COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF MICHIGAN

Learning Cache Replacement with CACHEUS [1]

Liana V. Rodriguez, Farzana Yusuf, Steven Lyons, Eysler Paz, Raju Rangaswami, Jason Liu, Ming Zhao, Giri Narasimhan
USENIX Conference on File and Storage Technologies (FAST 21), 2021.

Group 15, EECS 583, Fall 2021

Jiachen Liu, Linyi Jin, Ziqiao Ma, Wanqi Liang
(In Speaking Order)

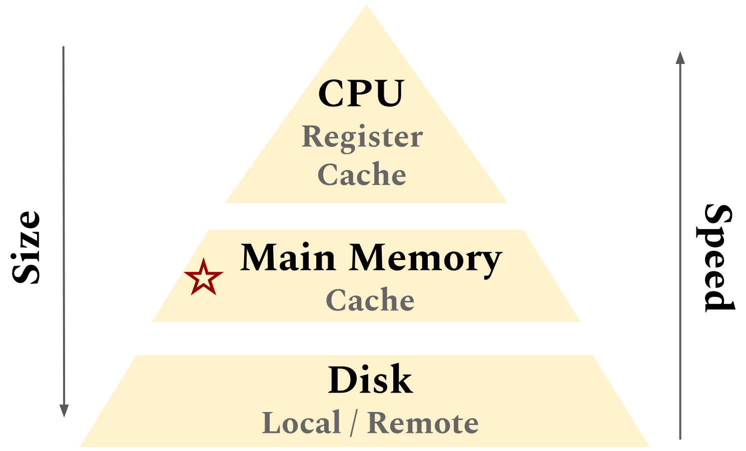
Dec 6th, 2021

Introduction

- Memory System:
  - Cache:

    Fast but relatively small in capacity;
  - Permanent storage:

    Large but relatively slow in speed;
- Machine Learning (ML):
  - Improves decision making;
- Cache Management + Machine Learning:
  - Improves performance.



Workload Primitives

- LRU-Friendly
  - Best handled by the least recently used (**LRU**) caching algorithm;
- LFU-friendly
  - Best handled by the least frequently used (**LFU**) caching algorithm;
- Scan
  - A subset of stored items are accessed **exactly once;**
- Churn
  - **Repeated accesses** to a subset of stored items with equal probability.

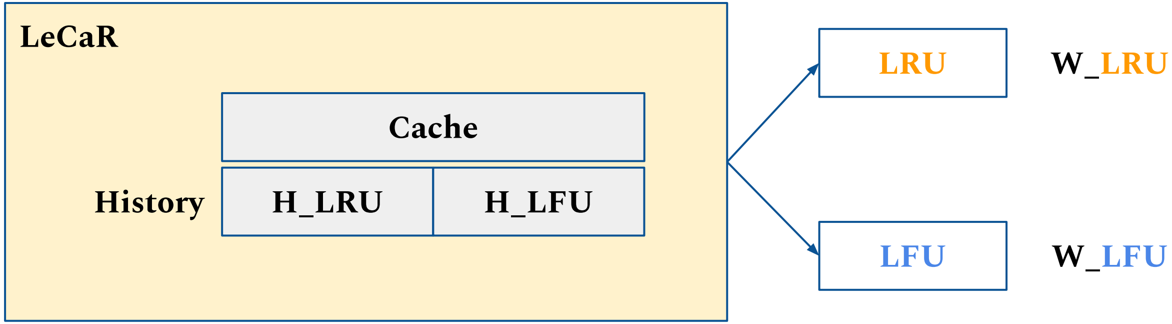
Workload Primitives

- Caching algorithms handling of workload primitive types:

Algorithm	Churn	Scan	LRU	LFU
ARC [3]	✗	✓	✓	✗
LIRS [4]	✗	✓	✗	✗
DLIRS [5]	✗	✓	✓	✗
LeCaR [2]	✓	✗	✓	✓

LeCaR: Introduction

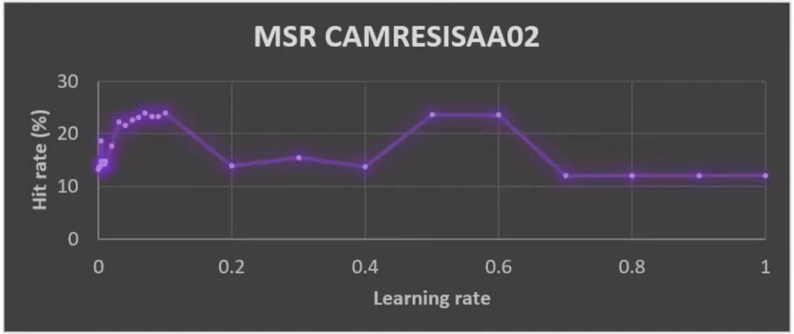
- ML-Based: Reinforcement **Le**arning On **Ca**che **R**eplacement [2]
  - Simple: LRU, LFU as experts;
  - Adaptive: Update weights;
  - Outperforms state-of-the-art: Small cache sizes.

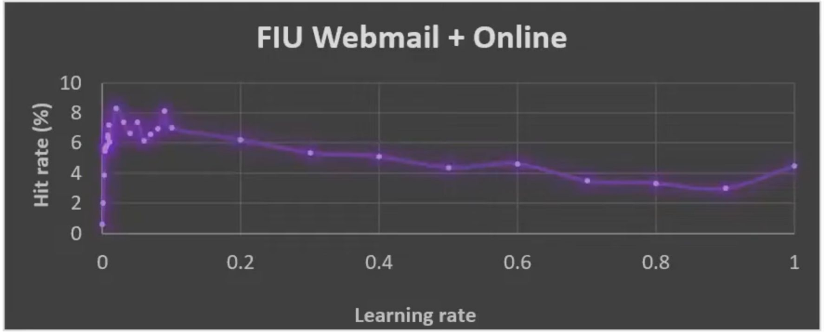


COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF MICHIGAN

LeCaR: Limitations

- Fixed Learning Rate:
  - Empirically chosen: 0.45.





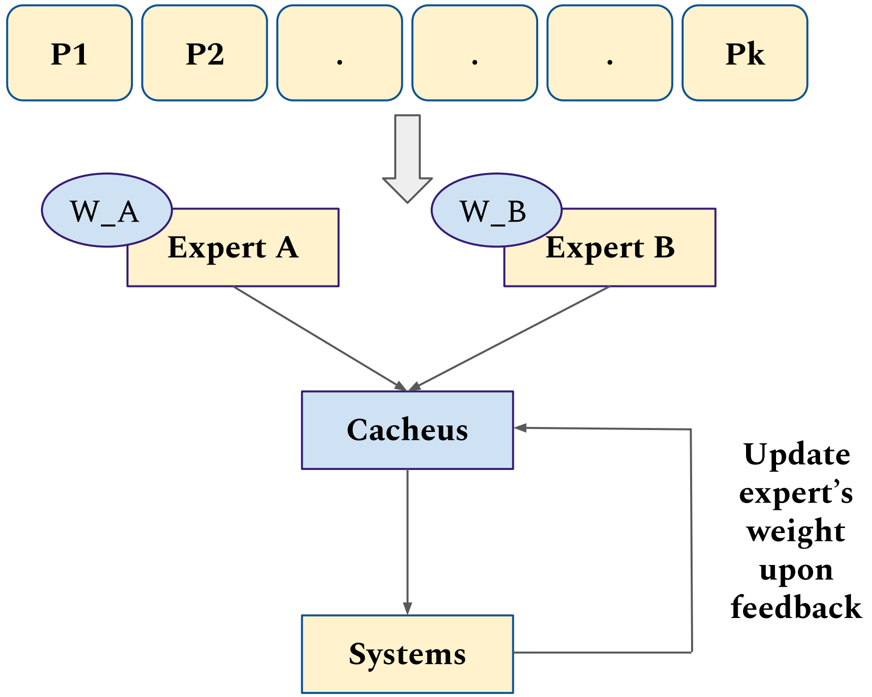
LeCaR: Limitations

- Cannot Handle Scan:

Algorithm	Churn	Scan	LRU	LFU
ARC	✗	√	√	✗
LIRS	✗	√	✗	✗
DLIRS	✗	√	√	✗
LeCaR	√	✗	√	√

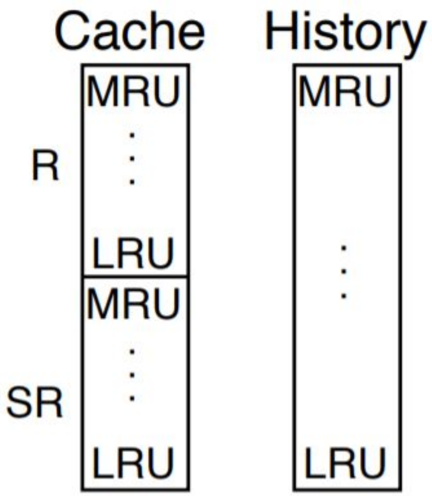
CACHEUS: Solutions

- Adaptive learning rate;
- Improve experts:
  - Introduce scan resistance
    - Replace LRU: ARC/LIRS/DLIRS -> Failed
    - **Scan resistant LRU: SR-LRU**
  - Improve churn resistance
    - **Churn resistant LFU: CR-LFU**



SR-LRU: Cache Partitioning

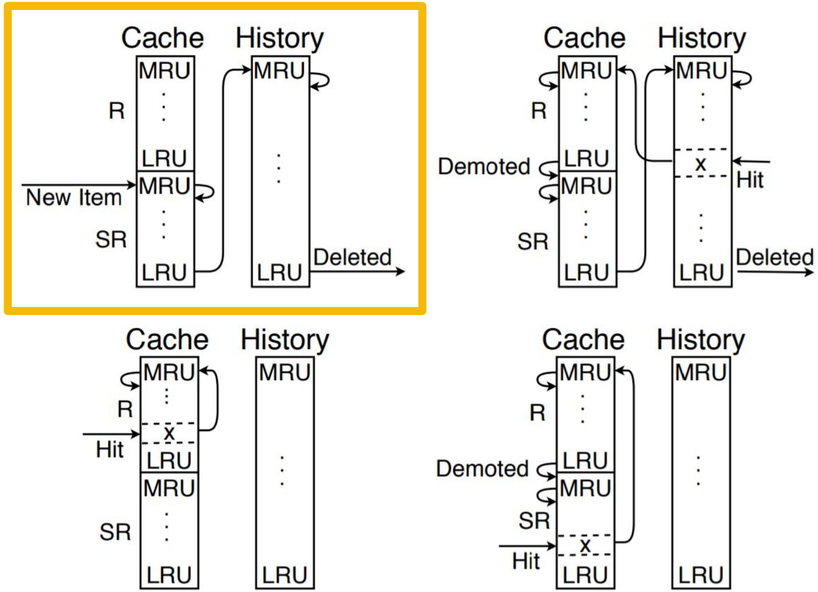
- Cache partitioning: similar to ARC and LIRS.
  - Partition Reuse (R):
    - Items with multiple accesses;
  - Partition Scan Resistance (SR):
    - Single access items;
    - Older items with multiple accesses.
- Why Partition SR?
  - MRU evicts the previously inserted page placed at the top of the stack;
  - SR Houses new items so that they don't affect important items in R;
  - SR allows SR-LRU to be scan resistant.



MRU: Most Recently Used
LRU: Least Recently Used

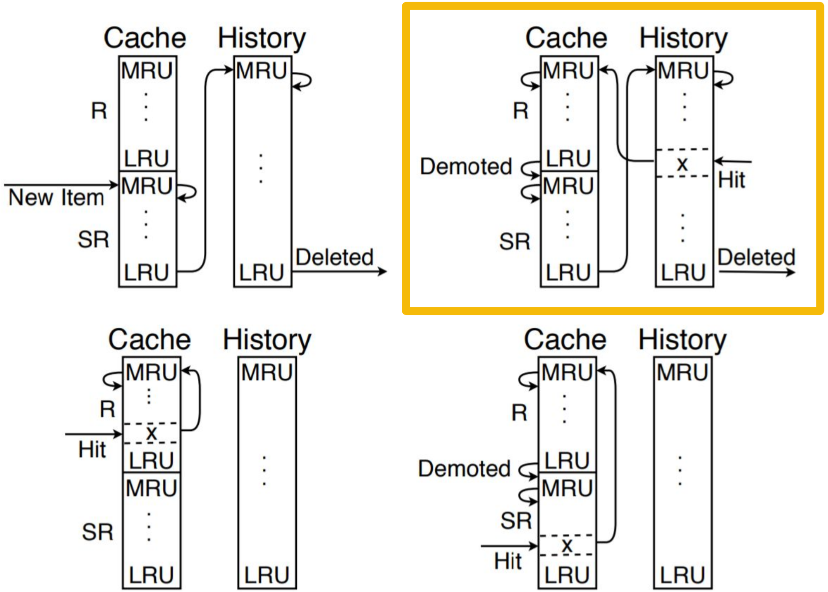
SR-LRU: Algorithm Explained

- **Miss in Cache + Miss in History:**
  - **Insert new item to the MRU position of SR;**
- Miss in Cache + Hit in History:
  - Move x to the MRU position of R;
- Hit in Cache R:
  - Move x to the MRU position of R;
- Hit in Cache SR:
  - Move x to the MRU position of R;



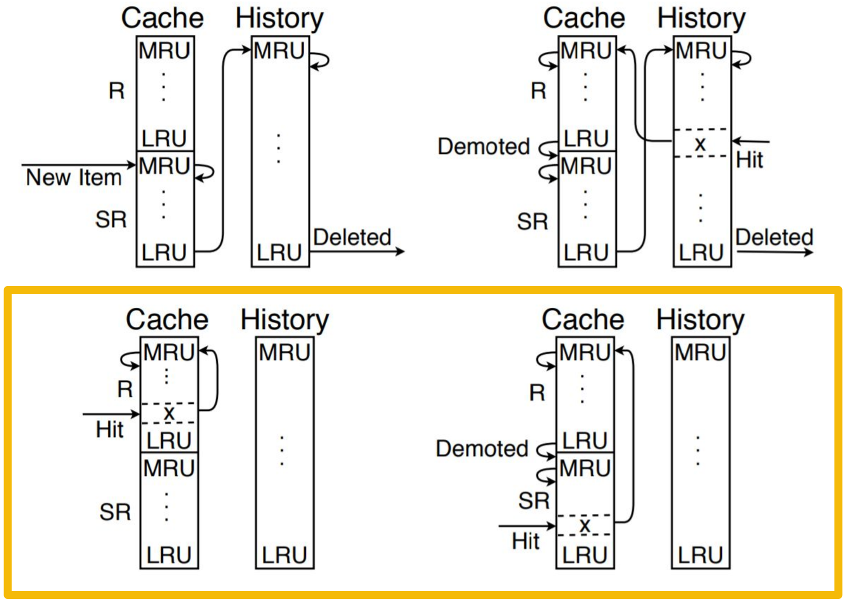
SR-LRU: Algorithm Explained

- Miss in Cache + Miss in History:
  - Insert new item to the MRU position of SR;
- **Miss in Cache + Hit in History:**
  - **Move x to the MRU position of R;**
- Hit in Cache R:
  - Move x to the MRU position of R;
- Hit in Cache SR:
  - Move x to the MRU position of R;



SR-LRU: Algorithm Explained

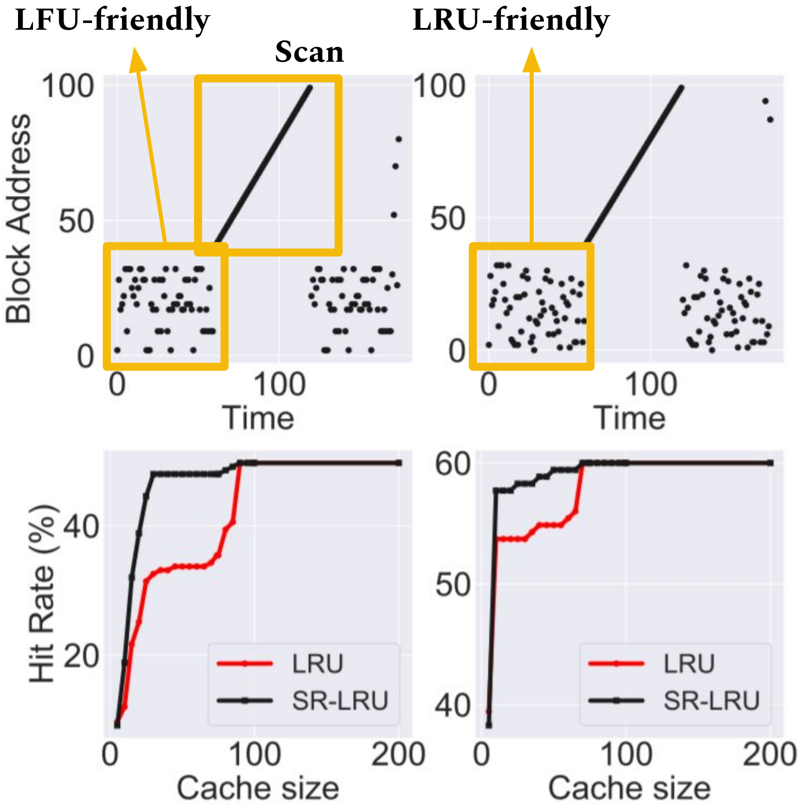
- Miss in Cache + Miss in History:
  - Insert new item to the MRU position of SR;
- Miss in Cache + Hit in History:
  - Move x to the MRU position of R;
- **Hit in Cache R:**
  - **Move x to the MRU position of R;**
- **Hit in Cache SR:**
  - **Move x to the MRU position of R;**



COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF MICHIGAN

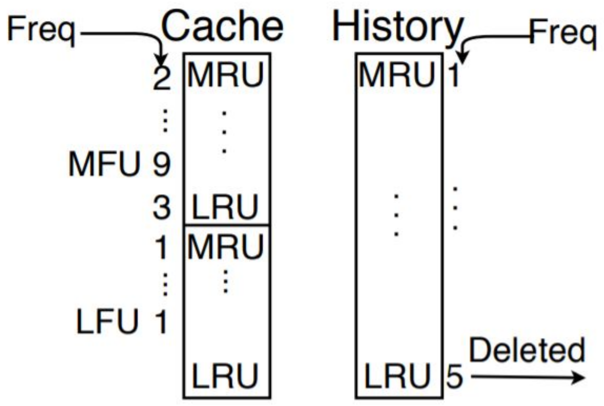
SR-LRU: Evaluation

- Scan + **LFU**-Load: Left;
  - A performance increase in small cache sizes;
- Scan + **LRU**-Load: Right.
  - A performance increase in small cache sizes;



CR-LFU: Cache Partitioning

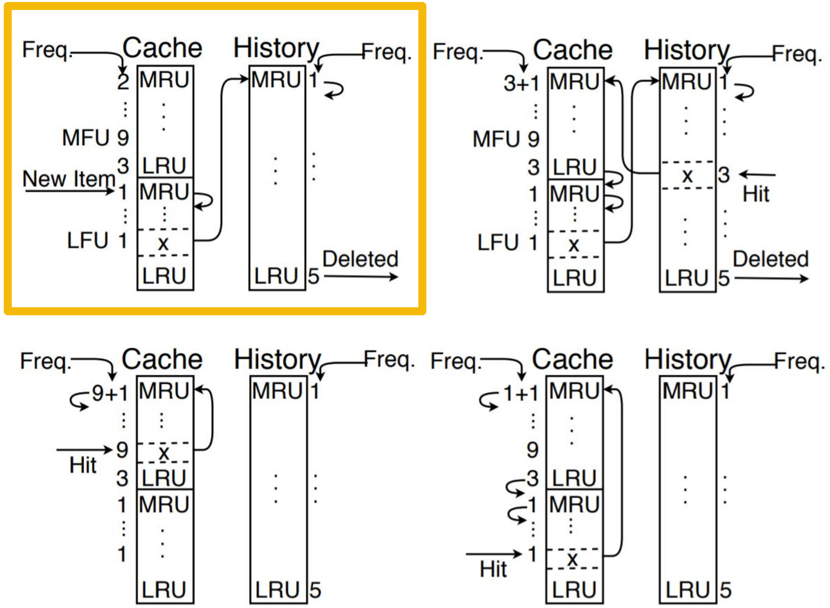
- Cache partitioning:
  - Partition by Frequency (MFU/LFU):
    - Cache partitioned by frequency of use;
  - Ordered by Recency (MRU/LRU):
    - Each partition maintaining recent uses.
- Why Frequency + Recency?
  - LRU repeatedly inserted and evicted items into/from the cache if #accessed > cache size;
  - LFU assigns equal importance to all items with the same frequency;
  - CR-LFU Chooses the MRU item to break ties when several LFU.



MRU: Most Recently Used
LRU: Least Recently Used
MFU: Most Frequently Used
LFU: Least Frequently Used

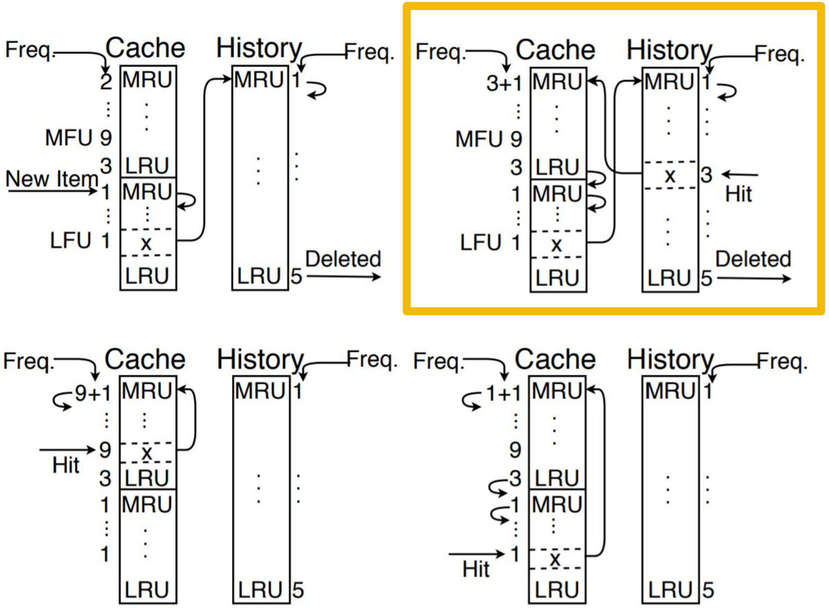
CR-LFU: Algorithm Explained

- **Miss in Cache + Miss in History:**
  - **Evict x at the MRU position of LFU;**
- Miss in Cache + Hit in History:
  - Move x to the MRU position of MFU;
- Hit in Cache MFU:
  - Move x to the MRU position of MFU;
- Hit in Cache LFU:
  - Move x to theMRU position of MFU;



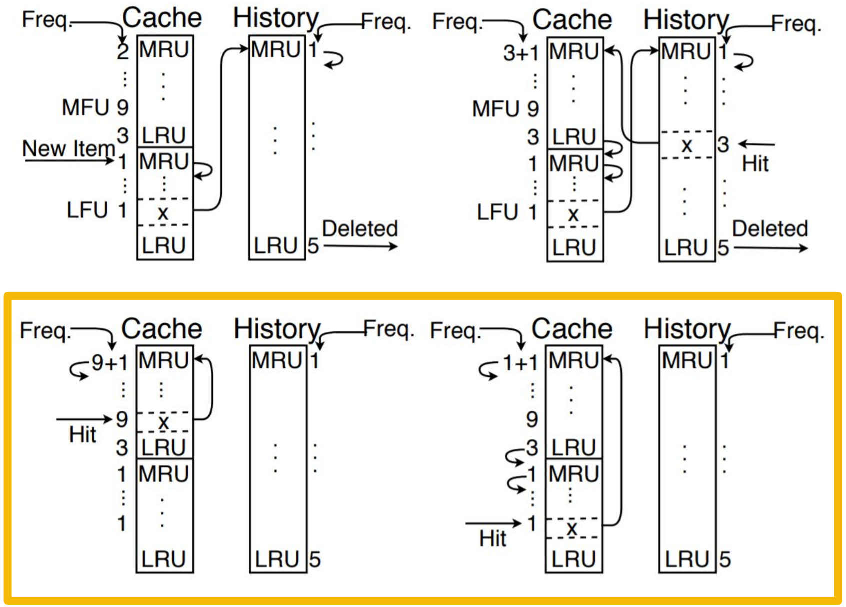
CR-LFU: Algorithm Explained

- Miss in Cache + Miss in History:
  - Insert new item to the MRU position of LFU;
- **Miss in Cache + Hit in History:**
  - **Move x to the MRU position of MFU;**
- Hit in Cache MFU:
  - Move x to the MRU position of MFU;
- Hit in Cache LFU:
  - Move x to theMRU position of MFU;



CR-LFU: Algorithm Explained

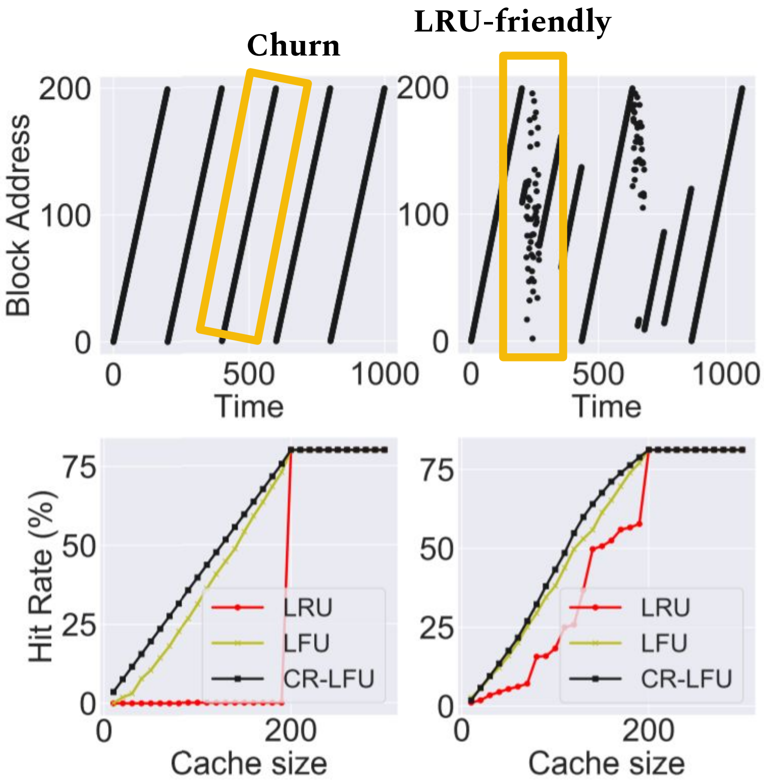
- Miss in Cache + Miss in History:
  - Insert new item to the MRU position of LFU;
- Miss in Cache + Hit in History:
  - Move x to the MRU position of MFU;
- **Hit in Cache MFU:**
  - **Move x to the MRU position of MFU;**
- **Hit in Cache LFU:**
  - **Move x to theMRU position of MFU;**



COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF MICHIGAN

CR-LFU: Evaluation

- Pure Churn: Left;
  - Avg Performance Increase: 8.67%;
- Churn + **LRU**-Load: Right.
  - Avg Performance Increase: 3.83%;



COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF MICHIGAN

Experiments

- Datasets: 5 different sources;
- Cache size: 0.05, 0.1, 0.5, 1,5, 10%;
- Comparison:
  - 3 CACHEUS variants:
    - (ARC, LFU)
    - (LIRS, LFU)
    - (SR-LRU, CR-LFU)
  - 6 baselines:
    - LRU, LFU, ARU, LIRS, LeCaR, DLIRS
- Total experiments: 17,766

Dataset	#Traces
FLU	**184**
MSR	**22**
CloudPhysics	**99**
CloudVPS	**18**
CloudCache	**6**
Total	**329**

CACHEUS: Evaluation

- Paired t-test analysis;
- Significance:
  - P-value: threshold of 0.05;
  - **Green**: CACHEUS variant significantly better;
  - **Red**: CACHEUS variant significantly worse;
  - **Gray**: no significant difference;
- Effective size:
  - Cohen's d-measure;
  - Bright color: high effective size.

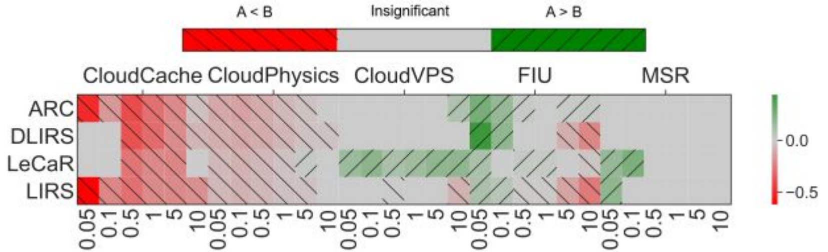


Figure: CACHEUS (ARC, LFU) vs.others

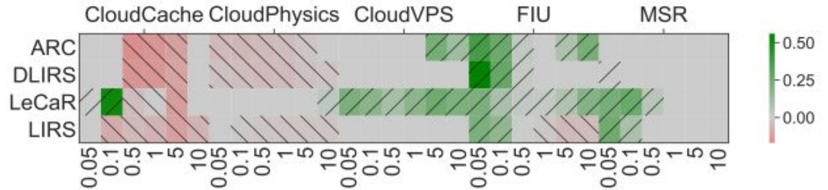


Figure: CACHEUS (LIRS,LFU) vs.others

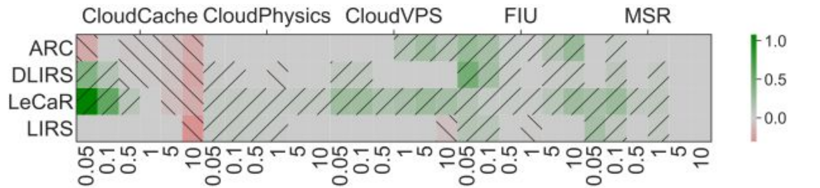


Figure: CACHEUS (SR-LRU,CR-LFU) vs.others

CACHEUS: Statistical Analysis

- CHACHEUS Variants:
  - (SR-LRU, CR-LFU) is distinctly the best;
- Results:
  - Best in **47%**;
  - Worse in **13%**;
  - Insignificant in **40%**;
  - Effective size `[-0.31, 1.08]`.

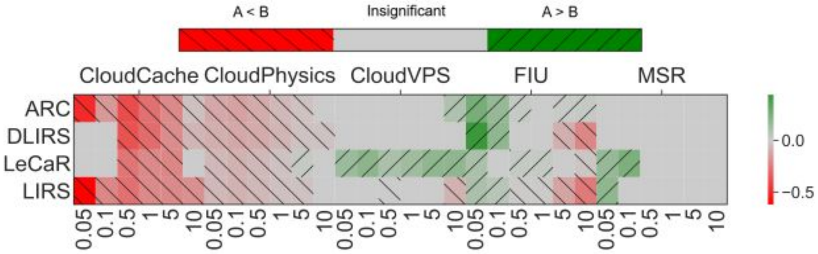


Figure: CACHEUS (ARC, LFU) vs.others

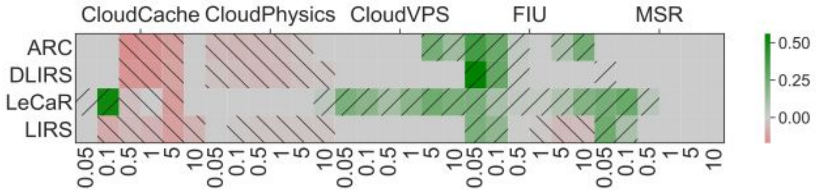


Figure: CACHEUS (LIRS,LFU) vs.others

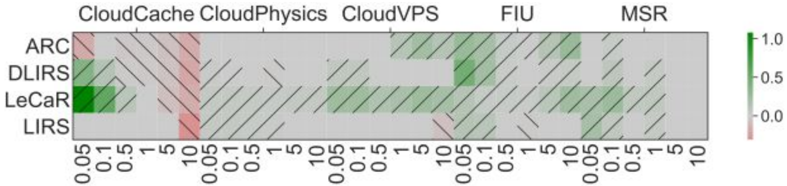


Figure: CACHEUS (SR-LRU,CR-LFU) vs.others

Conclusion

- Cache Management + Machine Learning:
    - Improves performance;
- Workload primitive types:
    - LRU-friendly, LFU-friendly, Churn, Scan;
- CACHEUS: Improved Cache replacement algorithm:
    - **Adaptive** learning rate;
    - Improved **experts**: SR-LRU and CR-LFU;
    - **Comprehensive** evaluations;
    - Outstanding **performance.**

COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF MICHIGAN

References

[1] **Learning Cache Replacement with CACHEUS** [Paper] [Code] [Video]
Liana V. Rodriguez, Farzana Yusuf, Steven Lyons, Eysler Paz, Raju Rangaswami, Jason Liu, Ming Zhao, Giri Narasimhan
USENIX Conference on File and Storage Technologies (FAST 21), 2021.

[2] **Driving Cache Replacement with ML-based LeCaR**
Giuseppe Vietri, Liana V. Rodriguez, Wendy A. Martinez, Steven Lyons, Jason Liu, Raju Rangaswami, Ming Zhao, Giri Narasimhan
USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 18), 2018.

[3] **ARC: A Self-tuning, Low Overhead Replacement Cache**
N. Megiddo, D. S. Modha
USENIX Conference on File and Storage Technologies (FAST 03), 2003.

[4] **LIRS: An Efficient Low Inter-reference Recency Set Replacement Policy to Improve Buffer Cache Performance**
S. Jiang and X. Zhang
ACM Sigmetrics Conference (SIGMETRICS 02), 2002.

[5] **DLIRS: Improving low inter-reference recency set cache replacement policy with dynamics**
C. Li
11th ACM International Systems and Storage Conference (SYSTOR 18), 2018.

[6] **ACME: Adaptive caching using multiple experts**
I. Ari, A. Amer, R. B. Gramacy, E. L. Miller, S. A. Brandt, D. D. Long
WDAS, 2002.