HLD

Mikhail Pershin

February 6, 2008

1 Requirements

Implement object management like precreation, create-on-write and destroy in new uOSS obdfilter code.

2 Functional specification

2.1 General approach

Full FID implementation requires new client with FID allocation on it and FID sequence management on both client and uOSS code. Though the current design is based on FID usage there is need for temporary approach to support FID-based functionality with old object identification through group/id.

Full mode: the client generates FID for OST objects, they are stored in LOV EA at MDS and these FIDs are used in OSS as object identificator, object are created in FID/ directory. CROW (create_on_write) can be used to create objects instead of precreate.

Compatible mode: the MDS takes object id as usual from pool of the precreated IDs. OSS pack object id into IDIF to use it instead of FID in new OSS code. CROW is used to creat objects but last_id is supported too for MDS needs.

Current design document is about the ***compatible mode*** only.

2.2 IDIF

For compatibility with FID-based API and future use the IDIF is defined. IDIF is a FID used to identify pre-fid object on an OST (for the purpose of a future fid-on-the-ost implementation).

```
IDIF: 0:32bits, 1:1bit, ost-index:15bits, obj-index:48bits, 0:32bits
```

48 bits for obj-index are enough to sustain 10K/sec creations for 5 years, so we shouldn't have problems upgrading any OST.

Object ID is packed to IDIF in OST during request unpacking and stored in `oti` structure

2.3 Precreation of object IDs

MDS uses precreated pool of object IDs to get new one and OSC sends regular requests for new IDs. We keep this intact but OSS doesn't create objects actually. It maintain the last_id value instead and creates objects on write. This allows to use old MDS behaviour and introduce CROW functionality that will be needed in near future.

2.4 Create on write

CROW is object creation mechanism in OSS now. Obsoleted precreation code and orphan handling code is not needed. CROW code creates objects during first write/setattr/punch/truncate and also it should hanlde read/getattr to don't return errors but simulate empty object instead.

2.5 Object destroy

Object destroy is implemented in general but should be completed fully by comparision with HEAD code.

2.6 Compatiblity with future approaches

Current implementation is temporary until FID for OSS objects will be really generated on client and used. There will be no API changed needed and code modifications, but OSD will need code to distinguish the IDIF from real FID and object index will be needed to find object by its FID. Both functionality don't require attention in current design and can be designed separately later.

3 Use cases

3.1 Precreate cases

- *filter_create() -> filter_precreate() -> filter_set_last_id()*
  Filter set new last_id value during both precreate and recreate of objects

3.2 Create on write cases

- *filter_setattr()/filter_preprw_write()/filter_truncate()/filter_punch() -> filter_object_find_or_create()*
  The new object should be created during first write operation.
- *filter_preprw_read()/filter_getattr()*
  These methods can found that object is not yet created, but should simulate empty object.

3.3 Destroy

- *filter_destroy()*
  Destroy the object

3.4 Orphans handling

- *filter_create([flags OBD_FL_DELORPHAN]) -> filter_destroy_precreated()*
  MDS require orphans to be destroyed.

4 Logic specification

4.1 Last_id management

OSS implementation requires support for last_id and cleanups in precreation code.

`Last_id:`

- is written transactionally during precreate/recreate call from MDS
- used to store last precreated(reserved) value
- used to check the object id is valid (not greater than last_id)

Precreation code is quite simple and needs no more recreation code. Only last_id handling is needed.

4.2 CROW

Create-on-write is used in filter_preprw_write(), filter_setattr() and filter_punch/truncate() to create new object if it doesn't exist yet and set proper attributes.

CROW functionality should care about correct handling of read/getattr for objects which are not yet created. To do that it simulate 'empty' object with zero size and time.

Client should handle zero time properly too.

Note: current CROW implementation should be considered as CROW prototype. The full CROW design is out of scope this document and will be done by Lustre Protocol Team later.

5 State management

5.1 State invariants

Object on OSS lifecycle:

1. not yet created but valid
2. created and valid
3. destroyed and invalid

5.2 Scalability & performance

Using CROW technique will improve performance

5.3 Recovery changes

Recovery of OSS: the last_id is taken from disk and corrected from MDS last_id value during object recreate. Orphan destroy call from MDS is processed as previously - all objects with IDs greater than last_id from MDS should be destroyed.

Recovery of MDS is not changed