Linux Plumbers Conference 2010

# Hot Cold Data Tracking and Migration in btrfs

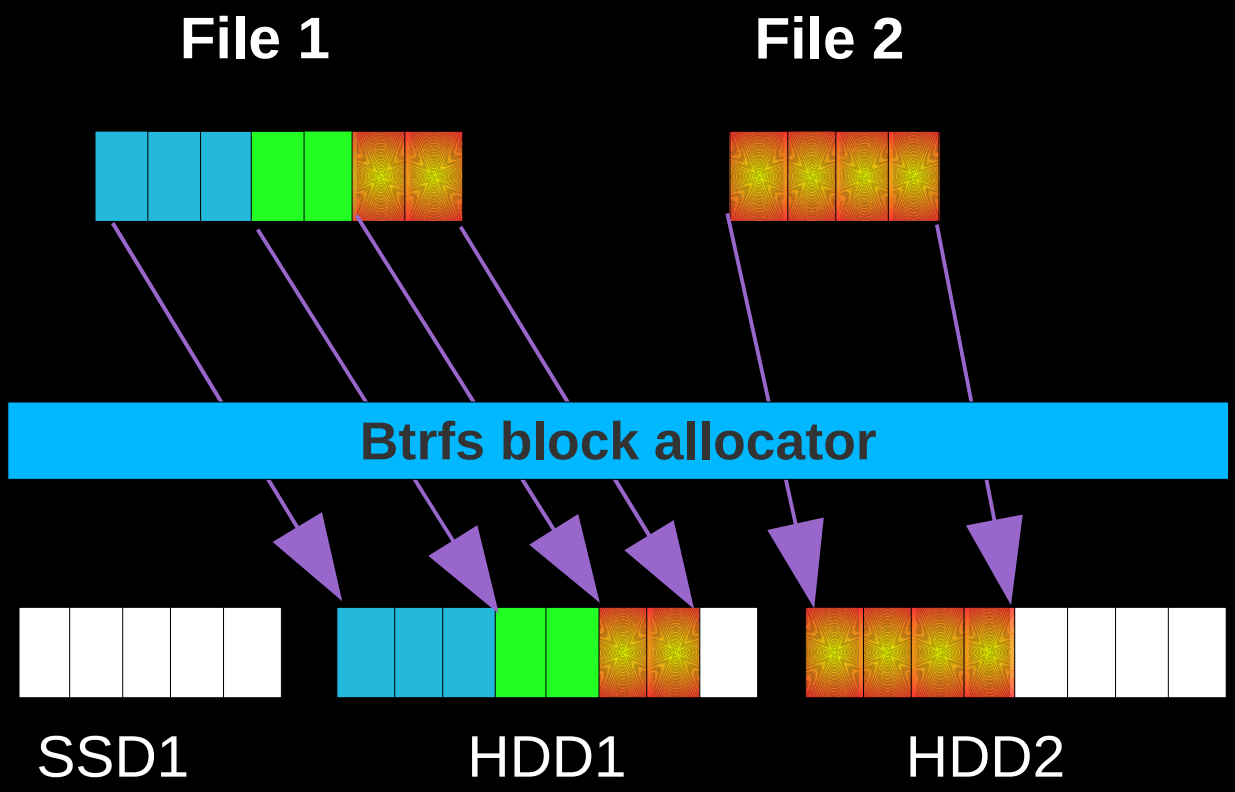


Nov 3rd, 2010 Mingming Cao <cmm@us.ibm.com>

# Problem

- Fast solid state drives provide high IOPS and low capacity, While traditional hard disks have large capacity
- Combine both SSD and HDD together could
  - Enhanced I/O performance
  - Enhanced Capacity utilization
  - Reduced power consumption
- More often applications have data are highly accessed and data rarely used, filesystem spread them across entire
- Got to place hot data on fast disks!
- Userspace could tracking data heat and do relocation manually, but could only do at the file level

# Current data placement



# Proposal

- Track and detect the hot data on the filesystem
- Automatically migrate hot data to fast disks(SSD)

# High Level Implementation Specifics

- Record file and sub-file access information in Btrfs
  - Filesystem implementation has unique advantages
- Relocate hot data
  - Move hot files and ranges to SSDs
  - Move cold data back to spinning disks
- Hybrid (SSD increases overall space), not tiered (cache)
  - Leverage Copy-on-Write for easy migration to SSD
  - Advantages in reducing capacity and energy
- Tracking and relocation done automatically, but tunable
  - Modify frequency, granularity of migration
  - Allow marking certain data as hot or cold (e.g. metadata, certain file types, etc.)

# Challenges

- Tracking
  - Track the real disk IO access vs IO hit in page cache
  - Quick update the access counter vs quick look up the heat info
  - Tracking range granularity,  large chunk of contiguous extent
  - When should we stop tracking?
- Relocating
  - ENOSPC issue, we need a efficient way to move cold data back to slow disks when it gets cold
  - Hot data already moved to SSD still need to be tracked
  - Thrashing issue, prevent frequent balancing between SSDs and HDDs

# Where are the hot spots?

- Tracking data access frequency
- Converting data access frequency to a heat tempreture
- Sorting data based on temperature

# Tracking ....

- Track frequently accessed files and ranges within files
- Track at logic {inode,offset, bytes} pairs, rather than physical blocks
  - COW will spread the hot writes into different places
- Track every read/write, add hooks on IO path – buffered and Direct IO read/writes
- Track the access frequency on recent accessed data
  - Not interested in data haven't been accessed in last few days!

# Getting the temperature ...

- Add data structures to hold access counters

```
struct btrfs_freq_data {
        struct timespec last_read_time;
        struct timespec last_write_time;
        u32 nr_reads;
        u32 nr_writes;
        u64 avg_delta_reads;
        u64 avg_delta_writes;
        u8 flags;
        u32 last_temp;
};
```

- Frequent data are used to determine temperature of files and file ranges. Two types of frequency data
  - FREQ_DATA_TYPE_INODE
  - FREQ_DATA_TYPE_RANGE
- btrfs_get_temp() is responsible for converting those 6 heat criteria into a single temperature, range from 0 to HEAT_MAX_VALUE

# Finding the hot spot ...

- Heat info are cached/indexed in two ways
  - One by inode, offset, for quick update data access frequency
  - One by heat value, for quick look up the hot spot of the filesystems
- Two tiers of tracking and caching
  - Inode heat info tree/hashlist
  - Range heat info tree/hashlist

# Migrate hot data to fast disks ...

- Aware of where are the fast disk(SSDs)
- A dedicated fast allocation group is built on top of fast disks

  BTRFS_BLOCK_GROUP_DATA_SSD.
- Relocator looks up the heat cache indexed by heat value to identify the hot/cold data to migrate
- Using hashlists indexed by heat to ensure constant look up time
- Migration is triggered by two watermarks, those watermarks are adjustable and sensible to SSD space pressure

# Handling frequent writes to fast disk

- COW write the data to new location via cow_file_range()
- Checking inode/range temperature and set the allocation preference
  - EXTENT_PREFER_ROTATING
  - EXTENT_PREFER_NONROTATING
- Translate extent preferred allocation location to right chunk type
  - chunk_type to 1 to indicate write to BTRFS_BLOCK_GROUP_DATA
  - chunk_type 2 to indicate BTRFS_BLOCK_GROUP_DATA_SSD
- Underlying volume management layer will use the chunk_type hint to determine where to do the allocation
- The real placement info will be saved

# Handling frequent reads from disk

- Using a background kernel threads to catch the frequent reads from disk and relocate them to fast disks
- Again, walk through the heat cache and select the hot ranges to migration
- Translate the range (logic offset,bytes) to the current extent covering that range
- Set the allocation preference of the extent on the right disks
- Similar to online defrag, bring those hot data to page cache and let the writeback auto flush data to preferred location
- Flag the range with current placement info

# Migrate cold data back to slow disks ...

- Data are placed on slow disks are drop off from tracking list when they haven't been updated for a while
- Data getting cold on fast disks are to move back to slow disks
  - Triggered by SSD space pressure
  - Avoid thrashing by limit the cold space to be moved at once
  - background relocator prepare a cold list before moving hot data to fast disks
  - Set the allocation preference to slow disk and do similar thing like move data to SSD

# Others

- Expose the heat information out …
  - Using debugfs to export whole filesystem data temperatures, used to find the hotnest file in the filesystem
  - Retrieve specified file heat information
- Makes the tracking and migration optional
  - Turn on/off tracking and/or migration per file bases or overall filesystem
- Initial evaluation shows 5x performance gain with only 20% space add by SSDs

# Challenges and todos....

- ENOSPC issue
  - When the slow disks are filled up and new data coming
  - Hard to detect cold data placed on SSD after filesystem remount
- Preserve the hot inode temperature and hot range info across file remount
- Move cold data back to original location
- Move filesystem metadata to SSDs
  - Perhaps add a new type of block allocation group on SSD for metadata

# Credits and Thanks

- Ben Chociej
- Matt Lupfer
- Conor Scott
- Chris Mason
- Steve French
- Steve Pratt

Questions / Comments?
cmm@us.ibm.com