ECE 521 - Computer Design Techniques

Spring 2014

Project 1 Report

CACHE AND MEMORY HIERARCHY DESIGN

Aravind Sankar - asankar3

Experiments and Reports

Using the simulator, different configurations of the cache and stream buffer are explored and the results of miss rate and the average access times are recorded. From the statistics, the influence of different parameters on the cache performance is discussed.

a) Varying L1 Cache size against Miss Rate

For this test, only consider L1 cache is considered, i.e the stream buffer and the L2 cache are disabled. This test is performed for the four trace files with four different associativity. Two rounds of testing are done for this case, first one is with LRU and Write-Back Write-Allocate policy, second one is with LRU and Write-Through Not-Allocate policy.

For LRU and WBWA policy:

Associativity = 1

Miss rate table -

L1 size	gcc	go	perl	vortex
1024	0.1935	0.2424	0.2801	0.1601
4096	0.1002	0.0738	0.1108	0.0707
16384	0.0461	0.0607	0.0492	0.0338
65536	0.0329	0.0498	0.0202	0.0161

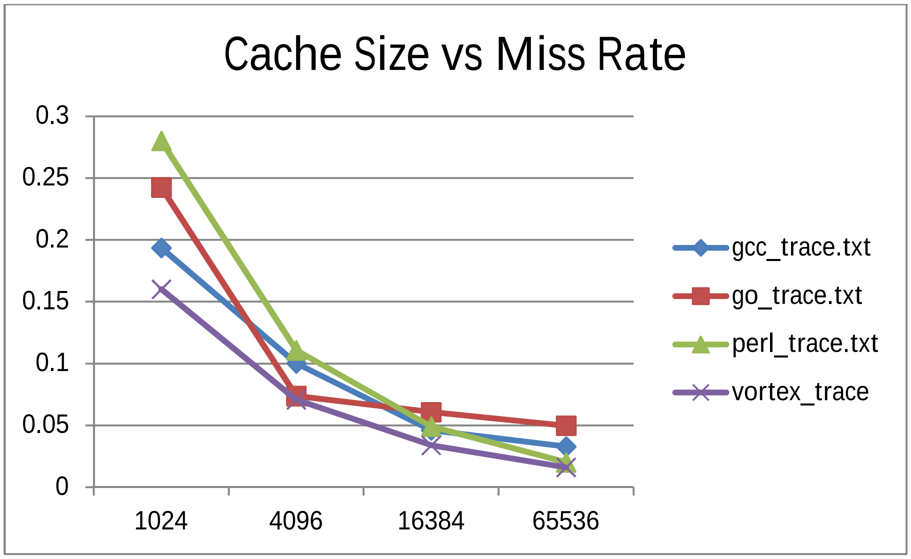
Average access times

gcc_trace.txt: (c:1024) 4.3925, (c:4096) 2.4481, (c:16384) 1.371, (c:65536) 1.3228

go_trace.txt: (c:1024) 5.4192, (c:4096) 1.8933, (c:16384) 1.677, (c:65536) 1.6837

perl_trace.txt: (c:1024) 6.2116, (c:4096) 2.6718, (c:16384) 1.4367, (c:65536) 1.0617

vortex_trace.txt: (c:1024) 3.6924, (c:4096) 1.8294, (c:16384) 1.1131, (c:65536) 0.9762



We can observe from the above graph that the miss rate decreases with increase in cache size. Accordingly, the average access time also reduces for the cache.

Associativity = 2

Miss rate table -

L1 size	gcc	go	perl	vortex
1024	0.156	0.106	0.2404	0.1423
4096	0.0753	0.0559	0.0886	0.0453
16384	0.0338	0.0538	0.0208	0.0174
65536	0.0271	0.0448	0.0152	0.014

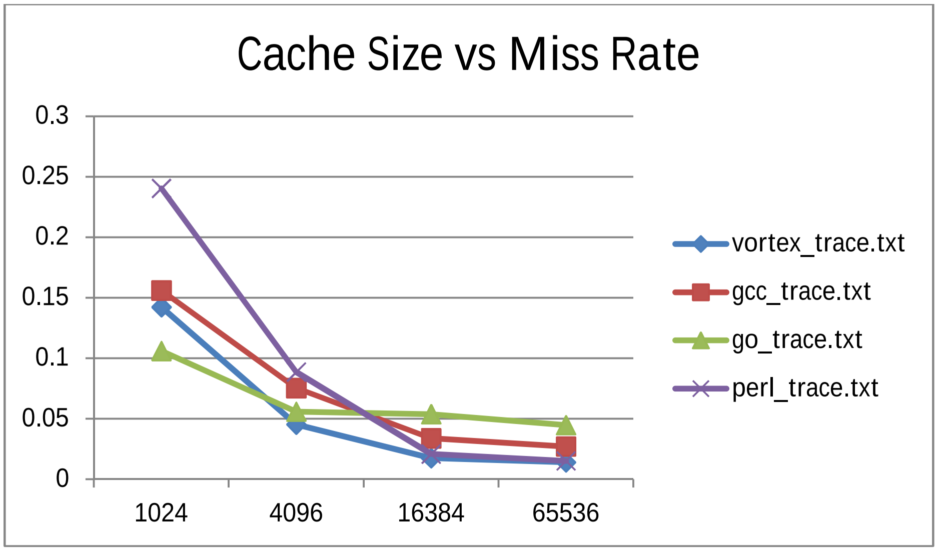
Average access times

gcc_trace.txt: (c:1024) 3.6315, (c:4096) 1.9504, (c:16384) 1.1388, (c:65536) 1.2322

go_trace.txt: (c:1024) 2.5798, (c:4096) 1.5441, (c:16384) 1.5577, (c:65536) 1.6037

perl_trace.txt: (c:1024) 5.4039, (c:4096) 2.2308, (c:16384) 0.8641, (c:65536) 0.9817

vortex_trace.txt: (c:1024) 3.3428, (c:4096) 1.3208, (c:16384) 0.7927, (c:65536) 0.9561



Associativity = 4

Miss rate table -

L1 size	gcc	go	perl	vortex
1024	0.1427	0.0664	0.2345	0.1193
4096	0.0599	0.0542	0.0595	0.0313
16384	0.0283	0.0538	0.0177	0.0158
65536	0.026	0.0467	0.0152	0.0138

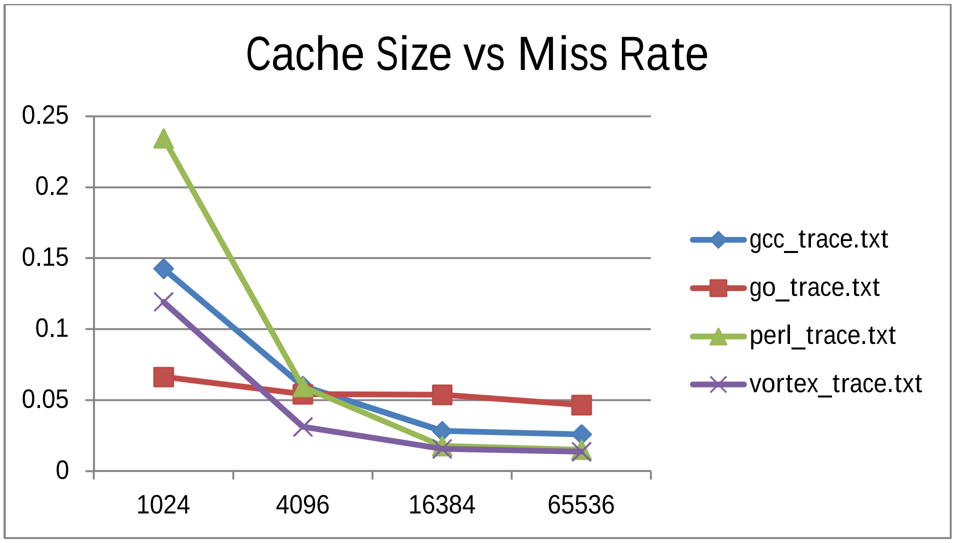
Average access times

gcc_trace.txt: (c:1024) 3.4016, (c:4096) 1.6779, (c:16384) 1.0728, (c:65536) 1.2574

go_trace.txt: (c:1024) 1.7991, (c:4096) 1.5588, (c:16384) 1.6069, (c:65536) 1.6934

perl_trace.txt: (c:1024) 5.3298, (c:4096) 1.6697, (c:16384) 0.849, (c:65536) 1.0306

vortex_trace.txt: (c:1024) 2.9108, (c:4096) 1.0768, (c:16384) 0.8091, (c:65536) 1.0017



We can see that the number of misses reduce as the cache size increases. This can be attributed to the reduction in the number of compulsory and capacitive misses.

Associativity = 8

Miss rate table

L1 size	gcc	go	perl	vortex
1024	0.1363	0.0605	0.229	0.1151
4096	0.0536	0.0538	0.0461	0.029
16384	0.0277	0.0538	0.0168	0.0152
65536	0.0259	0.047	0.0151	0.0138

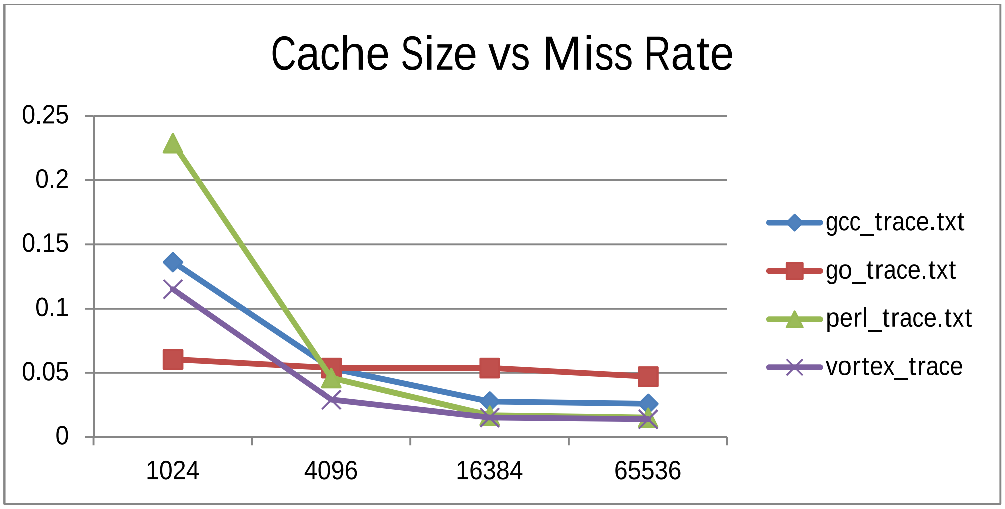
Average access times -

gcc_trace.txt: (c:1024) 3.3666, (c:4096) 1.6462, (c:16384) 1.1607, (c:65536) 1.3562

go_trace.txt: (c:1024) 1.7752, (c:4096) 1.6495, (c:16384) 1.7069, (c:65536) 1.8005

perl_trace.txt: (c:1024) 5.3133, (c:4096) 1.4872, (c:16384) 0.9309, (c:65536) 1.1296

vortex_trace.txt: (c:1024) 2.9226, (c:4096) 1.1277, (c:16384) 0.8969, (c:65536) 1.1015



For LRU and WTNA policy:

For the below configurations, we can observe that the miss rate for write-through is on the higher side compared to the miss rate of the write-back policy.

Associativity = 1

L1 size	gcc	Go	perl	vortex
1024	0.3883	0.4401	0.3701	0.2417
4096	0.2827	0.3693	0.1789	0.1162
16384	0.2171	0.3651	0.1057	0.0705
65536	0.1981	0.3597	0.0671	0.0447

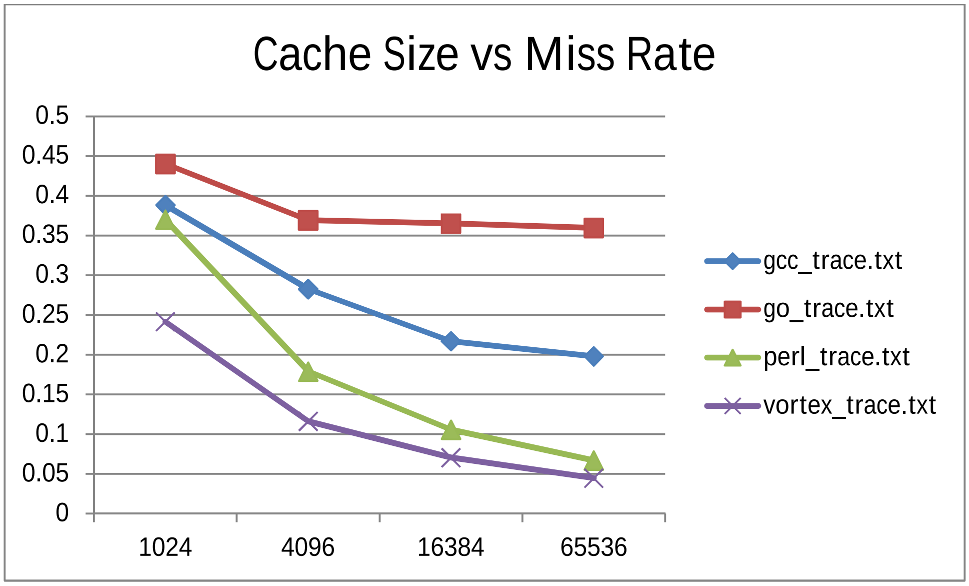
Average access time –

gcc_trace.txt: (c:1024) 8.4848, (c:4096) 6.281, (c:16384) 4.9161, (c:65536) 4.798

go_trace.txt: (c:1024) 9.5716, (c:4096) 8.1005, (c:16384) 8.0964, (c:65536) 8.1906

perl_trace.txt: (c:1024) 8.1011, (c:4096) 4.0945, (c:16384) 2.6232, (c:65536) 2.0462

vortex_trace.txt: (c:1024) 5.4047, (c:4096) 2.7841, (c:16384) 1.8834, (c:65536) 1.5764



Comparing the above graph with Writ-back Write-allocate policy, we can see that the miss rate is higher in Write-through policy.

Associativity = **2**

L1 size	gcc	go	perl	vortex
1024	0.3571	0.3745	0.3222	0.2154
4096	0.2556	0.364	0.143	0.0851
16384	0.1955	0.3632	0.0704	0.0467
65536	0.1878	0.3588	0.0636	0.0416

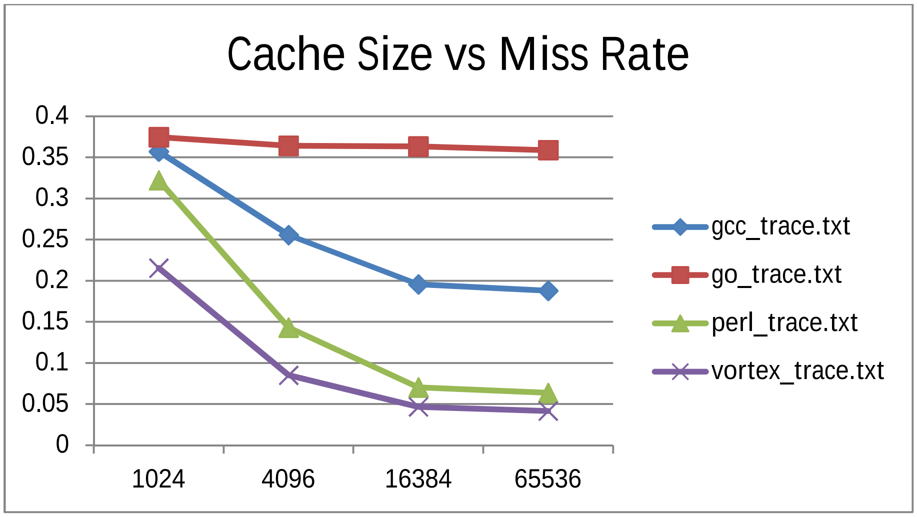
Average access times -

gcc_trace.txt: (c:1024) 7.8531, (c:4096) 5.7369, (c:16384) 4.5388, (c:65536) 4.6061

go_trace.txt: (c:1024) 8.2198, (c:4096) 8.0133, (c:16384) 8.0549, (c:65536) 8.1963

perl_trace.txt: (c:1024) 7.1211, (c:4096) 3.3736, (c:16384) 1.9074, (c:65536) 1.9973

vortex_trace.txt: (c:1024) 4.8779, (c:4096) 2.1575, (c:16384) 1.4084, (c:65536) 1.5357



Similarly, the miss rate is higher in write-through policy. We can see that for go trace, the miss rate is very high compared to the miss rate of write-back policy.

Associativity = 4

L1 size	gcc	go	perl	vortex
1024	0.3489	0.3681	0.3159	0.1888
4096	0.2259	0.3633	0.1159	0.0674
16384	0.1883	0.3622	0.0661	0.0437
65536	0.1862	0.3589	0.0635	0.0412

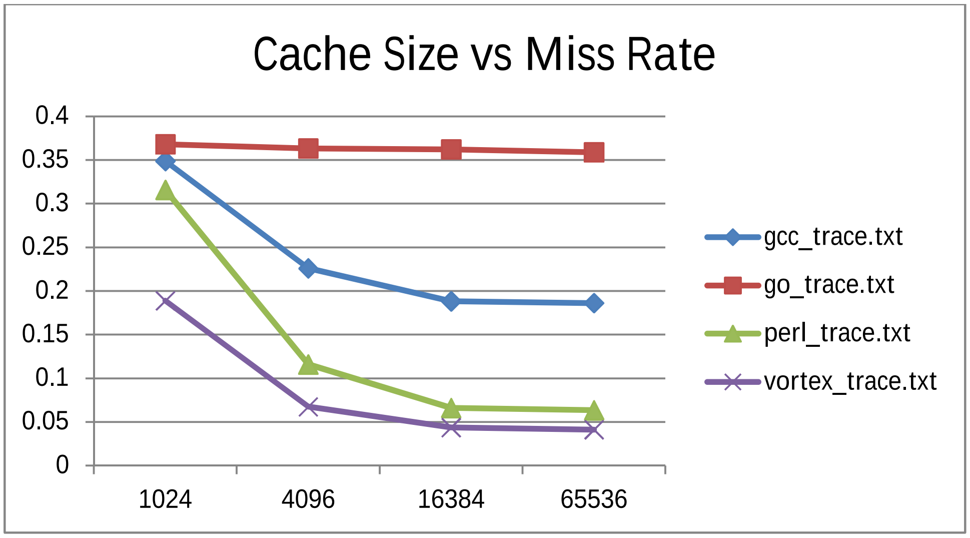
Average access times -

gcc_trace.txt: (c:1024) 7.7316, (c:4096) 5.1362, (c:16384) 4.4322, (c:65536) 4.6238

go_trace.txt: (c:1024) 8.1358, (c:4096) 8.0484, (c:16384) 8.1043, (c:65536) 8.25

perl_trace.txt: (c:1024) 7.0386, (c:4096) 2.8532, (c:16384) 1.8664, (c:65536) 2.0454

vortex_trace.txt: (c:1024) 4.3703, (c:4096) 1.8351, (c:16384) 1.395, (c:65536) 1.5769



The observations in this configuration is similar to the previous configuration.

Associativity = 8

L1 size	gcc	go	perl	vortex
1024	0.3458	0.3677	0.3025	0.1838
4096	0.2185	0.3622	0.1024	0.065
16384	0.1877	0.3632	0.0652	0.0429
65536	0.1862	0.3581	0.0634	0.0412

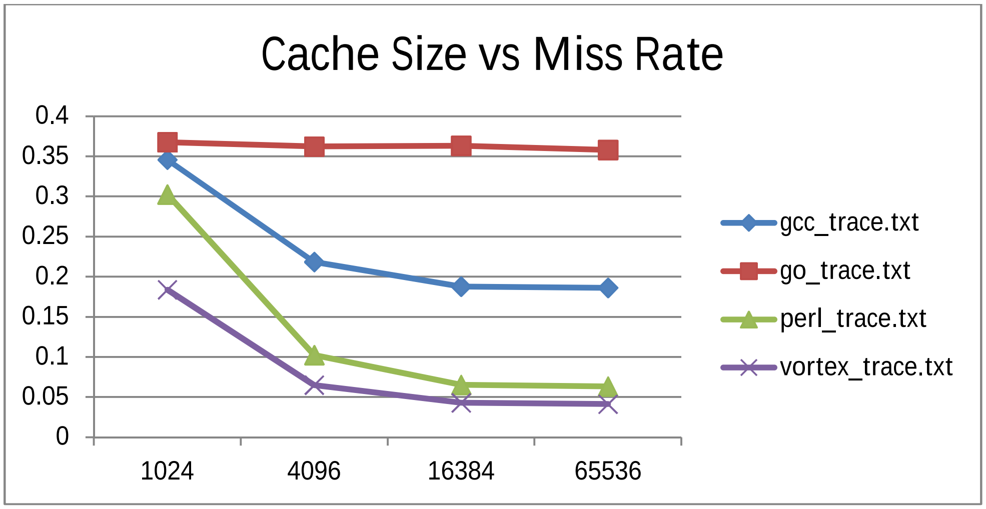
Average access times -

gcc_trace.txt: (c:1024) 7.761, (c:4096) 5.1085, (c:16384) 4.5196, (c:65536) 4.7238

go_trace.txt: (c:1024) 8.226, (c:4096) 8.1467, (c:16384) 8.2043, (c:65536) 8.332

perl_trace.txt: (c:1024) 6.8578, (c:4096) 2.6689, (c:16384) 1.9465, (c:65536) 2.1441

vortex_trace.txt: (c:1024) 4.3638, (c:4096) 1.8845, (c:16384) 1.4799, (c:65536) 1.6766



We can see that higher cache sizes do not impact on the miss rate of the files. Hence we have more number of capacitive misses compared to conflict misses in this case, especially in go trace file.

b) Varying Associativity against Miss Rate

To compare the associativity against miss rate, the L1 and L2 cache size are kept constant for this test. The L1 cache associativity is varied and the miss rate is record for each of the four trace files. Below was the configuration of the system when this test was performed.

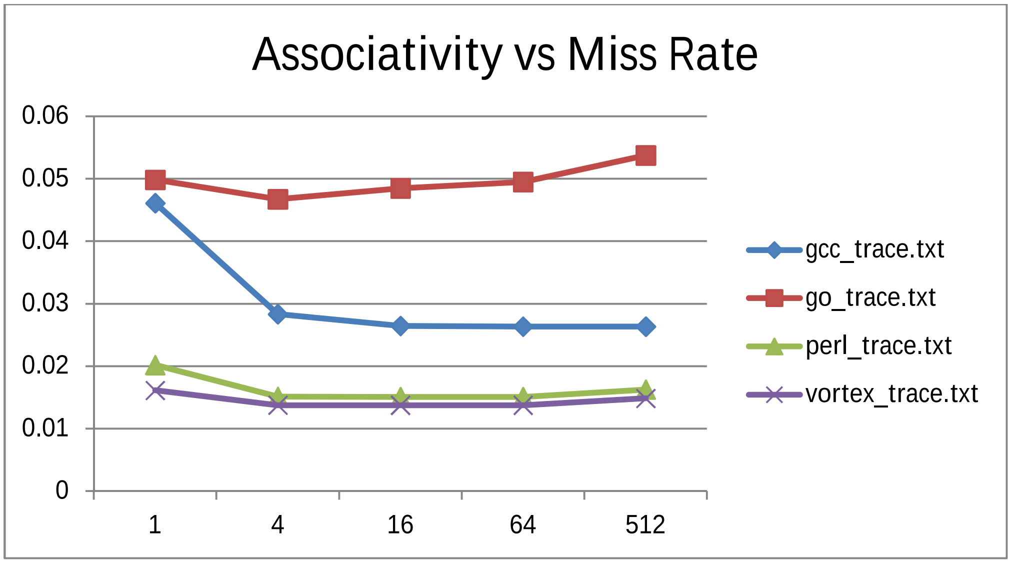
L1 Cache Size = 16KB

L2 Cache Size = 64KB

Block Size = 32

L2 Associativity = 8

Associativity	gcc	go	perl	vortex
1(Direct)	0.04609	0.04982	0.0202	0.01613
4	0.02832	0.04671	0.01515	0.01377
16	0.02646	0.04846	0.0151	0.01376
64	0.02634	0.04949	0.0151	0.01376
512(fully)	0.02634	0.05374	0.01628	0.01485



The number of misses is on the higher side for fully associative cache, i.e there are less number of conflict misses for this case as fully associative cache has higher number of misses. Thus, we can say that for scenarios where the conflict misses are less, fully associative cache is not the right choice.

c) Varying L2 Cache size against Miss Rate

Keeping the configurations of L1 cache constant, the size of the L2 cache is varied for each of the trace files. For this test, the stream buffer was disabled, so only L1 and L2 cache were operational. Below is the configuration of cache during this test.

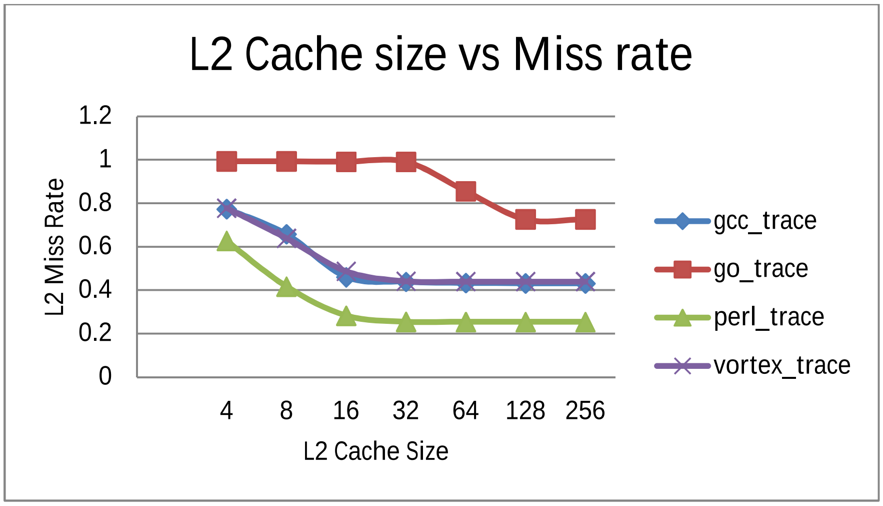
Block size = 32 B

L1 Cache size = 4 KB

L1 Cache associativity = 4

L2 Cache associativity = 8

L2 size	gcc_trace.txt	go	perl	vortex
4	0.773031	0.99318	0.627247	0.777316
8	0.657543	0.992811	0.416765	0.638978
16	0.45978	0.990968	0.282379	0.486901
32	0.438084	0.990599	0.254158	0.441214
64	0.432577	0.8553	0.253654	0.439617
128	0.430908	0.726083	0.253654	0.439617
256	0.430908	0.726083	0.253654	0.439617



As expected, we can see that the miss rate reduces as the size of the L2 cache is increased. However, after a certain size of L2 cache, the miss rate remains constant irrespective of the size of the L2 cache.

d) Vary number of stream buffers vs. miss rate (Keep L1 and L2 constant)

In this test, the number of stream buffer rows is varied and the performance of the L2 cache is measured by recording the miss rate. During this test, configuration of L1 cache and L2 cache was kept constant. Below is the configuration of the two cache for this test.

Block size = 32 B

L1 Cache size = 64 KB

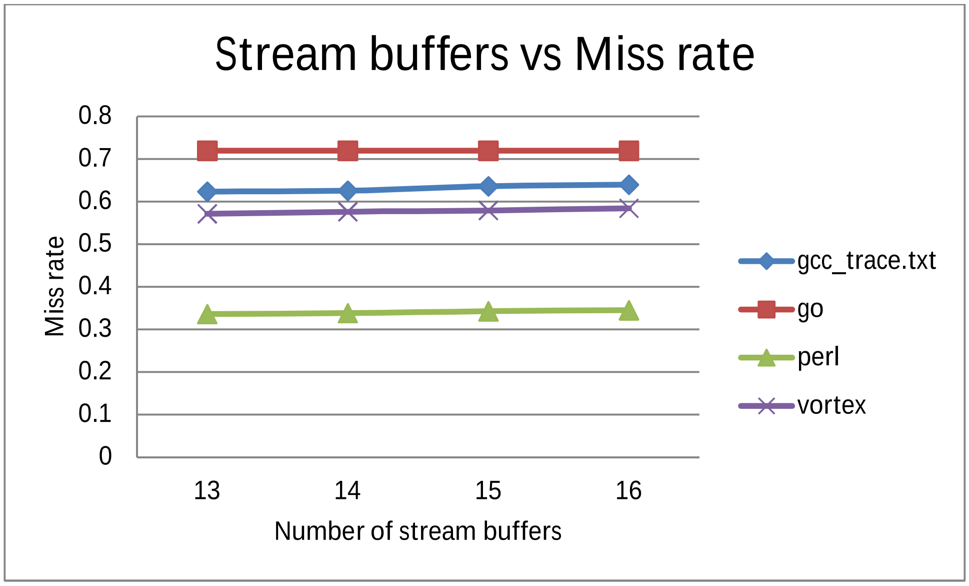
L1 Cache associativity = 4

L2 Cache Size = 256 KB

L2 Cache associativity = 8

Number of blocks per stream buffer = 5

# of Stream buffers	gcc_trace.txt	go	perl	vortex
13	0.623188	0.719101	0.336093	0.571184
14	0.625455	0.719101	0.338333	0.576125
15	0.63586	0.719101	0.342327	0.57913
16	0.639405	0.719101	0.344652	0.584211



We can see that the number of rows in the stream buffer does not impact much on the performance of the cache, as the miss rate remains almost constant for increasing number of rows in the stream buffer. It can also be noted that the miss rate of the L2 cache marginally increases for gcc and vortex trace files.

e) Vary depth of each stream vs miss rate

In this test, the number of stream buffer are kept constant while the number of blocks per stream buffer is varied and the performance of the L2 cache was measured by recording the miss rate. During this test, configuration of L1 cache and L2 cache is kept constant. Below is the configuration of the two cache for this test.

Block size = 32 B

L1 Cache size = 64 KB

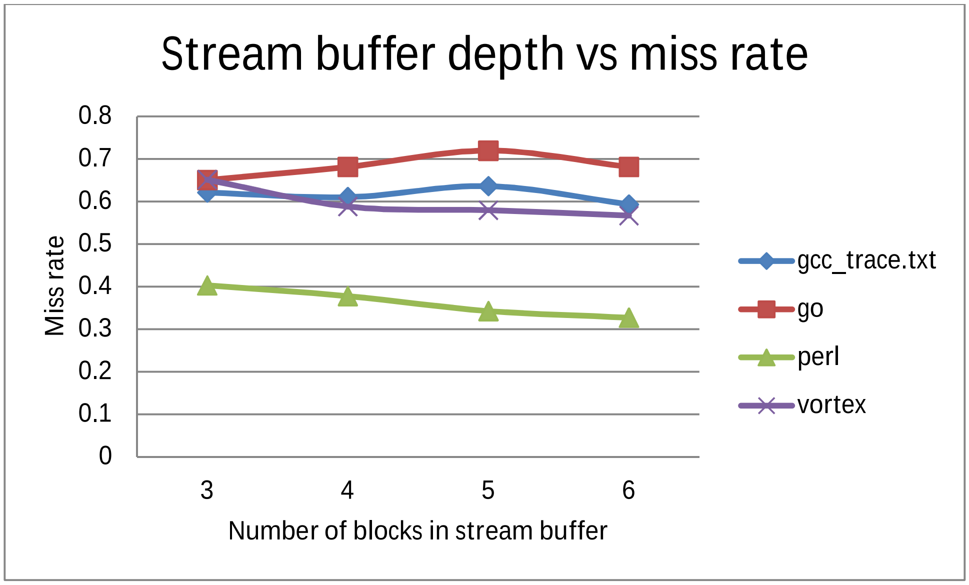
L1 Cache associativity = 4

L2 Cache Size = 256 KB

L2 Cache associativity = 8

Number of stream buffers = 15

Stream buffer blocks	gcc_trace.txt	go	perl	vortex
3	0.620853	0.65	0.402635	0.65109
4	0.609715	0.680851	0.377451	0.58794
5	0.63586	0.719101	0.342327	0.57913
6	0.592734	0.680851	0.326957	0.566547



Increasing the number of blocks per stream buffer eventually improves the performance of the cache, however there is a decrease in performance for go trace and gcc trace. Thus, increasing the blocks per stream buffer improves the performance of both L1 and L2 cache.

Best memory hierarchy configuration

Looking at the above different configurations, we can see that the average access time for each traces reduces with the presence of stream buffers for cache L1. Having a higher number of blocks per stream buffer might increase the time to fetch the block to the cache, but it further reduces the average access time.

Configuration -

Block size = 32 B

L1 Cache size = 64 KB

L1 Cache associativity = 4

L2 Cache Size = 256 KB

L2 Cache associativity = 8

Number of stream buffers = 15

Number of blocks per stream buffer = 10

Due to the presence of the stream buffer, the number of capacitive and conflict misses is reduced largely, thus leading to reduced average access time. Moreover, if an additional stream buffer is introduced for L2 cache, the average access time will further reduce.

Comparing and contrasting different benchmarks

The configuration mentioned above produces the same result for all the benchmarks. However, the results are dependent on the kind of inputs and may vary depending on the sizes of the cache and the stream buffer.

Thus, while designing a cache hierarchy, it is important to consider the size of the cache and the stream buffer as this factor impacts largely on the performance. Stream buffers help reduce the number of compulsory misses, while large cache size reduces the number of conflict and capacity misses. Although the best configuration depends on the scenario, if we consider the above different tests run, presence of stream buffer with the above configuration gives similar results for all the traces.