The road for thin-provisioning

Paolo Bonzini
Red Hat, Inc.
November 7th, 2012
Overview

- Why is thin provisioning important?
- Current status
- Where should we go?
- Implementation overview
- Summary
Logical block provisioning

- A disk is made of many logical blocks
- The user tells the disks how it's using them
  - The user can use its allocated resources better
  - The disk gains in speed and durability
- Automatic or manual
  - `mount -o discard`
  - `fstrim`
Logical block provisioning

- Obviously extends to virtualization
  - The user is the guest administrator
  - The disk is the storage backend
- Same tools
  - Automatic management: “mount -o discard”
  - Manual management: fstrim
  - Also via guest agent
Why is it useful?

- Guest admin only pays for actually used space
- Host admin reaps all the other benefits
  - Saved disk space
  - Improved wear-leveling for SSD
  - Shorter maintainance operations
Strategies

- In the storage
  - SCSI passthrough
- In the kernel
  - Raw images or logical volumes
- In QEMU
  - QCOW2 and other image formats
SCSI logical block provisioning

VPD page 0xb2

- LBPU: UNMAP command supported
- LBPWS/LBPWS10: WRITE SAME supported
- LBPRZ: “unmapped” blocks read zero
- ANC_SUP: ANCHOR supported
- Provisioning type

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNMAP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ANCHOR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRITE SAME</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ANCHOR</td>
<td>UNMAP</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET LBA STATUS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SCSI passthrough

- Entire LUNs passed to a guest via a virtual SCSI adapter
  - virtio-scsi, megasas, ...
- QEMU acts as a bridge to the LUN
  - SCSI commands passed 1:1
  - iSCSI via kernel or userspace initiator (libiscsi)
  - Other transports (SAS, FC,...) only via kernel
SCSI passthrough

- Logical block provisioning is enabled on the storage
- Mostly available on high-end disks
SCSI passthrough

• One advantage: available now :)

but...

• Needs libiscsi or CAP_SYS_RAWIO
• All maintenance is done outside QEMU
  • LUN configuration
  • Live block operations (snapshoting, migration,...)
Raw images or volumes

- Images stored on a file, partition or LV
- SCSI command set emulated by QEMU
- Limited feature set
  - No live snapshots
  - No image templates
- Easy access to underlying file system or device features
SCSI logical block provisioning

VPD page 0xb2

- LBPU: UNMAP command supported
- LBPWS/LBPWS10: WRITE SAME supported
- LBPRZ: “unmapped” blocks read zero
- ANC_SUP: ANCHOR supported
- Provisioning type

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ANCHOR</td>
<td></td>
</tr>
</tbody>
</table>

UNMAP

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ANCHOR</td>
<td>UNMAP</td>
<td></td>
</tr>
</tbody>
</table>

WRITE SAME

GET LBA STATUS
SCSI logical block provisioning

- Three types of disks
  - Fully-provisioned
  - Thin-provisioned
  - Resource-provisioned

- Three types of blocks
  - Deallocated
  - Anchored
  - Mapped

Logical block management enabled

On disk space allocated

Block not in use
SCSI logical block provisioning

<table>
<thead>
<tr>
<th>Command</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNMAP WRITE SAME</td>
<td>all -&gt; any (typically: deallocated)</td>
</tr>
<tr>
<td>+ anchor</td>
<td>all -&gt; anchored, mapped</td>
</tr>
<tr>
<td>WRITE</td>
<td>all -&gt; mapped</td>
</tr>
</tbody>
</table>
## SCSI commands vs. system calls

<table>
<thead>
<tr>
<th></th>
<th>File</th>
<th>Block</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UNMAP WRITE SAME</strong></td>
<td><strong>fallocate</strong>(FALLOC_FL_PUNCH_HOLE)</td>
<td><strong>ioctl</strong>(BLKDISCARD)</td>
</tr>
<tr>
<td><strong>UNMAP WRITE SAME</strong></td>
<td><strong>xfsctl</strong>(XFS_IOC_ZERO_RANGE)</td>
<td>?</td>
</tr>
<tr>
<td><strong>+ anchor</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>GET LBA STATUS</strong></td>
<td><strong>lseek</strong>(SEEK_HOLE/SEEK_DATA)</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td><strong>ioctl</strong>(FIEMAP)</td>
<td></td>
</tr>
</tbody>
</table>
## SCSI commands vs. system calls

<table>
<thead>
<tr>
<th>Feature</th>
<th>File</th>
<th>Block</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNMAP WRITE SAME</td>
<td><code>fallocate(FALLOC_FL_PUNCH_HOLE)</code></td>
<td><code>ioctl(BLKDISCARD)</code></td>
</tr>
<tr>
<td>+ anchor</td>
<td></td>
<td><code>fallocate</code></td>
</tr>
<tr>
<td>GET LBA STATUS</td>
<td><code>lseek SEEK_HOLE/SEEK_DATA</code></td>
<td><code>lseek SEEK_HOLE/SEEK_DATA</code></td>
</tr>
<tr>
<td><code>ioctl(FIEMAP)</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### SCSI commands vs. QEMU block layer

<table>
<thead>
<tr>
<th>Command</th>
<th>QEMU Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNMAP WRITE SAME</td>
<td><code>bdrv_discard</code></td>
</tr>
<tr>
<td>UNMAP WRITE SAME + anchor</td>
<td>Not supported</td>
</tr>
<tr>
<td>GET LBA STATUS</td>
<td><code>bdrv_is_allocated</code></td>
</tr>
<tr>
<td></td>
<td><code>(Allocated, search backing file)</code></td>
</tr>
</tbody>
</table>
# SCSI commands vs. QEMU block layer

<table>
<thead>
<tr>
<th></th>
<th>-drive prov=thin</th>
<th>-drive prov=full</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UNMAP WRITE SAME</strong></td>
<td>bdrv_discard</td>
<td>bdrv_anchor</td>
</tr>
<tr>
<td><strong>UNMAP WRITE SAME + anchor</strong></td>
<td></td>
<td>bdrv_anchor</td>
</tr>
<tr>
<td><strong>GET LBA STATUS</strong></td>
<td></td>
<td>bdrv_is_allocated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Deallocated, anchored, mapped, search backing file)</td>
</tr>
</tbody>
</table>
QCOW2 metadata

- “L2 table” holds pointers to data + some flags
- One entry per allocated cluster in L2 table
- Zero offset = search in backing file
- ZERO flag implies offset must be zero

![Diagram of QCOW2 metadata structure]

- More flags
- Block offset in the image
- Zero flag indicated by the last bit
QCOW2 metadata

- Add a new flag; if set, reads search backing file
  - ... even if offset is non-zero
  - ZERO clusters can have nonzero offset too
- UNMAPPPED + zero offset = deallocated
- UNMAPPPED + nonzero offset = anchored

<table>
<thead>
<tr>
<th>63</th>
<th>56 55</th>
</tr>
</thead>
<tbody>
<tr>
<td>more flags</td>
<td>block offset in the image</td>
</tr>
</tbody>
</table>

UNMAPPPED | ZERO
Discard/anchor on QCOW2 files

**Discard**
- Set unmapped bit
- Zero offset
- Add cluster to free list
- Discard the data in cluster

**Anchor**
- Set unmapped bit
- Anchor the data in the cluster
QCOW2 metadata preallocation

- Metadata is pre-initialized, clusters point to discarded regions (holes)

```
  0   0   0
  0  131072  0
  0   65536  0
  0  196608  0
```

- Only possible on bottom-level images!
QCOW2 metadata preallocation

- Metadata is pre-initialized, clusters point to unmapped regions

- Now works also with a backing file!
Streaming from backing files

- Data copied from backing file for faster access
Streaming from backing files

- QEMU 1.3 lets backing file data through after discard
Streaming from backing files

- “Real” discard ignores unmapped blocks
Summary

- Benefits of logical block management:
  - Saved disk space
  - Disk durability (SSD)
  - Metadata preallocation for improved performance
  - Shorter maintainance operations

- Storage configurations supported:
  - SCSI passthrough
  - Raw images or QCOW2
  - Files, partitions, logical volumes
Todo list

Kernel:
- Simplify passthrough of UNMAP & WRITE SAME
- BLKANCHOR ioctl
- Improve lseek for block devices (GET LBA STATUS)

QEMU:
- -drive prov=...
- Discard/anchor for files and host devices
- QCOW2 discard & anchor forwarding
- QCOW2 unmapped bit support
- Optimized streaming
Questions?