HOMEWORK 4

CSCE 614-600

ADIL HAMID MALLA

425008306

Implementation:

Implemented the SRRIP algorithm as per the paper described for the reading in the homework. Implementation included implementing the SRRIP in the "rrip_repl.h" file of zsim simulator. Also the initialization of this in the init.cpp.

Moreover, the implementation also needed to make config files for the LFU.

Please check the submitted code for all the implementations.

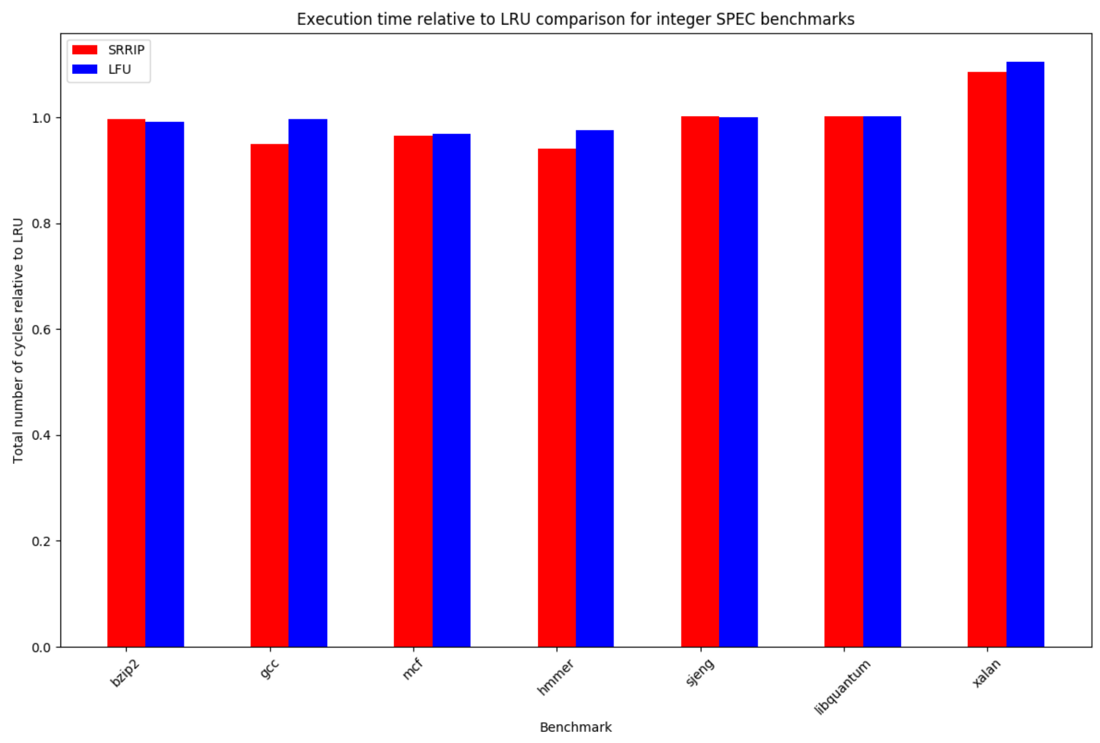
Configuration Comparison Between SPEC and PARSEC:

Item	**SPEC**	**PARSEC**
Number of cores	1	8
L1 instruction cache size	4 KB	4 KB (*8)
L1 instruction cache associativity	4-way	4-way
L1 data cache size	4 KB	4 KB (*8)
L1 data cache associativity	8-way	8-way
L1 latency	1 cycle (for both)	1 cycle (for both)
L2 cache size	32 KB	32 KB (*8)
L2 cache associativity	8-way	8-way
L2 latency	10 cycles	10 cycles
L3 cache size	256 KB	1 MB
L3 cache associativity	16-way	16-way
L3 cache latency	24 cycles	24 cycles

Simulation Results:

Ran the simulations on all the benchmarks for three LLC cache replacement policies i.e. LRU, LFU and SRRIP.

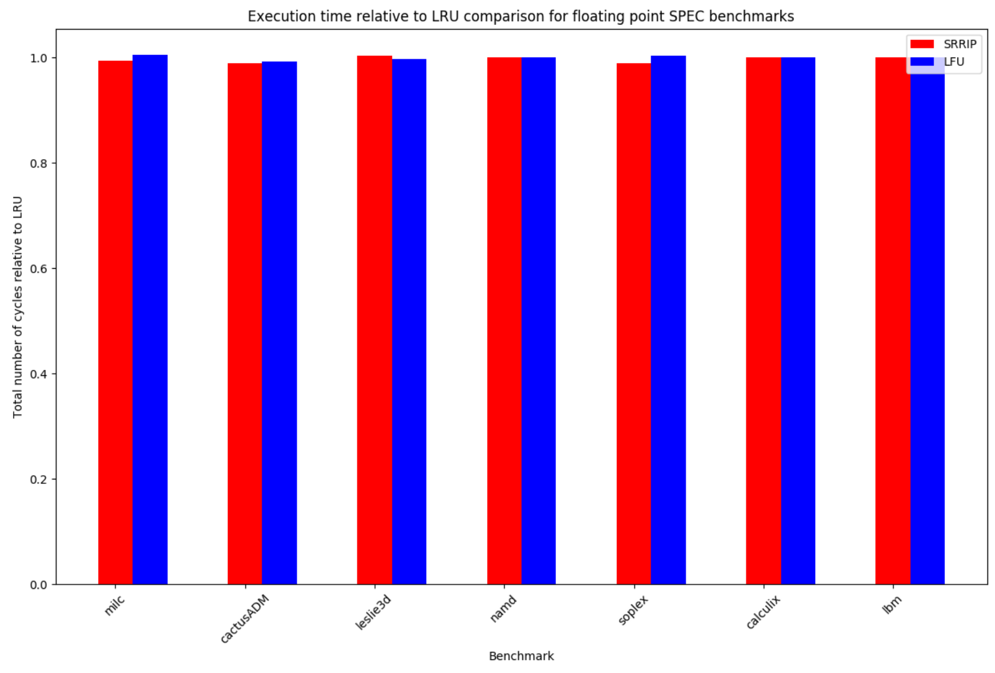
Performance Graph for the SPEC Integer Benchmarks:

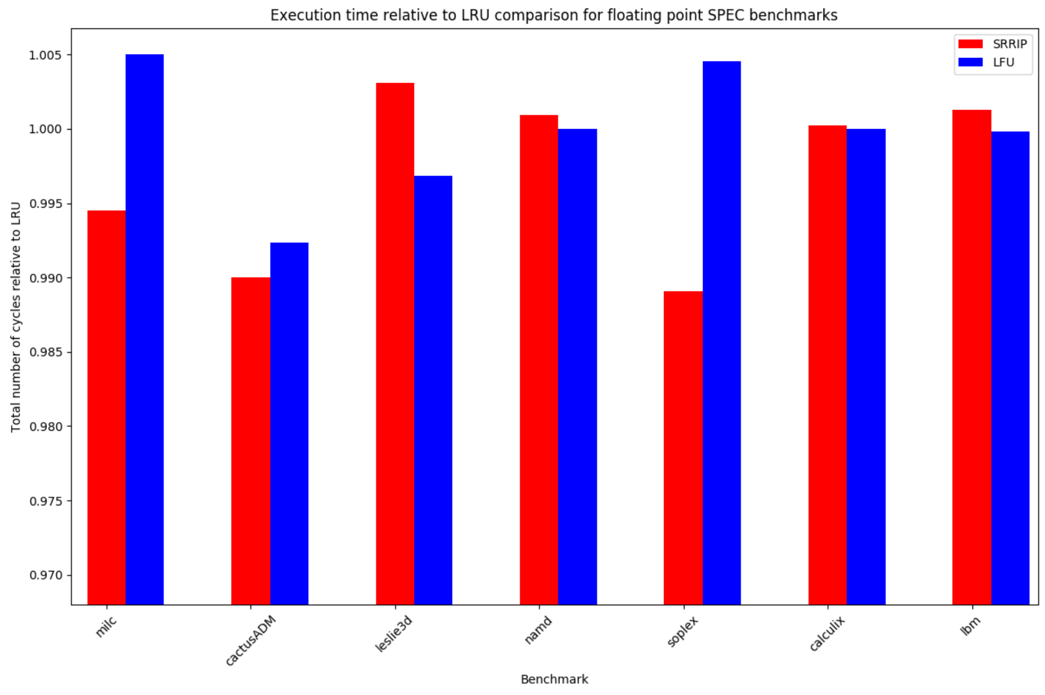


The above graph represents the execution time of the SRRIP and LFU in correspondence to the LRU. So all the times are normalized by the LRU time. As we can see that the SRRIP algorithm is performing better in almost all the case in comparison to that of the LRU, hence making sure the far-distance reference blocks are penalized accordingly.

As we can see that the SRRIP is also performing better than LFU in almost every case except the bzip2 benchmark and sjeng. Since the locality of bzip and sjeng (AI) applications have many algorithms which do the random path taking algorithmms, that is why the execution time of the LFU is better here.

Performance Graph for the SPEC Float Benchmarks:

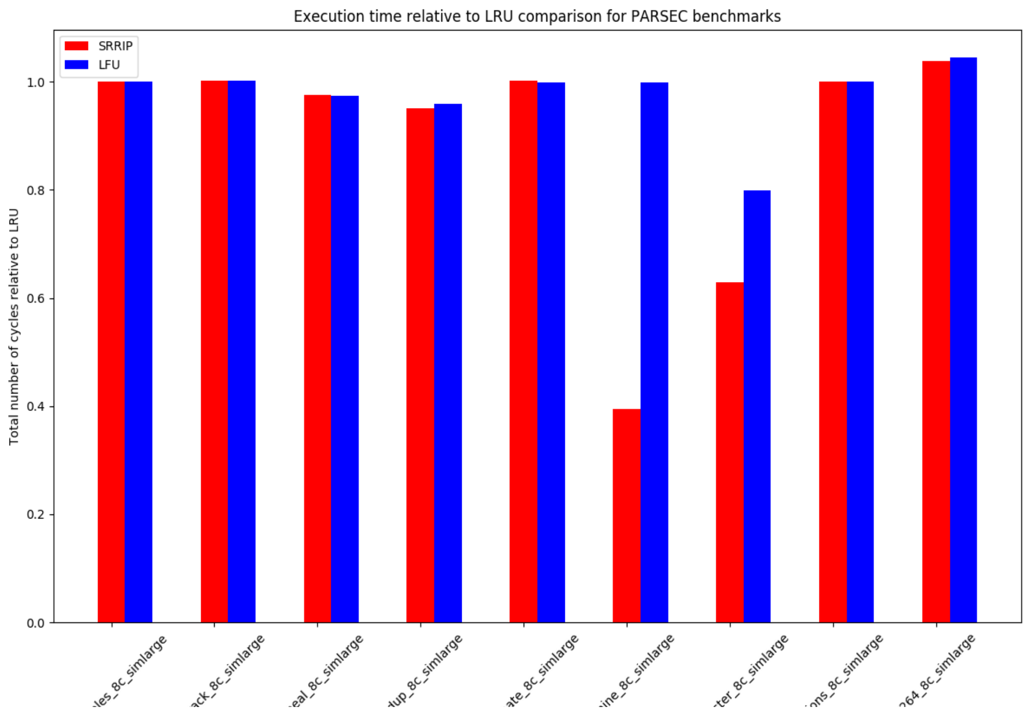




The above graphs show the original zoom level and higher zoom level of the execution time comparison of the SRRIP and LFU with that of the LRU. All the execution times are normalized with the LRU time.

As we can see the execution time of all the benchmarks is almost same as that of the LRU and SRRIP is doing exceptionally good in many cases like cactusADM and soplex, as these benchmarks are the scientific in nature thus the locality is high as compared to that of other benchmarks. Moreover, leslie3d is performing slightly worse than LRU as the leslie3d is related to fluid dynamics hence the differential equations solution makes the life bad for the caches.

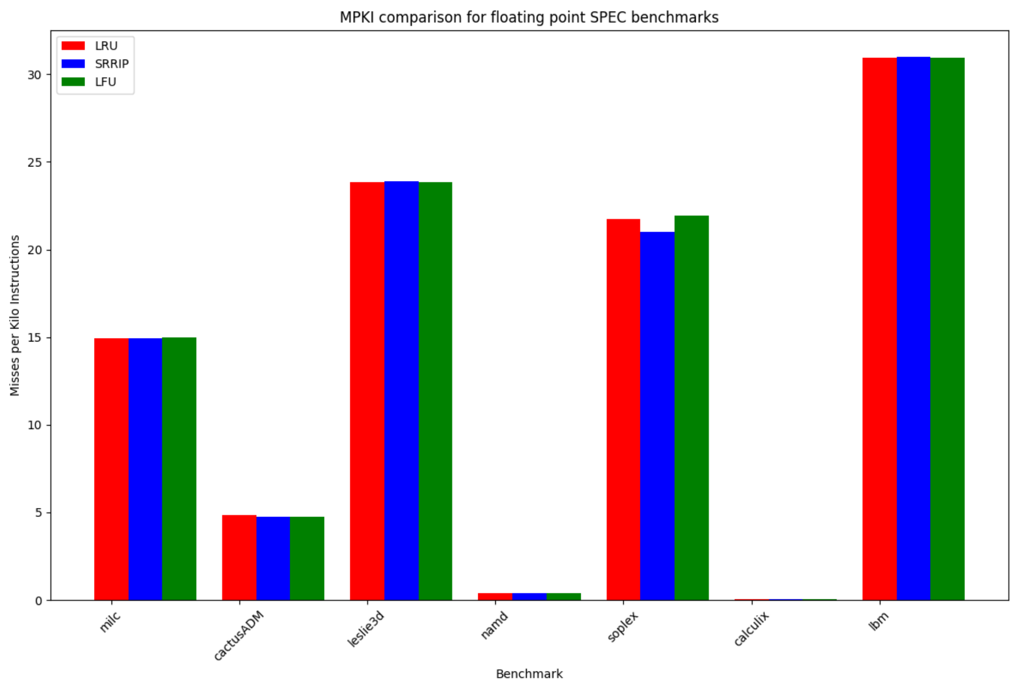
Performance Graph for the PARSEC Benchmarks:



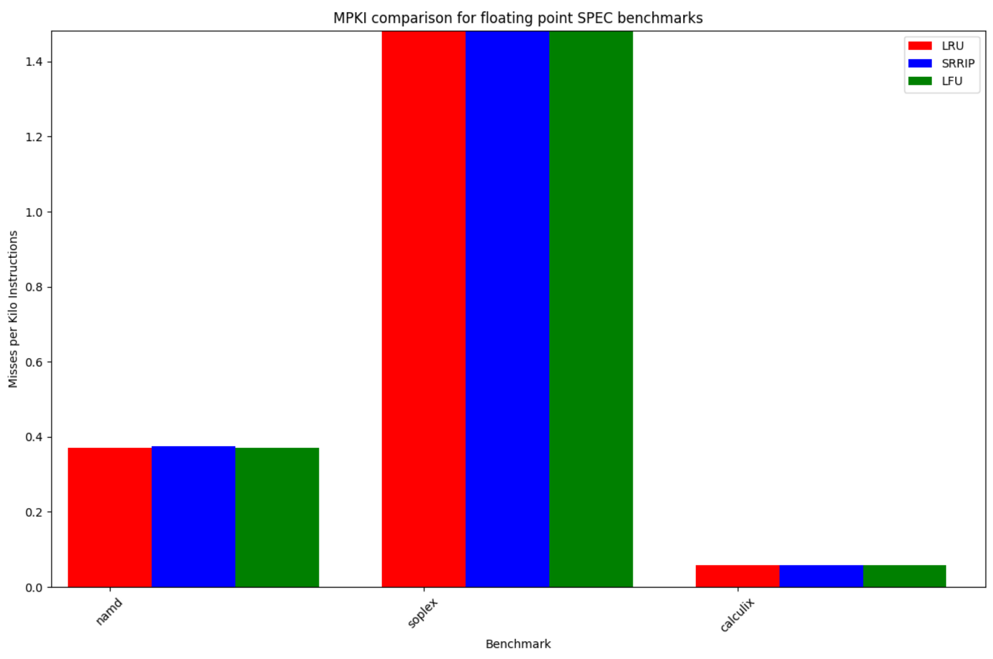
Comparison of the PARSEC benchmarks show that the execution time of almost all the benchmarks is same and here also the SRRIP is performing better as compared to other replacement policies. Also SRRIP is doing exceptionally well in the two of the benchmarks. i.e. freqmine and stream cluster. Both the benchmarks show low scan length references hence the better results and less misses.

MKPI Comparison for all the benchmarks:

MKPI Comparison of floating Point SPEC benchmarks:

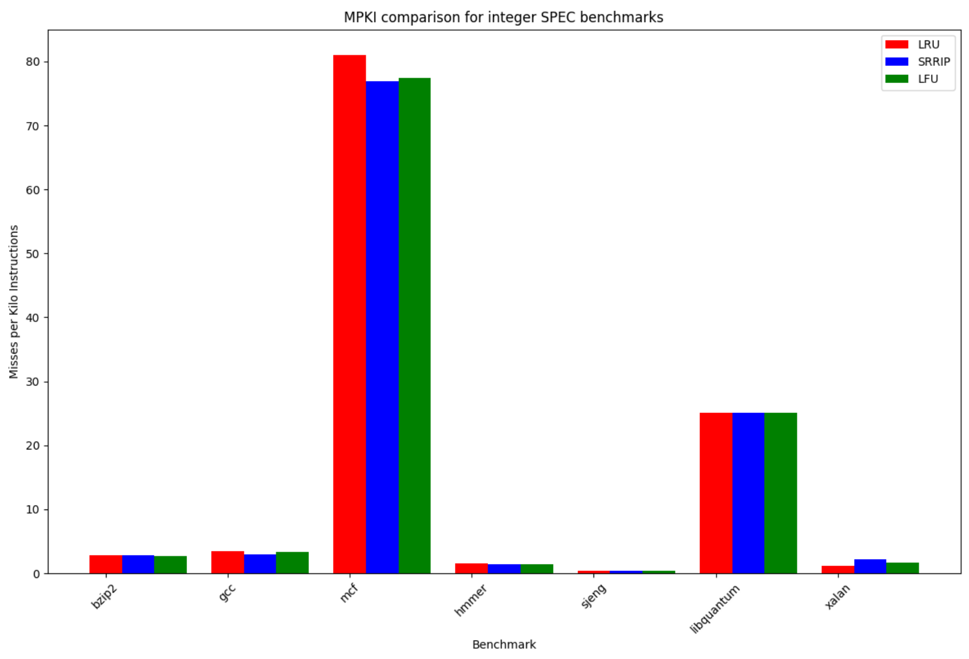


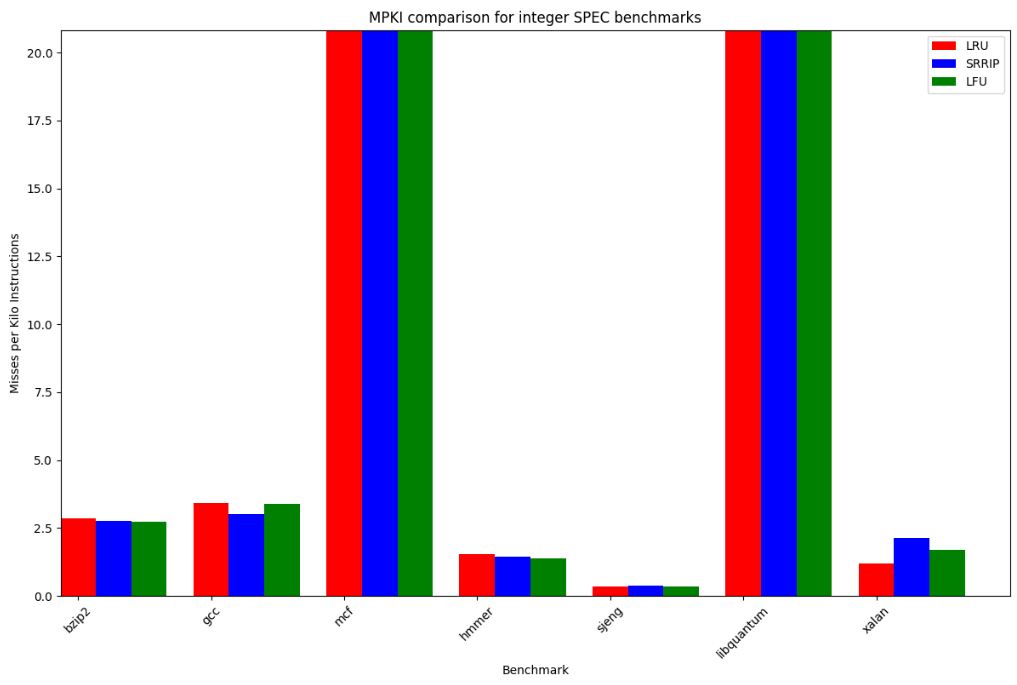
As we can see that the SRRIP is performing better than all other replacement policies. The misses in the namd and calculix is very less as these benchmarks represent the molecular and structural mechanics, hence the scan resistance make it miss very less. All other benchmarks have higher scan size thus leading to more misses.



Zoomed version of the SPEC Floating point Benchmarks.

MKPI Comparison of Integer SPEC benchmarks:

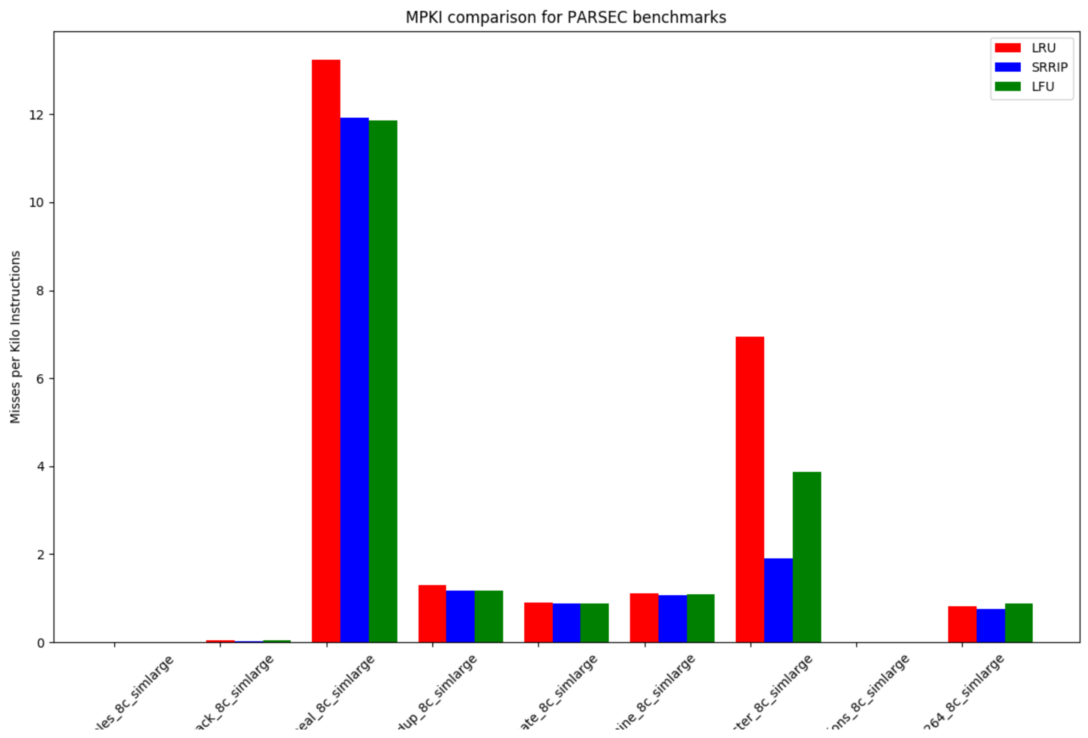


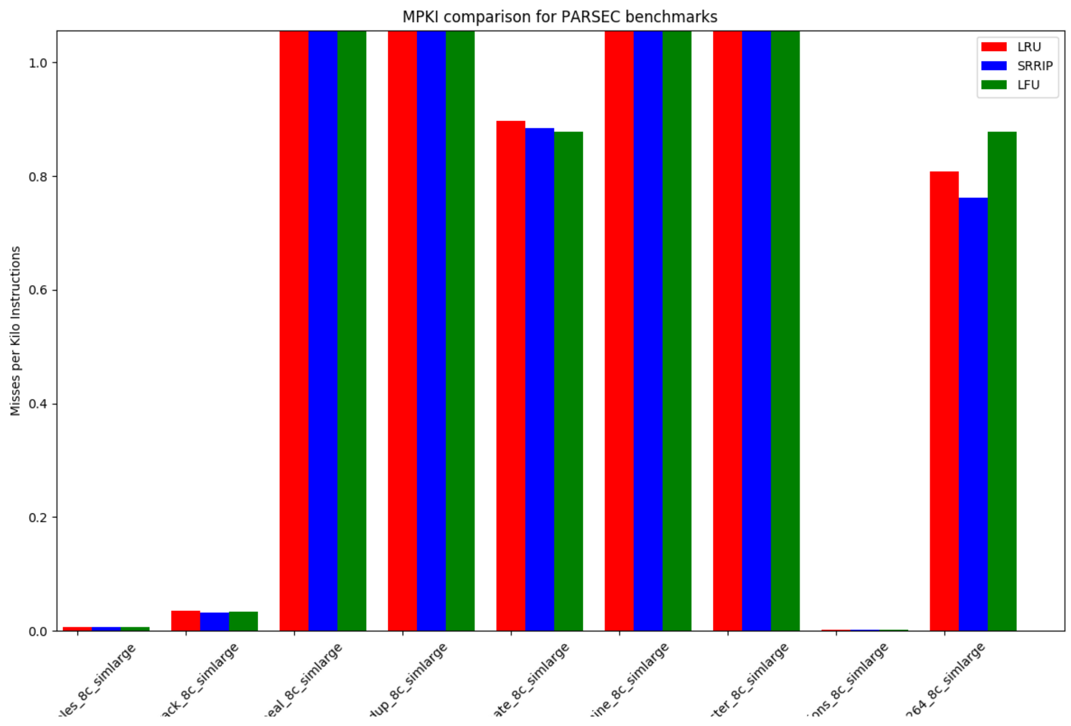


The above graphs represent the comparison of the three replacement policy comparison in terms of Misses per Kilo Instructions.

As we can see the number of the misses in the mcf and libquantum is more than that of the other benchmarks. This difference is mainly because of the nature of the benchmarks, i.e. Combinatorial Optimization and quantum physics algorithms. These algorithms have huge computational spaces and the scan path is very large hence leading to the increased number of misses.

MKPI Comparison of PARSEC benchmarks:





The above graphs show the comparison of the misses per kilo instructions in the three replacement policies i.e. LRU, LFU and SRRIP.

SRRIP is doing awesome in all the cases and also the benchmarks with less MKPI represents that the locality is high and thus the number of the misses is low.

Overall the SRRIP replacement policy has done better than that of LRU and LFU in general, in few cases whereas the structure of the code is somewhat different it has done little bad.