PLC-Cache: Endurable SSD Cache for Deduplication-based Primary Storage

Jian Liu, **Yunpeng Chai**, Yuan Xiao
Renmin University of China
Xiao Qin
Auburn University

Speaker: **Tao Xie**

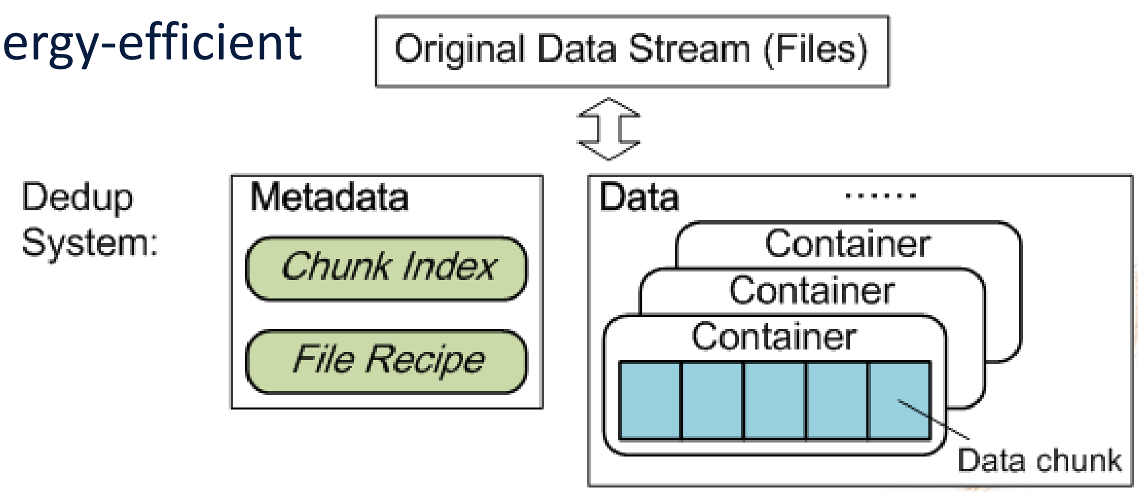
MSST, June 5, 2014

Deduplication

- Age of Big Data
  - Bad news:
    - Goble data will be more than 40 ZB in 2020 [IDC];
    - Capacity and price of disks changes slowly after reaching TB-level
  - Good new:
    - 75% data is redundant [IDC]
- Dedup improves space-efficiency
  - Secondary Dedup: backup, archive systems (most of previous work on dedup)
  - Primary Dedup: online storage systems (e.g., email, database, web, etc.) [*iDedup@FAST'12*, ZFS, SDFS, LessFS, LBFS]

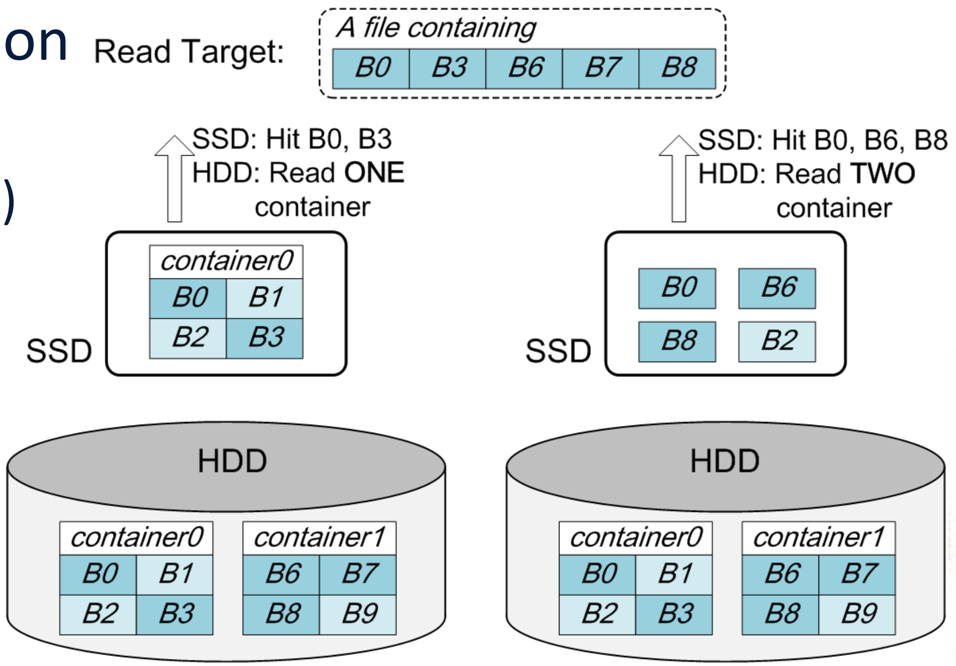
Primary Deduplication

- Reasons of limited performance of Dedup
  - Additional metadata accesses
  - Data Fragmentation on disks
    - Defragmentation:
      - Limited containers of each file [*Capping@FAST'13*]
      - Storing sequence of chunks [*iDedup@FAST'12*]
    - SSD data cache to boost [*SAR@NAS'12*] (focus of this paper)
      - Large, Fast, Energy-efficient



SSD Cache to Boost

- Container-level caching instead of chunk-level caching [e.g. *SAR*]
  - Reading less containers from HDD (as the example)
  - Write amplification reduction
    - Containers size match erase unit of Flash chips (several MB)



(1) SSD caches containers (2) SSD caches chunkss

Challenge of SSD Cache

- Too much writes on SSD cache leads to
  - Low performance caused by request congestion;
  - Importantly, VERY SHORT lifetime of SSD cache

Cache Algorithms	The measured writing speed of SSD	Expected 60GB Intel 520 SSD lifetime	Expected 60GB Intel 910 SSD lifetime
FIFO	56.7 MB/s	7.6 Days	162 Days
LRU	50.7 MB/s	8.5 Days	181 Days
LFU	53.1 MB/s	8.1 Days	173 Days

Intel 520 SSD, 60 GB
TBW: 36 TB (used in experiment)

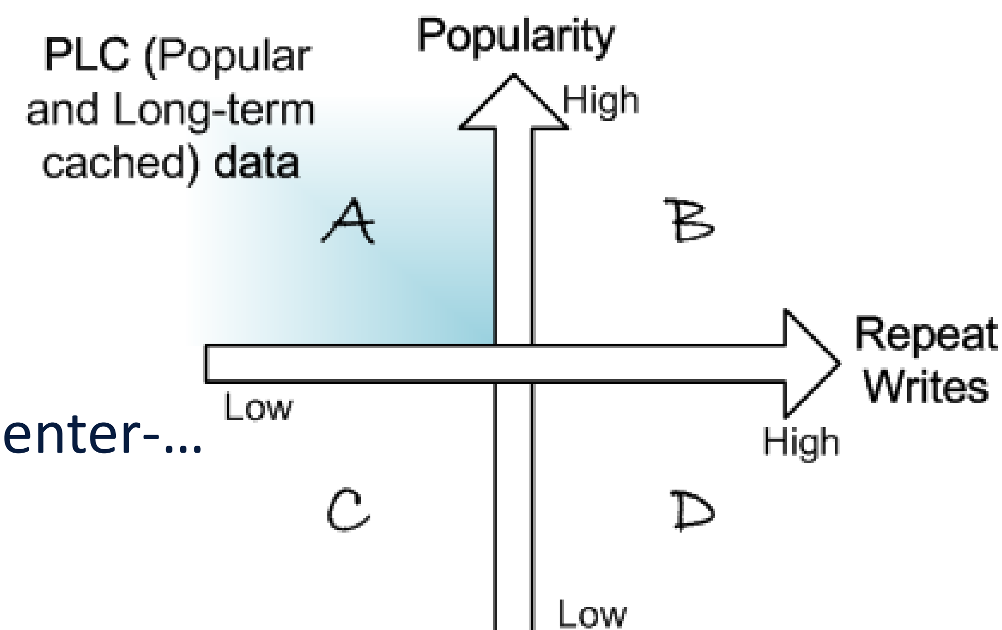
Intel 910 SSD, 400 GB
TBW: 5 PB
(enterprise SSD)

Results are from an experiment based on a dedup system with 60GB SSD cache

Analysis

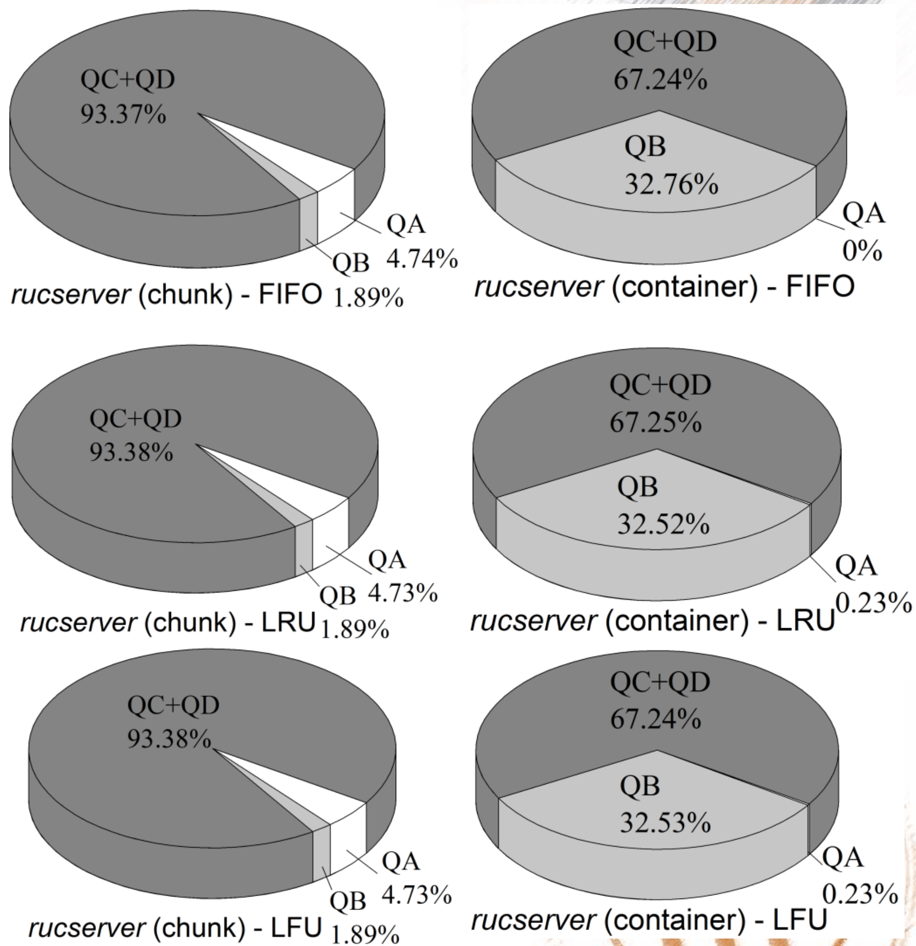
- Analyze the composition of written data set
  - Is all the written data NECESSARY?
- Four quadrants
  - Caching target:
    - PLC data (Best): QA
  - NOT necessary
    - QC/QD: few benefits
    - QB: repeat writes

    Loop: evict-enter-evict-enter-…



Analysis (Cont.)

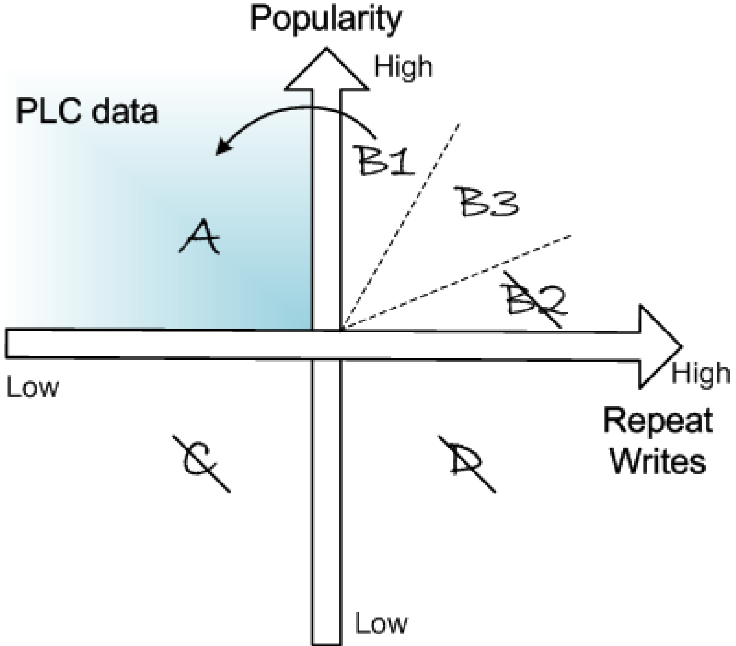
- Trace Analysis
  - PLC data (QA): low percentage
  - Chunk -> container:
    - QA (PLC data) ↓
    - QC+QD ↓
    - QB ↑
  - FIFO, LRU, LFU have similar results



Other real-world traces lead to similar observations

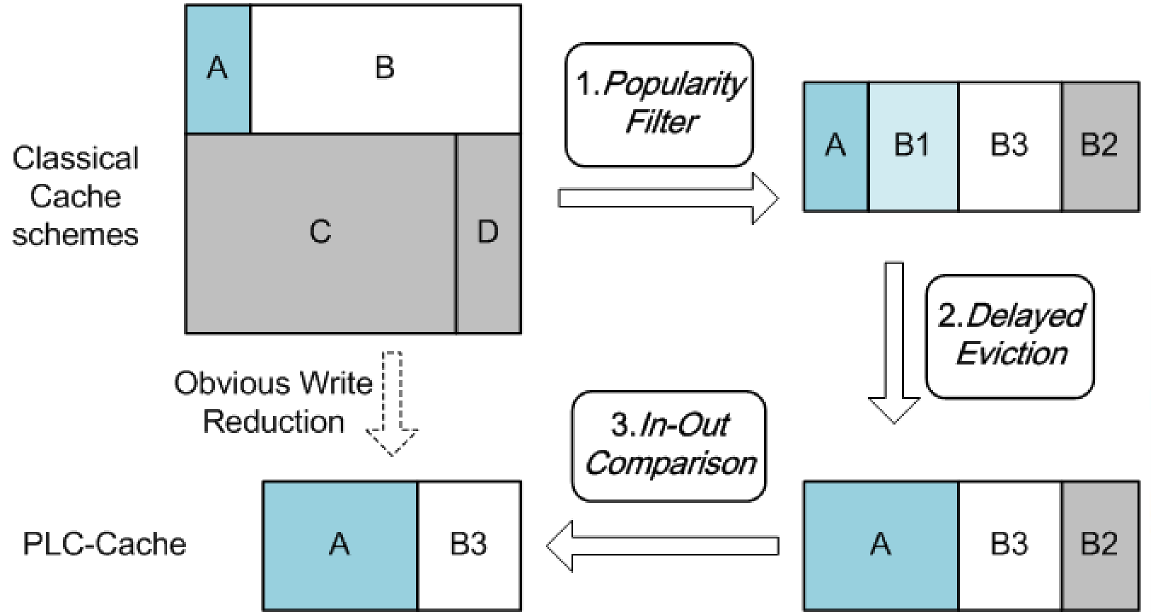
Basic Idea

- Strategies to cache PLC data
  - Exclude QC and QD data with low benefits
  - Convert QB1 data with long-term hot potential to PLC data
  - Exclude QB2 data with similar popularity with cached one



PLC-Cache

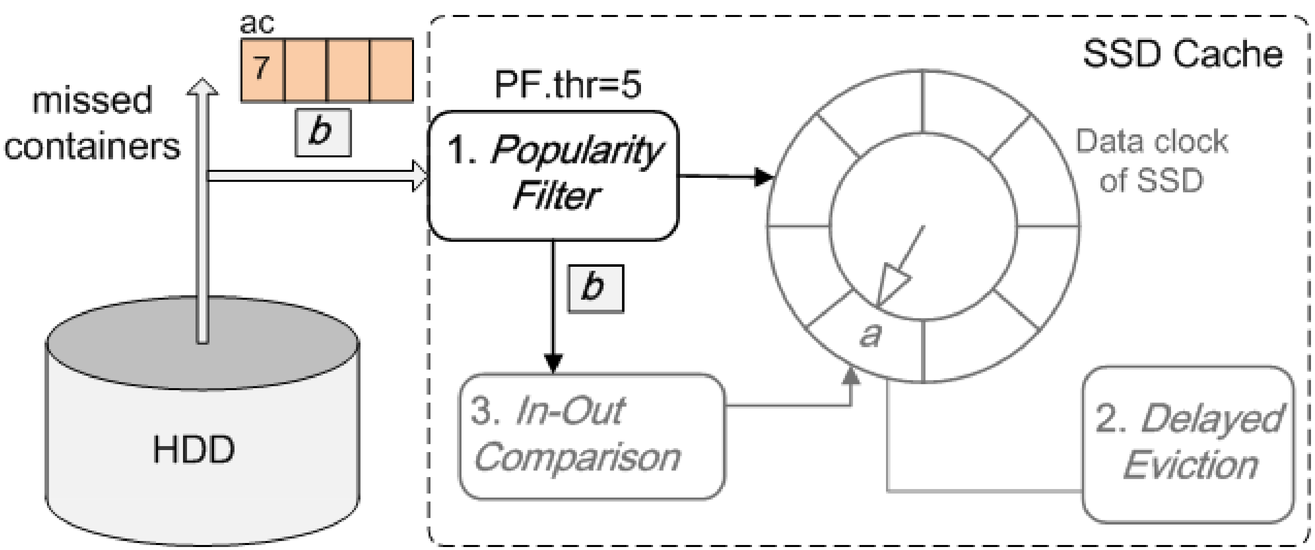
- Three modules (steps)
  1. ***Popularity Filter*** excludes QC and QD data;
  2. ***Delayed Eviction*** converts QB1 data into PLC data
  3. ***In-Out Comparison*** excludes QB2 data



Example (*Popularity Filter*)

- A missed container ***b***
  - ***b***.*ac* < *PF.thr*: not replace
  - ***b***.*ac* >= *PF.thr*:
    - SSD Cache is not full -> ***b*** enter SSD cache;
    - SSD cache is full -> …

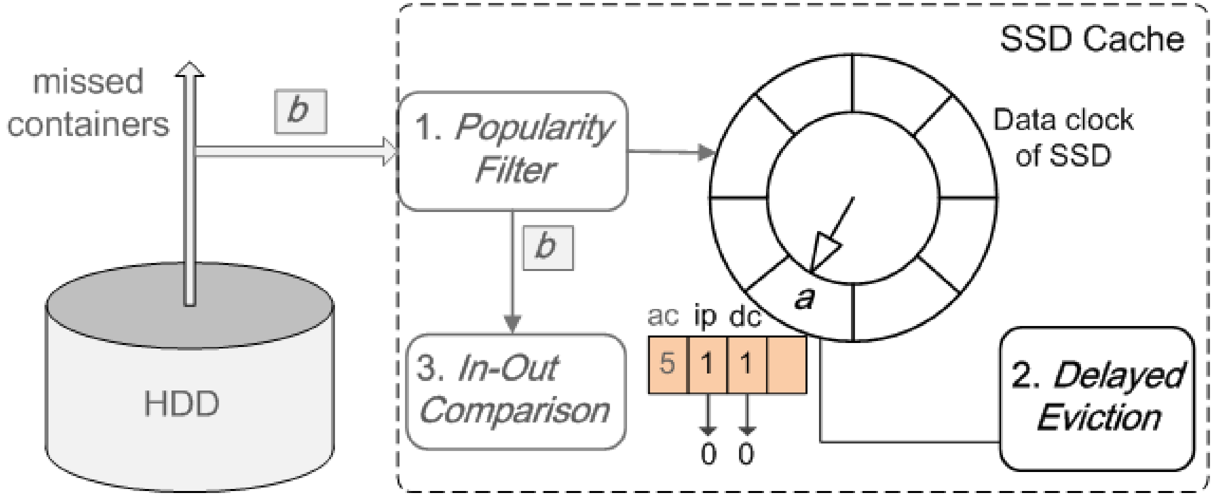
Container's properties:
ac*: access_count*



Example (*Delayed Eviction*)

- Selecting victim in SSD cache
  - Pointer scans, ***a***.*ip*--
  - ***a***.*ip* == 0:
    - ***a***.*dc*--;
      - ***a***.*dc* > 0 -> not replace
      - ***a***.*dc* == 0 -> …

Container's properties:
***ac**: access_count*
***ip**: inter_priority*
***dc**: delay_count*
+1 when hit (ac, ip)



Example (*In-Out Comparison*)

- Comparison between container ***b*** and ***a***
  - (***b***.*rc* > ***a***.*rc*) or (***b***.*ac* > IOC.*m*****a***.*ac*)

    -> replace
  - else -> not replace

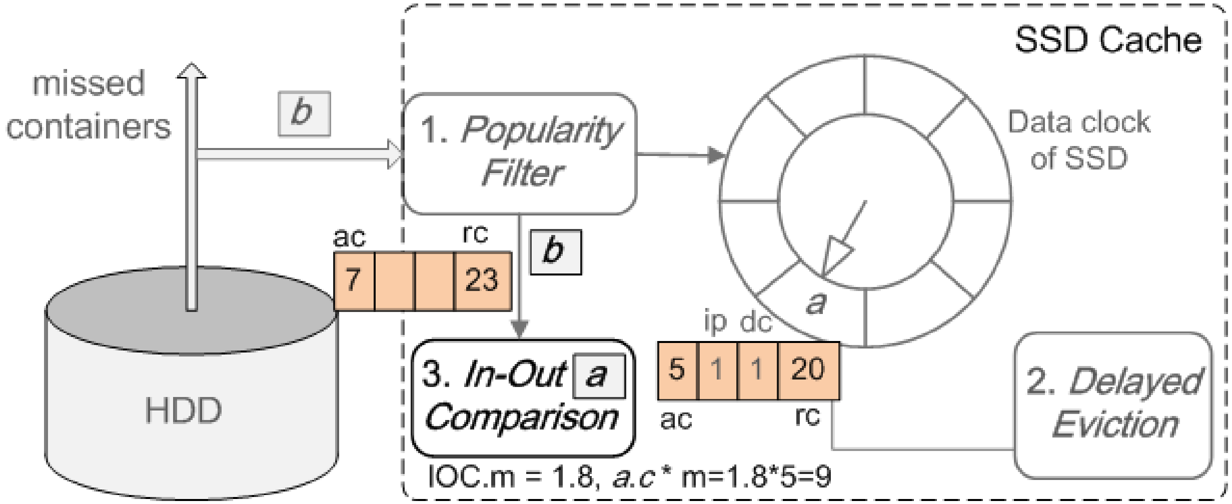
Container's properties:

ac*: access_count*

ip*: inter_priority*

dc*: delay_count*

rc*: reference_count* } Related file count of dedup

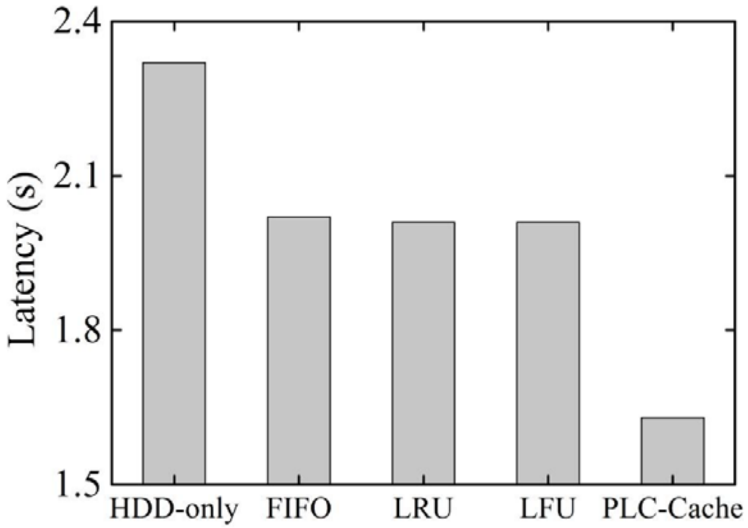


Evaluations

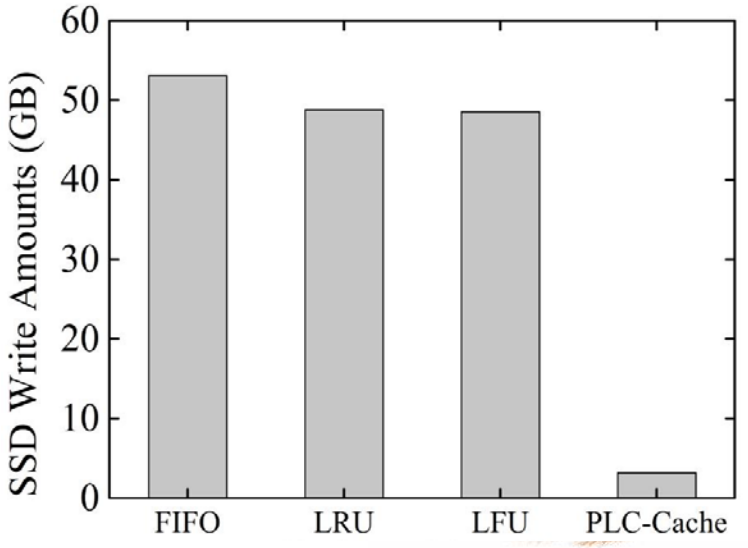
- Experiment Setup
  - Platform
    - Primary dedup enhancement based on **Destor** (https://github.com/fomy/destor)
  - Hardware
    - Intel Core(TM) i7-2600 quad-core CPU, 3.40GHz
    - 4GB RAM, 2TB Western Digital hard disk
    - 60GB Intel 520 SSD
  - Data
    - Collected in the laboratory server of RUC, 200GB
  - Access
    - Randomly read 520GB data (Zipf distribution)

Overall Results

- PLC-Cache vs. HDD-only
  - Performance: 41.9%↑
  - SSD: 20% capacity of HDD data

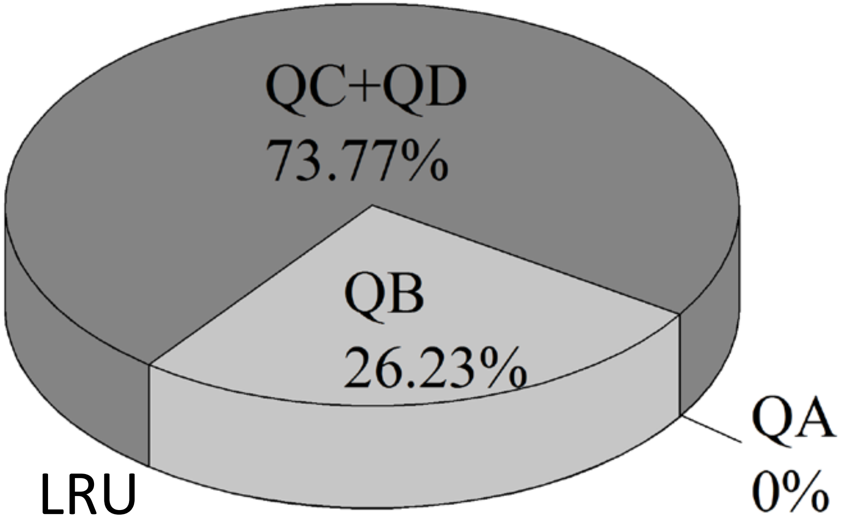


- PLC-Cache vs. traditional Cache
  - Performance: 23.4%↑
  - SSD written amount: >15x↓

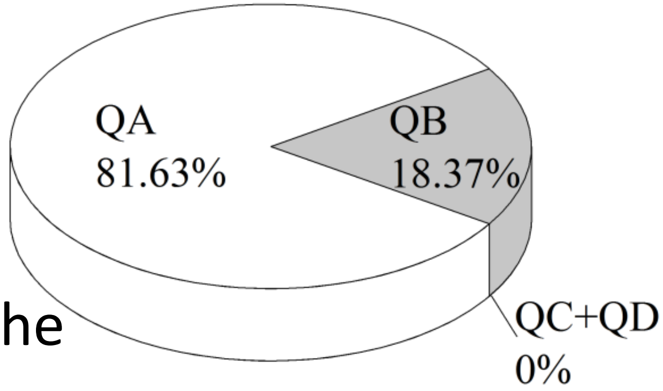


Distributions of four quadrants

- Amount of SSD written data
  - LRU: 48.77 GB
  - PLC-Cache: 3.18 GB
  - **15.34x ↓**
- Four quadrants
  - QA (PLC data): 0% ↗ 81.63%
  - QB: 26.23% ↘ 18.37%
  - QC+QD: 73.77% ↘ 0%



LRU

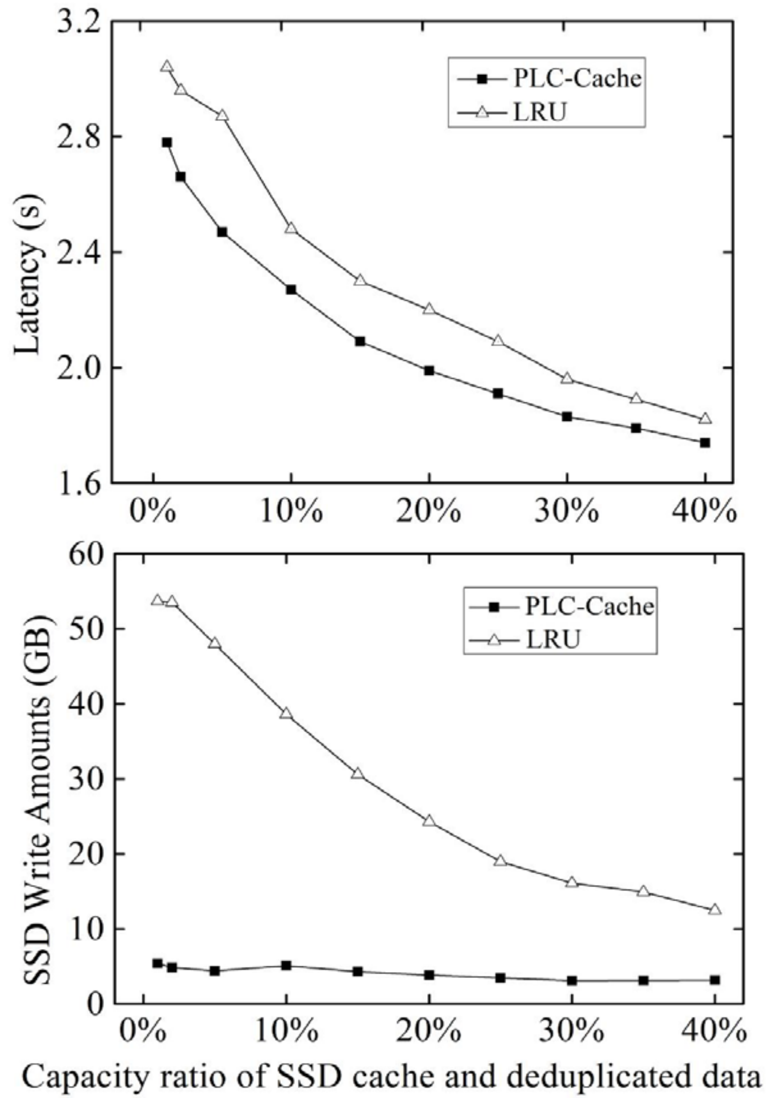


PLC-Cache

Under Various SSD Cache Size

- Performance
  - PLC-Cache always outperforms LRU

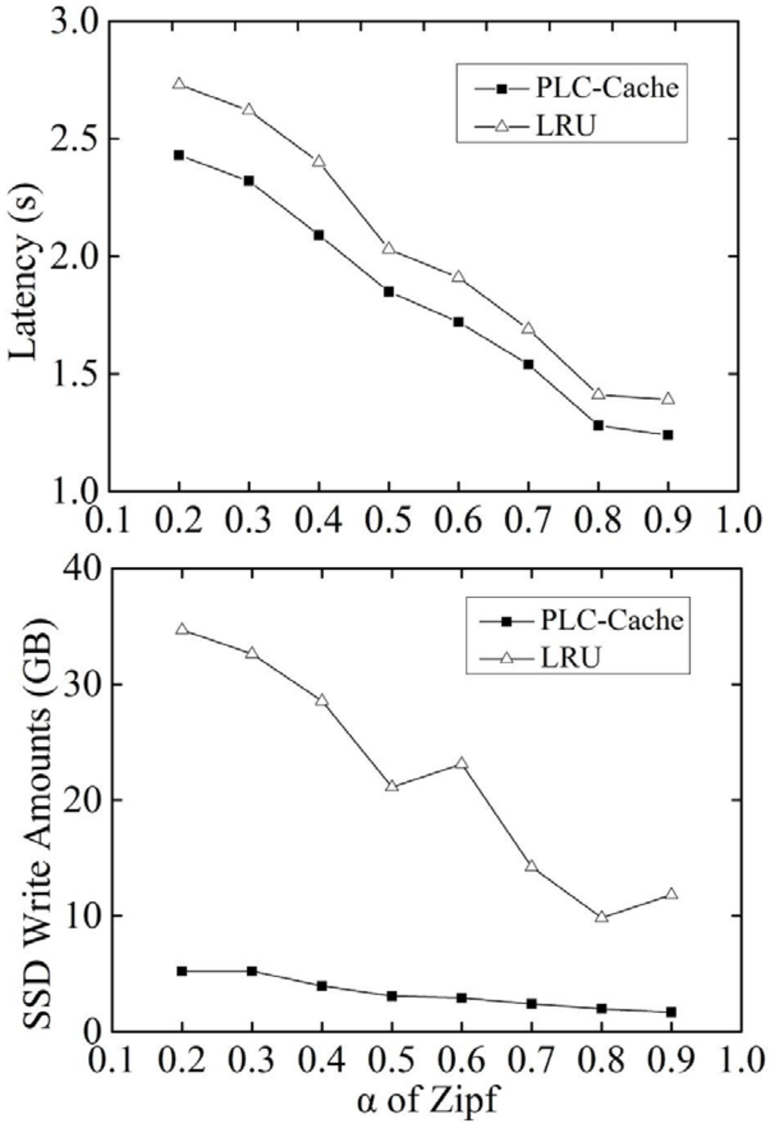
- Amount of SSD written data
  - PLC-Cache is steadily very low
  - LRU writes less for larger SSD, because of high hit rate.



Under Various α of Zipf

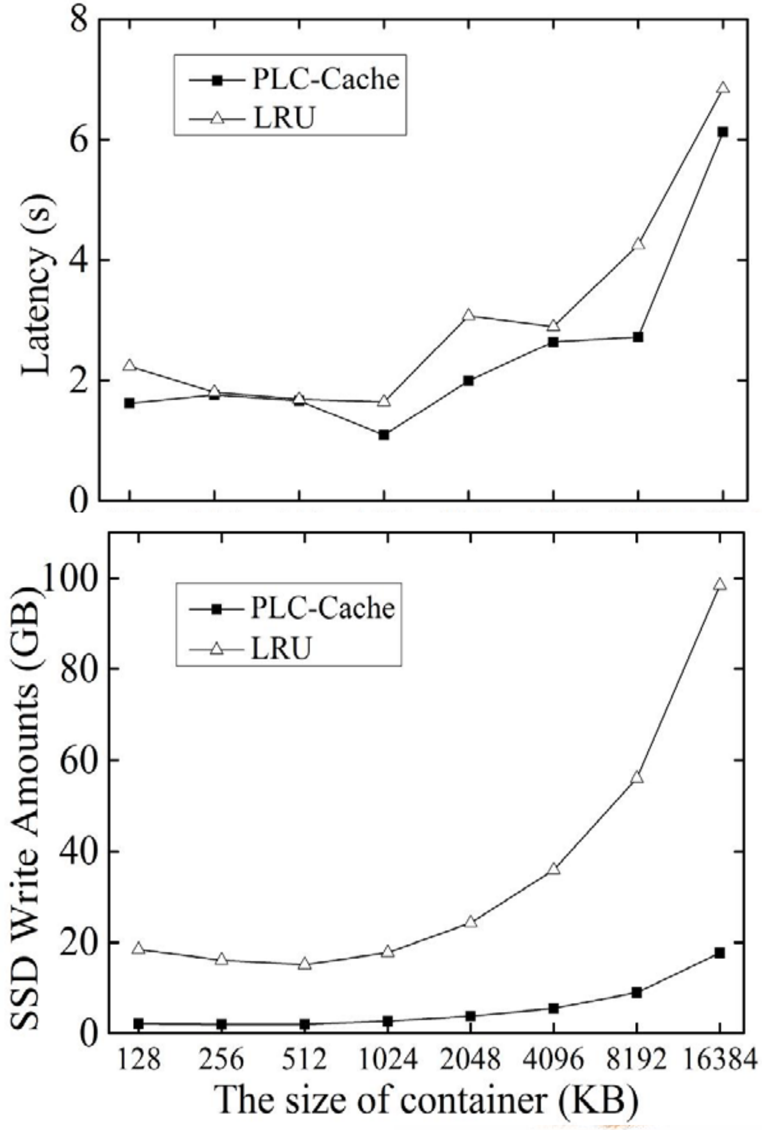
- Performance
  - PLC-Cache always outperforms LRU
  - Higher performance for a larger α (i.e. more concentrated access)

- Amount of SSD written data
  - PLC-Cache is steadily very low



Under Various Container Size

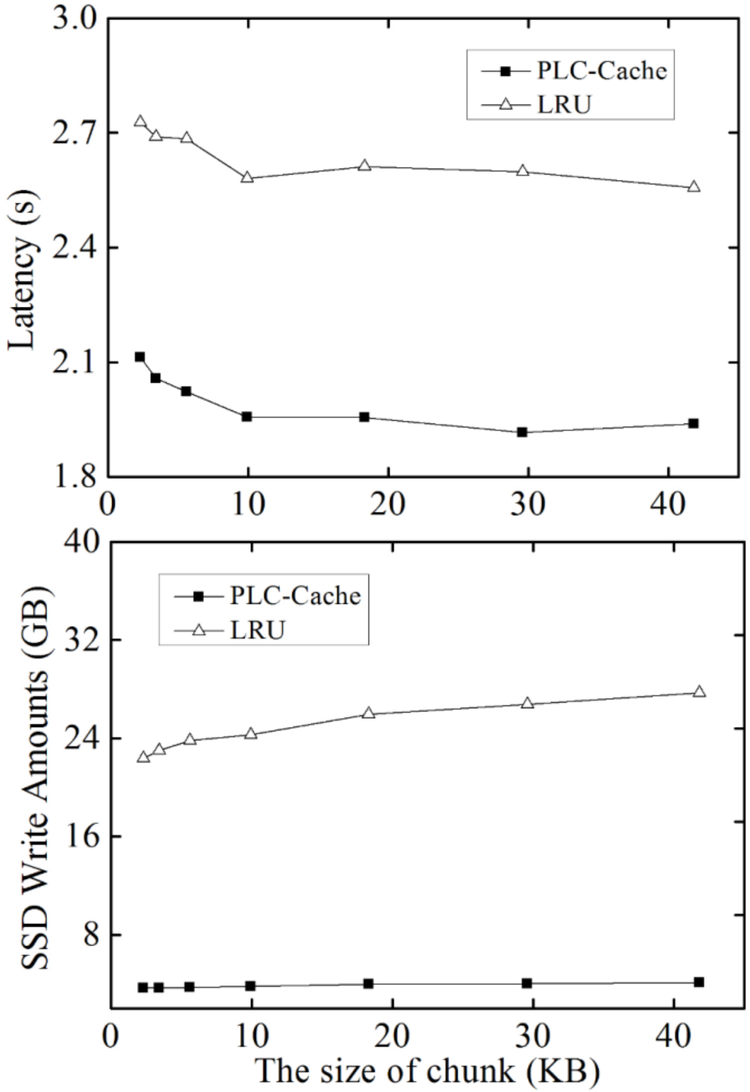
- Performance
  - PLC-Cache is better than LRU
  - Bad performance for too large container
- Amount of SSD written data
  - PLC-Cache is always much lower than LRU
- Container-level caching
  - Reduce erase operations by 54.4% compared with SAR
  - By reading S.M.A.R.T. info of SSD before and after experiment.



Under Various Chunk Size

- PLC-Cache outperforms LRU in both performance and SSD writes.

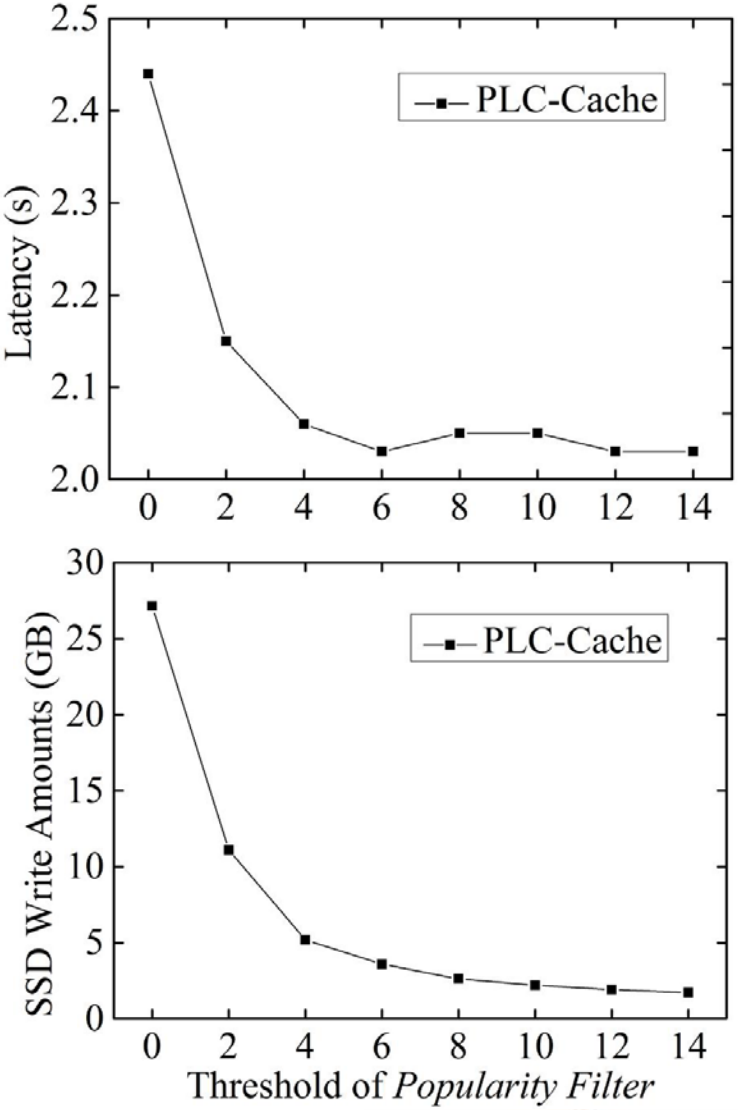
- Chunk size does not have obvious impacts
  - Only too small chunk leads to worse performance, because of more metadata to access for more chunks



Impacts of PF threshold

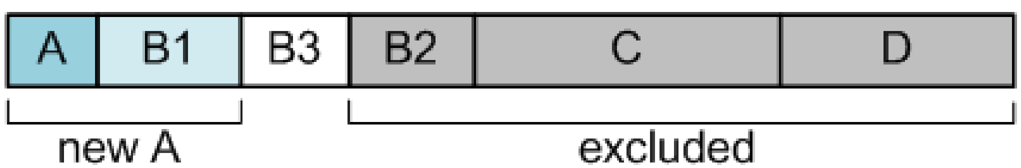
- Low threshold of PF means more QC/QD data enters SSD cache
  - More SSD writes
  - Request congestion

- Performance and SSD writing amount exhibit similar pattern



Summarizing SSD-age caching

- Previous cache algorithms
  - Unlimited write:
    - FIFO, LRU, LFU, MQ, LIRS, ARC, …
  - Simply remove some writes:
    - LARC, SieveStore, EMC FastCache, SUN L2ARC
- PLC-Cache:
  - Comprehensive analysis of cache’s written data set
  - A series method to improve distribution of 4 quadrants



  - Excellent results:
    - performance 23.4%
    - amount of SSD written data 15x

Thank You !

ypchai@ruc.edu.cn