Memory Simulator

Pravar Kataria 2021CS10075
Suren 2021CS10072

May 11, 2023

§1 Overview of design

The design consists of two caches L1 & L2. Both the caches are n-way associative and use the following policies for their implementation.

§1.1 Write Allocate

Given a write operation at a specific address, the cache and memory are searched for the address. The block which is just 'called' should be brought into the L1 and in L2 as well (or should be updated accordingly) even if the action done on block is to write. This is called as the write allocate policy.

§1.2 Write back policy

In case of a write hit, it is then performed on the cache block instead of the memory.The cache block is marked as "dirty". When the cache block is evicted, the last updated data is written back to the DRAM. Also the data is not written to the L2 if the changes are incurred in L1. It is only written if a dirty block is evicted and the new data will be lost in eviction, hence it is written back.

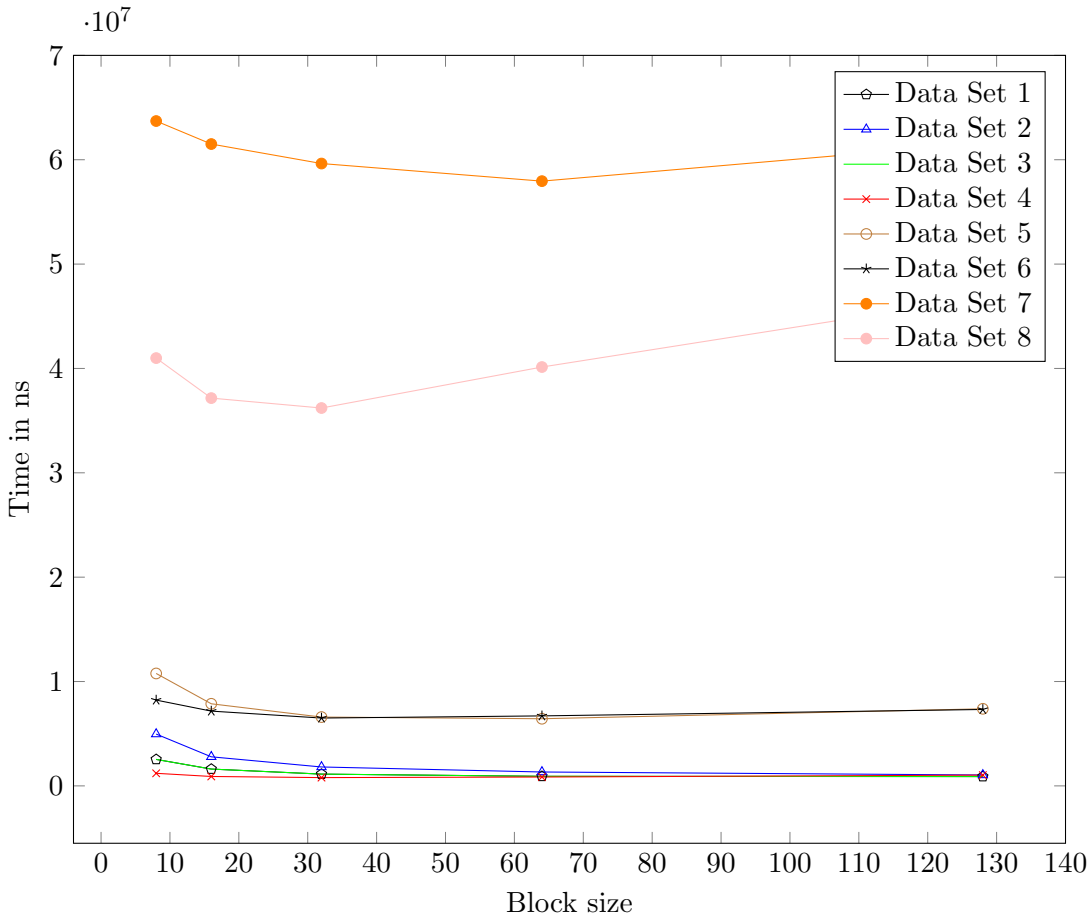
§1.3 Least replacement policy for replacement

LRU (Least Recently Used) is the policy for cache eviction policies. The main idea behind the LRU policy is to note the last used block, i.e. the block that was touched the longest ago. and when the cache is filled,the new block replaces the least recently used item.

§2 Plots for comparison

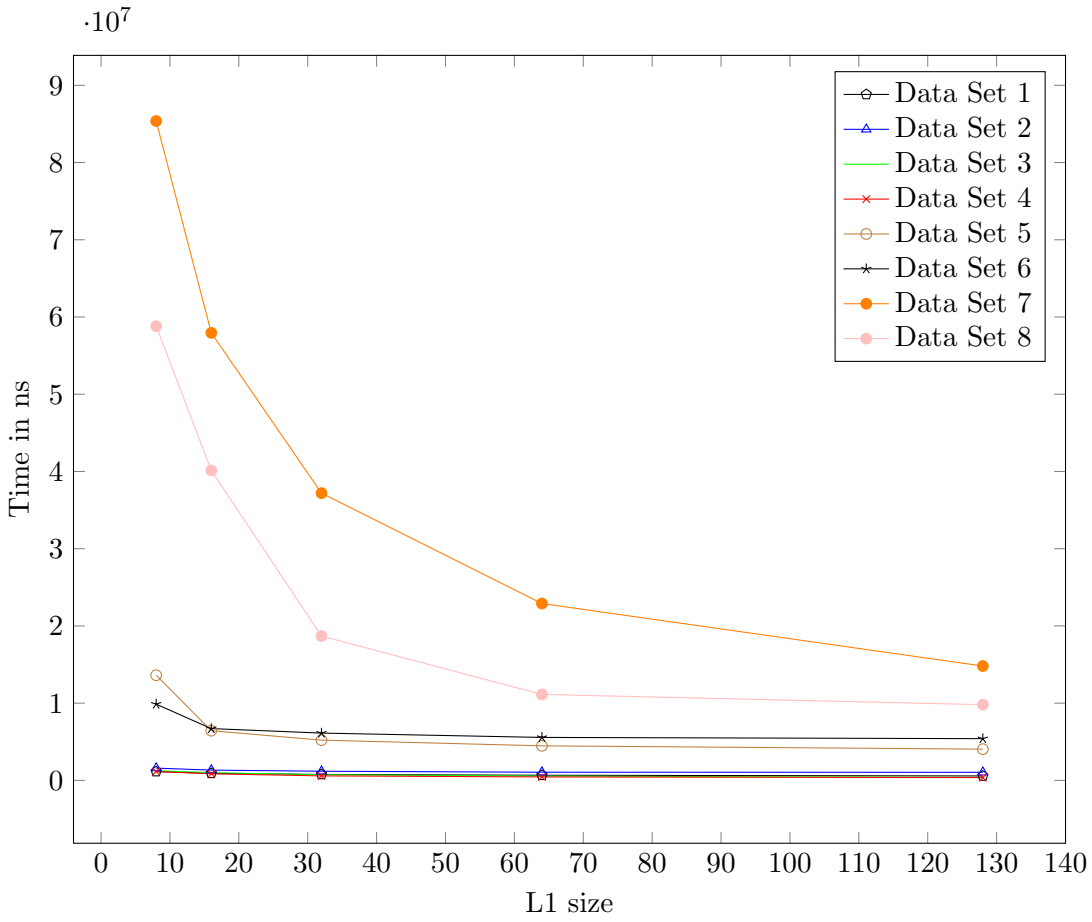
§2.1 Variation with Block size

- When block size increases, we can get the additional benefits of spatial locality as we tend to access nearby data more.
- Since the size of the cache is fixed, the number of sets decreases, which results in an increase in runtime.
- This is what can be seen in the graphs, which reach a minimum at an optimal block size, and then rise again.



§2.2 Variation with L1 Size

- Size of the cache increases, and so do the number of sets.
- This gives a direct benefit, as there are more indexes now, which can hold more data in the L1 cache.
- There is no tradeoff in this case, as in case 1, just a benefit.

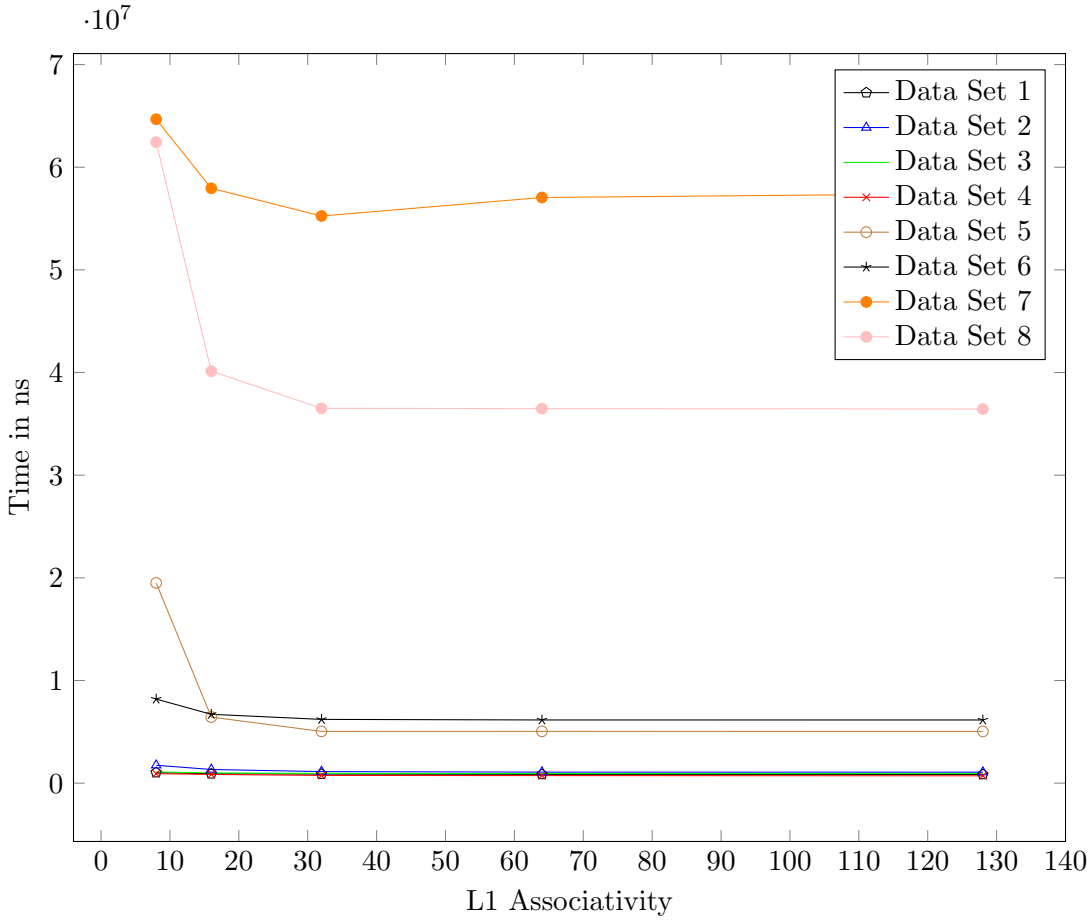


§2.3 Variation with L1 Associativity

- It is easy to see that increasing associativity would increase performance as there are still the same number of blocks that can be stored in the cache, but we can reduce conflict misses.
- The compulsory and capacity misses do not change in number.
- The graph is concave up as the benefit of increasing the associativity diminishes as the associativity goes up. This is because there are seldom any conflict misses between multiple elements which map to the same cache line in a temporally sequential fashion. So the majority of conflict misses are eliminated in the first increase in associativity.

L1 Size	Trace1	Trace2	Trace3	Trace4	Trace5	Trace6	Trace7	Trace8
8	1175140	1346700	1186940	946080	6439860	6718300	58386660	40142520
16	970740	1198300	1012340	868680	6436260	6707100	57589860	40135320
32	926940	1250500	1008140	855080	6433860	6706500	57643460	40135320
64	915740	1330500	998540	848880	6433860	6706500	57948460	40135320
128	914140	1346700	997340	844080	6433860	6706500	58457060	40135320

Table 1 L2 size vs time variation data

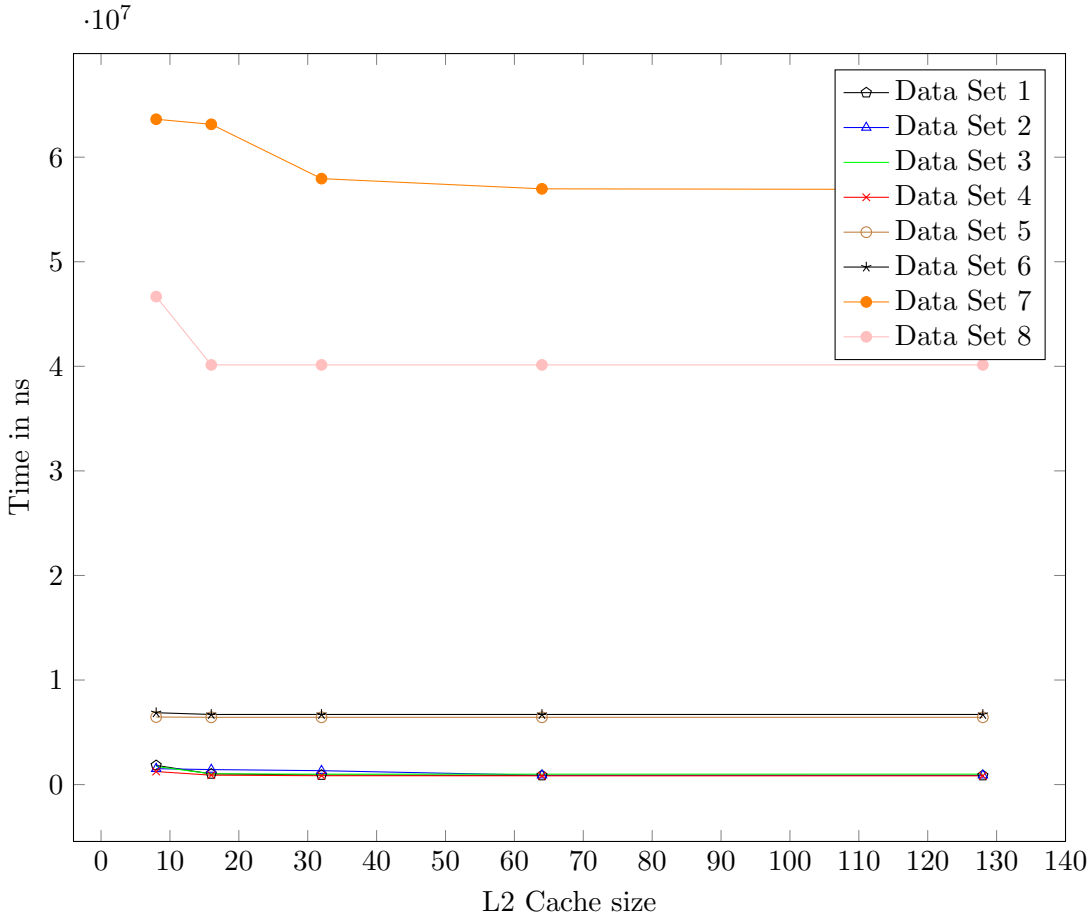


§2.4 Variation with L2 Cache Size

- It should seem that the observations should be exactly the same as in the variation with L1 size.
- This however is not exactly correct, as the L2 cache is not accessed as much as the L1 cache. The number of times the L2 cache is accessed is quite low, so the reduction in time is not as prominent.

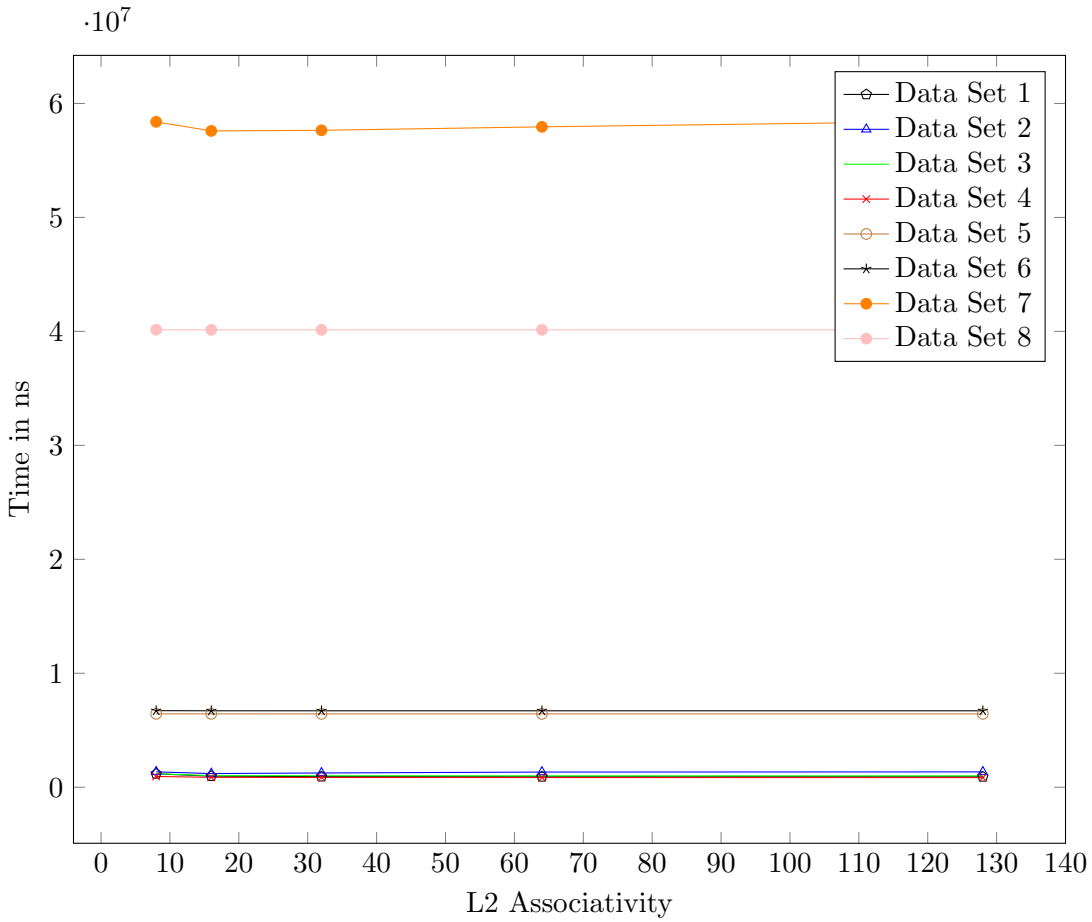
L2 Associativity	Trace1	Trace2	Trace3	Trace4	Trace5	Trace6	Trace7	Trace8
8	1175140	1346700	1186940	946080	6439860	6718300	58386660	40142520
16	970740	1198300	1012340	868680	6436260	6707100	57589860	40135320
32	926940	1250500	1008140	855080	6433860	6706500	57643460	40135320
64	915740	1330500	998540	848880	6433860	6706500	57948460	40135320
128	914140	1346700	997340	844080	6433860	6706500	58457060	40135320

Table 2 L2 Associativity vs time variation data



§2.5 Variation with L2 Associativity

- We can see that the effect of L2 associativity on the performance is negligible.
- To see why this is true, we should note that the fraction of instructions which access L2 is very low, which means that any improvement in the associativity of the cache is going to be low. This is the same thing which was earlier visible through the plots of the variation of time with L2 size.
- Even the largest of files show no significant change in performance on increasing cache associativity of L2.



§3 Conclusion

We both worked roughly equally in this assignment, and so the token distribution is as follows :
Suren : 50 %
Pravar Kataria : 50 %

§4 Acknowledgements

We have used the style file from here[1] to produce this document.

[1] `https://github.com/vEnhance/dotfiles/blob/main/texmf/tex/latex/evan/evan.sty`