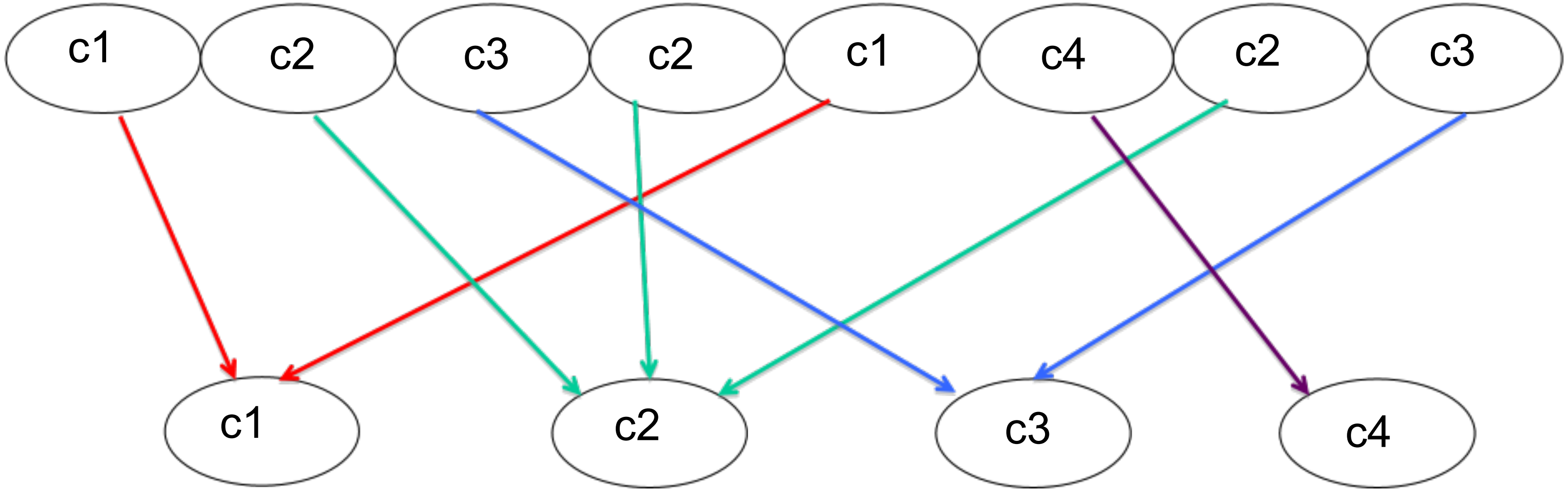
A Scalable Deduplication and Garbage Collection Engine for Incremental Backup

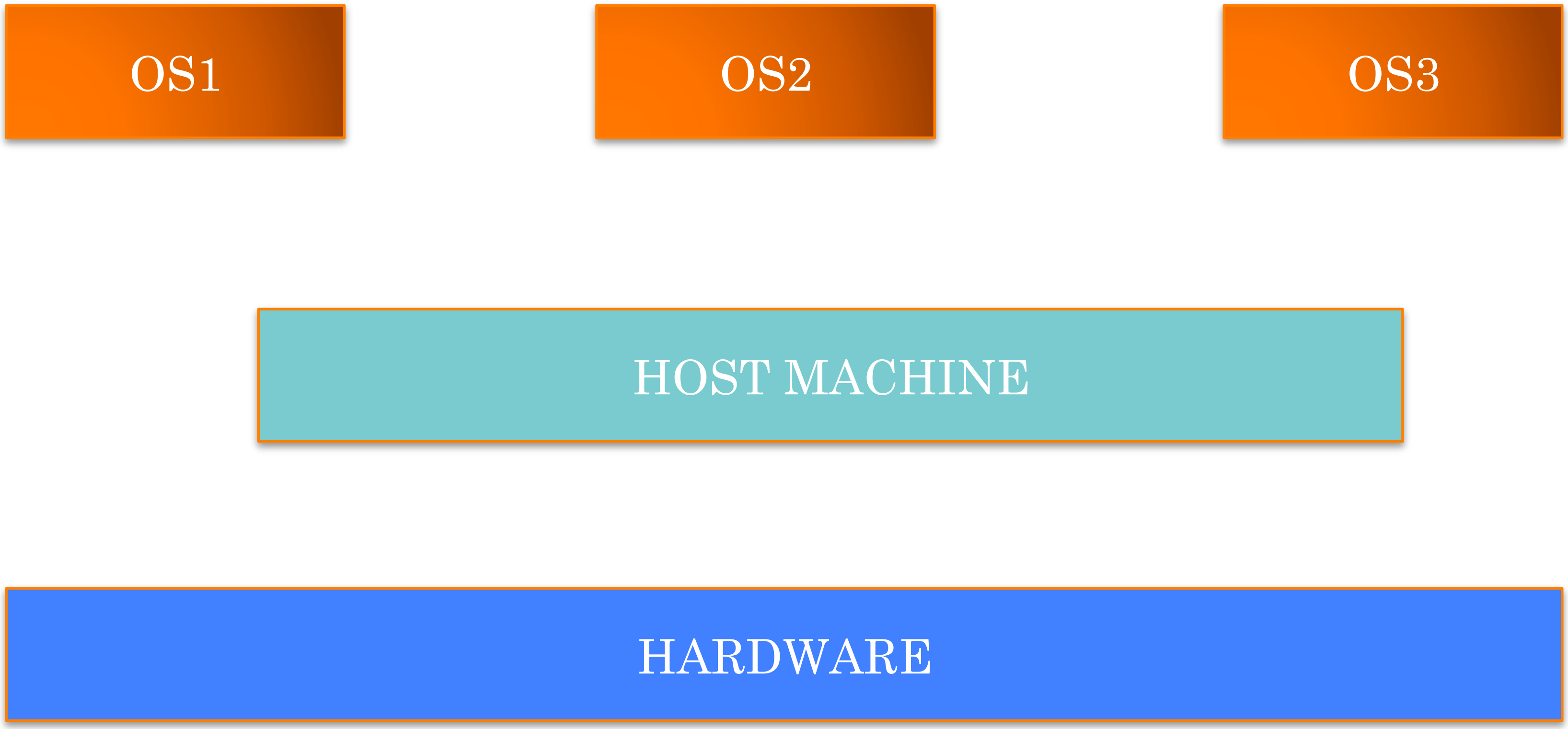
Dilip N Simha (Stony Brook University, NY & ITRI, Taiwan)
Maohua Lu (IBM Almaden Research Labs, CA)
Tzi-cker Chiueh (Stony Brook University, NY & ITRI, Taiwan)

WHAT IS DEDUPLICATION

Technique for eliminating redundant data



USE CASES



OS1

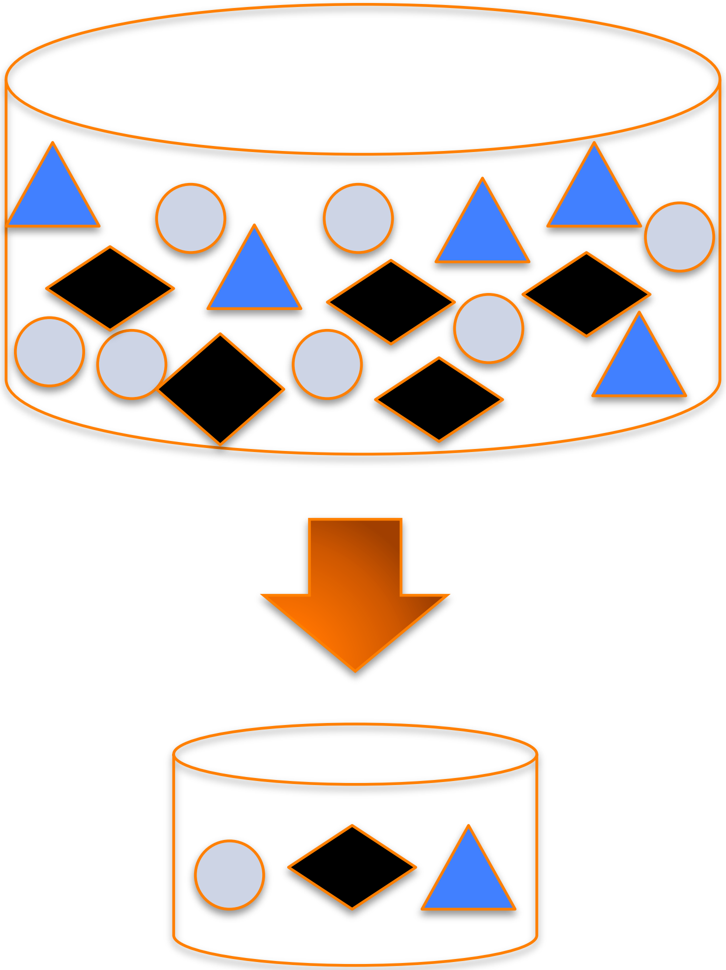
OS2

OS3

HOST MACHINE

HARDWARE

INTRODUCTION



Duplicity = Percentage of duplicate blocks / Blocks before deduplication

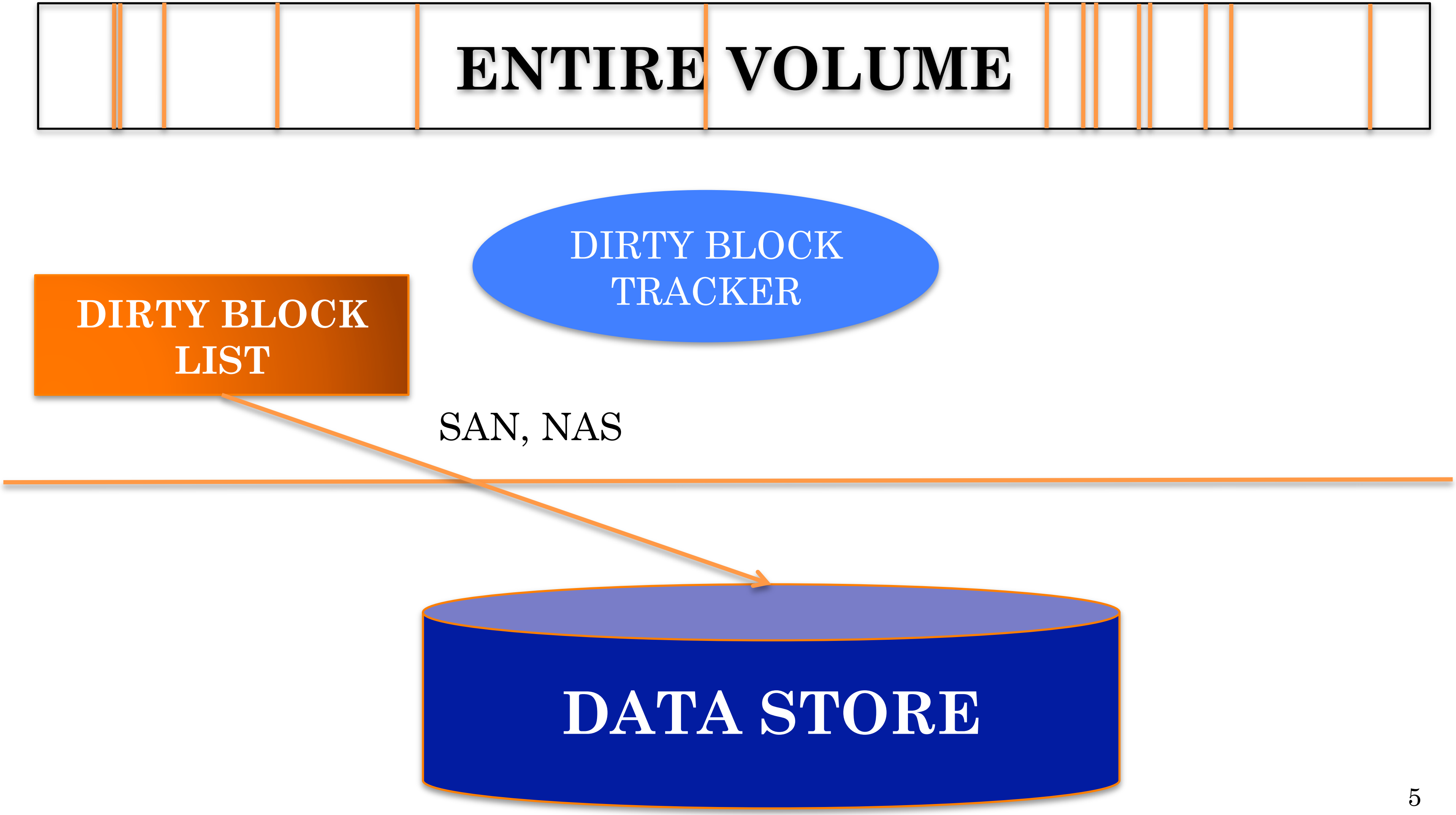
Deduplication Throughput = Number of blocks identified as a duplicate or not / second

Deeper inspection gives higher duplicity but at the cost of throughput

Incremental block level backups have **lesser locality** compared to full backups

A good balance requires sophisticated techniques to identify duplicates.

INCREMENTAL BACKUP



ENTIRE VOLUME

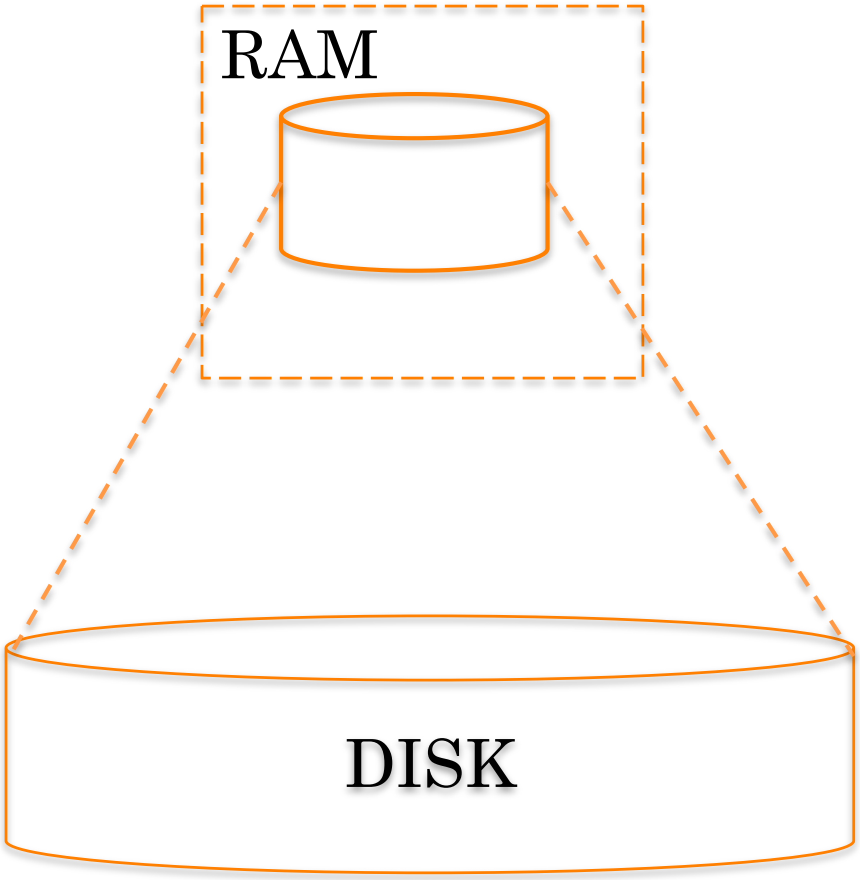
DIRTY BLOCK TRACKER

DIRTY BLOCK LIST

SAN, NAS

DATA STORE

MOTIVATION



1 PB Data Backup System

Block Size: 4KB

Fingerprint Size: 16 Bytes

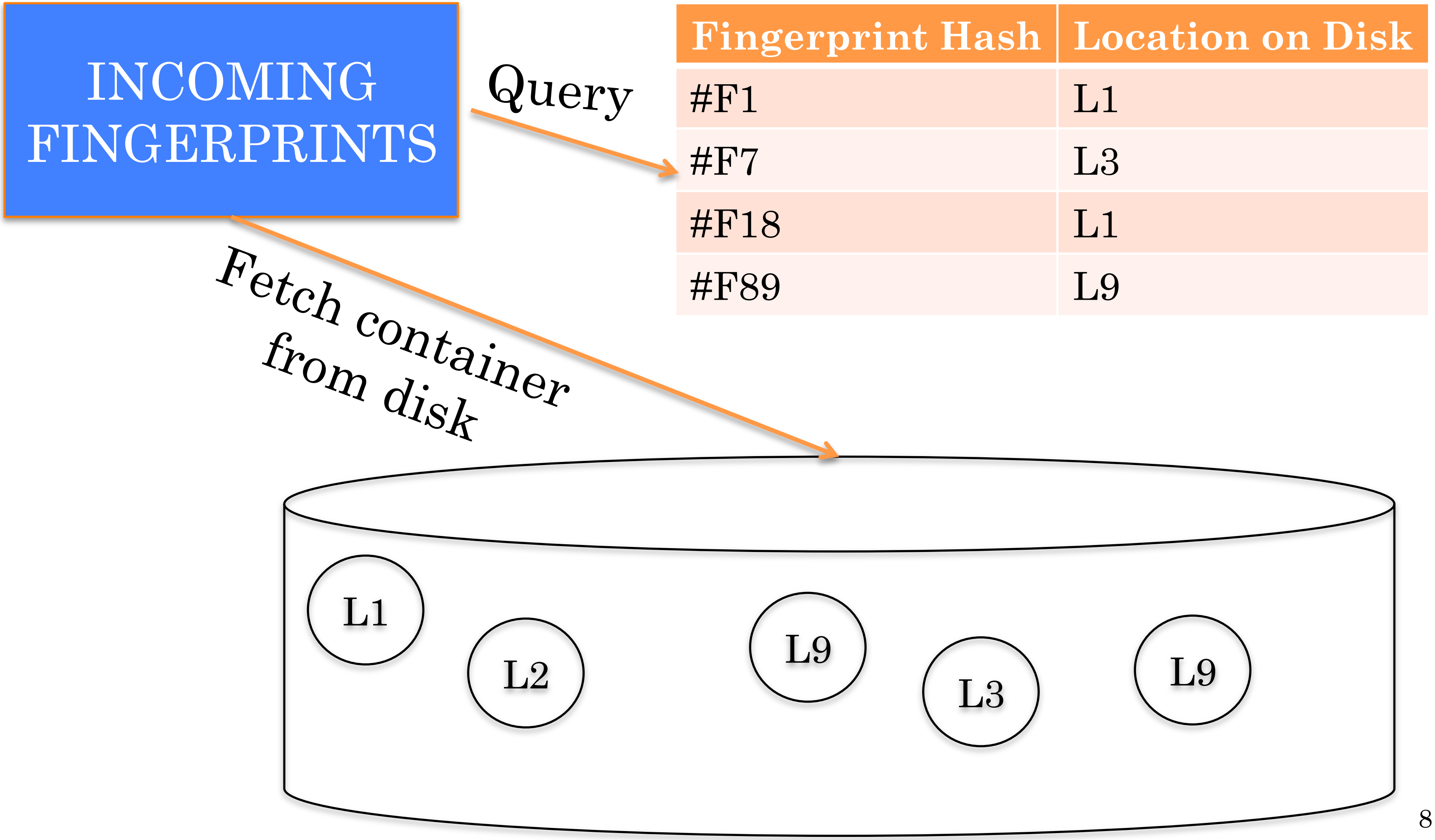
Fingerprint Index Table Size: **4 TB**

Cannot fit in RAM!

FINGERPRINT INDEX

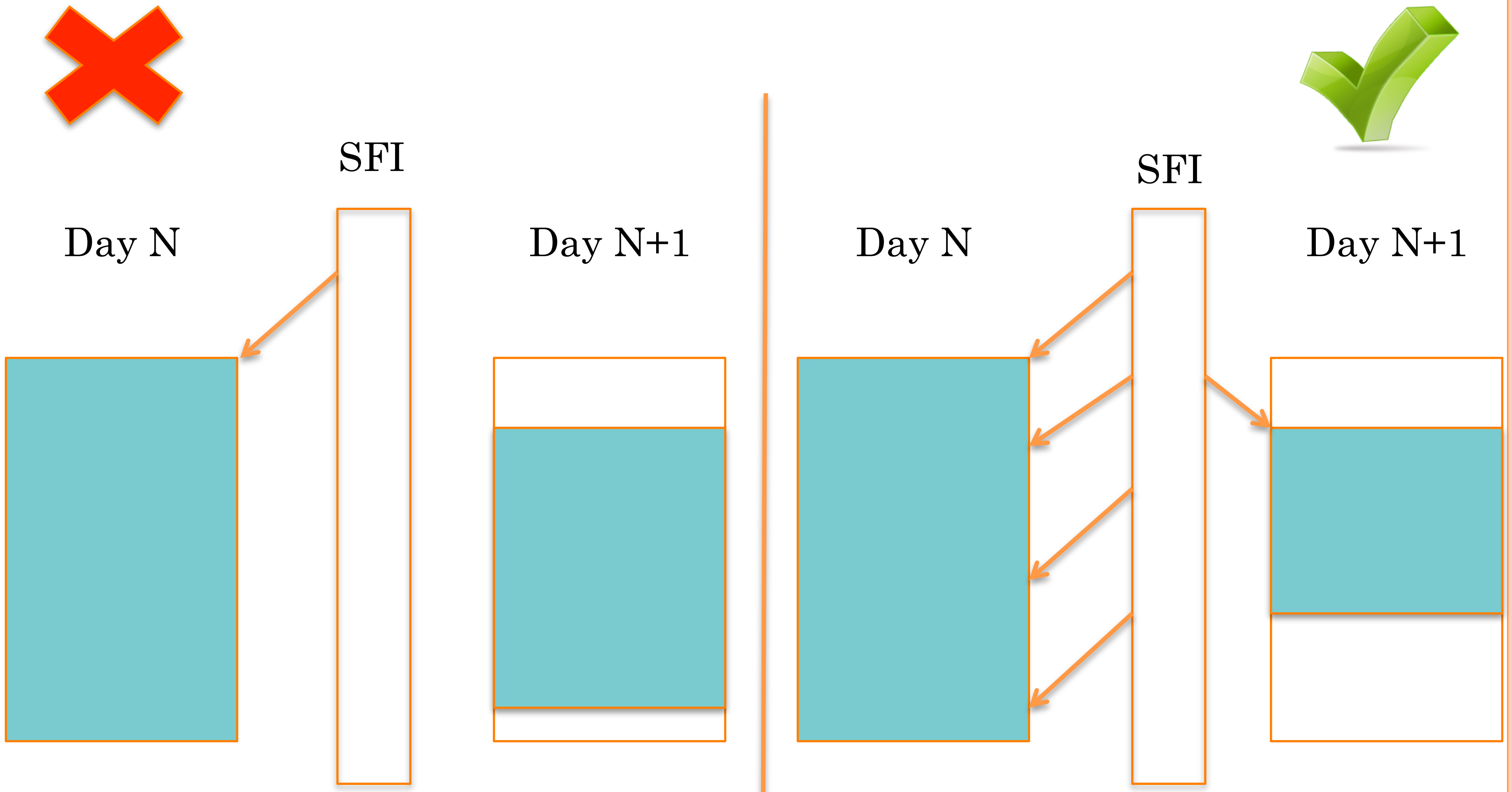
- Can you identify only useful fingerprints and avoid storing less useful fingerprints?
- Is it possible to control the usefulness factor in balancing duplicity and throughput?

SAMPLED FINGERPRINT INDEX: SFI



Fingerprint Hash	Location on Disk
#F1	L1
#F7	L3
#F18	L1
#F89	L9

SFI



SFI

Day N

Day N+1

SFI

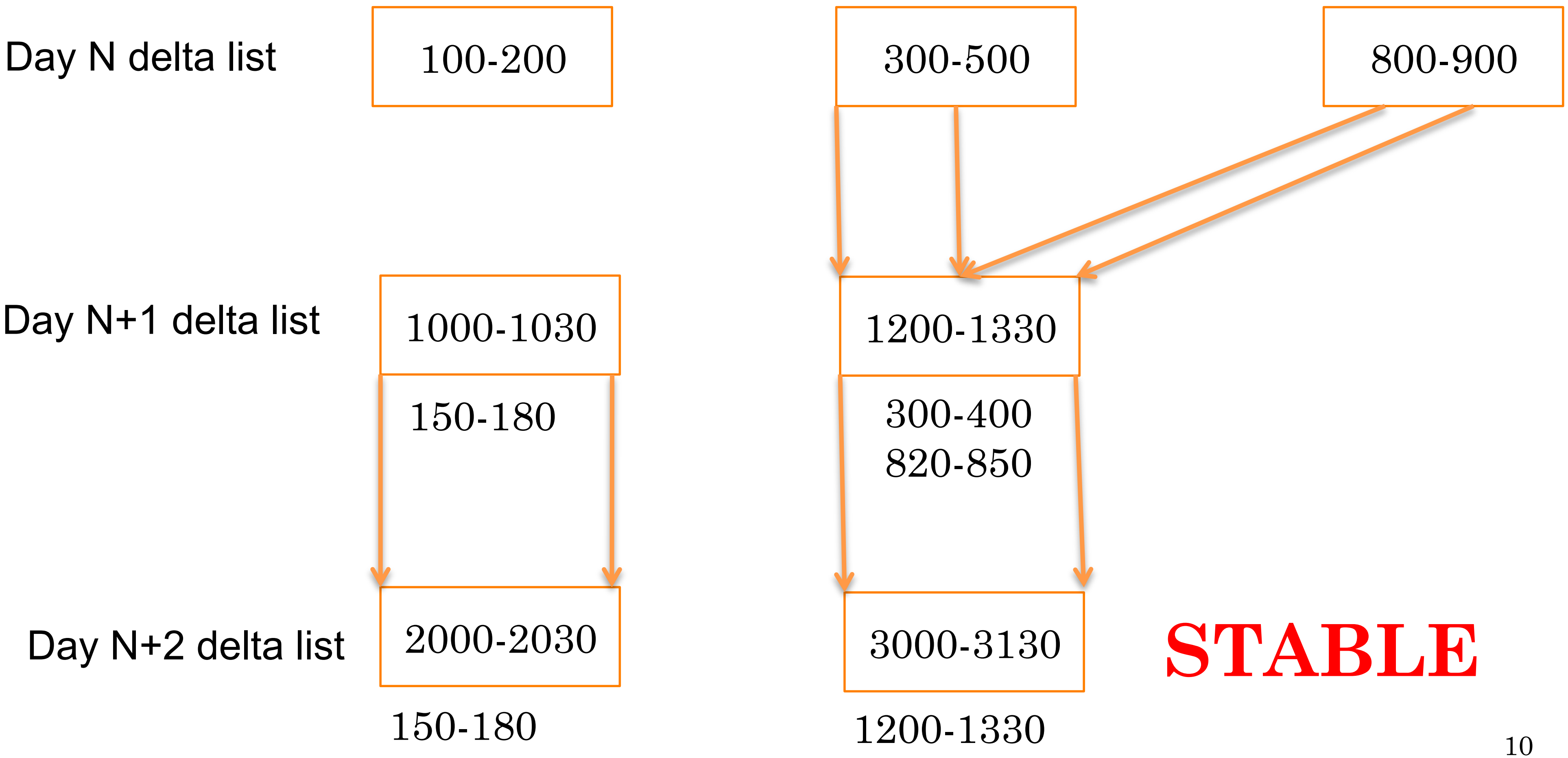
Day N

Day N+1

DUPLICATE NOT FOUND

DUPLICATE FOUND

WHY SAMPLING WORKS?



Day N delta list

100-200

300-500

800-900

Day N+1 delta list

1000-1030

150-180

1200-1330

300-400
820-850

Day N+2 delta list

2000-2030

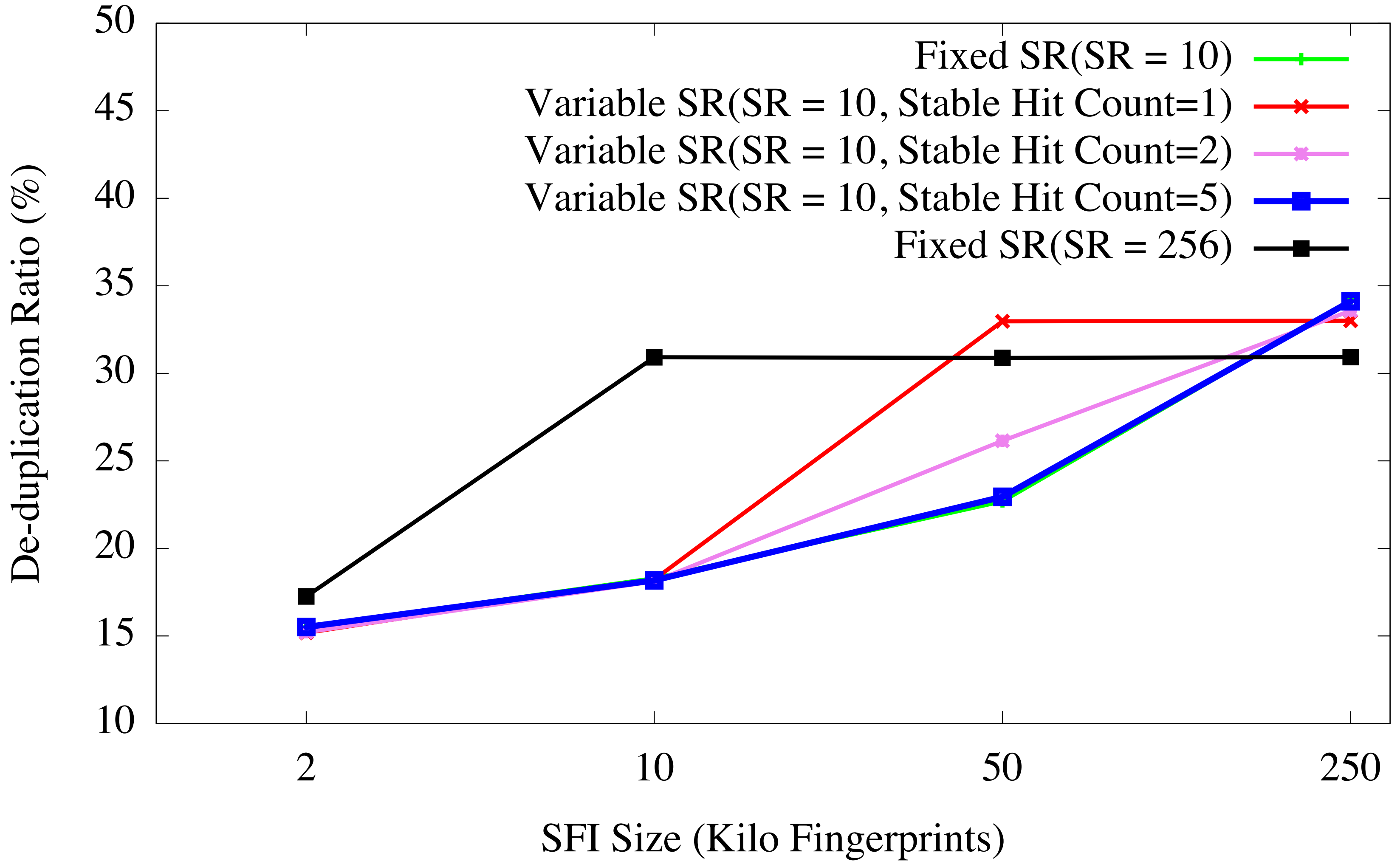
150-180

3000-3130

1200-1330

STABLE

SFI VARIATIONS



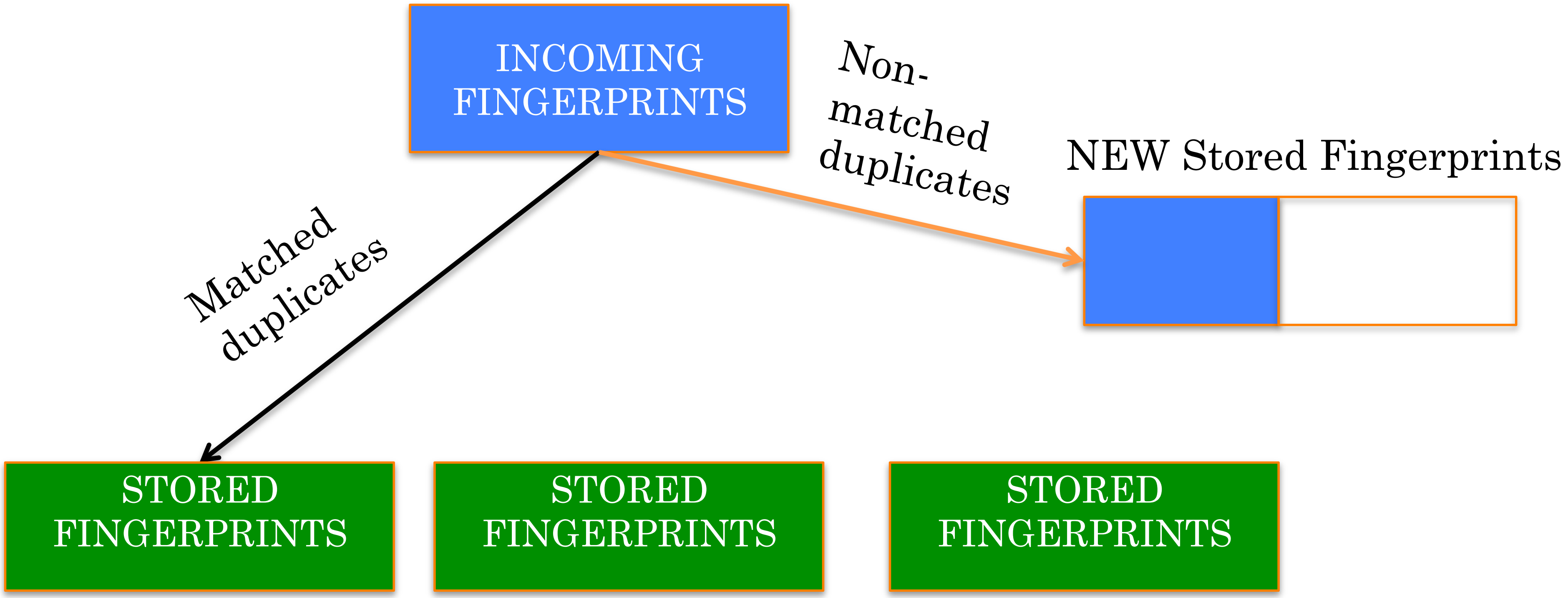
BOTTLENECKS

- Accessing the data disk to fetch fingerprints pointed to by SFI can be very expensive.
- How effective is caching?
  - Assuming repeated usage is one hint.
  - Prefetching is another caching strategy.

PREFETCHING OPTIONS

- Fingerprints are better fetched in a group (containers) and there are multiple options to choose the basis of group formation.
- **Temporal Proximity**: Prefetch with the assumption that fingerprints created at the same time are referred together later.
- **Content Proximity**: Prefetch with the assumption that fingerprints located near each other are referred together later.
- The most important factor that decides the best of these approaches is disk I/O activity.

TEMPORAL PROXIMITY(TP)



INCOMING FINGERPRINTS

Non-matched duplicates

NEW Stored Fingerprints

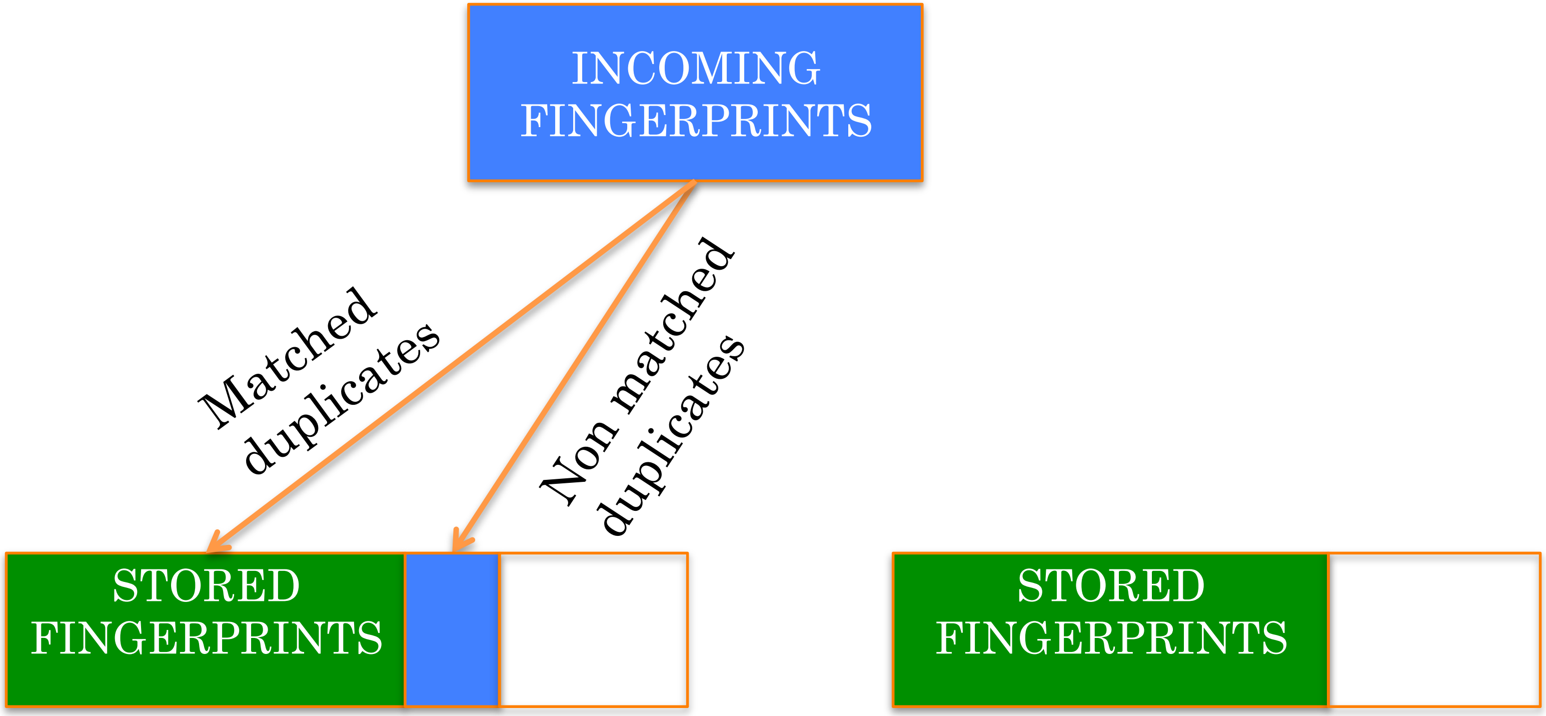
Matched duplicates

STORED FINGERPRINTS

STORED FINGERPRINTS

STORED FINGERPRINTS

CONTENT PROXIMITY (CP)



TP Vs CP

The approach in which ***disk I/Os are minimal*** is the best approach to choose

TP

- Similar to write-optimized file system.
  - Log Structured File System
- Fewer write I/Os
- More read I/Os
- Containers are 100% full

CP

- Similar to read-optimized file system
- More write I/Os
- Fewer read I/Os.
- Containers are X% full to accommodate space for future matches

CP Vs TP Performance

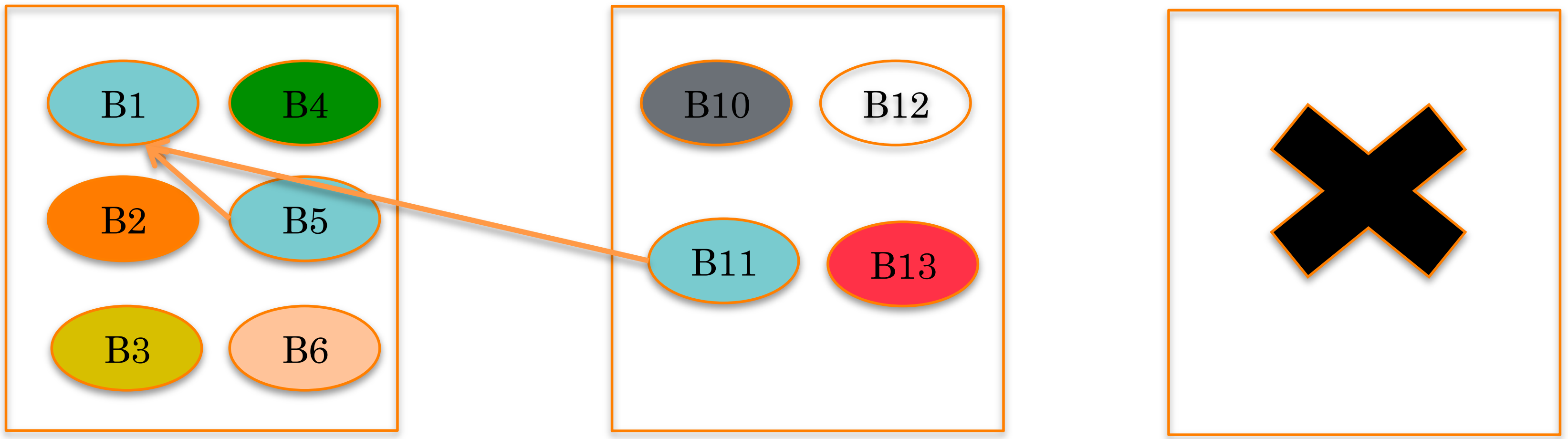
Fill-up Threshold	Dedupe Ratio	Dedupe Throughput	Container Read Count	Container Write Count	Per-Segment Comparison
70	93.11%	282.9K	1.238	0.0743	755
80	93.17%	290.7K	1.248	0.0739	814
90	93.14%	288.9K	1.259	0.0733	809
95	93.16%	287.2K	1.267	0.0733	807
100	93.26%	295.8K	1.264	0.0732	601

TP approach performs marginally better than all other CP variants.

GARBAGE COLLECTION

- Blocks have to be removed from the database:
  - Incoming block is a duplicate.
  - A snapshot retires and the block is not referred by any other snapshot.

Create Snapshot 1 **Create** Snapshot 2 **Delete** Snapshot 1



B1 should be retained because it is still referenced by B11 from snapshot 2

WHY IS GC IMPORTANT

- GC has to maintain some metadata for each block in the backup system to keep track of which block is referred to by blocks in some other snapshot.
- Metadata size exceeds in-memory requirements.
- Same problem of disk I/Os as seen with SFI and containers.
- Mishandling GC can bottleneck Deduplication process.

REFERENCE COUNT GARBAGE COLLECTOR

- **Reference count based method:**
  - Every volume is configured with an expiration time.
  - Every time a snapshot is taken, increase reference count for all blocks in the volume.
  - At the end of expiration time for volume, decrement the reference count for all blocks in volume.
  - All those blocks having reference count = 0, will be freed.
- **Costs:**
  - Fetch metadata for every block in volume every time a snapshot is taken.
  - To free a block, handle the metadata 2 times: One at the time of creating a snapshot and another at the time the snapshot expires.

EXPIRY TIME GARBAGE COLLECTOR

- **Expiry time based method:**
  - Every volume is configured with an expiration time.
  - Every time a snapshot is taken, update the new expiry time for all blocks in the volume to **maximum** of (*current time)* or (*current time + volume expiry time).*
  - No need to update anything when snapshot is deleted.
  - Free all the blocks whose expiry time has passed the current time.
- **Costs:**
  - This is better than reference count method by a factor of 2.
  - Since you do not update all blocks at snapshot expiration time.

HYBRID GARBAGE COLLECTOR

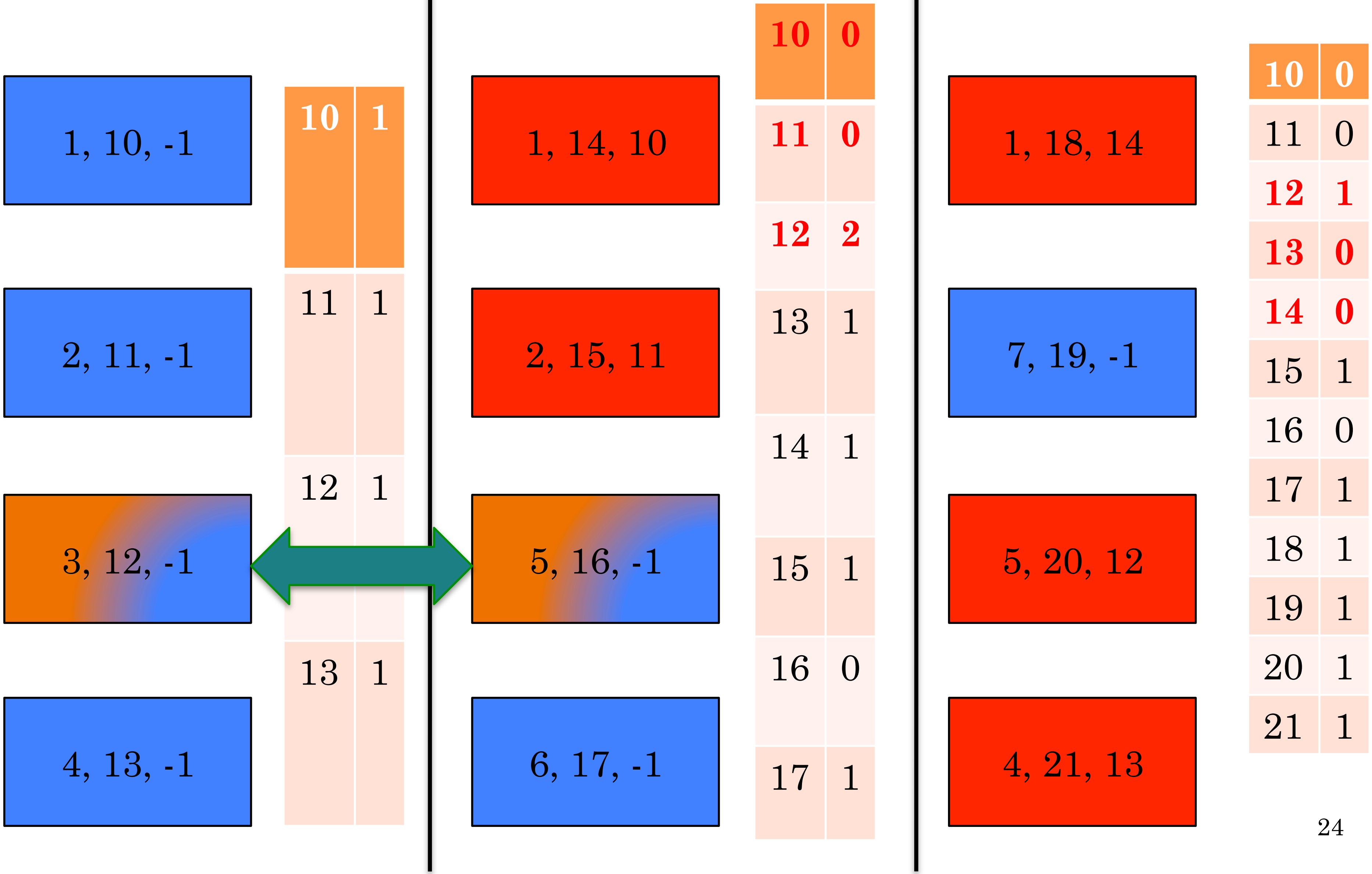
- Each block in delta list has <*LBN, CPBN, BPBN*>
  - *LBN*: Logical Block Number
  - *CPBN*: Current Image Physical Block Number
  - *BPBN*: Before Image Physical Block Number
- *At snapshot creation time, Reference count* for:
  - CPBN is incremented.
  - BPBN is decremented.
- *Expiry time* for BPBN is set to maximum of (current value) or (current time + volume's retention time)
- All blocks whose reference count is 0 are put in a separate queue and are freed when expiry time passes the current time.

Snapshot 1	Snapshot 2	Snapshot 3
🟦 1, 10, -1	🟥 1, 14, 10	🟥 1, 18, 14
🟦 2, 11, -1	🟥 2, 15, 11	🟦 7, 19, -1
🟦 3, 12, -1	🟦 5, 16, -1	🟥 5, 20, 16
🟦 4, 13, -1	🟦 6, 17, -1	🟥 8, 21, -1

Snapshot 1

Snapshot 2

Snapshot 3



Snapshot 1

1, 10, -1

2, 11, -1

3, 12, -1

4, 13, -1

10	1
11	1
12	1
13	1

Snapshot 2

1, 14, 10

2, 15, 11

5, 16, -1

6, 17, -1

10	0
11	0
12	2
13	1
14	1
15	1
16	0
17	1

Snapshot 3

1, 18, 14

7, 19, -1

5, 20, 12

4, 21, 13

10	0
11	0
12	1
13	0
14	0
15	1
16	0
17	1
18	1
19	1
20	1
21	1

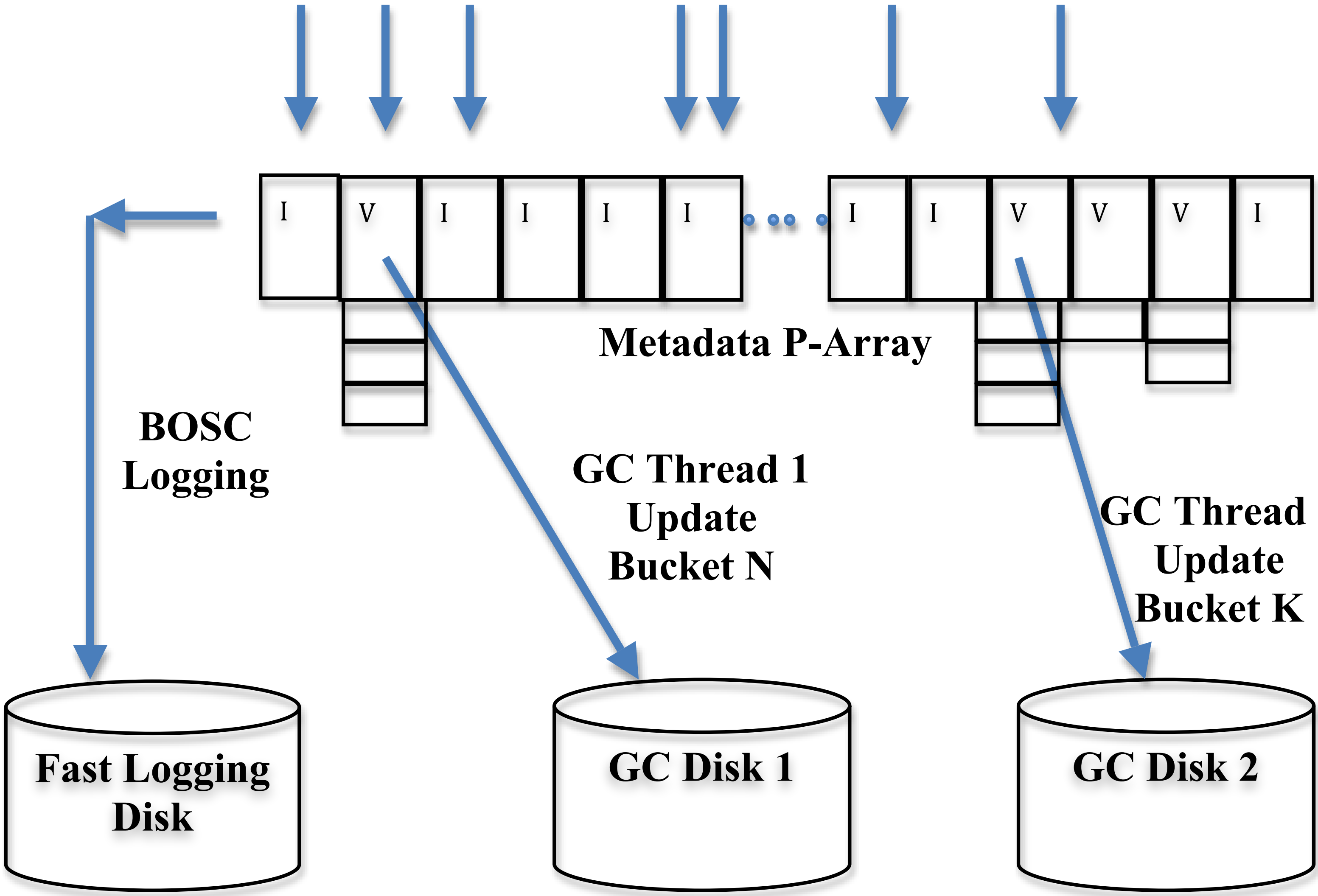
HYBRID GC WORKS!

- Reference Count is updated only for modified blocks in delta list and NOT for all blocks in the volume.
- Expiry time is checked only on the blocks that have reference count 0 and that are put in a separate queue.
- Metadata is not updated for any blocks when snapshot is expired.
- Blocks that never get modified, will have reference count > 1 and will never get garbage collected as it's still in use by atleast 1 current image.
- Scalable because delta list is typically much much smaller than the entire volume.

IMPLEMENTING GC

- Managing metadata updates in GC is non-trivial because of very low locality
  - Already existing blocks and the incoming duplicates have hardly any dependency.
- Use BOSC scheme to batch the updates and sequentially commit the batched updates to disk periodically.
  - Use TRAIL Logging to ensure data persistency

Incoming Fingerprints from Sungem to be updated in GC



Metadata P-Array

BOSC Logging

GC Thread 1 Update Bucket N

GC Thread 2 Update Bucket K

Fast Logging Disk

GC Disk 1

GC Disk 2

GC's INFLUENCE

Commit Threads	Dedupe +Vanilla GC	Dedupe +BOSC GC	Dedupe & NO GC
1	5879	54047	287204
2	6003	268218	287204
4	9858	277670	287204
10	8121	269272	287204

severe bottleneck

very much comparable

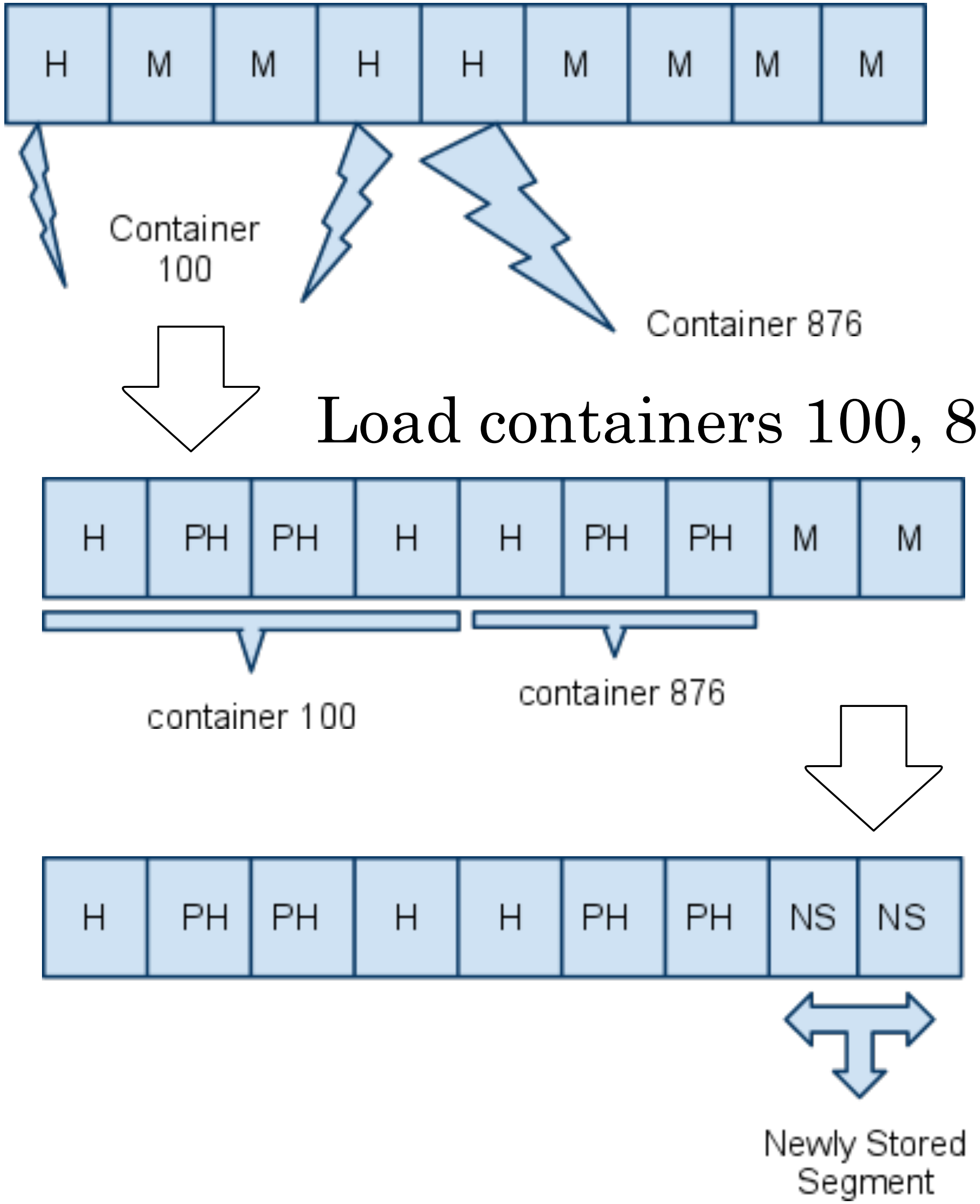
FINGERPRINTS PROCESSING

SFI Lookup

Load HIT containers

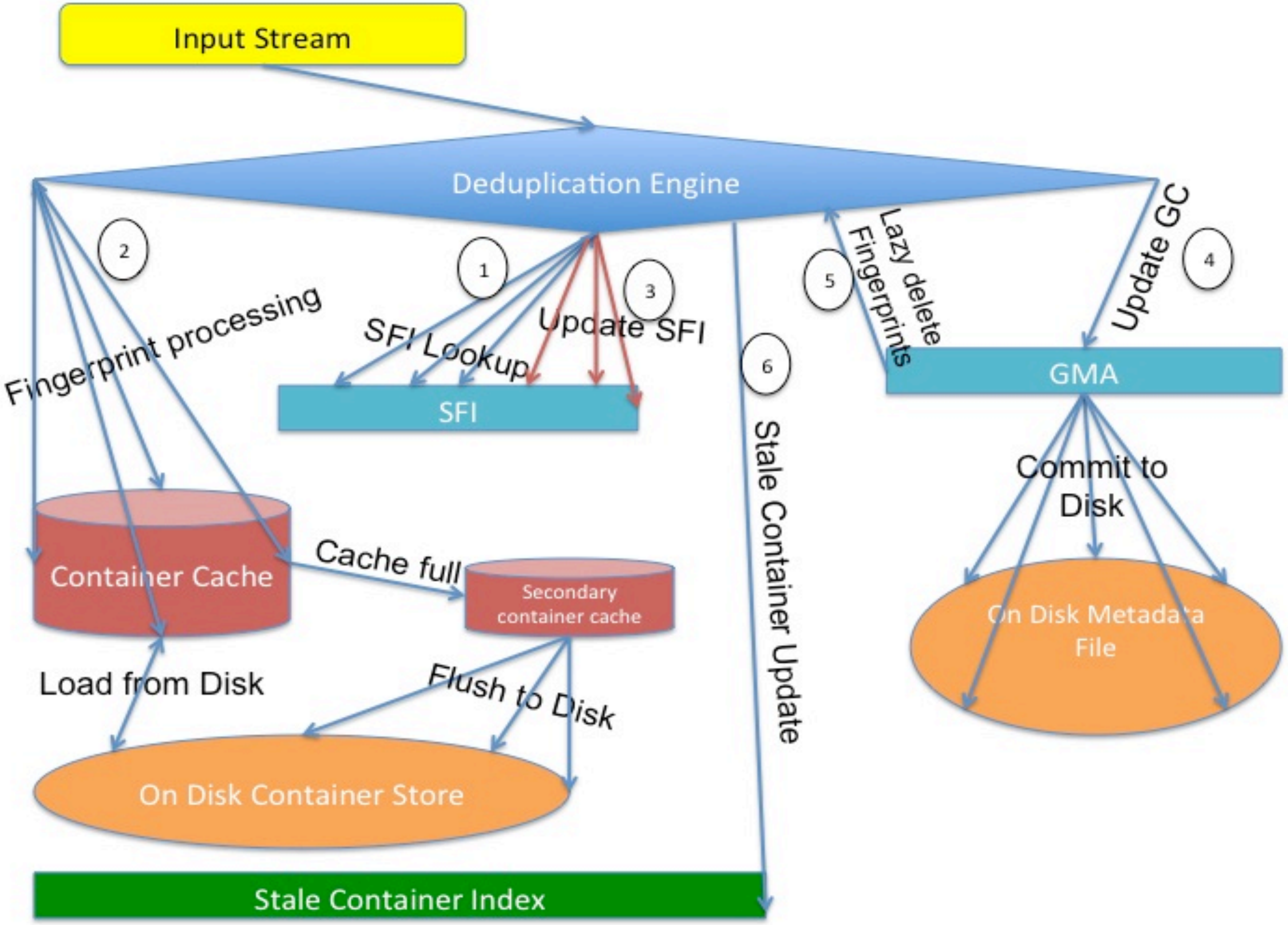
Process HIT fingerprints and convert some MISS to Pseudo HITS(PH)

Process MISS fingerprints and store in nearest container: 876

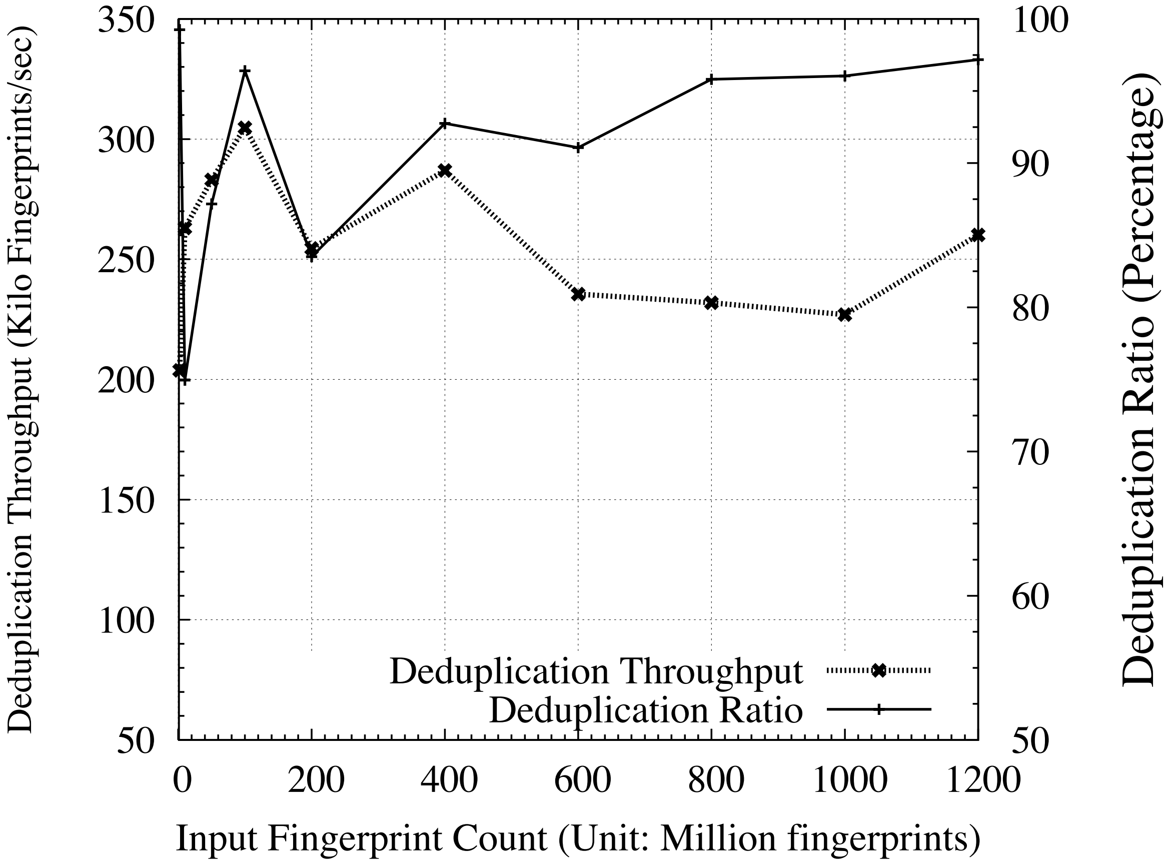


Load containers 100, 876

DEDUPLICATION AND GC OVERALL VIEW



SUSTAINED HIGH PERFORMANCE



SUMMARY

- Supports very high throughput across all ranges of deduplication ratios.
- Supports dynamic sampling rate to optimally store SFI without hurting the deduplication ratio.
- In depth comparison of TP and CP approaches to store containers.
- Scalable GC technique which scales only with changed data and NOT the entire volume size.