VM-Centric Snapshot Deduplication for Cloud Data Backup

Wei Zhang
Daniel Agun, Tao Yang, Rich Wolski
Hong Tang

Pure Storage Inc.
University of California, Santa Barbara
Alibaba Inc.
Overview

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

Guest OS
VM  VM
VM  VM
VM  VM
VM  VM

Virtual Device Driver
Snapshot Dedup
Storage Access
Runtime VM Disk
Snapshot Store

Distributed Memory Cache
Hadoop-like Cloud Services
Distributed File System

Host OS
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 based 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
Metadata: Snapshot Representation

Snapshot recipes

Segment fingerprint
- Offset
- Size
- Data pointer

Segment recipes

Block fingerprint:
- Offset
- Size
- Data pointer

Data blocks
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
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

![Graphs showing data blocks from OS disks and data disks](image)
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
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
Processing Flow of Multi-level Deduplication

(a) 1. Segment level check-up
(b) 2. Block level check-up
(c) 3. CDS check-up
(d) Write Down

Inner-VM Duplication
Cross-VM Duplication

Snapshot Block Store
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
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
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
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
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?

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

- Daily VM change rate: \( \Delta u/u \sim 2\% - 4\% \)
Example

- **Sampled weekly VM disk change rate**

<table>
<thead>
<tr>
<th>Cluster name</th>
<th>Avg. system disk change rate</th>
<th>Avg. data disk change rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>AY41A (4224 VMs)</td>
<td>17.29%</td>
<td>15.05%</td>
</tr>
<tr>
<td>AY41C (2083 VMs)</td>
<td>16.64%</td>
<td>16.61%</td>
</tr>
<tr>
<td>AY41D (2966 VMs)</td>
<td>16.42%</td>
<td>12.86%</td>
</tr>
<tr>
<td>AY41E (5603 VMs)</td>
<td>17.83%</td>
<td>21.85%</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>Approximate deletion with repair</th>
<th>Perfect hashing (Fabiano et. al FAST ‘13)</th>
<th>Grouped mark-and-sweep (Guo et.al ATC’11)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>~10 MB</td>
<td>~GB</td>
<td>~100 MB</td>
</tr>
<tr>
<td>Time</td>
<td>~1 minute</td>
<td>~hours</td>
<td>~10 minutes</td>
</tr>
<tr>
<td>Scan metadata?</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Tracking metadata to data dependency?</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Leakage?</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>
• 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
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:
  ▪ User data
    – From 1323 VM users
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
Cumulative coverage of popular data

Coverage is the summation of covered data block size*frequency

\[
\sum_{i=1}^{i} S_i \times F_i
\]

\[
\text{Total data size}
\]
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

[Bar chart showing impact percentages for different OS releases, with categories such as new data, saved by level 2, and saved by level 1.]
Dedup Efficiency (vs. SRB)

- Compare to similarity-based stateless routing
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)

<table>
<thead>
<tr>
<th>Concurrent backup tasks per machine</th>
<th>Throughput without I/O throttling (MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Backup</td>
</tr>
<tr>
<td>1</td>
<td>1369.6</td>
</tr>
<tr>
<td>2</td>
<td>2408.5</td>
</tr>
<tr>
<td>4</td>
<td>4101.8</td>
</tr>
<tr>
<td>6</td>
<td>5456.5</td>
</tr>
</tbody>
</table>
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
Thank You!

• Questions?