Layout Enhancement

John L. Hammond
High Performance Data Division

Agenda

- Design for replication
- Determine requirements for Crush
- Extend layout locking proto to allow changes while file is accessed (replication)
- Determine client side changes to request layout changes from within IO stack
- Determine protocol changes to support large layouts
- Discuss interoperability
- Address rolling upgrades

Design Layout Enhancements to support

Support for:

- Data on MDT
- Replication
- Online migration
- Very wide striping
- The future

Design only, no code to be delivered.

Data on MDT

- Need new placement hint
  - Use 1 bit from lmm_pattern
  - LOV_PATTERN_F_DATA_ON_MDT
- Need migration policy
  - Big files automagically kicked off MDT
  - Not clear this should go in layout
  - SEP!

Replication Container Layout

Define a new layout header

```
struct lov_md_repl_v1
        __u32 lrm_magic;                /* LRM_MAGIC_V1*/
        __u32 lrm_flags;
        __u32 lrm_size;
        __u16 lrm_entry_count;
        __u16 lrm_layout_gen;
        union {
        __u64 lrm_padding;
        } u;
```

Followed by entries of non-replicated layouts

Replication Container Layout (2)

```
struct lov_repl_md_entry_v1 {
        __u32 lrme_id;                  /* unique identifier of entry */
        __u32 lrme_flags;                /* LRME_FL_STALE
                                  * LRME_FL_OFFLINE */
        struct lu_extent lrme_extent;   /* file extent for entry */
        __u32 lrme_offset;               /* offset of entry data */
        __u32 lrme_size;                /* size of entry data */
        union {
        __u64 lrme_padding;
        } u;
};
```

Replication Container Layout Comments

Make no assumptions about containees!

… except, no recursion.

Treat migration as a case of temporary partial replication

Interaction of version and extent will be a headache – must handle/avoid splitting

Other headaches?

Very Wide Striping

How many OSTs do we assume?

64K, other.

How many stripes do we assume?

2000, 64K, #OST.

lmm_v1 + 2000s < 48K (12pp)

lmm_v1 + 64Ks > 1.5M > LNET_MTU

Very Wide Striping (2)

Can **mitigate** storage and network overhead using compression

- zlib compression ratios 1.8:1 to 2.4:1

- Any depends on #OSTs, #stripes, SEQ_MDT0

- Example (assume 64K OSTs, MDT0)

  lmm_v1 + 2000s < 48K (12pp)

  zlib.deflate(lmm_v1 + 2000s) < 17K

Very Wide Striping (3)

Compact Striping Layout

from Xyratex* Architectural Priorities WP

- Pack a single FID for objects
- Bitmap (+ ...) for OST indices
- Pro: lov_compact_md + 64Ks < 9K
- Con: must derive individual object FIDs from OST index and single FID

Old client compatibility

Options for replication:

- Return error
- Fake v1 for old clients
- Client that does not understand containee will treat like unavailable replica
- Has implications for writability

References

Xyratex* Lustre* Architecture Priorities Overview

-

Thank You

john.hammond@intel.com