Allocation Performance

George Wilson
gwilson@delphix.com
@zfsdude
Overview

- Allocation Overview
- Recent Performance Improvements
- Upcoming Perf Improvements
Allocation Overview -- Device selection

Devices are visited in round-robin fashion
A metaslab defines an allocatable region on a disk

Allocation Overview -- Metaslab selection

- Metaslabs are given a weight
- Sorted by weight
- Select metaslab with highest weight
- Attempt to allocate from region
What is a metaslab weight?

- Based on the free space of a metaslab
- Free space is “weighted” by the following factors
  - Weight the space down by fragmentation, if pool supports the space map histogram feature
  - Weight the space up by offset
  - Weight it up if the metaslab is currently loaded (i.e. its been recently used)
- The higher the weight the “better” the metaslab
Allocation Overview -- Block selection

Free space within a metaslab is stored in an AVL tree

- Select a free block from the next highest offset that has space (first fit)
- When space is low then pick a block that best fits the the size of the request (best fit)
What are we trying to improve

- Write performance of aged pools
  - Pools fragmentation increases over time
  - Performance suffers as pool nears full capacity
- Frag benchmark
  - Fills the pool to a specified capacity
  - Writes random data to random offsets
  - After benchmark reaches steady state, obtains the average random write IOPS
- Focused Investigation
  - Pool capacities <= 80%
  - Don’t kill performance above 80%
Looking back... 2013 improvements

- Device selection
  - zfs_mg_noalloc_threshold
- Metaslab selection (region on that device)
  - improved metaslab preloading
  - space_map histogram
  - fragmentation metric
- Block selection
  - cursor fit allocator
Defining fragmentation

- Segment-based metric

  - 16M or larger segment is 0% fragmented
  - 1K or smaller segment is 100% fragmented
  - 50% fragmentation means majority of free space is comprised of 128K segments
  - Metric is in-core only and may change in the future

<table>
<thead>
<tr>
<th>metaslab</th>
<th>offset</th>
<th>40000000</th>
<th>spacemap</th>
<th>52</th>
<th>free</th>
<th>163M</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

On-disk histogram:

- Fragmentation 80%

<table>
<thead>
<tr>
<th>Percentage</th>
<th>Segment Size</th>
<th>Count</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>100%</td>
<td>10 (1K)</td>
<td>2</td>
<td>*</td>
</tr>
<tr>
<td>98%</td>
<td>11 (2K)</td>
<td>11</td>
<td>*</td>
</tr>
<tr>
<td>95%</td>
<td>12 (4K)</td>
<td>749</td>
<td>***</td>
</tr>
<tr>
<td>90%</td>
<td>13 (8K)</td>
<td>11417</td>
<td>************</td>
</tr>
<tr>
<td>80%</td>
<td>14 (16K)</td>
<td>1654</td>
<td>*****</td>
</tr>
<tr>
<td>70%</td>
<td>15 (32K)</td>
<td>210</td>
<td>*</td>
</tr>
<tr>
<td>60%</td>
<td>16 (64K)</td>
<td>59</td>
<td>*</td>
</tr>
<tr>
<td>50%</td>
<td>17 (128K)</td>
<td>41</td>
<td>*</td>
</tr>
<tr>
<td>40%</td>
<td>18 (256K)</td>
<td>24</td>
<td>*</td>
</tr>
<tr>
<td>30%</td>
<td>19 (512K)</td>
<td>5</td>
<td>*</td>
</tr>
<tr>
<td>20%</td>
<td>20 (1MB)</td>
<td>1</td>
<td>*</td>
</tr>
<tr>
<td>15%</td>
<td>21 (2MB)</td>
<td>1</td>
<td>*</td>
</tr>
</tbody>
</table>
Looking back...

Frag Benchmark Comparison (2013)
Where are we going?

- Device selection
  - allocation throttle
- Metaslab selection (region on that device)
  - dynamic metaslab selection
- Block selection
  - hole-filling
Dynamic Metaslab Selection

- Change the weight from space to segments
  - Requires space map histogram feature
  - Encodes the largest contiguous region into the weight
  - Metaslabs with larger regions are considered “best”

**Space-based weighting:**

<table>
<thead>
<tr>
<th>PS1</th>
<th>weighted-free space</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>56</td>
</tr>
</tbody>
</table>

**Segment-based weighting:**

<table>
<thead>
<tr>
<th>PS0</th>
<th>idx</th>
<th>count of segments in region</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>56</td>
<td>48</td>
</tr>
</tbody>
</table>
Reducing Fragmentation

![Graph showing the reduction of fragmentation over pool capacity. The graph includes data for 2013 Fragmentation and 2014 Fragmentation, with 2014 IOPS increasing as pool capacity increases. The graph also shows the decrease in pool fragmentation as pool capacity increases.]
Going further

- **Hole-filling**
  - Metaslabs are sorted by holes
  - Allocate from crappy metaslabs during times of low write activity
  - Preserve pristine metaslabs for heavy write loads
What’s next

- Allocation throttle improvements
- Directed device selection
- Synchronous write improvements
Questions?
Thank You

George Wilson
gwilson@delphix.com
@zfsdude
Backup Slides
Recent changes

- `zfs_mg_noalloc_threshold`
  - percentage of free space that makes a device eligible for allocations
  - any device that does not have this percentage free is skipped
- Improved metaslab preloading
  - Load more metaslabs before we reach allocation path (avoid reading during writes)
Recent changes

- space_map histogram
  - Maintain on-disk histogram of free segments in power-of-2 buckets
    - Requires pool to be upgrade (new feature flag)
    - space maps have to be upgraded to maintain information (happens when space maps condense)
  - Ability to retrieve histogram of free segments
    - zdb -mm - provide on-disk histogram (requires feature flag)
    - zdb -mmm - add in-core histogram (requires all space maps to be loaded)
  - Running ‘zdb’ fails when pool is busy or mostly full