OpenZFS Development

Lustre User Group Developer Day

May 30th, 2017
OpenZFS is Available on Multiple Platforms

Developers from all platforms contribute to OpenZFS
ZFS on Linux Releases

- **Current Release (v0.6.x)**
  - Critical bug fixes
  - Linux kernel compatibility
  - Low-risk update for distributions

- **Upcoming Release (v0.7.x)**
  - New features
  - Performance improvements
  - v0.7.0-rc4 released May, 2017
Feature User/Group Object Accounting and Quota

- Works the same as space accounting and quota
- Extended ‘zfs userspace’ command
- Existing datasets can be upgraded online
Feature Large Dnodes

Xattr's stored in dnode, single IO for all small xattrs
Meta Data Performance Improvements

- Multi-threaded TXG syncing
- Multi-threaded object allocation
- Batched quota updates
- Reduced dnode lookups (added by_dnode functions)
- Additional optimizations
Profiling with `perf` and FlameGraph\(^1\)

Original scalar implementation

Parity generation

AVX2 implementation

Data checksum

\(^1\) “Flame Graphs”, Brendan Gregg, [http://www.brendangregg.com/flamegraphs.html](http://www.brendangregg.com/flamegraphs.html)
<table>
<thead>
<tr>
<th></th>
<th>Scalar</th>
<th>SSE</th>
<th>AVX2</th>
<th>AVX512</th>
<th>NEON</th>
</tr>
</thead>
<tbody>
<tr>
<td>P generate</td>
<td>2.1</td>
<td>2.9</td>
<td>4.3</td>
<td>5.9</td>
<td>1.9</td>
</tr>
<tr>
<td>P reconstruct</td>
<td>1.0</td>
<td>1.4</td>
<td>2.1</td>
<td>2.9</td>
<td>1.4</td>
</tr>
<tr>
<td>PQ generate</td>
<td>2.0</td>
<td>8.2</td>
<td>12.8</td>
<td>16.7</td>
<td>2.7</td>
</tr>
<tr>
<td>Q reconstruct</td>
<td>3.9</td>
<td>6.2</td>
<td>11.7</td>
<td>18.1</td>
<td>4.3</td>
</tr>
<tr>
<td>PQ reconstruct</td>
<td>3.0</td>
<td>8.9</td>
<td>16.2</td>
<td>23.8</td>
<td>11.6</td>
</tr>
<tr>
<td>PQR generate</td>
<td>2.6</td>
<td>9.1</td>
<td>14.6</td>
<td>19.6</td>
<td>3.5</td>
</tr>
<tr>
<td>R reconstruct</td>
<td>14.7</td>
<td>27.8</td>
<td>54.7</td>
<td>75.3</td>
<td>24.5</td>
</tr>
<tr>
<td>PR reconstruct</td>
<td>11.4</td>
<td>33.6</td>
<td>61.3</td>
<td>87.0</td>
<td>36.1</td>
</tr>
<tr>
<td>QR reconstruct</td>
<td>7.3</td>
<td>19.5</td>
<td>38.8</td>
<td>56.0</td>
<td>26.0</td>
</tr>
<tr>
<td>PRQ reconstruct</td>
<td>7.4</td>
<td>19.8</td>
<td>38.2</td>
<td>57.0</td>
<td>33.2</td>
</tr>
</tbody>
</table>

RAIDZ Vectorization - Speed-up relative to original methods
Vectorization – Checksums

- Adapted RAIDZ infrastructure for Fletcher 4
- Micro-benchmarks, ZFS Test Suite, etc
- RAIDZ SIMD implementations:
  - avx2, sse2, ssse3, avx512f, avx512bw, neon, neonx2
- Fletcher4 checksum SIMD implementations:
  - avx2, sse2, ssse3, superscaler, avx512f, neon
Slab Caches

- Large buffers for data blocks
- Slab allocator
  - Reduces allocation cost
  - Fragmentation is complicated
- Slabs can distort the ARCs internal memory accounting
- ARC collapse occurs because slabs cannot be free until all buffers are returned

ARC - 1M allocated
System - 2M allocated
ARC Buffer Data (ABD)

- All buffers are vectors of pages
  - Minimal waste
  - Fast allocations

- ARC pages compressed in memory

- Uncompressed cache of buffers maintained as for working set

4K-16M
dbuf buffers
JBOD / Drive Management Features

- Drive monitoring
- Flexible event notification infrastructure
- Auto-online / auto-replace / hot spares
- Fault LED management
- Extended ‘zpool iostat’ and ‘zpool status’ commands
Additional Features In-Progress

- Device Removal/Evacuation - Delphix
- TRIM/Discard - Nexenta
- Native Encryption - Datto
- Declustered Parity (DRAID) – Intel
- Pool Allocation Classes – Intel
- Channel Programs – Delphix
- Scrub/Resilver Performance – Nexenta