HPDedup: A Hybrid Prioritized Data Deduplication Mechanism for Primary Storage in the Cloud

Huijun Wu, Chen Wang, Yinjin Fu, Sherif Sakr, Liming Zhu, Kai Lu

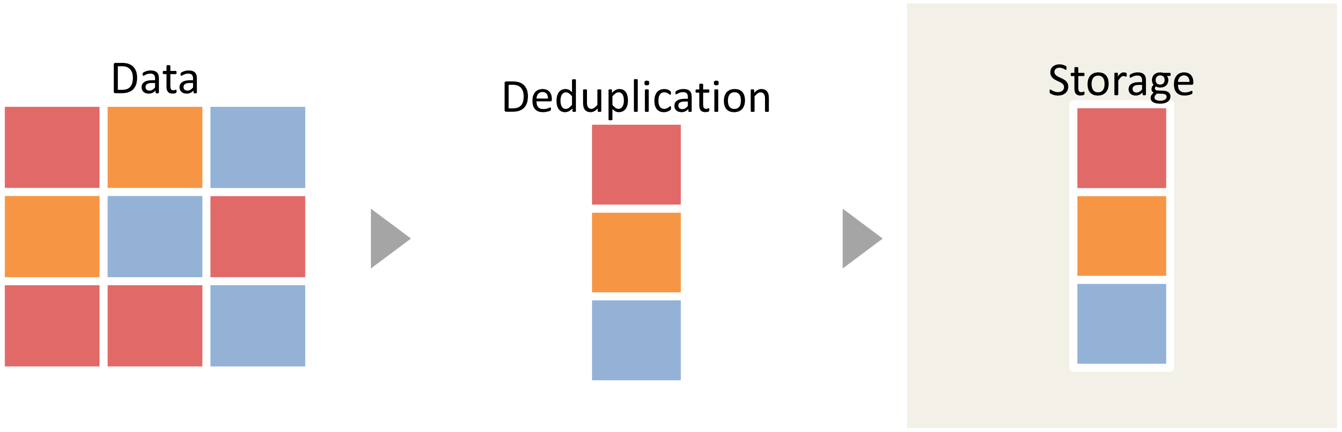
MSST 2017

Background

- **Primary Storage Deduplication**
  - No backup cycle
  - Low latency requirement
  - Deduplication ratio is relative low
- **Two deduplication methods**
  - Inline Deduplication
  - Post-processing deduplication
  - Inline & Post-processing combined

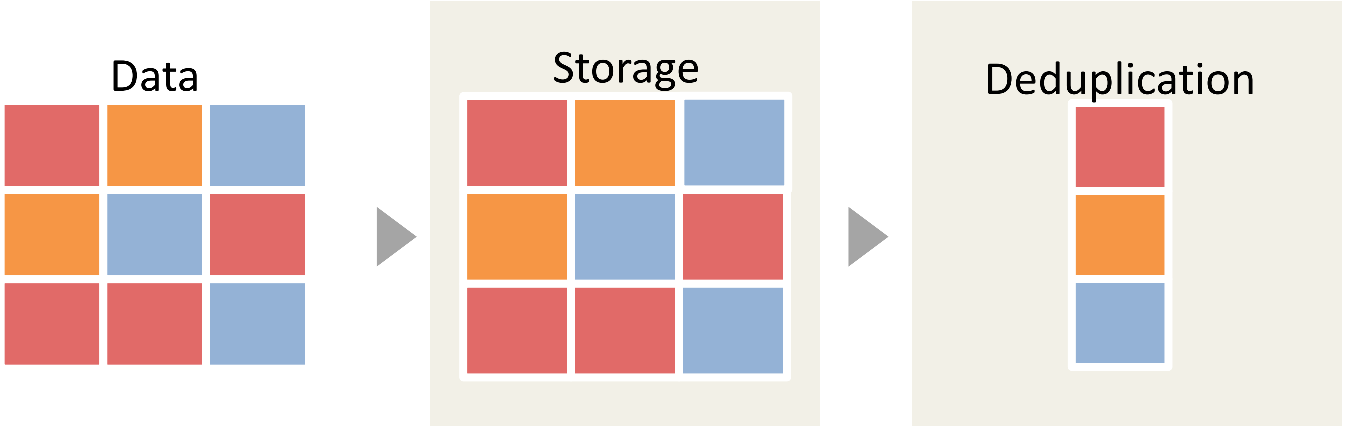
Inline vs Post-processing Deduplication

Inline Deduplication



- Perform on the write path
- Larger latency
- Space saving efficient

Post-processing Deduplication



- Perform during off-peak time
- Lower latency
- High peak space capacity

Motivation I: Deduplication

- **Inline Deduplication**
  - **Bottleneck**: Fingerprint lookup
  - Fingerprint table is large and on-disk fingerprint table incurs high latency
  - In-memory fingerprint cache to perform **non-exact** deduplication
    - Deduplication ratio → depends on cache efficiency → Locality

- **Post-processing Deduplication**
  - Perform **exact** deduplication
  - High peak storage capacity & Wear devices for duplicate writes (SSD)
  - **Resources contention(CPU, RAM, IO)** between post-processing process and foreground process (especially large amount of duplicates)
  - → Inline deduplication can alleviate this situation

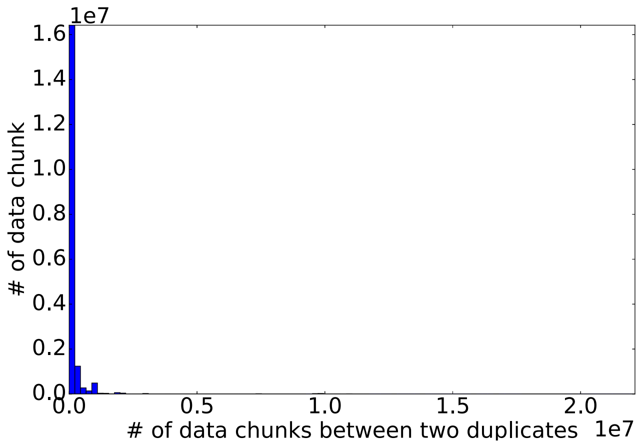
Motivation II: Deduplication in Clouds

- **Deduplication**
  - Impractical to deploy deduplication inside instance of VM or container
    - → High overhead of accessing storage devices
    - → Fail to check duplicates across multiple instances

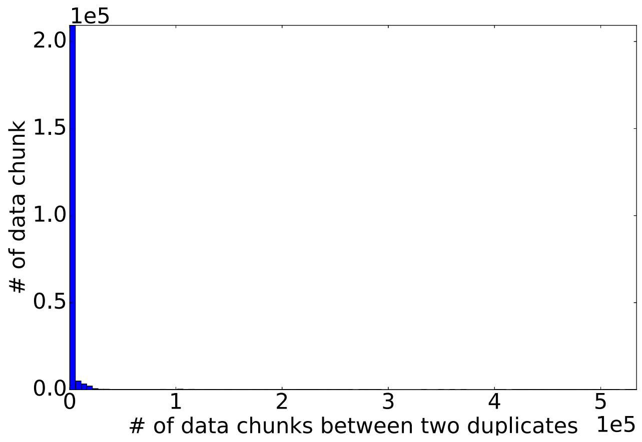
- **Multi-stream in clouds**
  - Locality may be weak in primary workloads[Yu, TPDS'16]
  - Locality between applications/workloads varies
  - Multiple streams are mixed and random → Destroy locality

Motivation III: Locality

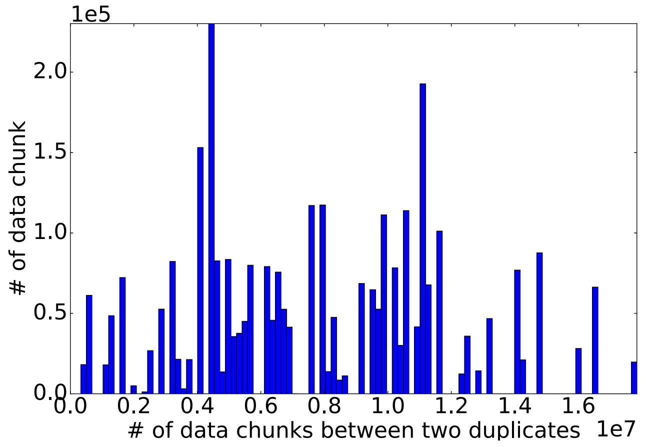
- **Temporal locality may be weak for workloads**
  - **FIU:** small and skewed → Good locality
  - **Cloud-FTP:** Large and scattered → Weak locality



(a) ***FIU-mail trace***



(b) ***FIU-web trace***



(c) ***Cloud-FTP trace***

Histogram for the distribution of distance between duplicate blocks

E.g.: "**abac**" represents series of data block
The number of data blocks between two adjacent occurrence of "**a**" is 1 (x-aixs)

Motivation III: Locality

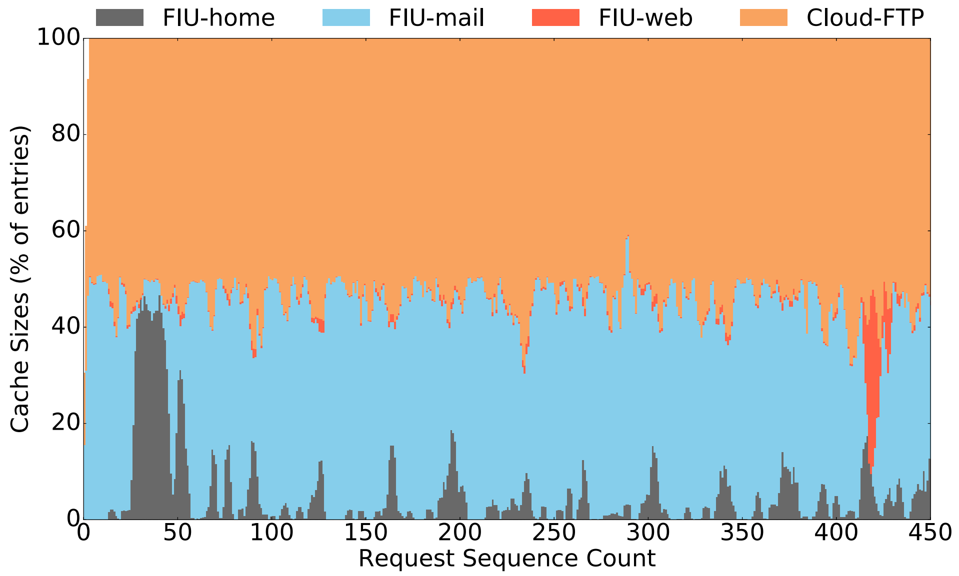
➢ Cache efficiency with mixed locality

- Cache size: 32K entries
- 2-hour traces are mixed according to timestamps

Trace	Request number	Write request ratio	Duplicate writes
Cloud-FTP	2293424	84.15%	387140
FIU-mail	1961588	98.58%	1633424
FIU-web	116940	49.36%	30534
FIU-home	293605	91.03%	32688

- Duplicate writes: FIU-mail > 4 * Cloud-FTP
- Cache size: FIU-mail < 0.8 * cloud-FTP

Cache allocation is not reasonable for mixed stream with different locallity

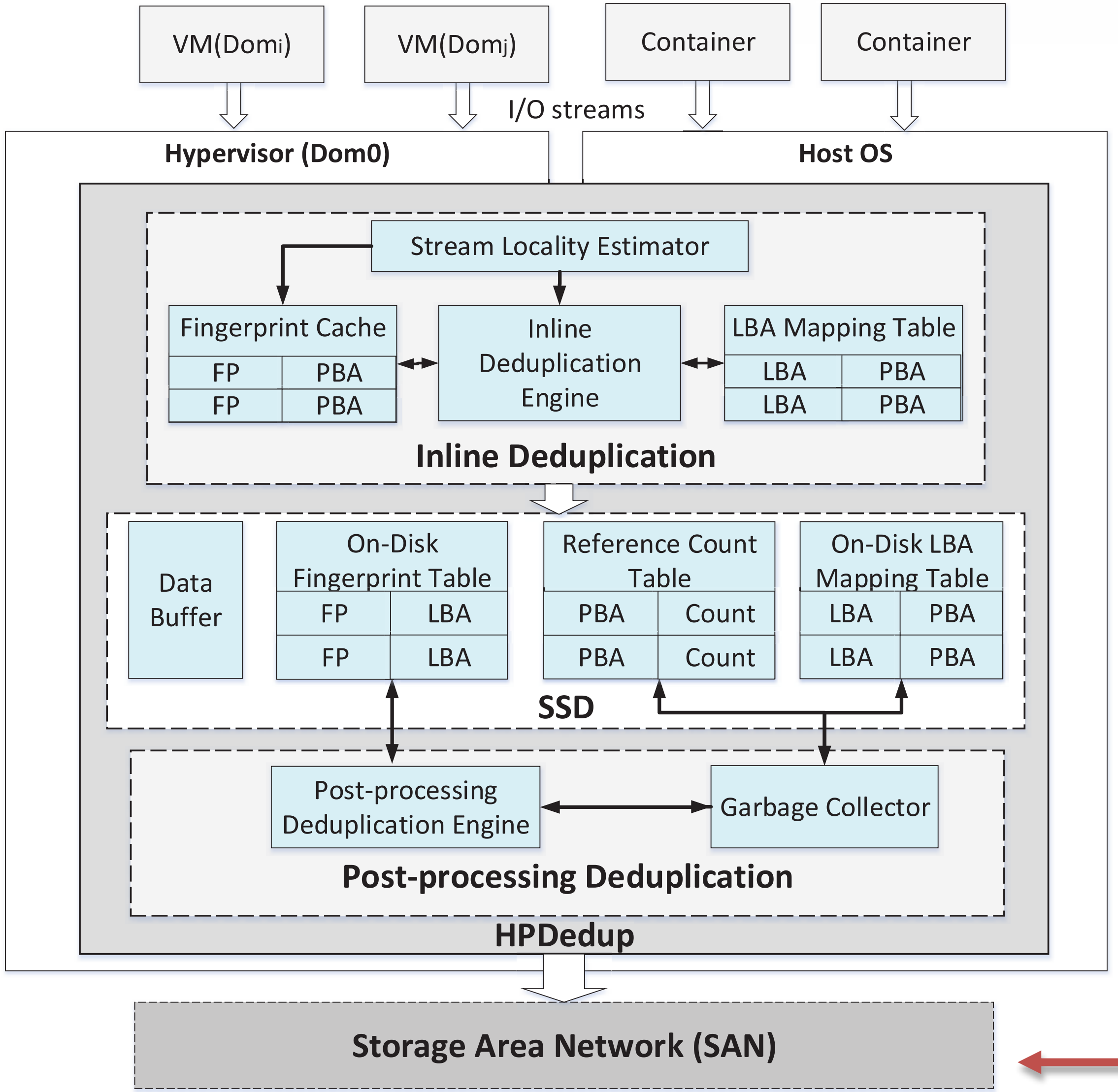


Cache size allocation occupied by each data stream using **LRU**

Hybrid Prioritized Deduplication

- ***HPDedup***, a **H**ybrid **P**rioritized **Dedup**lication method for primary storage in the cloud
  - Work at the hypervisor level to eliminate duplicate data blocks or block device layer of the host
  - Achieve exact-deduplication
  - Improve inline cache efficiency
- Techniques
  - Inline deduplication phase
    - Estimate the temporal locality for streams
    - Prioritize cache allocation for better locality streams
  - Post-processing deduplication phase
    - Scan on-disk fingerprint table and identify duplicates

Architecture



- **LBA**: Logical Block Address
- **PBA**: Physical Block Address

Hypervisor: translate LBA to PBA for block I/O requests from VMs running on top of it

- **Stream locality estimator**
  - Estimate the temporal locality and prioritize cache size allocation

- **Post-processing deduplication**
  - Scan on-disk fingerprint table to remove duplicates

Storage devices connected to it via SAN

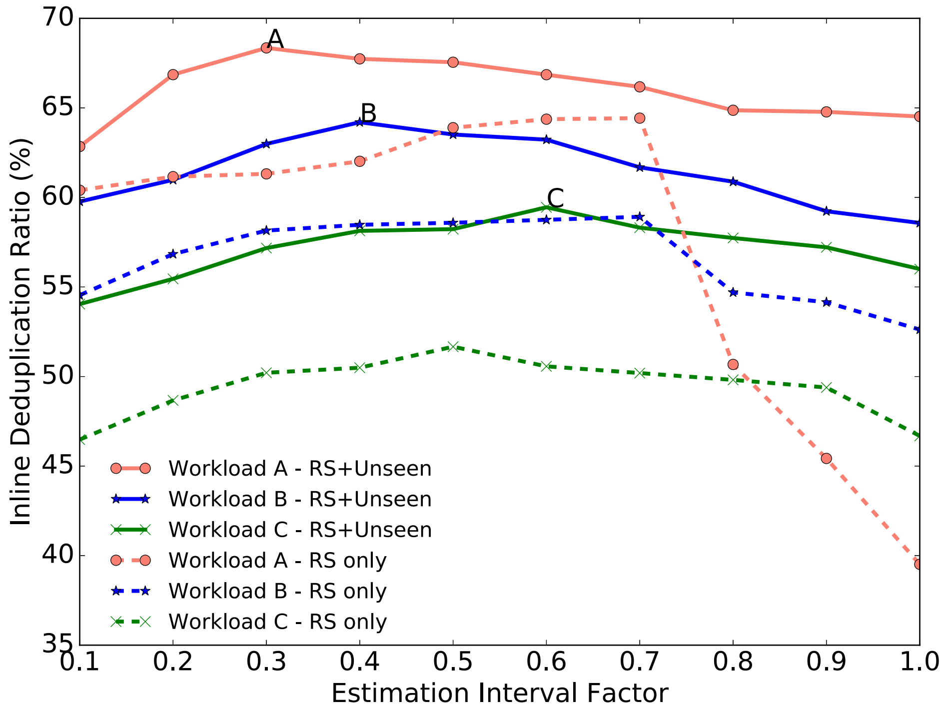
Evaluate the Temporal Locality

- **Metric: Local Duplicate Set Size (LDSS)**
  - The number of duplicate fingerprints in last *n* contiguous data blocks arriving before a given time (***n estimation interval***)
  - How to obtain historical **LDSS**?
- **Intuitive idea: Count distinct fingerprints for each stream**
  - High memory overhead (all fingerprints needs to be recorded)
- **Better idea:** sampling + estimate
  - **Reservoir sampling**: Guarantee that each fingerprint is sampled with same probability even if the set size is unknown
  - **Unseen**[Valiant, NIPS'13]: Estimate the unseen data distribution based on the histogram of the samples of observed data

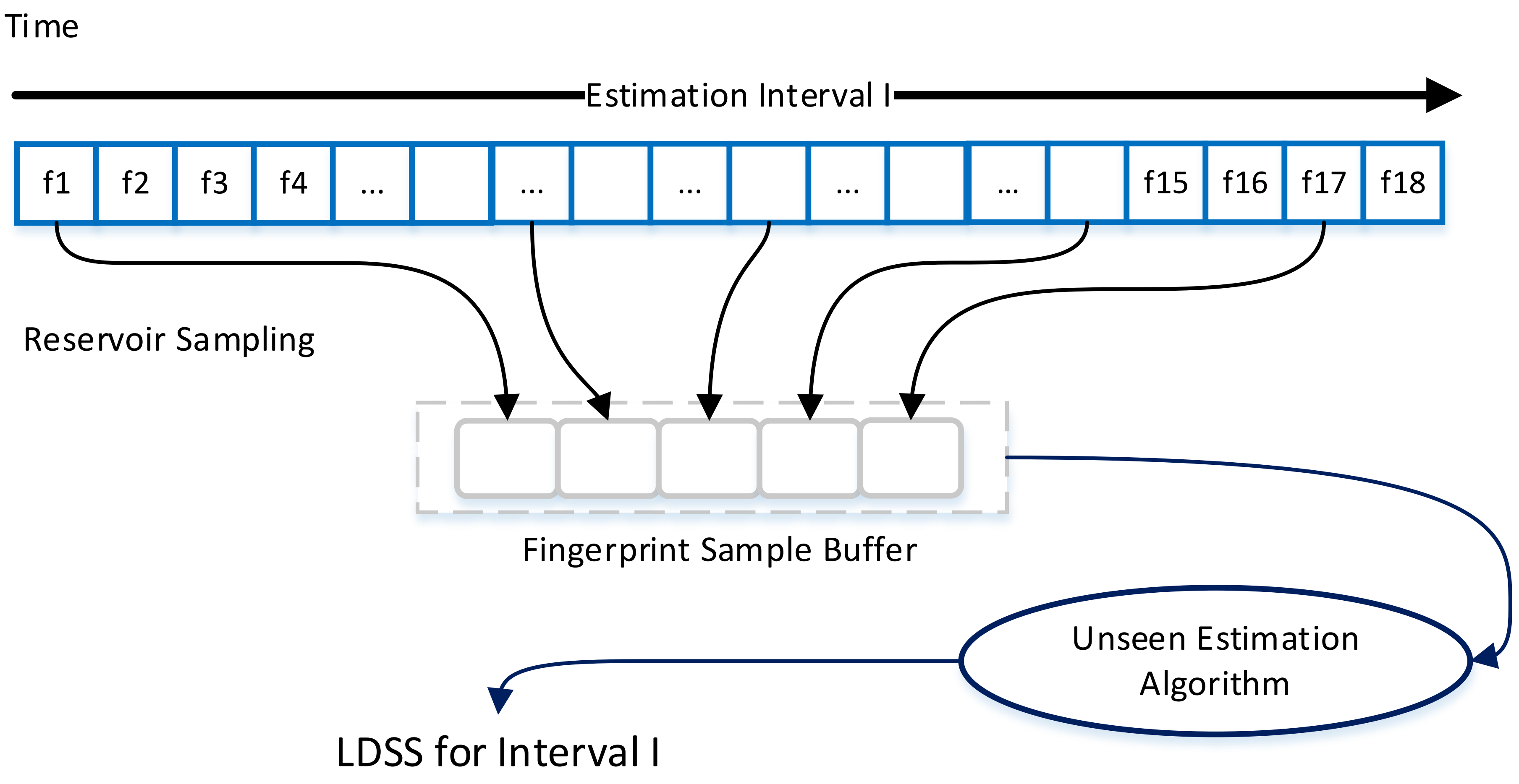
Temporal Locality Estimation

- **Intuitive idea:** Directly estimating the **LDSS** of data streams by the number of fingerprint samples → **Not feasible**
  - whether a sampled fingerprint is duplicate is not independent

- **Keys to estimation**
  - Unseen algorithm
  - A proper estimation interval
    - More unique data blocks → Larger interval
  - Predicted LDSS for **evict priority**
    - Allocate more cache to data stream with higher locality

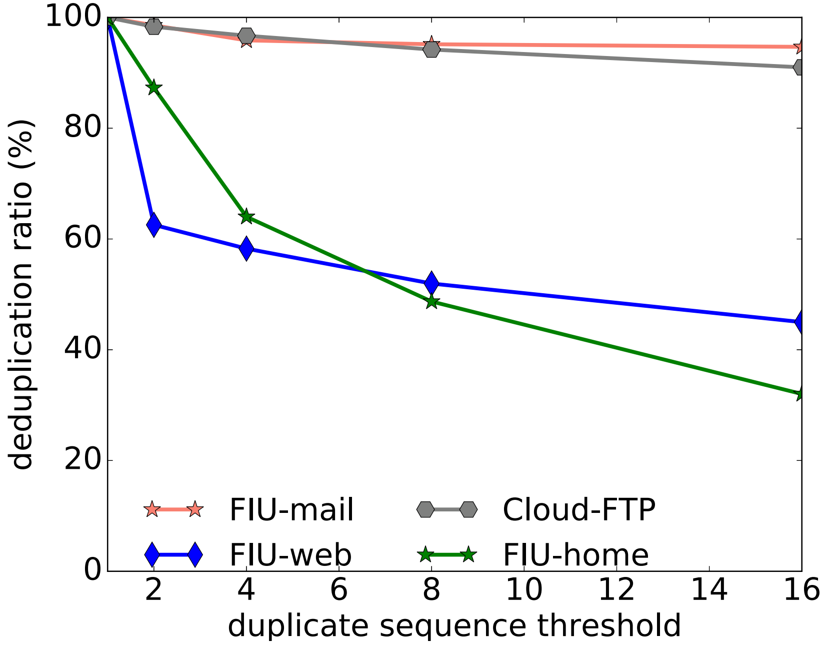


Estimation for LDSS



Exploit the Spatial Locality

- **Exploit spatial locality to reduce disk fragmentation**
  - Perform deduplication on block sequences longer than a fixed threshold
  - Spatial locality varies between different workloads



- **Threshold should be adaptive**
  - Sensitive or insensitive to change
- **Balance between write latency and read latency**
  - Write operations → Shorter T → Long sequences means more comparisons
  - Read operations → Longer T → Less Random I/O

$$T = (1 - r) * \text{Average length of block sequences} + \text{r} * \text{Average of read length}$$

Experimental Setup

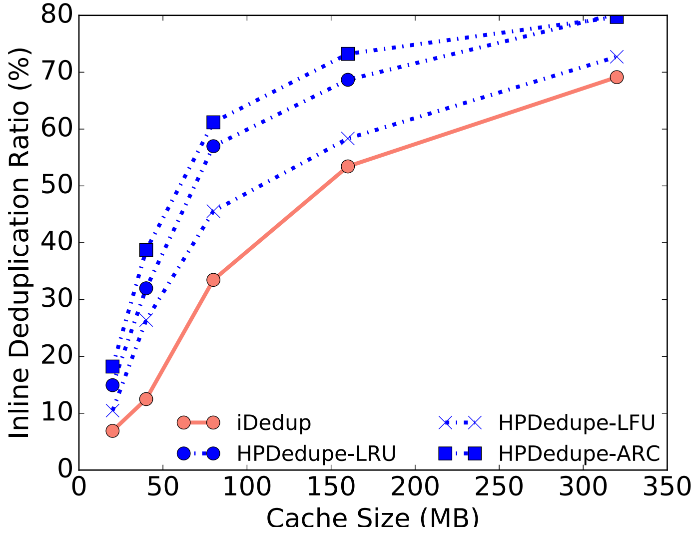
- Evaluation Setup
  - Intel Core i7-4790 CPU, 32GB RAM and 128GB SSD + 1TB HDD
- Datasets
  - **FIU** traces: **FIU-home**, **FIU-web**, **FIU-mail**
  - **Cloud-FTP**: trace from a cloud FTP server (collected by their research group)
- Comparison: Inline(**iDedup**[FAST'12]), post-processing and hybrid(**DIODE**[MASCOTS'16])
- Mixing workloads as multiple VMs
  - Different ratios between good locality(FIU, L) and bad locality(Cloud-FTP, NL)
  - Workload A → L:NL = 3:1
  - Workload B → L:NL = 2:2
  - Workload C → L:NL = 1:3

Trace	Num of requests	Write request ratio	Duplicate ratio
Cloud-FTP	21974156	83.94%	20.77%
FIU-mail	22711277	91.42%	90.98%
FIU-web	676138	73.27%	54.98%
FIU-home	2020127	90.44%	30.48%

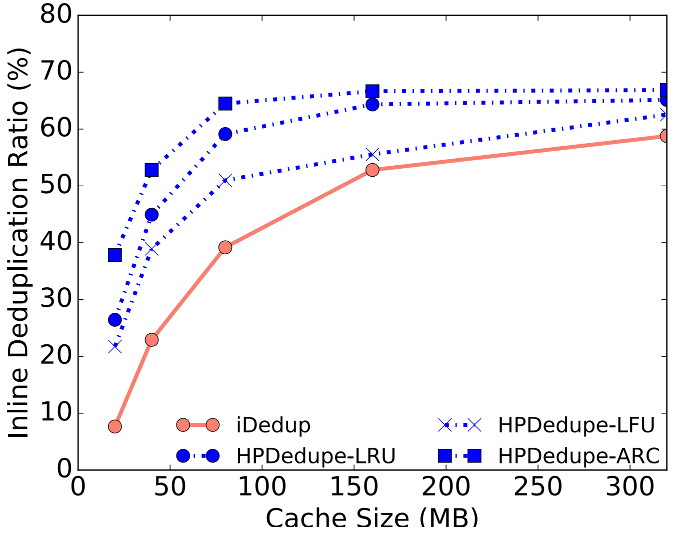
Inline Deduplication Ratio

➢ **Methodology**

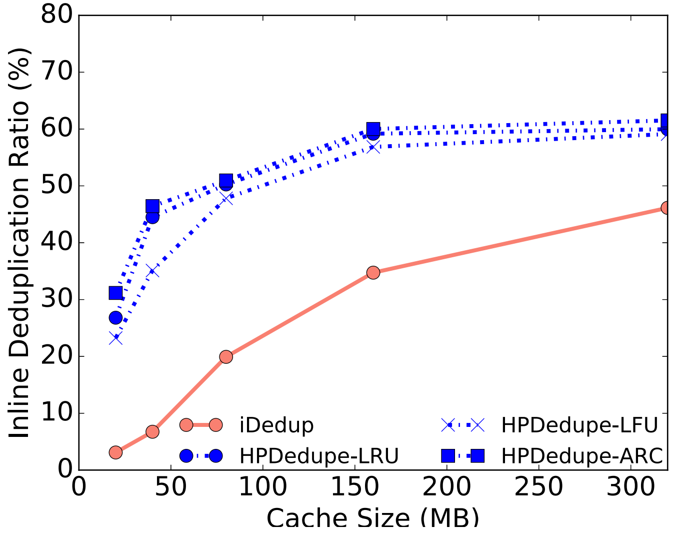
- Each write I/O request for the three workloads is a 4KB block
- Deduplication threshold: 4 for **iDedup** and **HPDedup**
- Cache size: 20MB-320MB



(a) ***Workload A (L:NL = 3:1)***



(b) ***Workload B (L:NL = 2:2)***

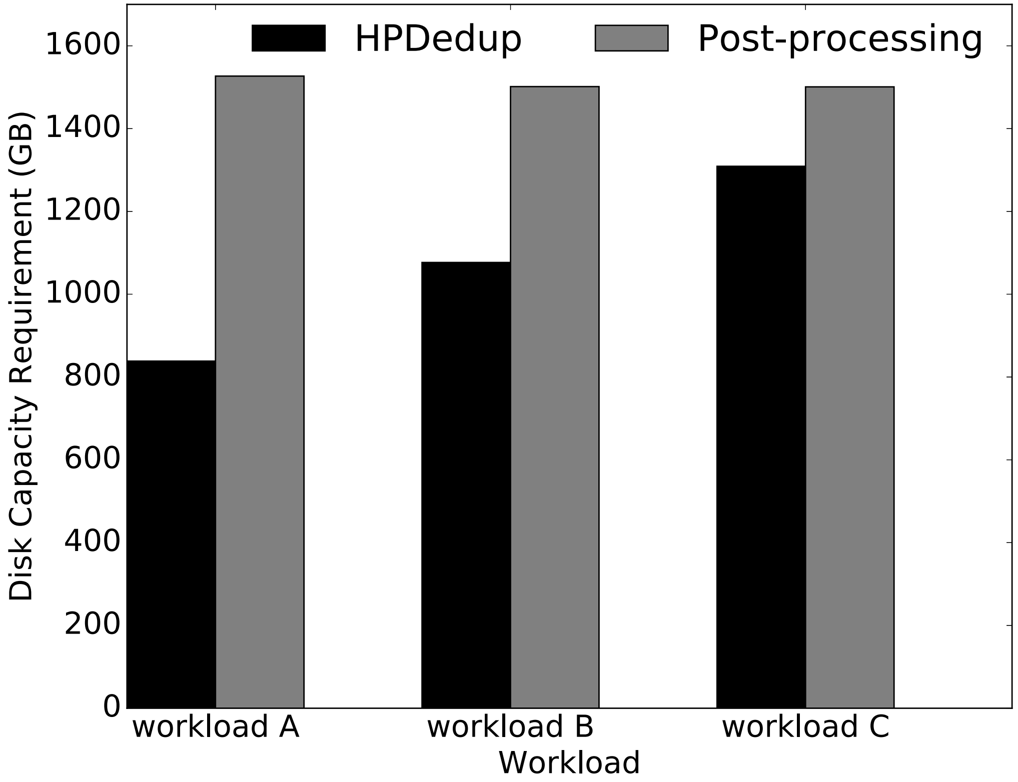


(c) ***Workload C (L:NL = 1:3)***

➢ **HPDedup** improves the overall cache efficiency for multiple VMs/applications, especially for ***workload C***

Peak Disk Capacity Requirement

- The disk capacity requirements for **HPDedup** and pure post-processing deduplication schemes

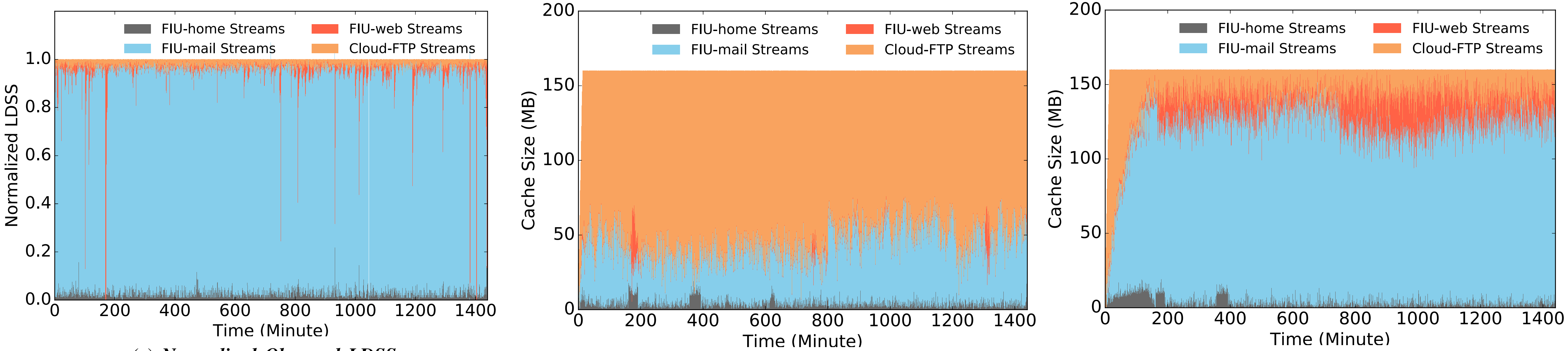


The better the locality
→ the more duplicates can be detected in the inline phase
→ the more duplicate data writes can be eliminated

LDSS Estimation Accuracy

➢ **Evaluation: Observed LDSS for workload B**

- a: observed LDSS over time



(a) *Normalized Observed LDSS*

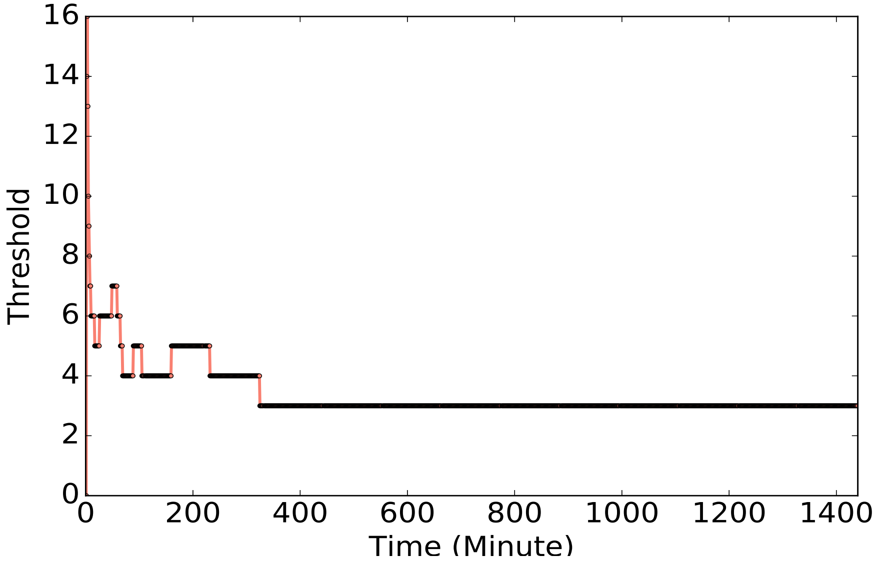
(b) *Cache Size Distribution (without LDSS estimation)*

(c) *Cache Size Distribution (with LDSS estimation)*

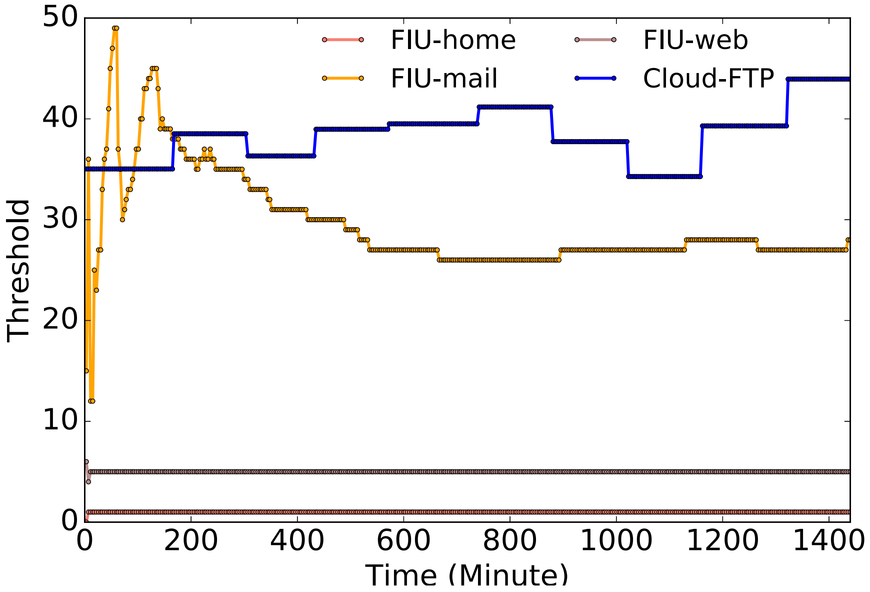
➢ This indicates the effectiveness of **LDSS** estimation (inline deduplication ratio: **12.5%**)

Thresholds Adjustment

- **Methodology**
  - Workload: Workload A
  - **DIODE** uses a global threshold



(a) ***DIODE***



(b) ***HPDedup***

- **HPDedup** can adjust thresholds for different streams
- Larger threshold leads to less disk fragmentation → Deduplication ratio for **HPDedup** and **DIODE**: 68.96% and 57.62%

Conclusion

- ***HPDedup***, a **H**ybrid **P**rioritized **Dedup**lication method for primary storage in the cloud
  - Work at the hypervisor level to eliminate duplicate data blocks or block device layer of the host
  - Inline deduplication phase: Estimation & Cache priority
  - Post-processing: Perform exact-deduplication
- Effectiveness
  - Improve Cache efficiency
  - Improve inline deduplication ratio
  - Reduce duplicate to post-processing phase and Reduce fragmentation