Delta Compressed and Deduplicated Storage Using Stream-Informed Locality

Philip Shilane, Grant Wallace, Mark Huang, & Windsor Hsu

*Backup Recovery Systems Division*
*EMC Corporation*
Motivation and Approach

• Improve storage compression
  – Decrease price per GB
  – Decrease data center space
  – Decrease power
  – Decrease management

• Combine deduplication and delta compression
  – Remove identical data regions
  – Compress with similar data regions
Previous Work on Similarity Indexing

• Version information [Burns’97, MacDonald’00]
• Similarity index in memory [Aronovich’09, Kulkarni’04]
• Similarity index on-disk [You’11]
• Stream-informed delta locality for WAN replication [Shilane’12]
  – Low WAN throughput
  – Did not store delta compressed data
Contributions

1. First deduplicated and delta compressed storage implementation using stream-informed locality
2. Quantify throughput and suggest improvement areas
3. Explore new complexities related to data integrity and cleaning
4. Report combination of deduplication and delta compression across chunk sizes
Stream-informed Deduplication

backup.tar
Stream-informed Deduplication

backup.tar

| Chunk 1 | Chunk 2 | Chunk 3 | Chunk 4 | Chunk 5 | Chunk 6 |

Content Defined Chunks
Stream-informed Deduplication

backup.tar

| Fp 1 | Fp 2 | Fp 3 | Fp 4 | Fp 5 | Fp 6 |

Secure Fingerprint of Chunks
Stream-informed Deduplication

Store fingerprints in a metadata section and chunks to a data section of a container. Also create an index from fingerprint to container.
Stream-informed Deduplication

Fp 1
Fp 2
Fp 3
Fp 4
Fp 5
Fp 6

backup.tar

1 week later

backup.tar

Fp Index

Chunks stored together on disk

Container C1

Fp 1 -> C1
...
Fp 6 -> C1

Meta Data

Fp 1 ... 6

Data

Chunks 1 ... 6
Stream-informed Deduplication

backup.tar

1 week later

backup.tar

content Defined Chunks

Chunks stored together on disk

Fp Index

Container C1

Meta Data

Data

Fp 1 -> C1

...     Fp 1 ... 6

Fp 6 -> C1

Chunks 1 ... 6

© Copyright 2012 EMC Corporation. All rights reserved.
Stream-informed Deduplication

Fp 1 -> C1
...
Fp 6 -> C1

Fp Index

Container C1

Meta Data
Fp 1 ... 6

Data
Chunks 1 ... 6

Secure Fingerprint of Chunks

Chunks stored together on disk

backup.tar
Fp 1 Fp 2 Fp 3 Fp 4 Fp 5 Fp 6

1 week later

backup.tar
Fp 1 Fp 2 Fp 3 Fp 4' Fp 5 Fp 6
Stream-informed Deduplication

backup.tar

Fp 1 | Fp 2 | Fp 3 | Fp 4 | Fp 5 | Fp 6

1 week later

backup.tar

Fp 1 | Fp 2 | Fp 3 | Fp 4' | Fp 5 | Fp 6

Lookup fingerprint 1 in the index

Fp Index

Chunks stored together on disk

Container C1

Fp 1 -> C1

Meta Data

Fp 1 ... 6

Data

Chunks 1 ... 6

© Copyright 2012 EMC Corporation. All rights reserved.
Stream-informed Deduplication

Load all fingerprints from container 1 into an in-memory cache.
Stream-informed Deduplication

**Fingerprint Cache**

- Chunks stored together on disk
- Container C1
  - Meta Data
    - Fp 1 ... 6
  - Data
    - Chunks 1 ... 6

**Backup.tar**

1 week later

FP cache hit on 1,2,3,5, & 6

**Fp Index**

- Fp 1 -> C1
- ... (omitted)
- Fp 6 -> C1

**Fingerprint Cache**

- Fp 1
- Fp 2
- Fp 3
- Fp 4'
- Fp 5
- Fp 6
Stream-informed Deduplication and Delta Compression

backup.tar
Stream-informed Deduplication and Delta Compression

backup.tar

| Chunk 1 | Chunk 2 | Chunk 3 | Chunk 4 | Chunk 5 | Chunk 6 |

Content Defined Chunks
Stream-informed Deduplication and Delta Compression

Secure Fingerprint and Sketch of Chunks

Calculate fingerprints used for deduplication and sketches used for similarity detection
Stream-informed Deduplication and Delta Compression

Store fingerprints and sketches in a metadata section and chunks to a data section of a container. Also create an index from fingerprint to container.
Stream-informed Deduplication and Delta Compression

backup.tar

Fp 1 → C1
...
Fp 6 → C1

Container C1

Meta  Fp 1 ... 6
Data   Sk 1 ... 6
Data   Chunks 1 ... 6

Chunks stored together on disk

1 week later

backup.tar
Stream-informed Deduplication and Delta Compression

1 week later

Chunks stored together on disk

Container C1

<table>
<thead>
<tr>
<th>Meta</th>
<th>Fp 1 ... 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Sk 1 ... 6</td>
</tr>
<tr>
<td>Data</td>
<td>Chunks 1 ... 6</td>
</tr>
</tbody>
</table>

Fp Index

<table>
<thead>
<tr>
<th>Fp 1 -&gt; C1</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
</tr>
<tr>
<td>Fp 6 -&gt; C1</td>
</tr>
</tbody>
</table>

Content Defined Chunks

backup.tar

Fp 1 Sk 1  Fp 2 Sk 2  Fp 3 Sk 3  Fp 4 Sk 4  Fp 5 Sk 5  Fp 6 Sk 6

Chunk 1  Chunk 2  Chunk 3  Chunk 4'  Chunk 5  Chunk 6
Stream-informed Deduplication and Delta Compression

Secure Fingerprint and Sketch of Chunks

Fp Index

Container C1

Chunks stored together on disk

© Copyright 2012 EMC Corporation. All rights reserved.
Stream-informed Deduplication and Delta Compression

<table>
<thead>
<tr>
<th>Fp 1</th>
<th>Fp 2</th>
<th>Fp 3</th>
<th>Fp 4</th>
<th>Fp 5</th>
<th>Fp 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sk 1</td>
<td>Sk 2</td>
<td>Sk 3</td>
<td>Sk 4</td>
<td>Sk 5</td>
<td>Sk 6</td>
</tr>
</tbody>
</table>

1 week later

Chunks stored together on disk

Lookup fingerprint 1 in the index

Container C1

<table>
<thead>
<tr>
<th>Meta</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fp 1 ... 6</td>
<td>Sk 1 ... 6</td>
</tr>
</tbody>
</table>

Data

Chunks 1 ... 6

Fp Index

Fp 1 -> C1

...
Stream-informed Deduplication and Delta Compression

Load all fingerprints and sketches from container 1 into an in-memory cache
Stream-informed Deduplication and Delta Compression

backup.tar

1 week later

backup.tar

FP cache hit on 1,2,3,5, & 6

Fingerprint and Sketch Cache

Container C1

Fp Index

Chunks stored together on disk

Meta
Fp 1 ... 6

Data
Sk 1 ... 6

Data
Chunks 1 ... 6

Fp 1 -> C1

... Fp 6 -> C1
Stream-informed Deduplication and Delta Compression

**Backup.ta**

- *Fp 1 Sk 1*
- *Fp 2 Sk 2*
- *Fp 3 Sk 3*
- *Fp 4 Sk 4*
- *Fp 5 Sk 5*
- *Fp 6 Sk 6*

1 week later

**backup.tar**

- *Fp 1 Sk 1*
- *Fp 2 Sk 2*
- *Fp 3 Sk 3*
- *Fp 4 Sk 4*
- *Fp 5 Sk 5*
- *Fp 6 Sk 6*

**Fingerprint and Sketch Cache**

- Sketch cache hit on 4
- Delta encode

**Container C1**

- **Meta**
  - Fp 1 ... 6
- **Data**
  - Sk 1 ... 6
  - Chunks 1 ... 6

**Fp Index**

- Fp 1 -> C1
- ...
- Fp 6 -> C1

**Chunks stored together on disk**

**Fingerprints and Sketch Cache**

- Fp 1
- ...
- Fp 6
- Sk 1
- ...
- Sk 6
Deduplication and Delta Compression

Chunk

Sketches based on Broder‘97
Deduplication and Delta Compression

```
chunk
Maximal Value 1  Maximal Value 2  Maximal Value 3  Maximal Value 4
```

\[
super\_feature = \text{Rabin\_fp}(\text{feature}_1...\text{feature}_4)
\]

sketch is one or more super_features

Sketches based on Broder‘97

© Copyright 2012 EMC Corporation. All rights reserved.
Deduplication and Delta Compression

\[
\text{super\_feature} = \text{Rabin\_fp(feature}_1\ldots\text{feature}_4)\\
\text{sketch is one or more super\_features}
\]

Sketches based on Broder‘97
Deduplication and Delta Compression

Chunk

Chunk

(duplicate of earlier chunk)

Fingerprint is a match, so do not store

\[ \text{super} \_\text{feature} = \text{Rabin} \_\text{fp}(\text{feature}_1 \ldots \text{feature}_4) \]

sketch is one or more super\_features

Sketches based on Broder‘97
Deduplication and Delta Compression

\[ \text{super\_feature} = \text{Rabin\_fp}(\text{feature}_1 \ldots \text{feature}_4) \]

sketch is one or more super\_features

Sketches based on Broder'97
Deduplication and Delta Compression

Chunk

Chunk

(similar to earlier chunk)

Regions of difference

Fingerprint is not a match, so calculate a sketch

\[
\text{super\_feature} = \text{Rabin\_fp}(\text{feature}_1 \ldots \text{feature}_4)
\]

sketch is one or more super\_features

Sketches based on Broder’97
Deduplication and Delta Compression

\[
\text{super\_feature} = \text{Rabin\_fp}(\text{feature}_1 \ldots \text{feature}_4)
\]

sketch is one or more super\_features

Sketches based on Broder’97
Deduplication and Delta Compression

Chunk

(similar to earlier chunk)

Regions of difference

Calculate a delta and store the changed bytes and a reference to the earlier chunk

Store fp and differences

Sketches based on Broder’97
# Backup Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Type</th>
<th>Backup Policy</th>
<th>TB</th>
<th>Months</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workstations</td>
<td>16 desktops</td>
<td>Weekly full Daily incremental</td>
<td>2.3</td>
<td>4</td>
</tr>
<tr>
<td>Email</td>
<td>MS Exchange server</td>
<td>Daily full</td>
<td>2.5</td>
<td>5</td>
</tr>
<tr>
<td>Source Code</td>
<td>Version control repository</td>
<td>Weekly full Daily incremental</td>
<td>4.5</td>
<td>6</td>
</tr>
<tr>
<td>System Logs</td>
<td>Server’s /var directory</td>
<td>Weekly full Daily incremental</td>
<td>5.3</td>
<td>4</td>
</tr>
</tbody>
</table>
Compression Results

Delta adds 1.4 – 3.5X compression improvement over deduplication and LZ

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Deduplication</th>
<th>Delta</th>
<th>LZ</th>
<th>Total Compression</th>
<th>Delta Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workstations</td>
<td>5.0</td>
<td>4.2</td>
<td>1.6</td>
<td>33.6</td>
<td>3.5</td>
</tr>
<tr>
<td>Email</td>
<td>4.9</td>
<td>2.6</td>
<td>2.1</td>
<td>26.8</td>
<td>2.1</td>
</tr>
<tr>
<td>Source Code</td>
<td>16.7</td>
<td>3.6</td>
<td>2.5</td>
<td>150.3</td>
<td>1.4</td>
</tr>
<tr>
<td>System Logs</td>
<td>25.2</td>
<td>3.3</td>
<td>1.8</td>
<td>149.7</td>
<td>1.5</td>
</tr>
</tbody>
</table>
Throughput

- Delta compression requires extra computation and I/O
- Compare to deduplicated storage as baseline
- Throughput: 74% on first full backup
- Throughput: 53% on later full backups
## Throughput Stages

Single-stream timing for each stage

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Sketch MB/s</th>
<th>Lookup MB/s</th>
<th>Encode Mb/s</th>
<th>HDD MB/s</th>
<th>SSD MB/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workstations</td>
<td>47</td>
<td>1,528</td>
<td>94</td>
<td>5</td>
<td>400</td>
</tr>
<tr>
<td>Email</td>
<td>49</td>
<td>1,441</td>
<td>69</td>
<td>1</td>
<td>80</td>
</tr>
<tr>
<td>Source Code</td>
<td>30</td>
<td>30</td>
<td>31</td>
<td>2</td>
<td>160</td>
</tr>
<tr>
<td>System Logs</td>
<td>30</td>
<td>70</td>
<td>50</td>
<td>2</td>
<td>160</td>
</tr>
</tbody>
</table>

I/O to a hard drive is the bottleneck, switching to SSD may help

Aggregate throughput is higher due to: deduplication, 2/3 are delta encoded, multi-threading and asynchronous reads across multiple disks
Indirection Complexities

- Writing duplicates causes unintended read paths
  - Unpredictable read back times

Read back cat: There are multiple options, some that involve delta references
Indirection Complexities

- Writing duplicates causes unintended read paths
  - Unpredictable read back times
- Multi-level delta increases compression and complexity
  - We implemented 1-level delta
Indirection Complexities

- End-to-end validity checks are slow because of remote references
  - Reconstructing a delta chunk requires reading the base

Verify catcher
Indirection Complexities

- End-to-end validity checks are slow because of remote references
  - Reconstructing a delta chunk requires reading the base
- Incorrect garbage collection can cause loops and data loss

Verify catcher: Loop due to incorrect GC indicates data loss
Garbage Collection

- Cleaning deleted chunks in a log structured file system
  - Reference counts
  - Mark-and-sweep
- Copying live chunks forward changes data locality, which impacts delta compression
Garbage Collection Impact on Delta Compression

GC changes data locality but has only a small impact on delta compression.
Compression vs. Chunk Size

Consistent results on all datasets
Compression vs. Chunk Size

Delta adds significant compression beyond deduplication and LZ.
Compression vs. Chunk Size

Delta helps maintain high total compression as chunk size varies.
Summary and Future Work

- Deduplication and delta compression prototype
  - Stream-informed locality replaces sketch indexes and improves write path throughput
  - Adds 1.4X – 3.5X compression

- Studied throughput
  - Throughput 50% of underlying deduplication system
  - Areas for improvement: SSD

- Garbage collection and data integrity
  - Remote reference complexity
  - Affects speed and validity

- Delta helps maintain overall compression across a broad range of chunk sizes
Questions?