Low-Profile Source-side Deduplication for Virtual Machine Backup

Daniel Agun, Tao Yang
University of California at Santa Barbara

Wei Zhang
Pure Storage
Cloud Platform and VM Snapshot Backup

- Public and private IaaS cloud systems have become industry standard
- Frequent virtual machine snapshot backups improves system reliability
- Backup traffic of VM snapshots with limited source-side deduplication is huge.
  - 100,000 VMs with 76% dirty bit detection still requires ~1 petabyte of networking with 40GB per VM snapshot
Objective: Aggressive Source-Side Deduplication With low-profile computing

- Backup data daily for tens of thousand VMs within a few hours each day.
- Minimize network traffic via aggressive source-side deduplication
  - State-of-art deduplication algorithms are memory/compute-intensive
- Resource friendly – small memory footprint and CPU usage, minimum impact to primary services
Strategies for Scalable/Low-cost Aggressive Source-Side Deduplication

- Focus on popular data chunks shared among snapshots
  - Zipf distribution. Top 2-4% of most popular items (plus inner-VM dedup) accomplishes ~98% deduplication efficiency.
- Cluster-based deduplication
  - Distribute VM chunk signatures to cluster machines
- Minimize job completion time instead of individual chunk backup time.
- Approximated snapshot deletion
Low-cost source-side cluster-based deduplication

- Given a set of VMs to be backed up, find if their block signatures are duplicates of the existing snapshot blocks.

- Challenge in control buffer size during data shuffling
  - Complicated by uneven VM size distribution.

Example datasets from Alibaba.

**Left**: 4200 VMs with max/average VM size = 20.

**Right**: 8000 VMs with max/avg = 45
Multi-round Collaborative Deduplication

- Major stages of each duplicate detection round
  - Stage 1: Collect fingerprints in parallel
  - Stage 2: Detect duplicates in parallel
  - Stage 3: Perform actual VM backup in parallel

- k rounds
- k too small – more buffering needed
- k too large – more dedup overhead

Choose k so that buffer memory $\leq 100$MB
How many rounds of backup batches?

• Estimate # of rounds $k$ based on memory usage per node

$$\frac{D \times V}{r \times p} \left[ \frac{1}{k \times q} + \frac{b \mu}{q} + \frac{1}{k} \right] \leq 100\text{MB}$$

• $p$ is the number of physical machines.
• $V$ is number of VMs hosted per machine
• $q$ is the number of fingerprint partitions per machine
• $D$ is size of modified data per VM
• $\mu$ is percentage of unique chunks among dirty data accumulated
• $b$ is the average number of snapshot versions per VM.
• $r$ is the ratio of chunk size over index entry size

$p=100$, $V=25$, $D=8.8\text{GB}$, $\mu =22.8\%$, $b=10$, $r=136$, $q=400$  
$\rightarrow k=12$. 9% of VMs is handled per batch
Low-cost Design for Snapshot Deletion

- Snapshot deletion is as frequent as creation
- Identifying unused chunks with reference counting is costly
  - Grouped Mark-and-sweep [Guo et. al, ATC’11]: A block can be deleted if its reference count is zero
- Our approximate approach
  - Separate strategies for popular chunks (2-4%) and non-popular inner VM chunks.
  - Approximate deletion for VM-specific chunks with bloom filter
Approximate Deletion for VM-specific chunks

- Summary vector to detect the usage of a chunk within a VM.
  - Use bloom filter to summarize snapshots of VM
    - Summary vectors of live snapshots represent the chunks in use
  - Checking the existence of a chunk reference is fast
    - Tolerate small percentage of storage leak to allow fast deletion with approximation

How often to repair leakage?
Leakage Analysis: How Often to Repair?

- Periodically repair with mark-and-sweep to remove false negatives (those with 0 reference, but not removed).
- $u$: the initial size of a snapshot.
- $\Delta u$: average VM change between consecutive snapshots.
- Total chunks stored after $h$ snapshots per VM:
  \[ U = u + (h - 1) \Delta u \]
- Total leakage after $R$ rounds: $L = R \varepsilon \Delta u$
  $\varepsilon$ is the misjudgement rate of bloom-filter summary vector.
- How often to repair?
  \[ \frac{L}{U} = \frac{R \Delta u \varepsilon}{u + (h - 1) \Delta u} > \tau \iff R > \frac{\tau}{\varepsilon} \times \frac{u + (h - 1) \Delta u}{\Delta u} \]
  With daily backup, $\Delta u/u = 2.5\%$, $h = 10$, $t = 0.1$, $\Rightarrow R = 19.6$ days.
Evaluation

- Prototype implementation in C. Evaluated on a Linux cluster of 8-core 3.1 GHz, AMD FX-8120. 16GB memory
- Test data from Alibaba Aliyuan cloud
  - 41TB. 10 snapshots per VM for 2500 VMs
  - Segment size: 2MB. Avg. chunk size: 4KB. SHA-1 fingerprint hash.
- Evaluation objectives
  - Compare resource usage of three source-side deduplication methods: 1) dirty bit. 2) Synchronous method. 2) Collaborative multi-round with \( k=12 \).
  - Impact of multi-round scheduling on backup job span
  - Compare exact deletion with approximate deletion on resource usage, time, and space leakage.
Data Characteristics

- Each VM uses 40GB storage space on average
- OS and user data disks: each takes ~50% of space
- Zipf-like distribution of VM OS/user data:
  - frequency of any chunk is inversely proportional to its rank in the frequency table

![Graphs](a) Data blocks from OS disks
![Graphs](b) Data blocks from data disks
Resource Comparison

- Resource usage comparison per snapshot.
- Local disk IO and memory costs are per machine.
- Storage and network cost are for 100 physical machines after deduplication.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Mem (MB)</th>
<th>Local IO (GB)</th>
<th>Storage (GB)</th>
<th>Network (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dirty Bit</td>
<td>&lt;10</td>
<td>220</td>
<td>22000</td>
<td>22000</td>
</tr>
<tr>
<td>Synchronous</td>
<td>40</td>
<td>453</td>
<td>4840</td>
<td>5500</td>
</tr>
<tr>
<td>Collaborative</td>
<td>90</td>
<td>491</td>
<td>4840</td>
<td>5500</td>
</tr>
</tbody>
</table>

Aggressive source-side deduplication incurs 4.55x less space and 4x less network traffic
Job time comparison

- Job span in hours (total time for backup of all VM snapshots)
- Average per-VM backup time
- Even VM size distribution vs skewed distribution with max/average size=20.

<table>
<thead>
<tr>
<th>Hours</th>
<th>Job span</th>
<th>Backup time per VM (even)</th>
<th>Backup time per VM (skew)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dirty Bit</td>
<td>1.25</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>Synchronous</td>
<td>50.40</td>
<td>2.75</td>
<td>50.40</td>
</tr>
<tr>
<td>Collaborative</td>
<td>2.36</td>
<td>0.23</td>
<td>0.23</td>
</tr>
</tbody>
</table>

Multiround collaborative processing with k=12 is 21x faster than synchronous method for job span. 1.88x slower than dirty bit method but still finishes in 2.36 hours.
Effectiveness of Approximate Deletion

- Processing time and per-machine memory usage of four deletion methods.
- # of machines: \( p = 50 \) and \( 100 \) while # of VMs per machine=25

<table>
<thead>
<tr>
<th></th>
<th>Time ( p=50 ) (hours)</th>
<th>Time ( p=100 ) (hours)</th>
<th>Memory (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mark&amp;sweep</td>
<td>35.9</td>
<td>84.3</td>
<td>1.2–3</td>
</tr>
<tr>
<td>Grouped mark&amp;sweep</td>
<td>18.6</td>
<td>43.6</td>
<td>1.2–3</td>
</tr>
<tr>
<td>Local w/o sum.</td>
<td>0.7</td>
<td>0.82</td>
<td>0.05 – 1.96</td>
</tr>
<tr>
<td>Approx. local</td>
<td>0.012</td>
<td>0.014</td>
<td>0.015</td>
</tr>
<tr>
<td>Leak repair</td>
<td>0.7</td>
<td>0.82</td>
<td>0.05 – 1.96</td>
</tr>
</tbody>
</table>

- Approximate deletion is 3114x faster than the grouped mark&sweep method.
- Leakage repair is 53x faster with 35% to 96% less memory usage
Contributions & Conclusions

- Scalable low-profile multi-round source-side deduplication for frequent VM snapshot backup.

- For the tested dataset,
  - Network cost: 4x and storage cost is reduced by 4.55x compared to a dirty-bit based method.
  - Multi-round deduplication is an order of magnitude faster than a synchronous scheme in dealing a skewed load.
  - Approximate snapshot deletion only requires 15MB per machine
    - 3114x faster than the grouped mark&sweep method.
    - Leakage repair is 53x faster with 35% to 96% less memory usage.
Thank You!
Questions?