LIPA: A Learning-based Indexing and Prefetching Approach for Data Deduplication

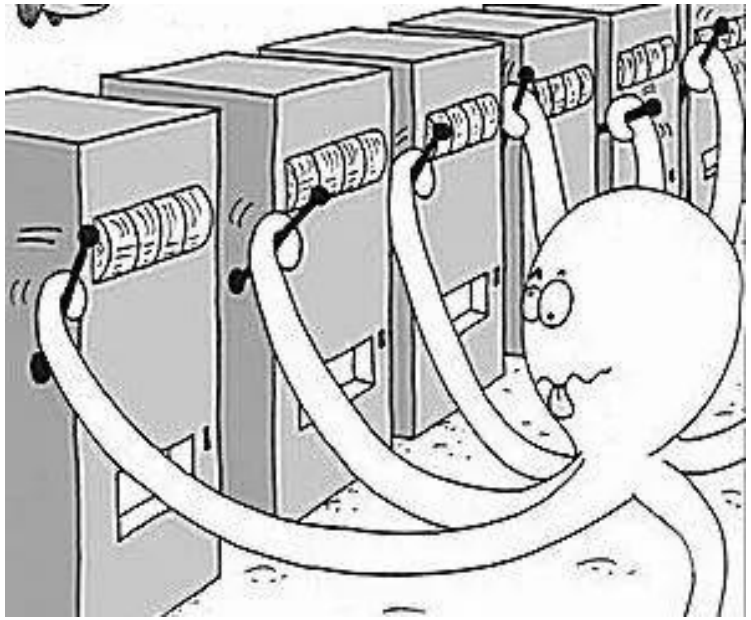
Guangping Xu*, Chi Wan Sung, Quan Yu, Hongli Lu, Bo Tang

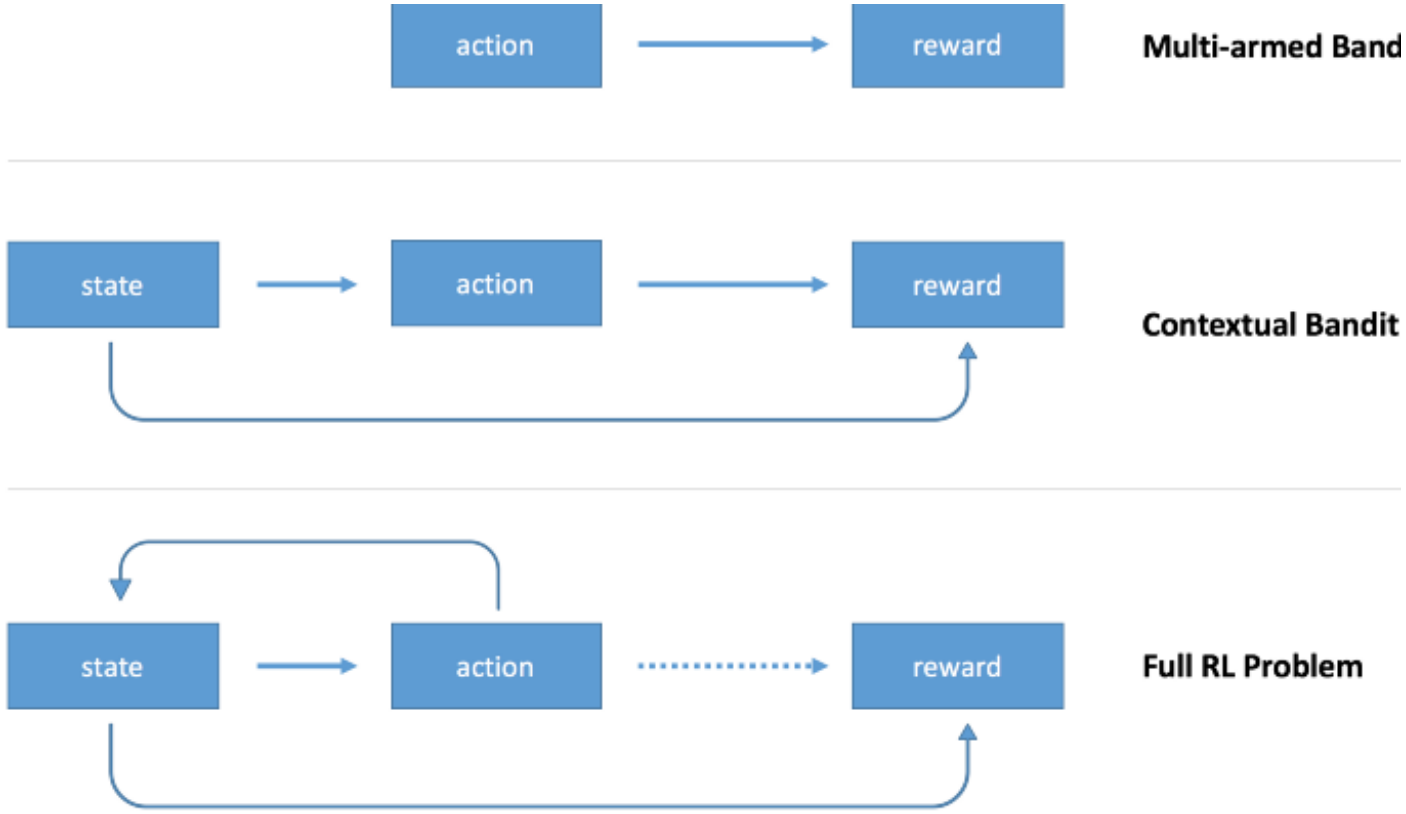
Tianjin University of Technology, City University of Hong Kong, WHUT, TJUT

MSST 2019

Background

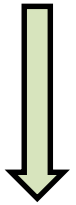
- What is Reinforcement Learning?
  - What is Muti-armed bandits?
    - Exploration and Exploitation
  - What is Contextual bandits?



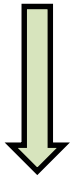


Problem

- Chunk-lookup disk bottleneck problem
  - Fingerprint--->storage position



  - Lots of data will have lots of fingerprints
    - 10PB data, 8kB chunk, 16bytes fingerprint, 99% deduplication ratio
    - 125T chunks * 16 bytes * 0.01



  - Impossible to store fingerprint in RAM
  - Low speed to look up fingerprint in Disk.

Previous approach

- Sample the index of chunk fingerprints in memory.
- Break the data stream into segments, chooses a few of the most similar segments for deduplication.

Motivation

- Memory overheads directly depend on sampling ratio for previous methods. Sampling ratio also influences deduplication ratio.

- Mapping relationship of sampled fingerprints and segments are rather static in previous methods. It is not suitable for dynamic data flow.

- Similarity-based deduplication often uses Top-k to choose a segment to prefetch into cache among many candidates in order to check a chunk duplicate or not. Thus it greatly influences performance how to select a segment into cache.

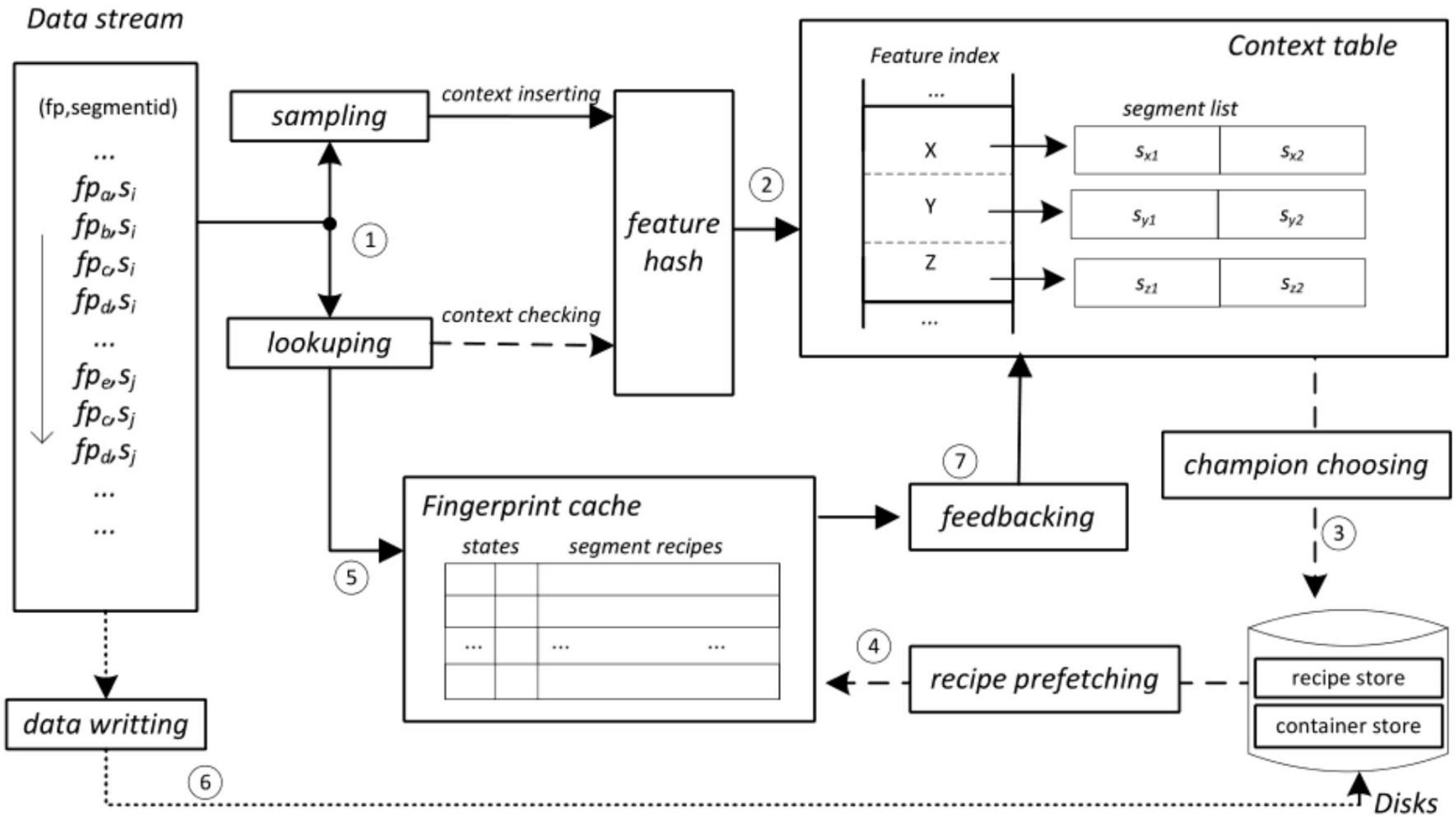
High level idea

- Formalize prefetching problem in reinforcement learning framework and validate its effectiveness by experiments with various real world datasets.

Rationality

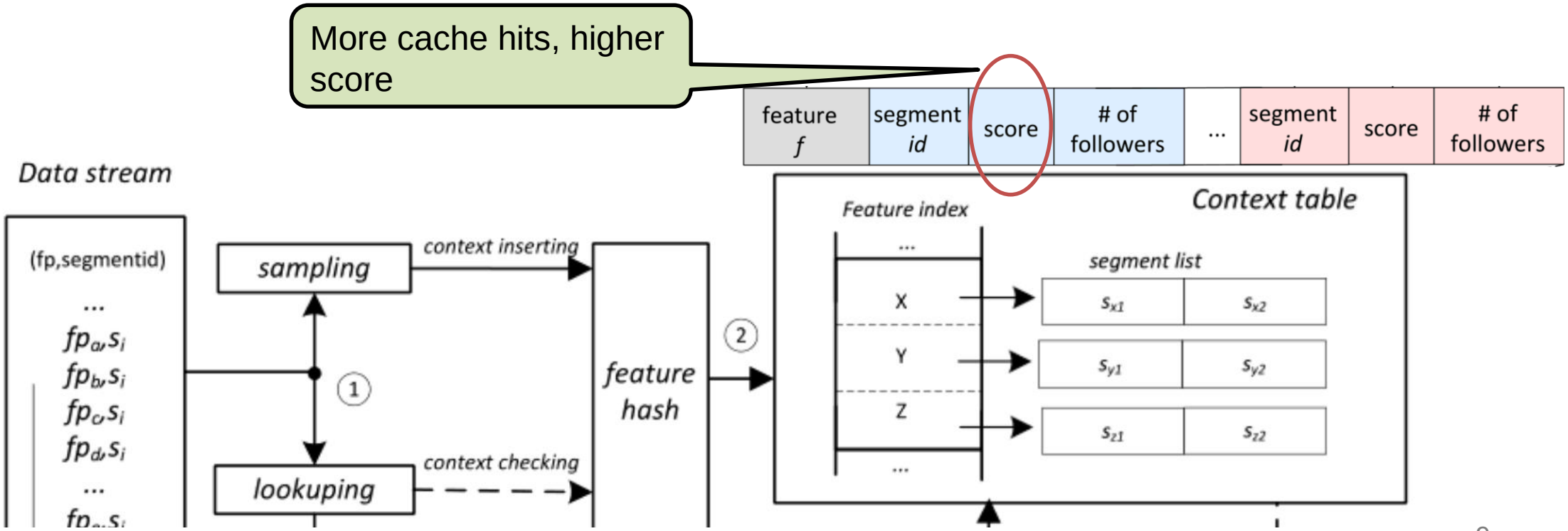
- Define two segments are similar if they share a number of the same chunks.
  - a certain number of fingerprints in a segment recipe are sampled to represent the corresponding segment, called **features** of a segment.
- Action: through features to find most similar segments. Putting their recipes into cache for deduplication.
- State change: change number of segments using for deduplication.
- Reward: cache hit ratio as a reward to influence action choose.

Architecture

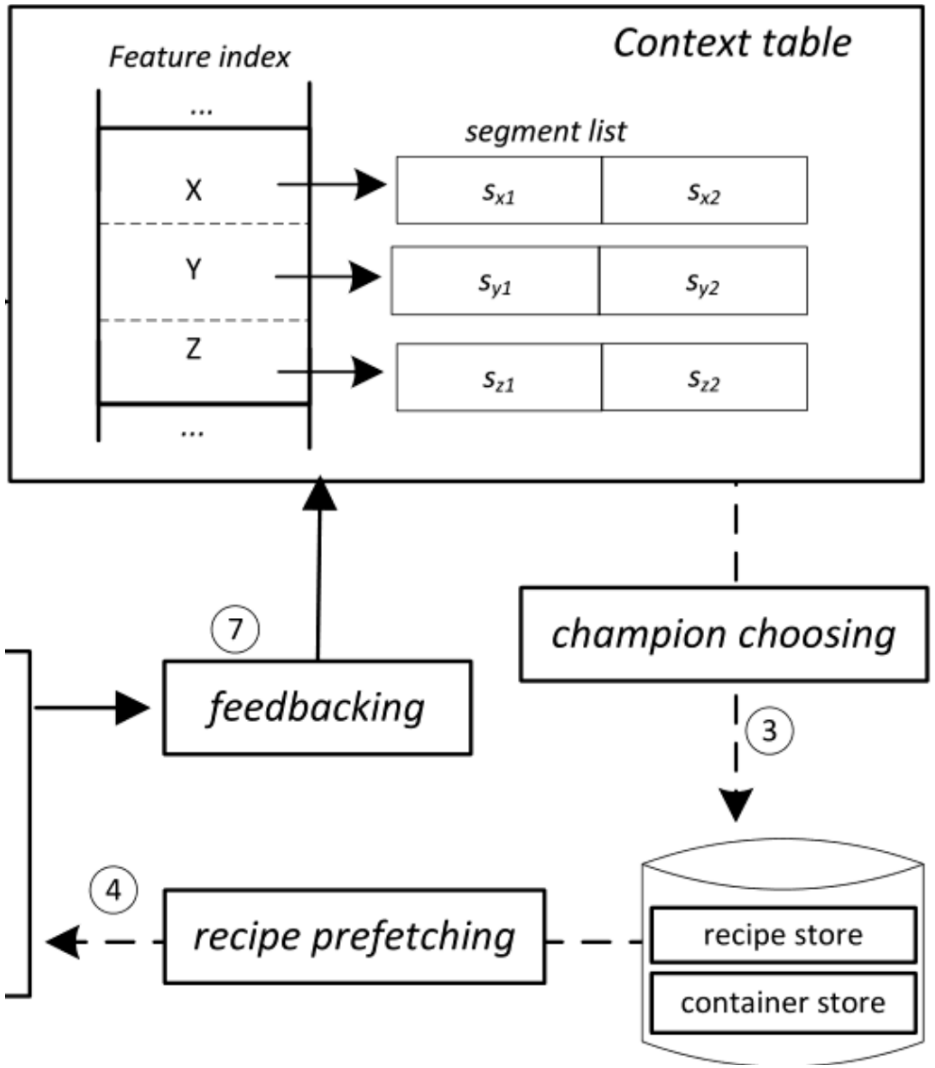


Feature Sampling

- Select the first fingerprint from every 2^n fingerprints in a segment.
- Choose minimal fingerprints by a specific comparison rule.
- Each feature corresponds to an entry in the context table.
- Each segment list has no more than K, FIFO or minimal policy for update.



Champion Choosing

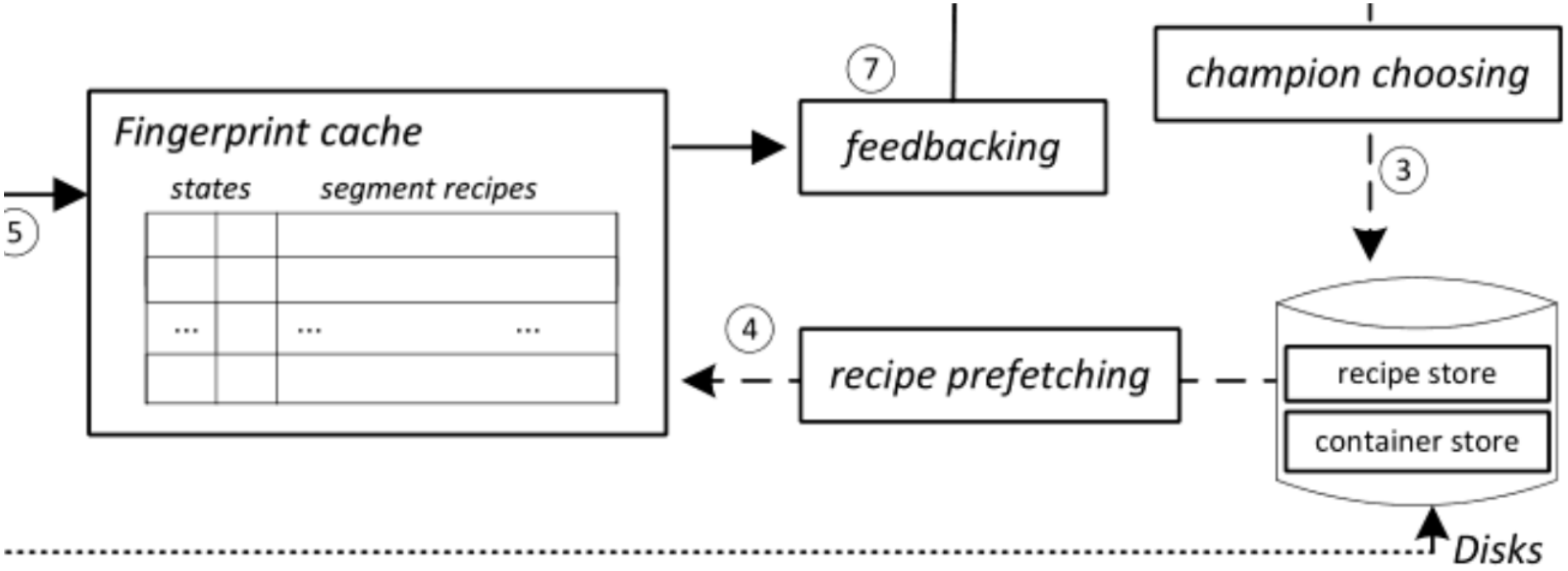


- Chose the most similar segments. (It is costly to load all of those segments into memory and these segments are much similar due to locality)
- Firstly choose the m candidates through features, then chose one of candidates.
- Method: ε-greedy
  - It may choose a segment with the higher score with 1 – ε probability (i.e., exploitation of the past).
  - ε = 0, random chose.
  - Better balance between exploitation by selecting a segment known to be useful and exploration by selecting a segment whose usefulness is unknown but which might provide a bigger reward and thus expand the known space.

Cache prefetching

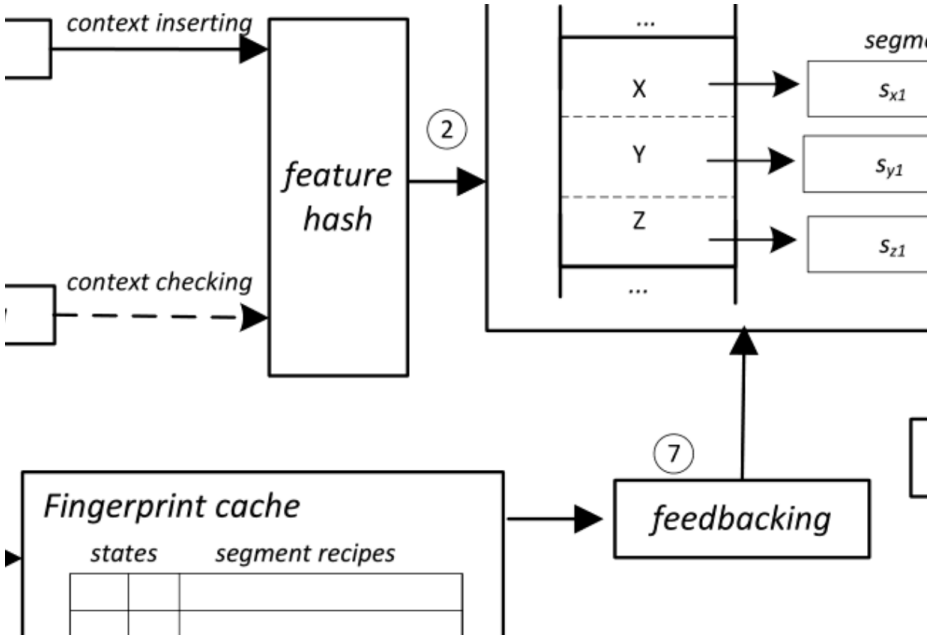
- Observation: similar or identical segments may appear in approximately the same order in a data stream or across multiple streams at a high probability.

- After champion is chosen, prefetch the next n segments' recipe into fingerprint cache



Reward feedback

- During deduplication process, if fingerprint cache hits, the corresponding segment's score update.
- Once fingerprint cache updates, it needs to update the score of segments in cache to context table.



Experimental setup

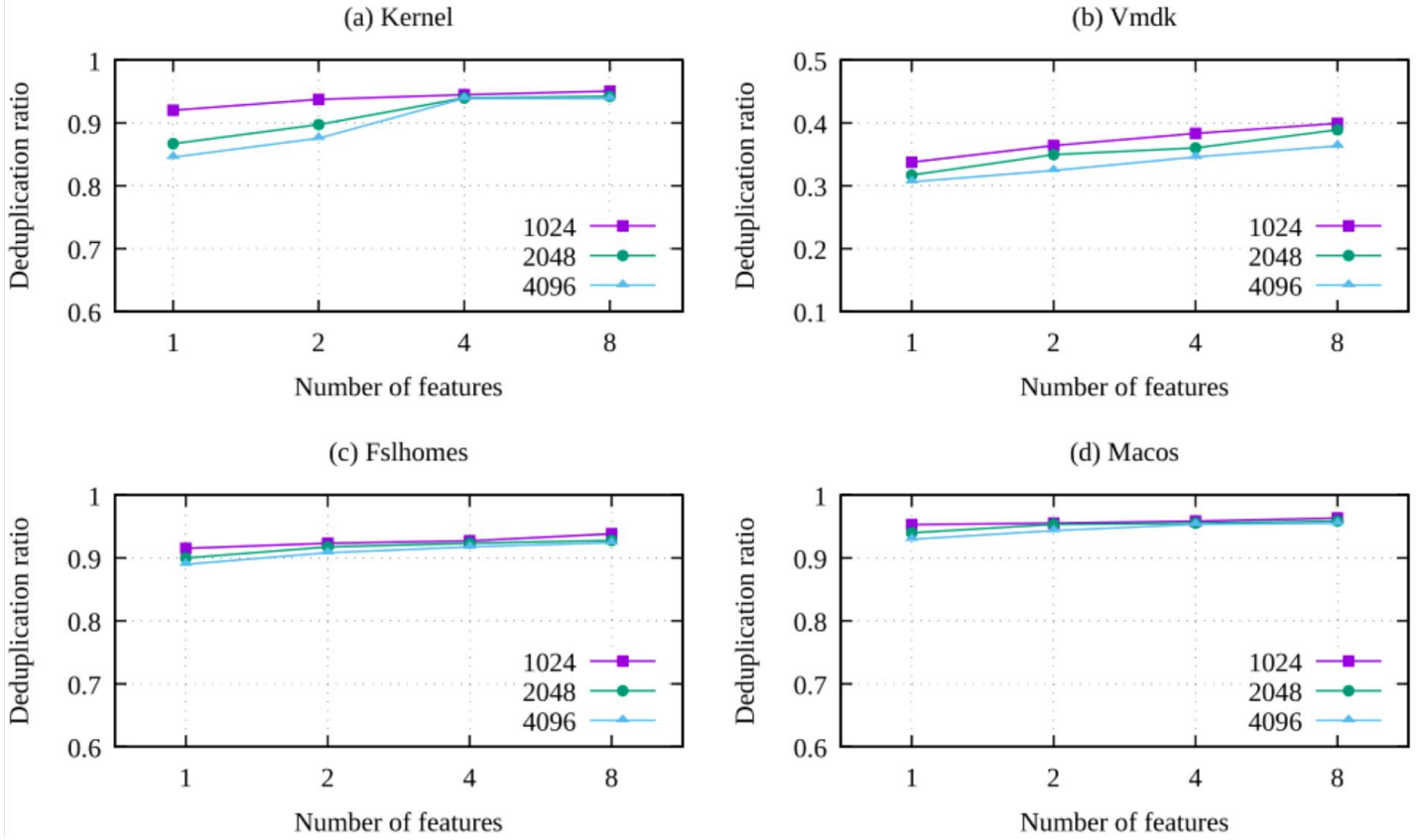
- Implemented based on platform called Destor, which supports a framework of the deduplication pipelining procedure.
- TTTD chunking, SHA-1 hashing, CDS segmenting and minimum sampling features.
- Running in Ubuntu, a quad-core CPU running at 2.4GHz with 4GB RAM and two 1TB 7200rpm hard disks
- Datasets:

Dataset name	number of version	Total size	Dupliation ratio
Kernel	115	24.9GB	95.06%
Vmdk	110	180.9GB	42.14%
Fslhomes	126	4.378TB	95.22%
Macos	18	1.628TB	96.69%

- Kernel: kernel sources of 155 versions collected from the Linux. Most files have small sizes; neighboring versions often have the temporal locality property.
- Vmdk: images of different Linux distributions, such as Ubuntu, CentOS. each image file size is large and images of different operating systems have little duplicate data.
- FLShomes: snapshots of students' home directories. Including different types of file. Use trace data of 9 users in 14 days from Sep. 16th to Sep.30th,
- Macos: collected on a Mac OS X Snow Leopard server running in an academic computer lab. Use its snapshots of selected 18 days at the beginning of year 2014.

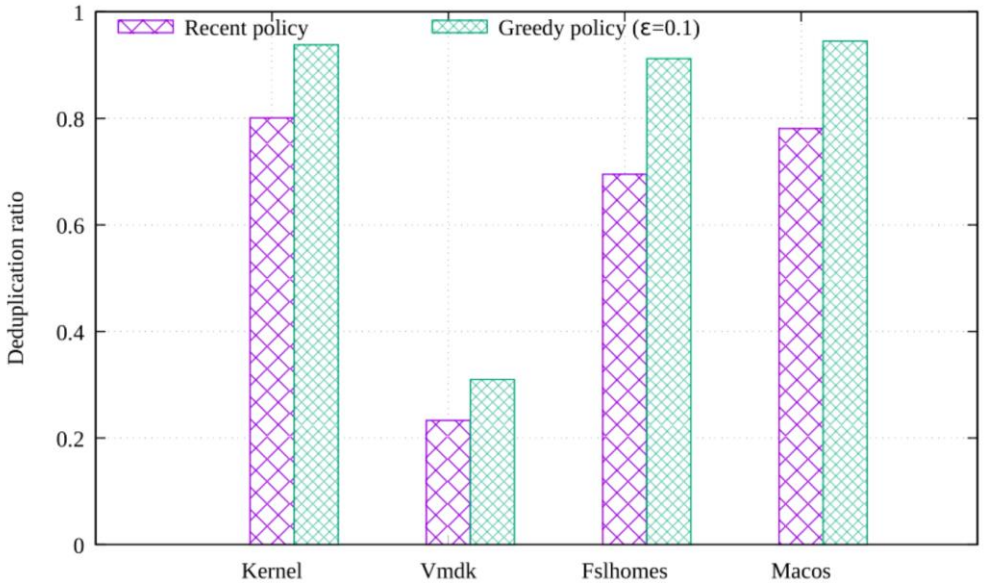
Evaluation

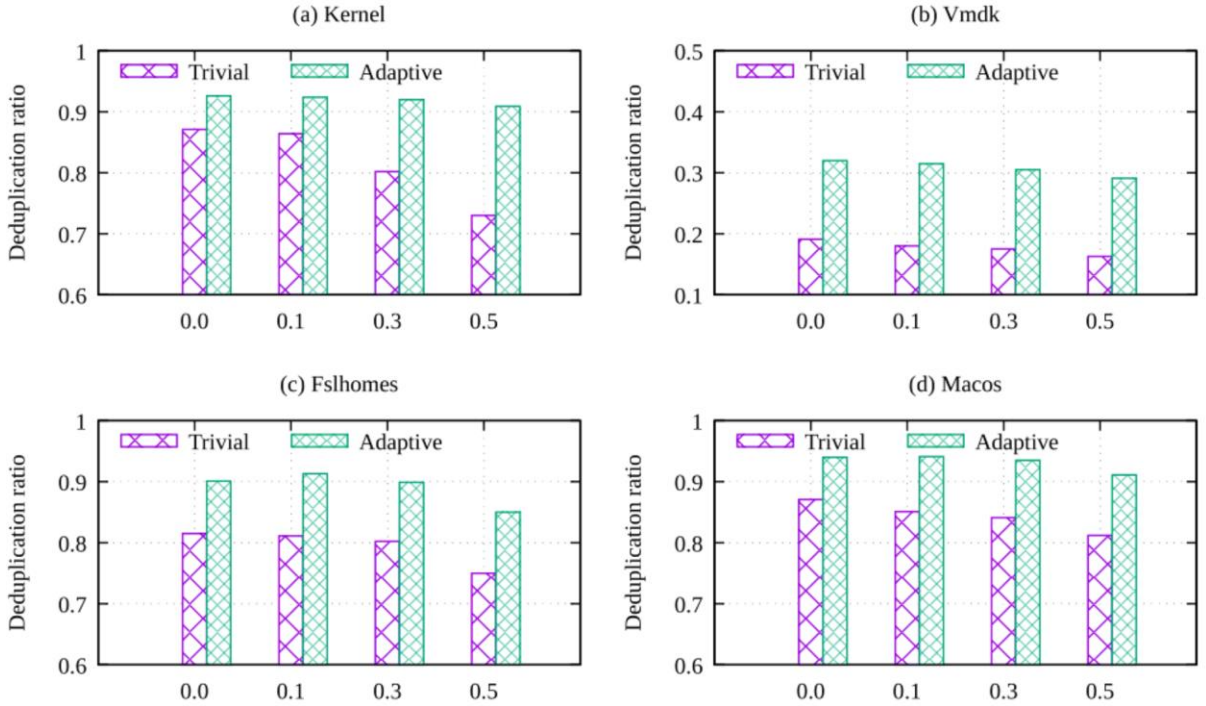
➢ Impact of feature sampling



- Deduplication ratio increases as sampling ratio increases and as the segment size increases. Small segment size results a high deduplication ratio. More features, higher deduplication ratio.
- Trend that the deduplication could converge to the optimal when the sample number is large enough.
- Set the segment size to 1024 and sample 1 feature per segment by default in the following experiments. (reason in paper: deduplication ratio with 1 sampled feature is much closer to those with 2,4,8 sampled features.)

Impact of the champion choosing



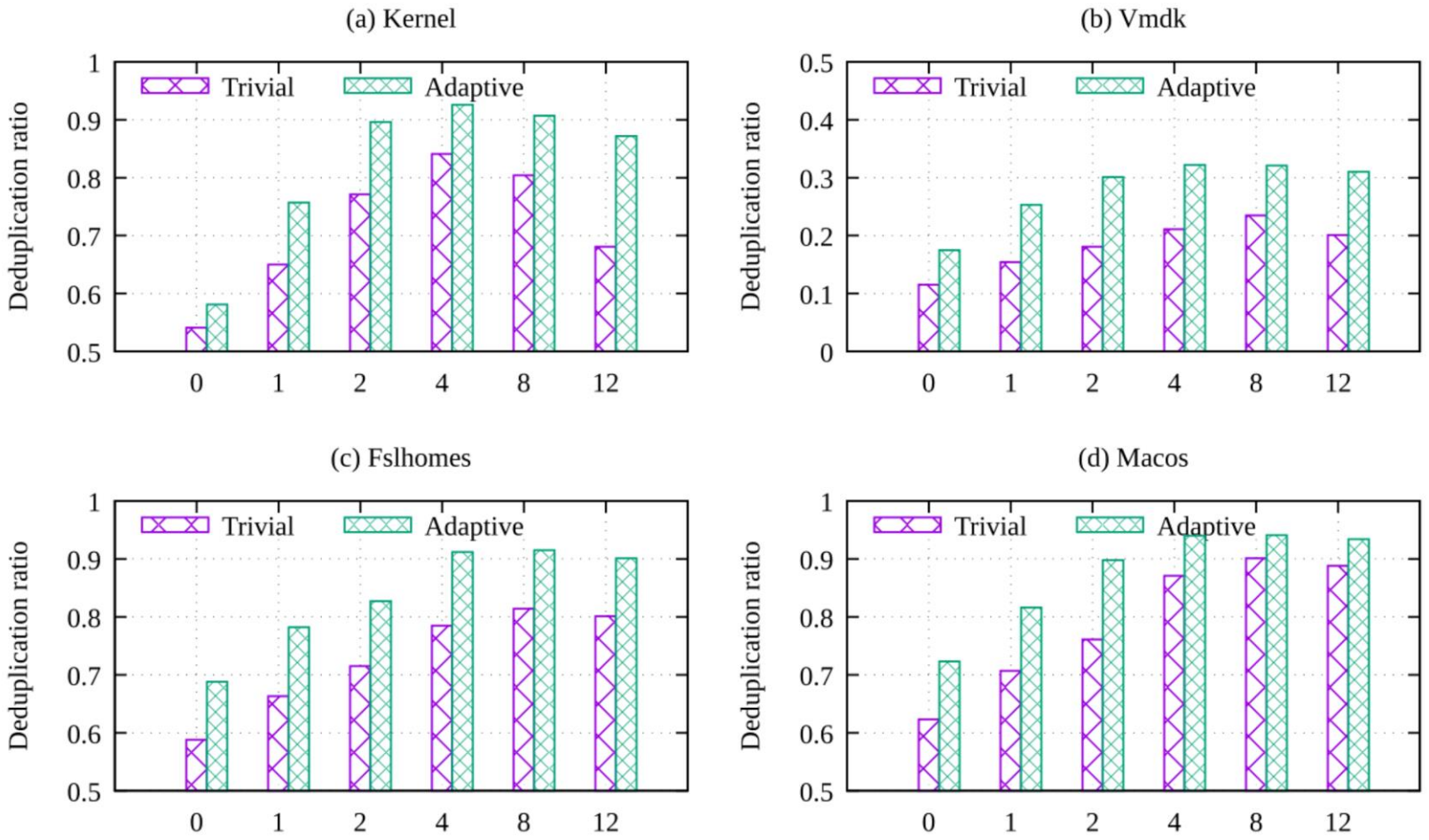


- Greedy is obviously better than Recent.

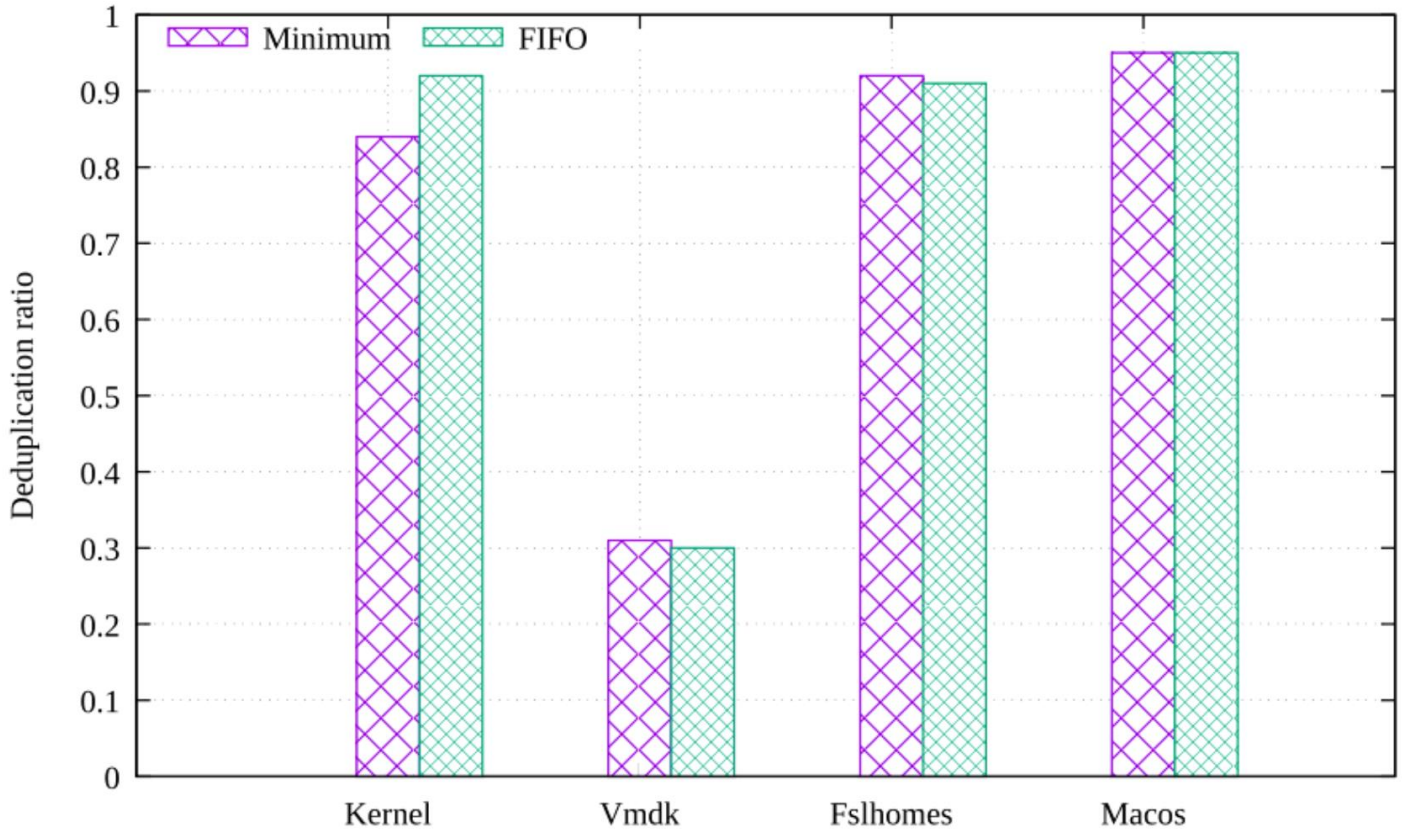
- Trivial way prefetches 4 next segments while adaptive way prefetches a dynamic number which is initialized as 4.
- Smaller ε value results in higher deduplication ratio. That's to say, learned score reflects positive: a segment with higher score could help detect more duplicate chunks.

Impact of the default n followers

- Different default n values affect deduplication for the adaptive prefetching of LIPA.
- If n is too large, some non-useful followers waste some cache space; otherwise, some useful followers are not cached

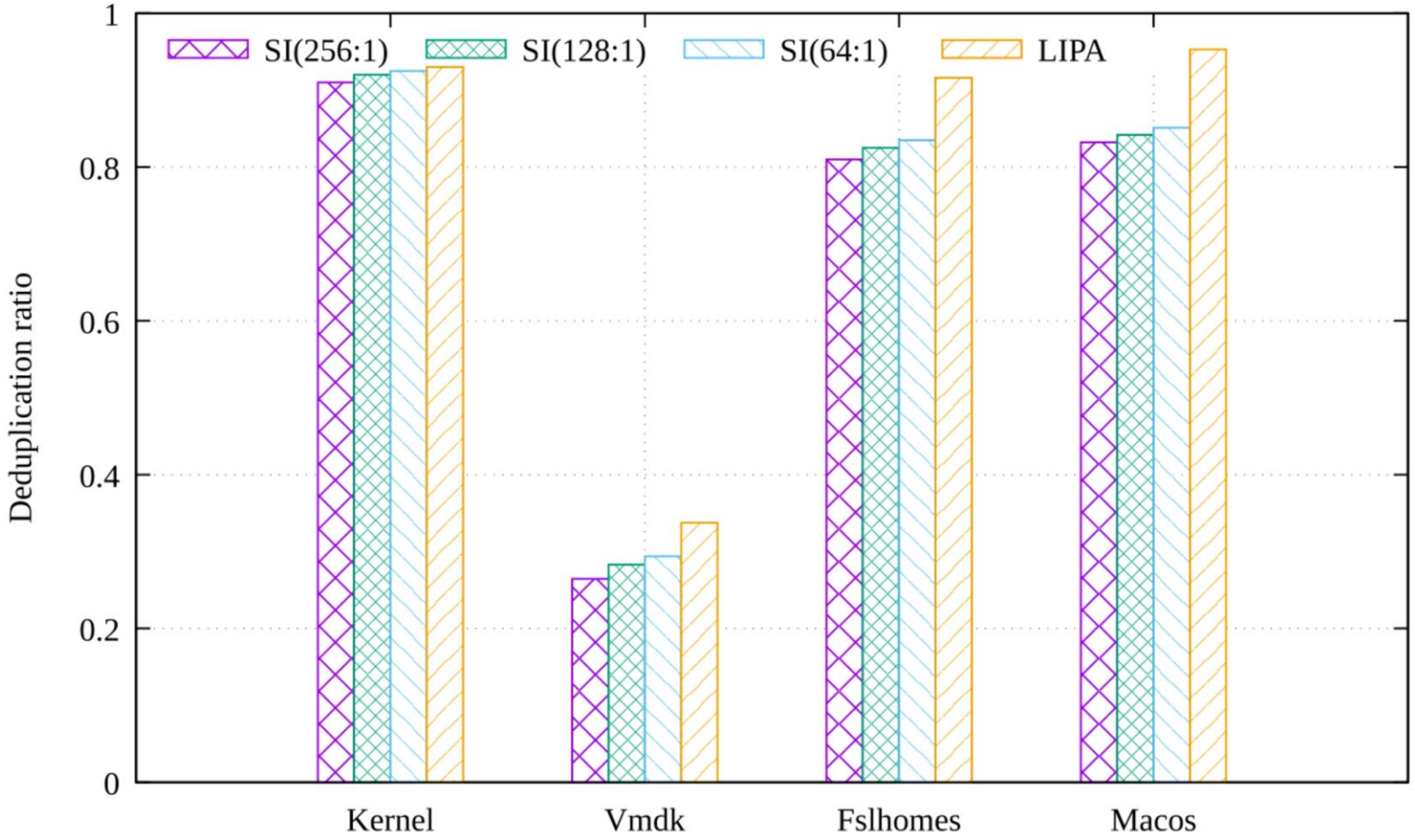


Impact of context table maintenance



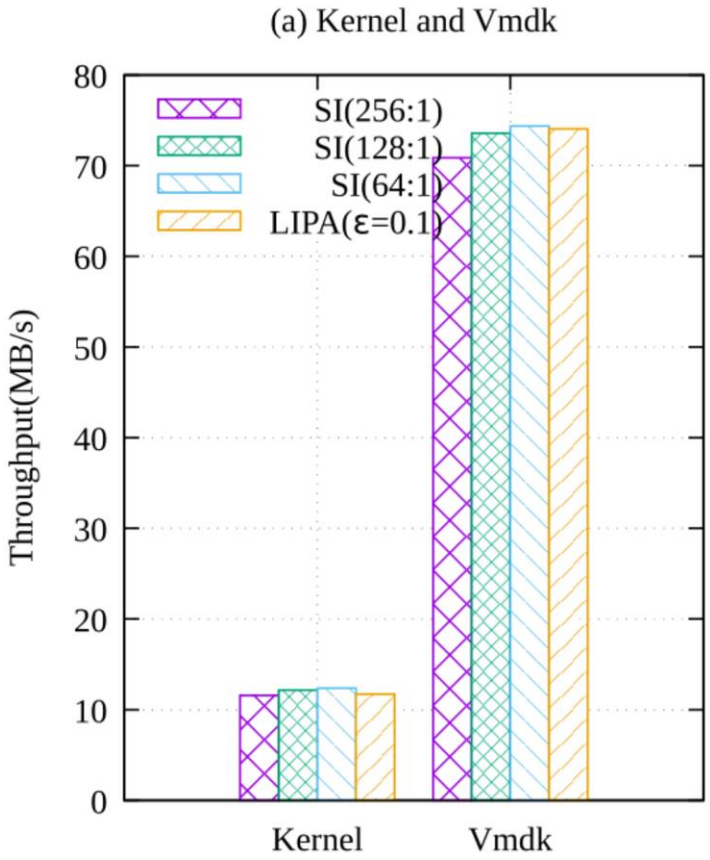
FIFO is much better.

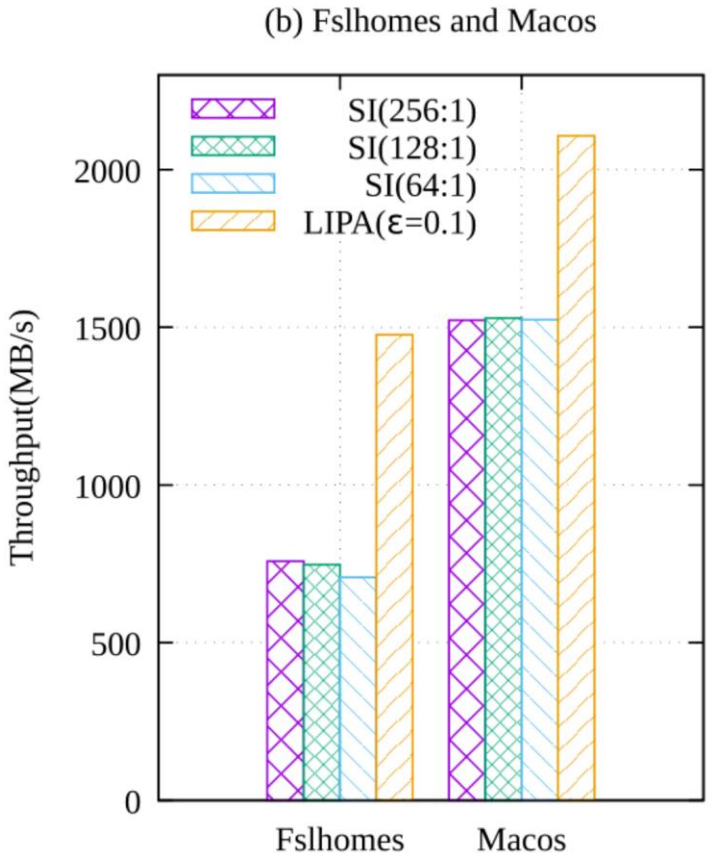
➢ Comparision in deduplication ratio between LIPA.



- **LIPA increases the deduplication ratio by 4% ~ 10% than the sparse indexing.**

throughput





- LIPA and sparse indexing have almost same throughput for Kernel and Vmdk, which are is dominated by small files and Vmdk has much less redundant data to remove.
- For the other two datasets, LIPA achieves much better throughput Two reasons: one is LIPA reduces the accesses of on-disk index and the other ist much more duplicate data identified doesn't need to store.

Conclusion

- Good example applies reinforcement learning to deduplication.
  - Update the association relationship between features and their corresponding segments by a feedback mechanism.
  - Dynamically adjust cache mechanism to improve the performance of deduplication system.