Thinking of a Title

Navya

November 7, 2023

Abstract

Enjoy!

Contents

1 Part I : Implementing and Analyzing Replacement Policies

The simulations for the replacement policies has been tested with five traces given below, alongside the metrics of **Miss Rate**, **IPC**, **SpeedUp**, **Miss Count** and **Total Count** are provided.
We will make possible infrences about the performance of these policies and the possible structure or stress/ test parameters tested by the trace files in the next section after we list out the results and charts.
**concerned metrics are for L2C level*

1.1 602.gcc_s-1850B.champsimtrace.xz

Policy	**IPC**	**Speedup**	**Miss Rate* (%)**
LRU	2.554	1	73.09
FIFO	2.556	1.0008	74.48
LFU	2.563	1.0035	73.39
BIP ($\epsilon = 0$)	2.554	1.0000	74.52
BIP ($\epsilon = 0.25$)	2.525	0.9886	75.69
BIP ($\epsilon = 0.5$)	2.550	0.9984	76.73
BIP ($\epsilon = 0.75$)	2.554	1.0000	82.18
BIP ($\epsilon = 1$)	2.553	0.9996	84.17

1.2 603.bwaves_s-1740B.champsimtrace.xz

Policy	**IPC**	**Speedup**	**Miss Rate* (%)**
LRU	1.005	1	85.24
FIFO	0.999	0.9956	85.19
LFU	1.005	1.0036	87.37
BIP ($\epsilon = 0$)	1.005	1.0000	85.24
BIP ($\epsilon = 0.25$)	1.005	1.0000	85.67
BIP ($\epsilon = 0.5$)	1.002	0.9970	86.24
BIP ($\epsilon = 0.75$)	1.003	0.9980	86.82
BIP ($\epsilon = 1$)	1.003	0.9980	87.51

1.3 619.lbm_s2677B.champsimtrace.xz

Policy	**IPC**	**Speedup**	**Miss Rate (%)**
LRU	0.2188	1.0000	34.14
FIFO	0.2191	1.0013	33.03
LFU	0.2192	1.0018	85.79
BIP ($\epsilon = 0$)	0.2188	1.0000	34.14
BIP ($\epsilon = 0.25$)	0.2187	0.9995	43.75
BIP ($\epsilon = 0.5$)	0.2185	0.9986	55.32
BIP ($\epsilon = 0.75$)	0.2174	0.9936	70.97
BIP ($\epsilon = 1$)	0.2195	1.0031	86.03

1.4 bc-0.trace.gz

Policy	**IPC**	**Speedup**	**Miss Rate* (%)**
LRU	0.1637	1	66.81
FIFO	0.1642	1.002	68.04
LFU	0.1676	1.0254	84.25
BIP ($\epsilon = 0$)	0.1637	1.0000	66.81
BIP ($\epsilon = 0.25$)	0.1637	1.0000	69.63
BIP ($\epsilon = 0.5$)	0.1636	0.9994	73.10
BIP ($\epsilon = 0.75$)	0.1645	1.0049	76.40
BIP ($\epsilon = 1$)	0.1640	1.0018	79.82

1.5 sssp-3.trace.gz

Policy	**IPC**	**Speedup**	**Miss Rate* (%)**
LRU	0.4396	1	60.37
FIFO	0.4397	1.0002	62.08
LFU	0.4596	1.0458	72.63
BIP ($\epsilon = 0$)	0.4396	1.0000	60.37
BIP ($\epsilon = 0.25$)	0.4390	0.9986	62.60
BIP ($\epsilon = 0.5$)	0.4389	0.9984	65.63
BIP ($\epsilon = 0.75$)	0.4447	1.0116	69.96
BIP ($\epsilon = 1$)	0.4445	1.0111	79.82

1.6 The Charts(The Graph Is In Log Scale , Hence Extrapolated)

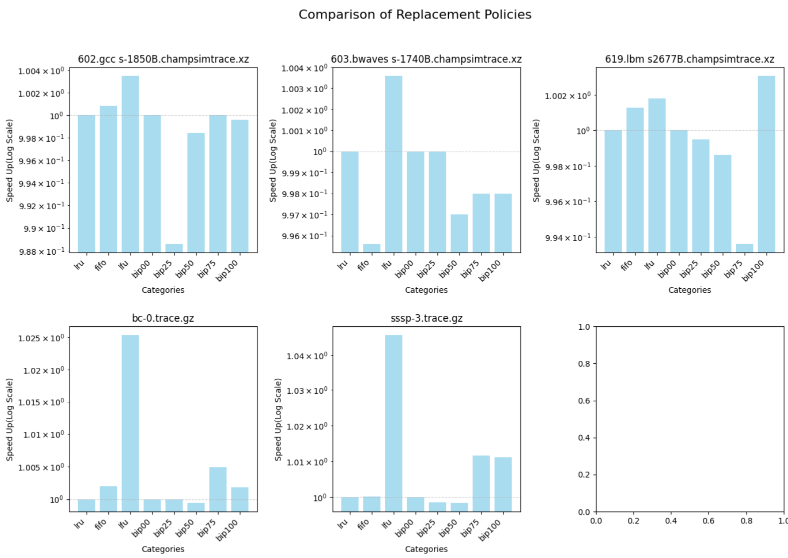

Figure 1: Observed SpeedUp for different Policies

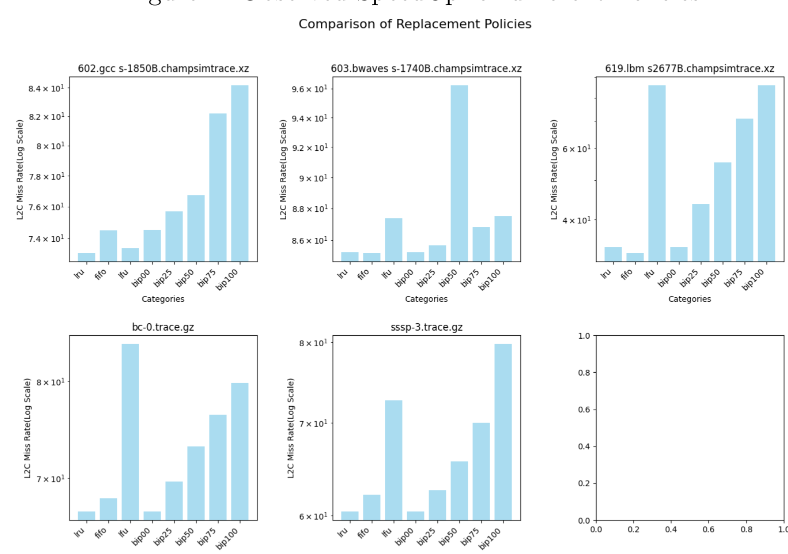

Figure 2: L2C Miss Rate for Different Policies

1.7 Analysis of Replacement Policies

1.7.1 LRU (Least Recurrent Unit) and LFU(Least Frequently Used)

LRU evicts the data which is least recent. This implies that LRU will perform good when the recent data has high chances to be accessed again. Which means that it is useful for temporal locality where certain addresses are asked again and again. LFU evicts the one with the lowest counter value(frquency).

The problem that arises with the LFU is that even blocks which were accessed earlier will have high counter value if they were accessed many folds in past. Instead, the recent data will have a lower counter value initially. But it's not good to choose past data over presnt just because it has higher coutner value. In technical terms, LFU doesn't evaluate on the basis of recency.

We can see that the miss rate of LFU is high,especially for **619.lbm ...xz**, and the main reason I think is as mentioned above.

1.7.2 FIFO (First in and First Out)

First In and First Out evicts on the basis of the timestamp of the cache line when it was fetched into the cache. So if the data has been fetched in past but we are accessing the stuff often, for fifo that is a past data and is a candidate for eviction, which it should not be. Unlike lfu or lru, it doesn't evaluate on the bais of either recency of access or frequency of access.

But as we noticed that the fifo works fairly well so a natural question is why? The reason is that thnigs are temporarily accessed. So the starting timestamp is related to the recency and usage. If a cache line has a low timestamp then mostly that data is not going to be accessed here. Which means the **Birth Timestamp** metrics has a close relation with the **Recency** although they are not same.

The miss rate observed with for **FIFO** matches with it's functinality.

1.7.3 BIP (Binary Insertion Policy)

Binary insertion policy chooses between the least recently used and most recently used cache lines for eviction, on a probability basis. Different epsilon are good for different siuations. When epsilon is close to zero, which means we will evict MRU with least probability. This is useful for those situation where the most recent data is more likely to be accessed. In the programs where the older data is more likely to be used and therefore should be preserved, epsilon close to 1 works better. But that is certainly not a usual case. As you can see that the miss rate increases with an increment in the epsilon, this tells us that the recency is vital and the recently accessed data should be given priority for preservation.

2 Part II : Implementing and Analyzing Stream-Prefetching

Here, we have implemented a stream-prefetcher in the ChampSim framework, the implementation can be inspected from the code. The prefetcher was then tested against the ip-stride prefetcher which was provided with Champsim. The metrics reported here are **SpeedUp**, **Prefetcher Accuracy**, **L2C Load MPKI** and **L1D Load MPKI**.
**notes L1D and ** denotes L2C*

2.1 602.gcc_s-1850B.champsimtrace.xz

Metric	**IP-Stride**	**Stream-Prefetcher**
IPC	2.502	2.766
SpeedUp	1	1.105
LOADS	3519471	3548480
LOAD_MISS	1894863	1831825
LOAD_MPKI	538.54	516.22
*LOADS	330631	327472
*LOAD_MISS	209043	175740
*LOAD_MPKI	632.25	536.65
USEFUL	102667	186395
ISSUED	637794	711502
ACCURACY	16.09%	26.19%

2.2 603.bwaves_s1740B.champsimtrace.xz

Metric	**IP-Stride**	**Stream-Prefetcher**
IPC	1.395	1.421
SpeedUp	1	1.018
LOADS	5194117	5178741
LOAD_MISS	552776	558891
LOAD_MPKI	106.48	107.92
*LOADS	330075	333496
*LOAD_MISS	171371	156728
*LOAD_MPKI	519.35	469.95
USEFUL	161318	209730
ISSUED	567028	285192
ACCURACY	28.44%	73.53%

2.3 619.lbm_s2677B.champsimtrace.xz

Metric	**IP-Stride**	**Stream-Prefetcher**
IPC	0.2188	0.2194
SpeedUp	1	1.0027
LOADS	1523797	1539892
LOAD_MISS	611562	636738
LOAD_MPKI	401.34	413.49
*LOADS	283732	285355
*LOAD_MISS	181990	208019
*LOAD_MPKI	511.09	728.98
USEFUL	75975	67800
ISSUED	247630	119235
ACCURACY	30.6%	56.86%

2.4 bc0.trace.gz

Metric	IP-Stride	Stream-Prefetcher
IPC	0.1637	0.163
SpeedUp	1	0.9957
LOADS	6578142	6580325
LOAD_MISS	2166772	2180303
LOAD_MPKI	329.38	331.33
*LOADS	2020161	2020278
*LOAD_MISS	1744006	1772547
*LOAD_MPKI	863.30	877.37
USEFUL	107733	61668
ISSUED	488320	167172
ACCURACY	22.06%	36.88%

2.5 sssp-3.trace.gz

Metric	IP-Stride	Stream-Prefetcher
IPC	0.4396	0.42
SpeedUp	1	0.955
LOADS	6970689	6972005
LOAD_MISS	1291043	1359542
LOAD_MPKI	185.21	195.00
*LOADS	1066627	1067597
*LOAD_MISS	791647	818912
*LOAD_MPKI	742.21	767.06
USEFUL	137580	93927
ISSUED	462775	195646
ACCURACY	29.72%	48.00%

2.6 Analysis

The stream prefetcher I implemented used **PREFETCH DISTANCE 10** and **PREFETCH DEGREE 3**. I also implemented this for other combinations like **(6,3), (6,2)** etc. But I got best performance for this. This prefetcher start prediction once it get to the mature stage, but we can also do prefetching once it is in second stage. That would be like prefetching when we are not sure that we will fetch in that direction or not. But that is better than **next line prefetcher** which just prefetches the next line without confirming the direction the way we do in stream prefetcher.

One of the good things is that it first assure the direction of access then start predicting. This improves the quality of prediction as we can observe in the final accuracy of the stream prefetcher in comparison to **ip stride prefetcher**.

One important question even know is what should we do when there is a cache hit in the monitoring region? How can utilize the information that we get from the this? In an earlier attempt I did that if we get two hits in the a mature monitoring region then we will shift the region. Or we can also do something like if hit happens in the monitoring region change the timestamp of the region.

2.7 The Charts

These are the graphical realizations of the above metrics.

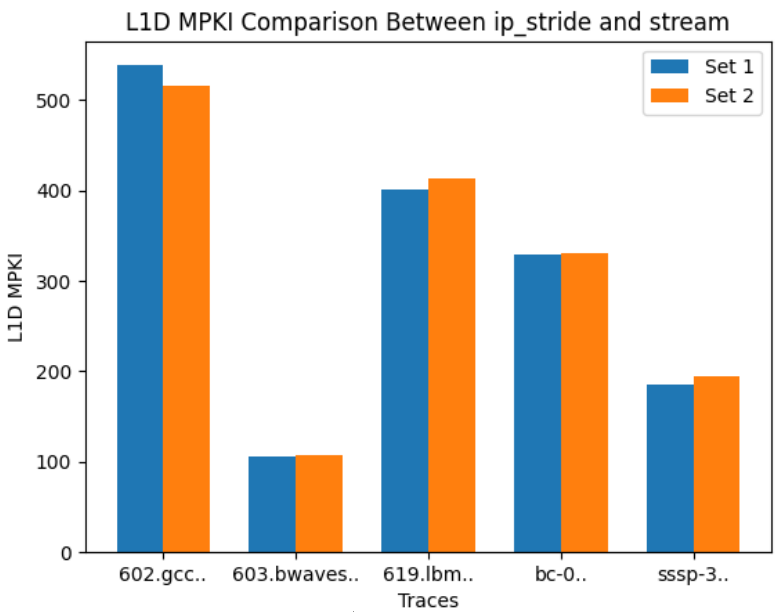

Figure 3: L1D_LOAD_MPKI `(lower is better)`

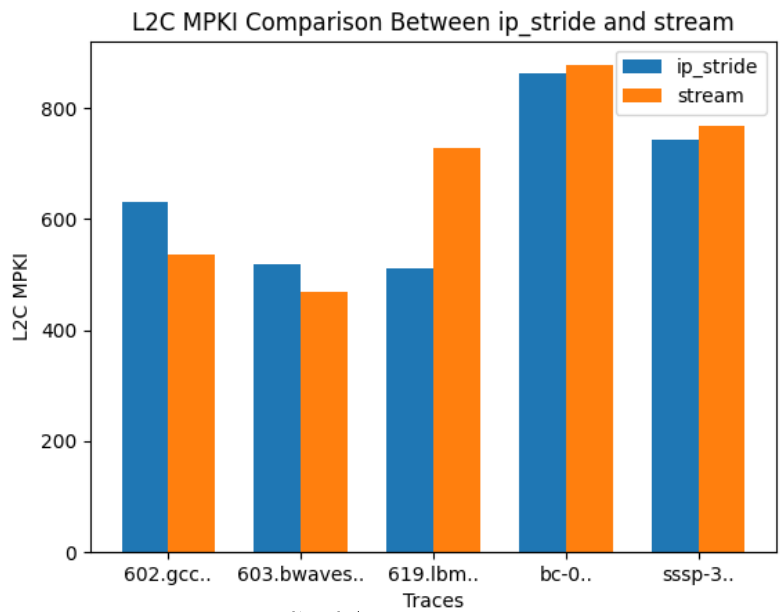

Figure 4: L2C_LOAD_MPKI (lower is better)

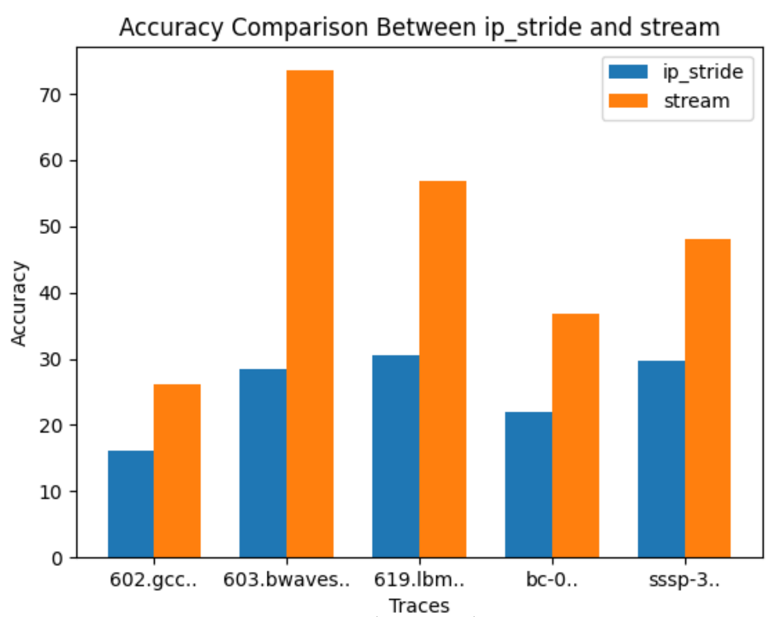

Figure 5: PREFETCHING ACCURACY (higher is better)

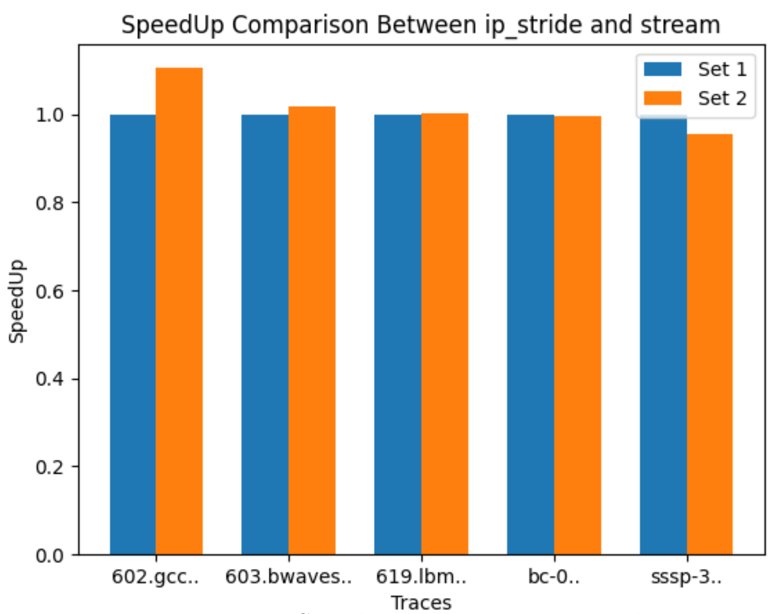

Figure 6: SpeedUP `(higer is better)`