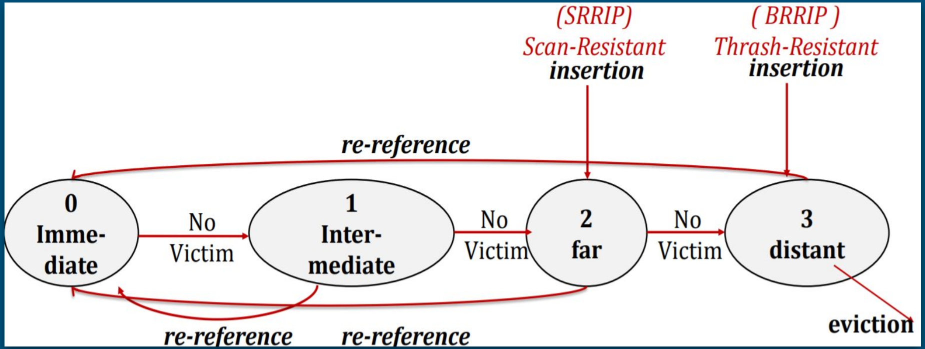
Grad Lab - Contrasting LRU with (S/D)RRIP

Snehil Verma (sv24792)
Abhishek R. Kumar (ark2878)

RRIP Summary



- LRU
  - good with recency friendly workloads
  - cannot perform well with mixed workloads
- RRIP -Thrash and Scan resistant policy - Aamer Jaleel [ISCA 10]
  - Goal: save some data in the cache in case of thrashing or scans
  - Introduced the idea of distant, (far,) intermediate, and near-immediate re-reference
  - Motivated from DIP, employed set dualing to introduce dynamic behaviour - DRRIP

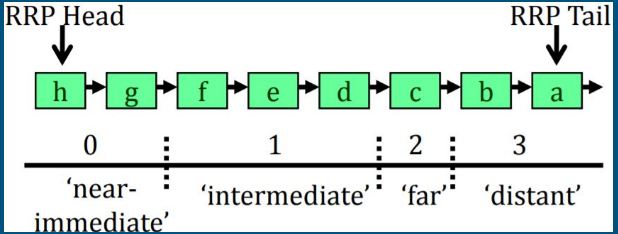


Image Credits (undergrad class): https://www.cse.iitk.ac.in/users/biswap/CS698Y/lectures/L6.pdf

Simpoints [ASPLOS-X 02] Setup

Parameters

- Slice length: 100M
- Warmup length: 1M
- Max K: 10
- Reference dataset
- Benchmarks:
  - bzip2
  - cactusADM
  - hmmer
  - mcf
  - sphinx3

Regions and Slices

	# Regions	# Slices
bzip2	8	5269
cactusADM	5	27755
hmmer	10	18511
mcf	7	3042
sphinx3	8	32676

Input Set: selected the representative input as mentioned in the paper - “Analysis of redundancy and application balance in the SPEC CPU2006 benchmark suite” - Lizy John [ISCA 07]

Weights for Benchmarks

Cluster #	**bzip2**	**hmmer**	**mcf**	**sphinx3**	**cactusADM**
0	0.10704	0.19313	0.33728	0.11739	0.26003
1	0.11596	0.12171	0.12952	0.17276	0.00011
2	0.24597	0.16239	0.05293	0.12348	0.0018
3	0.11786	0.00005	0.28501	0.20538	0.73515
4	0.09395	0.07109	0.00592	0.09368	0.00292
5	0.17821	0.13938	0.01085	0.13346	
6	0.07914	0.02393	0.1785	0.07072	
7	0.06187	0.08541		0.08312	
8		0.02939			
9		0.17352			

Metrics

- **LLC’s MPKI** (misses per kilo instructions) - to determine how the replacement policy reduces the number of LLC misses
- **IPC** - to determine the performance gained by employing a particular replacement policy
    - is not directly related with number of misses

Methodology

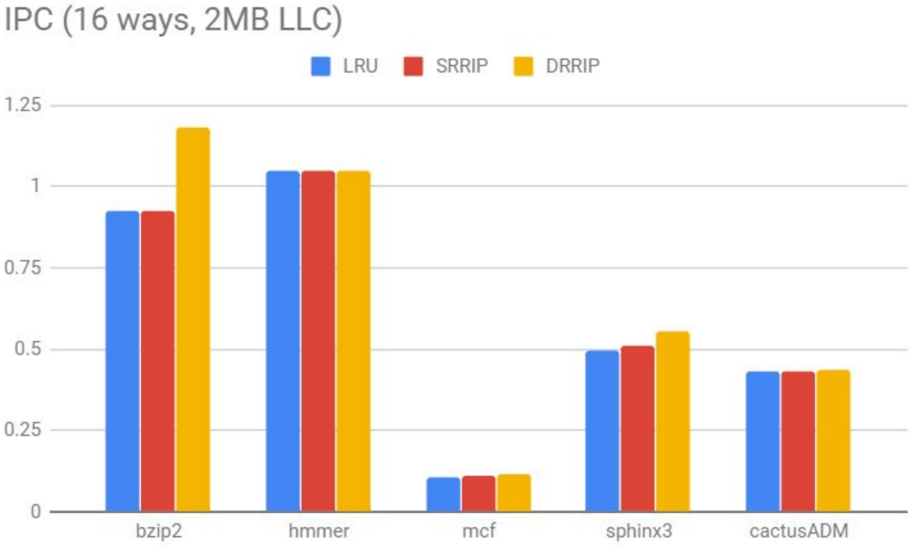
Default System Configurations

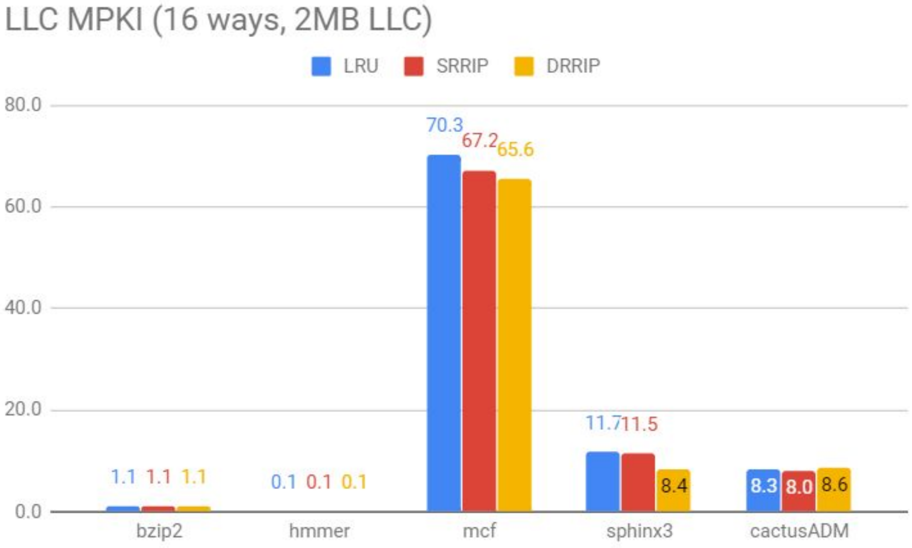
- PIN version: 3.5 (pinplay-drdebug-3.5-pin-3.5-97503-gac534ca30-gcc-linux.tar.bz2)
- GCC compiler: 5.4.0
- Fortran compiler: 5.4.0

Methodology -- Running on ChamSim

- Default branch predictor: **Bimodal**
- Replacement Policies: **LRU, SRRIP, DRRIP**
- L1D prefetcher: **no, next-line**
  - Next-line prefetcher is anticipated to perform good in L1 data cache - more spatial locality
- L2C prefetcher: **no, IPC stride**
  - Stride prefetcher is anticipated to perform good for L2 cache - spatial locality with some filtered by L1 data cache.
- #Cores: **one**
  - Multi core simulation seems complicated with simpoints
- Warmup length: **5M**
- Detailed Simulation: **95M**

MPKI and IPC (LLC size: 2MB , #Ways: 16)

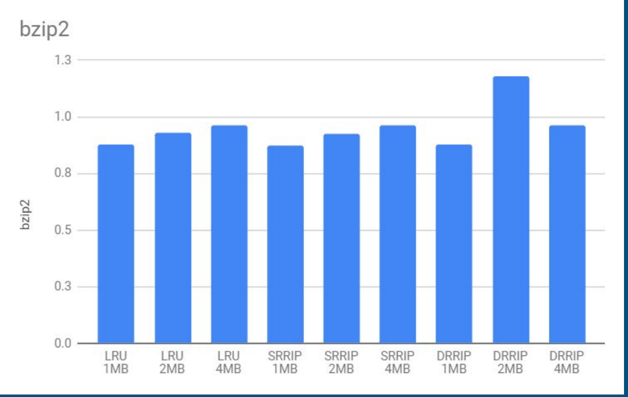


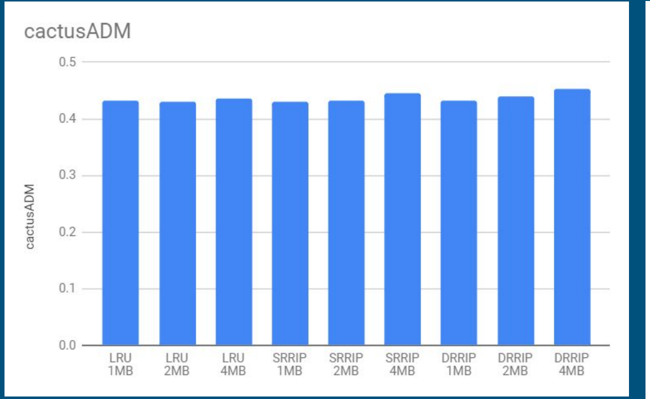


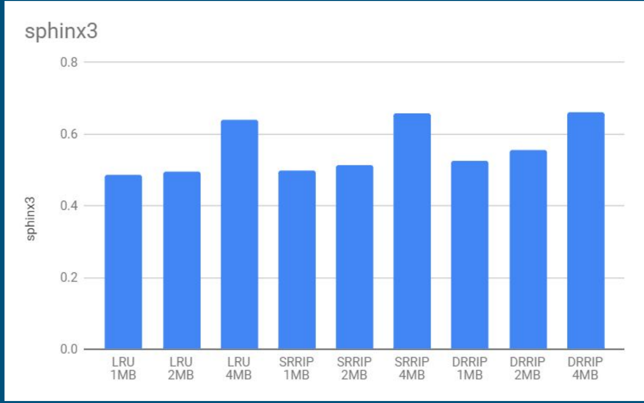
- LLC MPKI and IPC are not always co-related
- mcf is highly memory intensive benchmark

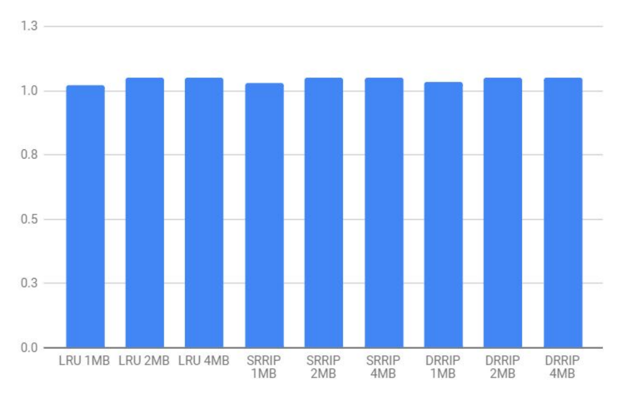
- Reduced most LLC MPKI in mcf (must have a large working set)

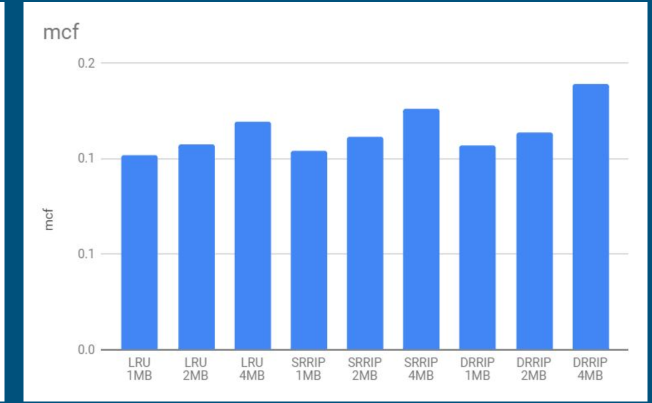
IPC Variation with Cache Size (#Ways: 16)





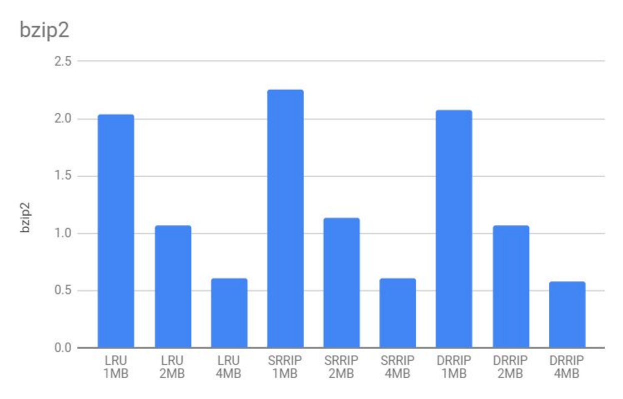


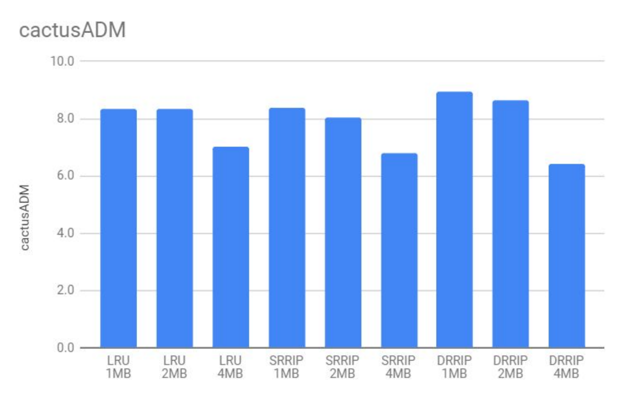


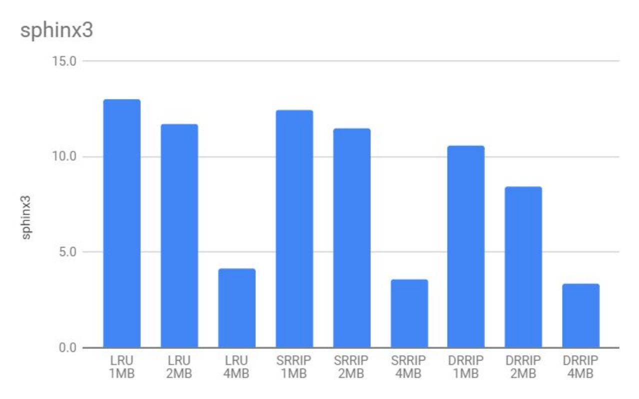


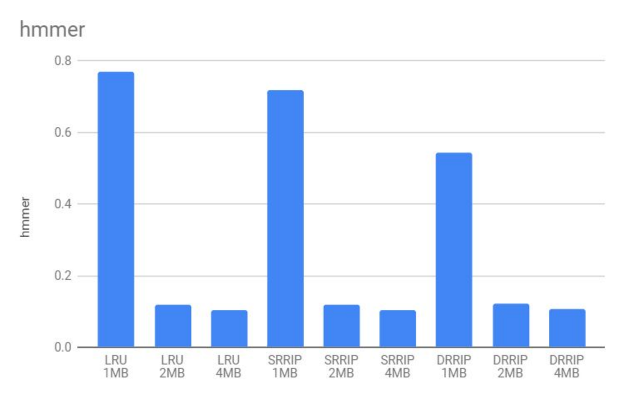
- Size: 1MB, 2MB, 4MB
- Increasing cache size not always showed benefits
- Significant improvement observed in mcf and sphinx3

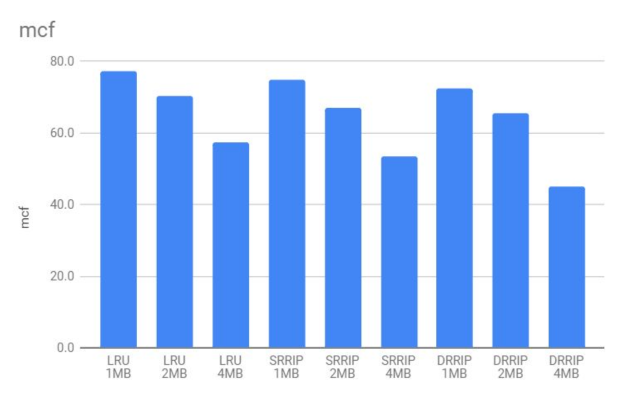
LLC MPKI Variation with Cache Size (#Ways: 16)


bzip2


cactusADM

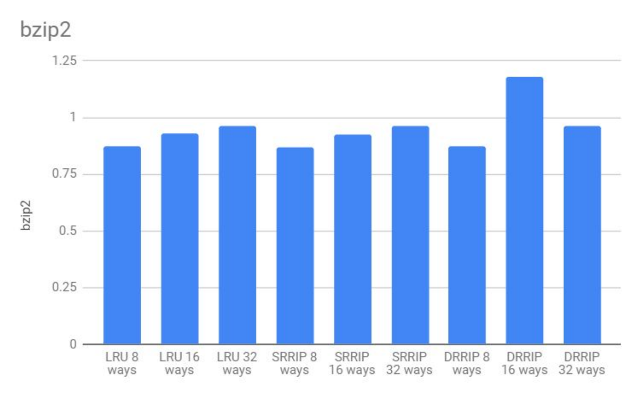

sphinx3

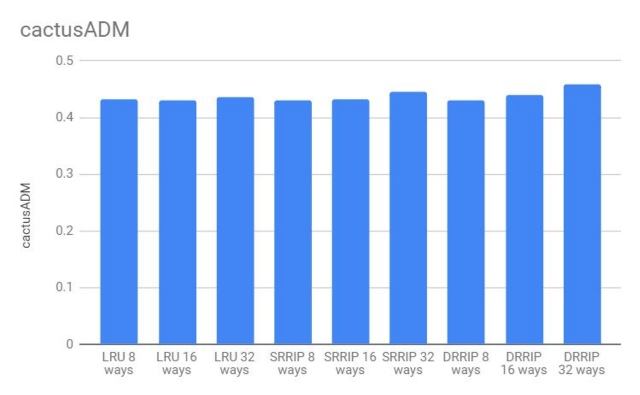

hmmer

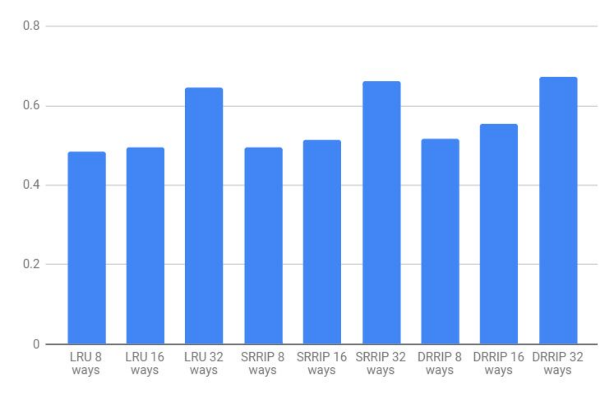

mcf

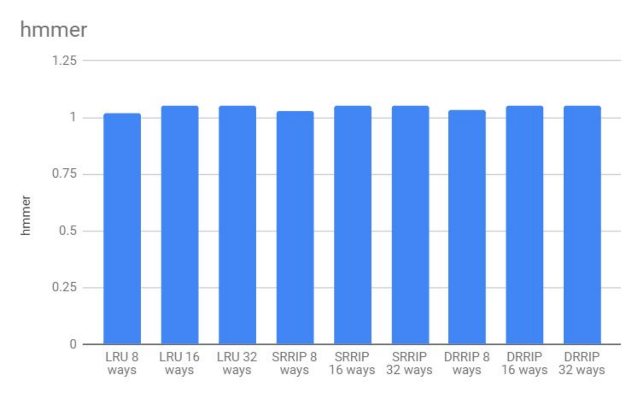
- Size: 1MB, 2MB, 4MB
- Operating dataset of bzip2, hmmer, sphinx3 are not very large, hence benefited more from large cache size (less MPKI)

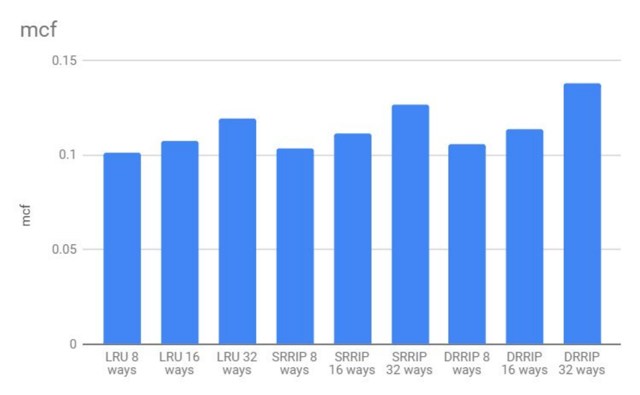
IPC Variation with number of Ways (LLC size: 2MB)





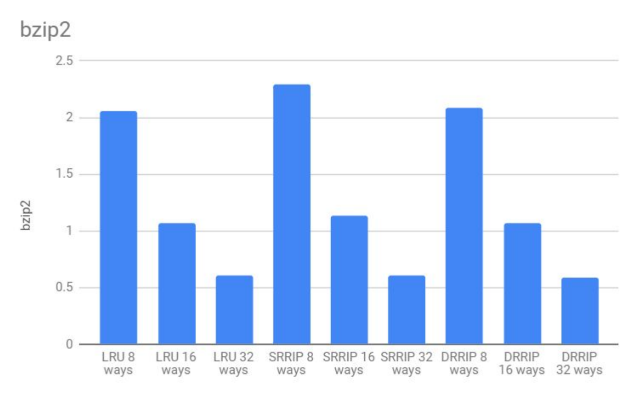


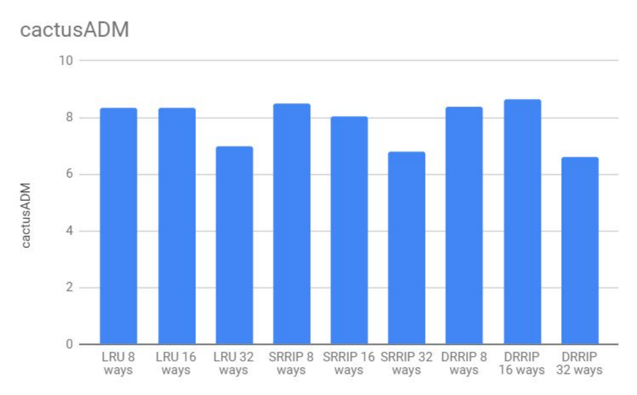


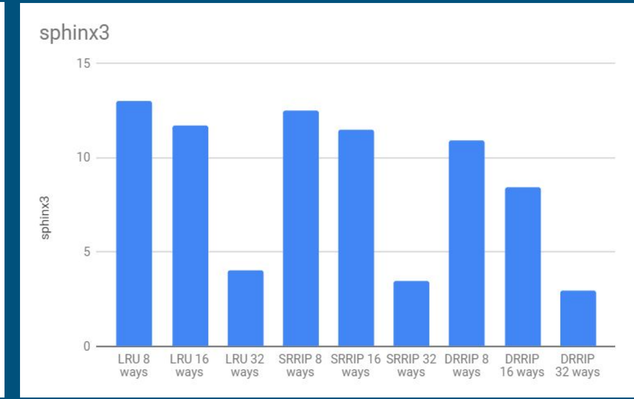


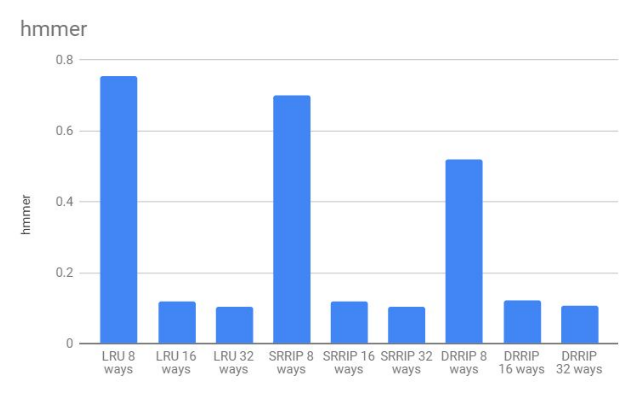
- Ways: 8, 16, 32
- Significant improvement only in sphinx3 and mcf (accesses must not have been equally spread across less ways)

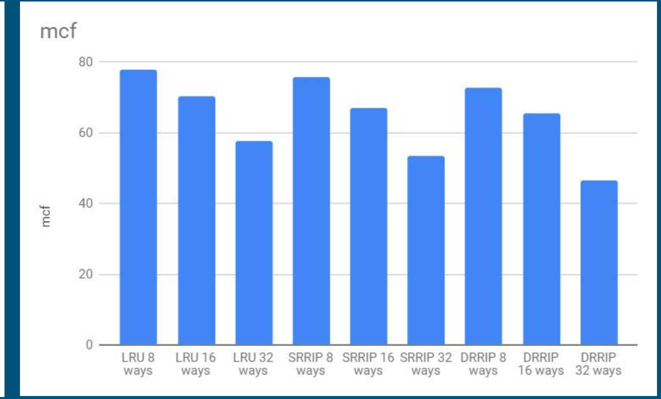
LLC MPKI Variation with number of Ways (LLC size: 2MB)





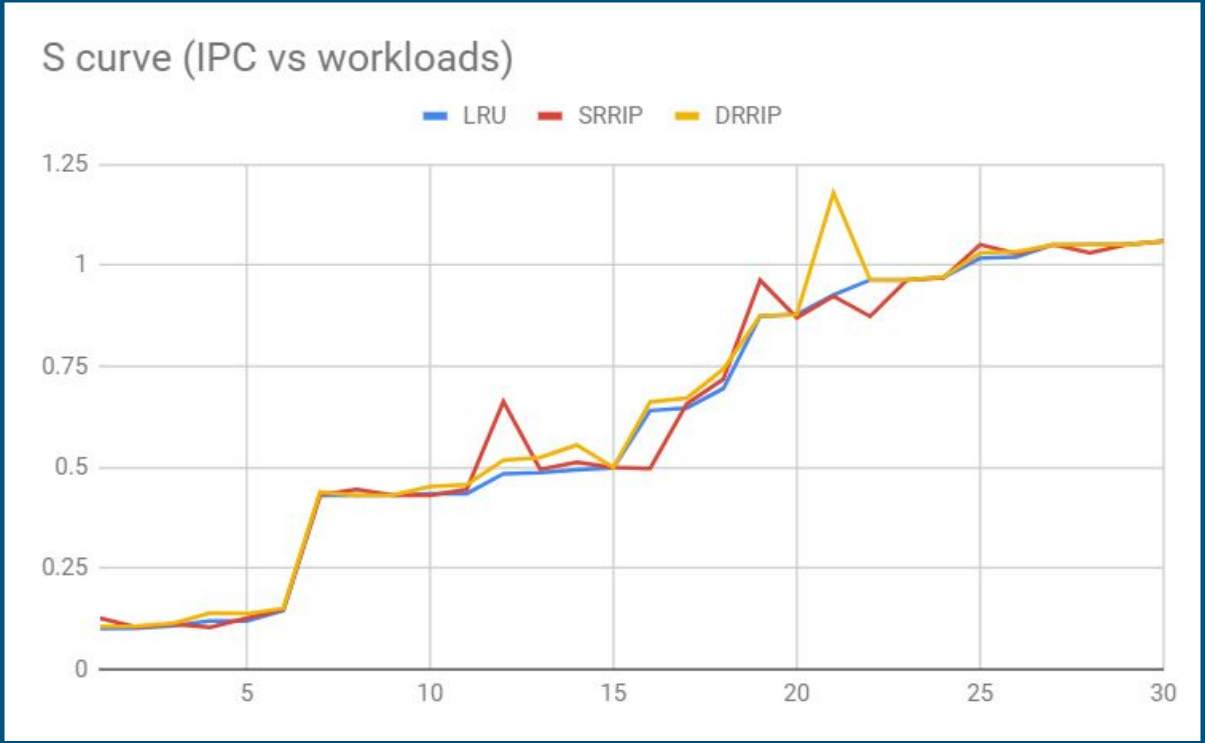






- Ways: 8, 16, 32
- Larger number of ways resulted in lower MPKI for all the benchmarks (which is expected)

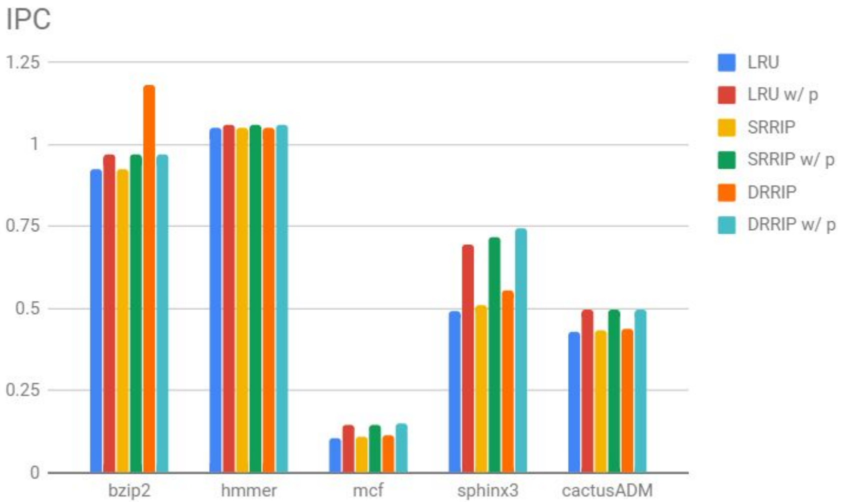
S - curve

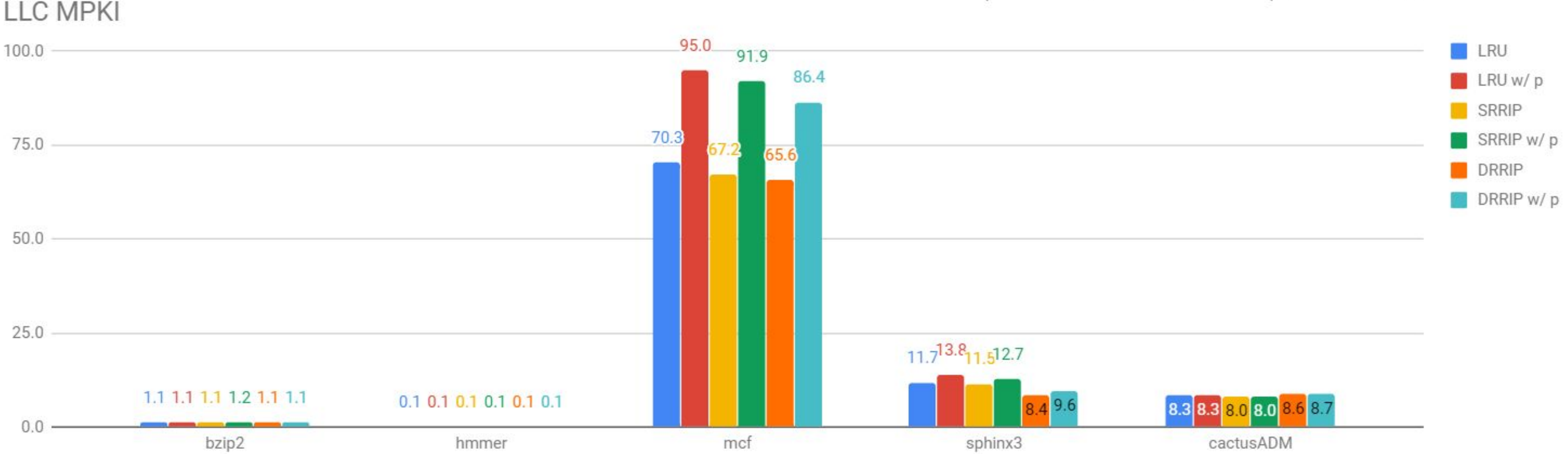


- RRIP policies hover around LRU, sometimes better and sometimes worse
- The performance of SRRIP varies a lot over the workloads
- DRRIP is more stable as compared to SRRIP
- Still a large room for improvement (SDBP [MICRO 10], SHiP [MICRO 11], Hawkeye [ISCA 16])

Effect of prefetcher

- In general, the prefetcher increases the performance, however, due to increase in number of requests the LLC MPKI also increases
- The policies should be made prefetch-aware





Conclusion

- Better replacement policies should improve on large range of workloads
- Introducing some dynamic-ness in the policy improves performance over a range of workloads (SRRIP vs DRRIP)
- The replacement policies should be made prefetch-aware to provide significant improvements when the prefetcher is turned on.

Thank You !