Making the Archive Real

Hakim Weatherspoon
University of California, Berkeley

ROC/OceanStore Retreat, Lake Tahoe. Tuesday, June 11, 2002
Archival Storage

• Every version of every object is archived.
  – At least in principle.
• How long does it take to...
  – Write?
  – Read?
• Can the Archive be run *inline*?
  – Or, is it best to batch?
Outline

• Archival Process context.
• Archival Modes.
• Archival Performance.
• Future Directions
Path of an Update
Archival Dissemination Built into Update

- **Erasure codes**
  - redundancy without overhead of strict replication
  - produce $n$ fragments, where any $m$ is sufficient to reconstruct data.
  - $m < n$. Rate $r = m/n$. Storage overhead is $1/r$. 
Durability

- Fraction of Blocks Lost Per Year (FBLPY)*
  - $r = \frac{1}{4}$, erasure-encoded block. (e.g. $m = 16$, $n = 64$)
  - Increasing number of fragments, increases durability of block
    - Same storage cost and repair time.
  - $n = 4$ fragment case is equivalent to replication on four servers.

**Naming and Verification Algorithm**

- Use cryptographically secure hash algorithm to detect corrupted fragments.

- **Verification Tree:**
  - $n$ is the number of fragments.
  - store $\log(n) + 1$ hashes with each fragment.
  - Total of $n(\log(n) + 1)$ hashes.

- Top hash is a **block GUID** *(B-GUID).*
  - Fragments and blocks are self-verifying

---

**Fragment 1:**
- $H2$  $H34$  $Hd$  $F1$ - fragment data

**Fragment 2:**
- $H1$  $H34$  $Hd$  $F2$ - fragment data

**Fragment 3:**
- $H4$  $H12$  $Hd$  $F3$ - fragment data

**Fragment 4:**
- $H3$  $H12$  $Hd$  $F4$ - fragment data

Data:
- $H14$  data
Complex Objects I

Verification Tree

Encoded Fragments:
Unit of Archival Storage

Unit of Coding

GUID of d

data
Complex Objects II

- VGUID
- Data B-Tree
- Indirect Blocks
- Blocks Data
- Unit of Coding
- Verification Tree
- GUID of \( d_1 \)
- Encoded Fragments: Unit of Archival Storage
- Unit of Coding
- \( d_1 \), \( d_2 \), \( d_3 \), \( d_4 \), \( d_5 \), \( d_6 \), \( d_7 \), \( d_8 \), \( d_9 \)
Complex Objects III

AGUID = hash\{name+keys\}

VGUID\_i

Data B -Tree

Indirect Blocks

Data Blocks

\[ d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9 \]

\[ d_1', d_2', d_3', d_4', d_5', d_6', d_7', d_8', d_9' \]

VGUID\_i+1

backpointer

copy on write

copy on write

ROC/OceanStore ©2002 Hakim Weatherspoon/UC Berkeley OceanStore Archive:11
Mutable Data

• Need mutable data for real system.
  - Entity in network.
  - A-GUID to V-GUID mapping.
  - Byzantine Commitment for Integrity
  - Verifies client privileges.
  - Creates a serial order.
  - Atomically applies update.

• Versioning system
  - Each version is inherently read-only.
Archiver Server Architecture
- Requests to archive objects recv’d thru network layers.
- Consistency mechanisms decides to archive obj.
# Archival Modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>synch?</th>
<th>latency</th>
<th>durability</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Archive</td>
<td>synch</td>
<td>minimum</td>
<td>none</td>
</tr>
<tr>
<td>Inlined Archive</td>
<td>synch</td>
<td>maximum</td>
<td>maximum</td>
</tr>
<tr>
<td>Delayed Archive</td>
<td>asynch</td>
<td>minimum</td>
<td>medium</td>
</tr>
</tbody>
</table>
Performance

Performance of the Archival Layer

- Performance of an OceanStore server in archiving objects.
- Analyze operations of archiving data (this includes signing updates in a BFT protocol).
- $m = 16$, $n = 32$

• Experiment Environment
- OceanStore servers were analyzed on a 42-node cluster.
- Each machine in the cluster is a
  - IBM xSeries 330 1U rackmount PC with
  - Two 1.0 GHz Pentium III CPUs
  - 1.5 GB ECC PC133 SDRAM
  - Two 36 GB IBM UltraStar 36LZX hard drives.
  - The machines use a single Intel PRO/1000 XF gigabit Ethernet adaptor to connect to a Packet Engines
  - Linux 2.4.17 SMP kernel.
Data Throughput

- No archive 5MB/s.
- Delayed 3MB/s.
- Inlined 2.5MB/s.
Performance: Throughput II

Operations Throughput
- No archive 55 opt/s. Small writes.
- Delayed 8 opt/s. Small writes.
- Inlined 50 opt/s. Small writes.
Performance: Latency I

- Latency
  - Archive only
    - Y-intercept 3ms, slope 0.3s/MB.
  - Inlined Archive = No archive + only archive.

![Update Latency vs. Update Size]

- $y = 0.6x + 29.6$
- $y = 0.3x + 3.0$
- $y = 1.2x + 36.4$
Performance: Latency II

Latency
- No Archive. Low variance.
- Delayed Archive. High variance.
Performance: Latency III

• Latency
  - Inlined Archive. Low variance.
  - Delayed Archive. High variance.
Read Latency

- Low median.
- High variance.
  - Queuing effect.
Future Directions

• Caching for performance
  - Automatic Replica Placement
• Automatic Repair
• Low Failure Correlation Dissemination
Caching

• **Automatic Replica Placement**
  - Replicas are *soft-state*.
  - Can be constructed and destroyed as necessary.

• **Prefetching**
  - Reconstruct replicas from fragments in advance of use
Efficient Repair

• Global.
  – Global Sweep and repair not efficient.
  – Want detection/notification of node removal in system.
  – Not as affective as distributed mechanisms.

• Distributed.
  – Exploit DOLR’s distributed information and locality properties.
  – Efficient detection and then reconstruction of fragments.
Low Failure Correlation Dissemination

- **Model Builder.**
  - Various sources.
  - Model failure correlation.

- **Set Creator.**
  - Queries random nodes.
  - **Dissemination Sets.**
    - Storage servers that fail with low correlation.

- **Disseminator.**
  - Sends fragments to members of set.
Conclusion

• Storage efficient, self-verifying mechanism.
  - Erasure codes are good.
• Performance is good.
  - Improvements forthcoming.
• Self-verifying data assist in
  - Secure read-only data
  - Secure caching infrastructures
  - Continuous adaptation and repair

For more information:
http://oceanstore.cs.berkeley.edu/