MMP: Safer Pool Import With High Availability Clusters

OpenZFS Developer Summit

October 2017

Olaf Faaland
MMP: Problem Statement

Catastrophic corruption will occur if a ZFS pool is simultaneously imported on more than one host

• MMP prevents ZFS from importing a pool that is active on another host, under most circumstances
• Merged to ZFSonLinux, available from v. 0.7.0
  • https://github.com/zfsonlinux/zfs/pull/6073
MMP: Motivation

Our Use Case:
- Host B is a hot spare for Host A
- High Availability (HA) package starts services on B when A goes down
- … but what if A is not really down?

Existing Mechanisms are not sufficient
- Namespace check – scope is single host, Host B
- Hostid – Host B must always use “force” import, disabling this check
- HA package
  - May be misconfigured (we’ve done this)
  - May be fooled, e.g. by bad power control SW or HW (done this too)
Don’t Make Trouble
- Don’t change existing behavior – e.g. rollback still works
- Don’t degrade performance for non-failover users
- Preserve on-disk compatibility

Reliable
- Simple configuration – no unsafe configurations
- Communicate via devices already shared
- Detect import even if some devices are not visible to Host B
- Enable automated single-node testing to catch regressions

Available Sooner Not Later
Low Performance Impact for failover users
MMP: Where do we look for activity?

VDEV

0 256k 512k 4M  N - 512k  N - 256k  N

Label 0  Label 1  Boot Block  Pool Blocks  Label 2  Label 3

Blank  Boot Header  Name=Value Pair List  K-slot Uberblock Ring

...  256k

MMP block
MMP: Options for signaling

DMU Blocks
• Importing pool (even R/O) for reading signal is unsafe (and unreliable)

Config nvlist
• Repeatedly overwriting likely results in inconsistent reads

Uberblock ring
• Code exists for reading and writing Uberblocks
• Import is not required for such reads
• Uberblocks written by txg sync are a free activity indicator
• Quiet pools need another mechanism for reflecting change
  • Forcing a new txg we may lose rollback
  • Writing over existing slots we may lose rollback
• Partition Uberblock ring
  • Dedicate 1 slot to MMP Uberblock writes only
  • Dedicate remaining slots to txg sync Uberblock writes
MMP: Use Uberblocks for signaling

```c
struct uberblock {
    uint64_t ub_magic;
    uint64_t ub_version;
    uint64_t ub_txg;
    uint64_t ub_guid_sum;
    uint64_t ub_timestamp;
    blkptr_t ub_rootbp;
    uint64_t ub_software_version;
    uint64_t ub_mmp_magic;
    uint64_t ub_mmp_seq;
    uint64_t ub_mmp_delay;
};
```

- `ub_timestamp`: wallclock time the uberblock was written, 1-second resolution.
- `ub_mmp_magic`: used to determine whether these fields are valid.
- `ub_mmp_delay`: at time this Uberblock was written, decaying average of time between successful MMP writes.
- `ub_mmp_seq`: currently unused, but intended to provide sub-second change detection.
## MMP: Existing Import Process (abridged)

<table>
<thead>
<tr>
<th>Userspace</th>
<th>Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find devices, assemble partial config</td>
<td>Load Uberblock (latest txg &amp; timestamp)</td>
</tr>
<tr>
<td>Tryimport ioctl w/config</td>
<td>Load MOS config via root block ptr</td>
</tr>
<tr>
<td></td>
<td>Generate updated full config</td>
</tr>
<tr>
<td></td>
<td>Fetch &amp; verify other pool info</td>
</tr>
<tr>
<td>Import ioctl w/config and flags</td>
<td>Return full config nvlist &amp; import info</td>
</tr>
<tr>
<td></td>
<td>Do it all again</td>
</tr>
<tr>
<td></td>
<td>Attempt import (Possibly roll back and retry)</td>
</tr>
<tr>
<td></td>
<td>Return import info</td>
</tr>
<tr>
<td>Report result to user</td>
<td></td>
</tr>
</tbody>
</table>

Notes: (1) Illumos has other code path(s)       (2) Tryimport also used for ‘zpool status’
MMP: Complications

• Method
  • MMP thread writes Uberblocks on scheduled basis
  • Extend tryimport to return txg and timestamp
  • Userspace polls tryimport, watching for txg/timestamp change

• Problem
  • Host panics sometimes during tryimport, if the userspace-built config is stale when kernel loads MOS or compares MOS config with userspace config (could be many seconds old!)
  • Such user/kernel config coherency panics are not new; we chose to avoid them rather than trying to find and fix all such issues

• Solution
  • Perform poll in tryimport (kernel), and exit immediately if change is detected
MMP: Complications

• Method
  • MMP thread writes Uberblocks on scheduled basis
  • ✔ Tryimport polls for change (in kernel)
• Problem
  • What if there is a long delay between tryimport and import? Activity check result is no longer valid
• Solution
  • Perform poll in both tryimport (kernel) and import (kernel)
MMP: Complications

- **Method**
  - MMP thread writes Uberblocks on scheduled basis
  - ✔ Tryimport and import both poll for change (in kernel)

- **Problem**
  - User must wait 2x polling period for import to succeed

- **Solution**
  - If no activity detected, tryimport returns found txg and timestamp with config.
  - Userspace passes these values in when import ioctl issued; if they match what is found by import when the uberblock is loaded, still valid
MMP: Complications

• Method
  • MMP thread writes Uberblocks on scheduled basis
  • Tryimport polls for change
  • ✔ Tryimport records txg and timestamp
  • ✔ Import polls if txg and timestamp do not match

• Problem
  • What if user settings for MMP write period differ on Host A and B?
  • What if there are large I/O delays due to some problem?

• Solution
  • Host A records the average time between MMP writes at the end of the Uberblock.
  • Host B reads that to compute required polling period
MMP: Complications

• Method
  • MMP thread writes Uberblocks on scheduled basis
  • Tryimport polls for change
  • Tryimport records txg and timestamp
  • Import polls if txg and timestamp do not match
  • ✔ Polling period based on MMP write period recorded in Uberblock

• Problem
  • What if two hosts attempt to import pool at the same time?

• Solution
  • Add a small random term when calculating the polling period. One will finish sooner and the others will see its MMP writes
  • (caveat) If the pool was cleanly exported this is defeated – needs thought
MMP: Complications

• Method
  • MMP thread writes Uberblocks on scheduled basis
  • Tryimport polls for change
  • Tryimport records txg and timestamp
  • Import polls if txg and timestamp do not match
  • Polling period based on MMP write period recorded in Uberblock
  • ✔ Polling period includes random term for simultaneous imports

• Problem
  • How do we avoid all this for non-failover configurations?

• Solution
  • We cannot detect whether the storage is shared, so the user must tell us.
  • Introduce a property, multihost="on" means we perform activity test
    • We can also check that hostid is set when property set
MMP: Complications

• Method
  • MMP thread writes Uberblocks on scheduled basis
  • Tryimport polls for change
  • Tryimport records txg and timestamp
  • Import polls if txg and timestamp do not match
  • Polling period based on MMP write period recorded in Uberblock
  • Polling period includes random term for simultaneous imports
  • ✔ Multihost property allows user to turn MMP on

• Problem
  • Host B cannot tell whether the property is on before import

• Solution
  • When the property is off, we zero the MMP fields in Uberblock
  • Host B polls for change if MMP fields are nonzero
MMP: Merged Implementation

• Method
  • MMP thread writes Uberblocks on scheduled basis
  • Tryimport polls for change
  • Tryimport records txg and timestamp
  • Both tryimport and import skip poll if MMP fields in Uberblock zeroed
  • Import polls if txg and timestamp do not match ones from tryimport
  • Polling period is based on MMP write period recorded in Uberblock
  • Polling period includes random term for simultaneous imports
  • Multihost property allows user to turn MMP on
  • ✔ Zero MMP fields in Uberblock when multihost=off
• And…
  • MMP blocks are written to randomly selected leaves and labels at frequency \((1000 \times \text{zfs\_multihost\_interval} / \# \text{vdevs})\) Hz
  • Pool is suspended if (time since last successful MMP write) > \((1000 \times \text{zfs\_multihost\_interval} \times \text{zfs\_multihost\_fail\_intervals})\)

\((\text{zfs\_multihost\_interval \ is \ in \ milliseconds})\)
MMP: Testing

- Challenges
  - Namespace checks prevent two imports on same node
  - Hostid kernel sees will be the same for both import attempt
  - Multi-node testing much more difficult, even with VMs
- Solution: ztest is the “remote host”
  - Separate namespace since it runs entirely in userspace
  - Altered to allow hostid to be set via environment variable
  - Added option to skip some tests that halt activity to the pool
MMP: Limitations / Future Work

- MMP is defeated by long delays in I/O
  - Algorithm assumes import is safe after some period but there is no guarantee this is true
  - For example admin disconnects a SAS cable, replaces after 30 sec
  - HW/SW problems can create similar delays
- No ongoing (post-import) check
- No protection when a pool is suspended
  - Host A imports pool
  - Host A encounters errors and the pool is suspended
  - Host B imports the pool while there is no activity
  - Host A admin issues ‘zpool clear’ and resumes I/O
- MMP offers no protection to ‘zpool create/add/attach/replace’
  - For example, if a new device (no label) is added to two pools at the same time
  - The window of vulnerability is small as label writes happen early in the process
- Zpool labelclear does not check for activity
MMP: Questions?
Multi-Modifier protection for ZFS was developed by Lawrence Livermore National Laboratory.

The work was funded by Intel Inc. via a CRADA.

The design built on an earlier project, with a design authored by Ricardo Correia in 2009.

Thanks to the following individuals who provided code, tests, reviews, design feedback, ideas, and even a couple slides: Brian Behlendorf, Ned Bass, Giuseppe Di Natale, Matthew Ahrens, Andreas Dilger, and Don Brady.

Apologies to anyone I overlooked!