RAID 4/5/6 cache

Raid 4/5/6 could include an extra disk for data cache besides normal RAID disks. The role of RAID disks isn't changed with the cache disk. The cache disk caches data to the RAID disks. The cache can be in write-through (supported since 4.4) or write-back mode (supported since 4.10). mdadm (supported since 3.4) has a new option '--write-journal' to create array with cache. Please refer to mdadm manual for details. By default (RAID array starts), the cache is in write-through mode. A user can switch it to write-back mode by:

```
echo "write-back" > /sys/block/md0/md/journal_mode
```

And switch it back to write-through mode by:

```
echo "write-through" > /sys/block/md0/md/journal_mode
```

In both modes, all writes to the array will hit cache disk first. This means the cache disk must be fast and sustainable.

write-through mode

This mode mainly fixes the 'write hole' issue. For RAID 4/5/6 array, an unclean shutdown can cause data in some stripes to not be in consistent state, eg, data and parity don't match. The reason is that a stripe write involves several RAID disks and it's possible the writes don't hit all RAID disks yet before the unclean shutdown. We call an array degraded if it has inconsistent data. MD tries to resync the array to bring it back to normal state. But before the resync completes, any system crash will expose the chance of real data corruption in the RAID array. This problem is called 'write hole'.

The write-through cache will cache all data on cache disk first. After the data is safe on the cache disk, the data will be flushed onto RAID disks. The two-step write will guarantee MD can recover correct data after unclean shutdown even the array is degraded. Thus the cache can close the 'write hole'.

In write-through mode, MD reports IO completion to upper layer (usually filesystems) after the data is safe on RAID disks, so cache disk failure doesn't cause data loss. Of course cache disk failure means the array is exposed to 'write hole' again.

In write-through mode, the cache disk isn't required to be big. Several hundreds megabytes are enough.

write-back mode

write-back mode fixes the 'write hole' issue too, since all write data is cached on cache disk. But the main goal of 'write-back' cache is to speed up write. If a write crosses all RAID disks of a stripe, we call it full-stripe write. For non-full-stripe writes, MD must read old data before the new parity can be calculated. These synchronous reads hurt write throughput. Some writes which are sequential but not dispatched in the same time will suffer from this overhead too. Write-back cache will aggregate the data and flush the data to RAID disks only after the data becomes a full stripe write. This will completely avoid the overhead, so it's very helpful for some workloads. A typical workload which does sequential write followed by fsync is an example.

In write-back mode, MD reports IO completion to upper layer (usually filesystems) right after the data hits cache disk. The data is flushed to raid disks later after specific conditions met. So cache disk failure will cause data loss.

In write-back mode, MD also caches data in memory. The memory cache includes the same data stored on cache disk, so a power loss doesn't cause data loss. The memory cache size has performance impact for the array. It's recommended the size is big. A user can configure the size by:

```
echo "2048" > /sys/block/md0/md/stripe_cache_size
```

Too small cache disk will make the write aggregation less efficient in this mode depending on the workloads. It's recommended to use a cache disk with at least several gigabytes size in write-back mode.

The implementation

The write-through and write-back cache use the same disk format. The cache disk is organized as a simple write log. The log consists of 'meta data' and 'data' pairs. The meta data describes the data. It also includes checksum and sequence ID for recovery identification. Data can be IO data and parity data. Data is checksumed too. The checksum is stored in the meta data ahead of the data. The checksum is an optimization because MD can write meta and data freely without worry about the order. MD superblock has a field pointed to the valid meta data of log head.

The log implementation is pretty straightforward. The difficult part is the order in which MD writes data to cache disk and RAID disks. Specifically, in write-through mode, MD calculates parity for IO data, writes both IO data and parity to the log, writes the data and parity to RAID disks after the data and parity is settled down in log and finally the IO is finished. Read just reads from raid disks as usual.

In write-back mode, MD writes IO data to the log and reports IO completion. The data is also fully cached in memory at that time, which means read must query memory cache. If some conditions are met, MD will flush the data to RAID disks. MD will calculate parity for the data and write parity into the log. After this is finished, MD will write both data and parity into RAID disks, then MD

can release the memory cache. The flush conditions could be stripe becomes a full stripe write, free cache disk space is low or free in-kernel memory cache space is low.

After an unclean shutdown, MD does recovery. MD reads all meta data and data from the log. The sequence ID and checksum will help us detect corrupted meta data and data. If MD finds a stripe with data and valid parities (1 parity for raid4/5 and 2 for raid6), MD will write the data and parities to RAID disks. If parities are incompleted, they are discarded. If part of data is corrupted, they are discarded too. MD then loads valid data and writes them to RAID disks in normal way.