Strategies for running unmodified filesystems on SMR drives

Dr. Hannes Reinecke
SUSE Linux GmbH
Challenges when using SMR drives
Challenges when using SMR

- Partitioning
  - most end-users are used to partitioning devices
- Uninitialized READ
  - ZBC per default will return I/O errors when reading from uninitialized sectors
- Zone alignment: SMR devices require or at least benefit greatly from aligning data to zones
Partitioning

- GPT partitioning required
- Most partitioning tools create a GPT backup sector at the end of the disk
- Write needs to be buffered if last zone is a Sequential Write Required zone.
Uninitialized READ

- If 'UNSZW' bit isn't set any READ to an uninitialized zone will return I/O errors
- None of the drivers currently existing are aware of this peculiarity
- I/O errors will be presented to the user upon first access
- I/O needs to be buffered to prevent detrimental user experience
Zone alignment

- Most filesystems have a fixed disk layout
  - All filesystems require to have certain bits at specific, pre-defined location. This location is part of the on-disk format and cannot be changed.
  - Some filesystems either have a sequential allocation algorithm (btrfs, ZFS), or allow to specify one (ext4, xfs)
- Filesystems need to be aware of the zone layout
Sequential Write requirement

- Host-aware devices require sequential writes
- Some filesystems provide matching capabilities
  - Btrfs always tries to write sequentially, due to its CoW nature.
  - Ext4 has an SMR-optimized allocation strategy (packed_meta_blocks), which should allow for sequential writes
- XFS at the moment is not capable of ensuring sequential writes
Challenges for OS vendors

- Filesystem layout cannot be changed
  - On-disk format is required to be stable
  - Adapting filesystems possible if on-disk format isn't changed
- Adding new filesystems very unlikely
  - Only with compelling use-case
  - Not possible with existing distributions
Challenges for Linux

- SMR host-aware patches have been posted to upstream
  - Held by procedural issues (touching several subsystems)
  - ATA Sense code handling under discussion
- SMR host-managed patches pending
  - Core functionality already in 4.1
  - Extended functionality pending
General considerations

- Host-aware (and device-managed) implementations require only limited support from the OS
  - Data alignment
  - Reset Write Pointer handling
- Focus on host-managed devices
Strategies for SMR drives
Possible strategies

- Modify filesystems to match SMR capabilities
  - Requires updates to existing or entirely new filesystems
  - On-disk format likely to be changed
  - Additional support overhead for OS Vendors
  - Unknown stability
Possible strategies

- Remap unaligned I/O to CMR zones
  - Requires remapping of the entire disk
  - New on-disk format
  - Remapping functionality required for disk access
- Presentation by Albert Chen
Possible strategies

- Cache non-sequential I/O:
  - Cache entire zones
  - High memory consumption
  - On-disk format unchanged
  - No additional functionality required for access
Caching non-sequential I/O
Caching non-sequential I/O

- Per-zone writeback cache
- Zones are being read in upon access
- Two-stage writeout:
  - RESET WRITE Pointer
  - Write zone data
- Zone cache is kept until expiry or memory pressure
Caching non-sequential I/O

- Zone cache eviction:
  - User-selectable cache expiration time
  - User-selectable upper bound on number of caches
- LRU eviction:
  - Select LRU cache
  - Flush old cache contents
  - Read in new data
- Possible cache trashing depending on I/O load
Cache exceptions

- Caching of all I/O leads to heavy cache usage
  - Aligned writes can be exempted
  - Reads to initialized areas can be exempted
  - Reads to non-initialized areas can be zero-filled
  - Writes beyond WP can be zero-extended
Test results
Filesystem tests

- 'Real-life' scenario for testing:
  - Create filesystem
  - Mount filesystem
  - Copy linux kernel tarball
  - Unpacking linux kernel
  - Applying 5949 patches
  - Unmount filesystem
Filesystems for testing

- Tested with btrfs and ext4
  - Xfs suffers from heavy cache trashing
  - Xfs needs to be modified for SMR
- Standard options for btrfs
- Ext4 tweaking:
  - packed_meta_blocks, flex_bg
  - Aligned 'stride' and journal size/location to zones
## Btrfs results

<table>
<thead>
<tr>
<th></th>
<th>Standard disk</th>
<th>SMR disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mkfs</td>
<td>0.2 sec</td>
<td>1.4 sec</td>
</tr>
<tr>
<td>Mount</td>
<td>0.2 sec</td>
<td>0.4 sec</td>
</tr>
<tr>
<td>Cp</td>
<td>0.2 sec</td>
<td>0.4 sec</td>
</tr>
<tr>
<td>Tar</td>
<td>17.2 sec</td>
<td>17.6 sec</td>
</tr>
<tr>
<td>Patch</td>
<td>133.1 sec</td>
<td>136.0 sec</td>
</tr>
<tr>
<td>Umount</td>
<td>2.2 sec</td>
<td>12.3 sec</td>
</tr>
</tbody>
</table>
# ext4 results

<table>
<thead>
<tr>
<th>Operation</th>
<th>Standard disk</th>
<th>SMR disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mkfs</td>
<td>3 sec</td>
<td>(28 sec)</td>
</tr>
<tr>
<td>Mount</td>
<td>0.4 sec</td>
<td>0.4 sec</td>
</tr>
<tr>
<td>Cp</td>
<td>4.8 sec</td>
<td>4.8 sec</td>
</tr>
<tr>
<td>Tar</td>
<td>13.6 sec</td>
<td>13.8 sec</td>
</tr>
<tr>
<td>Patch</td>
<td>135.1 sec</td>
<td>133.0 sec</td>
</tr>
<tr>
<td>Umount</td>
<td>3.1 sec</td>
<td>41.8 sec</td>
</tr>
</tbody>
</table>
Btrfs zone usage
Btrfs zone usage
Ext4 zone usage
Ext4 zone usage
Summary
Result summary

- SMR writeback caching achieves performance comparable to native usage
- Cache efficiency on ext4 better than on btrfs:
  - 46% vs. 78% aligned cache accesses
- Low zone usage on btrfs offsets reduced cache efficiency
Btrfs result summary

- Btrfs operation matches SMR parameters very closely:
  - Low concurrent zone usage: nearly all writes are aligned
  - Low overall zone usage: nearly all writes are sequential
- High number of misaligned write accesses; points to an issue with btrfs itself
Ext4 result summary

- Less efficient zone usage
- Performance comparable to btrfs
- High number of cached zones
  - Writeback might be an issue
- Frequent cache flushes
  - FUA causes cache to be flushed
Misaligned I/O handling

- Misaligned I/O (ie I/O beyond the write pointer) quite common
- Implemented with normal WRITE commands, writing NULLs
- Switch to WRITE SAME might increase performance
Summary

- Using a per-zone writeback cache allows the use of unmodified filesystems
- Suitable for btrfs and ext4
- Performance comparable to native filesystem usage
- Increased memory usage
Thank you!