Behaviour of Cache Replacement Policies for Graph Data

Mahendra Kishor
Department of Computer Science and Engineering
IIT ROPAR
2022csm1009@iitrpr.ac.in

I. Problem Statement

Graph analytics is a powerful tool for cybersecurity, contact tracing, and social networking. It consists of algorithms that investigate the relationships between entities involved in transactions, interactions, and organizations. It is used in finance, networking, and business logistics. A common property of graphs used in graph analytics is a power-law distribution of vertex connectivity, where a small number of vertices are responsible for a high fraction of all connections in the graph.

Graph processing is a widely used technique in many fields, but it poses significant challenges for memory and cache management due to the unique access patterns of graph data structure and algorithms. Accessing data from the main memory becomes a performance headache and the computational resources are not utilised efficiently, leading to long processing time. Graph analytics involves processing large graph data, which poses significant challenges for memory and cache management due to the unique access patterns of graph data structure and algorithms.

As we have many existing replacement policies like LRU, SHIP, DRRIP, SRRIP, HAWKEYE and MOCKINGJAY. Then which one will work best if we don't have specific replacement policies for graphs data.

II. Motivation for the solution

Because of irregular access patterns, locality of graph data is not good in cache, to evaluate this i have checked the IPC (Instruction per cycle) of normal trace and Graph trace and percentage of misses in cache for both.

In the Given table bwaves-98B is normal trace, BC-2 and BFS-3 are Graph traces.

IPC	bwaves_98B	BC-3	BFS-3
LRU	1.09958	0.158242	0.276896
DRRIP	1.09944	0.162511	0.284985
SHIP	1.09967	0.166049	0.287356
SRRIP	1.09933	0.15759	0.279496
Hawkaye	1.09968	0.170856	0.286383
Mockingjay	1.0997	0.174214	0.288693

Fig. 1. IPC for different replacement policies for Normal and Graph Traces

In the given Fig 1 and 2 we can see that there is a huge drop in IPC of graph data. Now we can see in Fig.3 Miss

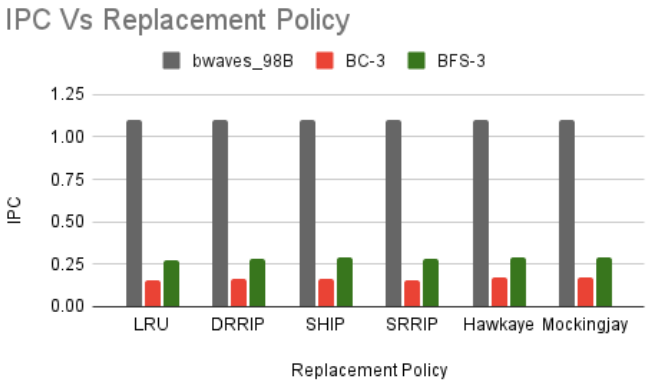

Fig. 2. Graphical View of given table

percentage at cache for normal traces and graph traces there is huge difference between them.

%OF MISS	bwaves_98B	BC-3	BFS-3
LRU	0.18%	41.70%	16.77%
DRRIP	0.18%	41.72%	16.75%
SHIP	0.18%	41.76%	16.74%
SRRIP	0.18%	41.70%	16.78%
Hawkaye	0.18%	41.79%	16.73%
Mockingjay	0.18%	41.82%	16.75%

Fig. 3. Percentage of misses at cache

So after analysing the above figures and facts there is a huge motivation to check how given replacement policies will react with graphs data and which one to choose among them if we want to get less number of misses in cache.

III. Proposed Idea

My proposed idea was to first implement the base paper "Domain-Specialized Cache Management for Graph Analytics" written by Priyank Faldu, Jeff Diamond and Boris Grot. In this paper they have explained that A power-law distribution of vertex connectivity, in which a few vertices account for a large portion of all connections in the network, is a characteristic of graphs utilized in the field of graph analytics. These hot, densely linked vertices have a high reuse rate by nature. They have discovered that because of the very erratic access

patterns of graph analytics, cutting-edge hardware cache management approaches have difficulty maximizing their reuse. In order to depict graphs in a way that is storage-effective, the Compressed Sparse Row (CSR) format is frequently utilized. Vertex and Edge are two arrays that CSR employs to encode the graph. The Vertex Array keeps track of an index that points to each vertex's first in-edge in the Edge Array. By destination vertex ID, all in-edges are organized and stored in the Edge Array. The related source vertex ID is kept in the Edge Array entry for each in-edge. For each vertex, the graph applications keep partial or complete results in an extra Property Array (or arravs).

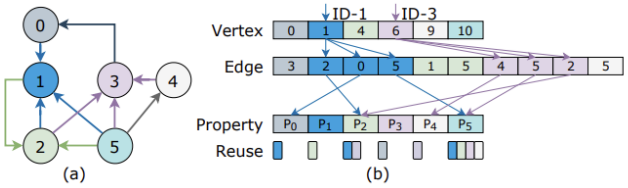

Fig. 4. CSR Format

They suggest GRASP, which is domain-specialized cache management for graph analytics at the last-level cache. By shielding them from cache thrashing, GRASP supplements current cache strategies to optimize the reuse of hot vertices while retaining enough flexibility to capture the reuse of other vertices as necessarv.

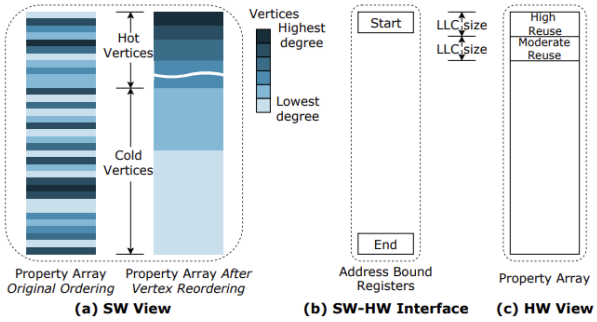

Fig. 5. GRASP Policy

GRASP is a skew-aware reordering software technique that uses vertex reordering to segregate hot vertices in a contiguous memory region. The interface of GRASP is minimal, consisting of a few configurable registers that software populates with the bounds of the Property Array during the initialization of an application. The cache admission policy chooses which data to cache, and the cache replacement policy removes the least important data. GRASP also implements specialized cache policies that protect cache blocks associated with High-Reuse LLC accesses against thrashing. Insertion policy inserts High-Reuse accesses at the MRU position, Moderate-Reuse accesses near the LRU position, and Low-Reuse accesses at the LRU position. Hit-promotion policy promotes High-Reuse LLC accesses to the MRU position on a hit. These policies provide preferential treatment to High-Reuse blocks while maintaining flexibility in exploiting temporal reuse in other cache blocks.

As i was trying to implement this base paper and explore further to do some improvement on GRASP but i was not able to implement this and changed the idea to check if we don't have replacement policies which are specific for graphs and have only normal data cache replacement policies like LRU (Least Recently Used), SHIP (Signature-based Hit Predictor for high performance caching), SRRIP (Static Re-reference Interval Prediction), DRRIP (Dynamic Re-reference Interval Prediction), Hawkeye and Mockingjay. so which one will perform best and give minimum miss percentage at LLC (Last Level Cache) for graphs data and also check that in case of SRRIP and DRRIP policy, which RRPV value will get the best results.

IV. Experimental Setup

Champsim simulator has been used for all experiments.
1- 1-core CPU
2- DRAM- 4GB
3- L1 Instuction Cache- 32KB
4- L1 Data Cache- 48KB
5- L2 Cache- 512KB
6- LLC cache- 1 MB
GAP benchmark application suite used.
3 Traces have been taken for the analysis-
BFS-3, BFS-8 are traces of graph running Breadth First Search Algorithm and BC-3 running Betweenness Centrality Algorithm.
300 Million Instructions taken for Evaluation.

V. Results and Analysis

First of all, I took DRRIP replacement policy with different RRPV values and ran simulation for all mentioned replacement policies and results are shown in the figure 6.

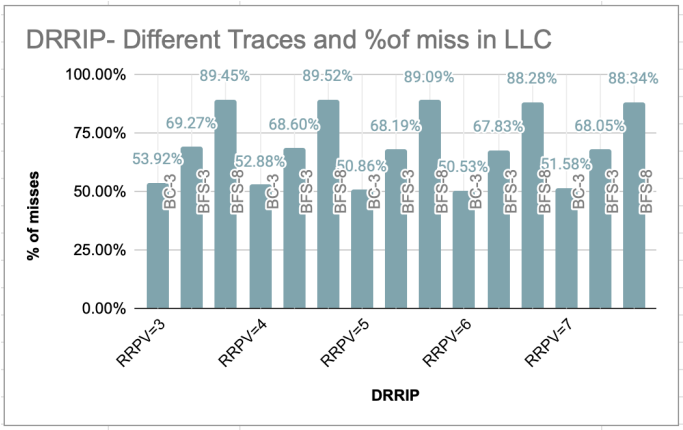

Fig. 6. % of miss in DRRIP for different RRPV values

In figure 6, we can see in the graph that for RRPV=6 we are getting minimum percentage of cache misses at LLC for all three traces, from RRPV=3 as we increase its value percentage of miss started decreasing gradually till RRPV=6 and after

that again percentage of misses started increasing at LLC in all three traces.

So it is concluded that for RRPV=6 in DRRIP replacement policy, will get least number of misses for graph data.

In the second step i took SRRIP policy for analysis of minimum percent of miss with different RRPV values. In fig. 7 we can see that for RRPV=3 we are getting least number of misses at LLC for all three traces.

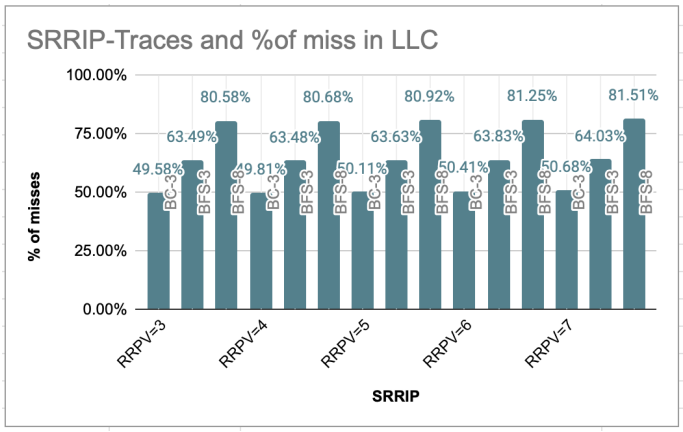

Fig. 7. % of miss in DRRIP for different RRPV values

From fig.7 has also been analysed that as we increase RRPV value increases, Percentage of miss at LLC is also increasing with some decimal points.

So it is concluded that for SRRIP replacement policy, if we take RRPV value 3 then it will give least number of misses.

In the next step, I have evaluated percentage of misses in all replacement policies mentioned above.

I have taken RRPV=6 in DRRIP and RRPV=3 for SRRIP for further evaluation.

	BC-3	BFS-3	BFS-8
LRU	50.06%	64.60%	81.07%
SHIP	51.98%	64.64%	85.75%
SRRIP	49.58%	63.49%	80.58%
MockingJay	53.60%	63.91%	80.94%
Hawkaye	55.72%	63.89%	83.09%
DRRIP	50.53%	67.83%	88.28%

Fig. 8. % of miss in different Replacement Policies

In the above figures we can see that SRRIP outperforms all the replacement policies taken into consideration for analysis and we can not say anything about which one is performing worst among all of them because Hawkeye is performing worst for BC-3 trace and DRRIP performs worst for BFS-3 and BFS-8 traces, i have also ran simulation with one more BFS trace and in that DRRIP also performs worst. so only for BFS algorithm in graph we can say that DRRIP is the worst performing replacment policy.

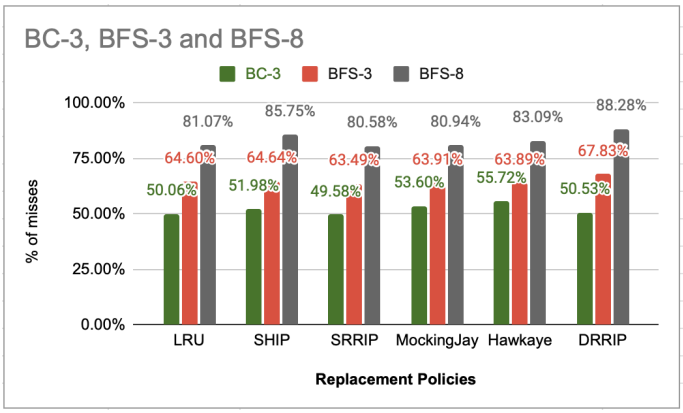

Fig. 9. Graphical representation % of miss in different Replacement Policies

VI. Conclusion

Graph access patterns are very critical and irregular, so if we don't have graph specific cache replacement policies and have replacement policies for normal data like LRU, SHIP , SRRIP ,DRRIP , Hawkeye and Mockingjay, so how these are reactning with graph data.

It has been seen that SRRIP replacement policy will outperform all the replacement policies mentioned above with RRPV value equals to 3 at LLC, there was not very much difference between miss percentage because only 300 million instructions have been taken into consideration but as number of instructions will increase difference in miss percentage will also increase .

Worst performing replacement policy can not bee decided because for BFS algorithm graph traces DRRIP performed worst and for BC algorithm graph traces Hawkeye performed worst.

References

[1] Carole-Jean Wu, Aamer Jaleel, Will Hasenplaugh, Margaret Martonosi, Simon C. Steely Joel Emer "ship: Signature-based Hit Predictor for high performance caching" 2011 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)

[2] Aamer Jaleel, Kevin B. Theobald, Simon C. Steely, Joel Emer "High performance cache replacement using re-reference interval prediction (RRIP)" ,19 June 2010

[3] Akanksha Jain Calvin Lin "Hawkeye: Leveraging Belady's Algorithm for Improved Cache Replacement", 2017.

[4] Ishan Shah, Akanksha Jain, Calvin Lin "Effective Mimicry of Belady's MIN Policy" 2022.

[5] Priyank Faldu, Jeff Diamond, Boris Grot "Domain-Specialized Cache Management for Graph Analytics"26th International Symposium on High-Performance Computer Architecture (HPCA 2020).