High level design of truncate without extent lock on the client.

February 11, 2008

1 Requirements

Lustre client (both llite and liblustre) should be able to perform file truncate without obtaining an extent lock on [new-size, EOF]. This is deemed to be an important optimization, especially due to implicit truncates done by open(file, O_TRUNC).

2 Functionality specification

Do not take extent DLM lock on client during truncate. Instead, lock is taken by the OST.

3 Use cases

- open(path, O_TRUNC): calls

  ll_setattr_raw()->vmtruncate()->ll_truncate()->obd_punch() -> ..... network .... -> ost_punch() [take DLM lock here] -> filter_truncate() -> ...

- truncate(path, newsize) the same as open(path, O_TRUNC)

- ftruncate(path, newsize) the same as open(path, O_TRUNC)

4 Logic specification

Remove grabbing of DLM lock from truncate path on the client, add it to the server.

5 State management

DLM lock protects file data and i_size. We do not care about data integrity because

- during shrinking truncate data are thrown away anyway;
- during expanding truncate no data are covered by [new-size, EOF] lock;
- as DLM lock is per-node rather than per-thread, no new races against concurrent threads on the same client appear;
- races with other nodes are "legal".

Protection of i_size. Not clear. ll_truncate() compares inode->i_size (which equals newsize at that point) with KMS. Without exent lock, KMS can be completely out of date.

6 Protocol, API's, Disk format

Compatibility bit should be used to detect when clients can use OST-side locking for truncate. OBD_CONNECT_SRVLOCK can be reused for this.

7 Scalability and performance

Scalability should improve due to fewer locks. Performance should improve due to reduced traffic.

8 Recovery.

No visible changes.

9 Alternatives

Aren't known.