COL216 Assignment 3

Kushagra Gupta (2021CS50592)
Parth Patel (2021CS10550)

May 11, 2023

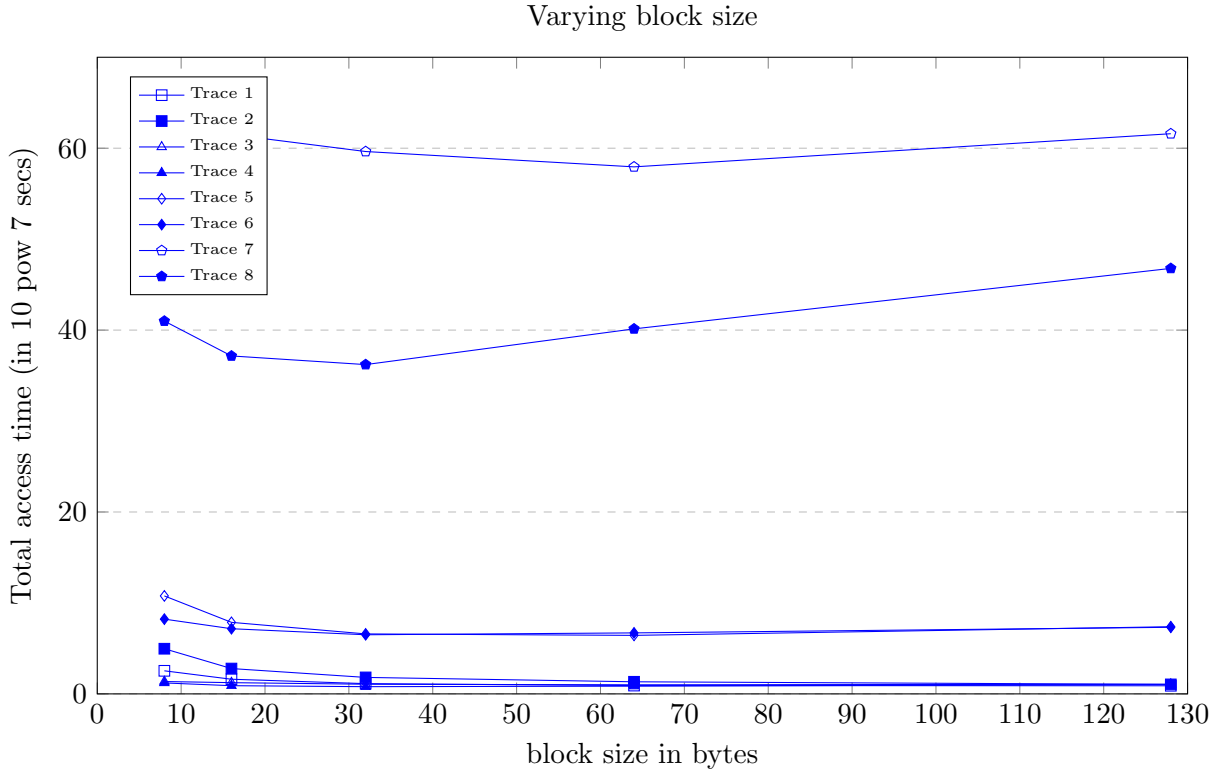
§1 Overview

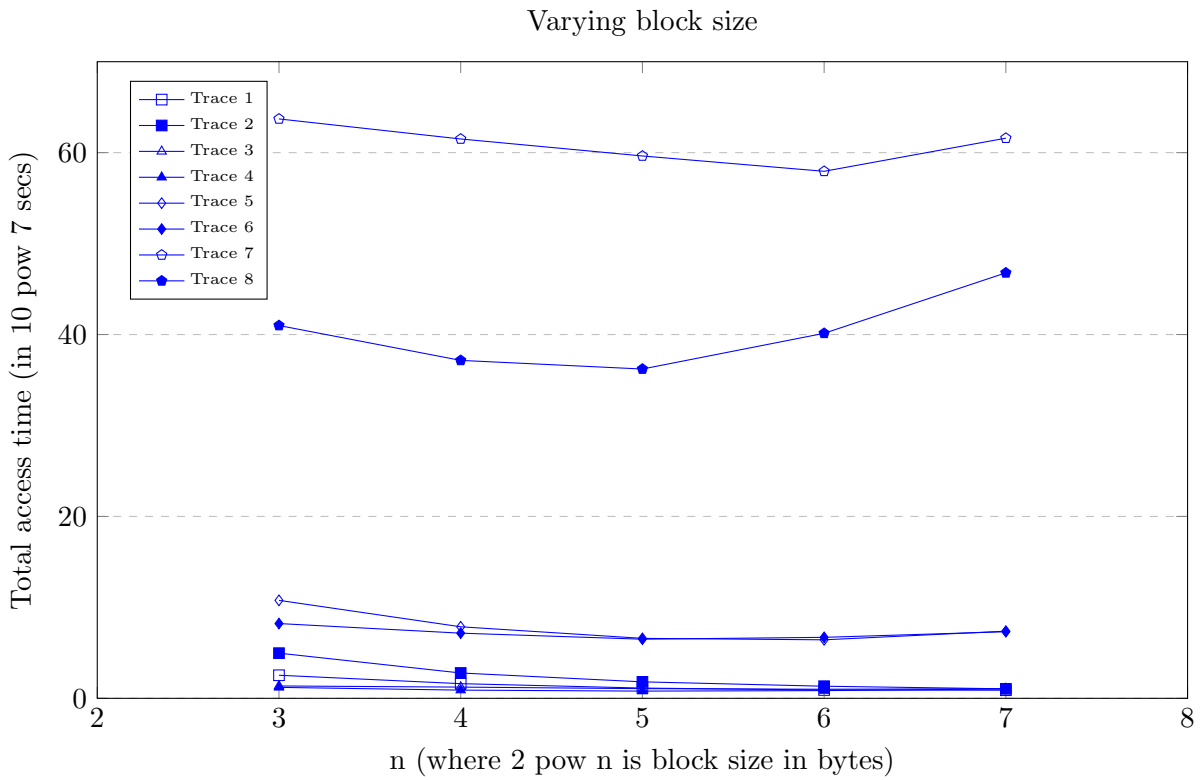
We have implemented a simulation for cache heirarchy. L1 and L2 are two caches. Write Back and Write Allocate Policy is being used along with the LRU (least replacement) policy. the caches are can be n-way set associativity.

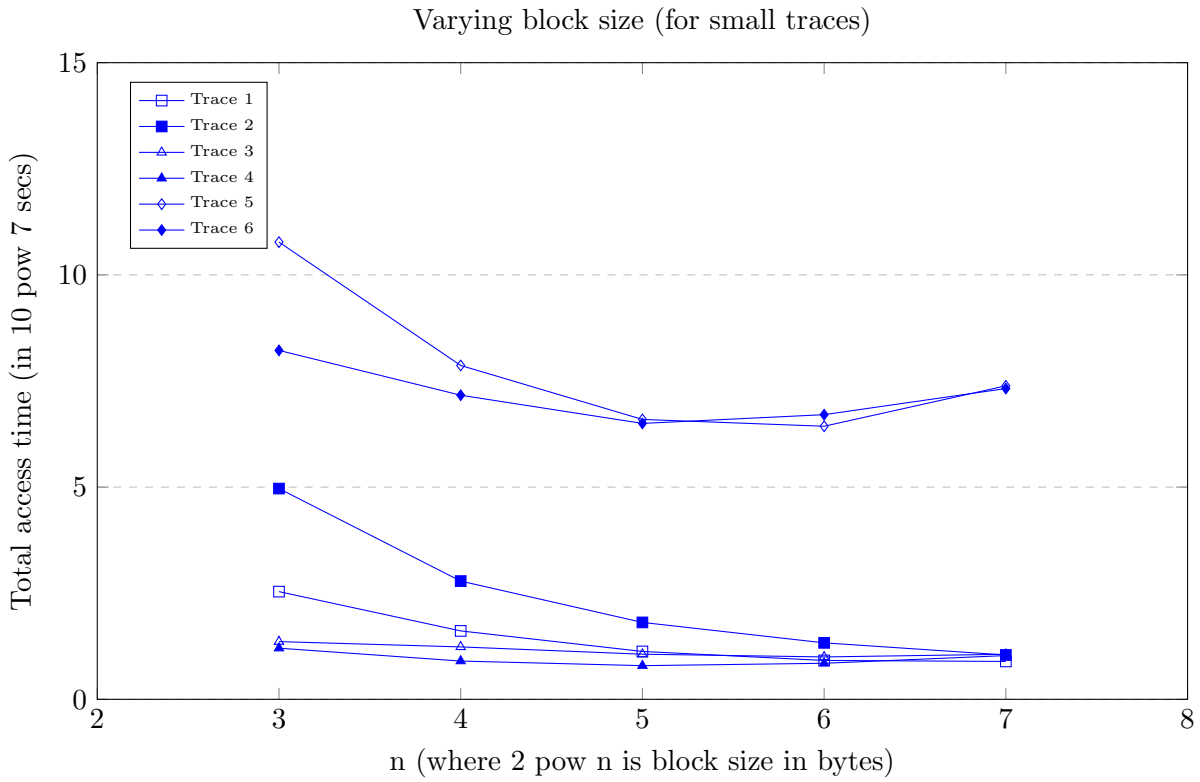
§2 Total access time comparison plots

§2.1 Varying block size

Vary block size on x-axis between 8, 16, 32, 64, 128, and keep L1 size, L1 associativity, L2 size and L2 associativity fixed at 1024, 2, 65536, 8.

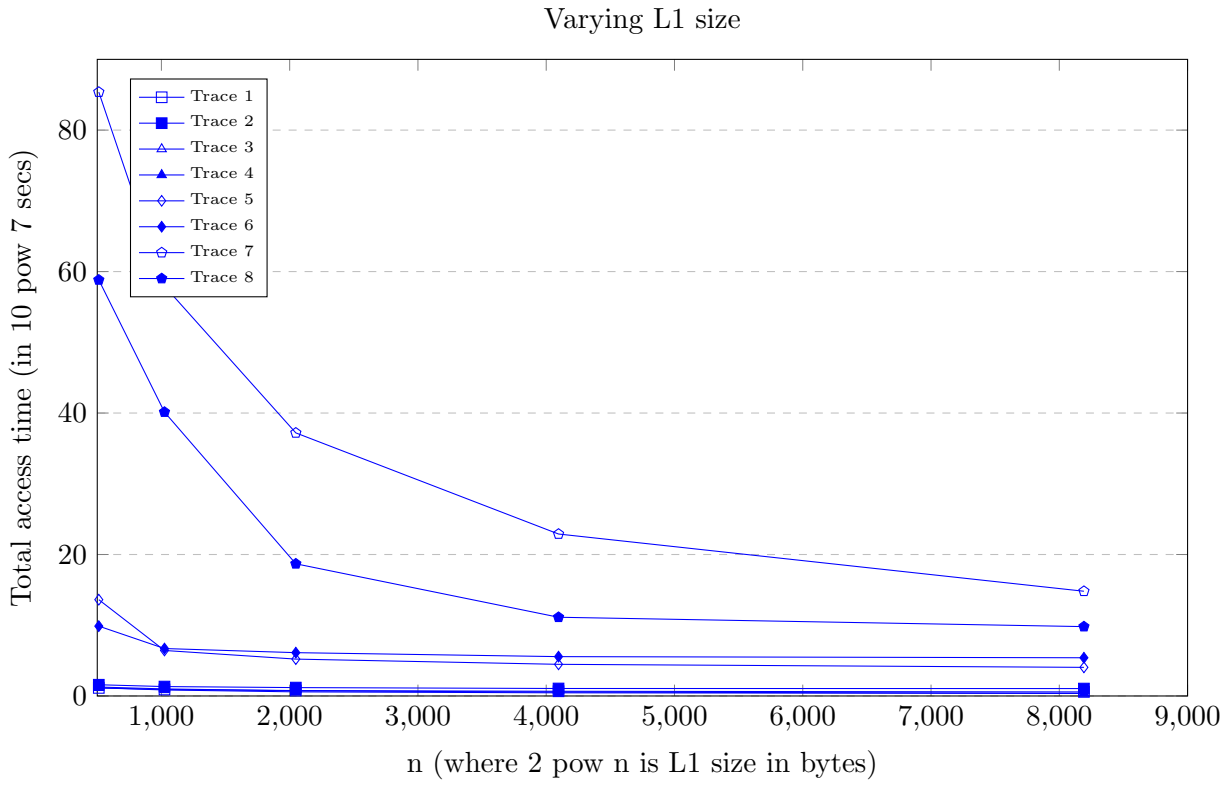


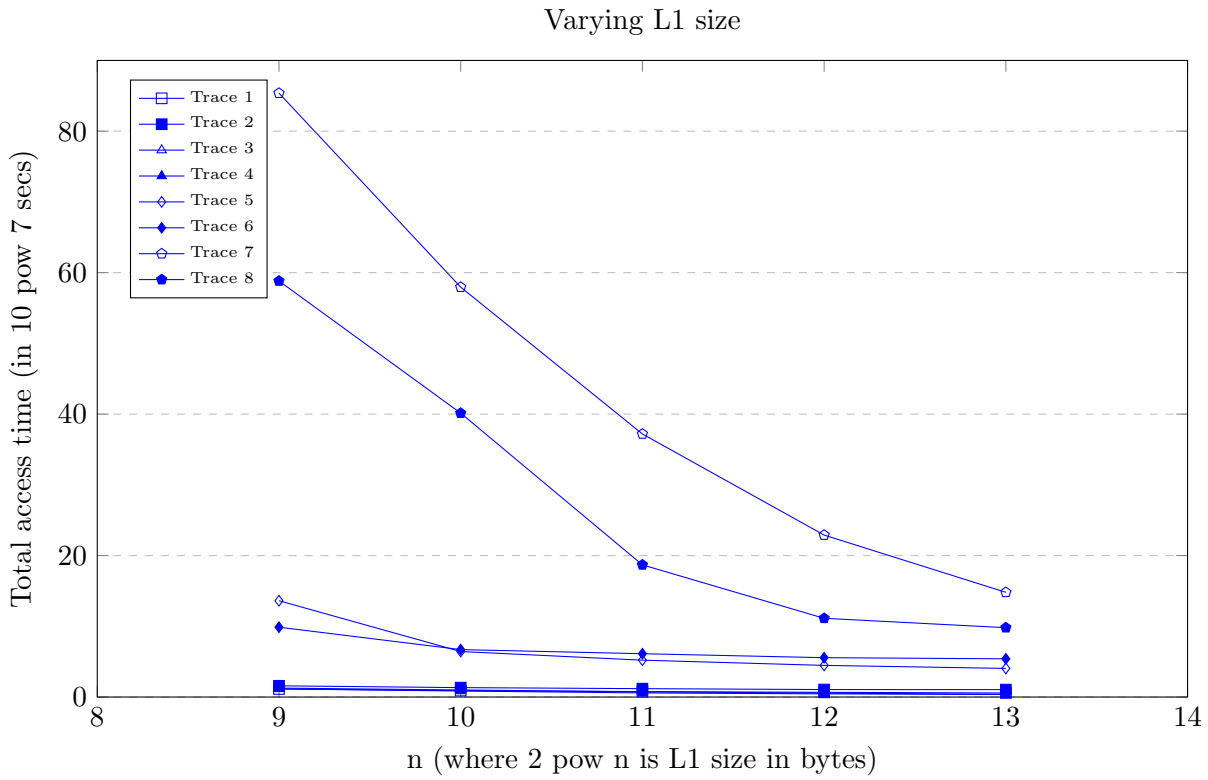


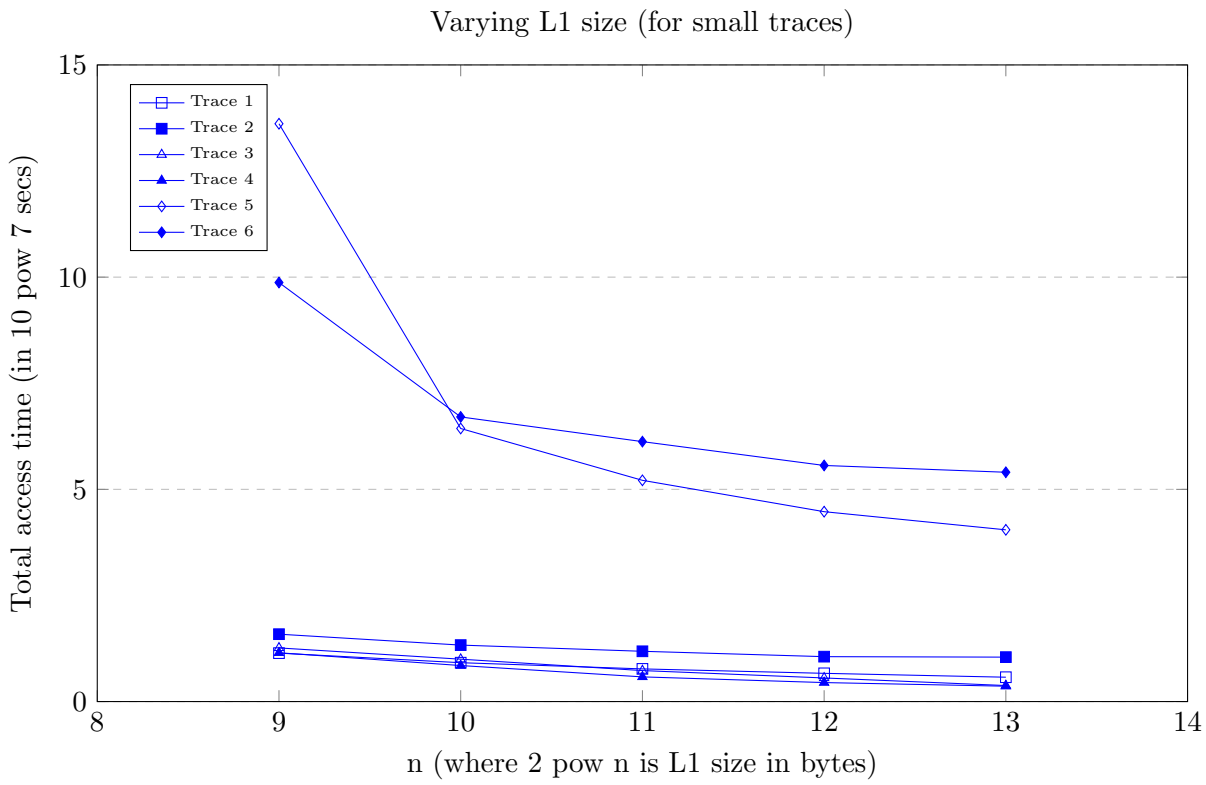


§2.2 Varying L1 size

Vary the second parameter L1 size between 512, 1024, 2048, 4096, 8192, and keep block size, L1 associativity, L2 size and L2 associativity fixed at 64, 2, 65536, 8

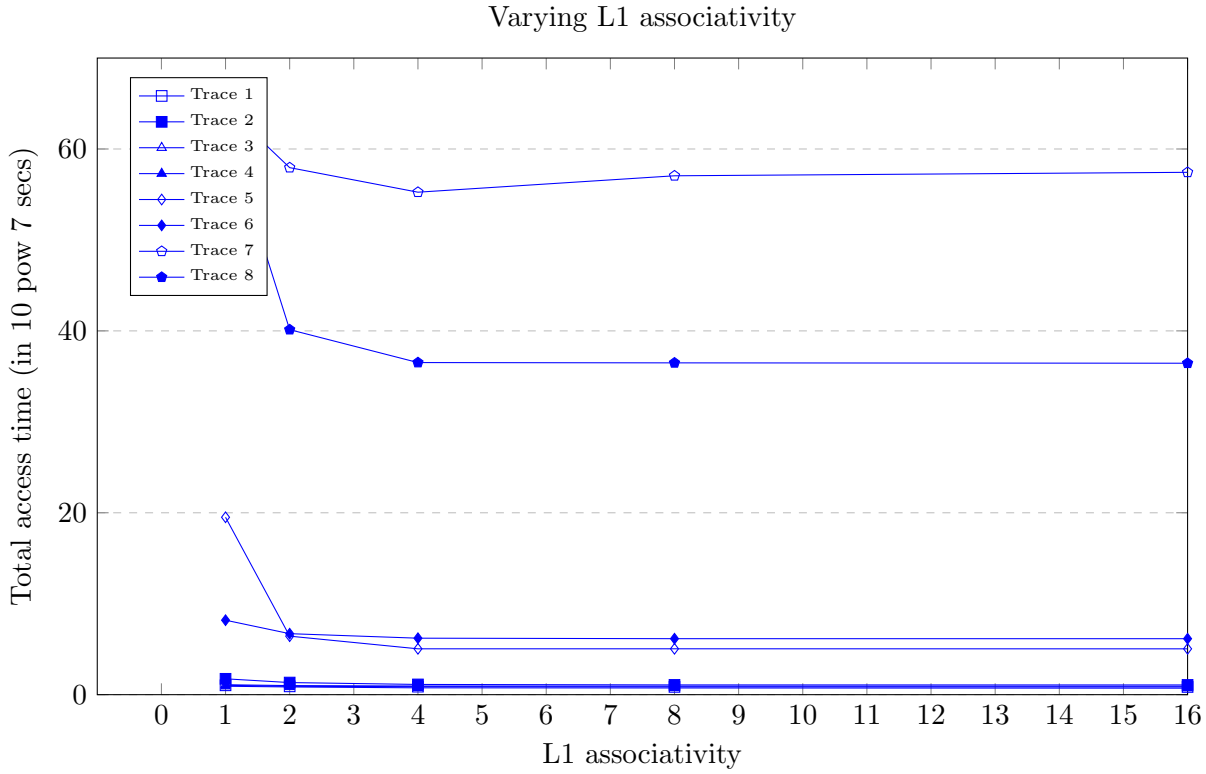

Varying L1 size
Trace 1
Trace 2
Trace 3
Trace 4
Trace 5
Trace 6
Trace 7
Trace 8
Total access time (in 10 pow 7 secs)
0
20
40
60
80
1,000
2,000
3,000
4,000
5,000
6,000
7,000
8,000
9,000
n (where 2 pow n is L1 size in bytes)

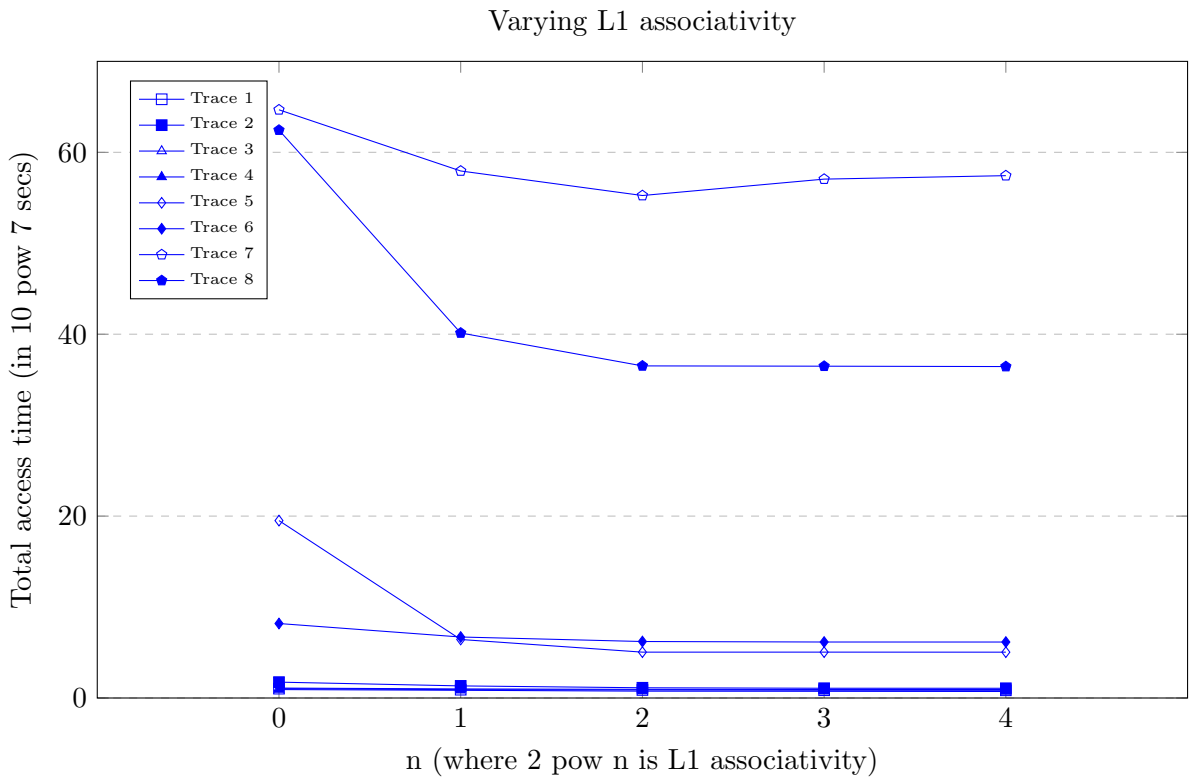

Varying L1 size
Trace 1
Trace 2
Trace 3
Trace 4
Trace 5
Trace 6
Trace 7
Trace 8
Total access time (in 10 pow 7 secs)
0
20
40
60
80
8
9
10
11
12
13
14
n (where 2 pow n is L1 size in bytes)

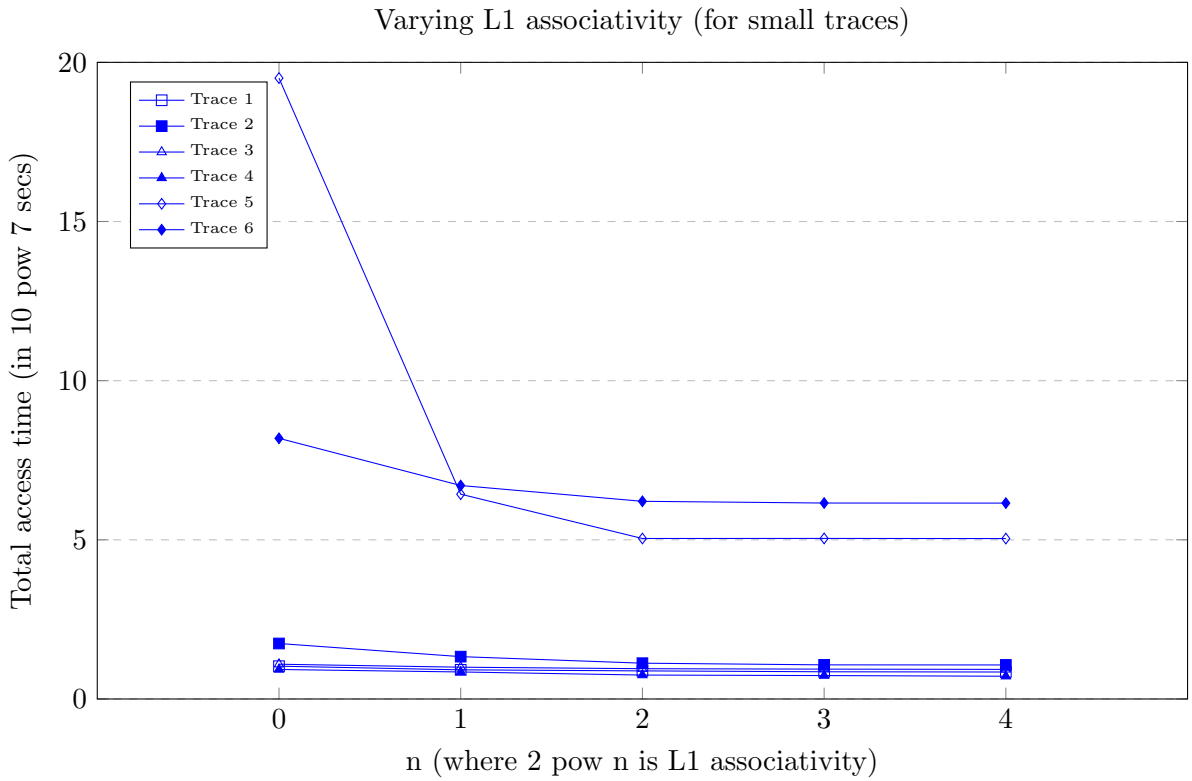


§2.3 Varying L1 associativity

Vary the third parameter L1 associativity between 1, 2, 4, 8, 16, and keep block size, L1 size, L2 size and L2 associativity fixed at 64, 1024, 65536, 8.

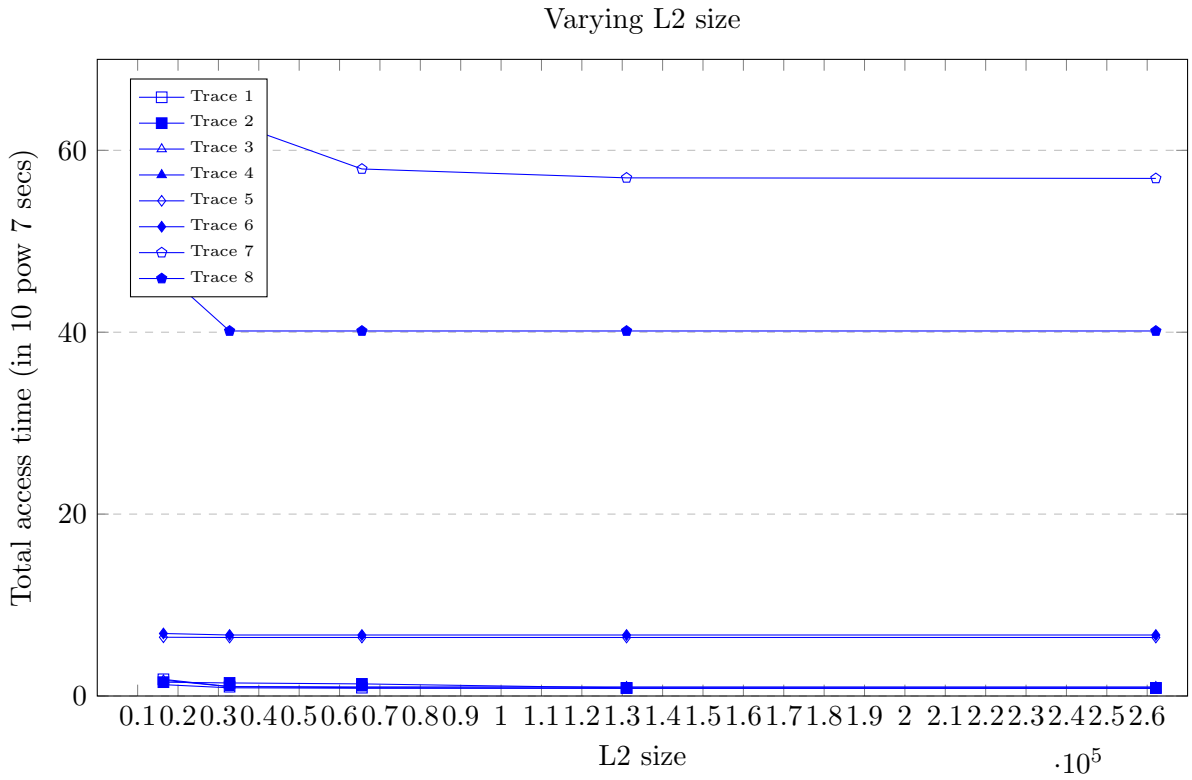


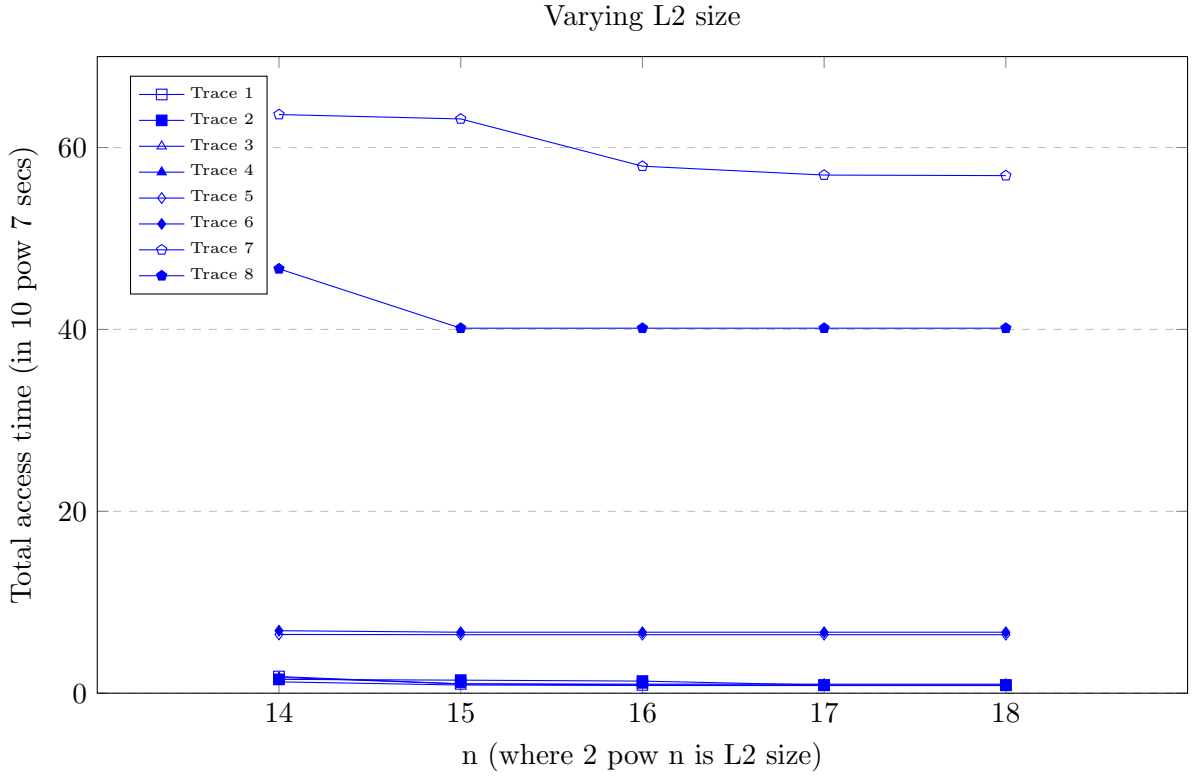


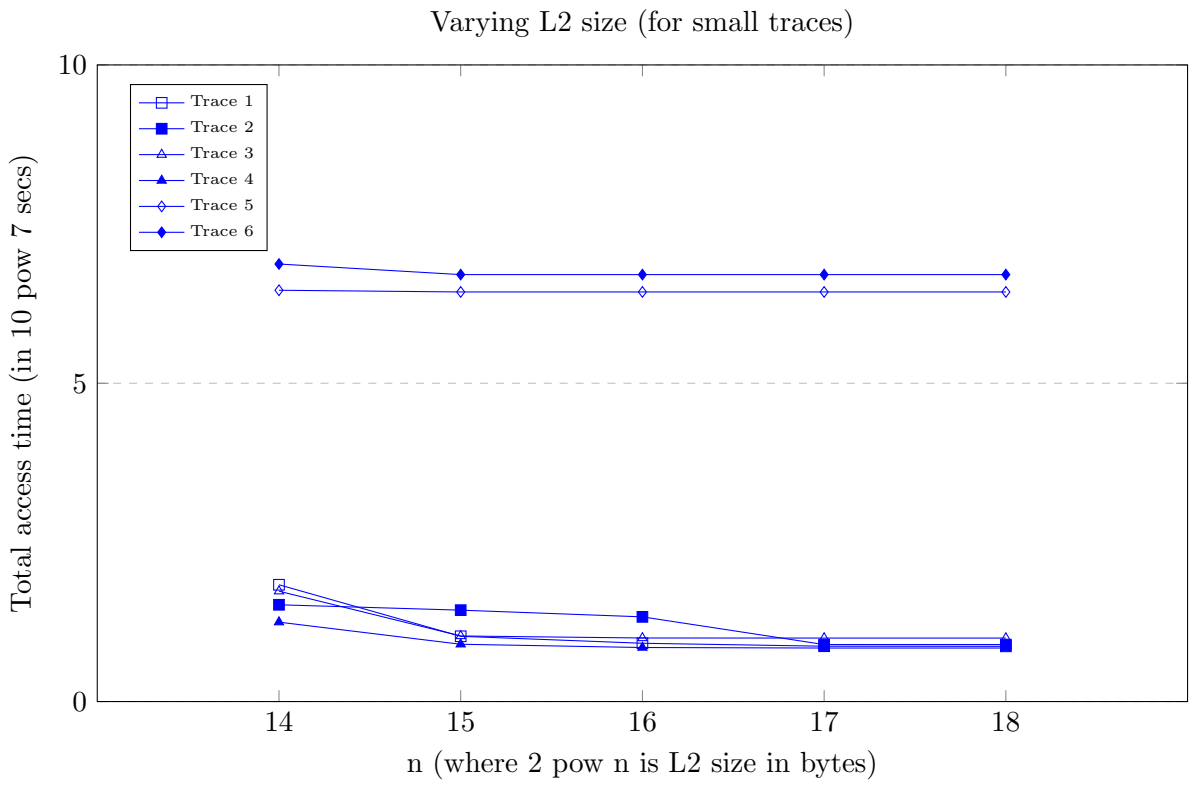


§2.4 Varying L2 size

Vary the fourth parameter L2 size between 16384, 32768, 65536, 131072, 262144, and keep block size, L1 size, L1 associativity and L2 associativity fixed at 64, 1024, 2, 8.

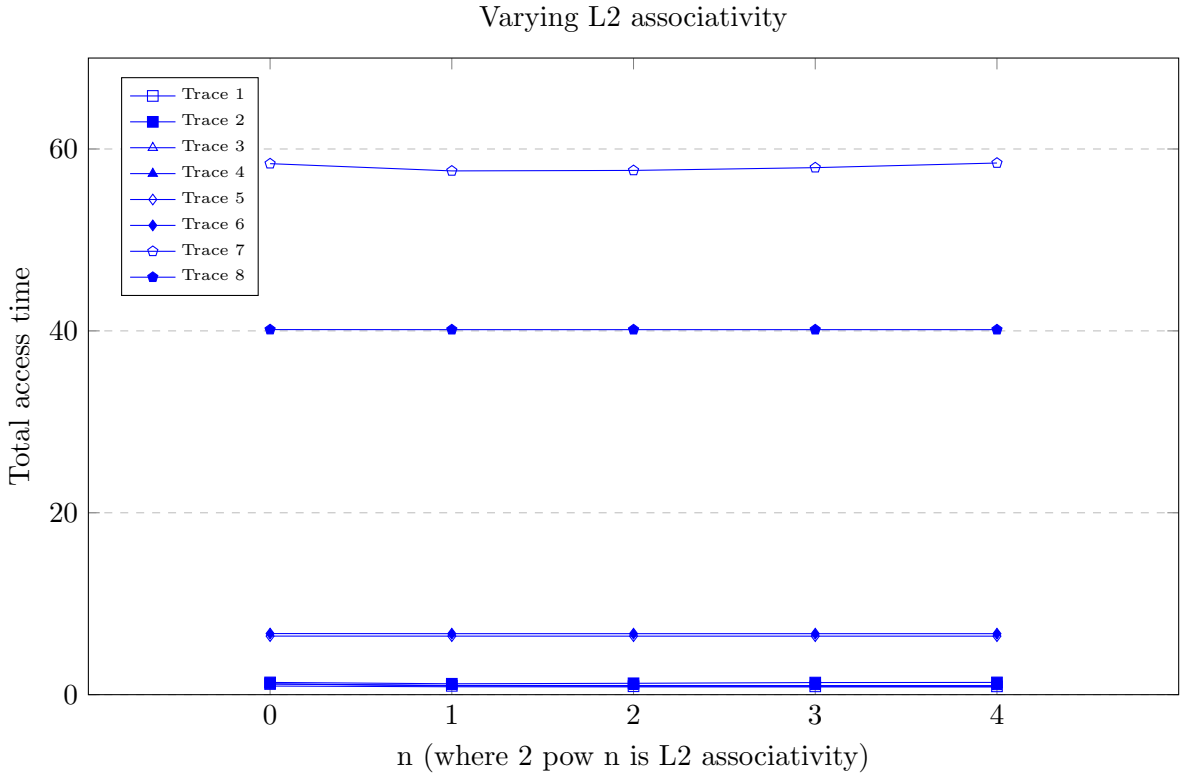

Varying L2 size
Trace 1
Trace 2
Trace 3
Trace 4
Trace 5
Trace 6
Trace 7
Trace 8
Total access time (in 10 pow 7 secs)
0
20
40
60
0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2 2.1 2.2 2.3 2.4 2.5 2.6
L2 size
$\cdot 10^5$


Varying L2 size
Trace 1
Trace 2
Trace 3
Trace 4
Trace 5
Trace 6
Trace 7
Trace 8
Total access time (in 10 pow 7 secs)
0
20
40
60
14
15
16
17
18
n (where 2 pow n is L2 size)



§2.5 Varying L2 associativity

Vary the fifth parameter L2 associativity between 1, 2, 4, 8, 16, and keep block size, L1 size, L1 associativity and L2 size fixed at 64, 1024, 2, 65536.



§3 Observations and Analysis

§3.1 Varying block size

Observation : For the large sized traces, i.e. 7 and 8, we observe that total access time decreases and then increases. This is in contrast to small sized traces where the total access time is decreasing monotonously as we increase the block size.

Explanation : For small sized traces, increasing the block size typically reduces the number of cache misses and improves cache hit rates. As a result, the total access time generally decreases monotonously as the block size increases. This is because the cache is able to serve more memory requests from caches rather than fetching data from slower, main memory.
However, for large sized traces, as the block size increases, the size of each cache line also increases, which means that more data is fetched from main memory in each cache line. This can lead to a higher cache pollution, where the cache becomes filled with data that is not likely to be reused soon. This can cause more cache misses and reduce the performance of the cache, increasing the total access time.
On the other hand, if the access patterns of the application exhibit temporal and spatial locality, larger block sizes can result in better cache utilization, leading to fewer cache misses and reducing the total access time

§3.2 Varying L1 size

Observation : For the large sized traces, i.e. 7 and 8, we observe that total access time significantly decreases as compared to smaller trace files. Smaller trace files' total access time also decreases initially but the gradient after that is very less.
Explanation : For small traces, the working set size (i.e., the set of memory blocks accessed during the execution of the program) is small, so the data in the working set is more likely to fit in the cache regardless of its size. Therefore, increasing the size of the L1 cache may not result in a significant reduction in cache misses and total access time for small traces because the cache is already able to hold most of the data. Saturation will occur for traces 7 and 8 when L2 size is increased (see 3.4).
On the other hand, for larger traces, the working set size is larger and may not fit entirely in the cache, leading to more cache misses and longer access times. In this case, increasing the size of the L1 cache can result in a significant reduction in cache misses and total access time because it can hold a larger portion of the working set and serve more memory requests from its fast memory.

§3.3 Varying L1 associativity

Observation : While moving from only one way associative cache to two way associative cache, the total access time decreases significantly but then there is not much further reduction compared to first step.
Explanation : A two way set associative cache will avoid the conflict misses compared to one way associative cache. Further increase will also reduce the total access time but not to too much extent because while moving from 1 to 2, most conflict cases have been already been resolved. Capacity misses are also reduced along with conflict misses.

§3.4 Varying L2 size

Observation and Analysis: The total access time is almost constant for smaller traces because since if saturation occurs in L1, then obviously since L2 size is greater than L1, total access time remains almost same. The decrease will be observed only in larger trace files. for trace 7, decrease is observed at higher n as compared to trace 8. After some n, again saturation will occur similar to the case where saturation occurs for smaller traces while varying L1 size.

§3.5 Varying L2 associativity

Observation : The total access time is almost constant for all the traces as we increase L2 associativity.
Explanation : L2 cache is already at second level, so it is accessed less. Also due to its large size, miss rate it less. Therefore increasing L2 associativity has little or no effect in total access time. If we want to reduce access time, we should increase size of L2 cache than L2 associativity. Less capacitive miss and even lesser conflict misses.

§4 Additional Note on Inclusivity (L1 is a subset of L2)

We have implemented our 2 level cache without inclusivity, so we mark our bit as dirty and do not propagate the updates from L2 to L1, in the case other than eviction. (We do not have the relation that L1 is a subset of L2.)
So, if we have a read miss from L1 cache then we want to fetch the read address from L2 cache to L1 cache so if here the block that we are going to replace from L1 is a dirty block then:- We write back the dirty block first to L2 cache (here it can replace our original read address block too) and then fetch the read address from L2.
The comparision between both of the implementations yield similar results on the given trace files but not in all case in general.

§5 Token Distribution

Both team members made equal contributions and efforts. Hence a 50-50 split
Kushagra - (Code and Logic)
Parth - (Report and starter code)