VM-Centric Snapshot Deduplication for Cloud Data Backup

Wei Zhang — *Pure Storage Inc.*
Daniel Agun, Tao Yang, Rich Wolski — *University of California, Santa Barbara*
Hong Tang — *Alibaba Inc.*

Alibaba.com

Overview

- Problem Statement and Motivation
- Architecture Overview
- Multi-level Selective Approach for Inline Deduplication
- VM-centric Storage Management with Approximate Deletion

UCSB

Background

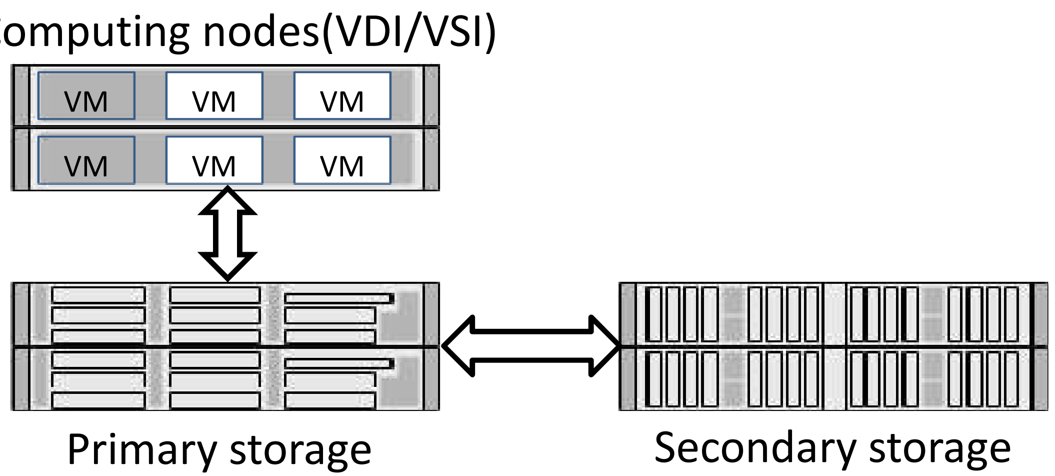
- **Virtual machines on the cloud uses frequent backup to improve service reliability**
  - Automatic or on-demand snapshot
- **Snapshot**
  - Preserves the data of a virtual machine disk at a specific point in time
- **High storage demand**
  - In Alibaba.com
    - Total num. of VMs: 500k+
    - Num. of VMs per cluster: 25k+
    - Daily backup workload: 1,000+TB

Motivation for Deduplication

- **Huge storage and network demand from snapshot backup**
  - Disk to store backup data
  - Network to transfer backup data
- **Ubiquitous duplicates among backup versions and virtual machines**
  - Most of the data don’t change during a backup period
  - Different VMs contain largely similar data: OS, apps
  - Tremendous data reduction: 1:20 ~ 1:50

Dedup in Dedicated Architecture

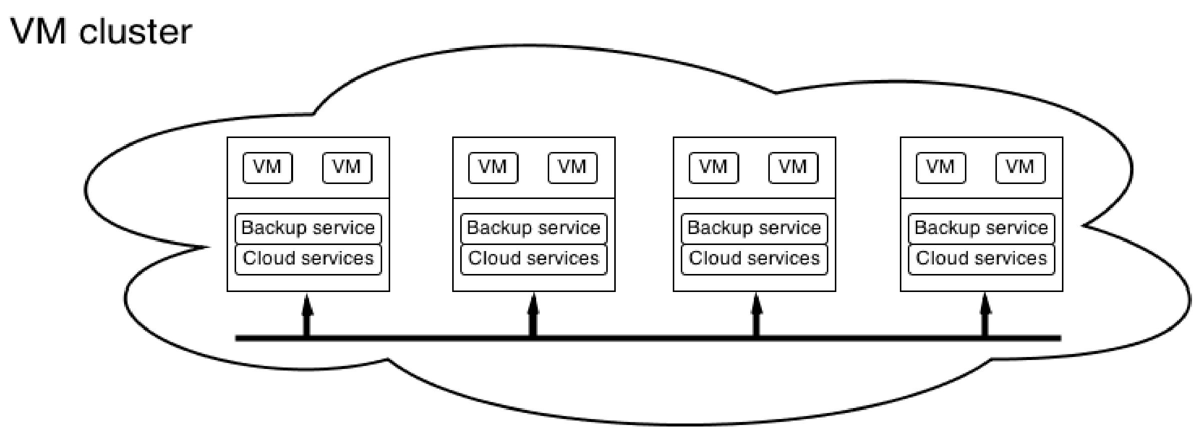
- **Incremental backups send to dedicated backup storage**
- **Changed Block Tracking (CBT)**
  - Virtual device driver can tell which disk regions have been changed



- **Pros**
  - Fast
  - Simple
  - Reliable
- **Cons**
  - Only removes 70%~80% of duplicates locally
  - High I/O and storage demand
  - Very expensive infrastructure cost

Dedup for Converged Architecture

- **A decentralized backup service collocated with VM cluster**



- **Requirements**
  - Cost consciousness
    - At Alibaba, CPU and disk usage available is 10% of total resource
  - High backup throughput
    - Dedup and backup for 10,000+ users within a few hours each day during light cloud workload.
  - Fault tolerance constraint
    - Addition of data deduplication should not decrease the degree of fault tolerance.

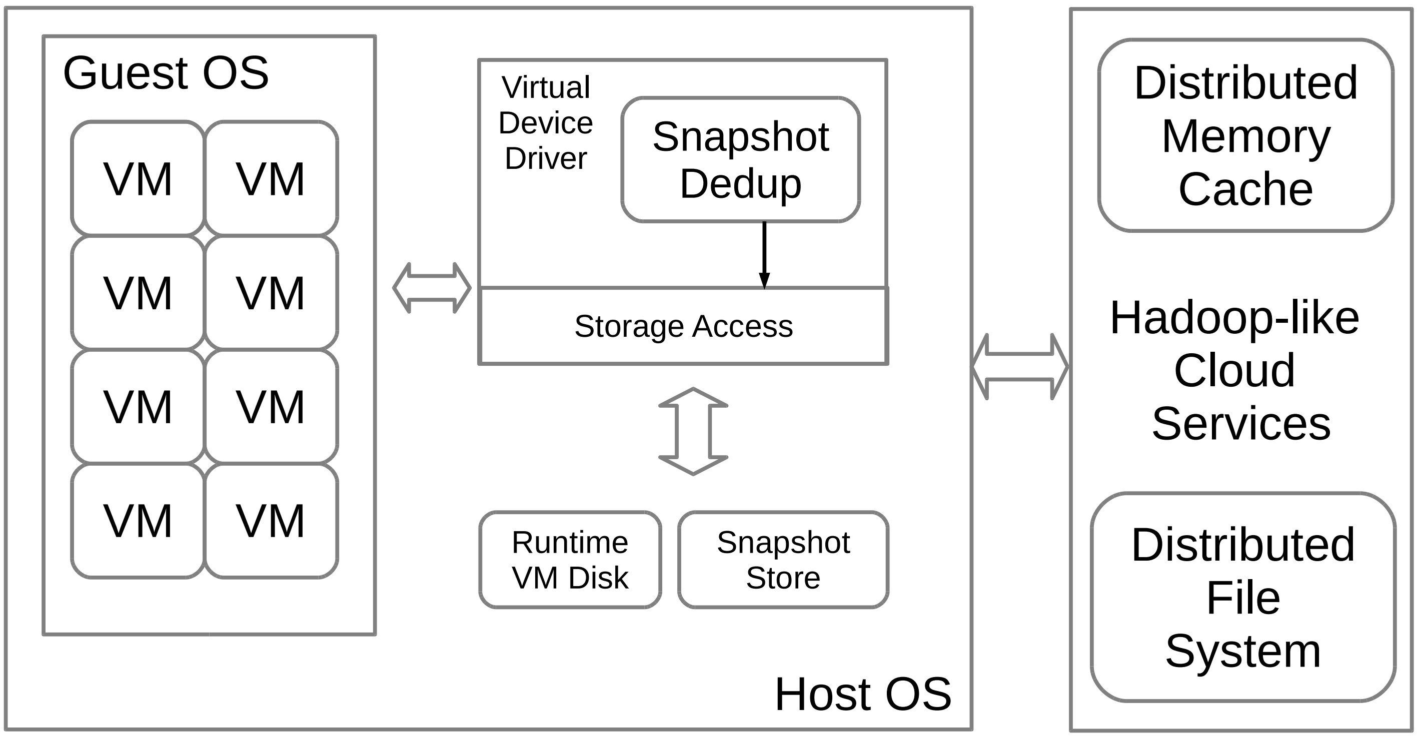
UCSB

Challenges in Converged Architecture

- **Challenges**
  - Limited resource as a collocated data service
  - GC is expensive
- **False-negative detection methods**
  1. Multi-level Selective Dedup
     - Similarity-guided local dedup
     - Popularity-guided global dedup
  2. VM-centric Data Management
     - Approximate Deletion
- **Goal**
  - Low-cost, scalable, high throughput

UCSB

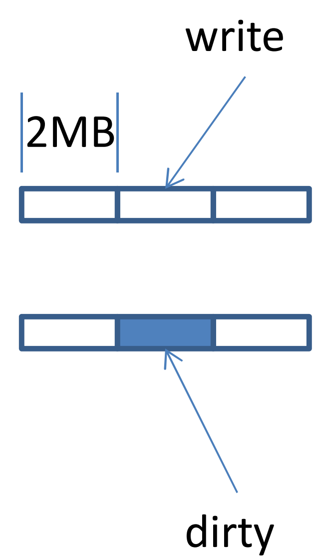
Architecture Overview



UCSB

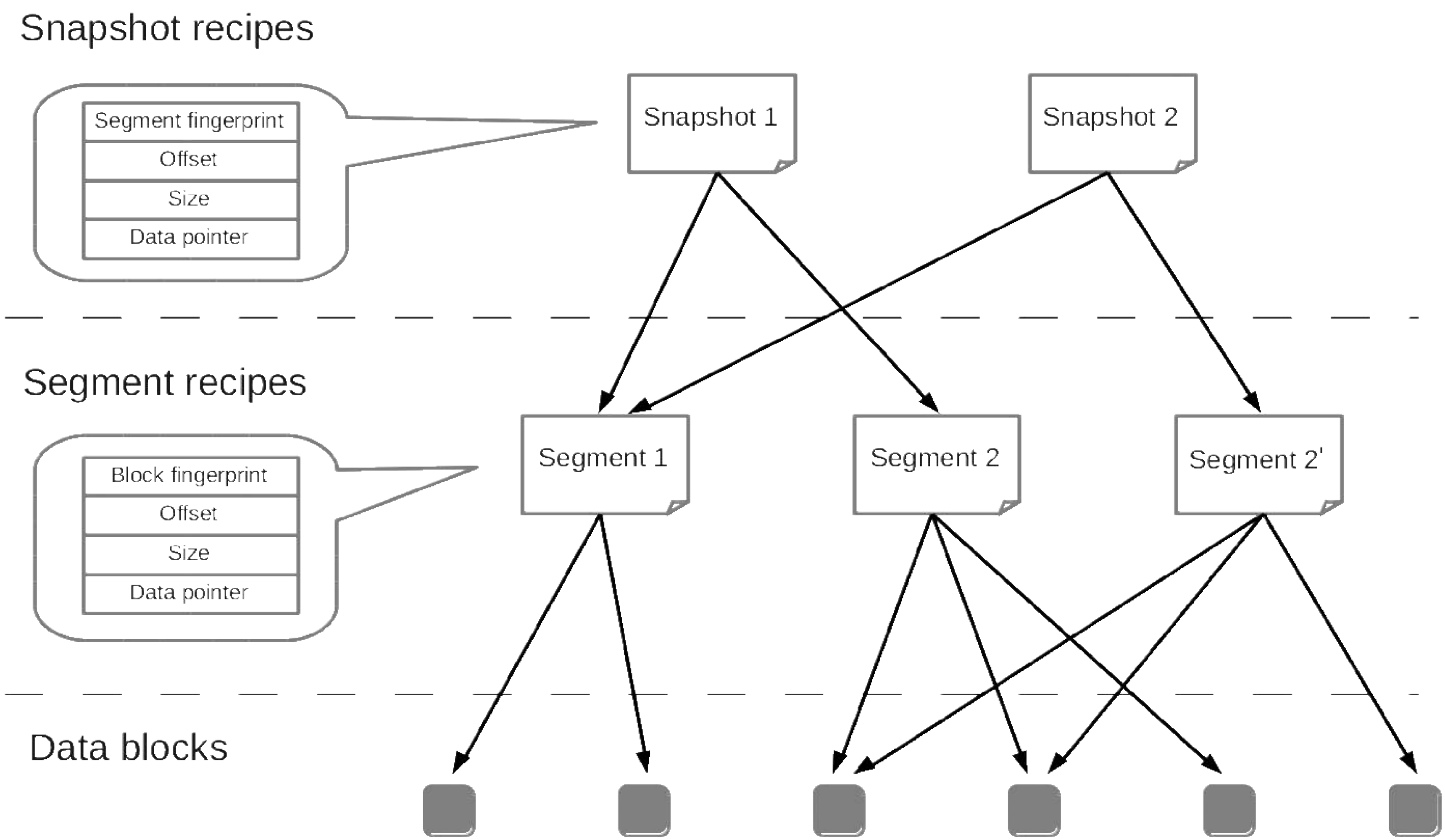
Baseline: Changed Block Tracking

- **Widely used in VM snapshot backup**
  - Available in different forms by major VM vendors: VMware, Microsoft, Xen
  - Track disk changes at segment level base on the VM I/O activity
  - Low-cost but not very efficient
- **Reason**
  - 2MB Segment is not entirely dirty
  - OS/fs/app/user may move data around



UCSB

Metadata: Snapshot Representation



Multi-level Selective Approach

- **Trade dedup efficiency for fast job completion**
- **Challenges**
  - Instant dedup and backup
  - Resource friendly
  - No compromise on fault tolerance
  - Fast deletion
- **Inline on-demand snapshot backup**
  - Similarity-guided Local Deduplication
  - Popularity-guided Global Deduplication

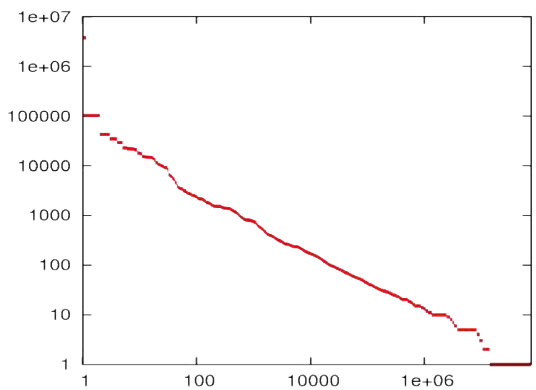
Similarity-guided Local Deduplication

- **Standard changed block tracking**
  - Track disk changes at segment level base on the VM I/O
  - Low-cost but not very efficient
- **Disadvantages**
  - 2MB Segment is not entirely dirty
  - OS/fs/app/user may move data around
- **Solution**
  - Fine-grained deduplication inside segments
    - Break 2MB segment into 4KB average chunks
  - Similarity-based content tracking
    - Track changed segments base on similarity metrics
    - Min-hash based similarity detection is proven efficient

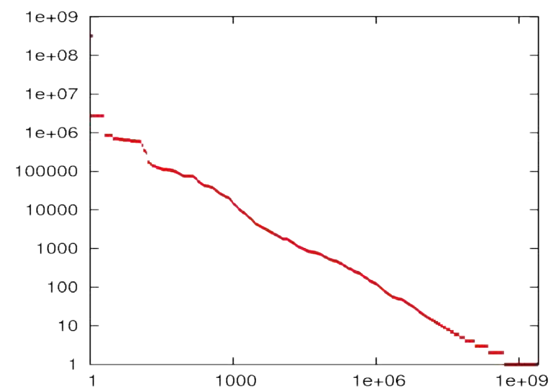
UCSB

Popularity-guided Global Deduplication

- **Different VMs contain largely duplicate data**
  - OS, apps
- **Key observations**
  - Exploit popular data to reduce resource need
  - Zipf-like distribution of VM OS/user data
    - frequency of any chunk is inversely proportional to its rank in the frequency table



(a) Data blocks from OS disks



(b) Data blocks from data disks

UCSB

Popularity-guided Global Deduplication

- **Indication of Zipf-like distribution**
  - A small portion of popular data represent the majority of global duplicates
- **Collecting PDS (popular data set)**
  - Periodically run reference counting on the fingerprint of stored chunks via map-reduce
- **Storing PDS**
  - Use distributed share memory to store PDS index
  - Data are stored in distributed file system
    - Referenced by everyone

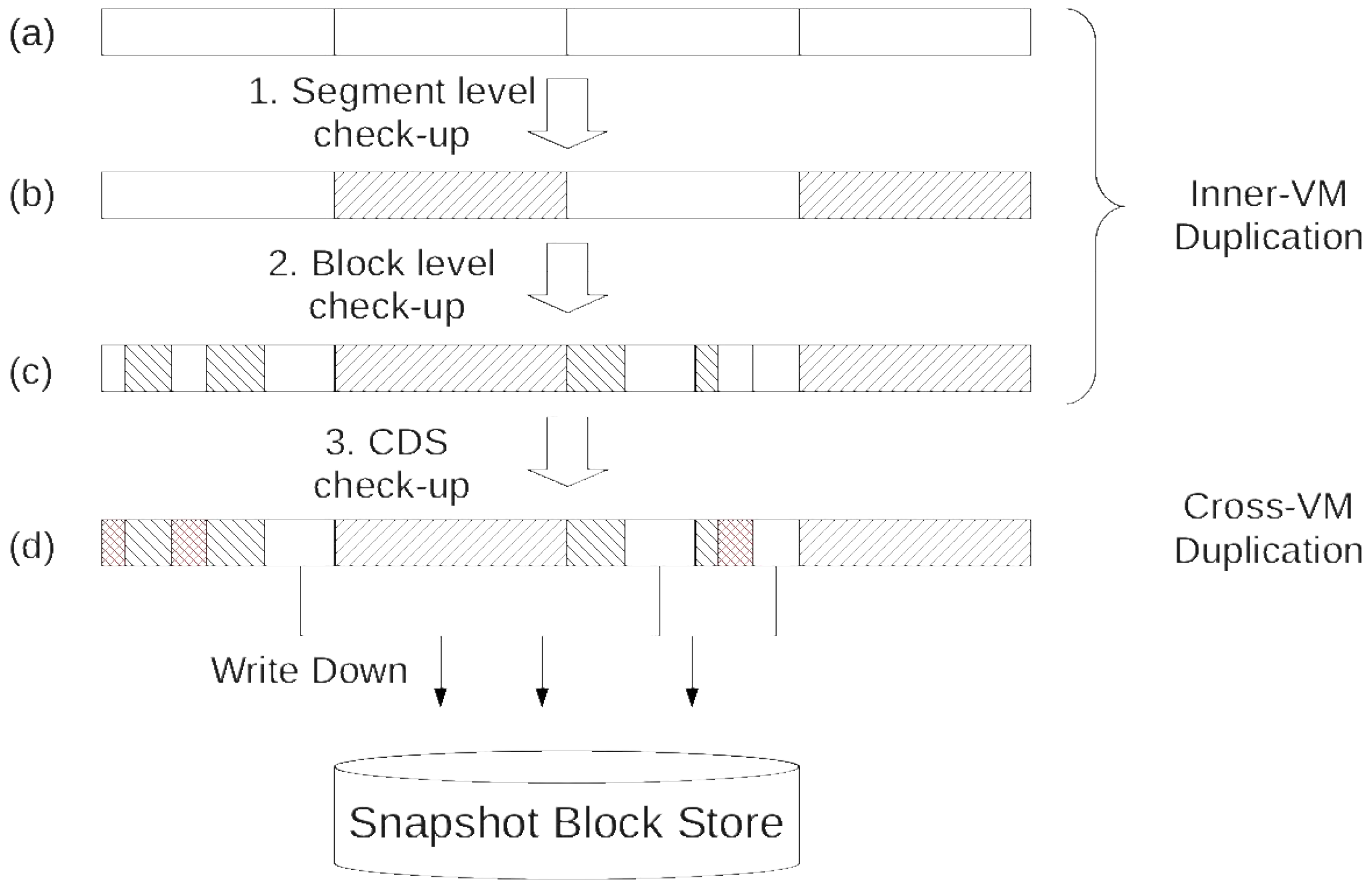
UCSB

Multi-level Data Processing Steps

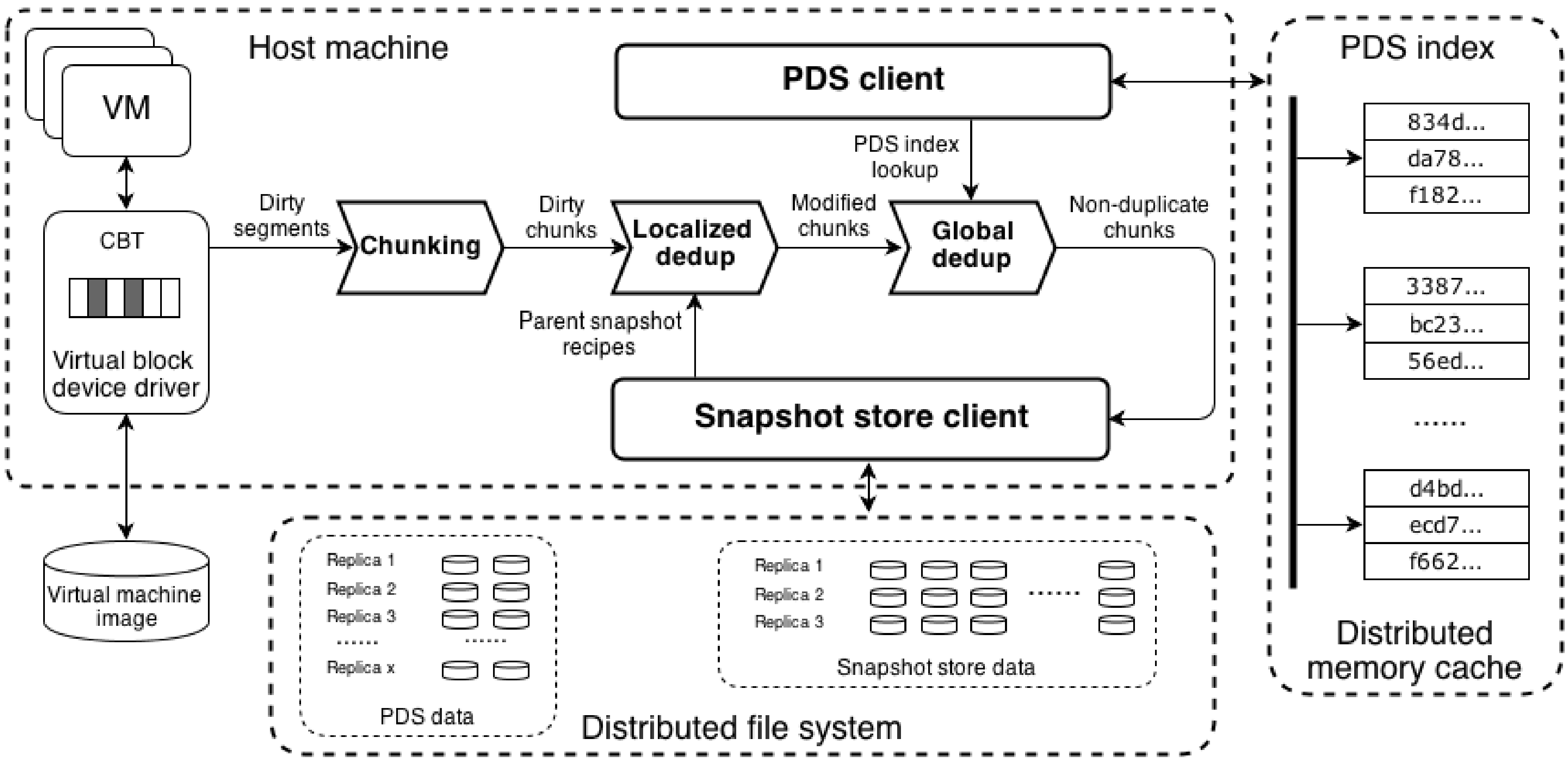
- **Segment level dedup**
  - Use dirty bitmap to see which segments are modified
- **Chunk level dedup**
  - Divide a segment into variable-sized blocks, and compare their fingerprints with the parent snapshot
- **Global dedup from common dataset (PDS)**
  - Identify duplicate chunks from PDS
- **Write new snapshot blocks**
  - Write new content chunks to snapshot store
- **Save metadata**
  - Write segment and snapshot recipes

UCSB

Processing Flow of Multi-level Deduplication



Architecture



UCSB

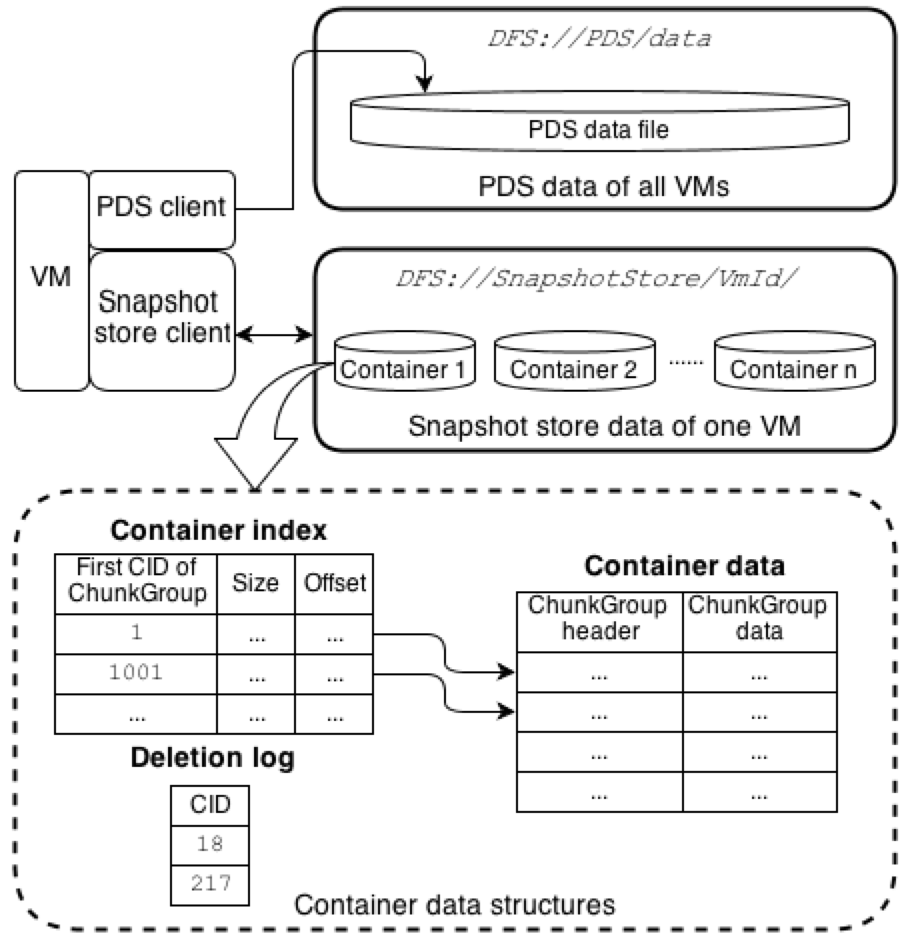
Snapshot Store

- **Design goals**
  - VM-centric: each VM has its own snapshot store
  - Small memory footprint
  - Efficient append-only writes and sequential reads
- **Operations**
  - Append
  - Get
  - Delete

UCSB

Snapshot Store

- **Log-structured data layout**
- **Append-only operations**
  - Monotonic increasing chunk ID (8 bytes)
  - Grouped reads/writes
  - Self-sorted block index
  - Delete is compaction



UCSB

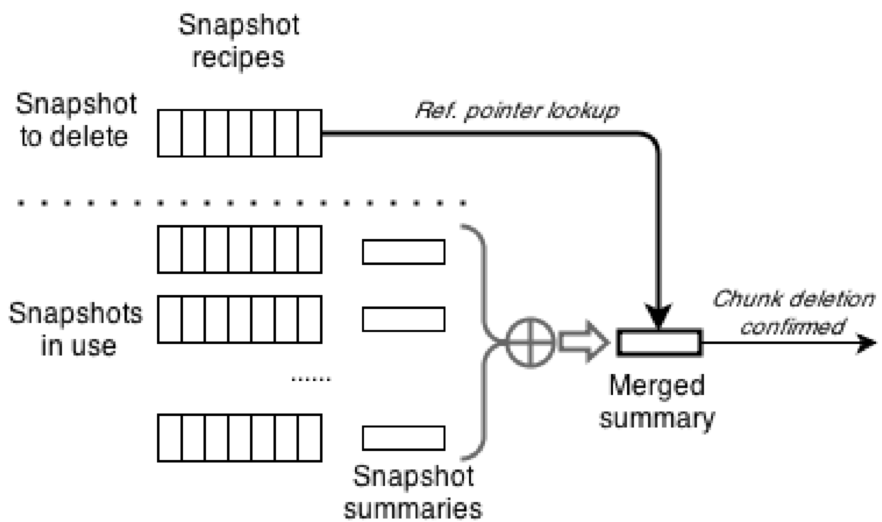
Approximate Deletion

- **Snapshot deletion is as frequent as creation**
- **Identifying unused chunks is difficult**
  - Reference Counting: costly, unsafe
  - Mark-and-Sweep: better than RC, still costly
- **Idea**
  - Tolerate small percentage of storage leak to allow fast deletion

UCSB

Approximate Deletion

- **Summary vector**
  - Use bloom filter to summarize every snapshot
  - Merged summary vectors of live snapshots represents the chunks that are in use
  - Checking the existence of a chunk is fast



UCSB

Leakage Analysis and Repair

- **Periodically repair with mark-and-sweep after *R* rounds**
- **Total leakage:** $L = R\varepsilon\Delta u$
- **Total blocks stored:** $U = u + (h-1)\Delta u$
- **How many rounds of approximate deletion need one repair?**

$$\frac{L}{U} = \frac{R\Delta u\varepsilon}{u+(h-1)\Delta u} > \tau \Longrightarrow R > \frac{\tau}{\varepsilon} \times \frac{u+(h-1)\Delta u}{\Delta u}$$

- **Daily VM change rate:** *Δu/u ~ 2% - 4%*

UCSB

Example

- **Sampled weekly VM disk change rate**

Cluster name	Avg. system disk change rate	Avg. data disk change rate
AY41A (4224 VMs)	17.29%	15.05%
AY41C (2083 VMs)	16.64%	16.61%
AY41D (2966 VMs)	16.42%	12.86%
AY41E (5603 VMs)	17.83%	21.85%

- **Example**
  - *let $\Delta u/u = 2.5\%$, $\mathcal{E} = 0.01$, $r = 0.05$.*
  - *$R = 245$ rounds*
  - *25 VMs per node, one repair scheduled for every 9.8 days*

UCSB

Comparison

	Approximate deletion with repair	Perfect hashing (Fabiano et. al FAST '13)	Grouped mark-and-sweep (Guo et.al ATC'11)
Memory	~10 MB	~GB	~100 MB
Time	~1 minute	~hours	~10 minutes
Scan metadata?	N	Y	Y
Tracking metadata to data dependency?	N	N	Y
Leakage?	Y	N	N

Implementation & Evaluation Settings

- **Prototype system running on Alibaba's Pangu FS and open-source QFS**
  - Based on Xen. 100 nodes and each has 16 cores, 48G memory, 25VMs.
  - Use <150MB per machine for backup & deduplication
  - Popularity is computed by using 90% of dataset. Re-compute PDS every 1-2 days to catch up the popularity trend
  - Segment size: 2MB. Avg. Block size: 4KB

UCSB

Implementation & Evaluation Settings

- **VM snapshot data collected from Alibaba**
  - Each VM uses 40GB storage space on average
  - OS and user data disks: each takes ~50% of space
  - OS data
    - 7 mainstream OS releases:
    - Debian, Ubuntu, Redhat, CentOS, Win2003 32bit, win2003 64 bit and win2008 64 bit.
  - User data
    - From 1323 VM users

UCSB

Comparison Targets

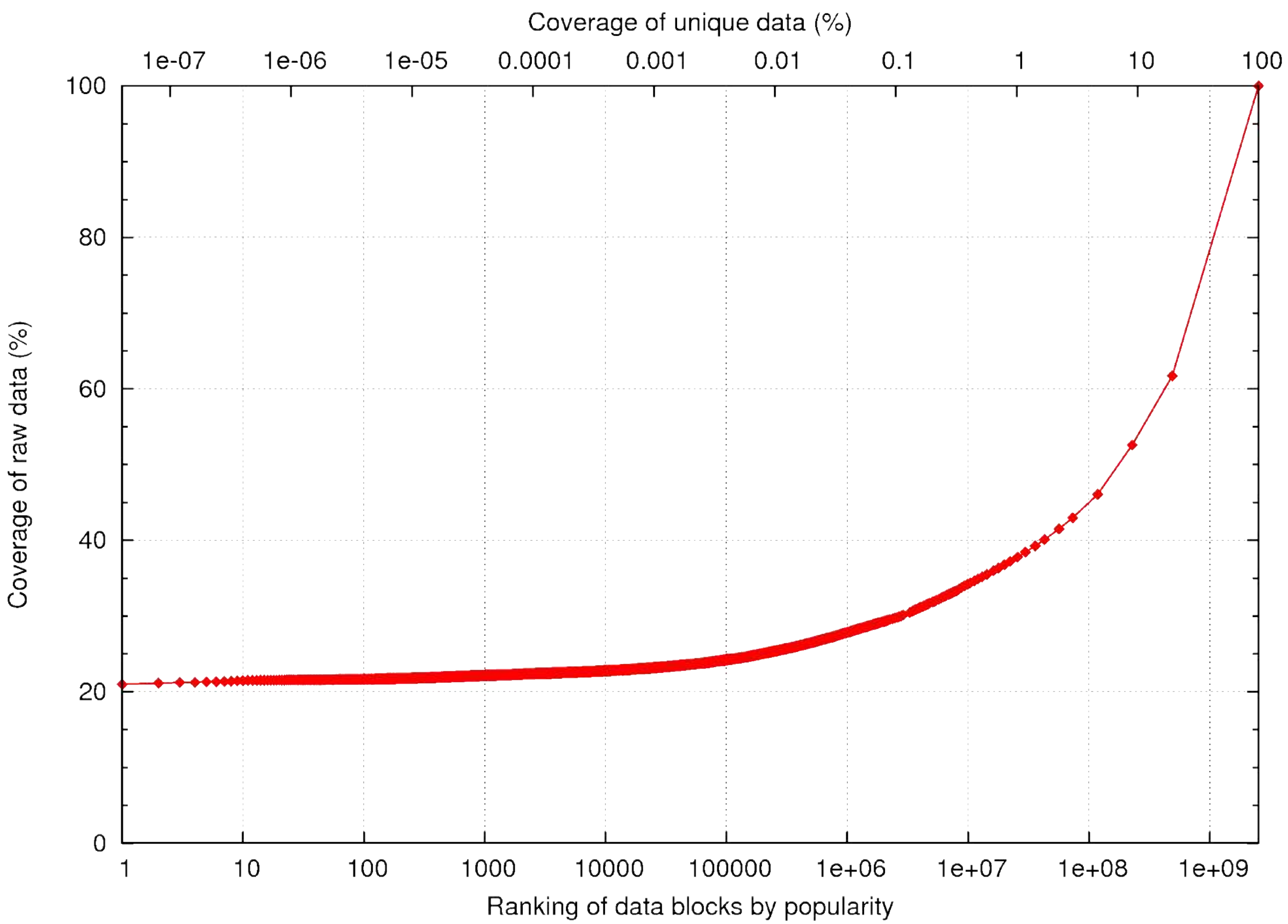
- **VM oblivious (VO)**
  - A theoretical model that all data are perfectly deduplicated and then stored on DFS
  - Sequential layout as data arrives
- **Similarity based Stateless routing (SRB)**
  - Segments are sent to similar data groups
  - Perfect deduplication within data group
  - No deduplication across data groups
  - Each node works on a partition of data groups

UCSB

Cumulative coverage of popular data

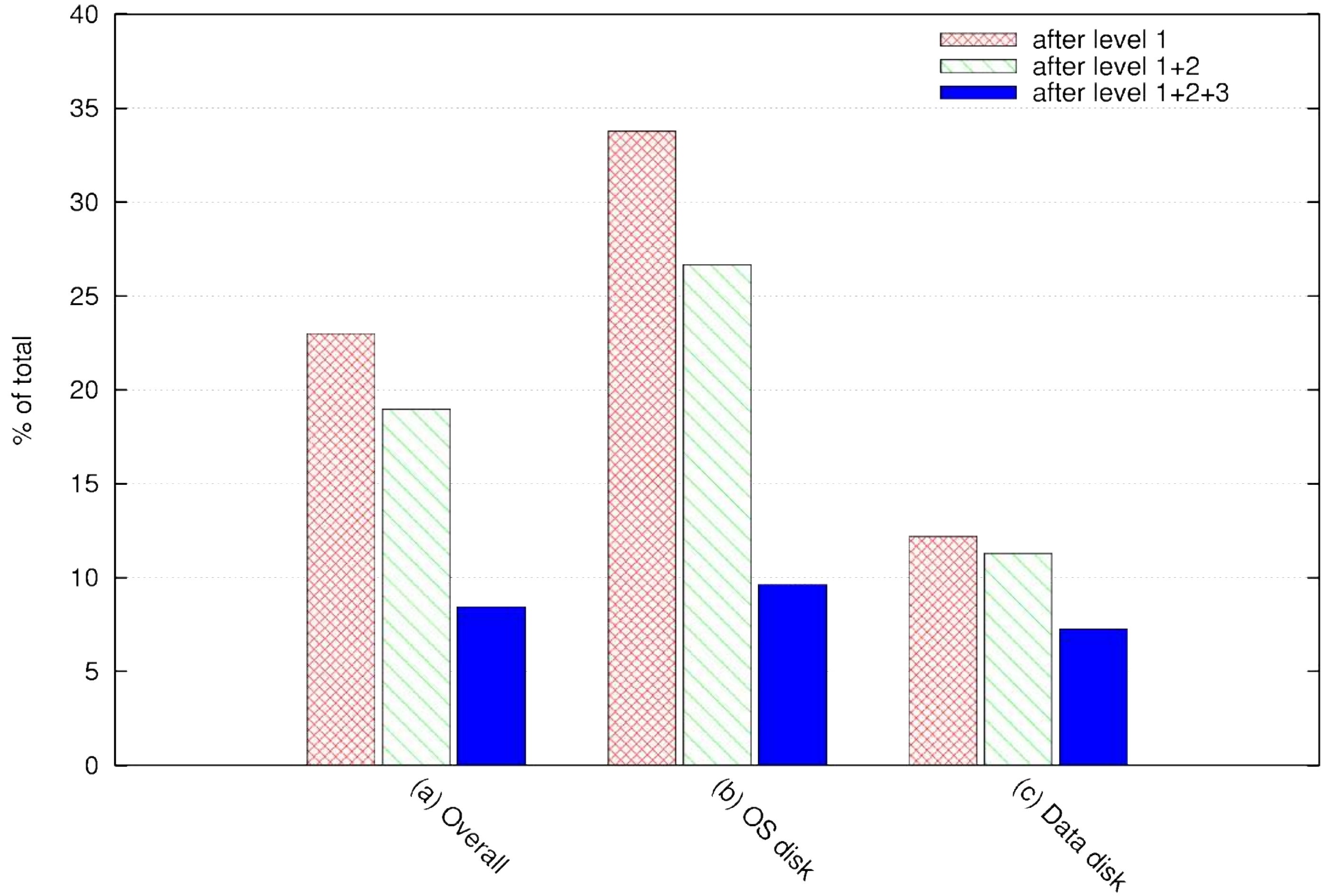
Coverage is the summation of covered data block size*frequency

$$\frac{\sum_{i=1}^{i} S_i * F_i}{\text{Total data size}}.$$

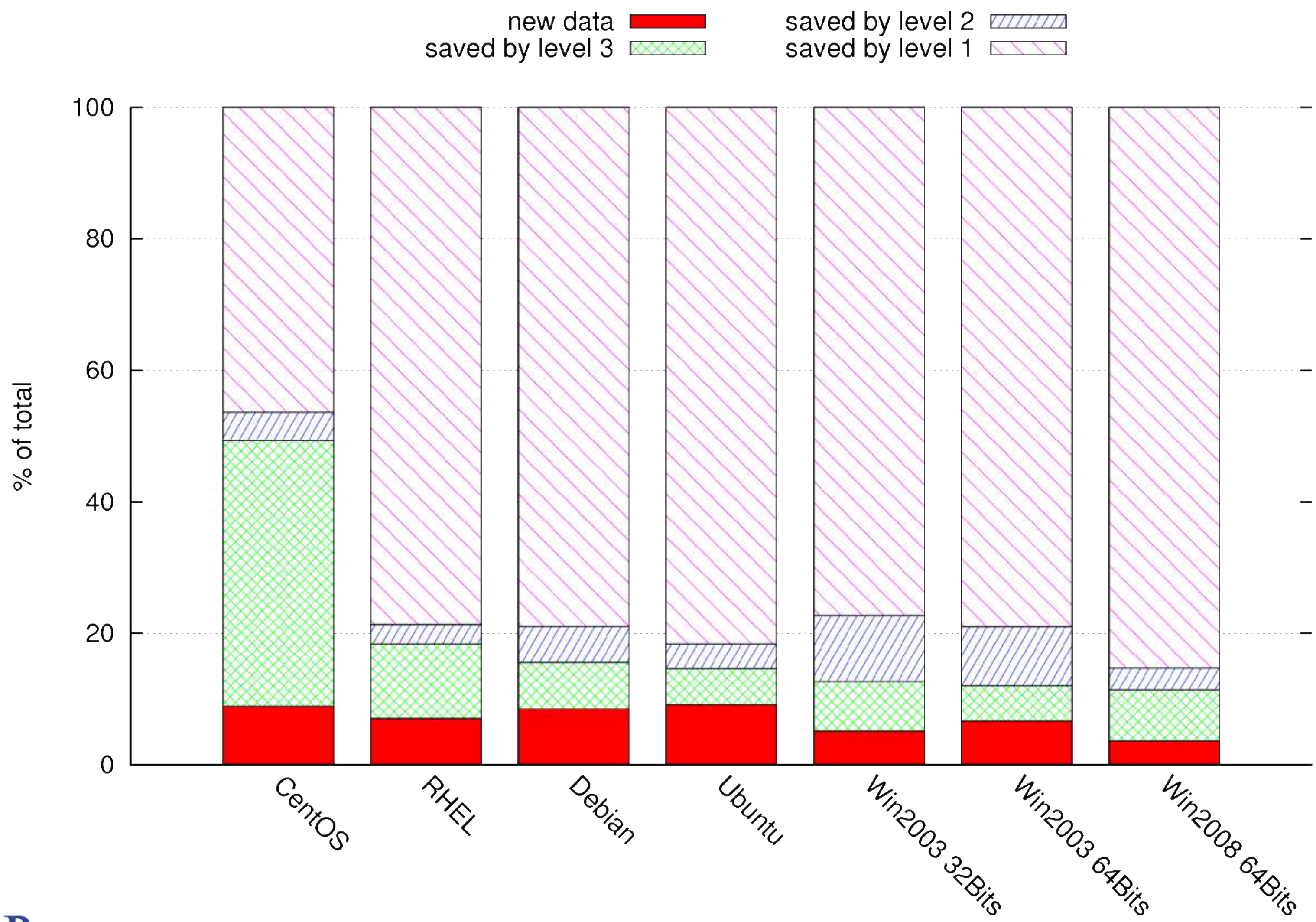


Impacts of 3-Level Deduplication

Level 1: Segment-level detection within VM
Level 2: Block-level detection within VM
Level 3: PDS detection across-VM



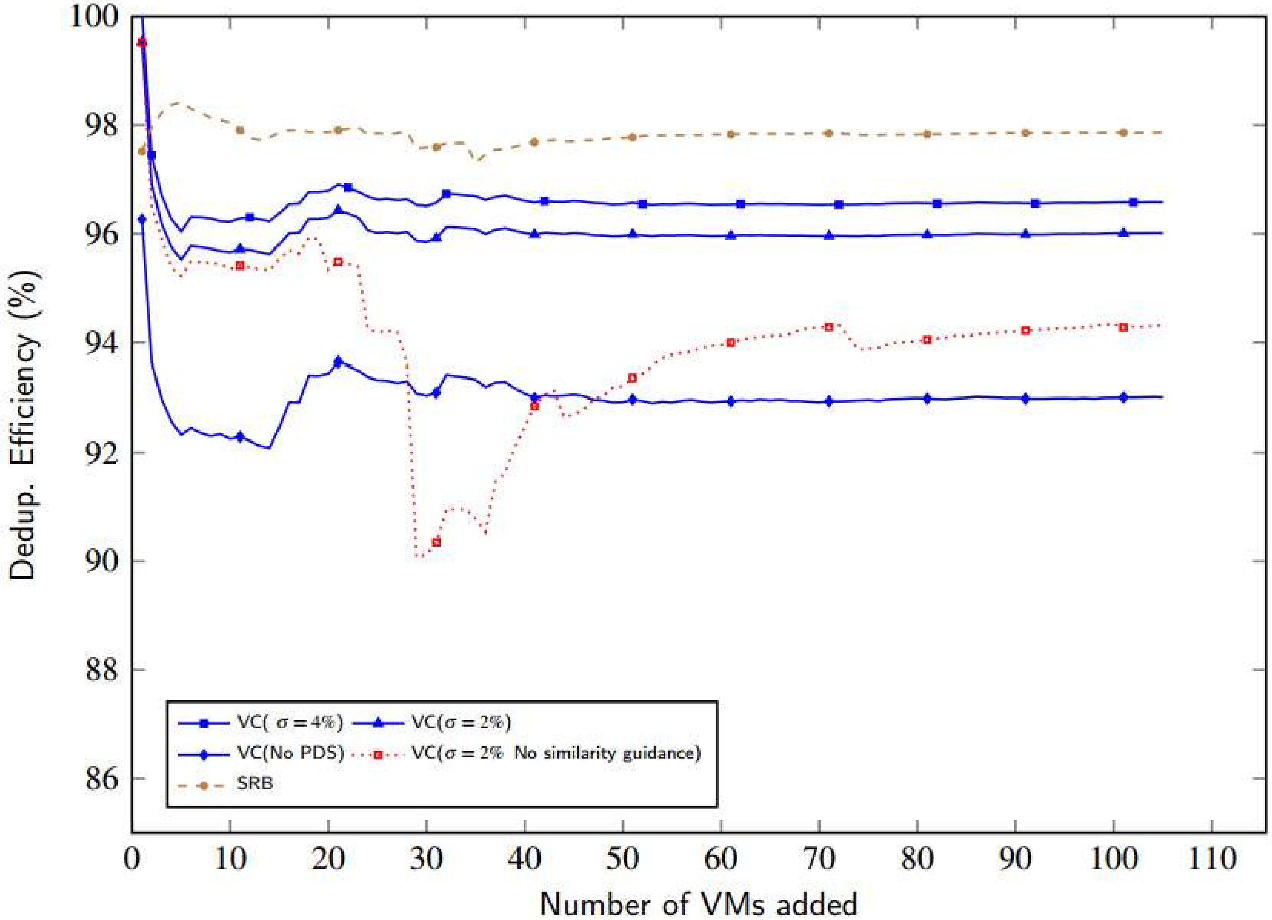
Impact for Different OS Releases



UCSB

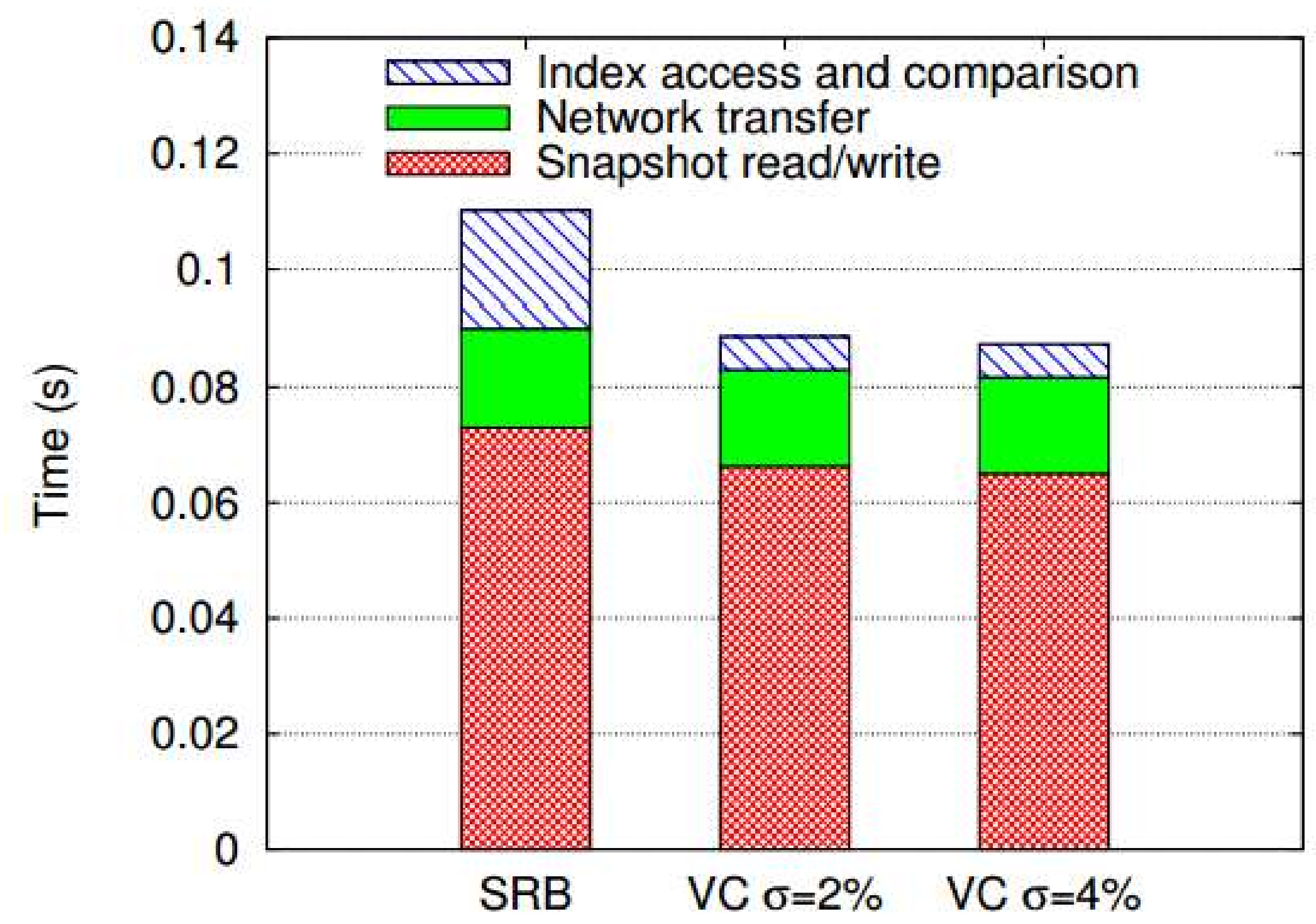
Dedup Efficiency (vs. SRB)

- **Compare to similarity-based stateless routing**



Process Time (vs. SRB)

- **Our method is faster due to less disk I/O**



Throughput

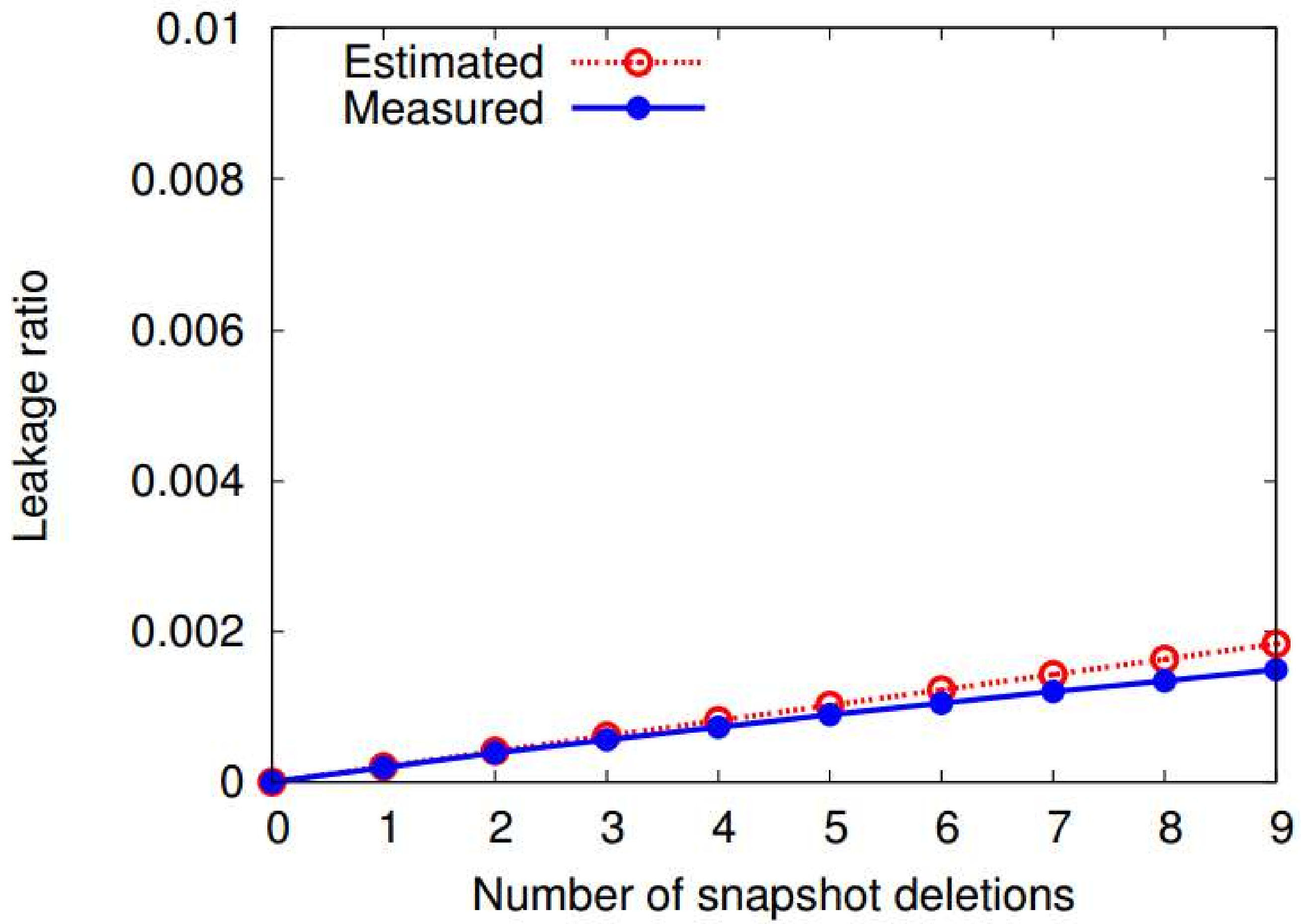
- **Backup throughput with multiple tasks running concurrently on every node (I/O not throttled)**

Concurrent backup tasks per machine	Throughput without I/O throttling (MB/s)		
	Backup	Snapshot Store (write)	QFS (write)
1	1369.6	148.0	35.3
2	2408.5	260.2	61.7
4	4101.8	443.3	103.1
6	5456.5	589.7	143.8

UCSB

Leakage of Approximation Deletion

- **Write down 10 snapshots**
- **Delete from the last one until only one left**



Summary of Contributions

- **Contributions to the field**
  - Low-cost backup storage solutions collocated with other cloud services.
  - VM-centric design with fault isolation
  - Data management with approximate deletion

UCSB

Thank You!

- **Questions?**

UCSB