CSN PROJECT
IMPLEMENTING VARIOUS CACHING STRATEGIES

October 31, 2019

Group members
Ritika Garg (CSE) 18114069
Rutuja Bhagwan Kendre (CSE) 18114070
Niharika Agrawal (CSE) 18114051
Saksham Sharma (ECE) 18116067
Sanyam Jain (ECE) 18116071
Sadique Wasim (ECE) 18116066
Vanshika Singh (ECE) 18116081

1 Description of the topic

Our topic basically plans to give us the real time application of the theoretical knowledge of caching techniques, lru implementation, temporal and spatial localities along with studing two new caching techniques which are annex and victime cache(relavent nowadays). we used cpp code to evaluate hit rates and evaluation time. we referred various research papers fo refernce.

2 Center of interest

We found the topic interseting as we collaborated our theoretical knowledge and previous coding experience to achieve practical comparisons. Also we came across two interesting caching strategies and tried implementing it as well.

3 References and Related work

3.1 Research papers referred

1. A Buffer Cache Management Scheme Exploiting Both Temporal and Spatial Localities by XIAONING DI
2. A Data Cache with Multiple Caching Strategies Tuned to Different Types of Locality by Antonio Gonzále
3. Annex cache: a cache assist to implement selective caching by L.K. Johna , T. Lia , A. Subramanianb
4. Design and performance evaluation of a cache assist to implement selective caching by Norman Jouppi
5.

3.2 Victim cache

Victim caching is a hardware technique to improve performance of caches proposed by Norman Jouppi. As mentioned in his paper:
As hardware architecture and technology advanced, processor performance and frequency grew much faster than memory cycle times, leading to a large gap in performance. The problem of increasing memory latency, relative to processor speed, has been dealt with by adding high speed cache memory.
Direct-mapped caches have faster access time than set-associative caches. However, for a direct-mapped cache if multiple cache blocks in the memory map to same cache-line they end up evicting each other when anyone of them is accessed. This is known as the cache conflict problem. This problem is resolved by increasing the associativity of the cache. But there is a limit to which associativity can be increased owing to the complexity in its implementation. Thus, for solving the cache conflict problem for a cache with limited associativity victim cache is employed. Small miss caches of 2 to 5 entries are shown to be very effective in removing mapping conflict misses in first-level direct-mapped caches. Victim caching is an improvement to miss caching.

3.3 Annex cache

Annex cache is a further modification of victim cache.In victim cache, the cache assist is meant to hold the items that would otherwise be displaced from the cache. But if the conflicting entries in the main cache and the assist cache are referenced alternately, each access would result in a swap between the main and the assist cache. The annex cache eliminates such continuous swapping. One of the entries will be accessed from the main cache and the other entry will be directly accessed from the assist cache thus improving performance(hit rate).

3.4 Algorithm

1. In the first cycle, the cache reads the tags and data of the relevant block in the primary cache and in the annex cache. The data from the primary cache is latched into the output buffer becoming available at the end of the first cycle.
2. The tag from the main cache and the K tags from the annex (annex is K-way

set associative) are compared against the appropriate bits of the address. 3. If the tag from the main cache matches the address, then a main cache hit occurs. The priority of the main cache entry is updated to be high. The search in the annex cache is immediately cancelled so that it can be used again in the next cycle. If the tag from the main cache does not match, the annex search has to be continued.
4. If there was no main cache hit and if one of the tags from the annex match, an annex-cache hit occurs. The data is read directly from the annex if the conliicting main cache entry has a high priority or if the annex cache entry has a low priority. Otherwise, the data is loaded into the main cache or swapped with the conflicting main cache entry.
5. If all tag comparators mismatch, the data is in the main memory. When the data returns from the main memory, it is placed in the main cache if the corresponding main cache block has no valid data;otherwise it is placed in the annex cache.

4 Description of methods used

We implemented our project from our previous knowledge of cpp and java. we took the data input using an array and varied the parameters on the basis of which we were going to compare the cache performance(data size and data count). we varied the value of 'k' in the code to achieve direct, associative and set associative mapping.In the other code wherein we tested the spatial and temporal localities we took the value of k as user input along with black and cache size. we changed the cache size to switch from temporal to spatial localities.Temporal locality refers to the reuse of specific data, and/or resources, within a relatively small time duration. Spatial locality refers to the use of data elements within relatively close storage locations.
we further devised the algorithm for annex and victim cache along with studying their importance.

5 Observations

5.1 Data repetition: Comparing different caching strategies

tt=time taken
hr=hit rate
X=no. of times data repeated

X k values	1	2	4	8
0	tt: 2.6047e-05s hr is 0.224609	tt:2.9218e-05s hr is 0.245117	tt:3.4586e-05s hr is 0.231445	tt:4.413e-05s ;hr is 0.224609
1	tt:2.4401e-05s hr is 0.277344	tt:2.771e-05s hr is 0.254883	tt:3.2306e-05s hr is 0.277344	tt:4.5048e-05s hr is 0.262695
2	tt:2.223e-05s hr is 0.235352	tt:2.5306e-05s hr is 0.230469	tt:2.9975e-05s hr is 0.234375	tt:3.928e-05s hr is 0.218750
3	tt:2.0083e-05s hr is 0.243164	tt: 2.1683e-05s hr is 0.250000	tt: 2.668e-05s hr is 0.250000	tt: 3.56e-05s hr is 0.242188
4	tt: 1.6043e-05s hr is 0.311523	tt: 1.9207e-05s hr is 0.249023	tt: 2.2509e-05s hr is 0.233398	tt: 3.1289e-05s hr is 0.216797
5	tt: 1.875e-05s hr is 0.216797	tt: 2.0973e-05s hr is 0.166992	tt: 2.6039e-05s hr is 0.171875	tt: 3.1907e-05s hr is 0.187500
6	tt: 1.1781e-05s hr is 0.371094	tt: 1.5075e-05s hr is 0.432617	tt: 1.5126e-05s hr is 0.560547	tt: 2.5262e-05s hr is 0.557617
7	tt: 1.1047e-05s hr is 0.498047	tt: 1.0943e-05s hr is 0.623047	tt: 1.5081e-05s hr is 0.498047	tt: 9.169e-06s hr is 0.996094
8	tt: 1.2545e-05s hr is 0.498047	tt: 6.338e-06s hr is 0.996094	tt: 7.983e-06s hr is 0.996094	tt: 1.0909e-06s hr is 0.996094
9	tt: 5.498e-06s hr is 0.998047	tt: 5.802e-06s hr is 0.999023	tt: 7.556e-06s hr is 0.999023	tt: 8.301e-06s hr is 0.998047

5.2 Data size: Comparing different caching strategies

X=data size

X k values	1	2	4	8
16	tt:5.65e-07s hr is 0.187500	tt:5.92e-07s hr is 0.375000	tt:7.24e-07s hr is 0.375000	tt:8.68e-07s ;hr is 0.250000
32	tt:8.91e-07s hr is 0.250000	tt:1.063e-06s hr is 0.312500	tt:1.38e-06s hr is 0.187500	tt:1.599e-06s hr is 0.218750
64	tt:1.707e-06s hr is 0.218750	tt:2e-06 s hr is 0.187500	tt:2.311e-06s hr is 0.187500	tt:2.976e-06s hr is 0.156250
128	tt:3.214e-06s hr is 0.218750	tt:3.849e-06s hr is 0.257812	tt:4.527e-06s hr is 0.257812	tt:5.682e-06s hr is 0.226562
256	tt:6.356e-06s hr is 0.234375	tt:7.078e-06s hr is 0.234375	tt: 8.553e-06s hr is 0.234375	tt:1.0867e-05s hr is 0.214844
512	tt:1.2254e-05s hr is 0.220703	tt:1.3877e-05s hr is 0.240234	tt:1.6118e-05s hr is 0.226562	tt:2.1236e-05s hr is 0.226562
1024	tt:2.4963e-05s hr is 0.225586	tt:2.7789e-05s hr is 0.247070	tt:3.2929e-05s hr is 0.229492	tt:4.1911e-05s hr is 0.227539

5.3 Temporal and Spatial localities

DIRECT MAPPING

cache size(KB)	temporal locality	spatial locality
2	tt:3.639e-07s hr is 0.75	tt:9.373e-08 s hr is 0.99
16	tt:2.47712e-06 s hr is 0.75	tt:6.525e-08 s hr is 0.99
32	tt:4.615e-06 s hr is 0.75	tt:9.849e-08 s hr is 0.99

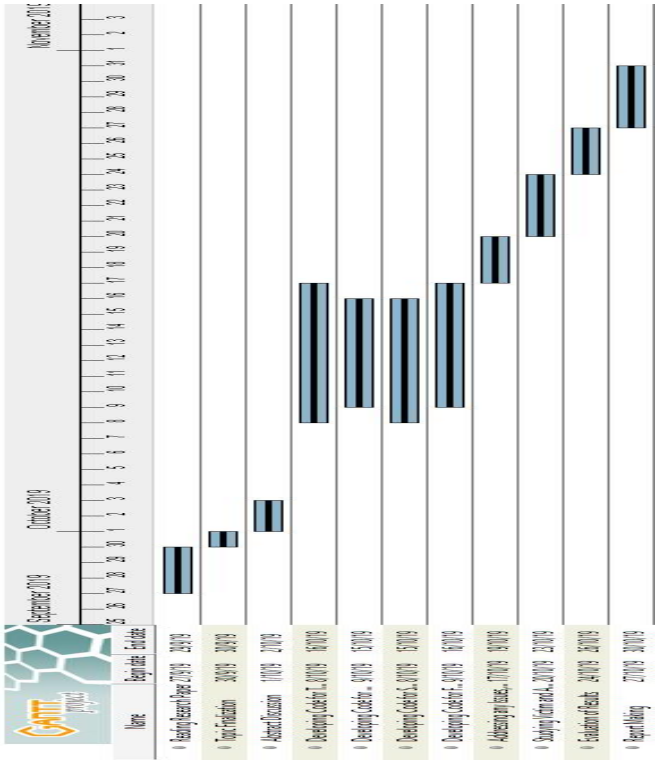
SET ASSOCIATIVE MAPPING(4 way associative)

cache size(KB)	temporal locality	spatial locality
2	tt:3.021e-07 s hr is 0.75576	tt:9.982e-08 s hr is 0.9923
16	tt:2.610e-07 s hr is 0.79608	tt:9.699e-06 s hr is 0.9936
32	tt:2.227e-07 s hr is 0.8382	tt:9.949e-06 s hr is 0.9949

FULLY ASSOCIATIVE MAPPING

cache size(KB)	temporal locality	spatial locality
2	tt:3.04e-07 s hr is 0.99974	tt:9.982e-07 s hr is 0.9999
16	tt:2.923e-07 s hr is 0.99974	tt:3.5125e-08 s hr is 0.9999
32	tt:2.877e-07 s hr is 0.99974	tt:5.532e-08 s hr is 0.9999

6 GANTT chart(timeline)



7 Result and conclusion

Table 1 and Table 2:
Looking and examining the table we found that the hit rate increases as the data repetition is increased i.e. when X is 9(maximum value) and performance is good when data size is small.
Table 3:
In every case implementing spatial locality is found to be more efficient as compared to temporal locality
Hit rate - Fully>set associative >direct
Time taken -Direct >set associative >fully