High Level Design for Mount Permission

Fan Yong

2007-12-10

1 Introduction

Lustre should provide an utility for system mount permission configuring. Such utility allow administrator to grant, revoke, modify, query mount permission configuration for specified client(s), or list all the mount permission configuration in specified lustre system. It is a new feature for lustre 1.8. The user level utility is "lctl mount_perm".

2 Requirements

"lctl mount_perm" works on MGS node. Since one MGS node can manage multiple lustre systems concurrently, the utility should be able to process the mount permission configuration for any lustre system on the MGS node separately.

- Support to initialize mount permission (load the last configuration from MGS disk partition) when MGS service starts up.
- Support to reconfigure mount permission for specified lustre system on-line.
- Support to add specified mount permission configuration for specified lustre system on-line.
- Support to delete specified mount permission configuration for specified lustre system on-line.
- Support to modify specified mount permission configuration for specified lustre system on-line.
- Support to query mount permission configuration for specified client(s) in specified lustre system on-line.
- Support to list all mount permission configuration for specified lustre system on-line.

In this design, we do not support mount permission configuring off-line. We will support such feature in the future if necessary.

3 Functional specification

3.1 Lustre mount permission types

Lustre mount permission includes three types as following:

```
NA: No Access, forbid to mount.
RO: Read Only permission.
RW: Read and Write permission, it is the default mount permission.
```

3.2 Utility command format

The "lctl mount_perm" utility supports the following command formats, and it works with MGS service running.

```
lctl mount_perm fsname <-A | -a | -D | -d | -M | -m | -r> [configuration_file]
lctl mount_perm fsname -q [nid(s) | net(s) | default | *]
lctl mount_perm -h
```

- The "fsname" is used for specifying the lustre system, it can be obtained from "/proc/fs/lustre/mgs/MGS/filesystems" on MGS node.
- The "configuration_file" is a text mode file (can be put anywhere on MGS node), each line contains one configuration items, and each configuration items consists of two parts, as following:

  ```
  {nid | net} {perm[,perm...]}
  @nid: client identifier, e.g. "192.168.1.23@tcp"; "default" indicates any unspecifi
  @net: client network identifier, e.g. "tcp1"
  @perm: mount permission, separated with ',', "*" indicates any permission.
  ```

  "net" can match a rang of "nid", it's priority is lower than "nid", but higher than "default". "*" has the same priority with "nid". If does not set "default" configuration item (or be deleted) in the system, then the system default mount permission is "RW" permission. In the following design, the "nid" case is applicative for the "net" case also.

  It supports specifying several permissions separated with ',' on one line for expanding some new types of mount permission in future. But permissions belong to the same nid must be compatible. Now, "NA" / "RO" / "RW" are all incompatible, so "nid NA,RO,RW" is invalid configuration, but "nid RO,RO,RO" is valid one, it is equal to "nid RO".

1. lctl mount_perm fsname <-A | -a> [configuration_file]

   Add mount permission configuration gotten from the specified file (if "configuration_file" specified) or standard input (if no "configuration_file" specified) for the specified lustre system ("fsname"). If the "nid" to be

added does not exist in the "fsname" system mount permission configuration, then create a new item , otherwise the new perm is "or" to the old one. If the new perm is not compatible with the old one, then failed for "-A" option, or ignore such configuration for "-a" option. Items with "*" perm will be ignored.

2. lctl mount_perm fsname <-D | -d> [configuration_file]

   Delete mount permission configuration gotten from the specified file (if "configuration_file" specified) or standard input (if no "configuration_file" specified) for the specified lustre system ("fsname"). If the "nid" to be deleted does not exist in the "fsname" system mount permission configuration, or the "perm" does not match the related mount permission configuration, then failed for "-D" option, or ignore such configuration for "-d" option. "*" nid matches any nid, "*" perm matches any permission. "* perm" means delete the specified permission for any client, "nid *" means delete any permission for the specified client; "* *" means delete any permission for any client.

3. lctl mount_perm fsname <-M | -m> [configuration_file]

   Modify mount permission configuration gotten from the specified file (if "configuration_file" specified) or standard input (if no "configuration_file" specified) for the specified lustre system ("fsname"). If the "nid" to be modified does not exist in the "fsname" system mount permission configuration, then failed for "-M" option, or ignore such configuration for "-m" option. Items with "*" perm will be ignored.

4. lctl mount_perm fsname -r [configuration_file]

   Reconfigure the "fsname" system mount permission configuration with the specified "configuration_file" or read from standard input. The old configuration will be overridden. Items with "*" perm will be ignored. Reconfiguration is not equal to delete first and add then, for the later case is not an atomic operation, maybe cause unexpected mount permission denial or approval between the "delete" and "add" operations.

5. lctl mount_perm fsname -q [nid(s) | default | *]

   Query the specified client(s) (nid(s)) mount permission configuration for the specified lustre system ("fsname"); if "default" nid is specified, then query the system default mount permission configuration; if "*" nid or no argument is specified, then list out all the mount permission configuration for the specified lustre system by "fsname".

6. lctl mount_perm -h

   Give usage information.

4 Use cases

SYSTEM1's fsname is lustre1.
SYSTEM2's fsname is lustre2.
CLIENT1's nid is NID1.
CLIENT2's nid is NID2.
CLIENT3's nid is NID3.
SYSTEM1 and SYSTEM2 share the same MGS node. CLIENT1, CLIENT2 and CLIENT3 all belong to the two lustre systems.

1. MGS service starts up without any mount permission configuration.

   SYSTEM1 and SYSTEM2 are both new system, when MGS service starts up, they both use the default mount permission configuration for its system, all clients in the two systems have RW mount permission.

2. Initialize the system mount permission configuration.

   (a) Edit file a.conf as following:

   ```
   NID1 RW
   NID2 RO
   default NA
   ```

   (b) lctl mount_perm lustre1 -r a.conf

   You can input the content of a.conf through standard input line by line with command "lctl mount_perm -r". The similar cases for following "lctl mount_perm" commands.

   Then in SYSTEM1: CLIENT1 has RW mount permission, CLIENT2 has RO mount permission, other clients have NA mount permission; in SYSTEM2, all clients have RW mount permission.

   (c) Restart MGS service

   When MGS service restarts up, they load the mount permission configuration set last time. Then in SYSTEM1: CLIENT1 has RW mount permission, CLIENT2 has RO mount permission, other clients have NA mount permission; in SYSTEM2, all clients have RW mount permission.

   (d) Edit file b.conf as following:

   ```
   NID1 NA
   NID2 RW
   default RO
   ```

   (e) lctl mount_perm lustre2 -r b.conf

   Then in SYSTEM1: CLIENT1 has RW mount permission, CLIENT2 has RO mount permission, other clients have NA mount permission; in SYSTEM2: CLIENT1 has NA mount permission, CLIENT2 has RW mount permission, other clients have RO mount permission.

(f) Restart MGS service

When MGS service restarts up, they load the mount permission configuration set last time. Then in SYSTEM1: CLIENT1 has RW mount permission, CLIENT2 has RO mount permission, other clients have NA mount permission; in SYSTEM2: CLIENT1 has NA mount permission, CLIENT2 has RW mount permission, other clients have RO mount permission.

3. Query specified client(s) mount permission for specified lustre system.

   (a) MGS service starts with the same configuration in test_2.

   (b) lctl mount_perm lustre1 -q NID1

   ```
   NID1 RW
   ```

   (c) lctl mount_perm lustre1 -q NID2 NID3

   ```
   NID2 RO
   NID3 NA
   ```

   (d) lctl mount_perm lustre1 -q default

   ```
   default NA
   ```

   (e) lctl mount_perm lustre2 -q NID1

   ```
   NID1 NA
   ```

   (f) lctl mount_perm lustre2 -q NID2 NID3

   ```
   NID2 RW
   NID3 RO
   ```

   (g) lctl mount_perm lustre2 -q default

   ```
   default RO
   ```

4. List all the mount permission configuration for specified lustre system.

   (a) MGS service starts with the same configuration in test_2.

   (b) lctl mount_perm lustre1 -q *

   ```
   NID1 RW
   NID2 RO
   default NA
   ```

   (c) lctl mount_perm lustre1 -q

   ```
   NID1 RW
   NID2 RO
   default NA
   ```

   (d) lctl mount_perm lustre2 -q

   ```
   NID1 NA
   NID2 RW
   default RO
   ```

5. Reconfigure mount permission for specified lustre system.

   (a) MGS service starts with the same configuration in test_2.

   (b) lctl mount_perm lustre1-q

   ```
   NID1 RW
   NID2 RO
   default NA
   ```

   (c) lctl mount_perm lustre2 -q

   ```
   NID1 NA
   NID2 RW
   default RO
   ```

   (d) Edit file a.conf as following:

   ```
   NID3 RW
   default RO
   ```

   (e) lctl mount_perm lustre1 -r a.conf

   (f) lctl mount_perm lustre1 -q

   ```
   NID3 RW
   default RO
   ```

   Then in SYSTEM1: CLIENT3 has RW mount permission, other clients have RO mount permission.

   (g) Edit file b.conf as following:

   ```
   NID3 RO
   default RW
   * NA
   NID2 *
   ```

   (h) lctl mount_perm lustre1 -r b.conf

   (i) lctl mount_perm lustre1 -q

   ```
   default NA
   ```

   Items with "*" perm are ignored. Then in SYSTEM1: all clients have NA mount permission.

   (j) lctl mount_perm lustre2 -q

   ```
   NID1 NA
   NID2 RW
   default RO
   ```

   Then in SYSTEM2: CLIENT1 has NA mount permission, CLIENT2 has RW mount permission, other clients have RO mount permission, all the mount permission configuration have not been affected by SYSTEM1's reconfiguring.

6. Add specified mount permission configuration for specified lustre system with "-A" option.

   (a) MGS service starts with the same configuration in test_2.

   (b) lctl mount_perm lustre1 -q

   ```
   NID1 RW
   NID2 RO
   default NA
   ```

   (c) lctl mount_perm lustre2 -q

   ```
   NID1 NA
   NID2 RW
   default RO
   ```

   (d) Edit file a.conf as following:

   ```
   NID1 *
   NID2 RW
   NID3 RO
   ```

   (e) lctl mount_perm lustre1-A a.conf

   Items with "*" perm are ignored. The operation should failed for NID2's new perm "RW" is not compatible with the old "RO".

   (f) lctl mount_perm lustre1 -q

   ```
   NID1 RW
   NID2 RO
   default NA
   ```

   The SYSTEM1's mount permission configuration should not be updated.

   (g) Edit file b.conf as following:

   ```
   NID1 *
   NID3 RO
   ```

   (h) lctl mount_perm lustre1-A b.conf

   (i) lctl mount_perm lustre1 -q

   ```
   NID1 RW
   NID2 RW
   NID3 RO
   default NA
   ```

   Items with "*" perm are ignored. Then in SYSTEM1: CLIENT1 and CLIENT2 have RW mount permission, CLIENT3 has RO mount permission, other clients have NA mount permission.

   (j) Edit file c.conf as following:

   ```
   * RW
   ```

(k) lctl mount_perm lustre1 -A c.conf

The operation should failed for NID3's "RO" perm is not compatible with the new "RW".

(l) lctl mount_perm lustre2 -q

```
NID1 NA
NID2 RW
default RO
```

Then in SYSTEM2: CLIENT1 has NA mount permission, CLIENT2 has RW mount permission, other clients have RO mount permission, all the mount permission configuration have not been affected by SYSTEM1's reconfiguring.

7. Add specified mount permission configuration for specified lustre system with "-a" option.

(a) MGS service starts with the same configuration in test_2.

(b) lctl mount_perm lustre1 -q

```
NID1 RW
NID2 RO
default NA
```

(c) lctl mount_perm lustre2 -q

```
NID1 NA
NID2 RW
default RO
```

(d) Edit file a.conf as following:

```
NID1 *
NID2 RW
NID3 RO
```

(e) lctl mount_perm lustre1-a a.conf

(f) lctl mount_perm lustre1 -q

```
NID1 RW
NID2 RO
NID3 RO
default NA
```

Items with "*" perm are ignored. NID2's new perm "RW" is not compatible with the old "RO", so ignored it. Then in SYSTEM1: CLIENT1 has RW mount permission, CLIENT3 and CLIENT2 have RO mount permission, other clients have NA mount permission.

(g) Edit file b.conf as following:

```
* RW
```

(h) lctl mount_perm lustre1 -a b.conf

The operation should succeed, all the incompatible items are ignored.

(i) lctl mount_perm lustre1 -q

```
NID1 RW
NID2 RO
NID3 RO
default NA
```

Then in SYSTEM1: CLIENT1 has RW mount permission, CLIENT3 and CLIENT2 have RO mount permission, other clients have NA mount permission.

(j) lctl mount_perm lustre2 -q

```
NID1 NA
NID2 RW
default RO
```

Then in SYSTEM2: CLIENT1 has NA mount permission, CLIENT2 has RW mount permission, other clients have RO mount permission, all the mount permission configuration have not been affected by SYSTEM1's reconfiguring.

8. Modify specified mount permission configuration for specified lustre system with "-M" option.

(a) MGS service starts with the same configuration in test_2.

(b) lctl mount_perm lustre1 -q

```
NID1 RW
NID2 RO
default NA
```

(c) lctl mount_perm lustre2 -q

```
NID1 NA
NID2 RW
default RO
```

(d) Edit file a.conf as following:

```
NID1 *
NID2 RW
NID3 RO
* RO
```

(e) lctl mount_perm lustre1 -M a.conf

Items with "*" perm are ignored. The operation should failed for NID3's configuration does not exist in SYSTEM1.

(f) lctl mount_perm lustre1 -q

```
NID1 RW
NID2 RO
default NA
```

The SYSTEM1's mount permission configuration should not be updated.

(g) Edit file b.conf as following:

```
NID1 *
NID2 RW
```

(h) lctl mount_perm lustre1 -M b.conf

(i) lctl mount_perm lustre1 -q

```
NID1 RW
NID2 RW
default NA
```

Items with "*" perm are ignored. Then in SYSTEM1: CLIENT1 and CLIENT2 have RW mount permission, other clients have NA mount permission.

(j) Edit file c.conf as following:

```
NID1 *
NID2 NA
* RO
```

(k) lctl mount_perm lustre1 -M c.conf

(l) lctl mount_perm lustre1 -q

```
default RO
```

Items with "*" perm are ignored. Then in SYSTEM1: all clients have RO mount permission.

(m) lctl mount_perm lustre2 -q

```
NID1 NA
NID2 RW
default RO
```

Then in SYSTEM2: CLIENT1 has NA mount permission, CLIENT2 has RW mount permission, other clients have RO mount permission, all the mount permission configuration have not been affected by SYSTEM1's reconfiguring.

9. Modify specified mount permission configuration for specified lustre system with "-m" option.

(a) MGS service starts with the same configuration in test_2.

(b) lctl mount_perm lustre1 -q

```
NID1 RW
NID2 RO
default NA
```

(c) lctl mount_perm lustre2 -q

```
NID1 NA
NID2 RW
default RO
```

(d) Edit file a.conf as following:

```
NID1 *
NID2 RW
NID3 RO
```

(e) lctl mount_perm lustre1 -m a.conf

(f) lctl mount_perm lustre1 -q

```
NID1 RW
NID2 RW
default NA
```

Items with "*" perm are ignored. NID3's configuration does not exist in the old system mount permission configuration, so ignore it. Then in SYSTEM1: CLIENT1 and CLIENT2 have RW mount permission, other clients have NA mount permission.

(g) Edit file b.conf as following:

```
NID1 *
NID2 NA
* RO
```

(h) lctl mount_perm lustre1 -m b.conf

(i) lctl mount_perm lustre1 -q

```
default RO
```

Items with "*" perm are ignored. Then in SYSTEM1: all clients have RO mount permission.

(j) lctl mount_perm lustre2 -q

```
NID1 NA
NID2 RW
default RO
```

Then in SYSTEM2: CLIENT1 has NA mount permission, CLIENT2 has RW mount permission, other clients have RO mount permission, all the mount permission configuration have not been affected by SYSTEM1's reconfiguring.

10. Delete specified mount permission configuration for specified lustre system with "-D" option.

(a) MGS service starts with the same configuration in test_2.

(b) lctl mount_perm lustre1 -q

```
NID1 RW
NID2 RO
default NA
```

(c) lctl mount_perm lustre2 -q

```
NID1 NA
NID2 RW
default RO
```

(d) Edit file a.conf as following:

```
NID1 RW
NID2 RW
NID3 RO
```

(e) lctl mount_perm lustre1 -D a.conf

The operation should failed for NID3's configuration does not exist in SYSTEM1.

(f) lctl mount_perm lustre1 -q

```
NID1 RW
NID2 RO
default NA
```

The SYSTEM1's mount permission configuration should not be updated.

(g) Edit file b.conf as following:

```
NID1 RW
```

(h) lctl mount_perm lustre1 -D b.conf

(i) lctl mount_perm lustre1 -q

```
NID2 RO
default NA
```

Then in SYSTEM1: CLIENT2 has RO mount permission, other clients have NA mount permission.

(j) Edit file c.conf as following:

```
NID2 *
```

(k) lctl mount_perm lustre1 -D c.conf

(l) lctl mount_perm lustre1 -q

```
default NA
```

Then in SYSTEM1: all clients have have NA mount permission.

(m) Edit file d.conf as following:

```
default NA
```

(n) lctl mount_perm lustre1 -D d.conf

(o) lctl mount_perm lustre1 -q

```
default RW
```

Then in SYSTEM1: all clients have have RW mount permission.

(p) Edit file e.conf as following:

```
default RW
```

(q) lctl mount_perm lustre1 -D e.conf

(r) lctl mount_perm lustre1 -q

```
default RW
```

Then in SYSTEM1: all clients have have RW mount permission. That means deleting "default RW" make no effect, for if default item is deleted, the system will generate a new default item automatically.

(s) Edit file f.conf as following:

```
NID1 RW
NID2 RO
NID3 RW
default NA
```

(t) lctl mount_perm lustre1 -r f.conf

(u) lctl mount_perm lustre1 -q

```
NID1 RW
NID2 RO
NID3 RW
default NA
```

(v) Edit file g.conf as following:

```
* RW
```

(w) lctl mount_perm lustre1 -D g.conf

(x) lctl mount_perm lustre1 -q

```
NID2 RO
default NA
```

(y) Edit file h.conf as following:

```
* *
```

(z) lctl mount_perm lustre1 -D h.conf

(aa) lctl mount_perm lustre1 -q

```
default RW
```

Then in SYSTEM1: all clients have have RW mount permission.

(bb) lctl mount_perm lustre2 -q

```
NID1 NA
NID2 RW
default RO
```

Then in SYSTEM2: CLIENT1 has NA mount permission, CLIENT2 has RW mount permission, other clients have RO mount permission, all the mount permission configuration have not been affected by SYSTEM1's reconfiguring.

11. Delete specified mount permission configuration for specified lustre system with "-d" option.

    (a) MGS service starts with the same configuration in test_2.

    (b) lctl mount_perm lustre1 -q

    ```
    NID1 RW
    NID2 RO
    default NA
    ```

    (c) lctl mount_perm lustre2 -q

    ```
    NID1 NA
    NID2 RW
    default RO
    ```

    (d) Edit file a.conf as following:

    ```
    NID1 RW
    NID2 RW
    NID3 RO
    ```

    (e) lctl mount_perm lustre1 -d a.conf

    (f) lctl mount_perm lustre1 -q

    ```
    NID2 RO
    default NA
    ```

    Then in SYSTEM1: CLIENT2 has RO mount permission, other clients have NA mount permission.

    (g) Edit file b.conf as following:

    ```
    NID2 *
    ```

    (h) lctl mount_perm lustre1 -d b.conf

    (i) lctl mount_perm lustre1 -q

    ```
    default NA
    ```

    Then in SYSTEM1: all clients have have NA mount permission.

    (j) Edit file c.conf as following:

    ```
    default NA
    ```

    (k) lctl mount_perm lustre1 -d c.conf

(l) lctl mount_perm lustre1 -q

```
default RW
```

Then in SYSTEM1: all clients have have RW mount permission.

(m) Edit file d.conf as following:

```
default RW
```

(n) lctl mount_perm lustre1 -d d.conf

(o) lctl mount_perm lustre1 -q

```
default RW
```

Then in SYSTEM1: all clients have have RW mount permission. That means deleting "default RW" make no effect, for if default item is deleted, the system will generate a new default item automatically.

(p) Edit file e.conf as following:

```
NID1 RW
NID2 RO
NID3 RW
default NA
```

(q) lctl mount_perm lustre1 -r e.conf

(r) lctl mount_perm lustre1 -q

```
NID1 RW
NID2 RO
NID3 RW
default NA
```

(s) Edit file f.conf as following:

```
* RW
```

(t) lctl mount_perm lustre1 -d f.conf

(u) lctl mount_perm lustre1 -q

```
NID2 RO
default NA
```

(v) Edit file g.conf as following:

```
* *
```

(w) lctl mount_perm lustre1 -d f.conf

(x) lctl mount_perm lustre1 -q

```
default RW
```

Then in SYSTEM1: all clients have have RW mount permission.

(y) lctl mount_perm lustre2 -q

```
NID1 NA
NID2 RW
default RO
```

Then in SYSTEM2: CLIENT1 has NA mount permission, CLIENT2 has RW mount permission, other clients have RO mount permission, all the mount permission configuration have not been affected by SYSTEM1's reconfiguring.

12. Get help information for "lctl mount_perm"

    (a) lctl mount_perm -h
    Detailed usage information should be supplied.

5 Logic specification

5.1 User level mount permission configure utility

"lctl mount_perm" is the user level utility for the system mount permission configuration as described above. It works on MGS node, and communicates with MGS module by ioctl MGS local filesystem mount point.

5.1.1 System mount permission configuration update

System mount permission configuration update operation includes "lctl mount_perm fsname <-A | -a | -D | -d | -M | -m | -r> [configuration_file]" commands. The user level utility parses the specified "configuration_file" or reads from standard input, and converts them into mount permission configuration items. And then it ioctls the MGS local filesystem mount point with the "fsname", the "update" ioctl command and the mount permission configuration items.

For large system, the system mount permission configuration maybe very large, passing all the mount permission configuration items through one ioctl command is not a good idea, we divides them into several continuous "update" ioctl commands.

5.1.2 System mount permission configuration query

System mount permission configuration query operation includes "lctl mount_perm fsname -q <nid(s) | net(s) | default>" commands. The user level utility parses the specified nid, then it ioctls the MGS local filesystem mount point with the "fsname", the "query" ioctl command and the nid. If several nids specified (separated by blank), it will repeat the preceding operations. "default" means query the default mount permission configuration.

5.1.3 System mount permission configuration list

System mount permission configuration list operation includes "lctl mount_perm fsname -q [*]" command. The most cases are the count of "fsname" system mount permission configuration items is not large and can be output through one "list" ioctl command. Otherwise, we have to get all the configuration items through several continuous "list" ioctl commands. What we should resolve is how to make the MGS module know which the next configuration item should be output for next "list" ioctl command. The answer is that MGS module will return the offset with the next mount permission configuration item nid for each "list" ioctl command, when the user level utility "list" next, it will use such offset to indicate MGS module where to start. For the first "list", such offset is zero. If all the "fsname" system mount permission configuration items have been output, the MGS module set such offset as zero to notify the user level utility no need next "list".

5.2 Client-side logic specification

The mount permission configuration is managed by MGS, the mount permission is authenticated by MDS (MDT), nothing to be done for client.

5.3 MDS kernel level logic specification

Each MDS maintains a mount permission configuration table in memory hashed with client's nid. When MDS (MDT service) starts up, it fetches system mount permission configuration from MGS to initialize such table, then it writes the configuration to the configuration file related with its fsname on its disk partition. The configuration file on MDS disk partition will be used to initialize MDS mount permission configuration table when MDS (MDT service) starts up with failed communication with MGS next time.

5.3.1 Mount permission authenticate

When client connects to MDS (when mount, recovery, replay, and so on), MDS searches its mount permission in the system mount permission configuration table, and compares with its mount flags, to decide whether such connection should be permitted or not.

If client wants to change its mount flags after connecting, e.g. change RO mount to RW mount, MDS will authenticate its mount permission first, if not approved, such changing operation is denied.

5.3.2 Mount permission configuration table update

MDS should update its mount permission configuration table when MGS updates the system mount permission configuration. To make sure MDS can follow MGS to update its mount permission configuration table, MDS need to hold the LDLM lock ("LCK_CR" type) attached to the system mount permission

configuration on MGS. When MGS updates the system mount permission configuration, it revokes such lock, then MDS needs to acquire such lock again, and re-fetch the mount permission configuration to reinitialize its mount permission configuration table. The similar processing scheme as other system configuration update between MGC-MGS.

One issue to be considered that if the system mount permission configuration is so large as to have to be fetched back through several continuous "fetch" RPC, then we meet the same issue with user level utility for "list" command: how to make the MGS module know which the next configuration item should be packed back to MDS? the same solution of "offset" with user level utility for "list" command case.

MDS updating its mount permission configuration maybe take several RPCs time. If when it is in update, and some client's connection need to be authenticated at the same time, how to process the authentication? wait for the update finished or just authenticate based on the half updated mount permission configuration table? Both are not. MDS will update a temporary mount permission configuration table first, when all the update finished, it replaces the working mount permission configuration table with the temporary one. Before such replacing, all the connection authentication will based on the working mount permission configuration table (the old one). That means the system mount permission configuration update is an asynchronous operation, at sometime, different MDSes in the same lustre system maybe use different mount permission configuration, but they will be updated to concordant status in a very limited period.

After updating the system mount permission configuration table, MDS writes the updated configuration to the configuration file related with its fsname on its disk partition. If writing back failed, no need to recover the configuration table, for this configuration file is only used when MDS (MDT service) starts up with failed communication with MGS next time. Such inconsistent configuration file maybe overridden by next update.

5.4 MGS kernel level logic specification

MGS module is in charge of kernel level mount permission configuration management. For each lustre system managed by it, it maintains a separated system mount permission configuration file on its disk partition, named "fsname.mpc". If no such file for a lustre system, then MGS uses the default mount permission configuration (RW mount permission) for it. If necessary (e.g. when communicates with user level mount permission configuration utility "lctl mount_perm"), MGS will establish the system mount permission configuration table in memory based on the on-disk mount permission configuration file. It has the same struct with the system mount permission configuration table on MDS. When user level mount permission configure utility ("lctl mount_perm") updates the system mount permission configuration, MGS writes the configuration back to the on-disk mount permission configuration file "fsname.mpc".

5.4.1 Transaction operation for system mount permission configuration update

As described above, for system mount permission configuration update operation, the configuration items maybe so many as to can not to be passed into MGS module through one "update" ioctl command, we divides them into several continuous "update" ioctl commands for the large update operation. Process the "update" operation as following:

1. the user level mount permission configure utility ("lctl mount_perm") opens the MGS local filesystem mount point, and obtains related lock;
2. the user level utility parses the specified configuration file (or reads from standard input) into configuration items;
3. the user level utility ioctls the opened the MGS local filesystem mount point with "fsname", "update" ioctl command and the configuration items;
4. MGS module processes the user level utility "update" ioctl commands based on a temporary mount permission configuration table for each "update" ioctl command;
5. repeats 3 & 4, until all the "update" ioctl commands of this update operation succeed;
6. MGS module acquires the LDLM lock ("LCK_EX" type) related with the system mount permission configuration, which revokes the LDLM lock held by MDSes to notify them the system mount permission configuration need to be updated;
7. MGS module writes the new configuration to the configuration file related with the "fsname" on its disk partition;
8. MGS module replaces the working mount permission configuration table with the temporary one, and releases the old one;
9. the user level mount permission configure utility releases the related lock, closes the MGS local filesystem mount point, and the update transaction operation finished.

5.4.2 The system mount permission configuration table concurrently access control

If we allow multi-update the system mount permission configuration concurrently, there maybe many temporary mount permission configuration tables. It is hidden trouble for the exhausting memory attack. On the other hand, if one is in the "list" system mount permission configuration, but another updated the system mount permission configuration, then the offset for next "list" ioctl command maybe invalid. To prevent such cases, we introduce a

kernel rw_semaphore for each mount permission configuration table. Such rw_semaphore is used for not only "update" operation, but also "list" operation.

For "update" operation, the updater must obtain such rw_semaphore with "write" permission before its first "update" ioctl, and hold such rw_semaphore until all the "update" are finished or it closes (in "release" method) the MGS local filesystem mount point when exit for some failure. For the later case, we should cleanup the system (including freeing the temporary table) before release such rw_semaphore.

For "list" operation, the lister must obtain such rw_semaphore with "read" permission before its first "list" ioctl, and hold such rw_semaphore until all the "list" are finished or it closes (in "release" method) the MGS local filesystem mount point when exit for some failure.

6 State management

6.1 Scalability & performance

- When system starts up, each MDS needs to fetch the system mount permission configuration from MGS, it will cause several RPCs (for the LDLM lock and the system mount permission configuration). It maybe make some effect on system starts up time, which depends on the system MDS count and mount permission configuration items count.

- For each connection between client and MDS, the mount permission authentication should be done. It is only query on hash table, so very little effect on the system performance.

- For each mount permission configuration update, there are several RPCs for each MDS. But such update is an asynchronous process by the specified thread, very limited effect on the system scalability and performance.

6.2 Recovery changes

- If MDS (MDT service) crash, when it restarts up, it fetches the system mount permission configuration from MGS to initialize its mount permission configuration table. All the replayed connection from client will be authenticated again.

- If MDS can not communicate with MGS (maybe MGS crash or network fail) when starts up, it loads the configuration from local configuration file on its disk partition to initialize its mount permission configuration table.

- If MGS crash, when it restarts up, all the MDSes refetch the system mount permission configuration from MGS to reinitialize its mount permission configuration table. It has no effect related with mount permission to client.

- If some failure cause client lost connection to MDS, it will try to reconnection to MDS, and MDS will authenticate it again. If the mount permission configuration for such client has been changed after its successful mount, then when it try to reconnect to the MDS, it maybe get unexpected permission deny.

6.3 Locking changes

- A spin_lock for each system mount permission configuration table is introduced to control the non-blocked concurrently access ("query") from user level mount permission configuration utility ("lctl mount_perm"), client mount permission authentication, and the exchange between working table and updating table.

- A rw_semaphore for each system mount permission configuration table is introduced to control the blocked concurrently access ("update" and "list") from user level mount permission configuration utility ("lctl mount_perm").

- A LDLM lock for each system mount permission configuration table is introduced to control MDS(es) update its system mount permission configuration table from MGS.

6.4 Disk format changes

For each lustre system, MGS creates a system mount permission configuration file named "fsname.mpc" under its MOUNT_CONFIGS_DIR. It is the "nid" and "perm" sequence as following:

```
<{nid}{perm}>[{nid}{perm}...]
```

MDS also creates a system mount permission configuration file named "fsname.mpc" under its MOUNT_CONFIGS_DIR. It has the same file format with MGS's disk configuration file.

The mount permission configuration file(s) on MGS disk partition is (are) the master one(s). The mount permission configuration file MDS disk partition is the copy for the master one.

6.5 Protocol changes

Introduce a new RPC proc between MGC (for MDS) module and MGS module named "MGS_MPC_GET" for MDS to fetch the system mount permission configuration from MGS.

7 Alternatives

1. Communication between "lctl mount_perm" utility and MGS module.

   There maybe other candidates, e.g. proc interface on MGS node. But how to pass the specified nid to MGS module for query (lctl mount_perm -q <nid(s) | *>) is an issue to be considered. For proc interface, it can "list" all to user level, then "grep" the specified nid, but is seems not a good idea. Thus the local filesystem mount point ioctl on MGS node maybe a good choice for that.

2. The system mount permission configuration table concurrently access control.

   We use kernel rw_semaphore for that, seem flock can perform the same thing. Why not select the later? For some error cases, when the current flock is released, next flock for system mount permission configuration update can be granted before the related system cleanup finished.

3. Why not choose MGS module for mount permission authentication?

   MGS module mainly manages configuration information. The functional work should be done by other modules. It makes the responsibility more clear. One MGS can manage many lustre systems concurrently, mount permission authentication on MGS maybe cause it overloaded.

   On the other hand, when MDS (MDT service) starts up with failed communication with MGS, it can use the local configuration file on its own disk partition to initialize its system mount permission configuration table, and then the mount permission authentication can be done. If chooses MGS for that, then mount permission authentication can not be done under such cases.

4. Why use a new LDLM lock for system mount permission configuration, why not use the same one with other system configuration?

   System mount permission configuration is security related information, it only exists on MGS and MDS, does not permit to be accessed by client. But other non-security related system configuration can be accessed be any lustre node. So the system mount permission configuration is separated into new configuration file from other system mount permission configuration. If all of them share the same LDLM lock, then when some configuration updated (e.g. "lctl mount_perm fsname -a", or new OST registers), it maybe cause unnecessary LDLM lock revocation.

5. Why introduce a new RPC proc "MGS_MPC_GET"? why not use the lustre existing LLOG processing for that?

   Mount permission configuration just uses very simple "{nid}{perm}" sequence on disk and network, and seems no need to build some complex struct for using LLOG. But we maybe consider to replace this work with

Yury's "LLOG OSD" when he finish such task in future. It is only related with network transfer and read/write on-disk configuration file.

6. Why not support to specify mount permission configuration file when mkfs?

   The master mount permission configuration file is on MGS disk partition, so such configuration file specified only for MGS mkfs. Mount permission configuration is related with "fsname", and one MGS can manage many lustre system concurrently, but only one "fsname" can be specified when MGS mkfs, thus support specifying mount permission configuration file when mkfs has little meaning.

   In fact, mount permission can be configured before the lustre system created, that means whenever the MGS service is running, you can configure the system mount permission even if the specified lustre system ("fsname") is not created yet. For such case, when the new lustre system created, it can use the existing mount permission configuration directly.

8 Focus for inspections

1. When the mount permission configuration update should take effect? mainly focus on degraded update, e.g. RW=>RO, RO=>NA, RW=>NA

   (a) at once, just when the configuration update.

   (b) next reconnection, replay or recovery.

   (c) new connection, next mount.

   In this design, we choose the first one. After the mount permission configuration updating, all the connection approved will be checked, the connections with degraded mount permission will be evicted immediately. That means operations based on such connections maybe get unexpected I/O error.