Per-VDEV Properties

Allan Jude -- allanjude@freebsd.org
Klara Systems
Introduction

- 17 Years as FreeBSD Server Admin
- 4 Years as FreeBSD committer
  - ZFS, installer, boot loader, GELI (FDE)
- FreeBSD Core Team (2016 - 2020)
- Co-Author of “FreeBSD Mastery: ZFS” and “FreeBSD Mastery: Advanced ZFS” with Michael W. Lucas -- ZFSBook.com
- Host of BSDNow.tv Podcast
- This is my 3rd OpenZFS Devsummit
Motivation: Per-VDEV Properties

- Most of the tunables in ZFS are system wide.
- It would be very useful if they were per-pool (SSD pool vs HDD pool), or even per-vdev.
- Different device types need different queue depths to achieve best performance/latency.
- Unbalanced LUNs and multi-device evacuation would benefit from marking devices as ‘do not allocate from here’.
How: VDEV ZAP

- As part of the Device Evacuation feature, a per-VDEV ZAP was created.
- This allows each vdev to store a key-value pair of properties.
- Applies to both top-level vdevs (mirror, RAID-Z*) and the leaf vdevs they contain.
- We can also expose many attributes of vdev_t without storing them.
Why?

- The properties interface in ZFS is well defined and well understood
- The existing tooling makes it easy to fetch many properties, display and use the data
- A better interface to retrieve data currently only available from `zpool status`
- Extended control over Fault Management with custom properties per top-level vdev
# zpool get

size@ada0p3, alloc@ada0p3, free@ada0p3, ashift@ada0p3, state@ada0p3

<table>
<thead>
<tr>
<th>NAME</th>
<th>PROPERTY</th>
<th>VALUE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>zroot</td>
<td>size@ada0p3</td>
<td>45.5G</td>
<td>-</td>
</tr>
<tr>
<td>zroot</td>
<td>alloc@ada0p3</td>
<td>19.5G</td>
<td>-</td>
</tr>
<tr>
<td>zroot</td>
<td>free@ada0p3</td>
<td>26.0G</td>
<td>-</td>
</tr>
<tr>
<td>zroot</td>
<td>ashift@ada0p3</td>
<td>12</td>
<td>-</td>
</tr>
<tr>
<td>zroot</td>
<td>state@ada0p3</td>
<td>ONLINE</td>
<td>-</td>
</tr>
</tbody>
</table>
What Properties Do You Want?

- noalloc - for unbalanced LUNs and `queued` device evacuation
- ashift - see George Wilson’s “Oh Shift!” from 2017
Challenges

• Inheritance: Should we support ‘zpool inherit sync_write_max_active@c0t0d0 tank`?
• Inherit the value from the top-level vdev, which might itself inherit from the pool?
• Could this indirection cause performance issues? Have to read leaf->top->pool every time we queue a write?
What Problems Do You Foresee?
TODO

- Get all properties (@vdev or all@vdev ??)
- ZCP Integration
- Caching?
- Inheritance
- More properties
  - All vdev_stat_t counters
  - Lifetime writes? (Flash Endurance Tracking)
  - Other information (disk serial#, SES data, smart?)
What Are Your Questions?
BSDNow.tv

- Weekly video podcast about the latest news in the BSD and IllumOS world
- Always looking for developers to interview
- Our archives are full of goodies (100+ Interviews):
  - Matt Ahrens
  - George Wilson
  - Bryan Cantrill
  - Adam Leventhal
  - Richard Yao
  - Alex Reese
  - Kirk McKusick
  - Josh Paetzel
  - Justin Gibbs
  - Paweł Jakub Dawidek
  - Sean Chittenden
  - Ryan Zezeski