Indian Institute of Technology Delhi

COL216 - Computer Architecture

Assignment 3

Simulating a two level Cache system

Garv Nagori, Aryan Sharma
2021CS10549, 2021CS10553

Contents

§1 Introduction

This assignment aims to simulate the working of a two-level cache system, the two-level caches being L_1 and L_2. We aim at ensuring inclusivity in our cache system, i.e. $L_1 \subseteq L_2$. This ensures that each block that is present in L_1 is also contained in L_2. Further we analyze the run-time of this cache system over various trace files.

§2 Inclusivity

We have ensured inclusivity i.e. if a block $B \in L_1 => B \in L_2$ in our program. We have used the following protocol to ensure that inclusivity is maintained throughout the execution time :
The Inclusivity Protocol : For a Block B, and caches L_1, L_2, The protocol is described as a pseudo-code:
Here, I is the instruction pair from the trace file, $I.first$ = operation type, $I.second$ = address. This program just works on one instruction pair, we run this program for each entry of the trace file to obtain the answer.

```
Algorithm 2.1
INCLUSIVITY ($I$,L$_1$, $L_2$) $\overset{df}{=}$
  $I.first = read$ $\rightarrow$
      I.first $\in L_1$ $\rightarrow$ $getBlock(I.second$,L$_1)$
      else $\rightarrow$
          I.first $\in L_2$ $\rightarrow$ $addBlock(I.second$,L$_1)$
          else $\rightarrow$
              $addBlock(I.second$,L$_1)$
              $addBlock(I.second$,L$_2)$
  else $\rightarrow$
      I.first $\in L_1$ $\rightarrow$ $writeBlock(I.second$,L$_1)$
      else $\rightarrow$
          I.first $\in L_2$ $\rightarrow$
              $addBlock(I.second$,L$_1)$
              $writeBlock(I.second$,L$_1$
          else $\rightarrow$
              $addBlock(I.second$,L$_1)$
              $addBlock(I.second$,L$_2)$
              $writeBlock(I.second$,L$_1)$
```

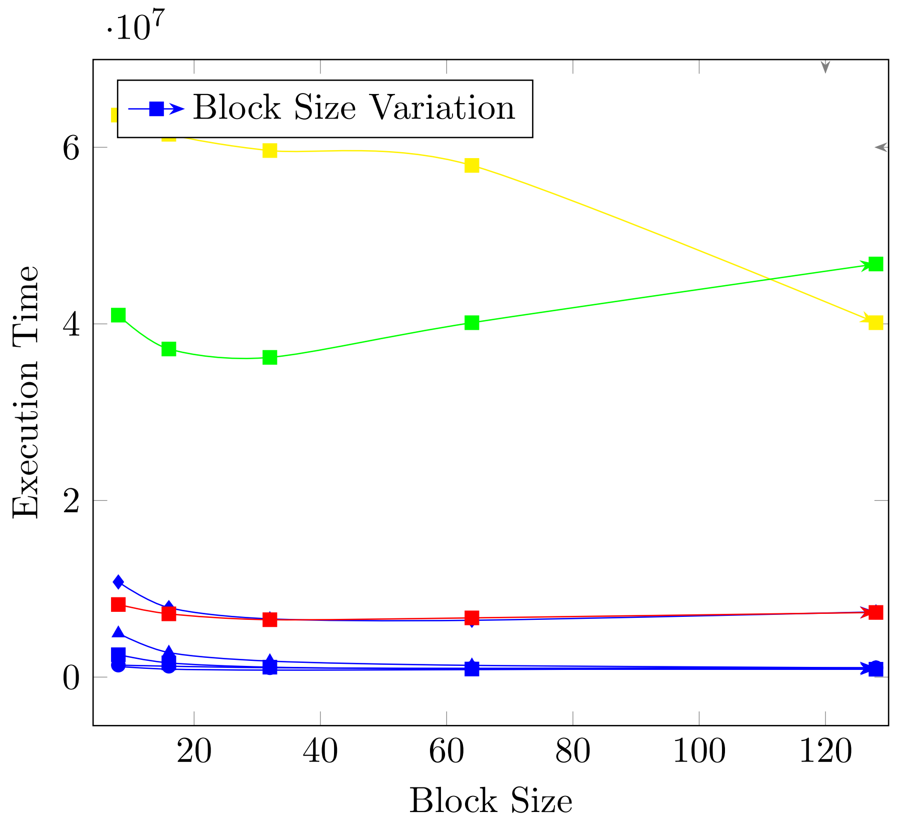
This protocol ensures inclusivity because of the way ***addBlock*** has been implemented. The function ***getBlock*** and ***writeBlock*** do not perform anything other than fetching values and writing values(including making the Block dirty). ***addBlock*** has ensured the following :

(a) If there is empty space in the cache then the Block is simply added.

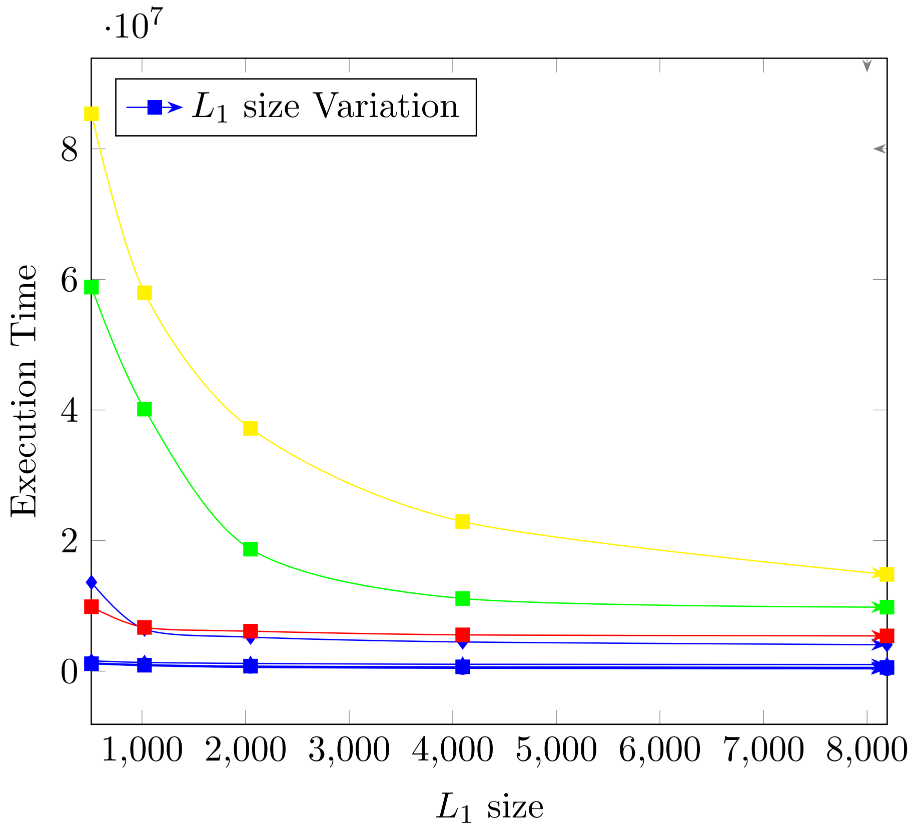
(b) In case there is no empty space in the cache corresponding to the required set, eviction takes place according to LRU policy. If the eviction takes place in L_2, then, the evicted block is invalidated in L_1 if it is not dirty else the the data is written into L_2 and the invalidated. If the eviction takes place in L_1, then, if the evicted block is dirty then it is simply written back to L_2 else just simply evicted.

§3 Plots

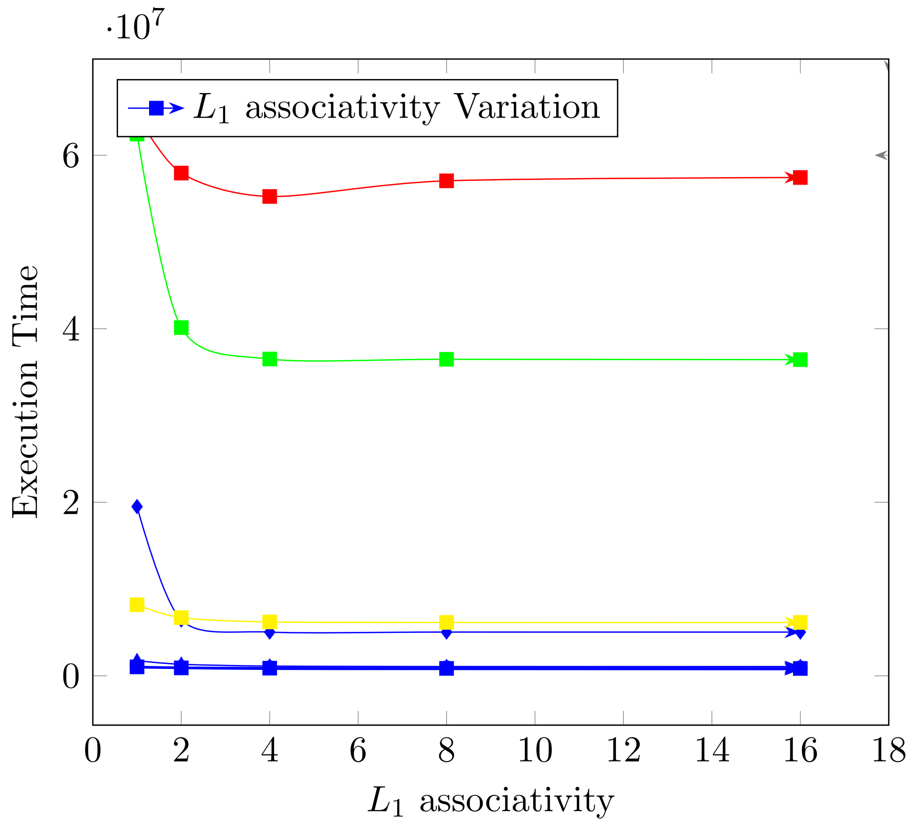
§3.1 Variation with Block Size



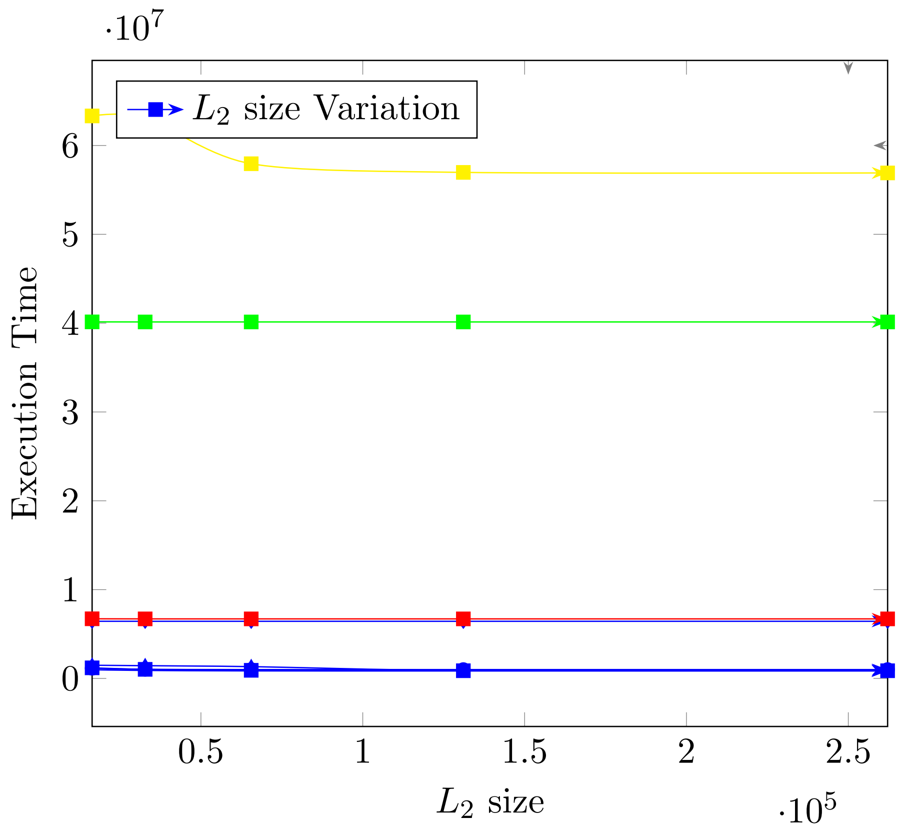
§3.2 Variation with L_1 size



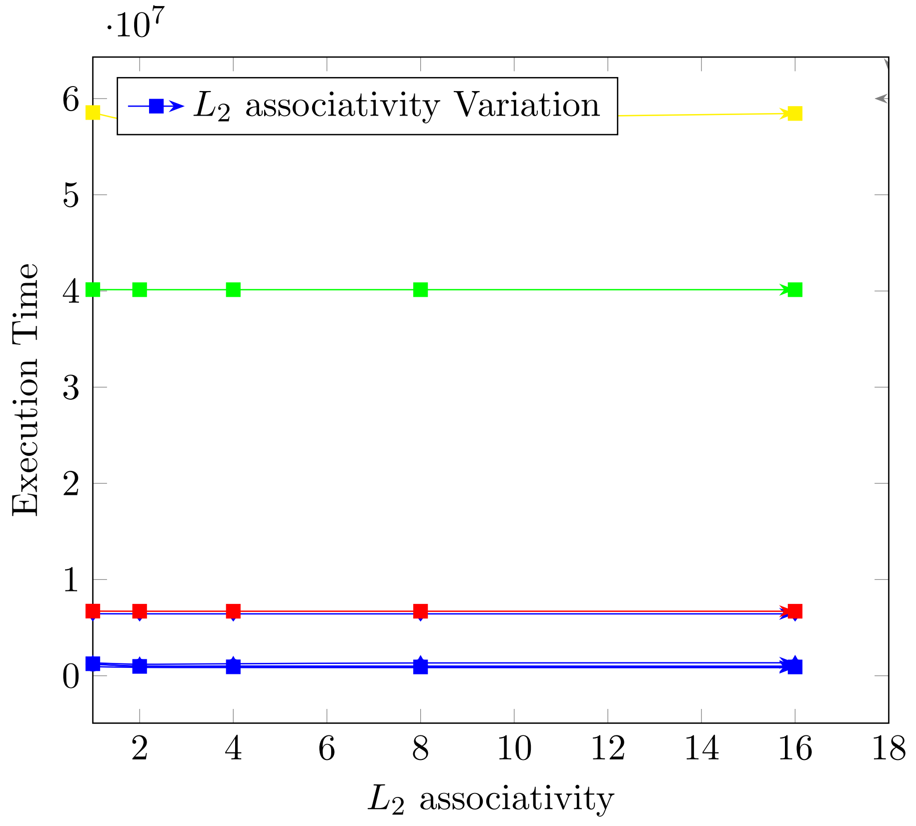
§3.3 Variation of L_1 associativity



§3.4 Variation of L_2 size



§3.5 Variation of L_2 associativity



§4 Plot Explanations

We analyze the graphs drawn above to intuitively understand the reason behind these trends

§4.1 Variation with Block Size

The competing factors in this case are :-

(a) **Number Of Sets** : As the number of sets increase, the use of temporal locality becomes greater and we can get between results. Sets in a way signify Temporal regions which are not spatially linked, each set in itself represents spatially linked region. Temporally more frequent calls now get answered with lesser time delay.
(b) **Block Size** :

As the block size increases, the use of spacial locality becomes greater and we can get better results. Spatially more frequent calls now get answered with lesser time delay.
Thus in total these competing factors lead to first a decrease followed by an increase as an overall trend.

§4.2 Variation with L_1 cache size

In our regime of modelling where we have assumed the access time to caches is the same irrespective of the size of the cache. Under the above said assumption the access time must reduce with increasing the size of cache block, which is also reflected by the plots. The reason for this being that now we have greater number of sets thus more space to accomadate blocks at the same cost.

Although in the case of actual caches, the time to access values inside the cache would vary as the size of the cache increases. Thus these plots do not reflect the true nature of access time variation with the size of the cache.

§4.3 Variation with L_1 associativity

The variation in this case is very similar to that in the case of varying block size, the parameters under considerations here are:

(a) **Number of Lines** :

As the associativity increases we can handle more addresses with the same set numbers, thus decreasing the miss rate. Thus decreasing the access time.
(b) **Number of Sets** As we increase associativity of the cache with fixed size, we also in turn reduce the number of sets available to us, thus many different sets get mapped to the same set number, this leads to increase in the access time.

In practice an issue with increasing the number of sets is that the search time through caches also increases but in our case we have assumed that access time to be constant thus this does not get reflected in our plots.

§4.4 Variation with L_2 cache size

The variation in L_2 cache size has the same effect as that of increasing L_1 cache size. But the graphs do not show as much prominent reduction in time with increase in size of the cache because : (a) **Infrequent L_2 accesses** :

L_2 is accessed much less number of time than L_1
(b) **Large Size of L_2** :

As the L_2 caches are usually of very large sizes (16384 bytes) itself which is quiet sufficient to handle almost all requests with great speed. Thus increasing size of caches does not effect access time effectively.s

§4.5 Variation with L_2 associativity

Since $L2$ is not accessed that many times, not much effect is observed when size of $L2$ is changed. (a) **Noticable effect**: In one case, increasing $L2$ size decreases execution time since more data can be stored into the $L2$ cache. However, when increasing even further the execution time remains almost constant or decreasing ever so slightly such that almost no effect is observed. (b) **Benfits vs cost**: The L2 hits increase but since it only changes by 1%, there is no benefit to increasing $L2$ size. It only increases the cost without much benefit.

§5 Token Distribution

Regarding the distribution of tokens, both of us have worked equally on the assignment. Even in quantitative measures, we have contributed almost equally to the point distribution for the different parts and checkpoints. So, we have decided to distribute 20 tokens uniformly (10 each). The list of the aspects each member has worked on is not strict since most of the time, we were working collaboratively, but if we still had to distribute the elements worked on, we would do according to the significant contributions as follows:

Name	**Tokens**	**Aspect worked on**
Aryan Sharma	10	L2 replace, L2 write miss, I/O handling, Debugging, LaTeX report
Garv Nagori	10	L2 read, L1 read and write, L1 replace, Debugging

Token Distribution

§6 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