1. Introduction to Cache and its Replacement policies :

In computing, a cache is a hardware or software component that stores data so that future requests for that data can be served faster; the data stored in a cache might be the result of an earlier computation or a copy of data stored elsewhere. A cache hit occurs when the requested data can be found in a cache, while a cache miss occurs when it cannot. Cache hits are served by reading data from the cache, which is faster than recomputing a result or reading from a slower data store; thus, the more requests that can be served from the cache, the faster the system performs. Cache replacement policies are optimizing instructions, or algorithms, that a computer program or a hardware-maintained structure can utilize in order to manage a cache of information stored on the computer. Caching improves performance by keeping recent or often-used data items in a memory locations that are faster or computationally cheaper to access than normal memory stores.

Least recently used (LRU) :

This replacement policy discards the least recently used items first. This algorithm requires keeping track of what was used when, which is expensive if one wants to make sure the algorithm always discards the least recently used item. General implementations of this technique require keeping "age bits" for cache-lines and track the "Least Recently Used" cache-line based on age-bits. In such an implementation, every time a cache-line is used, the age of all other cache-lines changes.

Re-reference Interval Prediction (RRIP) :

The commonly used LRU replacement policy always predicts a near-immediate re-reference interval on cache hits and misses. Applications that exhibit a distant re-reference interval perform badly under LRU. Such applications usually have a working-set larger than the cache or have frequent bursts of references to non-temporal data (called scans). To improve the performance of such workloads, we have cache replacement using Re-reference Interval Prediction (RRIP). RRIP uses M-bits per cache block to store one of 2 M possible Re-reference Prediction Values (RRPV). The primary goal of RRIP is to prevent blocks with a distant re-reference interval from polluting the cache. RRIP always inserts new blocks with a long re-reference interval. We use an RRPV of 2^M –2 to represent a long re-reference interval. On a cache miss, the RRIP victim selection policy selects the victim block by finding the first block that is predicted to be re-referenced in the distant future. The algorithm we used updates RRPV register based on RRIP hit promotion policy. The RRIP-HP policy predicts that the block receiving a hit will be re-referenced in the near-immediate future and updates the RRPV of the associated block to zero. The goal of the HP policy is to prioritize replacement of blocks that do not receive cache hits over any cache block that receives a hit. SRRIP emulates optimal replacement by correctly predicting a near-immediate re-reference interval for the actively used cache blocks and a distant re-reference interval for the scan blocks. Hence SRRIP is scan-resistant and it only requires 2-bits per cache block.

Signature-based Hit Predictor(SHiP) :

Re-reference predictions can be made at a finer granularity by categorizing references into different groups by associating a signature with each cache reference. The cache references that have the same signature will have a similar re-reference interval. This can be achieved using simple, high performing, and low over head mechanisms to associate cache references with a unique signature and to predict the re-reference interval for that signature. To learn the re-reference pattern of a signature, SHiP requires two additional fields to be stored with each cache line: the signature itself and a single bit to track the outcome of the cache insertion. Signature-based Hit Predictor (ShiP) is used to predict whether the incoming cache line will receive a future hit. For a given signature, SHiP uses a Signature History Counter Table (SHCT) of saturating counters to learn the re-reference interval for that signature. SHiP updates the SHCT on cache hits and cache evictions. On a cache miss, SHiP indexes the SHCT with the corresponding signature to predict the re-reference interval of the incoming cache line. SHiP makes re-reference predictions only on cache insertions. A zero SHCT value provides a strong indication that future lines brought into the Last Level Cache(LLC) by that signature will not receive any cache hits. In other words, references associated with this signature always have a distant re-reference interval. On the other hand, a positive SHCT counter implies that the corresponding signature receives cache hits.

* The key difference between these three is LRU always predicts a near-immediate re-reference interval on cache hits and misses, whereas RRIP conservatively predicts that all cache insertions have an intermediate re-reference interval. If the newly-inserted cache line is referenced quickly, RRIP

updates the re-reference prediction to near-immediate; otherwise, RRIP downgrades the re-reference prediction to distant. In doing so, RRIP learns the re-reference interval for all inserted cache lines upon re-reference. SHiP on the other hand explicitly and dynamically predicts the re-reference interval based on the SHCT. On a cache miss, SHiP consults the SHCT with the signature to predict the re-reference interval of the incoming cache line.

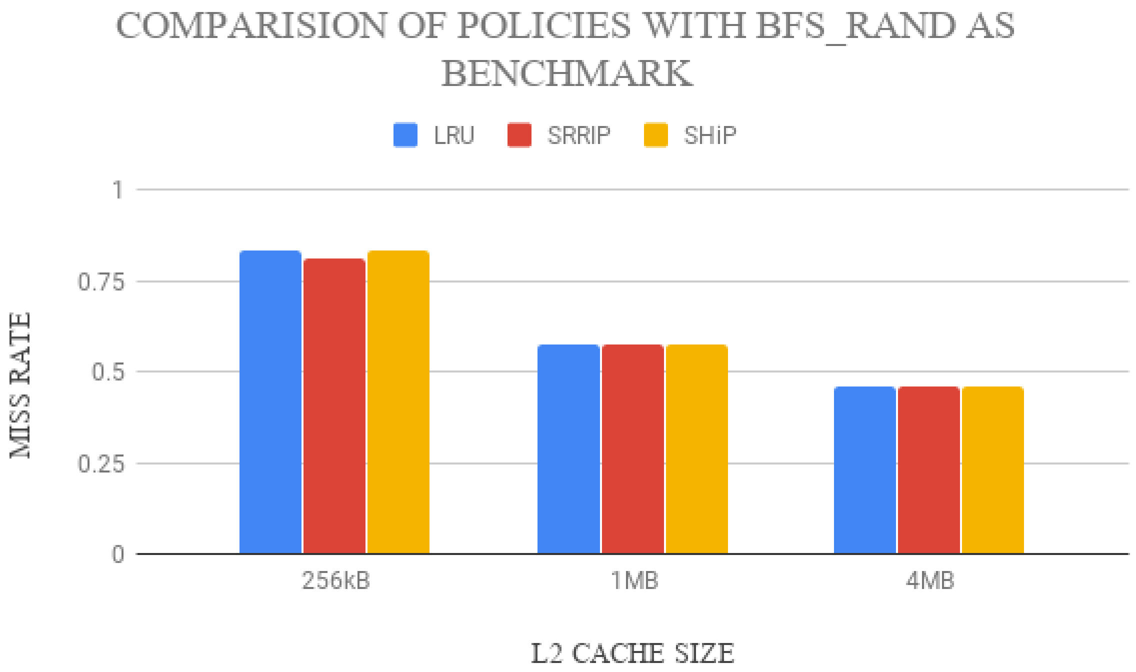
2. Result and Observations :

The PBBS Time for any replacement policy for same benchmark are same and are as follows: BFS rand – 0.0248, BFS rMat – 0.0212, MST rand – 0.113, MST rMat – 0.0471

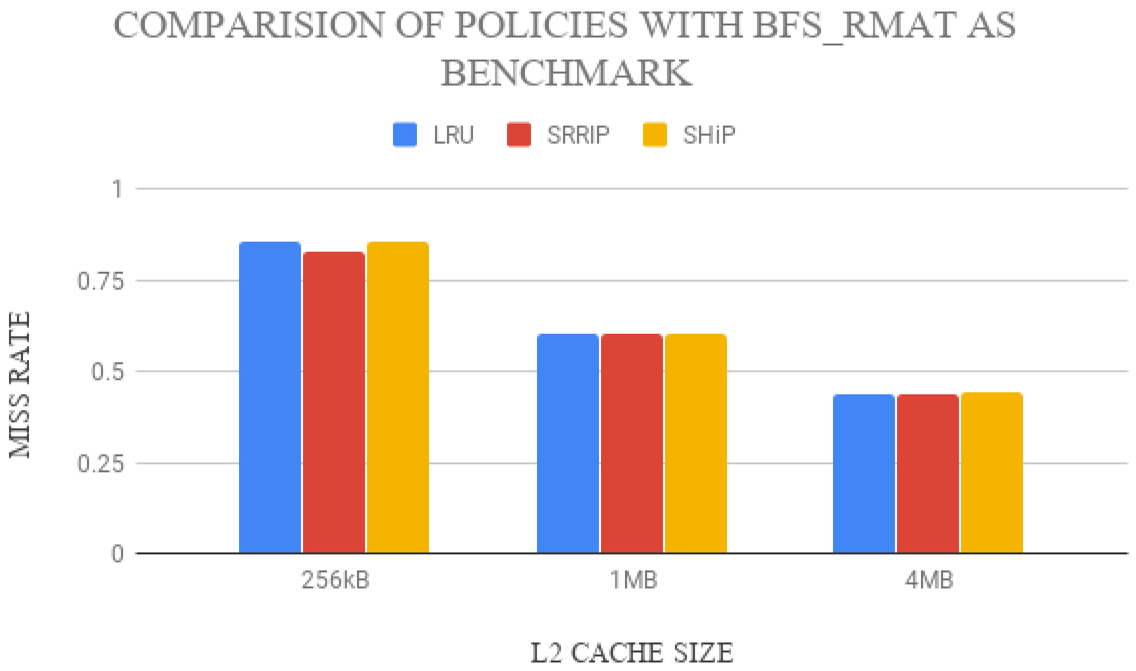
	LRU POLICY MISS RATE TABLE			
	BFS_OPT		MST_OPT	
L2 CACHE SIZE	RAND	RMAT	RAND	RMAT
256kB	0.832235	0.853288	0.80758	0.723259
1MB	0.574036	0.602824	0.624196	0.583532
4MB	0.457786	0.439413	0.526789	0.441119

	SRRIP POLICY MISS RATE TABLE			
	BFS_OPT		MST_OPT	
L2 CACHE SIZE	RAND	RMAT	RAND	RMAT
256kB	0.810148	0.830665	0.807533	0.721545
1MB	0.572217	0.600308	0.626930	0.588087
4MB	0.457624	0.43839	0.528451	0.434982

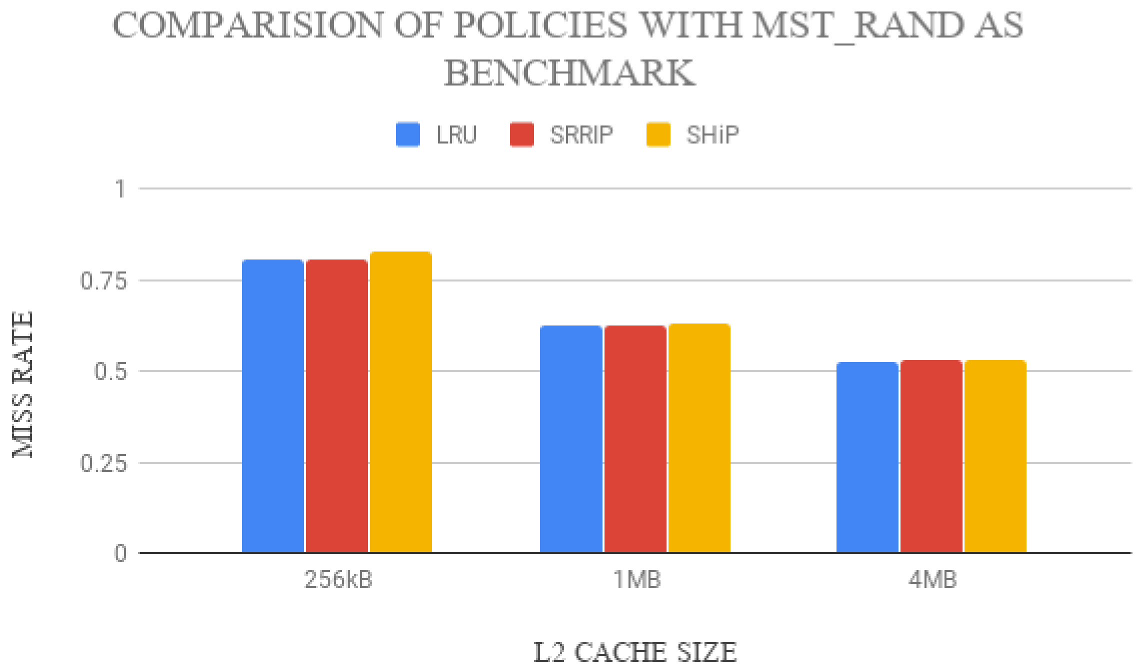
	SHiP POLICY MISS RATE TABLE			
	BFS_OPT		MST_OPT	
L2 CACHE SIZE	RAND	RMAT	RAND	RMAT
256kB	0.831225	0.853783	0.826268	0.761742
1MB	0.576541	0.605059	0.628538	0.589288
4MB	0.458035	0.440717	0.530212	0.44051

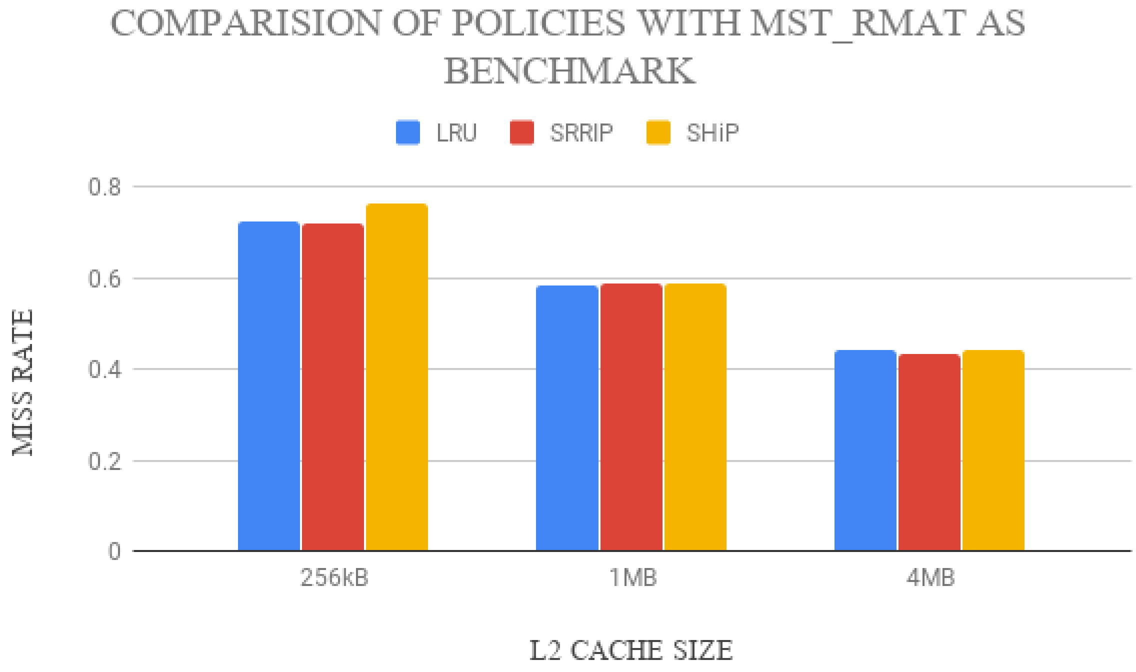


COMPARISION OF POLICIES WITH BFS_RAND AS BENCHMARK



COMPARISION OF POLICIES WITH BFS_RMAT AS BENCHMARK


COMPARISION OF POLICIES WITH MST_RAND AS BENCHMARK
LRU
SRRIP
SHiP
MISS RATE
1
0.75
0.5
0.25
0
256kB
1MB
4MB
L2 CACHE SIZE


COMPARISION OF POLICIES WITH MST_RMAT AS BENCHMARK
LRU
SRRIP
SHiP
MISS RATE
0.8
0.6
0.4
0.2
0
256kB
1MB
4MB
L2 CACHE SIZE

3. Inference and Conclusion :

LRU performs better than any other policy for recency-friendly access patterns in which the stack access pattern repeats N times. But in this assignment we considered BFS(Breadth First Search) and MST(Minimum Spanning Tree) as the benchmarks which has mixed access patterns. LRU performance degrades with mixed access patterns since it experiences cache thrashing in case of scans. In this case, SRRIP-HP(Hit Priority) which is scan resistant policy performs well as expected. On the other hand, SHiP should outperform both the SRRIP and LRU as it has the advantages of SRRIP along with finer granularity in re-reference prediction but for the benchmarks provided it is not showing good performance comparatively with other algorithms. This might be because, SHiP has extra overhead of calculating tags for the SHCT and cache line overhead since we include signature and a outcome bit for every cache line. We can clearly see that all the three policies performs similarly in all the cases with same L2 cache size. This might be because the reference papers presented might be using other benchmarks and different platforms for their testing which might not be the case in this assignment. Also might be because of the mixed patterns and configurations differences of the simulator environment.