TEXAS TECH UNIVERSITY

Whamcloud

Lustre Dynamic Nodemap

Lustre User Group 2024

sbuisson@whamcloud.com

ddn

Lustre Dynamic Nodemap

Whamcloud

- How does legacy nodemap work exactly?
- The need for dynamic nodemaps
- Dynamic nodemap design
- Other features that could leverage dynamic nodemaps

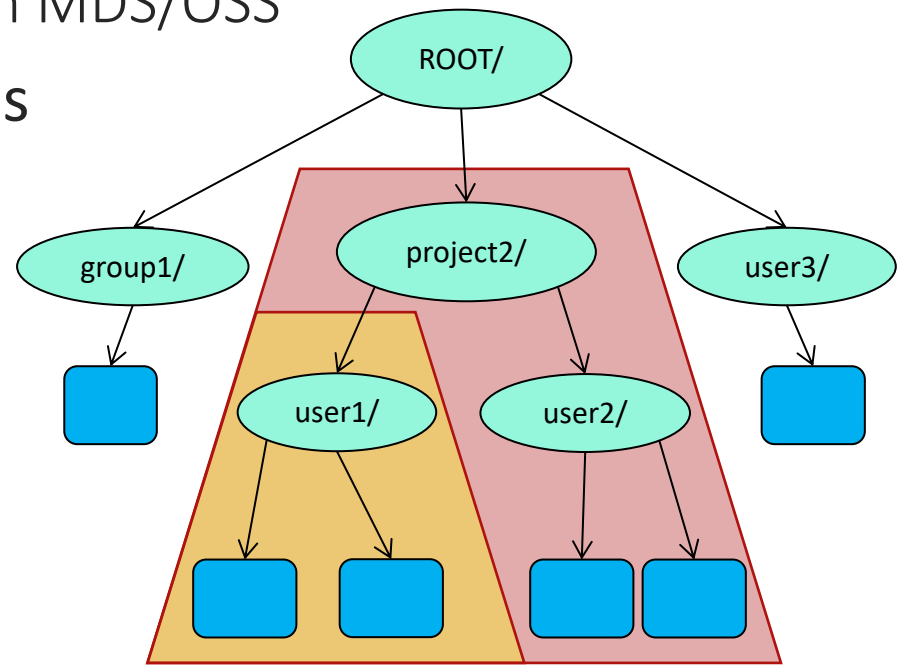
How Does Legacy Nodemap Work Exactly?

Whamcloud

- nodemap:
  - a policy group consisting of **one or more NID ranges**
  - several **properties**, such as trusted and admin, defining access conditions
  - a collection of **idmaps** determining how UIDs/GIDs/PROJIDs on the client are translated into the canonical file system UIDs/GIDs/PROJIDs
- When a client connects, servers decide which nodemap its NID belongs to
- Lustre clients are unaware of nodemaps
- Nodemaps are defined on the MGS
  - Stored on disk in nodemap dedicated llogs
  - Definitions pushed to all Lustre servers when updated

The Need for Dynamic Nodemaps

- Nodemap was initially designed with static configurations in mind
  - Define groups of clients (e.g. remote campus)
  - Assign properties, UID/GID maps, run in production for a long time
- Legacy nodemap is a heavyweight mechanism
  - Needs to create/update on MGS, then wait for sync on MDS/OSS
- Legacy nodemap is not design for frequent updates
  - Because llogs are not made for this
    - Consume space in the config log
    - Slower to process at mount time
- Makes virtualization for sub-tenants complex
  - Need to configure nodemaps as each job is run
  - Contain only nodes in specific job



Dynamic Nodemap Design LU-17431

Whamcloud

- Dynamic nodemaps can be created/updated directly on MDS/OSS
  - In-memory only, non persistent configuration
    - No llogs involved
  - Take effect immediately
  - Require some external orchestration to ensure consistency across all MDS/OSS
- Dynamic nodemaps are hierarchical
  - Inherit settings from parent if any, or *default* nodemap like legacy nodemaps
    - avoid the need to set every parameter for the new nodemap
    - safety in case of server restart, fallback to parent nodemap
  - Hierarchy based on NID ranges
    - NID ranges associated with dynamic nodemaps can be included in parent NID ranges
    - NID ranges cannot overlap with other dynamic nodemaps, just like static nodemap ranges

Dynamic Nodemap Design – Sub-nodemaps

Whamcloud

New fields in data structures

```
struct lu_nodemap {
        …
        /* is a dynamic nodemap */
        bool            nm_dyn;
        /* list of sub-nodemaps */
        struct list_head        nm_subnodemaps;
        /* list for parent nodemap */
        struct list_head        nm_subnm_list;
        /* link to parent nodemap */
        struct lu_nodemap      *nm_parent_nm;
};
```

- `nm_subnodemaps` and `nm_subnm_list` to ensure hierarchy consistency
- `nm_parent_nm` to easily refer to parent nodemap.

Dynamic Nodemap Design – Overlapping Ranges

Whamcloud

New fields in data structures

```
struct lu_nid_range {
        …
        /* sub ranges included in this NID range */
        struct nodemap_range_tree rn_subtree;
};
```

- `rn_subtree` to store sub-NID ranges
  - avoid lengthy initial nodemap find if there are many nodemap ranges
  - sub-nodemaps are searched only if parent is a match.

Dynamic Nodemap Design - User Interface

Whamcloud

- User interface on MDS/OSS
  - Same lctl commands, with some additional options
    - `lctl nodemap_add -d dyn1 --parent persistent1`
    - `lctl nodemap_modify --name dyn1 --property squash_projid --value 99`
    - `lctl nodemap_add_idmap --name dyn1 --idtype uid --idmap 500:60001`
    - `lctl nodemap_add_range --name dyn1 --range 192.168.56.[205-208]@tcp0`
      - `dyn1` is a dynamic sub-nodemap of `persistent1`
      - NID overlap allowed only if `192.168.56.[205-208]@tcp0` is a subset of `persistent1` ranges
    - `lctl nodemap_del dyn1`
  - On MDS/OSS, only dynamic nodemaps can be created/updated.

Dynamic Nodemap Design – Open Questions

Whamcloud

- Limit hierarchy depth?
  - Having sub-nodemaps/sub-NID ranges does not add much overhead
- Only allow **fewer** privileges for sub-nodemap vs. parent?
  - Would make sense for properties such as *admin* or *trusted*
- Nodemap property to block mounts matching persistent parent nodemap?
  - Not inherited from parent nodemap by dynamic nodemap
  - Catch case where sub-nodemap is temporarily missing after restart
  - Client will retry, dynamic nodemap should be configured at that point

Other Features That Could Leverage (Dynamic) Nodemaps

Whamcloud

- LU-17217 "Allow server to control/deny client connections"

  Servers should be able to selectively allow client connections based on policies defined by Lustre admins. Policies could be defined on a wide range of client properties, depending on what proves to be most useful. - Tim Day, Amazon

  - Add new roles to the nodemap's `rbac` property?
- LU-17410 "Add per-nodemap capabilities mask"
  - Finer-grained control for privileged userspace processes on nodes
  - Useful for nodes that do backup, restore, HSM or tiered data mover
- LU-11077 "Client-specific tunable parameter configuration"
  - Add *FSNAME*`-client-`*NODEMAP* config llog for special clients (e.g. CPU vs. GPU)
  - Clients request *FSNAME*`-client-nodemap` config from server on each mount
  - MGS replaces literal `-nodemap` string with actual client *-NODEMAP* name (if any)

TEXAS TECH UNIVERSITY

Whamcloud

Thank you!

sbuisson@whamcloud.com

ddn