Cumulus: Filesystem Backup to the Cloud

Michael Vrable, Stefan Savage, and Geoffrey M. Voelker

Presented by Hakim Weatherspoon
Good ‘ole Days

Replace your tape drives with something truly scalable
Amazon S3 to the rescue

In Spring 2006, Amazon released a new storage API: **Put, Get, List, Delete**

Build whatever you want! Quickly
Backing up the new way (S3)

- **Smart**

- **Scales**
  - no longer our concern... Amazon's concern
  - all servers backup in parallel

- **Cheap**
  - old cost = $XXX per year
  - new cost = $YYY per year
  - where $YYY < $XXX
Thin vs Thick Cloud

• E.g. Amazons S3 vs EMC’s MozyPro

• Thin
  – Can change provider easier
  – Applications can work across providers

• Thick
  – Better performance
  – Locked into a provider
  – Provider can go out of business
Cumulus

- Simple storage backup utility for Thin Clouds
- Evaluates efficacy of cloud storage
- Working prototype
  - http://www.cs.ucsd.edu/~mvrable/cumulus/
Outline

- Motivation/Intro
- Related Work
- Design
- Evaluation
- Thoughts and Conclusions
## Related Work

<table>
<thead>
<tr>
<th></th>
<th>Multiple snapshots</th>
<th>Simple server</th>
<th>Incremental forever</th>
<th>Sub-file delta storage</th>
<th>Encryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>rsync</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>rsnapshot</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rdiff-backup</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Box Backup</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Jungle Disk</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>duplicity</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Brackup</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Cumulus</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Outline

• Motivation/Intro
• Related Work
• Design
  – API
  – Segments
  – Snapshots
  – Subfile incrementals
  – Cleaning
  – restoring
• Evaluation
• Thoughts and Conclusions
API

- **Same as S3**
  - Put, Get, List, Delete

- **Thin cloud** – does not rely on integrated services
  - Can easily change provider and network protocols
  - S3, FTP, SFTP

- **WORM Model**
  - Write-once, read-many
  - Requires writing new entirely file if changes occur
  - What are the cleaning overheads?
Segments

• Aggregation via Segment Goals
  – Avoid costs due to small files
    • S3 charges on per file bases
    • Many small files
  – Avoid costs in network protocols
    • Small files have higher latency and other overheads
  – Compression
    • inter-file similarities
  – Privacy
    • Hide file boundaries

• Negative consequences?
  – Need an entire segment to write
Snapshots

Snapshot Descriptors
- Date: 2008-01-01 12:00:00
- Root: A/0
- Segments: A B

- Date: 2008-01-02 12:00:00
- Root: C/0
- Segments: B C

Segment Store
- Segment A
  - Name: file1
  - Owner: root
  - Data: B/0
- Segment B
  - Name: file1
  - Owner: root
  - Data: B/1 B/2
- Segment C
  - Name: file1
  - Owner: root
  - Data: C/1
  - Name: file2
  - Owner: root
  - Data: B/1 B/2
Sub-File Incrementalss

- Only stored changed part of files
- New snapshots point to old objects when data unchanged
  - Byte ranges – portions of old objects to be reused
Segment Cleaning

- Similar to a log-structured file system (LFS)
- Clean based on utilization of segment, $\alpha$
  - $\alpha = 0$, no cleaning
  - $\alpha = 1$, clean with the slightest change
- Cumulus
  - attempts to find an equilibrium for $\alpha$
  - Uses a different process to clean
  - Marks a local database as “expired”
  - Then, next snapshot will not refer to expired segment
• Full Restore
  – Download all segments for a snapshot
• Partial Restore
  – Download snapshot descriptor, metadata, and only necessary segments

• What happens if client machine dies?
• How is latest snapshot descriptors identified?
• What about sharing between client machines?
Outline

• Motivation/Intro
• Related Work
• Design
• Evaluation
  – Performance Case Study
  – Monetary Case Study
• Thoughts and Conclusions
# Evaluation Traces

<table>
<thead>
<tr>
<th></th>
<th>Fileservers</th>
<th>Users</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration (days)</td>
<td>157</td>
<td>223</td>
</tr>
<tr>
<td>Entries</td>
<td>26673083</td>
<td>122007</td>
</tr>
<tr>
<td>Files</td>
<td>24344167</td>
<td>116426</td>
</tr>
</tbody>
</table>

**File Sizes**

<table>
<thead>
<tr>
<th></th>
<th>Fileservers</th>
<th>Users</th>
</tr>
</thead>
<tbody>
<tr>
<td>Median</td>
<td>0.996 KB</td>
<td>4.4 KB</td>
</tr>
<tr>
<td>Average</td>
<td>153 KB</td>
<td>21.4 KB</td>
</tr>
<tr>
<td>Maximum</td>
<td>54.1 GB</td>
<td>169 MB</td>
</tr>
<tr>
<td>Total</td>
<td>3.47 TB</td>
<td>2.37 GB</td>
</tr>
</tbody>
</table>

**Update Rates**

<table>
<thead>
<tr>
<th></th>
<th>Fileservers</th>
<th>Users</th>
</tr>
</thead>
<tbody>
<tr>
<td>New data/day</td>
<td>9.50 GB</td>
<td>10.3 MB</td>
</tr>
<tr>
<td>Changed data/day</td>
<td>805 MB</td>
<td>29.9 MB</td>
</tr>
<tr>
<td>Total data/day</td>
<td>10.3 GB</td>
<td>40.2 MB</td>
</tr>
</tbody>
</table>
Backup over time (user trace)
Backup w/out Segment Cleaning
(user trace)
Average Daily Storage (fileserver)
Average Daily Upload (fileserver)
Average Segments per Day (fileserver)
Storage overhead for 16MB Segment (fileserver)
Optimal Cleaning Threshold

![Graph showing estimated optimal cleaning thresholds for fileservers and users against storage/network cost ratio.]
## Overheads

<table>
<thead>
<tr>
<th></th>
<th>File A</th>
<th>File B</th>
</tr>
</thead>
<tbody>
<tr>
<td>File size</td>
<td>4.860 MB</td>
<td>5.890 MB</td>
</tr>
<tr>
<td>Compressed size</td>
<td>1.547 MB</td>
<td>2.396 MB</td>
</tr>
<tr>
<td>Cumulus size</td>
<td>5.190 MB</td>
<td>3.081 MB</td>
</tr>
<tr>
<td>Size overhead</td>
<td>235%</td>
<td>29%</td>
</tr>
<tr>
<td>rdiff delta</td>
<td>1.421 MB</td>
<td>122 KB</td>
</tr>
<tr>
<td>Cumulus delta</td>
<td>1.527 MB</td>
<td>181 KB</td>
</tr>
<tr>
<td>Delta overhead</td>
<td>7%</td>
<td>48%</td>
</tr>
</tbody>
</table>
Monetary Case Study

- Storage: $0.15 per GB per Month
- Upload: $0.10 per GB
- Segment: $0.01 per 1000 files uploaded

*We are charged this amount, so please be careful with your labs and projects!!!*
## Monetary Costs for Backup

<table>
<thead>
<tr>
<th></th>
<th>Amount</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fileserver</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initial upload</td>
<td>3563 GB</td>
<td>$356.30</td>
</tr>
<tr>
<td>Upload</td>
<td>303 GB/month</td>
<td>$30.30/month</td>
</tr>
<tr>
<td>Storage</td>
<td>3858 GB</td>
<td>$578.70/month</td>
</tr>
<tr>
<td><strong>User</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initial upload</td>
<td>1.82 GB</td>
<td>$0.27</td>
</tr>
<tr>
<td>Upload</td>
<td>1.11 GB/month</td>
<td>$0.11/month</td>
</tr>
<tr>
<td>Storage</td>
<td>2.68 GB</td>
<td>$0.40/month</td>
</tr>
</tbody>
</table>
Costs for Backup (fileserver)
## Monetary Cost Comparison (user trace)

<table>
<thead>
<tr>
<th>System</th>
<th>Storage</th>
<th>Upload</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jungle Disk</td>
<td>≈ 2 GB</td>
<td>1.26 GB</td>
<td>30000</td>
</tr>
<tr>
<td></td>
<td>$0.30</td>
<td>$0.126</td>
<td>$0.30</td>
</tr>
<tr>
<td>Brackup</td>
<td>1.340 GB</td>
<td>0.760 GB</td>
<td>9027</td>
</tr>
<tr>
<td>(default)</td>
<td>$0.201</td>
<td>$0.076</td>
<td>$0.090</td>
</tr>
<tr>
<td>Brackup</td>
<td>1.353 GB</td>
<td>0.713 GB</td>
<td>1403</td>
</tr>
<tr>
<td>(aggregated)</td>
<td>$0.203</td>
<td>$0.071</td>
<td>$0.014</td>
</tr>
<tr>
<td>Cumulus</td>
<td>1.264 GB</td>
<td>0.465 GB</td>
<td>419</td>
</tr>
<tr>
<td></td>
<td>$0.190</td>
<td>$0.047</td>
<td>$0.004</td>
</tr>
</tbody>
</table>
Outline

• Motivation/Intro
• Related Work
• Design
• Evaluation
• Thoughts and Conclusions
Discussion

• Thoughts?

• Did paper make case for Thin Clouds?

• Sharing between clients ignored?

• What every happened to P2P?!
Lab 0

Next Time

• Read NFS and write review:

• Do Lab 0

• Check website for updated schedule