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

DIRTY BLOCK TRACKER

SAN, NAS

DATA STORE

DIRTY BLOCK LIST

ENTIRE VOLUME
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

INCOMING FINGERPRINTS

<table>
<thead>
<tr>
<th>Fingerprint Hash</th>
<th>Location on Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>#F1</td>
<td>L1</td>
</tr>
<tr>
<td>#F7</td>
<td>L3</td>
</tr>
<tr>
<td>#F18</td>
<td>L1</td>
</tr>
<tr>
<td>#F89</td>
<td>L9</td>
</tr>
</tbody>
</table>

**Query**

Fetch container from disk
WHY SAMPLING WORKS?

Day N delta list
- 100-200

Day N+1 delta list
- 1000-1030
- 150-180

Day N+2 delta list
- 2000-2030
- 150-180

STABLE

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

Matched duplicates → STORED FINGERPRINTS

Non-matched duplicates → NEW Stored Fingerprint
CONTENT PROXIMITY (CP)
TP VS CP

The approach in which \textit{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

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

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 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.
<table>
<thead>
<tr>
<th>Snapshot 1</th>
<th>Snapshot 2</th>
<th>Snapshot 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 10, -1</td>
<td>1, 14, 10</td>
<td>1, 18, 14</td>
</tr>
<tr>
<td>2, 11, -1</td>
<td>2, 15, 11</td>
<td>7, 19, -1</td>
</tr>
<tr>
<td>3, 12, -1</td>
<td>5, 16, -1</td>
<td>5, 20, 16</td>
</tr>
<tr>
<td>4, 13, -1</td>
<td>6, 17, -1</td>
<td>8, 21, -1</td>
</tr>
</tbody>
</table>
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

BOSC Logging

Fast Logging Disk

GC Disk 1

GC Disk 2

GC Thread 1
Update Bucket N

GC Thread 2
Update Bucket K

Metadata P-Array
## GC’s INFLUENCE

<table>
<thead>
<tr>
<th>Commit Threads</th>
<th>Dedupe +Vanilla GC</th>
<th>Dedupe +BOSC GC</th>
<th>Dedupe &amp; NO GC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5879</td>
<td>54047</td>
<td>287204</td>
</tr>
<tr>
<td>2</td>
<td>6003</td>
<td>268218</td>
<td>287204</td>
</tr>
<tr>
<td>4</td>
<td>9858</td>
<td>277670</td>
<td>287204</td>
</tr>
<tr>
<td>10</td>
<td>8121</td>
<td>269272</td>
<td>287204</td>
</tr>
</tbody>
</table>

*very much comparable*

*severe bottleneck*
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
SUSTAINED HIGH PERFORMANCE

Deduplication Throughput (Kilo Fingerprints/sec)

Deduplication Ratio (Percentage)

Input Fingerprint Count (Unit: Million fingerprints)

Deduplication Throughput

Deduplication Ratio

31
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.