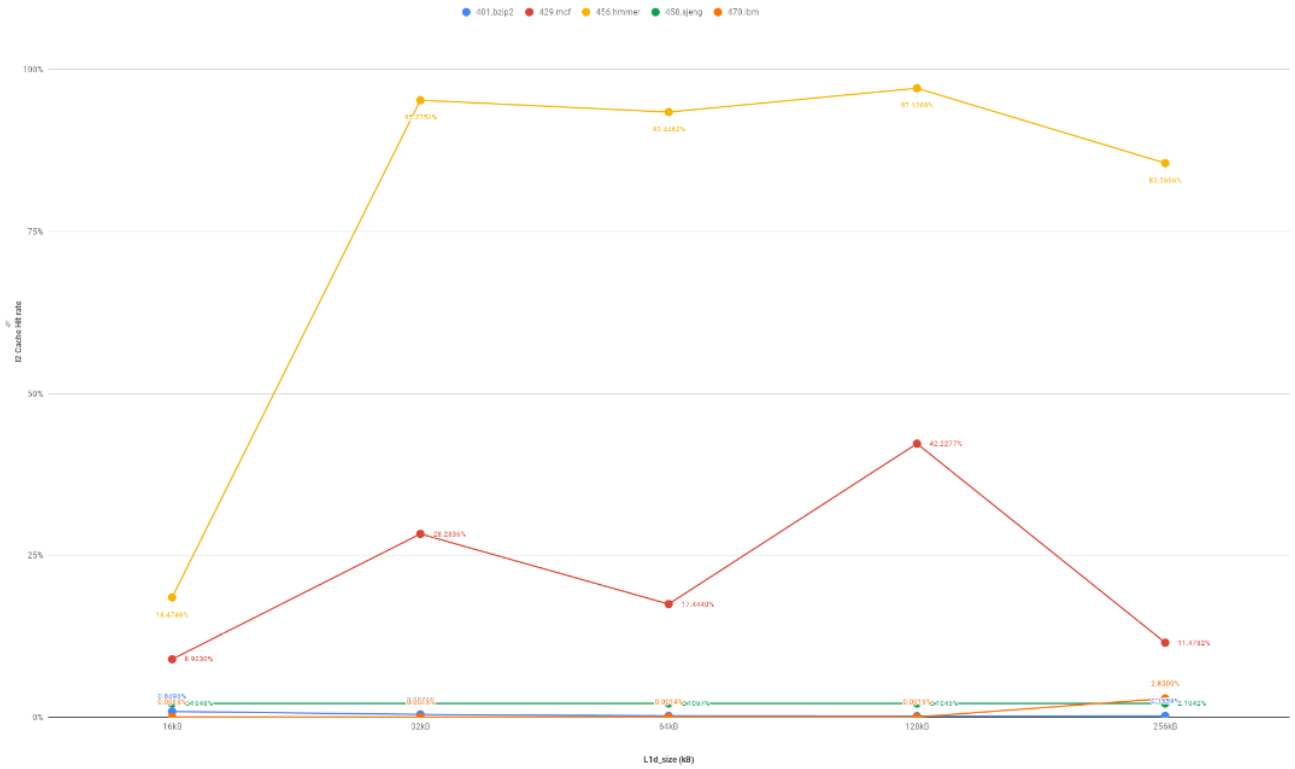
4 is not as fast as others. We can look into more results to confirm it. Also we can observe that with the change in L1d size there is only slight variation in hit rates.

### 1.4 L 1 D Cache Size vs Miss Rate of L1 I Cache



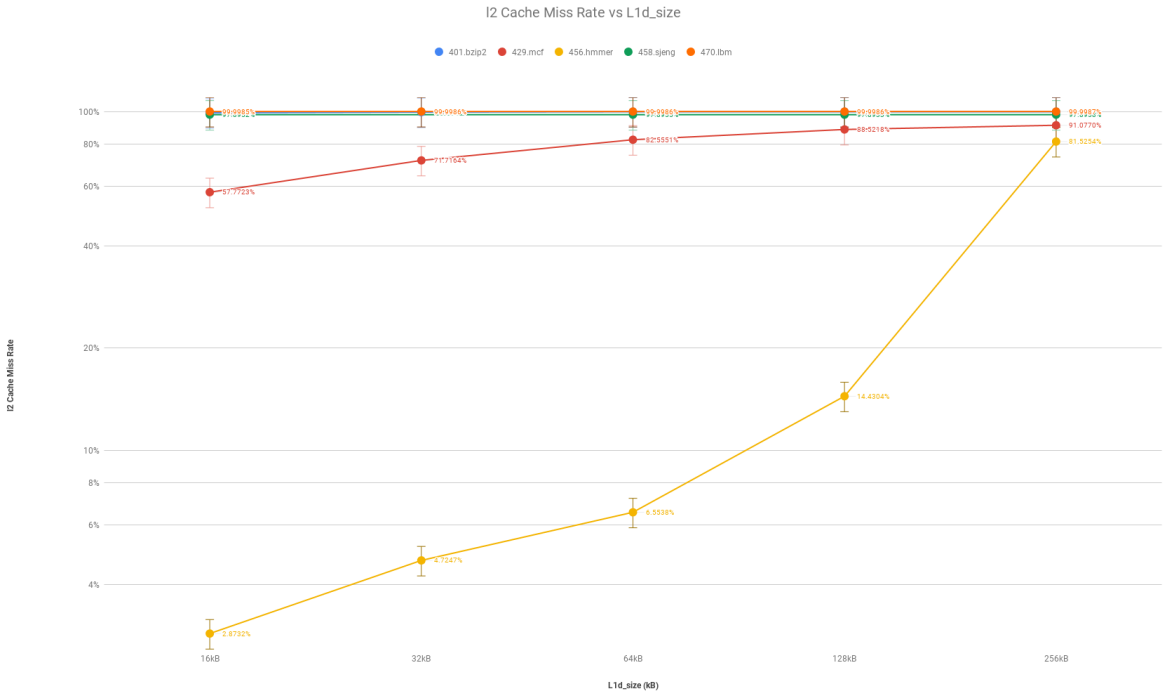

Miss Rate is (1-hit rate), thus it is just the inverse of what we discussed above. That is, for benchmarks 1,2,3 and 5 the L1 D miss rate is less. Benchmark 1 and 5 have lowest miss rates of around 0.0004 % and 0.0005%, followed by Benchmark 2 and 4 having 0.0007 % and 0.00015% respectively. The highest miss rate can be seen in Benchmark-4, around 0.1293%. Another interesting thing to be noted is that miss rates of Instructions are less than miss rates of data in L1 cache. Since we know the max capacity of L1 cache is 512kB and also the fact that d cache and i cache helps in fetching instructions and data in parallel we can say that, probably the benchmarks provided have more data items than instructions.

1.5 L1 D Cache Size vs Hit Rate of L2 Cache



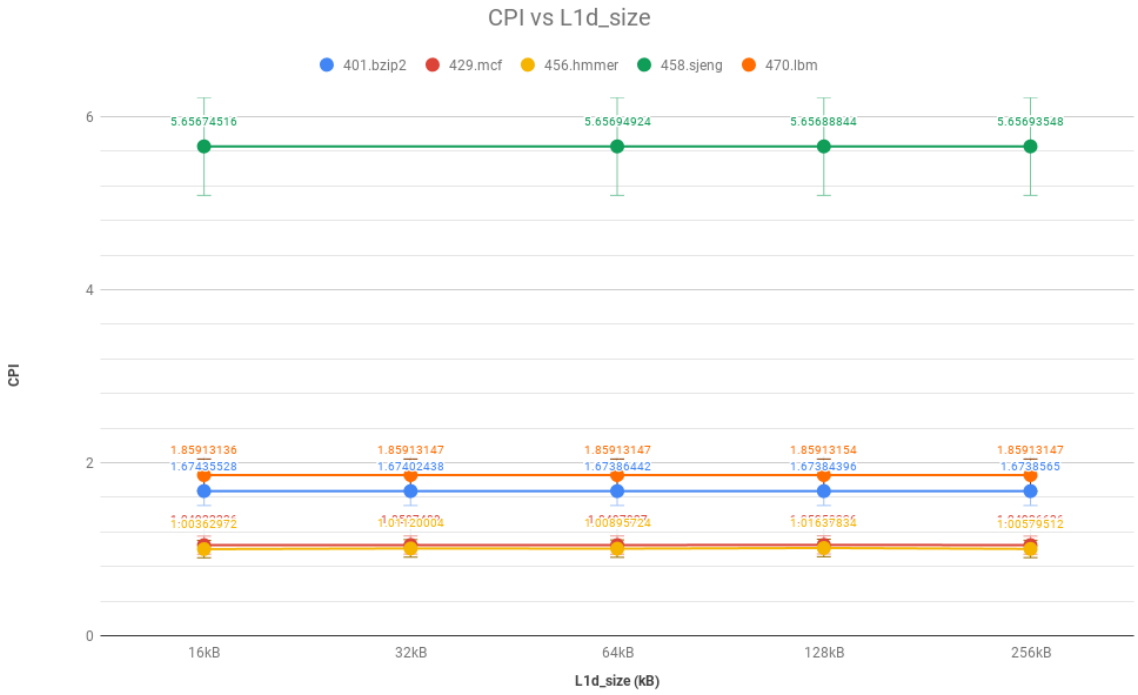

When we increase the size of L2 d , what we mean is we are increasing the size od cache which is closer to the memory. Also we do know that L1 cache is subset of L2 cache. And that L2 cache size is greater than L1 cache. Here while we are varying L1 cache L2 cache is kept constant at 2 MB. If we notice the hit rates for Benchmark 3,4 and 5 are very low approximately 0.0014%.Which implies that either the L2 d cache has already got the memory address needed by the data or either that L2 cache doesn't have the memory location yet written in its cache lines. As we have seen from L1 D size vs L1 D hit rates that , the hit rate for Benchmark 3 and 5 are high, however that of benchmark 4 is low, which implies that for benchmark 3 and 5 the memory location has already been identified in L1 D cache , however for benchmark 4 there has been more number of misses in L1 as well as in L2 cache.

## 1.6 L 1 D Cache Size vs Miss Rate of L2 Cache




Miss rate is 1- Hit rate. As explained above, the inverse is applied for the miss rates. Note: A higher miss rate implies the fraction of memory access was not found in L2 cache. Also note that here the miss rate increases as we increase the L1 D size (for benchmark 3), this is because of the fact that L1 D cache has already achieved the memory location for the data values in cacheline.

1.7 L 1 D Cache Size vs CPI




We notice that with the increase in L1 D size, there is not much difference in Clocks needed per instruction. This is because the number of cache lines are still the same, and thus the total number of instructions that can be executed in L1 D cache per clock cycle is still the same. Also CPI was calculated by noting down the hit numbers, miss numbers of L1d , L1 I and L2 cache. Also note the total number of instructions were : 100000000

# 2 L1 I Cache Size

***Commands used are as follows:***

***Benchmark-1: 401.bzip2***

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/swati/Project1_SPEC-master/401.bzip2/src/benchmark -I 100000000 -o /home/swati/Project1_SPEC-master/401.bzip2/data/input.program --caches --l2cache –l1i_size =16kB

***Benchmark-2: 429.mcf***

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/swati/Project1_SPEC-master/429.mcfsrc/benchmark -I 100000000 -o /home/swati/Project1_SPEC-master/429.mcf/data/inp.in --caches --l2cache –l1i_size =16kB

***Benchmark-3: 456.hmmer***

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/vshah3/Project1_SPEC-master/456.hmmer/src/benchmark -I 100000000 -o /home/vshah3/Project1_SPEC-master/456.hmmer/data/bombesin.hmm --caches --l2cache --l1i_size=16kB

***Benchmark-4: 458.sjeng***

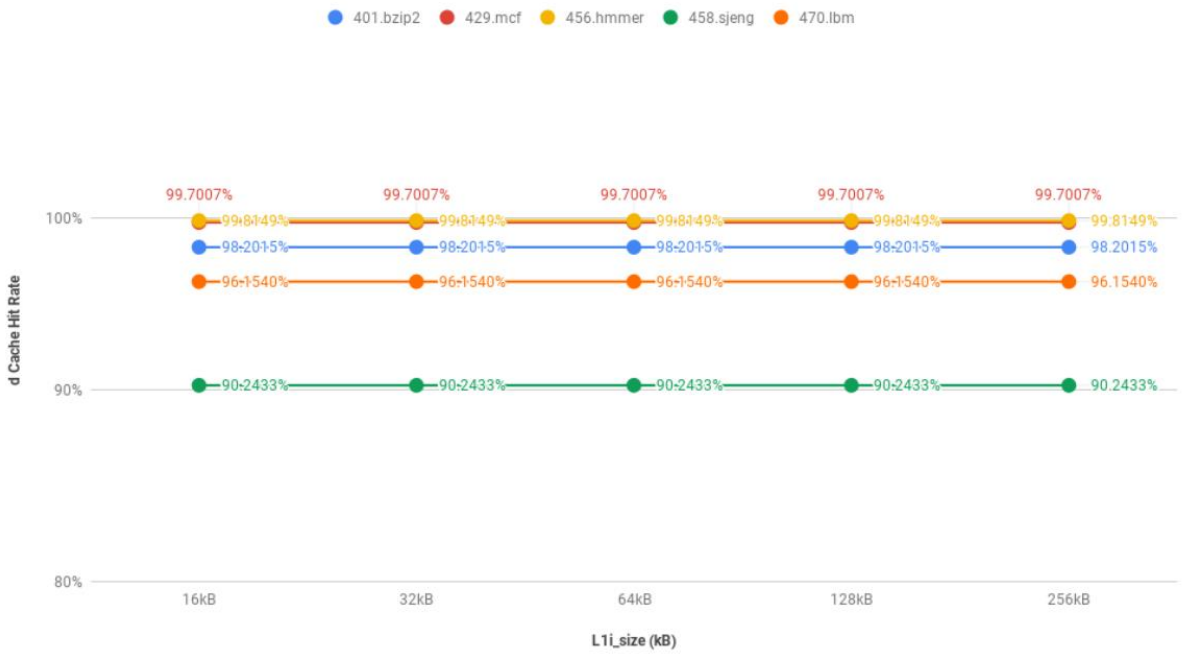
sudo ./build/X86/gem5.opt configs/example/se.py -c /home/swati/Project1_SPEC-master/458.sjeng/src/benchmark -I 100000000 -o /home/swati/Project1_SPEC-master/458.sjeng/data/test.txt --caches --l2cache --l1i_size=16kB

***Benchmark-5: 470.lbm***

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/vshah3/Project1_SPEC-master/470.lbm/src/benchmark -I 100000000 -o "20 reference.dat 0 1 /home/vshah3/Project1_SPEC-master/470.lbm/data/100_100_130_cf_a.of" --caches --l2cache --l1i_size=16kB

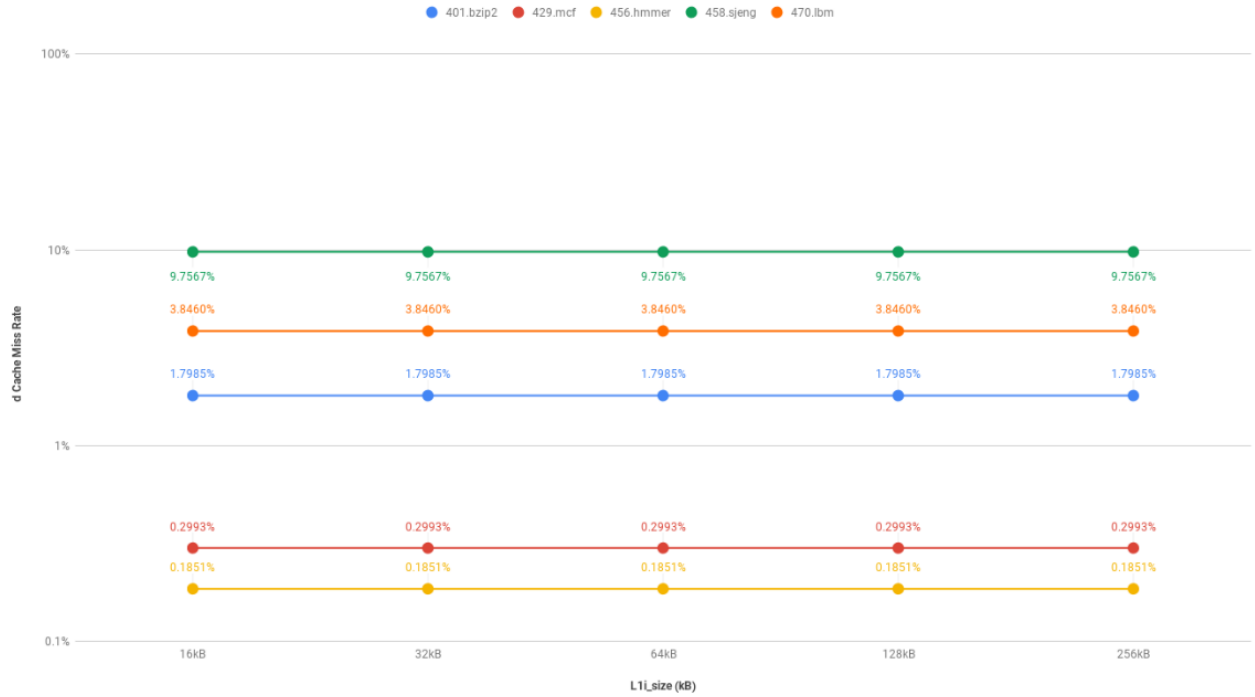
***Tested Values of l1i_size :*** 16kB, 32kB, 64kB, 128kB, 256kB

2.1 L1 I Cache Size vs Hit Rate of L1 D Cache



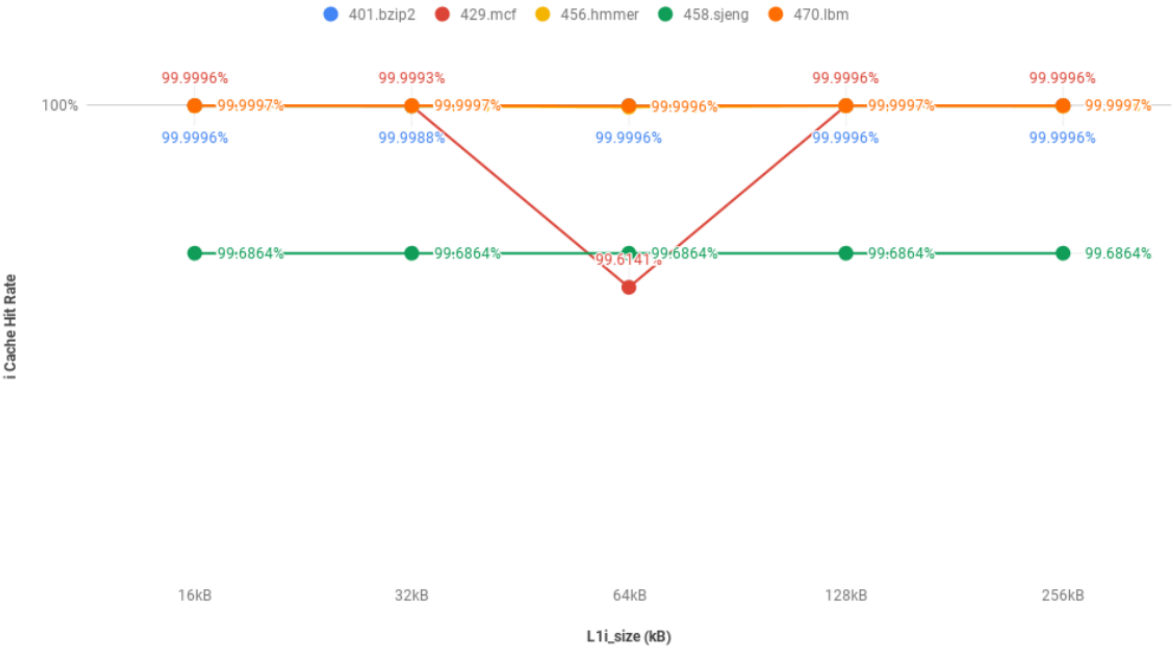

As we know L1 cache is divided into L1 d and L1 I , we noted that the sum of both L1 d and L1 I at any point would not be more than 512kB. While varying L1 I cache size , we kept L1 d size constant at 64kB. As we notice that the hit rate of L1 d cache is constant as we vary the L1i size, since we know that a cache line parallelly fetches instructions and data using dcache and icache,and we notice that varying I cache size hasn't affected the d hit rate, since L1 d cache has 64kB size, and a 64 byte cacheline cannot fetch more than that much data at a time.

2.2 L1 I Cache Size vs Miss Rate of L1 D Cache




As we know Miss rate is just 1-hit rate , the reason for the constant miss rate is same as we discussed above for the hit rates.

2.3 L1 I Cache Size vs Hit Rate of L1 I Cache



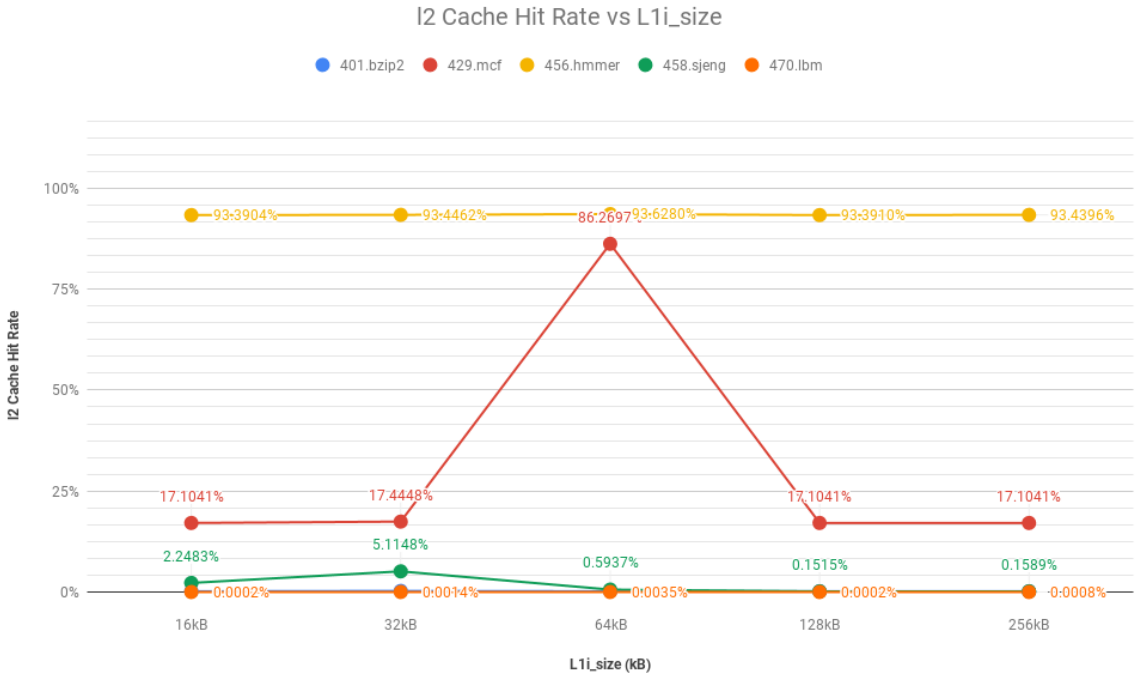

As we notice the hit rates of L1 I is quite high for all the benchmarks, most of them in the range of 99.9996%, except benchmark 4 which is around 99.6864%. These high rates imply that a large fraction of memory access were found in L1 I cache itself. Also notice that there was a deviation in hit rate when we went from 32kB L1i size to 64 kB L1i size for benchmark 2, we

should keep this in notice, since in further graphs we may notice that the miss in L1 cache might be covered by a hit in L2 cache

### 2.4 L 1 I Cache Size vs Miss Rate of L1 I Cache

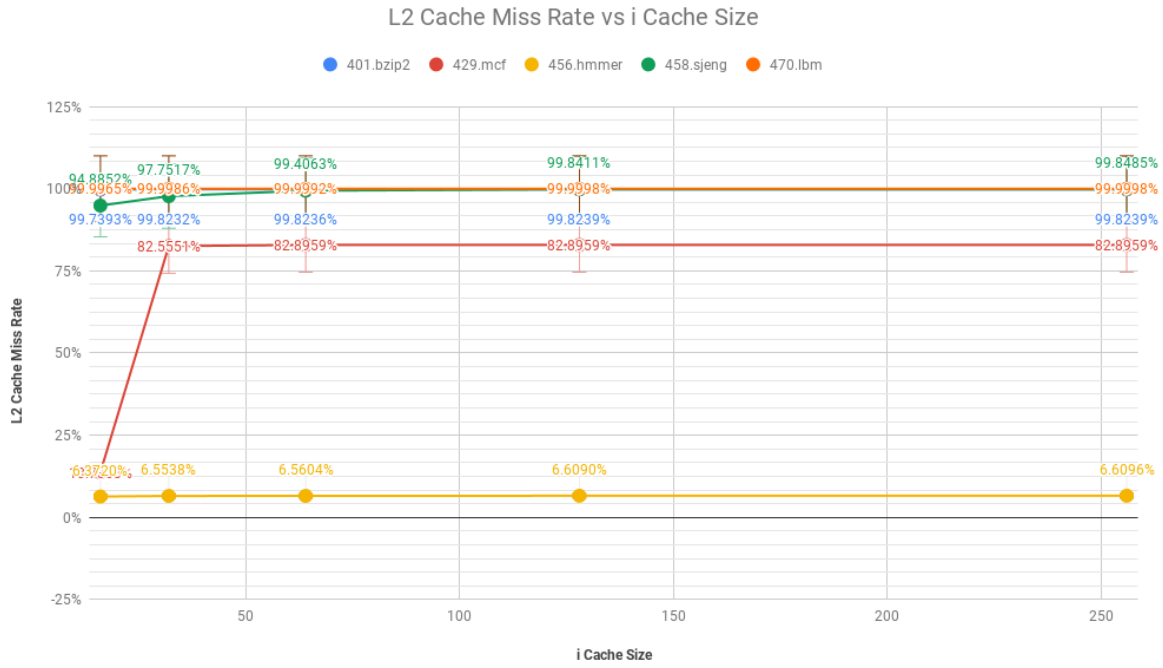

As we can see as we increase the size of I cache the I cache miss rates decreases for both benchmark 4 and benchmark 2. For rest it remains same.

### 2.5 L1 I Cache Size vs Hit Rate of L2 Cache



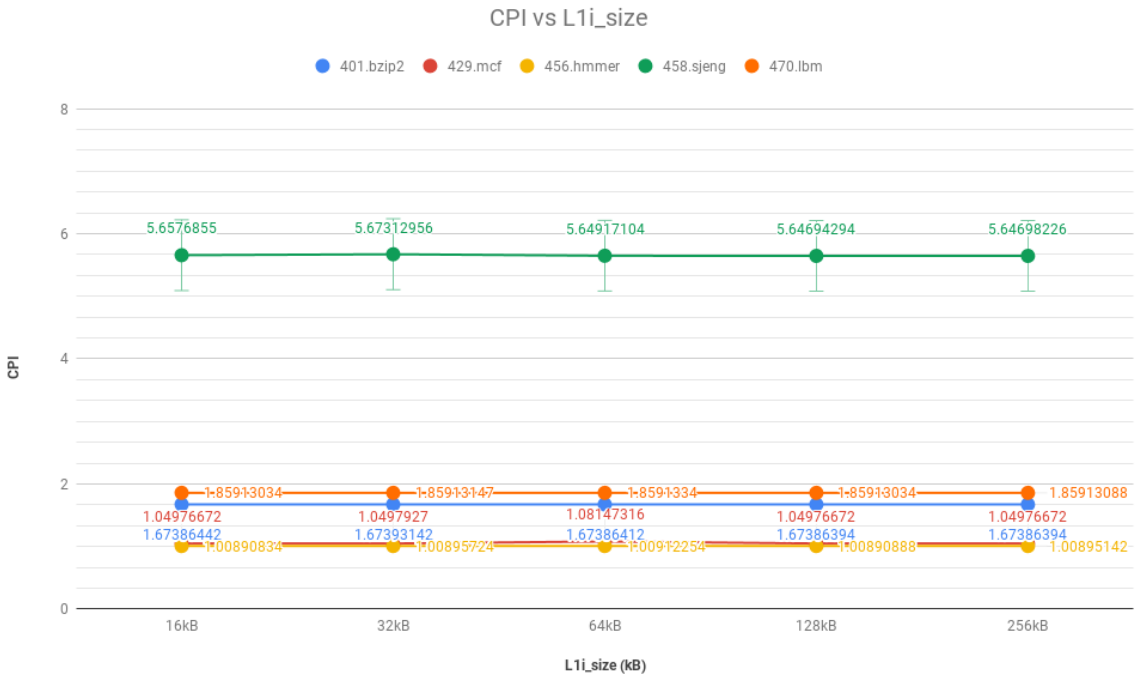
Here we notice that on varying L1i size, the L2 cache hit rate varies slightly,notice that the miss rates for benchmark 3 and 2 were quite high in L1 cache, however L2 cache compensates it by providing a high hit rate. Also notice that for benchmark-2 the hit rate is high when the cache size is 64kB, as we had seen in previous graph there was a high miss rate in L1i cache for 64kB which again gets compensated by a hit in L2. However this would definitely slower down the processor, since the higher hit rate at cache near the motherboard was missed.

2.6 L 1 I Cache Size vs Miss Rate of L2 Cache




Here we notice that on varying L1i size, the L2 cache miss rate varies slightly,notice that the miss rates for benchmark 3 and 2 were quite high in L1 cache, however L2 cache compensates it by providing a high hit rate. Also notice that for benchmark-2 the miss rate is less when the cache size is 64kB, as we had seen in previous graph there was a high miss rate in L1i cache for 64kB which again gets compensated by a hit in L2.

2.7 L 1 I Cache Size vs CPI




We notice that with the increase in L1 I size, there is not much difference in Clocks needed per instruction. This is because the number of cache lines are still the same, and thus the total number of instructions that can be executed in L1 I cache per clock cycle is still the same. Moreover, notice that the CPI when we changed L1 D size (graph 1.7)and now when we change L1 I size are the same. Also CPI was calculated by noting down the hit numbers, miss numbers of L1d , L1 I and L2 cache. Also note the total number of instructions were : 100000000

# 3 L2 Size

***Commands used are as follows:***

***Benchmark-1: 401.bzip2***

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/swati/Project1_SPEC-master/401.bzip2/src/benchmark -I 100000000 -o /home/swati/Project1_SPEC-master/401.bzip2/data/input.program --caches --l2cache –l2_size =2MB

***Benchmark-2: 429.mcf***

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/swati/Project1_SPEC-master/429.mcfsrc/benchmark -I 100000000 -o /home/swati/Project1_SPEC-master/429.mcf/data/inp.in --caches --l2cache – l2_size =2MB

***Benchmark-3: 456.hmmer***

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/vshah3/Project1_SPEC-master/456.hmmer/src/benchmark -I 100000000 -o /home/vshah3/Project1_SPEC-master/456.hmmer/data/bombesin.hmm --caches --l2cache -- l2_size =2MB

***Benchmark-4: 458.sjeng***

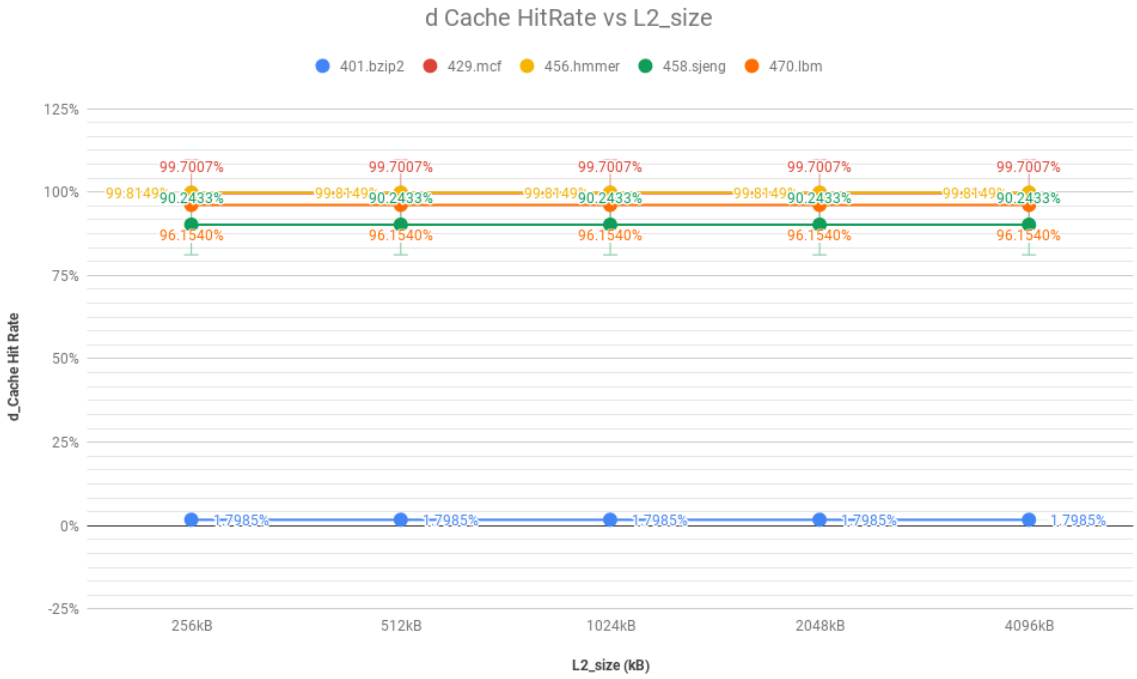
sudo ./build/X86/gem5.opt configs/example/se.py -c /home/swati/Project1_SPEC-master/458.sjeng/src/benchmark -I 100000000 -o /home/swati/Project1_SPEC-master/458.sjeng/data/test.txt --caches --l2cache -- l2_size =2MB

***Benchmark-5: 470.lbm***

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/vshah3/Project1_SPEC-master/470.lbm/src/benchmark -I 100000000 -o "20 reference.dat 0 1 /home/vshah3/Project1_SPEC-master/470.lbm/data/100_100_130_cf_a.of" --caches --l2cache -- l2_size =2MB

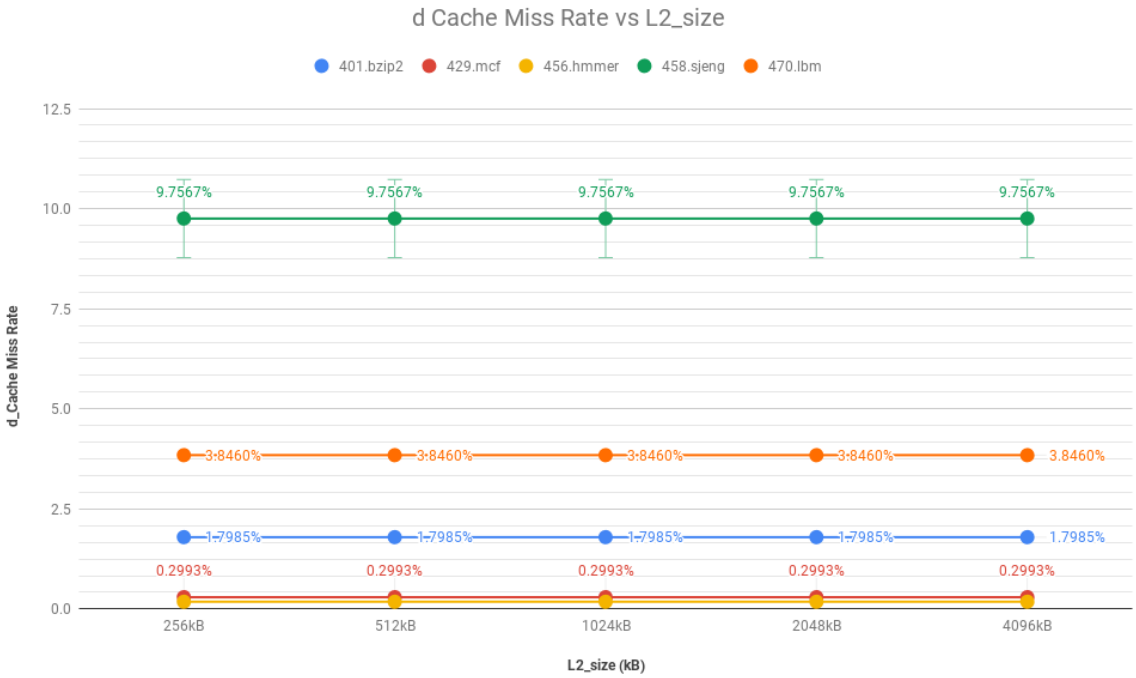
***Tested Values of l2_size :*** 256kB, 512Kb, 1MB, 2MB, 4MB

3.1 L2 Cache Size vs Hit Rate of L1 D Cache



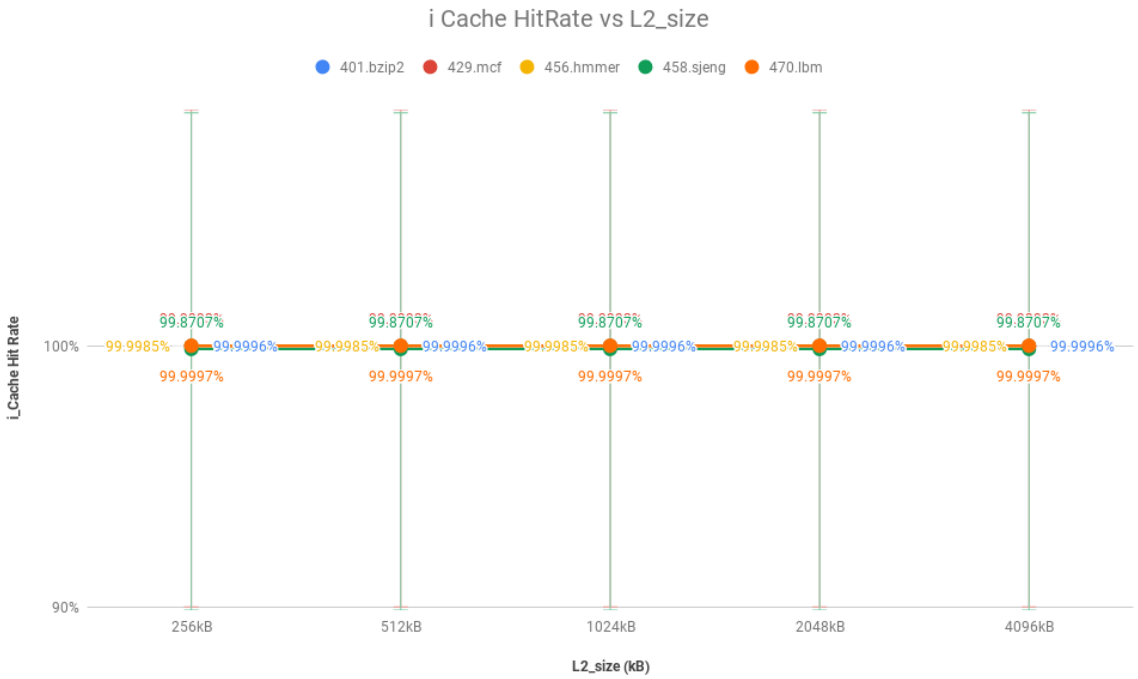

As we can see on varying L2 size (25kB, 512kB, 1MB, 2MB, 4MB), there is no change in the hit rate of L1 d cache. Which is practically true too, if we recall the memory hierarchy architecture, we would find L2 lies below L1, thus increasing or decreasing L2 size is not going to effect the hit rates of L1 D cache

3.2 L2 Cache Size vs Miss Rate of L1 D Cache



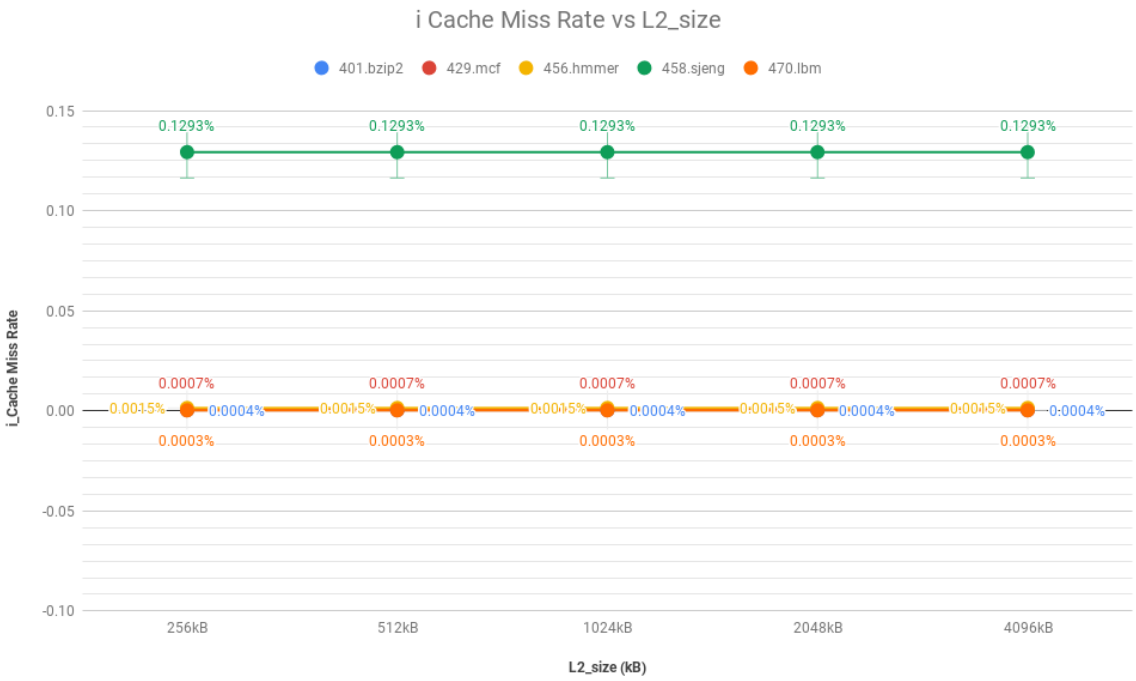

As we have mentioned earlier as well miss rate is 1-hit rate, here we can see again that varying L2 size has no effect on the miss rate of L1 cache

3.3 L2 Cache Size vs Hit Rate of L1 I Cache



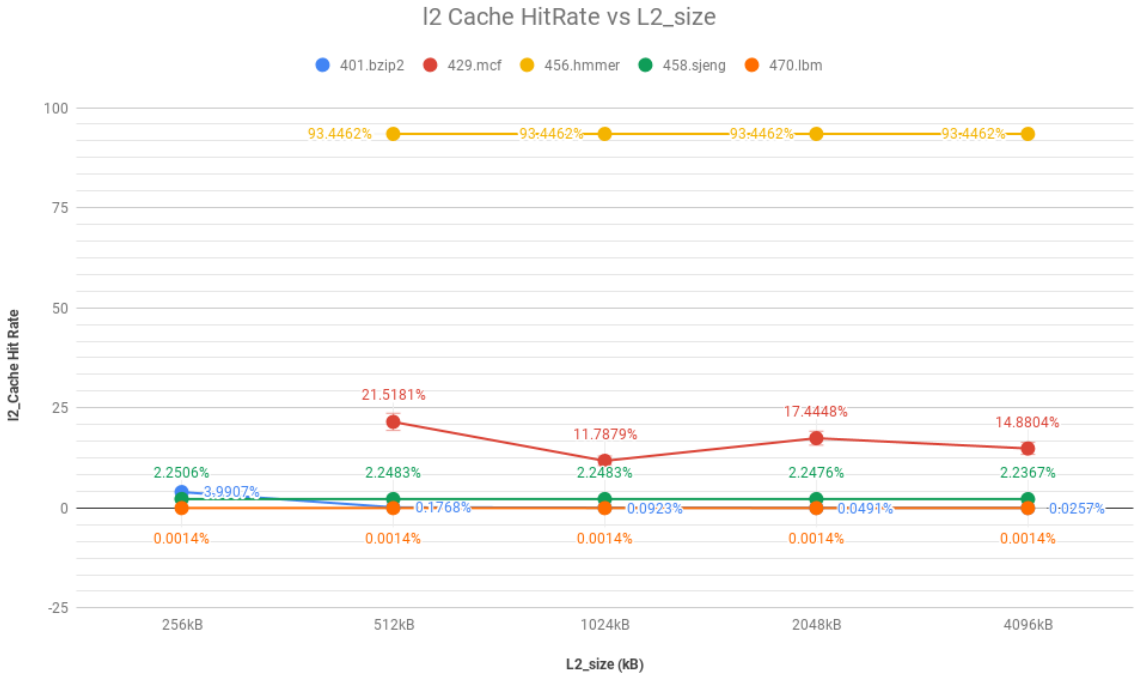

As we can see on varying L2 size (25kB, 512kB, 1MB, 2MB, 4MB), there is no change in the hit rate of L1 i cache. Which is practically true too, if we recall the memory hierarchy architecture, we would find L2 lies below L1, thus increasing or decreasing L2 size is not going to effect the hit rates of L1 i cache

### 3.4 L2 Cache Size vs Miss Rate of L1 I Cache



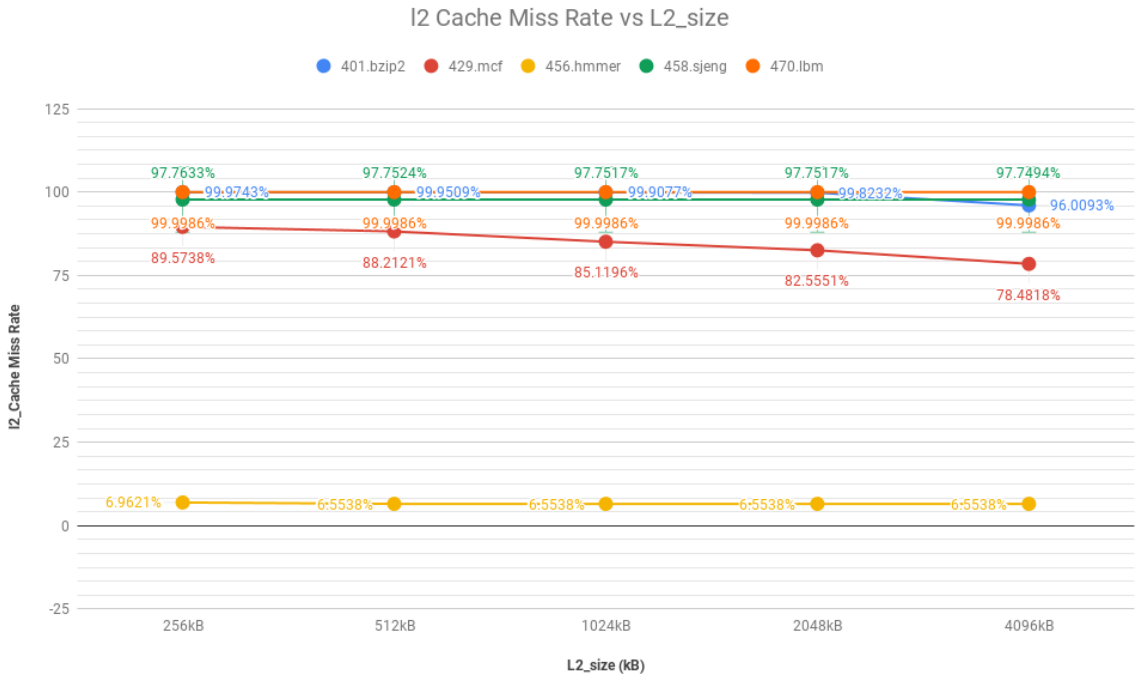

As we have mentioned earlier as well miss rate is 1-hit rate, here we can see again that varying L2 size has no effect on the miss rate of L1 i cache

### 3.5 L2 Cache Size vs Hit Rate of L2 Cache



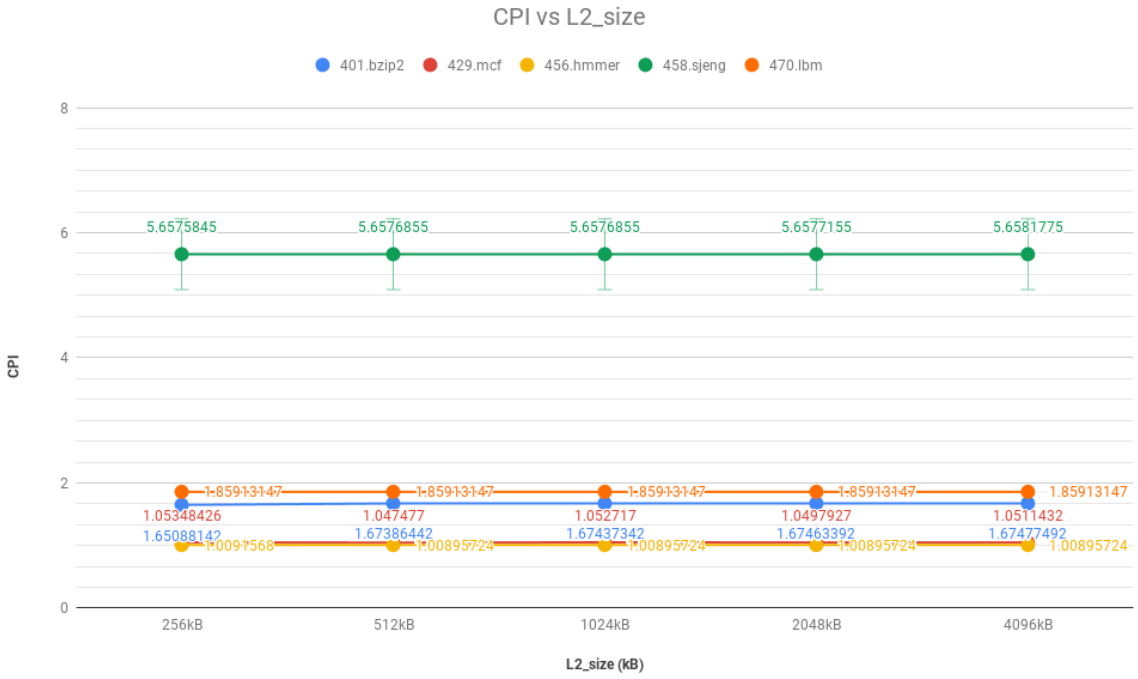

We can see that on varying L2 size (25kB, 512kB, 1MB, 2MB, 4MB) the hit rates of benchmark 1 and 2 got affected widely, also notice that low hit rates in L2 cache can either be because an already detected hit in L1 , or because the data is not yet stored in cache memory.

### 3.6 L2 Cache Size vs Miss Rate of L2 Cache




Though miss rates of most of the benchmarks are constant, however, we can see that the miss rate in benchmark 2 decreases with the increase in cache size. This proves the theory that as we increase the size of cache memory the miss rates decreases.

### 3.7 L2 Cache Size vs CPI




We notice that with the increase in L2 size, there is not much difference in Clocks needed per instruction. This is because the number of cache lines are still the same, and thus the total number of instructions that can be executed in L1 I cache per clock cycle is still the same.

Moreover, notice that the CPI when we changed L1 D size (graph 1.7), L1 I size (graph 2.7) and now when we change L2 size are approximately the same. Also CPI was calculated by noting down the hit numbers, miss numbers of L1d , L1 I and L2 cache. Also note the total number of instructions were : 100000000

# 4 L1 D Associativity

***Commands used are as follows:***

***Benchmark-1: 401.bzip2***

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/swati/Project1_SPEC-master/401.bzip2/src/benchmark -I 100000000 -o /home/swati/Project1_SPEC-master/401.bzip2/data/input.program --caches --l2cache –l1d_assoc =2

***Benchmark-2: 429.mcf***

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/swati/Project1_SPEC-master/429.mcfsrc/benchmark -I 100000000 -o /home/swati/Project1_SPEC-master/429.mcf/data/inp.in --caches --l2cache – l1d_assoc =2

***Benchmark-3: 456.hmmer***

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/vshah3/Project1_SPEC-master/456.hmmer/src/benchmark -I 100000000 -o /home/vshah3/Project1_SPEC-master/456.hmmer/data/bombesin.hmm --caches --l2cache -- l1d_assoc =2

***Benchmark-4: 458.sjeng***

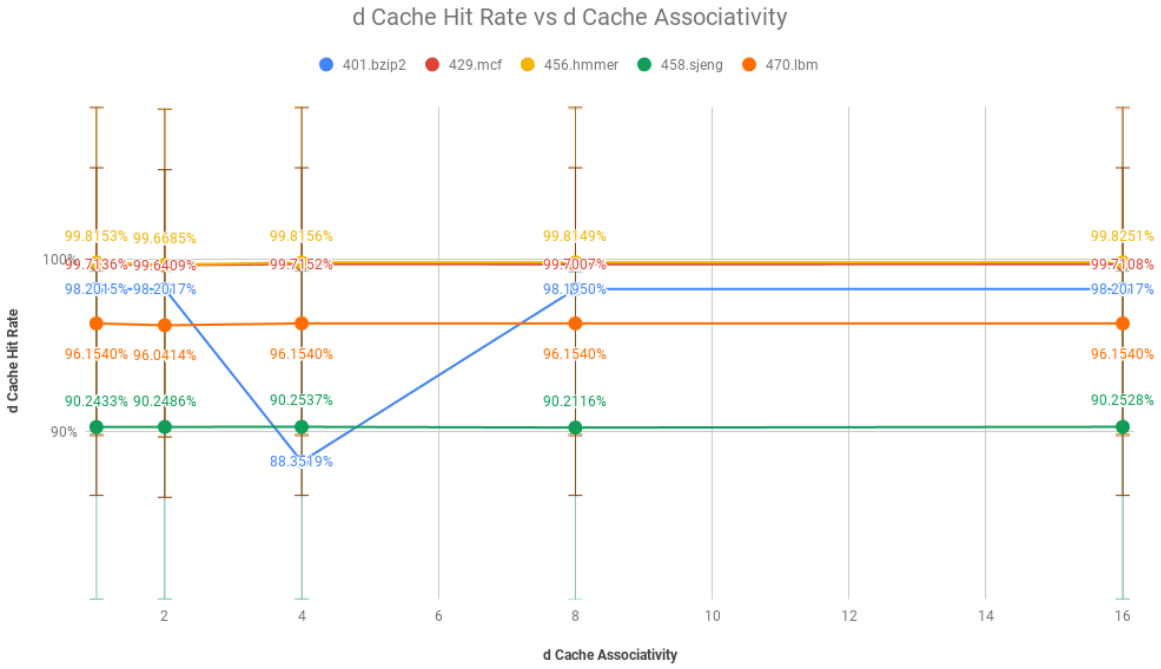
sudo ./build/X86/gem5.opt configs/example/se.py -c /home/swati/Project1_SPEC-master/458.sjeng/src/benchmark -I 100000000 -o /home/swati/Project1_SPEC-master/458.sjeng/data/test.txt --caches --l2cache -- l1d_assoc =2

***Benchmark-5: 470.lbm***

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/vshah3/Project1_SPEC-master/470.lbm/src/benchmark -I 100000000 -o “20 reference.dat 0 1 /home/vshah3/Project1_SPEC-master/470.lbm/data/100_100_130_cf_a.of” --caches --l2cache -- l1d_assoc =2

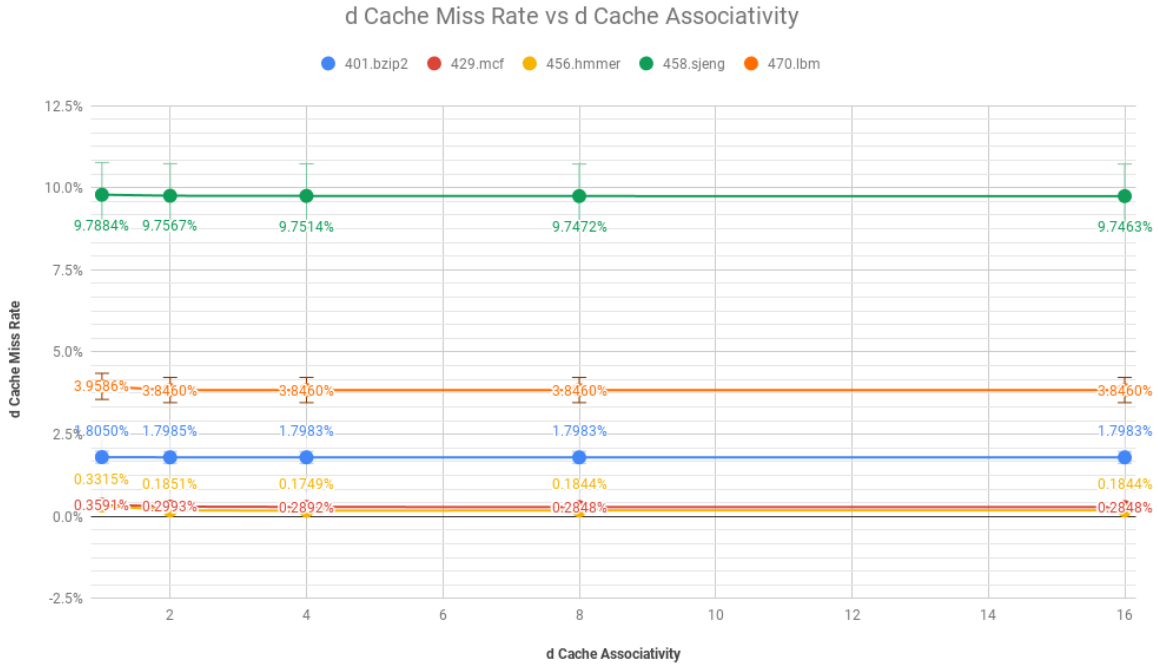
***Tested Values of l1d_assoc : 1, 2, 4, 8 , 16***

## 4.1 L1 D Associativity vs Hit Rate of L1 D Cache



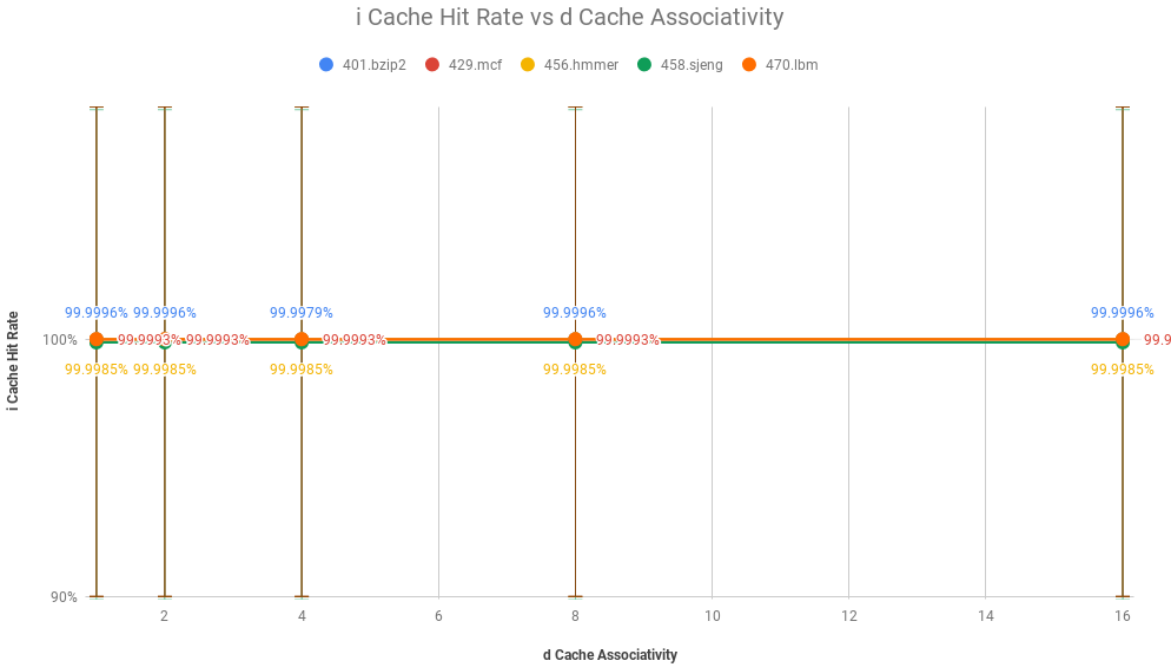

We have tested for direct mapping (or 1-way associativity), 2-way,4-way,8-way and 16 way associativity.On a general basis with the increase in associativity the hit rate should increase, however after 4-way associativity hit rate should be constant or would decrease. However in our graph we can see that most of the benchmarks have high hit rates for 4-way set associative except the benchmark 2.

## 4.2 L1 D Associativity vs Miss Rate of L1 D Cache



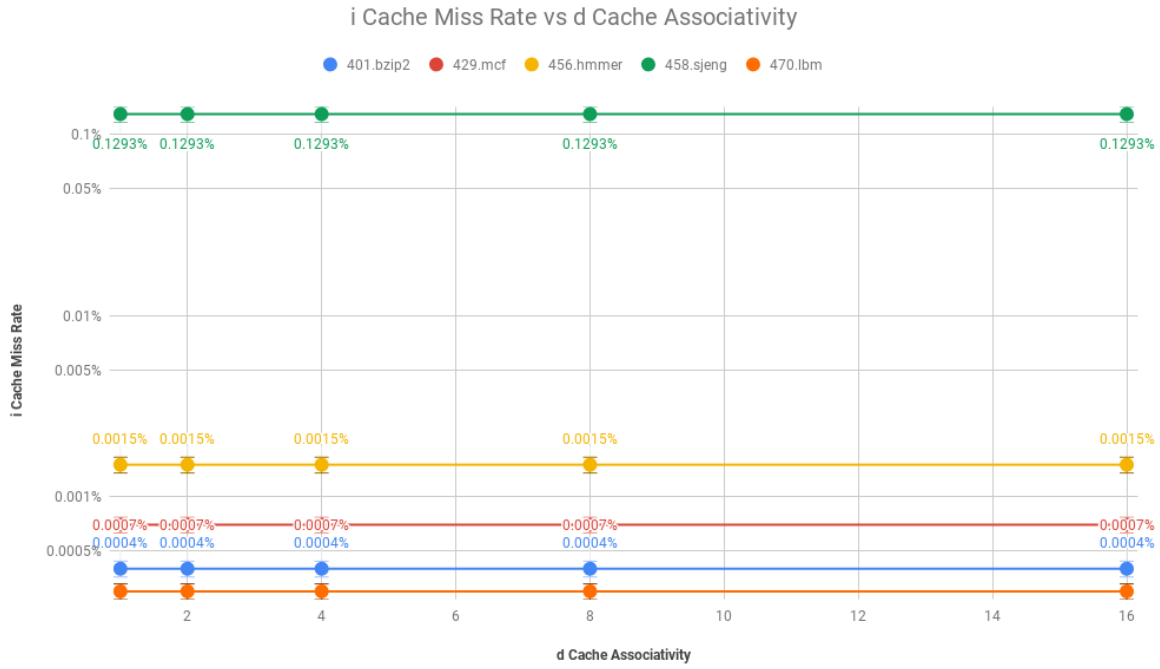
We can notice that the miss rate is almost the same, however there is slight decrease in miss rates as increase the n-way set associativity.For example for benchmark 4, the miss rate is 9.7884% when it is direct mapped(or 1 way associativity), however it drops to 9.7514% when it is 4 way associative.

4.3 L1 D Associativity vs Hit Rate of L1 I Cache



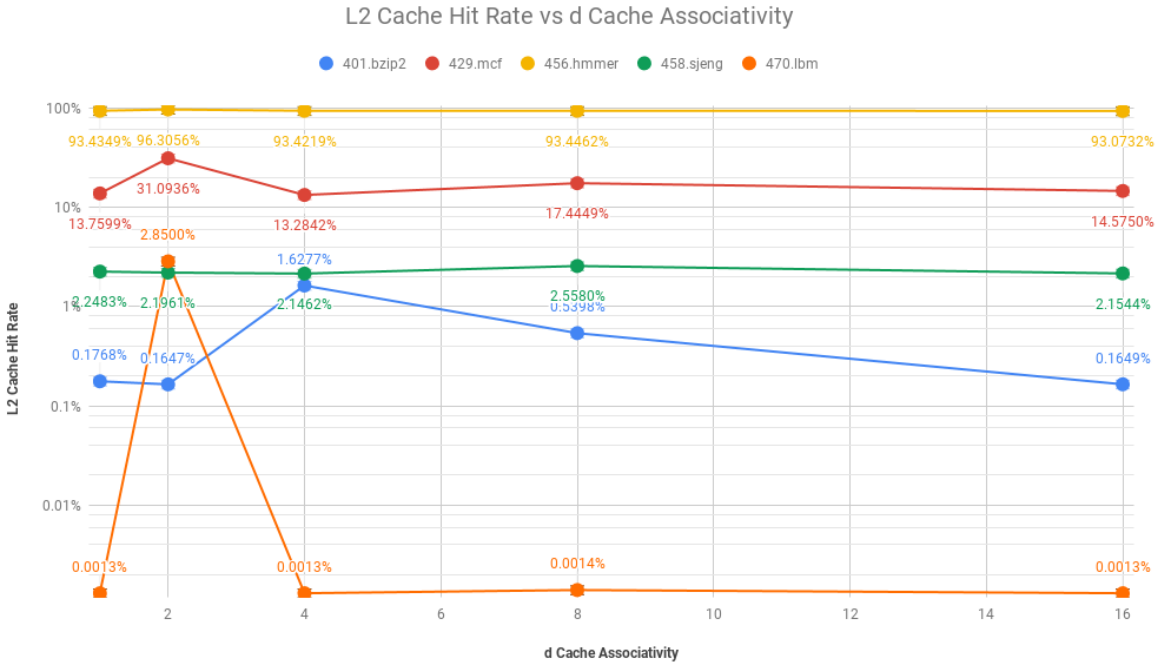

The hit rates for I cache is constant as we change the d cache associativity. As discussed before d cache and I cache fetches instructions parallelly , hence changing set associativity in d cache would not affect I cache hit rates.

## 4.4 L1 D Associativity vs Miss Rate of L1 I Cache



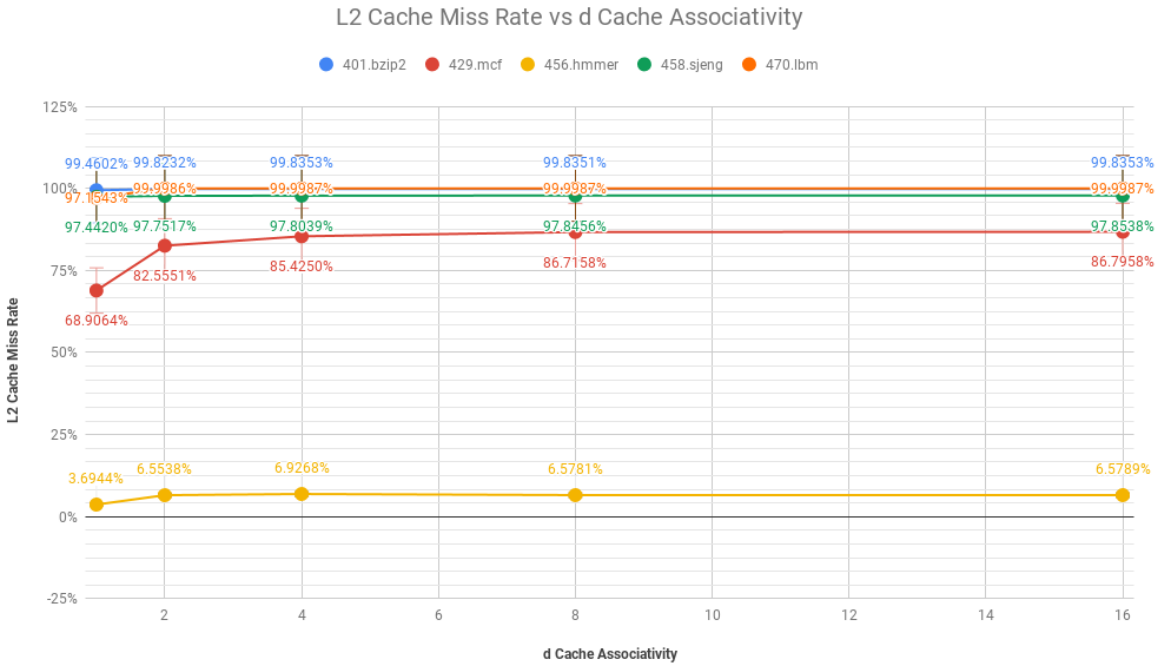

The miss rates for I cache is constant as we change the d cache associativity. As discussed before d cache and I cache fetches instructions parallelly , hence changing set associativity in d cache would not affect I cache miss rates.

## 4.5 L1 D Associativity vs Hit Rate of L2 Cache



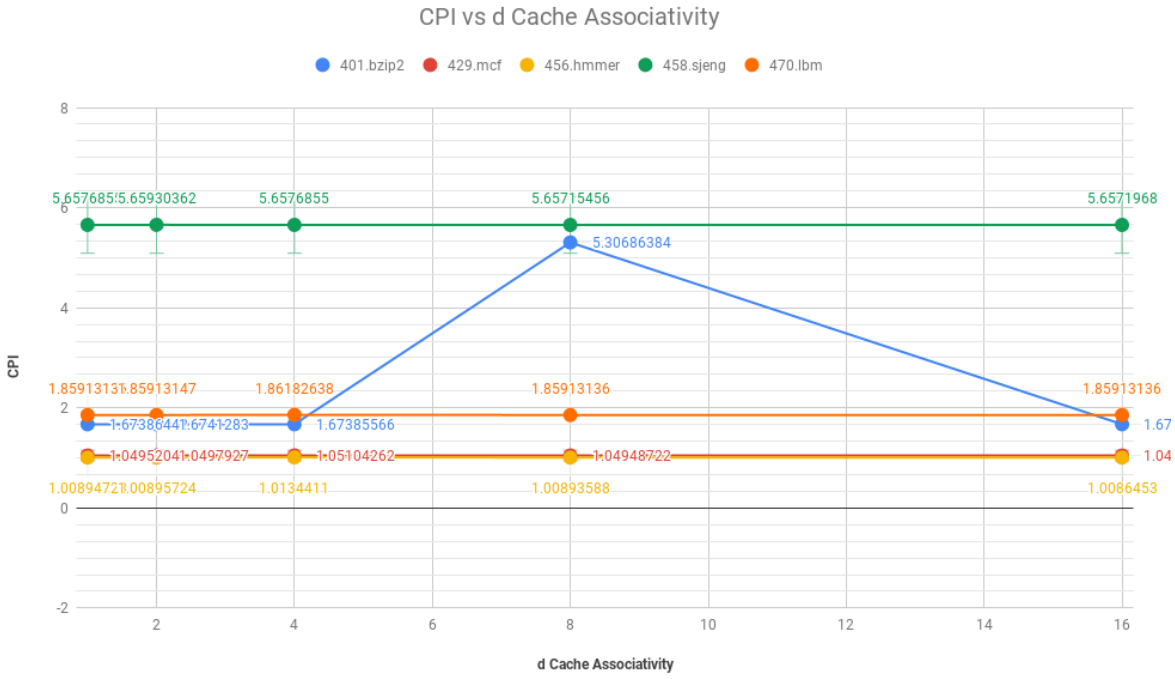

We can see that the varying L1 d associativity does affect L2 hit rates at few points such as, wen L1 d assoc is 2, the hit rate mostly is greater than in other cases.

### 4.6 L1 D Associativity vs Miss Rate of L2 Cache




We can notice d cache associativity has a very slight impact on the miss rates of L2 cache.

### 4.7 L1 D Associativity vs CPI




W can notice that there is slight variation in CPI, when we change the associativity. As you can see at 8-way associativity for benchmark-1, the total CPI needed is around 5.306 compared to what it had at 4-way set (1.67). This is probably because when we increase the number of sets means, checking more places, which would indeed increase the time and clocks needed per instruction.

# 5 L1 I Associativity

***Commands used are as follows:***

***Benchmark-1: 401.bzip2***

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/swati/Project1_SPEC-master/401.bzip2/src/benchmark -I 100000000 -o /home/swati/Project1_SPEC-master/401.bzip2/data/input.program --caches --l2cache –l1i_assoc =2

***Benchmark-2: 429.mcf***

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/swati/Project1_SPEC-master/429.mcfsrc/benchmark -I 100000000 -o /home/swati/Project1_SPEC-master/429.mcf/data/inp.in --caches --l2cache – l1i_assoc =2

***Benchmark-3: 456.hmmer***

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/vshah3/Project1_SPEC-master/456.hmmer/src/benchmark -I 100000000 -o /home/vshah3/Project1_SPEC-master/456.hmmer/data/bombesin.hmm --caches --l2cache -- l1i_assoc =2

***Benchmark-4: 458.sjeng***

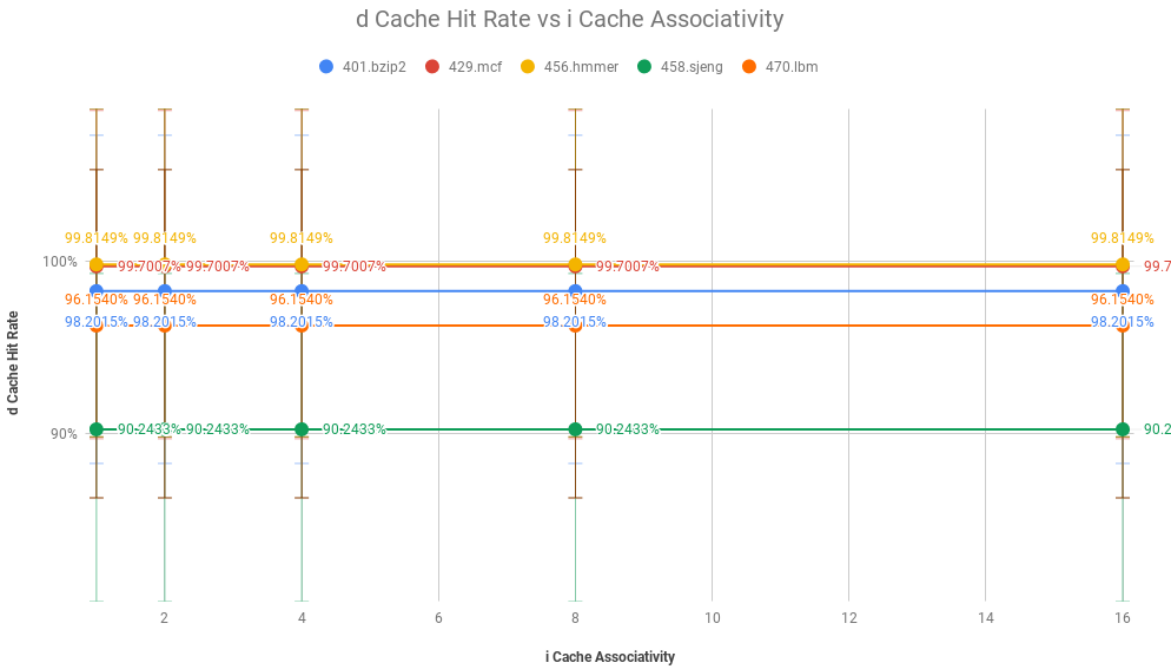
sudo ./build/X86/gem5.opt configs/example/se.py -c /home/swati/Project1_SPEC-master/458.sjeng/src/benchmark -I 100000000 -o /home/swati/Project1_SPEC-master/458.sjeng/data/test.txt --caches --l2cache -- l1i_assoc =2

***Benchmark-5: 470.lbm***

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/vshah3/Project1_SPEC-master/470.lbm/src/benchmark -I 100000000 -o “20 reference.dat 0 1 /home/vshah3/Project1_SPEC-master/470.lbm/data/100_100_130_cf_a.of” --caches --l2cache -- l1i_assoc =2

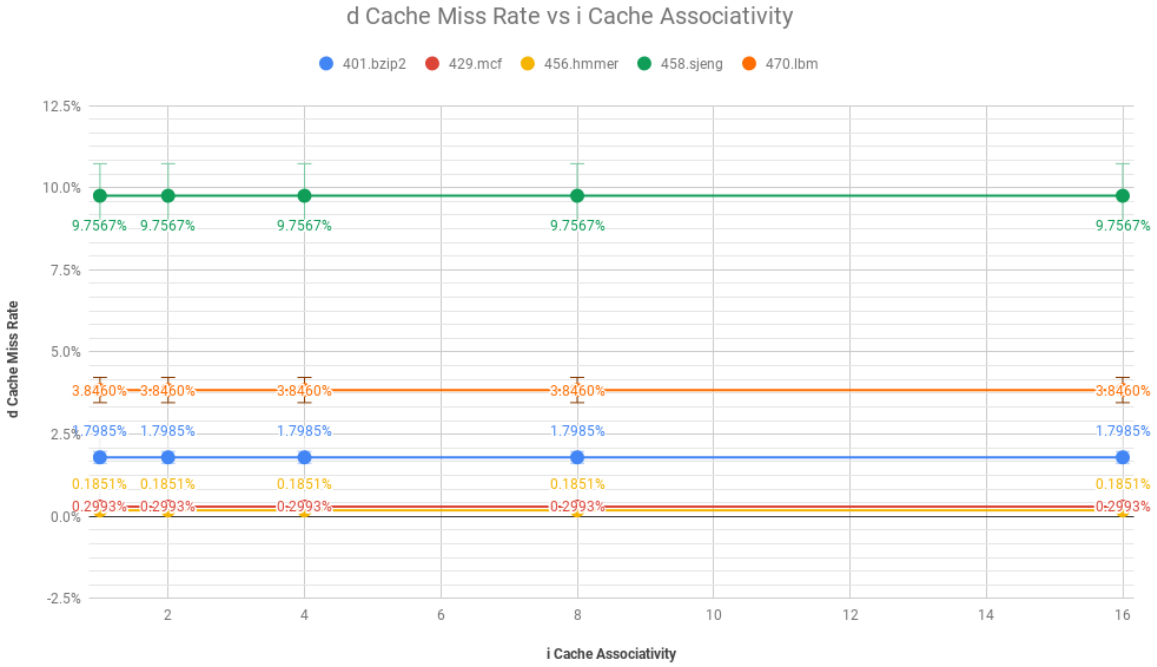
***Tested Values of l1i_assoc : 1, 2, 4, 8 , 16***

## 5.1 L1 I Associativity vs Hit Rate of L1 D Cache



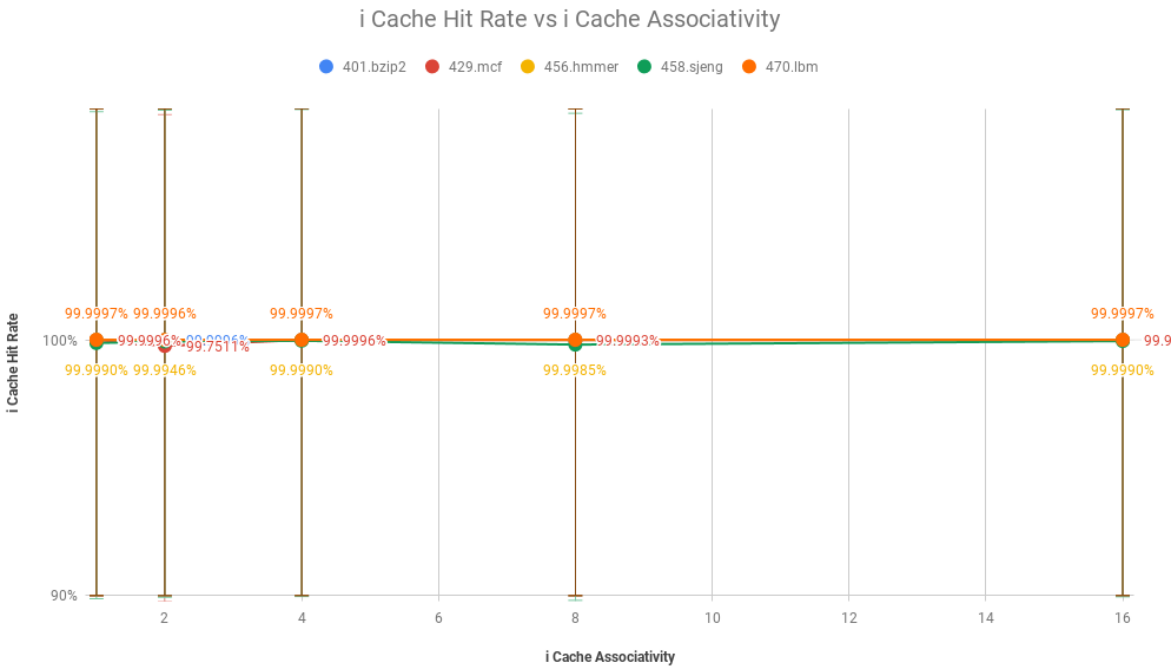
We have tested for direct mapping (or 1-way associativity), 2-way,4-way,8-way and 16 way associativity. The hit rates for D cache is constant as we change the I cache associativity. As discussed before d cache and I cache fetches instructions parallelly , hence changing set associativity in I cache would not affect D cache hit rates.

### 5.2 L1 I Associativity vs Miss Rate of L1 D Cache



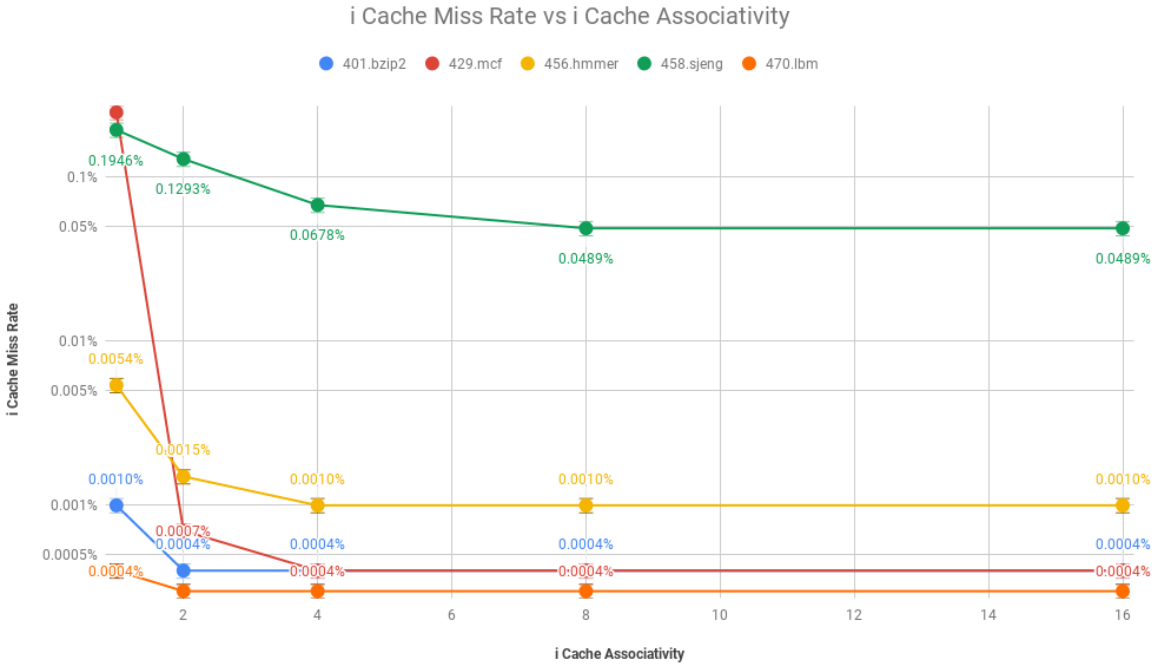

The miss rates for D cache is constant as we change the d cache associativity. As discussed before d cache and I cache fetches instructions parallelly , hence changing set associativity in I cache would not affect D cache miss rates.

5.3 L1 I Associativity vs Hit Rate of L1 I Cache



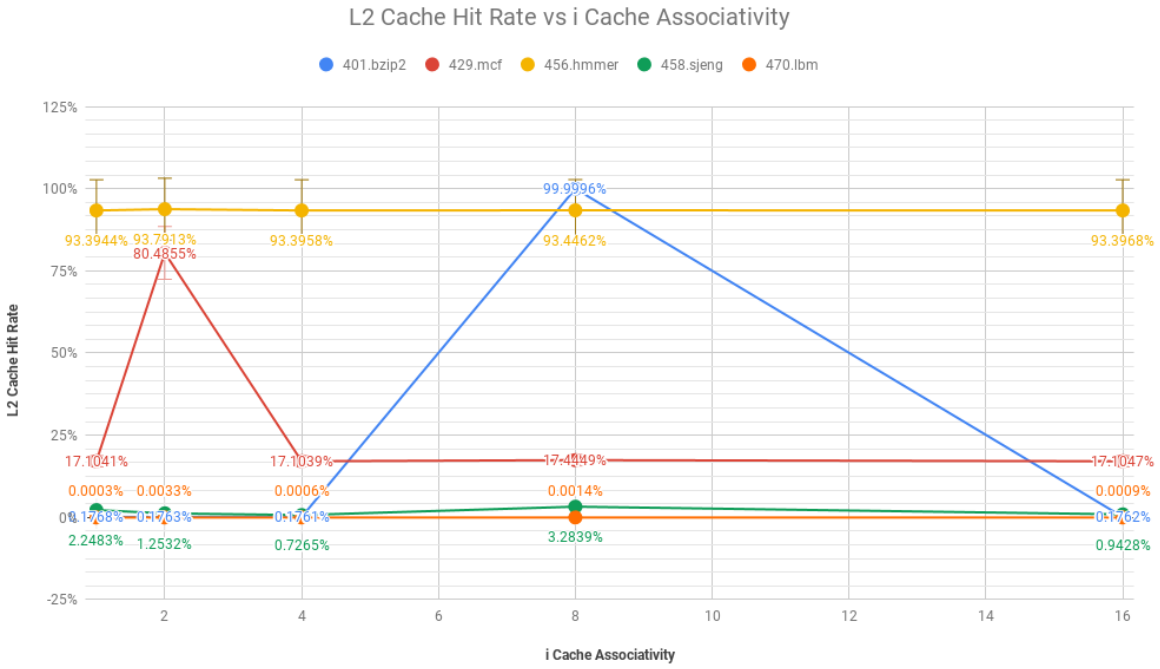

Hit rate is 1-miss rate, looking at the above graph we don't see much difference in I cache hit rate when we change the set associativity of I cache. However to get a better picture, we can look into miss rates graph below.

5.4 L1 I Associativity vs Miss Rate of L1 I Cache



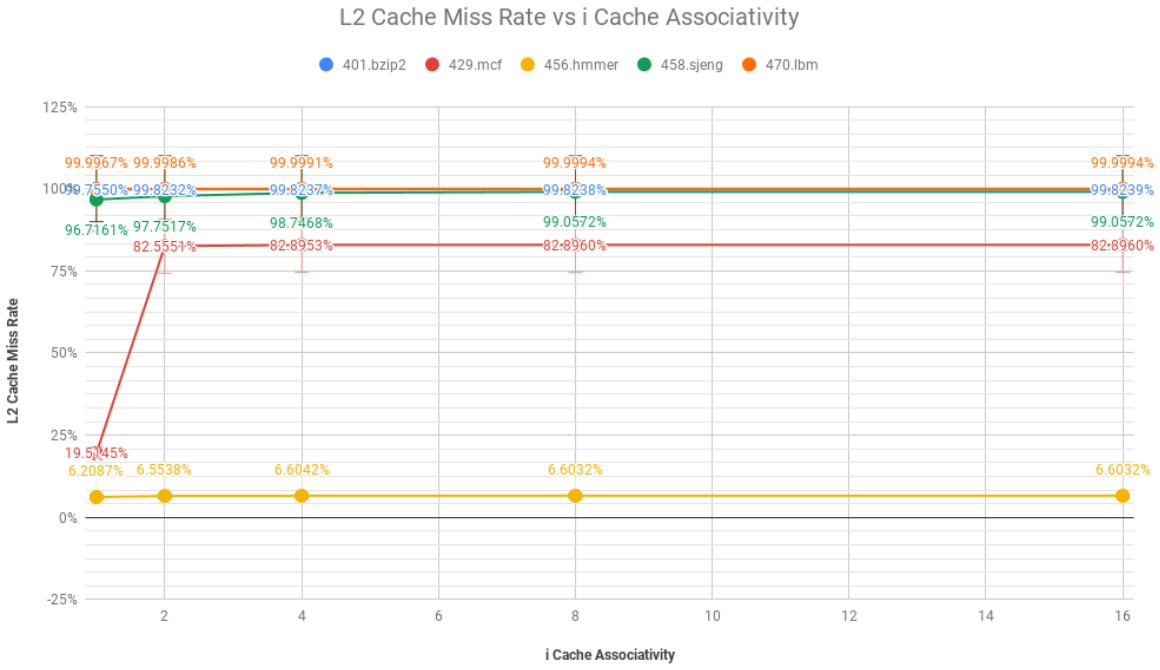

We can see that the miss rates decreases as we move from direct mapped cache to 2-way ,4-way,8-way and 16 -way caches. For example the miss rate for 1-way associativity of benchmark 1 is 0.0010% whereas that of 2-way set associativity is 0.0004%.

5.5 L1 I Associativity vs Hit Rate of L2 Cache



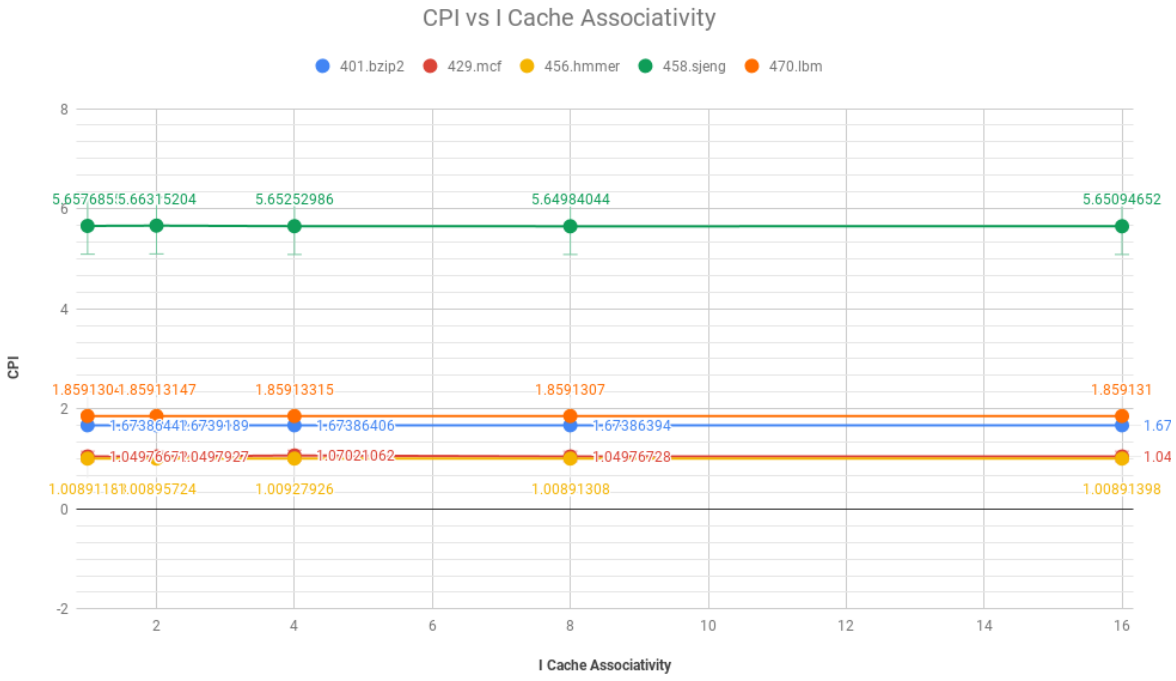

We can see that the hit rate of L2 cache slightly varies as we change the associativity in L1i cache, we can see high hit rates at 2-way associativity for benchmark2 and 8-way associativity for benchmark 1.

5.6 L1 I Associativity vs Miss Rate of L2 Cache




We can see that the miss rate of L2 cache slightly varies as we change the associativity in L1i cache.

5.7 L1 I Associativity vs CPI




We notice that with the increase in L1 I associativity, there is not much difference in Clocks needed per instruction. There are slight increase and decrease in CPIs which is because of the reason that time is consumed while searching for the data from memory for highly associative cache.

# 6 L2 Associativity

***Commands used are as follows:***

***Benchmark-1: 401.bzip2***

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/swati/Project1_SPEC-master/401.bzip2/src/benchmark -I 100000000 -o /home/swati/Project1_SPEC-master/401.bzip2/data/input.program --caches --l2cache –l2_assoc =2

***Benchmark-2: 429.mcf***

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/swati/Project1_SPEC-master/429.mcfsrc/benchmark -I 100000000 -o /home/swati/Project1_SPEC-master/429.mcf/data/inp.in --caches --l2cache – l2_assoc =2

***Benchmark-3: 456.hmmer***

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/vshah3/Project1_SPEC-master/456.hmmer/src/benchmark -I 100000000 -o /home/vshah3/Project1_SPEC-master/456.hmmer/data/bombesin.hmm --caches --l2cache – l2_assoc =2

***Benchmark-4: 458.sjeng***

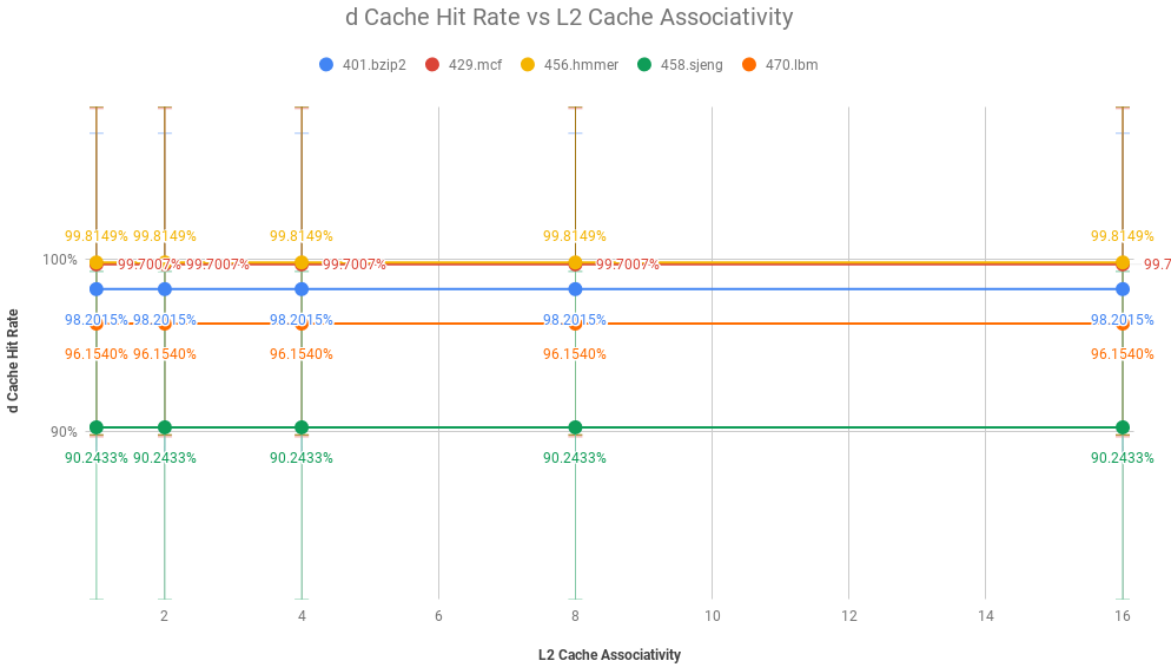
sudo ./build/X86/gem5.opt configs/example/se.py -c /home/swati/Project1_SPEC-master/458.sjeng/src/benchmark -I 100000000 -o /home/swati/Project1_SPEC-master/458.sjeng/data/test.txt --caches --l2cache – l2_assoc =2

***Benchmark-5: 470.lbm***

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/vshah3/Project1_SPEC-master/470.lbm/src/benchmark -I 100000000 -o "20 reference.dat 0 1 /home/vshah3/Project1_SPEC-master/470.lbm/data/100_100_130_cf_a.of" --caches --l2cache – l2_assoc =2

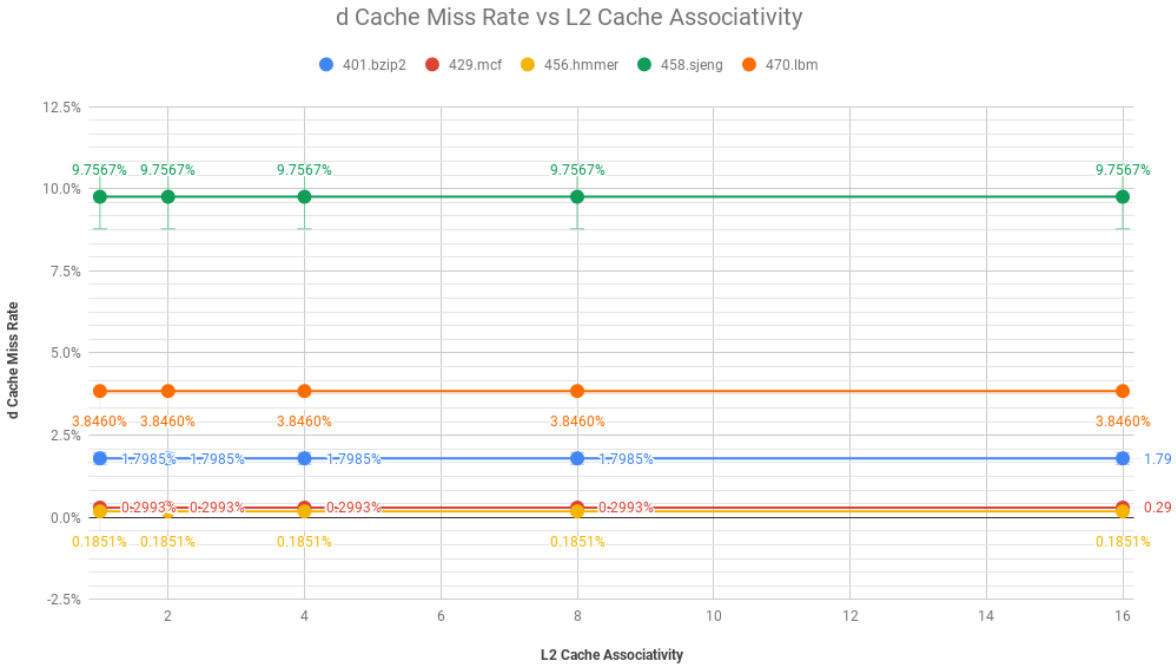
***Tested Values of l2_assoc : 1, 2, 4, 8 , 16***

6.1 L2 Associativity vs Hit Rate of L1 D Cache



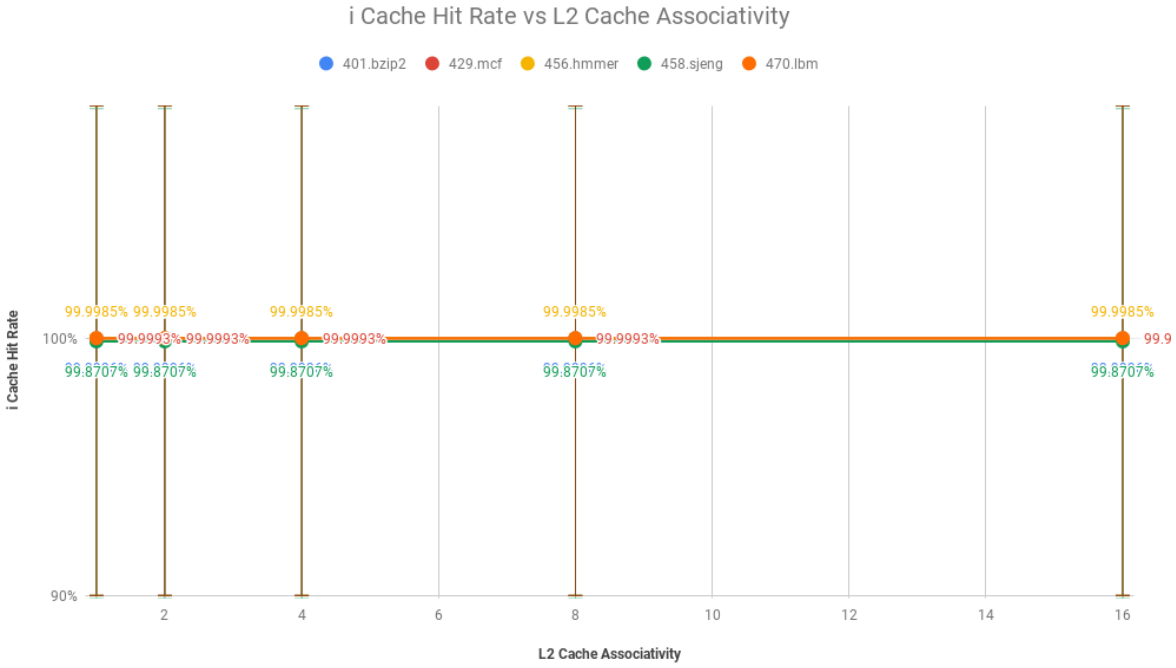

We can see that changing L2 cache associativity doesn't affect the hit rate of L1 d cache . This is because of the memory hierarchy , where L2 lies below L1, thus changing L2 would not affect the hit rates in L1.

6.2 L2 Associativity vs Miss Rate of L1 D Cache



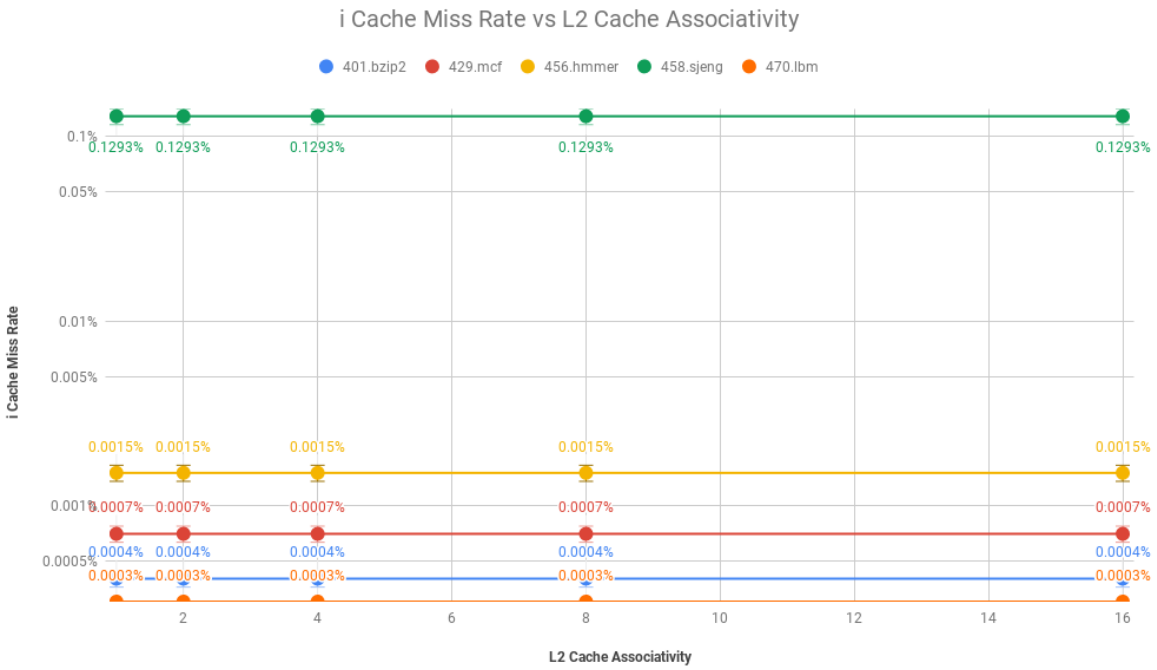

We can see that changing L2 cache associativity doesn't affect the miss rate of L1 d cache . This is because of the memory hierarchy , where L2 lies below L1, thus changing L2 would not affect the miss rates in L1.

6.3 L2 Associativity vs Hit Rate of L1 I Cache



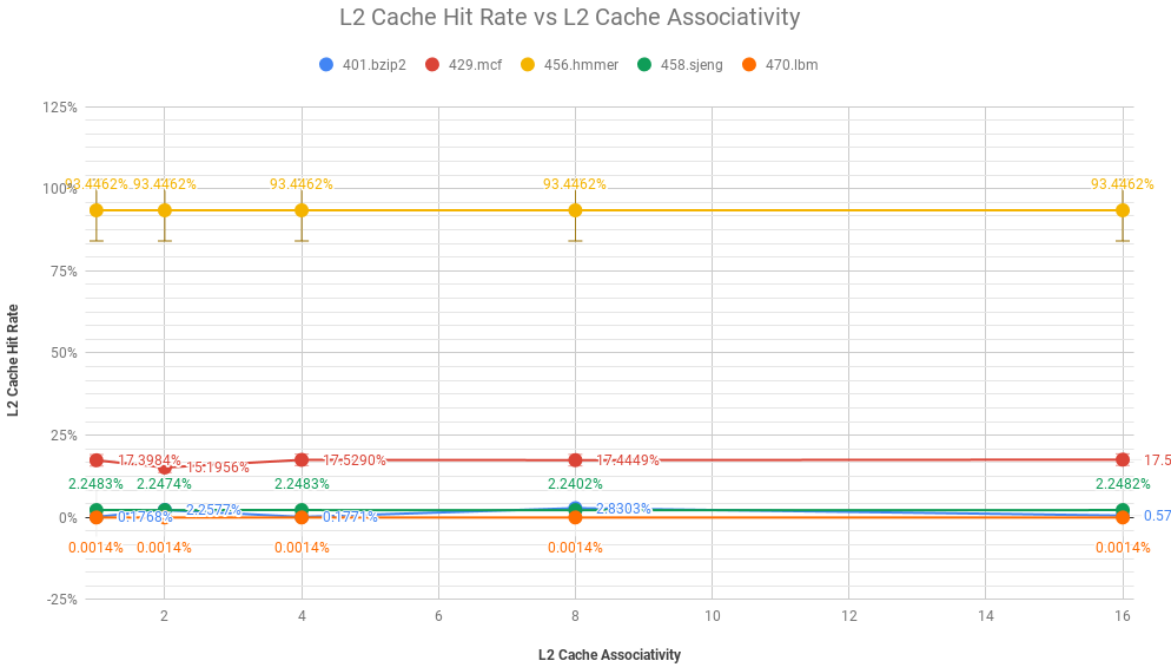

We can see that changing L2 cache associativity doesn't affect the hit rate of L1 i cache . This is because of the memory hierarchy , where L2 lies below L1, thus changing L2 would not affect the hit rates in L1.

## 6.4 L2 Associativity vs Miss Rate of L1 I Cache



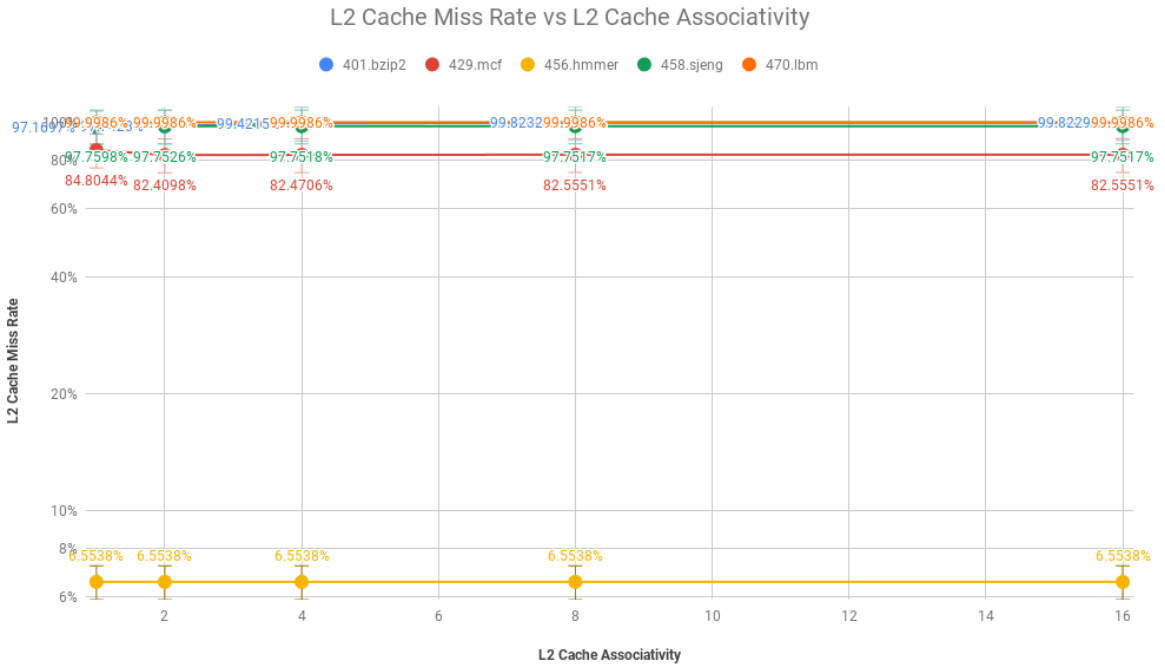

We can see that changing L2 cache associativity doesn't affect the miss rate of L1 i cache . This is because of the memory hierarchy , where L2 lies below L1, thus changing L2 would not affect the miss rates in L1.

## 6.5 L2 Associativity vs Hit Rate of L2 Cache



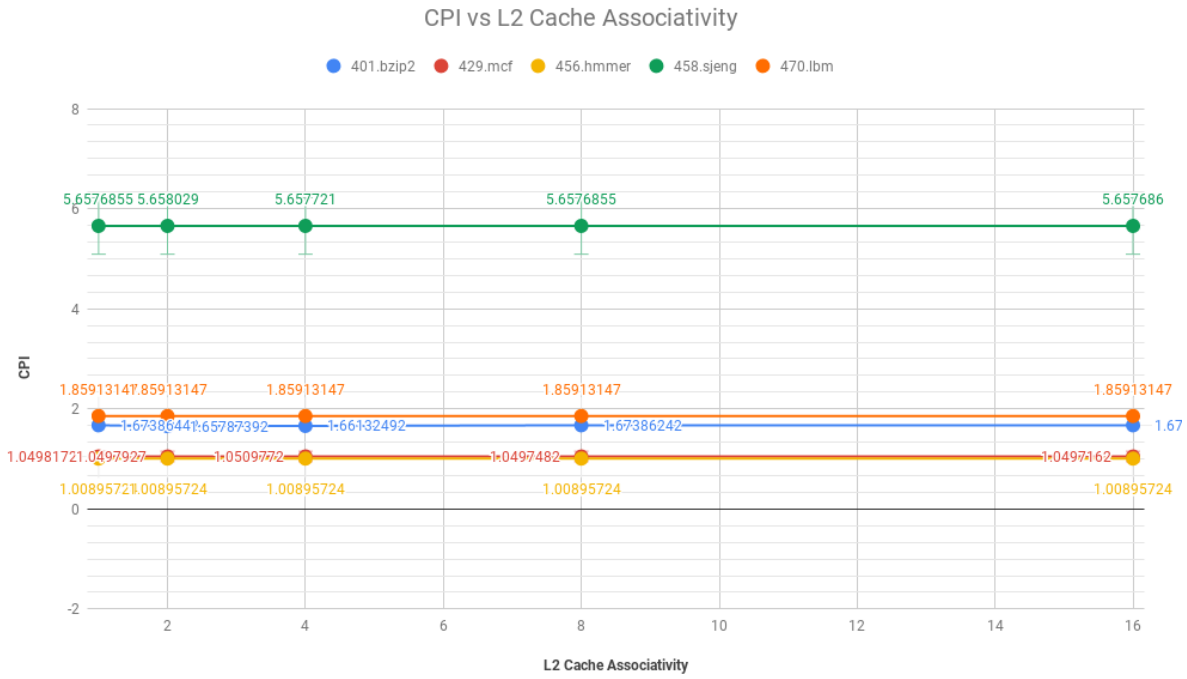

We can see that changing the L2 cache associativity has impact on the hit rates of L2 cache. We can see that for most of the benchmarks we get a high hit rate for 4-way associativity.

## 6.6 L2 Associativity vs Miss Rate of L2 Cache




In general miss rate is very high when we change the L2 associativity fir benchmarks 1,2,4, and 5. However we can see that we get even high miss rates for 1-way associativity than for 4-way or 8-way. For example, lets consider benchmark 2, it has a miss rate of 84.8044% for 1-way associativity however a lesser miss rate of 82.4706% for 4-way associativity.

## 6.7 L2 Associativity vs CPI



We can notice that on changing the L2 associativity there is not much change in the CPI. There are only slight variations which might be because of the fact that when we increase the number of sets, the processor take more time to search for the memory location.

# 7 Block Size

***Commands used are as follows:***

***Benchmark-1: 401.bzip2***

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/swati/Project1_SPEC-master/401.bzip2/src/benchmark -I 100000000 -o /home/swati/Project1_SPEC-master/401.bzip2/data/input.program --caches --l2cache –cacheline_size =64

***Benchmark-2: 429.mcf***

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/swati/Project1_SPEC-master/429.mcfsrc/benchmark -I 100000000 -o /home/swati/Project1_SPEC-master/429.mcf/data/inp.in --caches --l2cache – cacheline_size =64

***Benchmark-3: 456.hmmer***

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/vshah3/Project1_SPEC-master/456.hmmer/src/benchmark -I 100000000 -o /home/vshah3/Project1_SPEC-master/456.hmmer/data/bombesin.hmm --caches --l2cache – cacheline_size =64

***Benchmark-4: 458.sjeng***

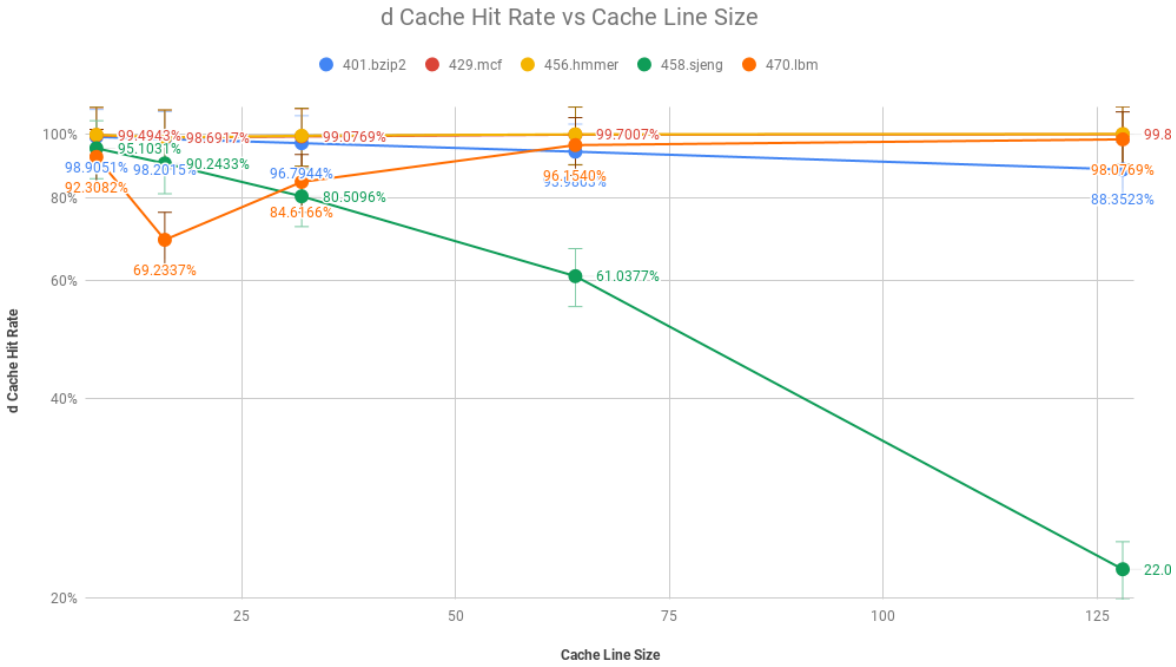
sudo ./build/X86/gem5.opt configs/example/se.py -c /home/swati/Project1_SPEC-master/458.sjeng/src/benchmark -I 100000000 -o /home/swati/Project1_SPEC-master/458.sjeng/data/test.txt --caches --l2cache – cacheline_size =64

***Benchmark-5: 470.lbm***

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/vshah3/Project1_SPEC-master/470.lbm/src/benchmark -I 100000000 -o “20 reference.dat 0 1 /home/vshah3/Project1_SPEC-master/470.lbm/data/100_100_130_cf_a.of” --caches --l2cache – cacheline_size =64

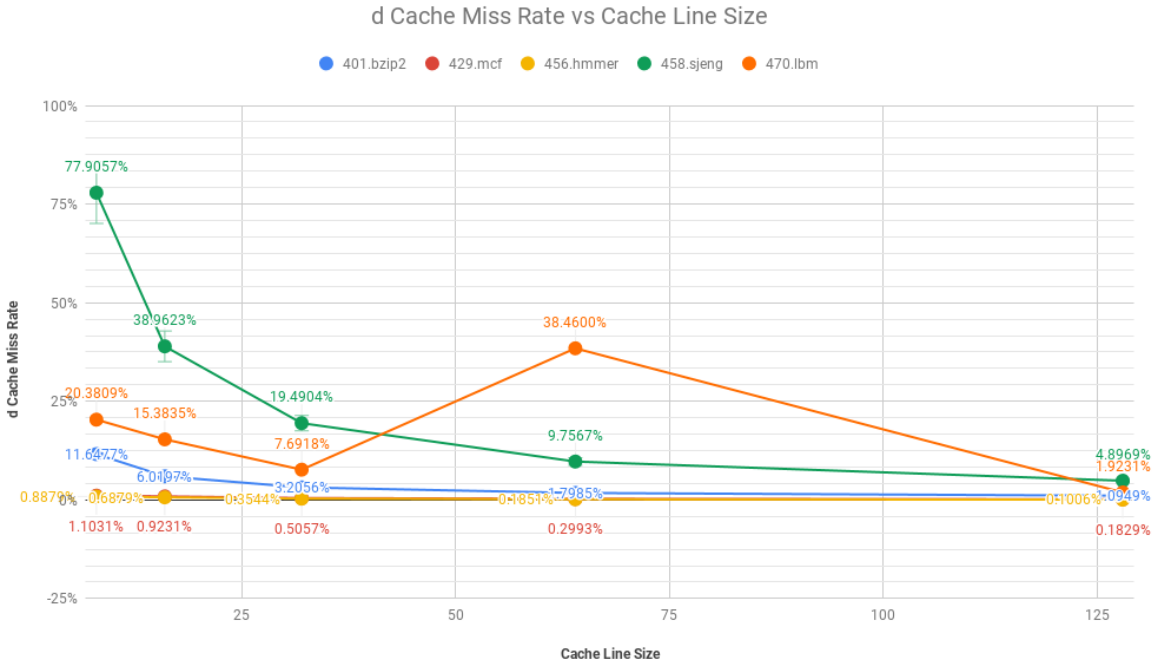
***Tested Values of cacheline_size : 8, 16, 32, 64, 128*** (All are in Bytes)

7.1 Block Size vs Hit Rate of L1 D Cache



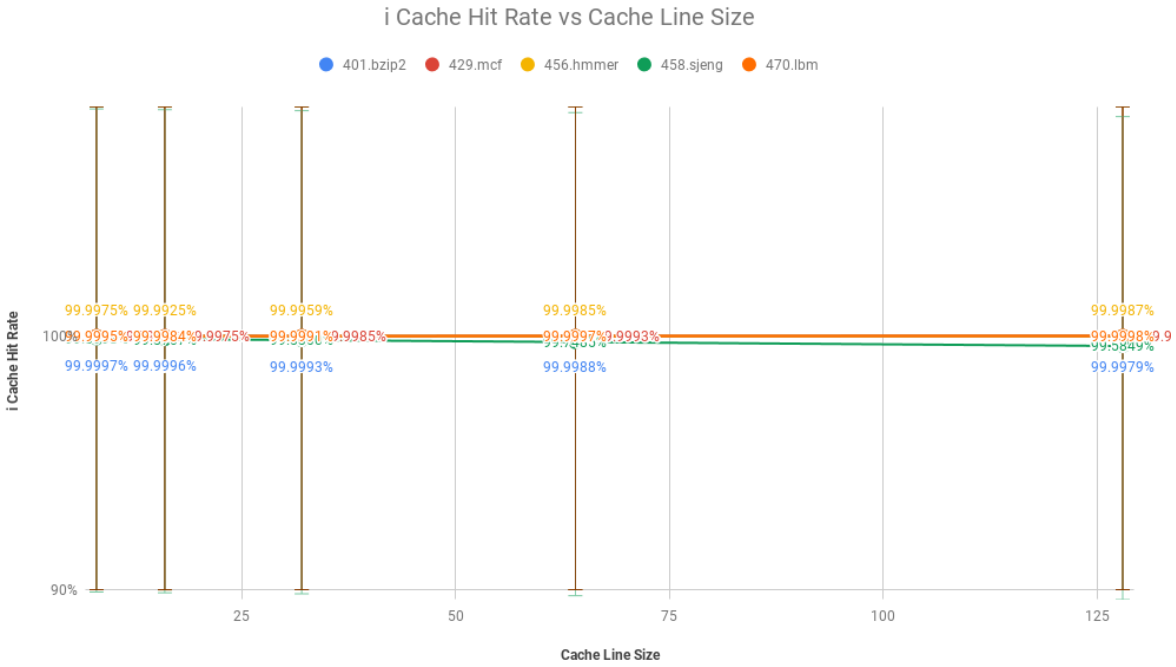

We have varied cache line size from 8, 16, 32, 64, 128 bytes. Increasing cache line size implies increasing the number of lines through which data can be fed to the cache memories. Increasing cacheline size has a direct affect on hit rate. As we can see the hit rates usually lie in the range of 99%, however for benchmark 4, the hit rate decreases as we increase the cache line size.

7.2 Block Size vs Miss Rate of L1 D Cache



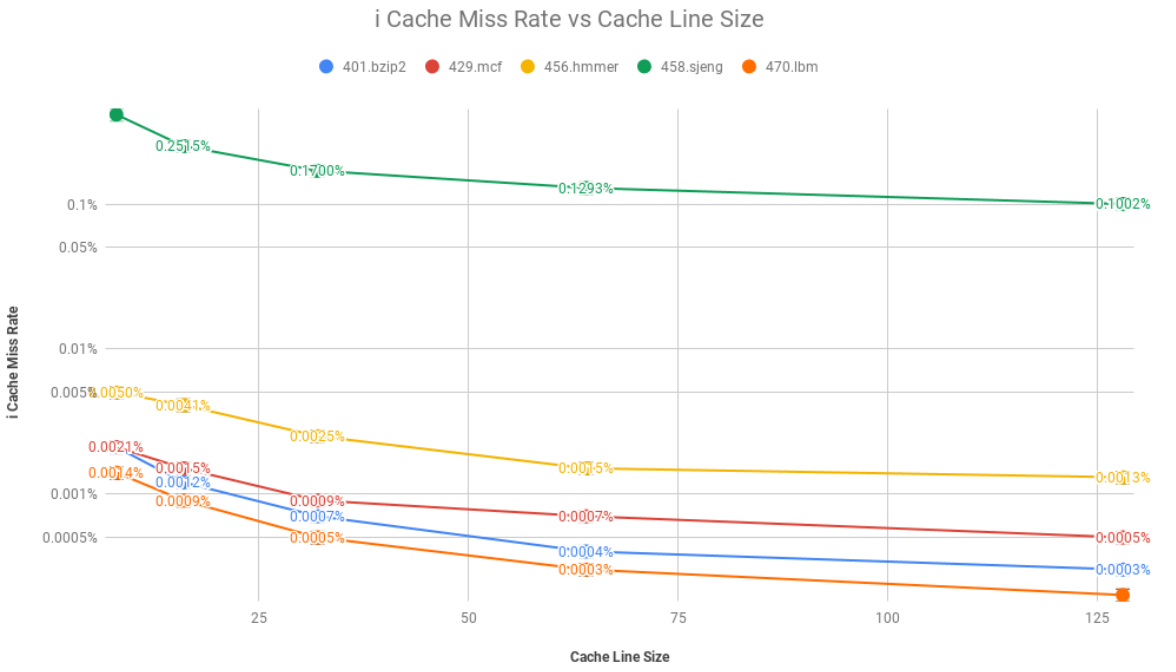

Again the change in cacheline size has a direct impact on miss rates of d cache, as we can see on increasing the cacheline the miss rates decreases.

7.3 Block Size vs Hit Rate of L1 I Cache



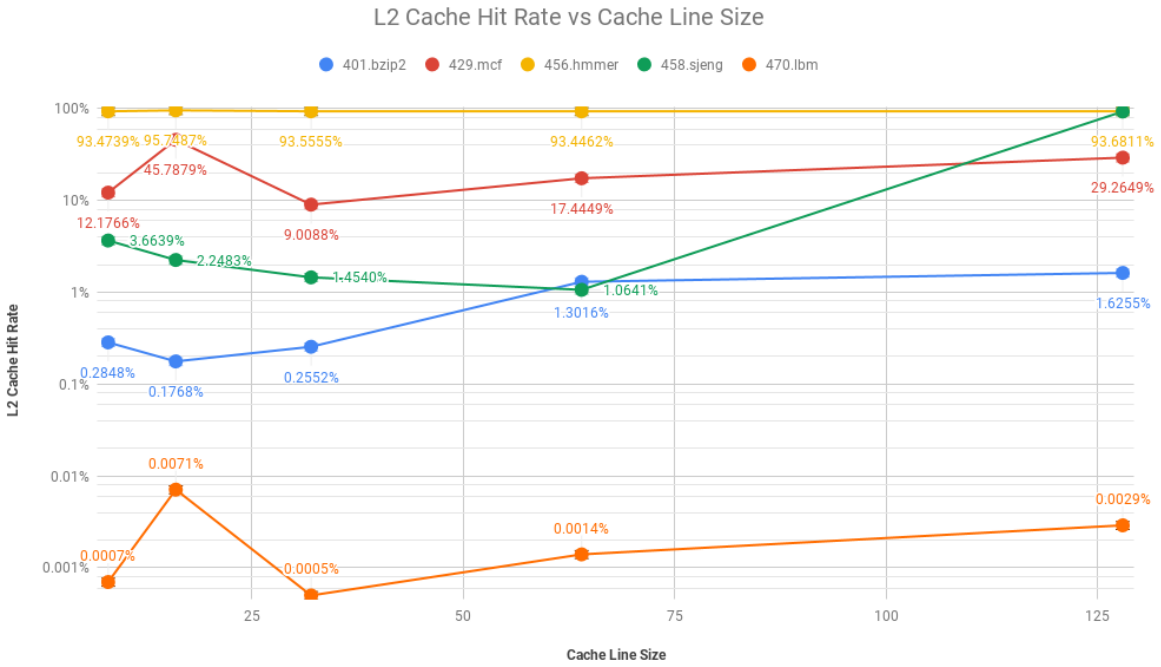

We can notice that not much change is there in hit rates of I cache when we vary the cache line size. However to get a clear look we can see below the miss rate of L1 I cache.

7.4 Block Size vs Miss Rate of L1 I Cache



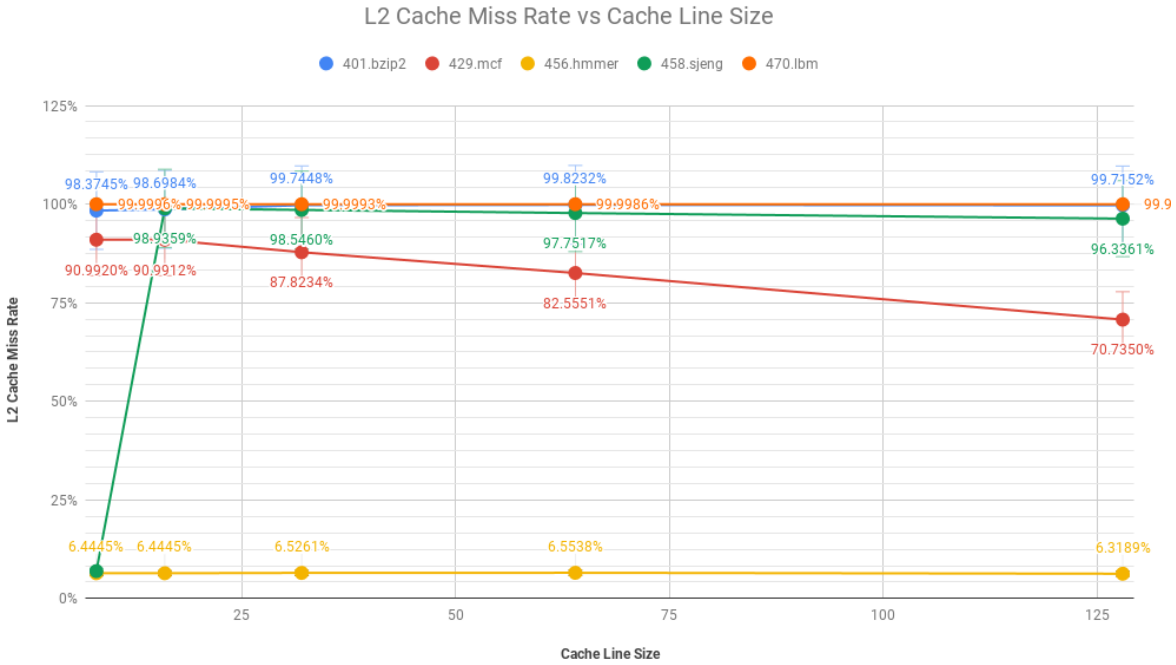

We can see that with that with the increase in cache line size the miss rates of I cache decreases in all benchmarks.The miss rate being lowest when cacheline size is 128bytes.

7.5 Block Size vs Hit Rate of L2 Cache



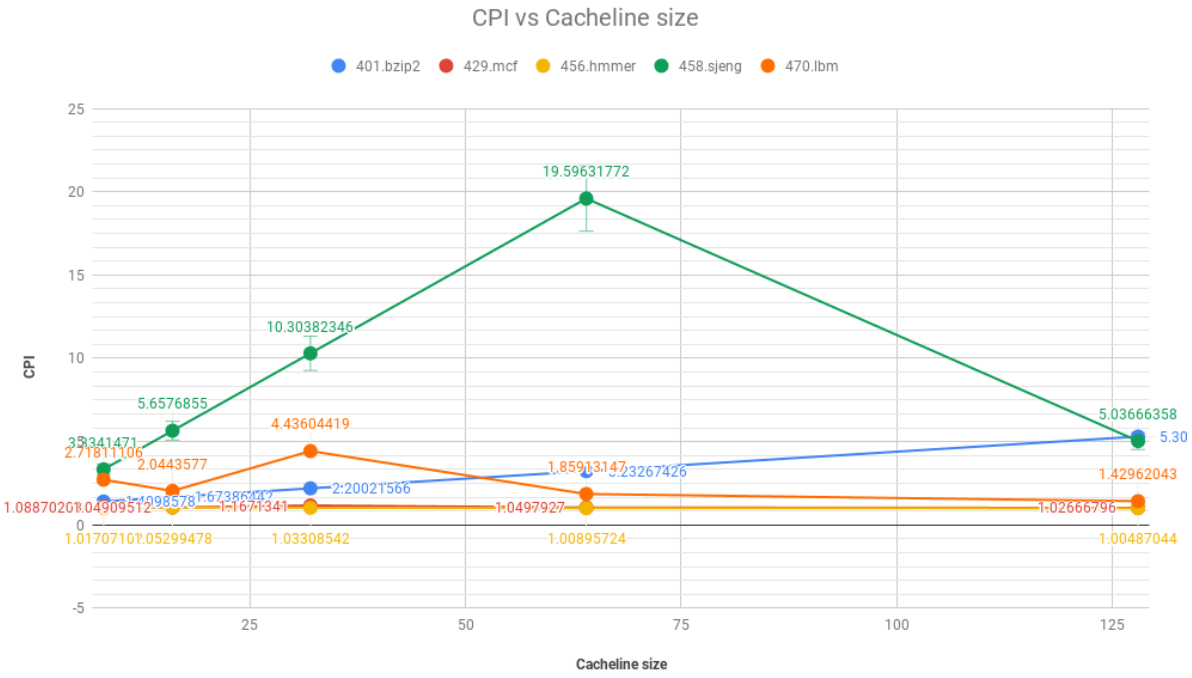

We can see that on increasing the cacheline size, the hit rates of L2 cache either increases or remains same. There can be seen a drastic rise in hit rate for benchmark-4, and also it increases for benchmark-1,2, and 5. However for benchmark 3 it remains the same.

7.6 Block Size vs Miss Rate of L2 Cache




We can see increasing the cacheline size has an affect on miss rates of L2 cache,especially for benchmark 2 it decreases as we increase the cacheline size.

7.7 Block Size vs CPI




We can notice that as we increase the cacheline size, the clocks needed for per instruction also changes drastically.We can see that when we kept the cache line size at 64 for benchmark-4 , it had the worst CPI of about 19.596 , for every benchmark the best and worst cases are different.

After doing all the analysis the best values for each benchmark has been noted below under conclusion ,for your kind perusal.

## CONCLUSION:

Analysed how varying the cache parameters influenced the performance of X86 processor in large extents. Though while changing some parameters hit rates and miss rates weren't affected. However, overall there were great extent of changes.

## TEAM WORK AND INDIVIDUAL CONTRIBUTION:

6. *Venugopal Shah :* Worked on understanding how and what commands need to be passed for running benchmarks on gem5.Wrote shell script to run the benchmarks together. Also wrote a python script to extract only the values needed into another file. Plotted the graph for all the values stored in the data base.
7. *Swati Sajee Kumar :* Worked on understanding how and what commands need to be passed for running benchmarks on gem5.Wrote shell script to run the benchmarks together. Analysed and stored the data we got from the stats.txt and worked on the report.

## REFERENCES:


1. https://en.wikipedia.org/wiki/CPU_cache
2. https://groups.google.com/forum/#!topic/comp.arch/9DcDSzs28Ow
3. https://en.wikipedia.org/wiki/Cycles_per_instruction
4. Piazza and Class Notes
5. Other google sources

## APPENDIX-I

1. An example of script used to run all the simulation of particular testbench together is as given below:

```
#changing l1d_size
sudo ./build/X86/gem5.opt configs/example/se.py -c /home/swati/Project1_SPEC-master/458.sjeng/src/benchmark -I 100000000 -o /home/swati/Project1_SPEC-master/458.sjeng/data/test.txt --caches --l2cache --l1d_size=2MB
sudo mv /home/swati/gem5/m5out/stats.txt /home/swati/Desktop/458.sjeng/stat_files/l1d_size/test1.txt

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/swati/Project1_SPEC-master/458.sjeng/src/benchmark -I 100000000 -o /home/swati/Project1_SPEC-master/458.sjeng/data/test.txt --caches --l2cache --l1d_size=512kB
sudo mv /home/swati/gem5/m5out/stats.txt /home/swati/Desktop/458.sjeng/stat_files/l1d_size/test2.txt

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/swati/Project1_SPEC-master/458.sjeng/src/benchmark -I 100000000 -o /home/swati/Project1_SPEC-master/458.sjeng/data/test.txt --caches --l2cache --l1d_size=1MB
sudo mv /home/swati/gem5/m5out/stats.txt /home/swati/Desktop/458.sjeng/stat_files/l1d_size/test3.txt

sudo ./build/X86/gem5.opt configs/example/se.py -c /home/swati/Project1_SPEC-master/458.sjeng/src/benchmark -I 100000000 -o /home/swati/Project1_SPEC-master/458.sjeng/data/test.txt --caches --l2cache --l1d_size=3MB
sudo mv /home/swati/gem5/m5out/stats.txt /home/swati/Desktop/458.sjeng/stat_files/l1d_size/test4.txt
sudo ./build/X86/gem5.opt configs/example/se.py -c /home/swati/Project1_SPEC-master/458.sjeng/src/benchmark -I 100000000 -o /home/swati/Project1_SPEC-master/458.sjeng/data/test.txt --caches --l2cache --l1d_size=4MB
sudo mv /home/swati/gem5/m5out/stats.txt /home/swati/Desktop/458.sjeng/stat_files/l1d_size/test5.txt
```

2. An example of python script used to get only the hit and miss rates, and hit and miss number from stats.txt file is as shown below:

```
import csv
import re
import glob

read = glob.glob("*.txt")


dcache_overall_miss_rate = []
dcache_overall_hit_rate = []
dcache_overall_misses = []

icache_overall_miss_rate = []
icache_overall_hit_rate = []
icache_overall_misses = []

l2_overall_miss_rate = []
l2_overall_hit_rate = []
l2_overall_misses = []

cpi = []

for filename in read:
        with open(filename,"r") as f:

                for line in f:

                        if 'system.l2.overall_miss_rate::total' in line:

                                l = ((float(line.split()[1])))
                                l2_overall_miss_rate.append((l))

                        elif 'system.cpu.dcache.overall_miss_rate::total' in line:

                                d = ((float(line.split()[1])))
                                dcache_overall_miss_rate.append((d))

                        elif 'system.cpu.icache.overall_miss_rate::total' in line:

                                i = ((float(line.split()[1])))
                                icache_overall_miss_rate.append((i))

                        elif 'system.l2.overall_misses::total' in line:

                                d = ((float(line.split()[1])))
                                l2_overall_misses.append(d)

                        elif 'system.cpu.dcache.overall_misses::total' in line:

                                i = ((float(line.split()[1])))
                                dcache_overall_misses.append(i)

                      elif 'system.cpu.icache.overall_misses::total' in line:

                                l = ((float(line.split()[1])))
                                icache_overall_misses.append(l)


cpi = list((1 + ((6 * (dcache_overall_misses[i] + icache_overall_misses[i])) + (50 * l2_overall_misses[i])) / 100000000) for i in range(len(dcache_overall_misses)))


icache_overall_hit_rate = list((1 - icache_overall_miss_rate[i] for i in range(len(icache_overall_miss_rate))))
dcache_overall_hit_rate = list((1 - dcache_overall_miss_rate[i] for i in range(len(dcache_overall_miss_rate))))
l2_overall_hit_rate = list((1 - l2_overall_miss_rate[i] for i in range(len(l2_overall_miss_rate))))


print("\n")
print("dCache miss rate")
print(dcache_overall_miss_rate)

print("\n")
print("dCache hit rate")
print(dcache_overall_hit_rate)
print("\n")

print("\n")
print("iCache miss rate")
print(icache_overall_miss_rate)

print("\n")
print("iCache hit rate")
print(icache_overall_hit_rate)
print("\n")

print("\n")
print("l2Cache miss rate")
print(l2_overall_miss_rate)

print("\n")
print("l2Cache hit rate")
print(l2_overall_hit_rate)
print("\n")

print("\n")
print("CPI")
```

## APPENDIX-II

Database of stats.txt file used for plotting all the 275 graphs:

| benchmark | 401.bzip2 | | | 429.mcf | | | 456.hmmer | | | 458.sjeng | | | 470.lbm | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **L1d_size** | dmiss | i miss | l2miss | dmiss | i miss | l2miss | dmiss | i miss | l2miss | dmiss | i miss | l2miss | dmiss | i miss | l2miss |
| 64kB | 0.017985 | 0.000004 | 0.998232 | 0.002993 | 0.000007 | 0.825551 | 0.001851 | 0.000015 | 0.065538 | 0.097420 | 0.001293 | 0.978955 | 0.038460 | 0.000005 | 0.999986 |
| 128kB | 0.017983 | 0.000004 | 0.998341 | 0.002790 | 0.000007 | 0.885218 | 0.000818 | 0.000015 | 0.144304 | 0.097423 | 0.001293 | 0.978933 | 0.038460 | 0.000005 | 0.999986 |
| 256kB | 0.017980 | 0.000004 | 0.998514 | 0.002712 | 0.000007 | 0.910770 | 0.000110 | 0.000015 | 0.815254 | 0.097421 | 0.001293 | 0.978958 | 0.038460 | 0.000005 | 0.999987 |
| 32kB | 0.018025 | 0.000004 | 0.996029 | 0.003450 | 0.000007 | 0.717164 | 0.002583 | 0.000015 | 0.047247 | | | | 0.038460 | 0.000005 | 0.999986 |
| 16kB | 0.018107 | 0.000004 | 0.991502 | 0.004288 | 0.000007 | 0.577723 | 0.004275 | 0.000015 | 0.028732 | 0.097417 | 0.001293 | 0.978952 | 0.038460 | 0.000005 | 0.999985 |

| benchmark | 401.bzip2 | | | 429.mcf | | | 456.hmmer | | | 458.sjeng | | | 470.lbm | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **L1i_size** | dmiss | i miss | l2miss | dmiss | i miss | l2miss | dmiss | i miss | l2miss | dmiss | i miss | l2miss | dmiss | i miss | l2miss |
| 32kB | 0.017985 | 0.000004 | 0.998232 | 0.002993 | 0.000007 | 0.825551 | 0.001851 | 0.000015 | 0.065538 | 0.097567 | 0.001293 | 0.977517 | 0.038460 | 0.000003 | 0.999986 |

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 16kB | 0.017985 | 0.0000012 | 0.9973933 | 0.0029933 | 0.0038589 | 0.1373033 | 0.0018511 | 0.0000035 | 0.063720 | 0.0975567 | 0.0031336 | 0.9485852 | 0.0384600 | 0.0000004 | 0.9999965 |
| 64kB | 0.017985 | 0.0000004 | 0.9982366 | 0.0029933 | 0.0000004 | 0.8289593 | 0.0018511 | 0.0000015 | 0.065604 | 0.0975567 | 0.0000277 | 0.9940633 | 0.0384600 | 0.0000003 | 0.9999992 |
| 128kB | 0.017985 | 0.0000004 | 0.9982399 | 0.0029933 | 0.0000004 | 0.8289593 | 0.0018511 | 0.0000009 | 0.066090 | 0.0975567 | 0.0000016 | 0.9984111 | 0.0384600 | 0.0000003 | 0.9999998 |
| 256kB | 0.017985 | 0.0000004 | 0.9982399 | 0.0029933 | 0.0000004 | 0.8289593 | 0.0018511 | 0.0000009 | 0.066096 | 0.0975567 | 0.0000011 | 0.9984845 | 0.0384600 | 0.0000003 | 0.9999998 |

| **benchmark** | 401.bzip2 | | | 429.mcf | | | 456.hmmer | | | 458.sjeng | | | 470.lbm | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **L2_size** | dmiss | i miss | l2miss | dmiss | i miss | l2miss | dmiss | i miss | l2miss | dmiss | i miss | l2miss | dmiss | i miss | l2miss |
| 2MB | 0.017985 | 0.0000004 | 0.9982322 | 0.0029933 | 0.0000007 | 0.8255551 | 0.0018511 | 0.0000015 | 0.0655538 | 0.0975567 | 0.0012933 | 0.9775177 | 0.0384600 | 0.0000003 | 0.9999986 |
| 512kB | 0.017985 | 0.0000004 | 0.9995509 | 0.0029933 | 0.0000007 | 0.8821211 | 0.0018511 | 0.0000015 | 0.0655538 | 0.0975567 | 0.0012933 | 0.9775244 | 0.0384600 | 0.0000003 | 0.9999986 |
| 1MB | 0.017985 | 0.0000004 | 0.9990777 | 0.0029933 | 0.0000007 | 0.8511196 | 0.0018511 | 0.0000015 | 0.0655538 | 0.0975567 | 0.0012933 | 0.9775177 | 0.0384600 | 0.0000003 | 0.9999986 |
| 256kB | 0.017985 | 0.0000004 | 0.9997433 | 0.0029933 | 0.0000007 | 0.8957388 | 0.0018511 | 0.0000015 | 0.0696211 | 0.0975567 | 0.0012933 | 0.9776333 | 0.0384600 | 0.0000003 | 0.9999986 |
| 4MB | 0.017985 | 0.0000004 | 0.9600933 | 0.0029933 | 0.0000007 | 0.7848188 | 0.0018511 | 0.0000015 | 0.0655538 | 0.0975567 | 0.0012933 | 0.9774944 | 0.0384600 | 0.0000003 | 0.9999986 |

| **benchmark** | 401.bzip2 | | | 429.mcf | | | 456.hmmer | | | 458.sjeng | | | 470.lbm | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **L1d_assoc** | dmiss | i miss | l2miss | dmiss | i miss | l2miss | dmiss | i miss | l2miss | dmiss | i miss | l2miss | dmiss | i miss | l2miss |
| 2 | 0.017985 | 0.000004 | 0.998232 | 0.002993 | 0.000007 | 0.825551 | 0.001851 | 0.000015 | 0.065538 | 0.097567 | 0.001293 | 0.977517 | 0.038460 | 0.000003 | 0.999986 |
| 1 | 0.018050 | 0.000004 | 0.994602 | 0.003591 | 0.000007 | 0.689064 | 0.003315 | 0.000015 | 0.036944 | 0.097884 | 0.001293 | 0.974420 | 0.039586 | 0.000003 | 0.971543 |
| 4 | 0.017983 | 0.000004 | 0.998353 | 0.002892 | 0.000007 | 0.854250 | 0.001749 | 0.000015 | 0.069268 | 0.097514 | 0.001293 | 0.978039 | 0.038460 | 0.000003 | 0.999987 |
| 8 | 0.017983 | 0.000004 | 0.998351 | 0.002848 | 0.000007 | 0.867158 | 0.001844 | 0.000015 | 0.065781 | 0.097472 | 0.001293 | 0.978456 | 0.038460 | 0.000003 | 0.999987 |
| 16 | 0.017983 | 0.000004 | 0.998353 | | | | | | | 0.097463 | 0.001293 | 0.978538 | | | |

| **benchmark** | 401.bzip2 | | | 429.mcf | | | 456.hmmer | | | 458.sjeng | | | 470.lbm | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **L1i_assoc** | dmiss | i miss | l2miss | dmiss | i miss | l2miss | dmiss | i miss | l2miss | dmiss | i miss | l2miss | dmiss | i miss | l2miss |
| 2 | 0.017 | 0.000 | 0.998 | 0.002 | 0.000 | 0.825 | 0.001 | 0.000 | 0.065 | 0.097 | 0.001 | 0.977 | 0.038 | 0.000 | 0.999 |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 985 | 004 | 232 | 993 | 007 | 551 | 851 | 015 | 538 | 567 | 293 | 517 | 460 | 003 | 986 |
| 1 | 0.017985 | 0.0000010 | 0.9975550 | 0.0029933 | 0.0024889 | 0.1951455 | 0.0018511 | 0.0000054 | 0.0620887 | 0.0975567 | 0.0019466 | 0.9671611 | 0.0384600 | 0.0000004 | 0.9999967 |
| 4 | 0.017985 | 0.0000004 | 0.9982237 | 0.0029933 | 0.0000004 | 0.8289533 | 0.0018511 | 0.0000010 | 0.0660422 | 0.0975567 | 0.0000678 | 0.9874688 | 0.0384600 | 0.0000003 | 0.9999991 |
| 8 | 0.017985 | 0.0000004 | 0.9982238 | 0.0029933 | 0.0000004 | 0.8289600 | 0.0018511 | 0.0000010 | 0.0660302 | 0.0975567 | 0.0000489 | 0.9905722 | 0.0384600 | 0.0000003 | 0.9999994 |
| 16 | 0.017985 | 0.0000004 | 0.998239 | | | | | | | 0.0975567 | 0.0000489 | 0.9905722 | | | |

| **benchmark** | 401.bzip2 | | | 429.mcf | | | 456.hmmer | | | 458.sjeng | | | 470.lbm | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **L2_assoc** | dmiss | i miss | l2miss | dmiss | i miss | l2miss | dmiss | i miss | l2miss | dmiss | i miss | l2miss | dmiss | i miss | l2miss |
| 8 | 0.017985 | 0.0000004 | 0.9982232 | 0.0029933 | 0.0000007 | 0.8255551 | 0.0018511 | 0.0000015 | 0.0653538 | 0.0975567 | 0.0012933 | 0.9777517 | 0.0384600 | 0.0000003 | 0.9999986 |
| 1 | 0.017985 | 0.0000004 | 0.9716697 | 0.0029933 | 0.0000007 | 0.8480444 | 0.0018511 | 0.0000015 | 0.0653538 | 0.0975567 | 0.0012933 | 0.9775598 | 0.0384600 | 0.0000003 | 0.9999986 |
| 2 | 0.017985 | 0.0000004 | 0.9774233 | 0.0029933 | 0.0000007 | 0.8240998 | 0.0018511 | 0.0000015 | 0.0653538 | 0.0975567 | 0.0012933 | 0.9775226 | 0.0384600 | 0.0000003 | 0.9999986 |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 0.0179985 | 0.0000004 | 0.9942155 | 0.0029933 | 0.0000007 | 0.8247706 | 0.0018511 | 0.0000015 | 0.0655538 | 0.0975667 | 0.0012933 | 0.9775180 | 0.0384600 | 0.0000003 | 0.9999986 |
| 16 | 0.0179985 | 0.0000004 | 0.9982229 | | | | | | | 0.0975667 | 0.0012933 | 0.9775177 | | | |

| **benchmark** | 401.bzip2 | | | 429.mcf | | | 456.hmmer | | | 458.sjeng | | | 470.lbm | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **cacheline_size** | dmiss | i miss | l2miss | dmiss | i miss | l2miss | dmiss | i miss | l2miss | dmiss | i miss | l2miss | dmiss | i miss | l2miss |
| 64 | 0.0179985 | 0.0000004 | 0.9982332 | 0.0029933 | 0.0000007 | 0.8255551 | 0.0018511 | 0.0000015 | 0.0655538 | 0.0975667 | 0.0012933 | 0.9775177 | 0.384600 | 0.0000003 | 0.9999986 |
| 16 | 0.0601967 | 0.0000012 | 0.9869844 | 0.0092311 | 0.0000015 | 0.9090912 | 0.0068779 | 0.0000041 | 0.0644455 | 0.3896233 | 0.0025155 | 0.9893559 | 0.153835 | 0.0000009 | 0.9999995 |
| 32 | 0.0320556 | 0.0000007 | 0.9974488 | 0.0050557 | 0.0000009 | 0.8782344 | 0.0035444 | 0.0000025 | 0.0652611 | 0.1949044 | 0.0017000 | 0.9854600 | 0.076918 | 0.0000005 | 0.9999993 |
| 128 | 0.0109499 | 0.0000003 | 0.9971522 | 0.0018229 | 0.0000005 | 0.7073500 | 0.0010006 | 0.0000013 | 0.0631899 | 0.0489699 | 0.0010002 | 0.9636361 | 0.019231 | 0.0000002 | 0.9999971 |
| 8 | 0.1164777 | 0.0000021 | 0.9837455 | | | | | | | 0.7790557 | 0.0041511 | 0.0698377 | | | |

Number of misses :

| **benchmark** | 401.bzip2 | 429.mcf | 456.hmmer | 458.sjeng | 470.lbm |
|---|---|---|---|---|---|
| **L1d_size** | CPI | CPI | CPI | CPI | CPI |
| 64kB | 1.6738644200000001 | 1.0497927 | 1.0089572 4 | 5.6569492 4 | 1.8591314 7 |
| 128kB | 1.6738439600000001 | 1.0525022 6 | 1.0163783 4 | 5.6568884 4 | 1.8591315 4 |
| 256kB | 1.6738564999999999 | 1.0493663 6 | 1.0057951 2 | 5.6569354 8 | 1.8591314 7 |
| 32kB | 1.67402438 | 1.0507489 | 1.0112000 4 | Nil | 1.8591314 7 |
| 16kB | 1.67435528 | 1.0492223 6 | 1.0036297 2 | 5.6567451 6 | 1.8591313 6 |

| **benchmark** | 401.bzip2 | 429.mcf | 456.hmmer | 458.sjeng | 470.lbm |
|---|---|---|---|---|---|
| **L1i_size** | CPI | CPI | CPI | CPI | CPI |
| 32kB | 1.67393142 | 1.0497927 | 1.0089572 4 | 5.6731295 6 | 1.8591314 7 |
| 16kB | 1.6738644200000001 | 1.0497667 2 | 1.0089083 4 | 5.6576855 | 1.8591303 4 |
| 64kB | 1.67386412 | 1.0814731 6 | 1.0091225 4 | 5.6491710 4 | 1.8591334 0 |
| 128kB | 1.67386394 | 1.0497667 2 | 1.0089088 8 | 5.6469429 4 | 1.8591303 4 |

| | | | | | |
|---|---|---|---|---|---|
| 256kB | 1.67386394 | 1.04976672 | 1.00895142 | 5.64698226 | 1.85913088 |

| **benchmark** | 401.bzip2 | 429.mcf | 456.hmmer | 458.sjeng | 470.lbm |
|---|---|---|---|---|---|
| **L2_size** | CPI | CPI | CPI | CPI | CPI |
| 2MB | 1.67463392 | 1.0497927 | 1.00895724 | 5.6577155 | 1.85913147 |
| 512kB | 1.6738644200 | 1.047477 | 1.00895724 | 5.6576855 | 1.85913147 |
| 1MB | 1.67437342 | 1.052717 | 1.00895724 | 5.6576855 | 1.85913147 |
| 256kB | 1.65088142000 | 1.05348426 | 1.0091568 | 5.6575845 | 1.85913147 |
| 4MB | 1.67477492 | 1.0511432 | 1.00895724 | 5.6581775 | 1.85913147 |

| **benchmark** | 401.bzip2 | 429.mcf | 456.hmmer | 458.sjeng | 470.lbm |
|---|---|---|---|---|---|
| **L1d_assoc** | CPI | CPI | CPI | CPI | CPI |
| 2 | 1.6741283 | 1.0497927 | 1.00895724 | 5.65930362 | 1.85913147 |
| 1 | 1.6738644200 | 1.0495204 | 1.00894728 | 5.6576855 | 1.85913136 |
| 4 | 1.67385566 | 1.05104262 | 1.0134411 | 5.6576855 | 1.86182638 |
| 8 | 5.30686384 | 1.04948722 | 1.00893588 | 5.65715456 | 1.85913136 |
| 16 | 1.67385578 | 1.04957812 | 1.0086453 | 5.6571968 | 1.85913136 |

| **benchmark** | 401.bzip2 | 429.mcf | 456.hmmer | 458.sjeng | 470.lbm |
|---|---|---|---|---|---|
| **L1i_assoc** | CPI | CPI | CPI | CPI | CPI |
| 2 | 1.6739188999999999 | 1.0497927 | 1.00895724 | 5.66315204 | 1.85913147 |
| 1 | 1.6738644200000001 | 1.04976672 | 1.00891188 | 5.6576855 | 1.8591304 |

| | | | | | |
|---|---|---|---|---|---|
| 4 | 1.67386406 | 1.07021062 | 1.00927926 | 5.65252986 | 1.85913315 |
| 8 | 1.67386394 | 1.04976728 | 1.00891308 | 5.64984044 | 1.85913070 |
| 16 | 1.673864 | 1.04976734 | 1.00891398 | 5.65094652 | 1.859131 |

| **benchmark** | 401.bzip2 | 429.mcf | 456.hmmer | 458.sjeng | 470.lbm |
|---|---|---|---|---|---|
| **L2_assoc** | CPI | CPI | CPI | CPI | CPI |
| 2 | 1.65787392000 | 1.0497927 | 1.00895724 | 5.658029 | 1.85913147 |
| 1 | 1.6738644200 | 1.0498172 | 1.00895724 | 5.6576855 | 1.85913147 |
| 4 | 1.66132492 | 1.0509772 | 1.00895724 | 5.657721 | 1.85913147 |
| 8 | 1.67386241999 | 1.0497482 | 1.00895724 | 5.6576855 | 1.85913147 |
| 16 | 1.67386241999 | 1.0497162 | 1.00895724 | 5.657686 | 1.85913147 |

| **benchmark** | 401.bzip2 | 429.mcf | 456.hmmer | 458.sjeng | 470.lbm |
|---|---|---|---|---|---|
| **cacheline_size** | CPI | CPI | CPI | CPI | CPI |
| 64 | 3.23267426 | 1.0497927 | 1.00895724 | 19.59631772 | 1.85913147 |
| 16 | 1.673864420 | 1.04909512 | 1.05299478 | 5.6576855 | 2.0443577 |
| 32 | 2.20021566 | 1.1671341 | 1.03308542 | 10.30382346 | 4.43604419 |
| 128 | 5.30680216 | 1.02666796 | 1.00487044 | 5.03666358 | 1.42962043 |
| 8 | 1.4098578 | 1.08870208 | 1.01707102 | 3.3341471 | 2.71811106 |

HIT RATES:

| **benchmark** | 401.bzip2 | | | 429.mcf | | | 456.hmmer | | | 458.sjeng | | | 470.lbm | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **l1d_size** | d_hit | i _hit | l2_hit | d_hit | i _hit | l2_hit | d_hit | i _hit | l2_hit | d_hit | i _hit | l2_hit | d_hit | i _hit | l2_hit |
| 64kB | 0.982015 | 0.999996 | 0.0017679999999918 | 0.999707 | 0.999993 | 0.17444 | 0.999849 | 0.999985 | 0.934462 | 0.902577 | 0.999887 | 0.021066999999 | 0.961544 | 0.999997 | 0.000014 |
| 128kB | 0.982022 | 0.999996 | 0.001485999 | 0.999572 | 0.999993 | 0.422277 | 0.999572 | 0.999985 | 0.971268 | 0.90257999 | 0.999887 | 0.021044999999 | 0.961544 | 0.999997 | 0.000013 |
| 256kB | 0.982017 | 0.999996 | 0.00165899999 | 0.999721 | 0.999993 | 0.114782 | 0.999182 | 0.999985 | 0.855696 | 0.902579 | 0.999887 | 0.021042000000 | 0.960414 | 0.999997 | 0.0285 |
| 32kB | 0.981975 | 0.999996 | 0.003970999999 | 0.999655 | 0.999993 | 0.282836 | 0.999417 | 0.999985 | 0.952753 | Nil | Nil | Nil | 0.961544 | 0.999997 | 0.000013 |
| 16kB | 0.981893 | 0.999996 | 0.0084980000 | 0.999728 | 0.999993 | 0.08923 | 0.999989 | 0.999985 | 0.184745 9 | 0.902583 | 0.999887 | 0.021047999999 | 0.961544 | 0.999997 | 0.000013 |

| **benchmark** | 401.bzip2 | | | 429.mcf | | | 456.hmmer | | | 458.sjeng | | | 470.lbm | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **l1i_size** | d_hit | i _hit | l2_hit | d_hit | i _hit | l2_hit | d_hit | i _hit | l2_hit | d_hit | i _hit | l2_hit | d_hit | i _hit | l2_hit |
| 32kB | 0.982015 | 0.999988 | 0.0026070 0 | 0.997007 | 0.999993 | 0.174448 | 0.998149 | 0.999985 | 0.934462 | 0.902433 | 0.996864 | 0.051147999999999 | 0.961544 | 0.999997 | 0.000014 |
| 16kB | 0.982015 | 0.999996 | 0.0017679999999 | 0.997007 | 0.999996 | 0.171041 | 0.998149 | 0.999991 | 0.933904 | 0.902433 | 0.996864 | 0.022483000000 | 0.961544 | 0.999997 | 0.000002 |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 64kB | 0.982015 | 0.999996 | 0.0017639999 | 0.997007 | 0.996141 | 0.862697 | 0.998149 | 0.999965 | 0.93628 | 0.902433 | 0.996864 | 0.00593699999 | 0.961544 | 0.999996 | 0.000035 |
| 128kB | 0.982015 | 0.999996 | 0.0017610 | 0.997007 | 0.999996 | 0.171041 | 0.998149 | 0.999991 | 0.93391 | 0.902433 | 0.996864 | 0.00151500000 | 0.961544 | 0.999997 | 0.000002 |
| 256kB | 0.982015 | 0.999996 | 0.00176100 | 0.997007 | 0.999996 | 0.171041 | 0.998149 | 0.999985 | 0.934396 | 0.902433 | 0.996864 | 0.00158899999999 | 0.961544 | 0.999997 | 0.000008 |

| **benchmark** | 401.bzip2 | | | 429.mcf | | | 456.hmmer | | | 458.sjeng | | | 470.lbm | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **l2_size** | d_hit | i _hit | l2_hit | d_hit | i _hit | l2_hit | d_hit | i _hit | l2_hit | d_hit | i _hit | l2_hit | d_hit | i _hit | l2_hit |
| 2MB | 0.017985 | 0.999996, | 0.0004910000 | 0.997007 | 0.999993 | 0.174448 | 0.998149 | 0.999985 | 0.934462 | 0.902433 | 0.998707 | 0.022476 0000 | 0.961544 | 0.999997 | 0.000014 |
| 512kB | 0.017985 | 0.999996, | 0.00176799999 | 0.997007 | 0.999993 | 0.215181 1 | 0.998149 | 0.999985 | 0.934462 | 0.902433 | 0.998707 | 0.022483000 | 0.961544 | 0.999997 | 0.000014 |
| 1MB | 0.017985 | 0.999996, | 0.000 92300 | 0.997007 | 0.999993 | 0.117 8789 | 0.998149 | 0.999985 | 0.934462 | 0.902433 | 0.998707 | 0.022483000 | 0.961544 | 0.999997 | 0.000014 |
| 256kB | 0.017985 | 0.999996, | 0.039 9070 | | | | | | | 0.902433 | 0.998707 | 0.022506000 | 0.961544 | 0.999997 | 0.000014 |
| 4MB | 0.017985 | 0.999996, | 0.000 2569999999 | 0.997007 | 0.999993 | 0.148804 | 0.998149 | 0.999985 | 0.934462 | 0.902433 | 0.998707 | 0.022367 0000 | 0.961544 | 0.999997 | 0.000014 |

| **benc** | 401.bzip2 | 429.mcf | 456.hmmer | 458.sjeng | 470.lbm |
|---|---|---|---|---|---|

| hmark | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| l1d_assoc | d_hit | i _hit | l2_hit | d_hit | i _hit | l2_hit | d_hit | i _hit | l2_hit | d_hit | i _hit | l2_hit | d_hit | i _hit | l2_hit |
| 8 | 0.98195 | 0.999996 | 0.0053980 | 0.9970007 | 0.999993 | 0.17444899 | 0.9981449 | 0.999985 | 0.934462 | 0.9021616 | 0.9987707 | 0.025580000000000047 | 0.961544 | 0.999997 | 0.000014 |
| 1 | 0.982015 | 0.999996 | 0.00176799 99 | 0.9971136 | 0.999993 | 0.137599 | 0.9981153 | 0.999985 | 0.934349 | 0.9024333 | 0.9987707 | 0.022483000 | 0.961544 | 0.999997 | 0.000013 |
| 2 | 0.982017 | 0.999996 | 0.00164699 99 | 0.9964009 | 0.999993 | 0.310936 | 0.9966685 | 0.999985 | 0.963056 | 0.9024886 | 0.9987707 | 0.021961000000 | 0.960414 | 0.999997 | 0.0285 |
| 4 | 0.883518999999 | 0.999979 | 0.01627699 | 0.9971152 | 0.999993 | 0.1328420 | 0.9981656 | 0.999985 | 0.934219 | 0.9025337 | 0.9987707 | 0.0214619999 | 0.961544 | 0.999997 | 0.000013 |
| 16 | 0.982017 | 0.999996 | 0.00164900 00 | 0.9971108 | 0.999993 | 0.145750 | 0.9982551 | 0.999985 | 0.930732 | 0.9025328 | 0.9987707 | 0.02154400 | 0.961544 | 0.999997 | 0.000013 |

| benchmark | 401.bzip2 | | | 429.mcf | | | 456.hmmer | | | 458.sjeng | | | 470.lbm | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| l1i_assoc | d_hit | i _hit | l2_hit | d_hit | i _hit | l2_hit | d_hit | i _hit | l2_hit | d_hit | i _hit | l2_hit | d_hit | i _hit | l2_hit |
| 8 | 0.9820155 | 0.999999 | 0.999996 | 0.9970007 | 0.999993 | 0.1744489 | 0.9981499 | 0.999985 | 0.9344622 | 0.9024333 | 0.9980544 | 0.0328389 9 | 0.961544 | 0.999997 | 0.0000014 |
| 1 | 0.9820155 | 0.999996 | 0.00176799 99 | 0.9970007 | 0.999996 | 0.171041 | 0.9981499 | 0.999999 | 0.9339444 | 0.9024333 | 0.9987077 | 0.0224830 0 | 0.961544 | 0.999997 | 0.0000003 |
| 2 | 0.982 | 0.999 | 0.00176299 | 0.997 | 0.997 | 0.804855 | 0.9981 | 0.999 | 0.937 | 0.902 | 0.999 | 0.0125 | 0.96 | 0.999 | 0.000 |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 015 | 996 | | 007 | 511 | | 149 | 946 | 913 | 433 | 322 | 3199 | 154 | 996 | 033 |
| 4 | 0.9820115 | 0.9999996 | 0.00176100 | 0.9970007 | 0.9999996 | 0.171039 | 0.9981499 | 0.999999 | 0.933958 | 0.9024333 | 0.9996433 | 0.00726499 | 0.961154 | 0.9999997 | 0.000006 |
| 16 | 0.9820115 | 0.9999996 | 0.00176200000 | 0.9970007 | 0.9999996 | 0.1710469 | 0.9981499 | 0.999999 | 0.933968 | 0.9024333 | 0.9995111 | 0.00942799 | 0.961154 | 0.9999997 | 0.000009 |

| **benchmark** | 401.bzip2 | | | 429.mcf | | | 456.hmmer | | | 458.sjeng | | | 470.lbm | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **l2_assoc** | d_hit | i _hit | l2_hit | d_hit | i _hit | l2_hit | d_hit | i _hit | l2_hit | d_hit | i _hit | l2_hit | d_hit | i _hit | l2_hit |
| 8 | 0.9820115 | 0.9999996 | 0.02830299 | 0.9970007 | 0.999993 | 0.1744489 | 0.9981499 | 0.999985 | 0.934462 | 0.9024333 | 0.998707 | 0.022402000 | 0.961154 | 0.9999997 | 0.000014 |
| 1 | 0.9820115 | 0.9999996 | 0.0017679 99 | 0.9970007 | 0.999993 | 0.173984 | 0.9981499 | 0.999985 | 0.934462 | 0.9024333 | 0.998707 | 0.022483000 | 0.961154 | 0.9999997 | 0.000014 |
| 2 | 0.9820115 | 0.9999996 | 0.02257699 99 | 0.9970007 | 0.999993 | 0.1519559 | 0.9981499 | 0.999985 | 0.934462 | 0.9024333 | 0.998707 | 0.022473999 | 0.961154 | 0.9999997 | 0.000014 |
| 4 | 0.9820115 | 0.9999996 | 0.0017709999 | 0.9970007 | 0.999993 | 0.17529 | 0.9981499 | 0.999985 | 0.934462 | 0.9024333 | 0.998707 | 0.02248300 | 0.961154 | 0.9999997 | 0.000014 |
| 16 | 0.9820115 | 0.9999996 | 0.0057850 0 | 0.9970007 | 0.999993 | 0.175902 | 0.9981499 | 0.999985 | 0.934462 | 0.9024333 | 0.998707 | 0.02248200 | 0.961154 | 0.9999997 | 0.000014 |

| **benchmark** | 401.bzip2 | 429.mcf | 456.hmmer | 458.sjeng | 470.lbm |
|---|---|---|---|---|---|

| **cacheline_size** | d_hit | i _hit | l2_hit | d_hit | i _hit | l2_hit | d_hit | i _hit | l2_hit | d_hit | i _hit | l2_hit | d_hit | i _hit | l2_hit |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 64 | 0.93980299999 | 0.999988 | 3.23267426 | 0.997007 | 0.999993 | 0.174448 | 0.998149 | 0.999985 | 0.934462 | 0.610377 | 0.997485 | 0.0106410 | 0.96154 | 0.999997 | 0.000014 |
| 16 | 0.982015 | 0.999996 | 1.67386442 0 | 0.986917 | 0.999975 | 0.457879 | 0.987149 | 0.999925 | 0.957487 | 0.902433 | 0.998707 | 0.02248 30 | 0.692337 | 0.999984 | 0.000071 |
| 32 | 0.967944 | 0.999993 | 2.20021566 | 0.990769 | 0.999985 | 0.090 0879 | 0.993121 | 0.999959 | 0.935555 | 0.805096 | 0.9983 | 0.01453999 | 0.84616 6 | 0.999991 | 0.000005 |
| 128 | 0.88352300 | 0.999979 | 5.30680216 | 0.998171 | 0.999995 | 0.292649 | 0.998994 | 0.999987 | 0.936811 | 0.220943 | 0.995849 | 0.930163 | 0.98076 9 | 0.999998 | 0.000029 |
| 8 | 0.989051 | 0.999997 | 1.40985 78 | 0.994943 | 0.999991 | 0.121766 | 0.996456 | 0.999975 | 0.934739 | 0.951031 | 0.998998 | 0.03663899 99 | 0.92308 2 | 0.999995 | 0.000007 |