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

Huijun Wu[1,4], Chen Wang[2], Yinjin Fu[3], Sherif Sakr[1], Liming Zhu[1,2] and Kai Lu[4]

The University of New South Wales[1]

Data61, CSIRO[2]

PLA University of Science and Technology[3]

National University of Defence Technology[4]

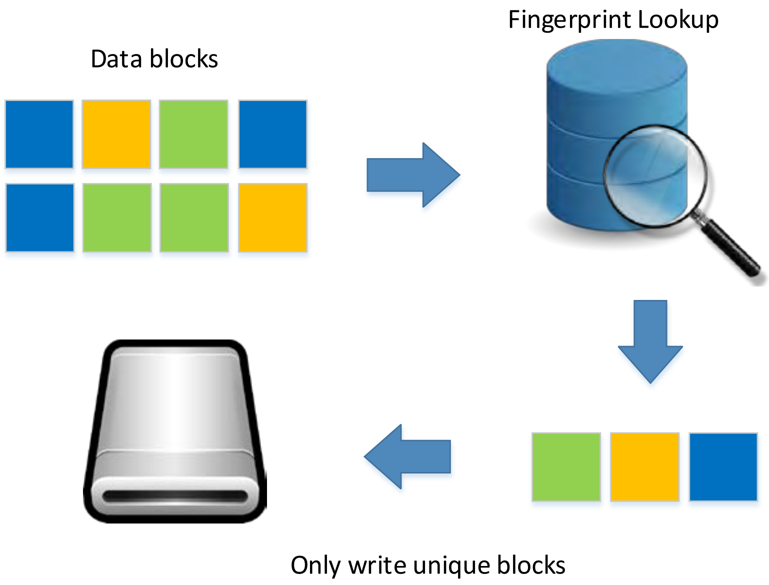
Outline

- **Background**
- Motivations
- Hybrid Prioritized Deduplication
- Experiment Results
- Conclusion

Background

- Primary Storage Deduplication
  - Save the storage capacity
  - Improve the I/O efficiency

- The state-of-the-art
  - Post-processing deduplication
    - Perform during off-peak time
  - Inline deduplication
    - Perform on the write path



Post-processing Deduplication

- The commodity product uses post-processing deduplication [TOS’16]
  - Windows Server 2012 [ATC’12]

- Challenges remain for real-world systems
  - Off-peak periods may not be enough
  - *More storage capacity is required*
  - *Duplicate writes shorten the lifespan of storage devices (e.g., SSD)*
  - *Does not help improving the I/O performance, but wastes I/O bandwidth*

- Inline deduplication can help

Inline Deduplication

- Fingerprint look-up is the bottleneck
  - On-disk fingerprint table introduces high latency
  - Fingerprint table is large and hard to fit in memory
  - Cache efficiency is critical

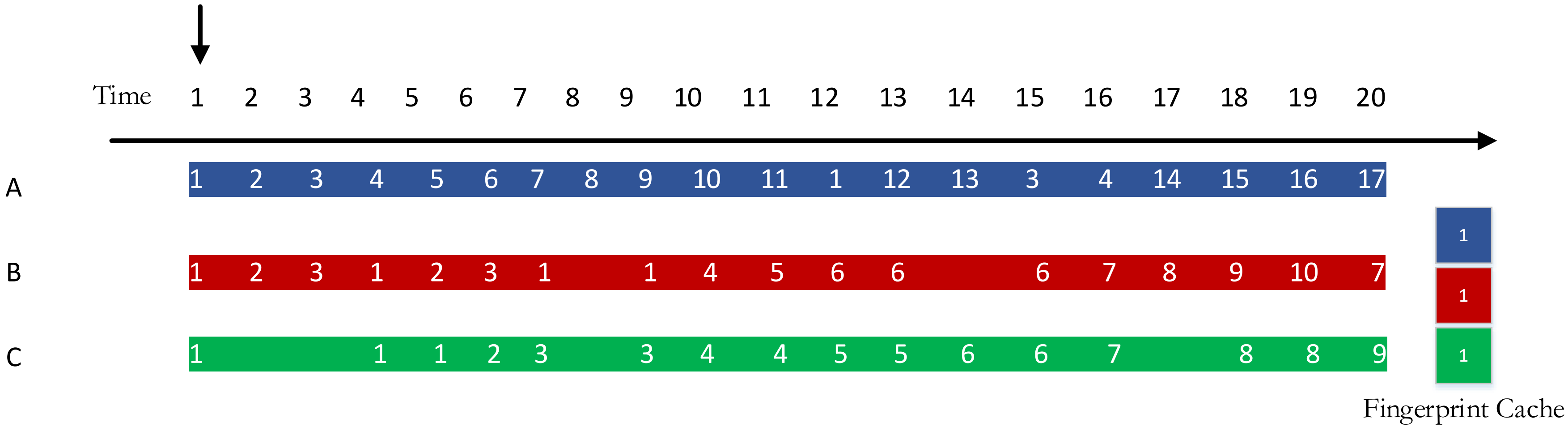
- The state-of-the-art solutions and challenges
  - Exploit the temporal locality of workloads [FAST'12][IPDPS'14]
    - But temporal locality may not exist [TPDS'17]
  - For cloud scenario,
    - locality for workloads of different VMs may be quite different
  - Workloads may interfere with each other and reduce the cache efficiency

Outline

- Background
- **Motivations**
- Hybrid Prioritized Deduplication
- Experiment Results
- Conclusion

Motivation

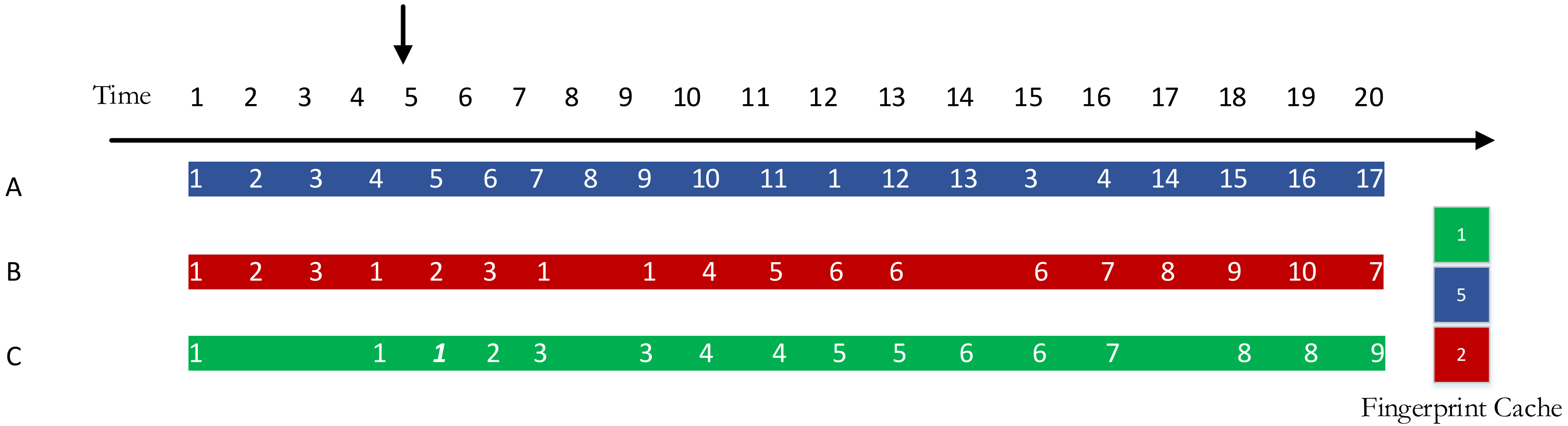
- Workloads with different temporal locality interfere with each other
  - A toy example.



# of Deduplicated Blocks: 0

Motivation

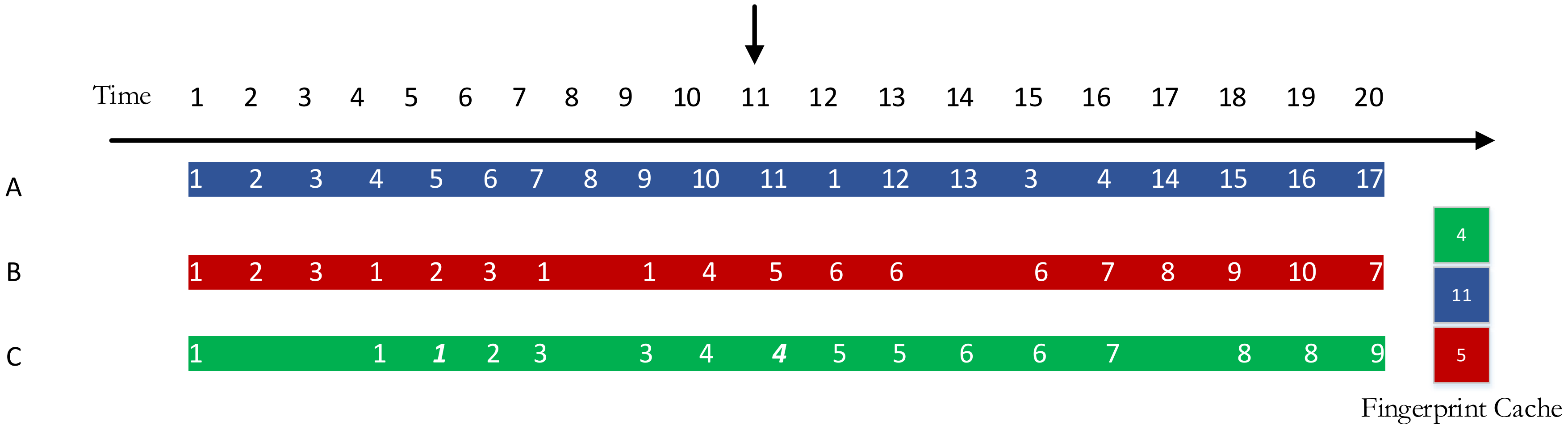
- Workloads with different temporal locality interfere with each other
  - A toy example.



# of Deduplicated Blocks: 1

Motivation

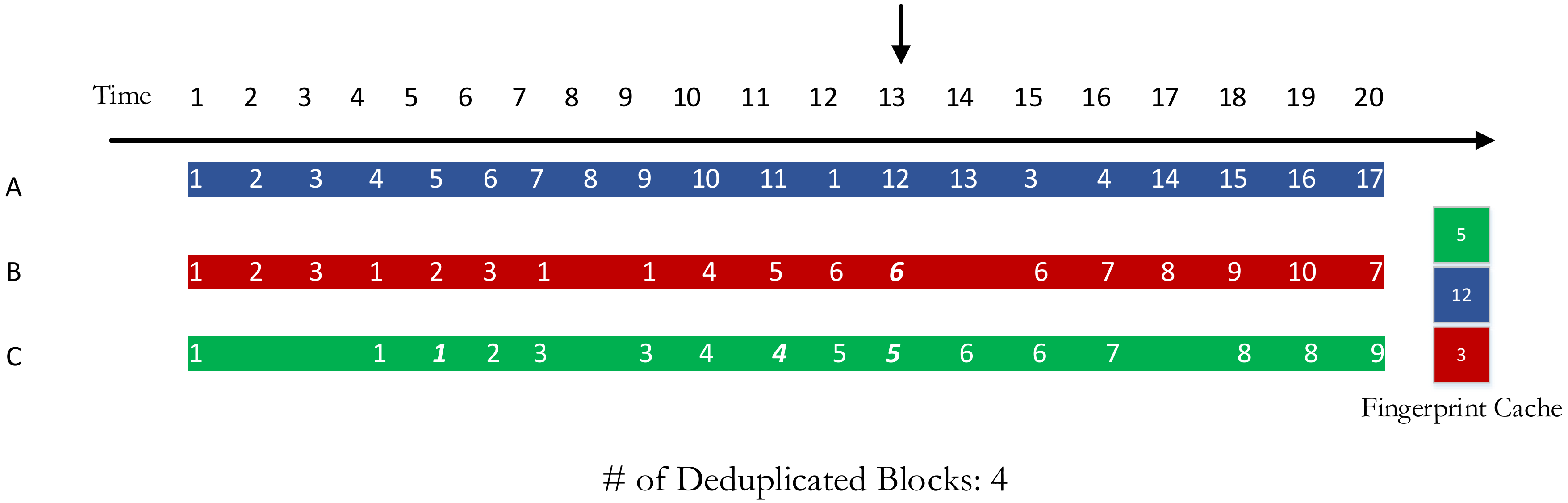
- Workloads with different temporal locality interfere with each other
  - A toy example.



# of Deduplicated Blocks: 2

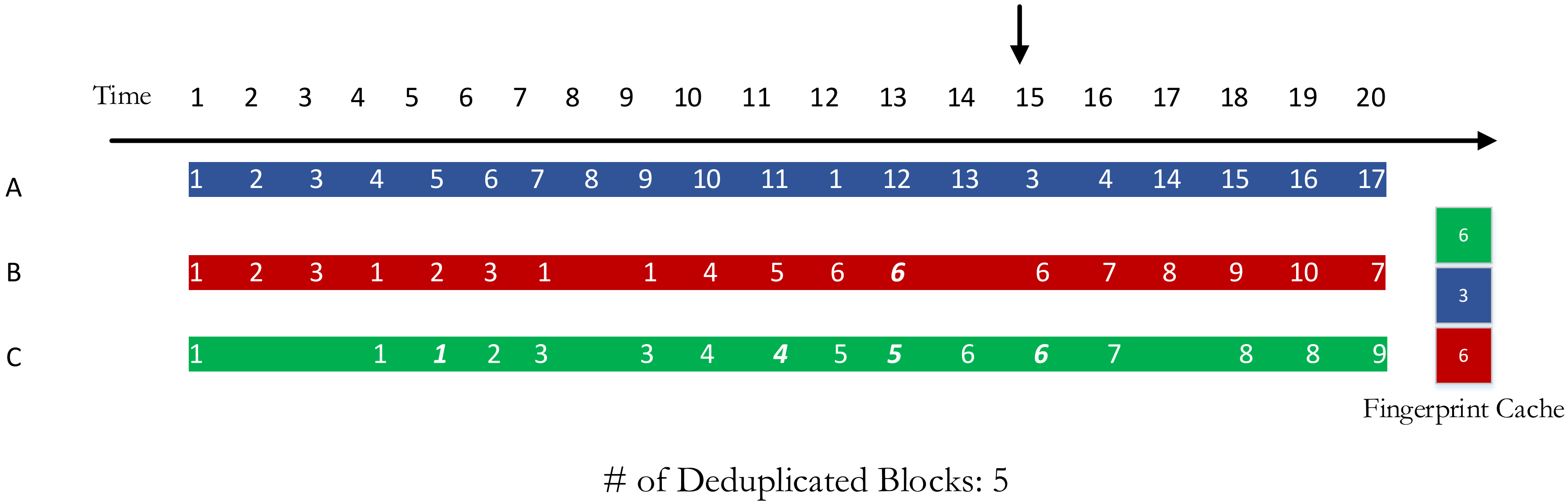
Motivation

- Workloads with different temporal locality interfere with each other
  - A toy example.



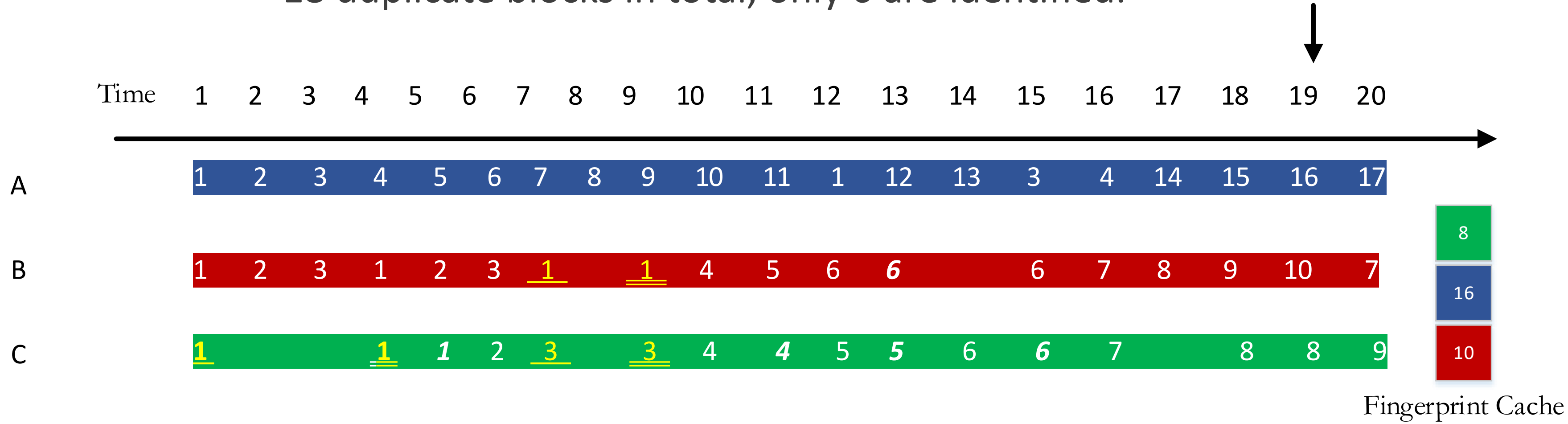
Motivation

- Workloads with different temporal locality interfere with each other
  - A toy example.



Motivation

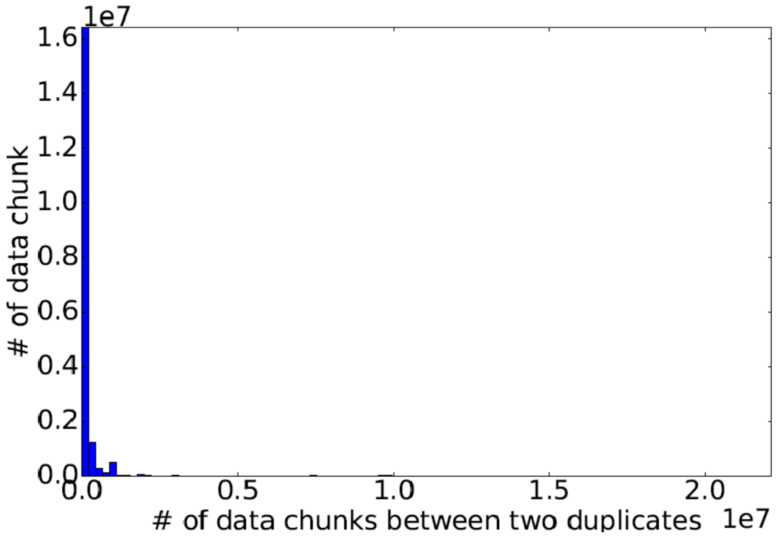
- Workloads with different temporal locality interfere with each other
  - A toy example.
    - 18 duplicate blocks in total, only 6 are identified.



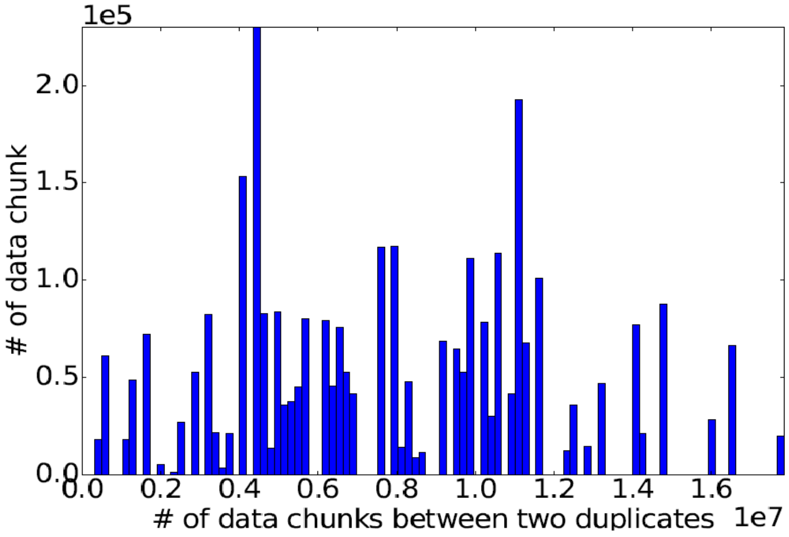
\# of Deduplicated Blocks: 6

Motivation

- Temporal locality may be weak for workloads
  - Histogram for the distribution of distance between duplicate blocks



FIU-mail



Cloud-FTP

Motivation

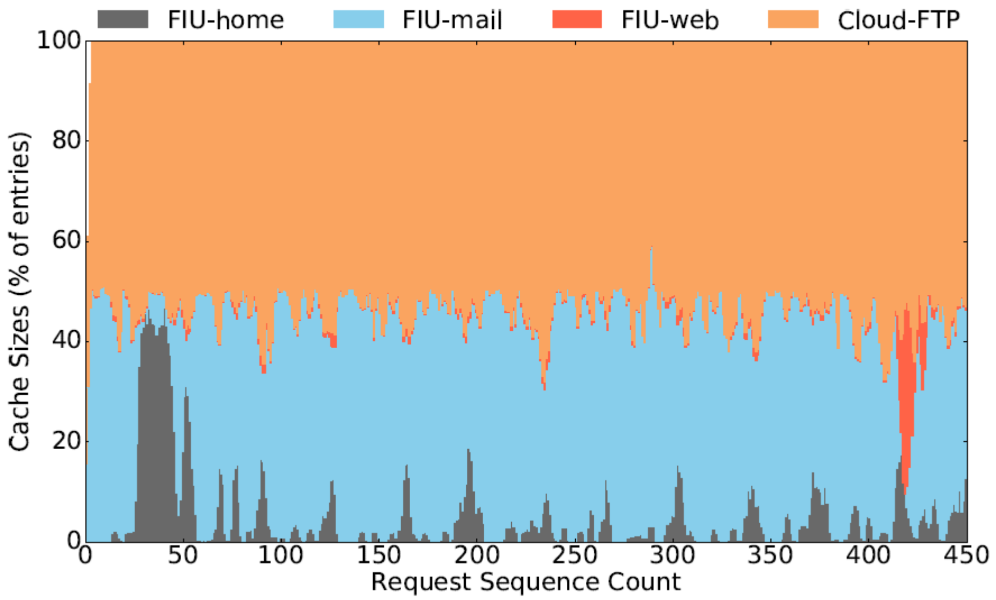
- Workloads with different temporal locality interfere with each other
  - Using real-world I/O trace. (LRU)

Table I: Workload Statistics of the 2-hour traces.

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

# of duplicate blocks: FIU-mail > 4*Cloud-FTP
Occupied cache size: FIU-mail < 0.8*Cloud-FTP

Cache resource allocation is unreasonable!



Outline

- Background
- Motivations
- **Hybrid Prioritized Deduplication**
- Experiment Results
- Conclusion

Hybrid Prioritized Deduplication

- **Hybrid** inline & post-processing deduplication
  - Either post-processing or inline deduplication works well
  - Solution: Combine inline and post-processing deduplication together
  - Identifying more duplicates by inline caching
  - Using post-processing to achieve exact deduplication

- Challenges: Interference compromises the temporal locality of workload, thus reducing the efficiency of fingerprint caching

- We differentiate workloads (data streams) to improve it

Hybrid Prioritized Deduplication

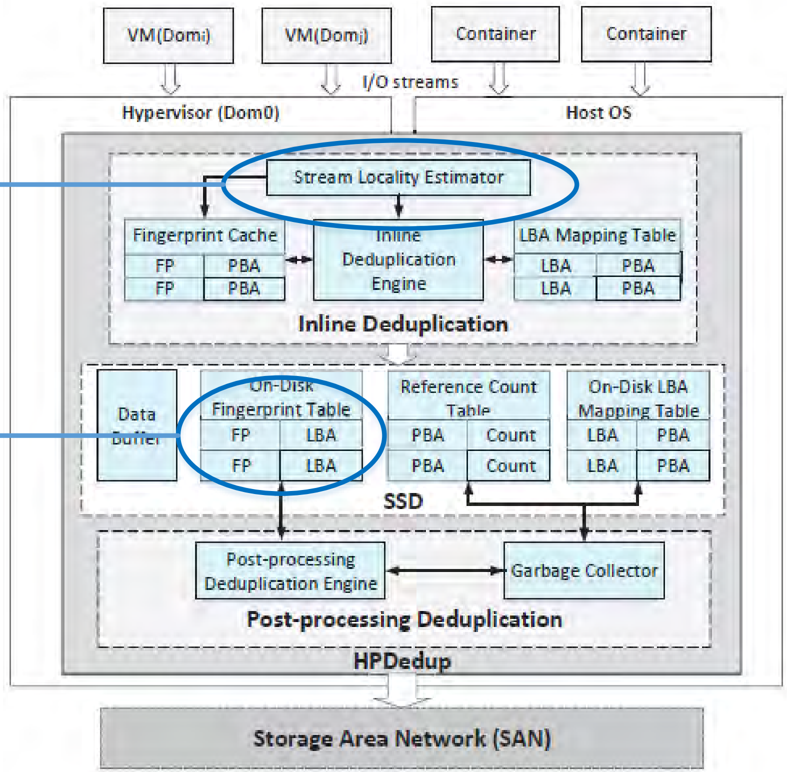
- **Prioritize** the cache allocation for inline deduplication
  - Data stream that contributes more deduplication ratio should get more cache resources
  - For inline phase, deduplication ratio comes from better temporal locality

- How to evaluate temporal locality ?
  - Changes dynamically with time
  - Accurate estimation is critical to achieve good cache allocation
  - Use # of duplicate blocks in *N* consecutive data blocks (*estimation interval*) as an indicator for temporal locality

System architecture

Estimate the temporal locality for streams and allocate cache according to this.

On-disk fingerprint table for post-processing deduplication.



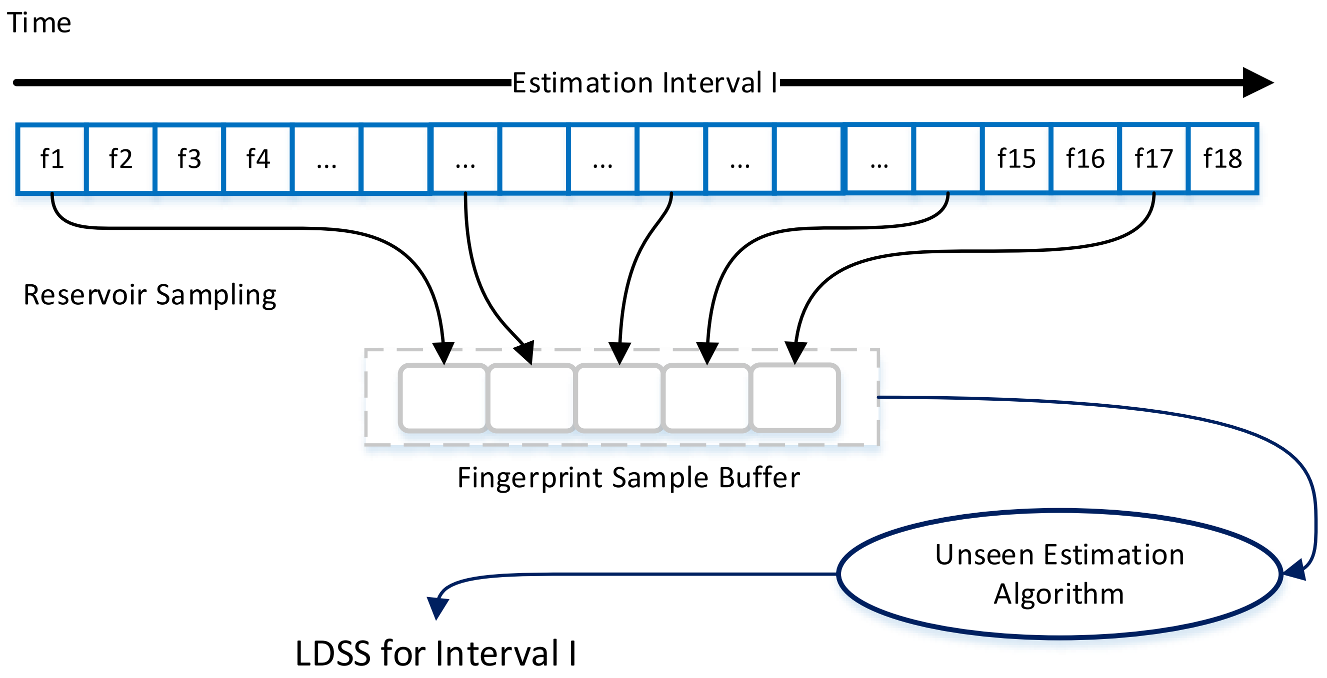
Evaluate the temporal locality

- Simple idea: Count distinct data block fingerprints for streams
  - Introduce high memory overhead
  - May be comparable to the cache capacity

- Estimate rather than count
  - Get the number of distinct fingerprints by small portion of samples
  - Essentially same as a classical problem ‘How many distinct elements exist in a set ?’ Origin – Estimate # of species of animal population from samples [Fisher, JSTOR’1940]
  - Sublinear estimator – Unseen Estimation Algorithm [NIPS’13]

Estimate the temporal locality

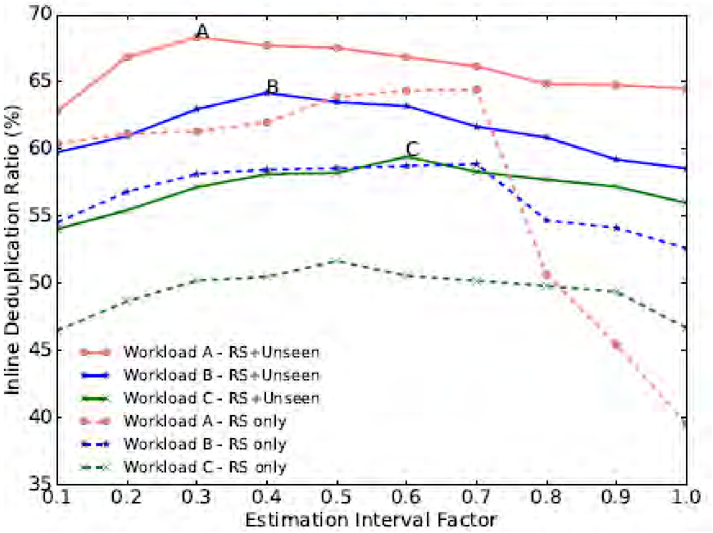
- Using unseen algorithm to estimate LDSS.



Key points to deploy the estimation

- Unseen algorithm requires uniform sampling
  - Each fingerprint should be sampled with the same probability
  - We use Reservoir Sampling [TOMS’04]

- Choose a proper *estimation interval*
  - More unique data blocks -> Larger interval
  - A good approximation
    - Historical inline deduplication ratio
  - Adaptive method

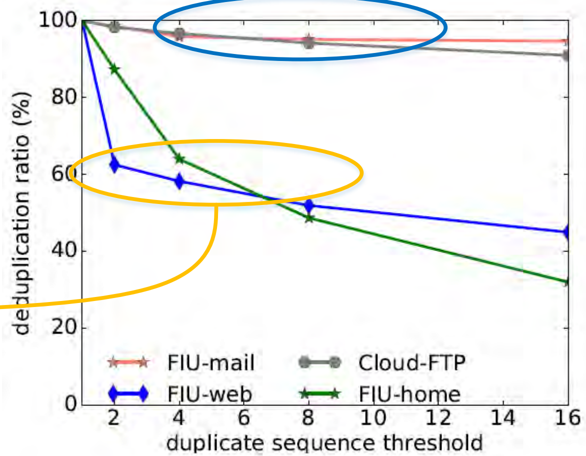


Differentiate the spatial locality

- Existing deduplication solutions exploit the spatial locality to reduce disk fragmentation
  - perform deduplication on block sequences longer than a fixed threshold.
- Workloads have different sensitivity to the increase of threshold
  - Differentiating the workloads achieves better deduplication ratio with less fragmentation

Insensitive to the threshold change

Sensitive to the threshold change



Outline

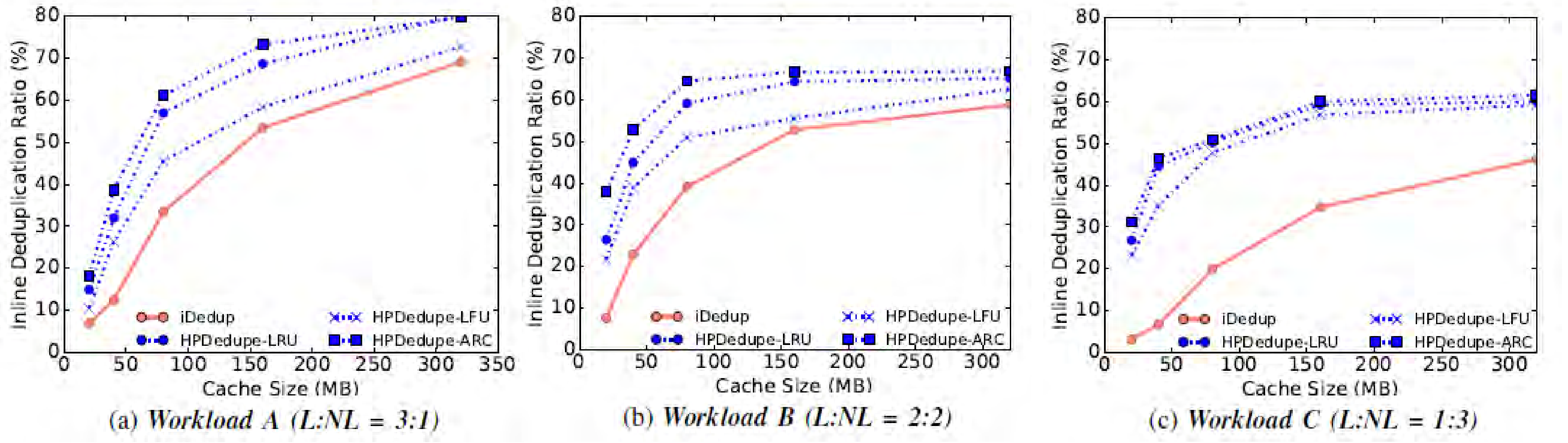
- Background
- Motivations
- Hybrid Prioritized Deduplication
- **Experiment Results**
- Conclusion

Evaluation Setup

- Evaluated Systems
  - compare with inline (iDedup), post-processing and hybrid (DIODE) deduplication schemas
- Workloads
  - FIU trace (FIU-home, FIU-web and FIU-mail)
  - Cloud-FTP (trace we collect from a FTP server by using NBD)
- Mixing workloads as multiple VMs
  - Different ratios between good locality (FIU, L) and bad locality (Cloud-FTP, NL) workloads
  - Workload A (L:NL = 3:1), workload B (L:NL = 1:1), workload C (L:NL = 1:3)

Evaluation

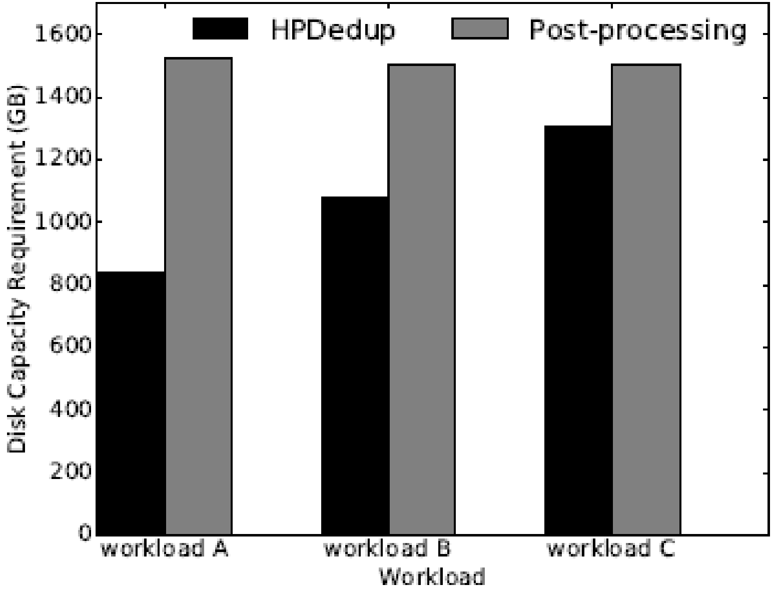
- Inline deduplication ratio
  - Cache size (20MB – 320MB)



  - HPDedup improves the inline deduplication ratio (8.04% - 37.75%)

Evaluation

- Data written to disks (Comparing with post-processing deduplication)



  - HPDedup reduce the data written to disks by 12.78% - 45.08%

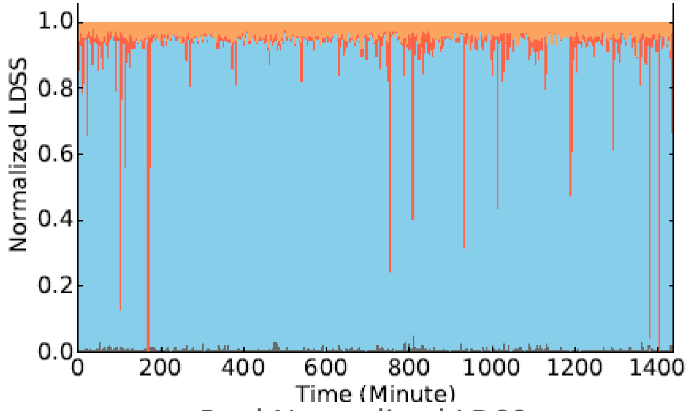
Evaluation

- Average # of hits for each cached fingerprint
  - DIODE [MASCOTS'16] skips files in inline deduplication based on file extensions

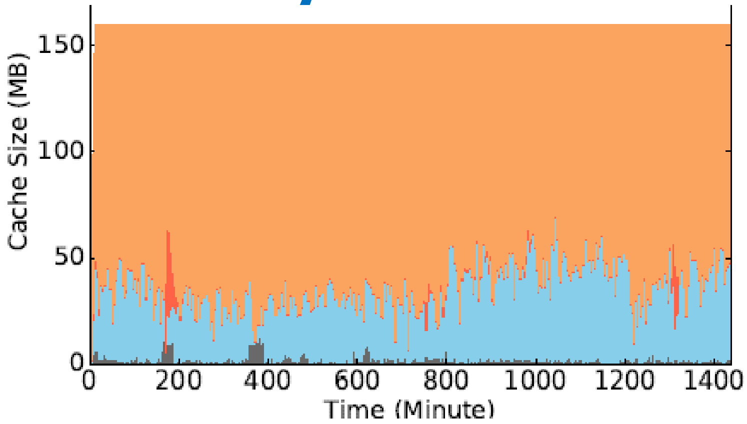
Schema	Cache Size	Workload A	Workload B	Workload C
Baseline	320MB	1.301	0.665	0.204
	160MB	0.781	0.551	0.148
DIODE	320MB	1.437	0.812	0.247
	160MB	0.805	0.598	0.181
HPDedup	320MB	1.812	4.024	5.918
	160MB	1.409	3.101	5.002

  - HPDedup classifies data at finer-grained (stream temporal locality level) so that the efficiency of inline deduplication can be further improved

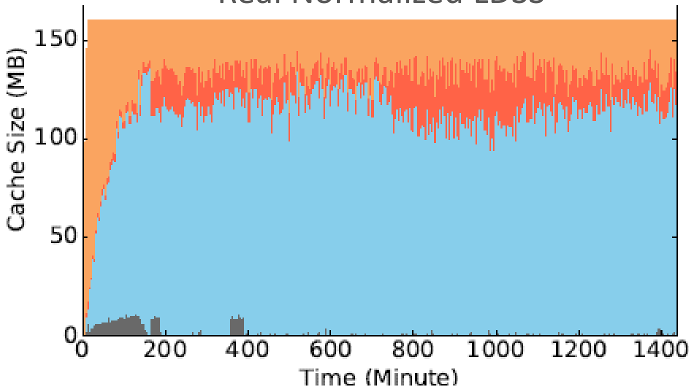
Evaluation – LDSS Estimation Accuracy



Real Normalized LDSS



Cache Occupation (without Locality Estimation)

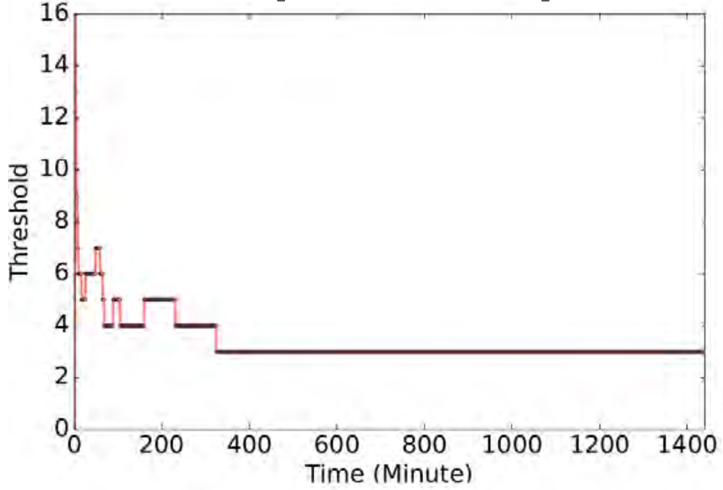


Cache Occupation (with Locality Estimation)

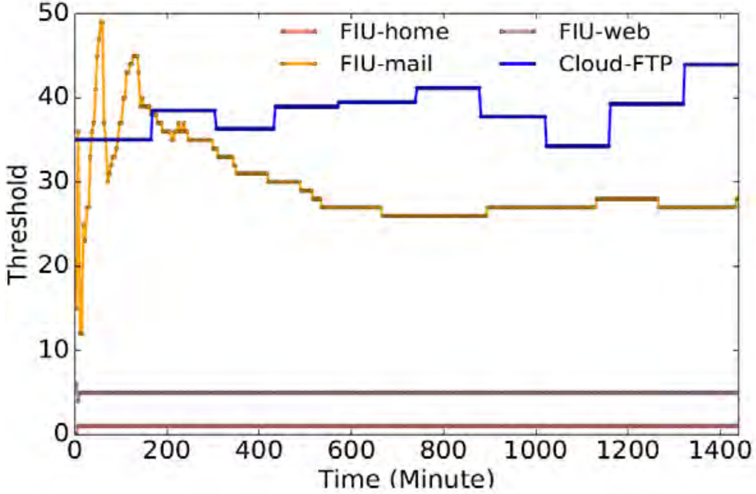
Locality estimation allocates cache resources according to the temporal locality of streams and improves the inline deduplication ratio by 12.53%.

Evaluation

- Deduplication threshold
  - DIODE [MASCOTS'16] does not differentiate workloads.



(a) ***DIODE***



(b) ***HPDedup***

- HPDedup introduces less fragmentation while achieving higher dedup ratio

Overhead Analysis

- Computational Overhead
  - Mainly includes (1) generating the fingerprint histogram and (2) linear programming of estimation algorithm
  - (1) 7ms for 1M fingerprints
  - (2) 27ms regardless of the estimation interval size
  - More intuitive feeling – ms level overhead for GBs of data writing
  - Can be computed asynchronously
- Memory Overhead
  - Up to 2.81% of cache capacity for the three workloads

Outline

- Background
- Motivations
- Hybrid Prioritized Deduplication
- Experiment Results
- **Conclusion**

Conclusion

- New hybrid, prioritized data deduplication
  - Fusing inline and post-processing deduplication
  - Differentiate the temporal and spatial locality of data streams coming from VMs and applications

- More efficient compared with the state of the art
  - Improve the inline deduplication ratio by up to 37.75%
  - Reduce the disk capacity requirement by up to 45.08%
  - Low computational and memory overhead

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

Huijun Wu

The University of New South Wales, Australia

Email: huijunw@cse.unsw.edu.au