CloudDrive: Smarter Block Level Cloud-Backed Storage

Ishani Ahuja, Suli Yang, Remzi H. Arpaci-Dusseau, Andrea C. Arpaci-Dusseau

Motivation
Cloud Storage at Block Level:
- Level below the disk in the storage hierarchy
- Higher latency and higher storage capacity
- Provides reliability and mobility
- Potential to further simplify file system[1]

Why is Block Level interesting and hard?
- Backwards compatible with existing file system
- Lack file system information and semantic inference at Block level is hard

What is Strawman Block Level Approach bad?

System Architecture

Disk is used as a cache. Cloud Blocks are stored in Disk Frames. The Mapping Information is persisted in the disk cache to allow usage of cached data across system reboot.

Evaluation

Microbenchmark

- Disk-like Performance for cached Data.
- Cloud-like Performance for uncached data.

- Disk-like performance always!!!

Cache Consistency Performance:

- Random Read 6.19ms/171.13ms/10.03ms
- Random Write 8.14ms/290.42ms/12.17ms
- Sequential Read 39.3MB/s/3.89MB/s
- Sequential Write 32.8MB/s/3.06MB/s

Fast Writes with Transactional Update

- Random Write 6.69ms/31.89ms/22.75ms
- Fast Writes with Journal Guided Checkpointing

- Random Write 6.69ms/21.34ms/8.87ms

Raw Disk

- Random Read 6.00ms/8.36ms
- Random Write 7.54ms/9.28ms
- Sequential Read 40MB/s/40MB/s
- Sequential Write 32.3MB/s/36.7MB/s

References and Related Work


Basics of Block Level Cloud Storage

1. File System View of Storage
2. Disk as a cache
3. On–disk mapping table
4. Storage in Cloud Bucket in 4MB Files
5. Memory File Cache
6. Disk (Infinite Storage)

File System View of Storage

System Components
- **Device Driver(Cloud Cache)** responsible for serving read/write request from page cache.
- The Driver manages the Disk as a cache and requests Cloud Client Process for eviction/installation requests.
- **Cloud Client Process** - responsible for eviction/installation from Cloud to the disk and writeback of dirty data.

Making Block-level Storage Smarter

Disk Cache Consistency Across Crashes

<table>
<thead>
<tr>
<th>Disk Frames</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frameld</td>
<td>1000</td>
<td>643</td>
<td>79</td>
<td>1000</td>
<td>125</td>
<td>79</td>
</tr>
<tr>
<td>CloudId</td>
<td>0</td>
<td>1000</td>
<td>0</td>
<td>1000</td>
<td>1</td>
<td>125</td>
</tr>
</tbody>
</table>

Performance - Fast Writes

File System writes do not require Cloud Blocks to be installed on Disk for execution.
- An optimization to provide Disk Like Latencies for writes.
- Pose concurrency challenges with transactional update to disk frame and **suggests check-pointing** as a smart method to ensure disk cache consistency.

Capacity - Deletes

File Systems like ext3 assumes the storage device to be a disk.
- In ext3, data liveness is inferred by block bitmaps.
- Files are deleted and data blocks are released.
- Bitmaps are reset
- The data still sits on the Disk.
- Release the data blocks which are no more in use by the File System.
- **Snoop journal => infer liveness[2]** at Block Level
- In Future, implement the trim command for explicitly deleting data.