Discussion about Persistent Client Cache, End-to-End Data Integrity (T10PI), Lazy Size on MDT

Shuichi Ihara

Li Xi

Li Dongyang

DDN Storage

Background of PCC

- **Persistent Client Cache**
  - Lustre client side cache which uses SSD/NVMe as local cache to speedup applications with certain I/O patterns
- **Two modes**
  - RW-PCC (LU-10092) keeps readwrite cache on local SSD/NVMe of a single client
  - RO-PCC (LU-10499) keeps readonly cache on local SSDs/NVMe of multiple clients
- **Work**
  - LU-10499, patch 29347: add RW-PCC feature
  - LU-10499, patch 29347: add RO-PCC feature
  - LU-10499, patch 31868: Add new DLM mode PCCRO for RO-PCC
  - LU-10918, patch 32022: Rule based auto PCC caching when create files
  - LU-10602, patch 31161: add file heat support

Q: How about add a new DLM mode PCCRO?

- **What is DLM mode of PCCRO?**
  - A new DLM mode of file data which is compatible with data read but conflict with data write
- **Why PCCRO?**
  - When a file is fetched to RO-PCC, original implementation grabs a LDLM grouplock of the file to protect the data
  - Problem: other clients without RO-PCC can not even read the data
  - PCCRO allows other clients to read from OSTs normally, yet prevent modification of the data
- **Any concern?**

Q: Which kind of policy do we need for auto-caching?

- **When a file is being created, a rule based policy can be used to determine whether it will be cached in RW-PCC or not (LU-10918)**
- **Use rule expression similar with NRS TBF policy**
- **Rule example:**
  - Only cache the new files to when 1.1) project ID is either 500, or 1000 and 1.2) suffix of file name is "doc" or 2) user ID is 1001
  - porjid={500 1000}&fname={*.doc},uid={1001}
- **What attributes are necessary in the rule**
  - Project ID/UID/GID/file name/JobID/

How to use file heat for PCC?

- **File heat is a relative attribute of files/objects which reflects the access frequency of the files/objects.**
  - LU-10602
- **Cache prefetch/eviction based on file heat**
- **Need management tools to apply suitable policies**
- **Which kind of interfaces do the management tools need?**
  - ioctl() to get the file heat
  - ioctl() to set/clear the file heat
  - /proc entry to print a short list of files with the highest heats

How many HSM archive IDs are needed?

- **PCC needs to run HSM copytool with unique archive ID on each client**
- **Currently only 32 archive IDs are supported**
- **How many IDs do we need to support in the future**
  - 128?
  - 1024?
  - Is it possible to extend the upper limitation to infinite?

Background of T10PI

- **T10PI: End-to-end data integrity to prevent silent data corruption**
- **A lot of disks and HBAs are starting to support T10PI**
- **We are trying to add end-to data integrity based on T10PI for Lustre**
  - RPC checksum will be integrated into this new framework
  - Lustre client, server and storage will be integrated together to prevent data corruption
- **Ticket:LU-10472**
  - Patch 30792: add T10PI support for BIO
  - Patch 30980: add T10PI support for RPC checksum
  - Patch 31513: add T10PI support for page cache

Q: When to verify the T10PI checksum?

- **Why that is a problem?**
  - More verifications cause performance impact
  - More verifications can help to find inconsistency earlier and make it easier to isolate the problematic component
- **Verify when receive RPC**
  - Reason: network RPC checksum verification is needed any way
- **Verify before submitting the BIO to disk**
  - Reason
    - HBA/disk might not support T10PI
    - No verification means risk of submitting broken data to disk
  - Reason not to: the HBA/Disk will check later if T10PI is supported
- **Verify after reading the data from disk**
  - Reason: data could be wrong at the very beginning
- **Verify before copy the page to user-space buffer**
  - Reason: that is the last chance to detect corrupted data
- **Possible solutions: /proc options to disable/enable verifications**

Q: How to add more fault injections?

- **Why fault injection?**
  - Most of the time, no data corruption happens
  - To make sure T10PI is able to detect as expected
- **Fault injection when sending/receiving RPC to simulate network issues**
  - Already exists, need to add T10PI support
- **Fault injection when submitting I/Os to disk**
  - To check whether HBA/disk detects error correctly
- **Fault injection on client/server side page cache**
  - To simulate memory corruption
- **How to add fault injection to simulate data corruption on HBA/disk?**
  - Need that to check whether HBA/disk can detect data corruption correctly
  - Any interfaces of HBA/disk we can use?

Q: How to add ZFS support for T10PI?

- **Currently only T10PI based on Ldiskfs is supported**
- **ZFS has its own end-to-end checksums**
  - ZFS stores checksum of each block in its parent block pointer
  - ZFS end-to-end data integrity doesn't require special hardware
  - Whether or how to combine ZFS-checksum with T10PI support of Lustre?

Q: How to add MDT support for T10PI?

- **Currently T10PI support is only implemented for OST**
- **Why OST support is relatively easy?**
  - Lustre OSD (ldiskfs) submits BIO directly
  - BIO has T10PI support
- **Why MDT support is hard?**
  - MDT uses functions provided by backend file systems instead of raw BIO operation
  - LDISKFS/Ext4 does not support T10PI by itself
- **Silent metadata corruption of file system is less likely**
  - A lot of internal check in metadata operations
- **HBA/Disk level T10PI support should be enabled**
  - T10PI support for multiple device driver (md-raid)
  - T10PI support for LVM

Q: How to avoid re-calculation of T10PI checksums

- **When reading data from OSS, the T10PI checksum will needed to be re-calculated if the data is in page cache**
- **This introduces overhead and is risky**
  - Need benchmarks to check how much performance decline
  - Introduce risk of un-protected data that is vulnerable to memory corruption
- **Why need to re-calculate T10PI checksum?**
  - There is no place to keep the T10PI checksum of a page of data
  - Size of “struct page” can not be enlarged at all
  - Reading from OST disk is fine, since the T10PI checksum can be copied from BIO
  - Client side is fine, because “struct cl_page” can be used
- **Any idea about how to avoid that?**

Q: How to use APP/REF tags of T10PI

- **Reference (REF) Tag**
  - Detects data writing to incorrect blocks
- **Application (APP) Tag**
  - Specific to applications
  - Application Tag defines the purpose of data
- **Guard Tag:**
  - Protects the data portion of the sector with CRC.
- **Currently, only guard tag of T10PI is used by Lustre**
- **Add userspace API to add APP tag?**
- **Fill APP tag it with FID and verify it when reading?**
- **Fill file/object offset to REF tag and verify it when reading?**

User-space API for T10PI?

- **Why?**
  - Allow user-space applications to attach protection information data when write and to verify the information when read
  - “Real” end-to-end integrity from application to disk
- **Darrick J. Wong (Oracle) made a Linux kernel patch to provide userspace PI passthrough via AIO/DIO**
  - https://lwn.net/Articles/592113/
  - Not merged
- **Whether to add user-space interfaces of Lustre file system for T10PI?**
  - ioctl()?

Background of Lazy Size on MDT

- **An estimation of file size is saved as a new extended attribute on MDT to speed up scanning of MDT**
- **The accuracy of the LSOM is not guaranteed**
- **To accelerate metadata scanning so as to apply policy rules based on file sizes**
- **A helper tool is provided to sync LSOM periodically by parsing Lustre changelog**
- **Ticket: LU-9538**
  - Patch 29960: Add lazy size on MDT
  - Patch 30124: Tool for syncing file LSOM xattr

Q: How to enable lazy stat for applications?

- **Add a new mount option "-o lazy_stat"**
  - Need to remount client
  - Problem: different applications on the same client might has different requirement about the file size accuracy
- **Add support for statx(2) with AT_STATX_DONT_SYNC**
  - All applications can use that syscall to accelerate stat operation
  - Problem: Need to change applications
- **Enable/disable lazy stat according to policy based on Job ID?**
  - If Job ID of an application matches a certain rule, use lazy stat, otherwise use normal stat
  - Advantage: no need to change application, flexibility

How to use LSOM for policy engine/scanning tools?

- **Scanning tools of MDT (e.g. LiPE) can now get all metadata from MDT, including file size**
- **Scanning tools should not take LSOM as strict file size**
- **Robinhood can read lazy file size quickly through LSOM**
  - Is the interface of getxattr() on MDT enough for Robinhood?

Thank you!