Logs on Logs on Logs No More

Append Atomic & Remap

Eric Mackay
Venkatesh Srinivas
Basics of Block Device Interfaces

- I/O is done in granularity of blocks
  - 512 bytes is pretty standard
  - Writing is sloooooooow
- Data is addressable by block number
Basics of an FTL

- Interface has access to logical block address
- Flash Translation Layer maps the logical block address to a physical block on the device
- Wear-levelling extends lifetime of device by writing new data to a different physical block and mapping the LBA to that location
Write-Ahead Logging

- To make data updates durable and consistent:
  - First write update to log
  - When update in log is durable, then write update to actual data
- If we experience a failure, consistent state can be recovered from any intermediate state
Write-Ahead Logging on an FTL

● WAL performs 2 writes for every modification to data
● Wear-levelling not done across entire disk
  ○ Modifying same data repeatedly will wear out some regions faster
Write Atomic & Write Scattered, Atomic

● SCSI / T10 added Write Atomic command
  ○ Writes data as atomic operation
  ○ Can only be contiguous blocks
● Databases need a scattered Write Atomic
  ○ Ability to write non-contiguous blocks atomically
  ○ Proposed but not accepted
● Some vendor-specific solutions
Append Atomic & Remap

- Write data to anonymous area of disk with Append Atomic
  - Return a token (ROD) identifying the data
  - Data not visible to anyone

- Remap the flash translation layer
  - Provide Logical Block Address and token
  - Associates LBA with the physical address corresponding to token
  - Free physical blocks that LBA used to point to

- Only intermediate state: data written but not remapped
  - Completely fine since it’s not addressable

- Also a good interface for SMR drives, not just SSDs
Append Atomic & Remap Example
Append Atomic & Remap Example

Append Atomic

Device

Anonymous

FTL

LBA 0
LBA 3
LBA 1
LBA 2

Data 0
Data 1

Anonymous

Token 0x1A6D
Token 0x2FF0
Append Atomic & Remap Example
Atomicity

Across a crash cycle, RODs not preserved

FTLs already support internal atomic update (for contiguous writes spanning channel / stripe boundaries)

If not, FTL could double-buffer map updates
Vs. Write Scattered

Pros :=
Deeper I/O concurrency -- overlap between flush groups is ok;
Does not require all data to be available at once
Vs. Write Scattered

Cons :=
Easy to fragment FTL tables with injudicious use of REMAP
Appended-but-not-remapped data is not visible; applications need to lock associated buffers until remap is done.
Implementation details

- Built a prototype using QEMU + iSCSI target
- Maintains in-core ‘divert’ table for non-linear regions of the disk
- Accessed via vendor-specific SCSI command
- Writes diverted blocks into a ‘divert’ file.
SQLite WAL vs. SQLite Append+Remap

Coming soon...