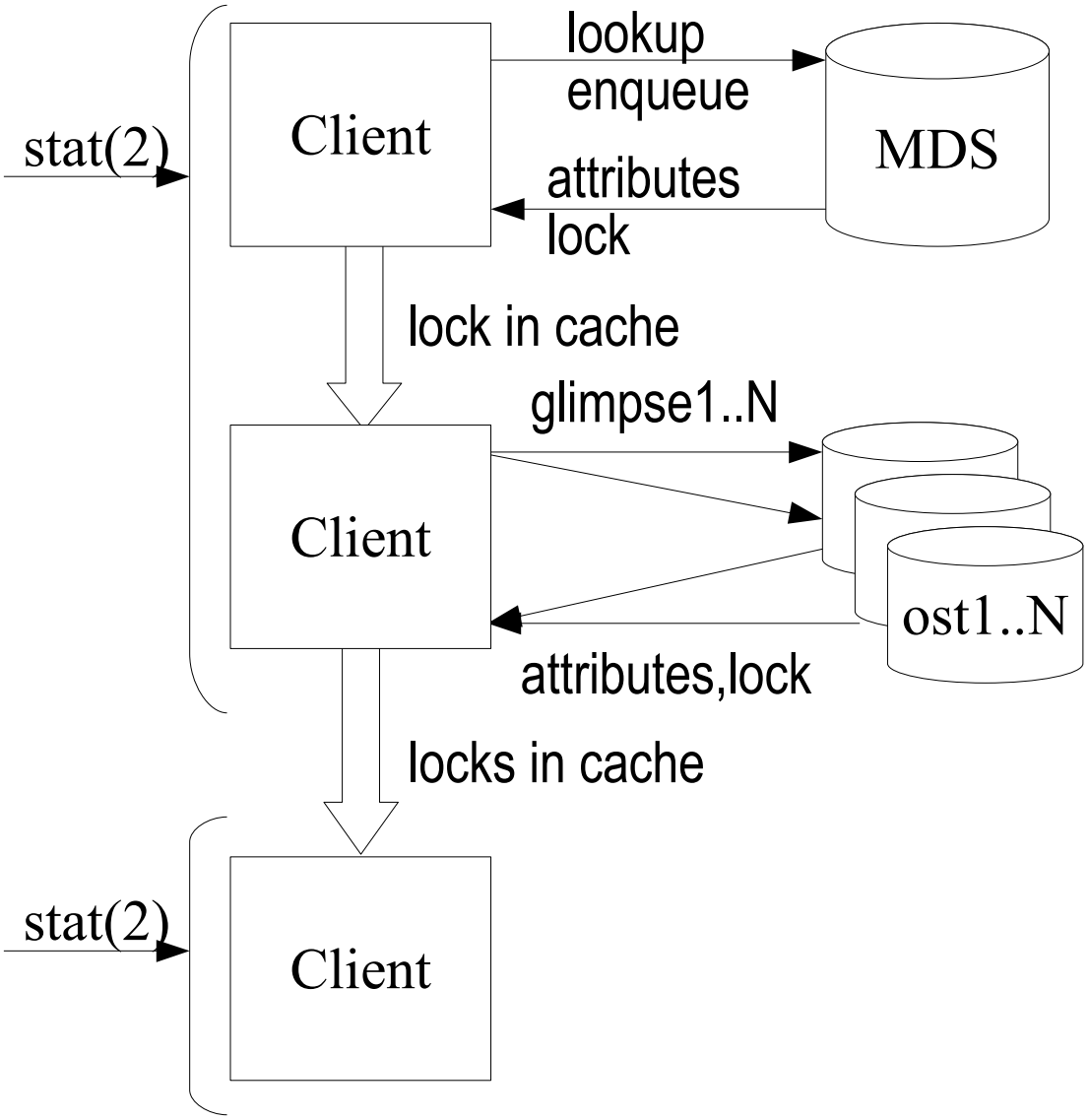
Size on MDS (SOM)

Vitaly Fertman
Lustre Group

Contents

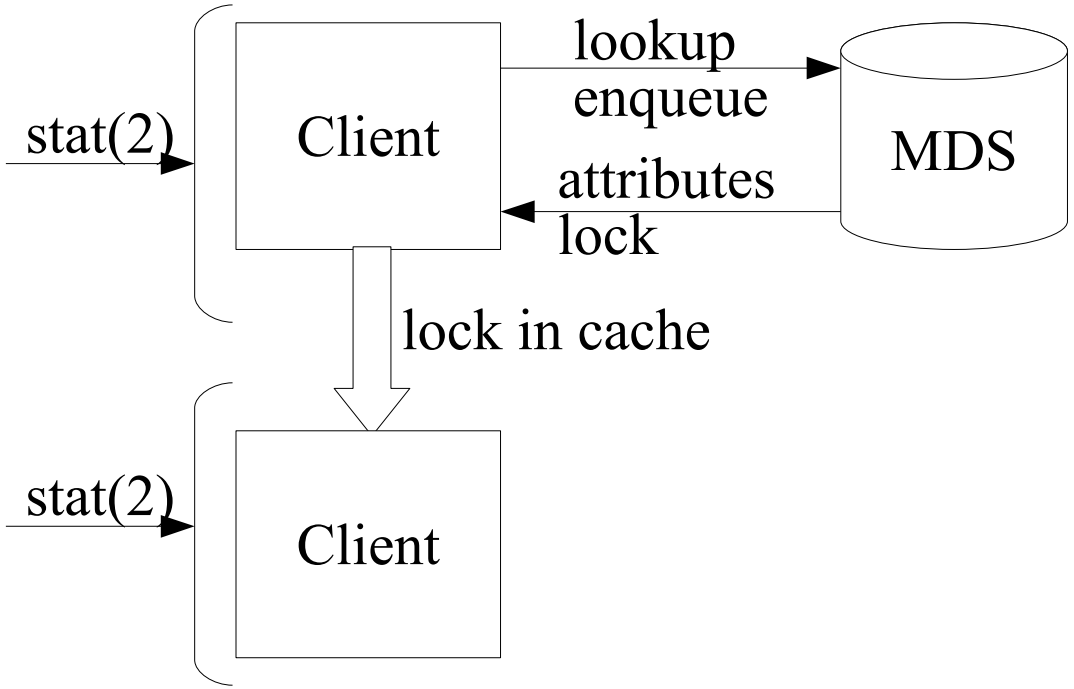
- Current approach: size on OST
- New approach: size on MDS
  - Cache validity control
  - Interoperability
  - Recovery
- Performance
- Future optimizations

Current approach: size on OST



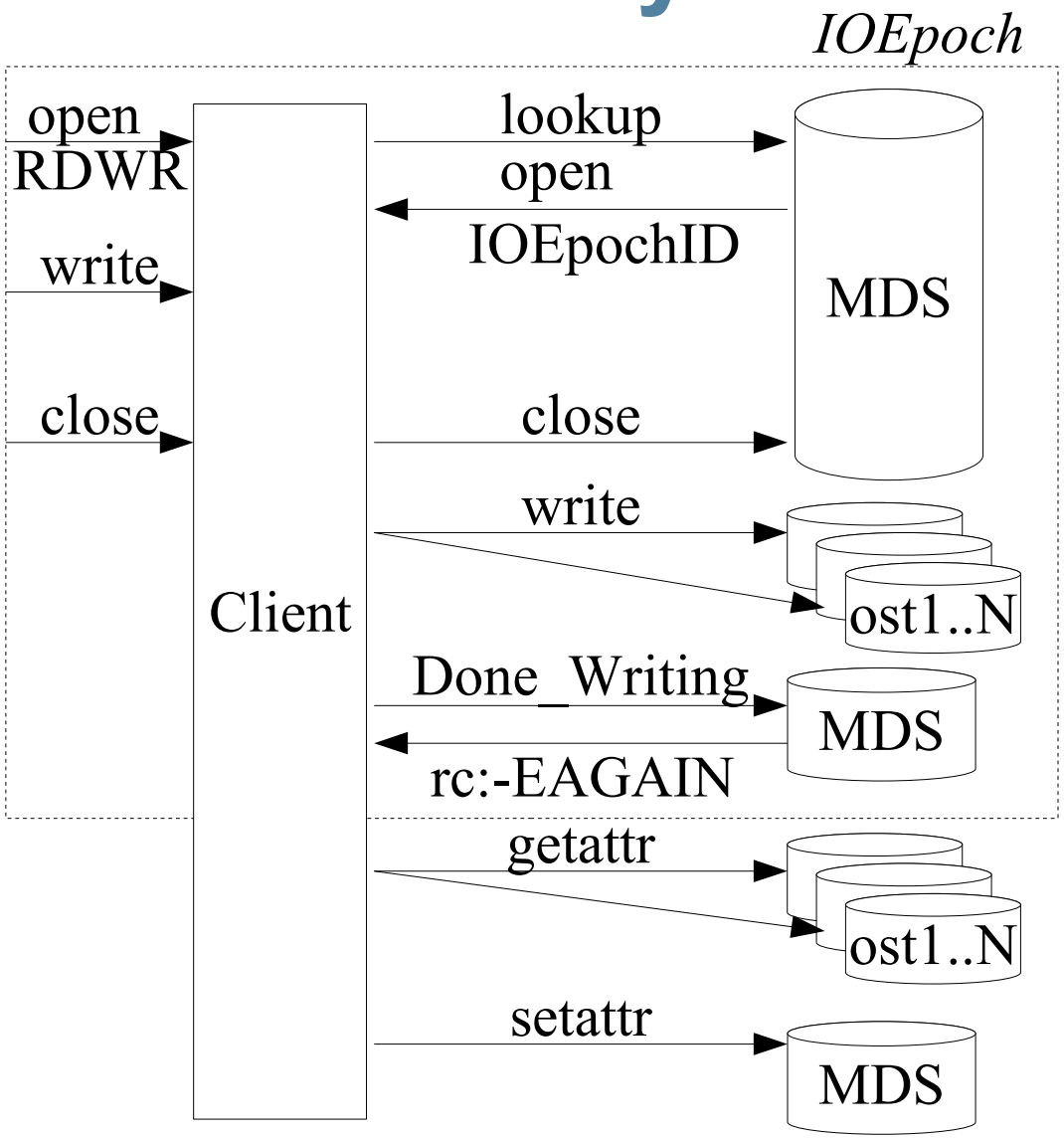
- MDS keeps file metadata, it has an authority for some inode attributes
- File body is kept in object stripe, some attributes (e.g. size) are encoded there
- stripe_count + 1 RPCs to get inode attributes
- result: ls -l is slow on large directories
- client caches attributes under locks

New approach: size on MDS



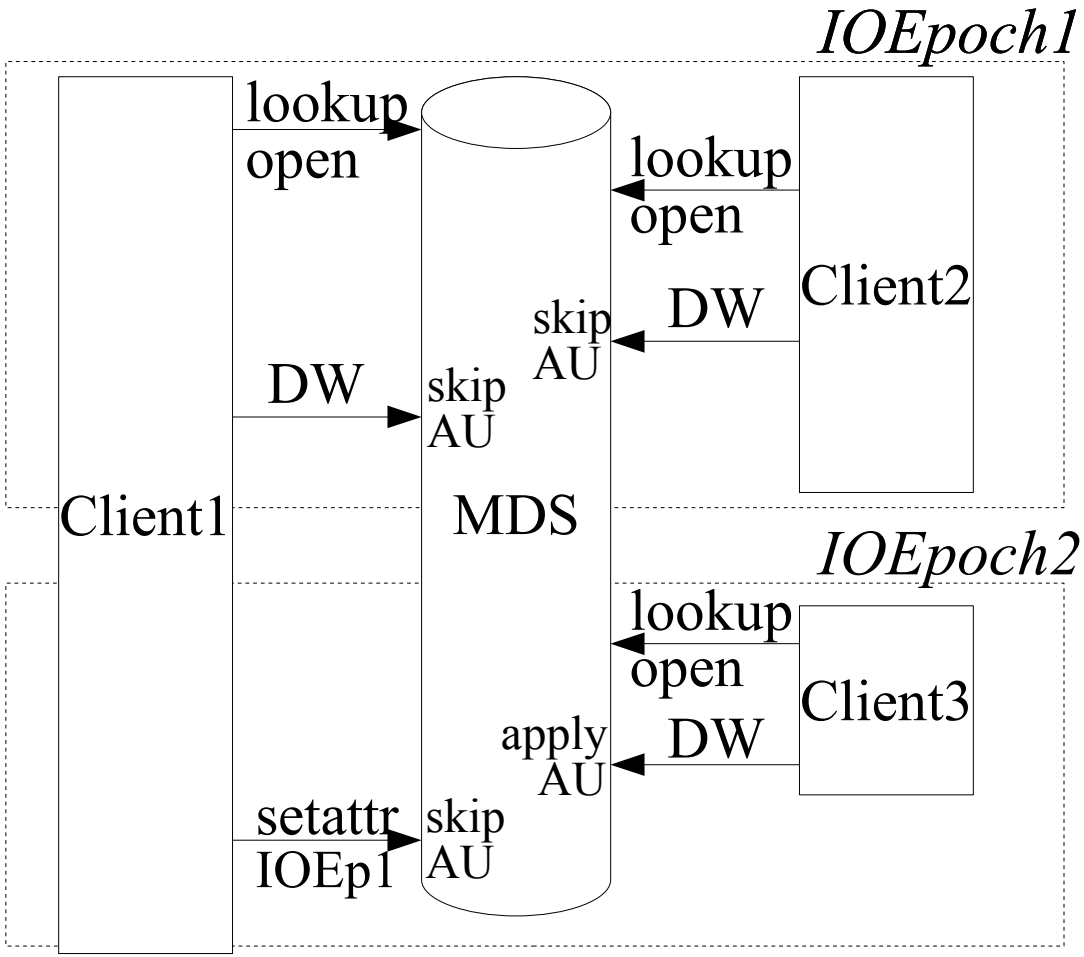
- MDS caches correct OST attributes (file size, blocks, ctime, mtime)
- 1 rpc to get inode attributes vs. stripe_count +1
- client caches attributes under lock

Cache validity control:IOEpoch



- File in IOEpoch is active for IO,size may change
- SOM is not valid, client gets attributes in old way
- Locks given to clients are cancelled on IOEpoch start.
- Done_Writing: no dirty cache on client
- EAGAIN: revalidate MDS cache.
- Attribute Update: cache is revalidated, SOM is valid.

Cache validity control: shared IOEpoch



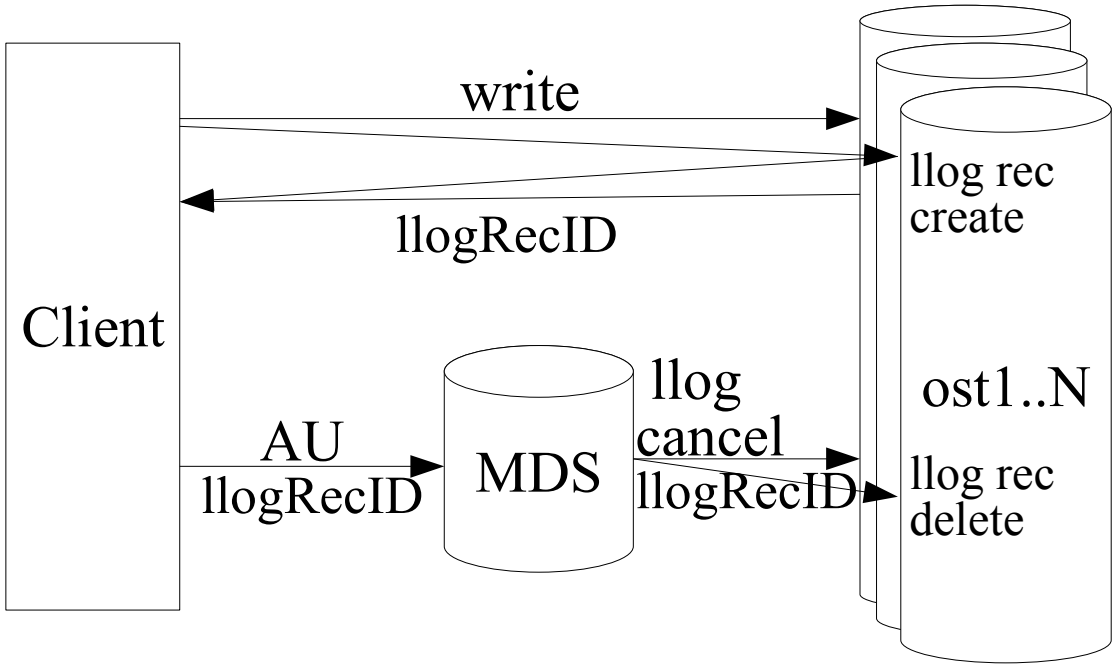
- Clients share IOEpoch for parallel IO
- MDS do not trust Attribute Updates for shared IOEpochs
- MDS returns -EAGAIN to the last client only.
- Setattr for old IOEpoch is skipped.

Interoperability

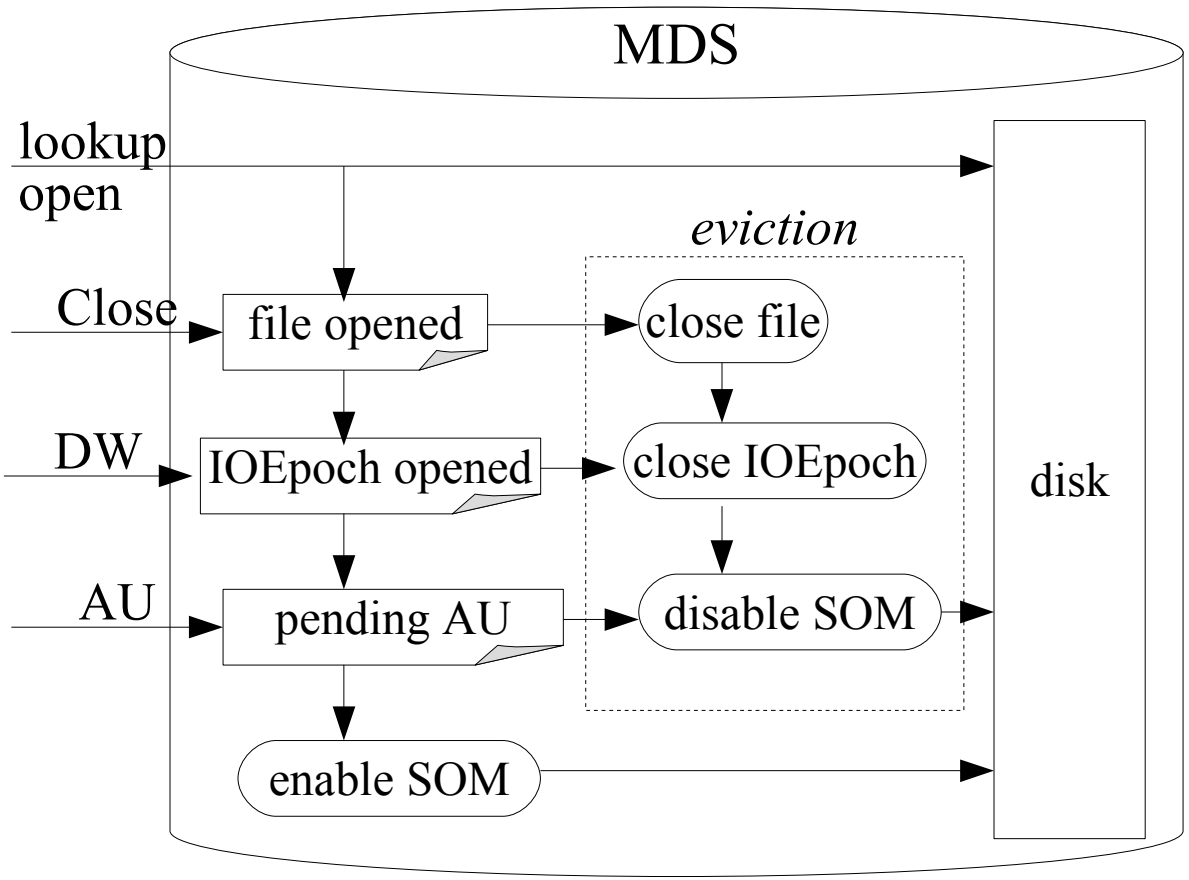
- SOM is for Lustre 2.0
- All the cluster to be upgraded to get SOM
- Upgrade
  - The node upgrade order is arbitrary
  - The last step: enable SOM through MDS failover
  - Only nodes with SOM support can connect to MDS
- Downgrade:
  - The first step: disable SOM through MDS failover
  - The node downgrade order is arbitrary
  - Limitations: MDS since Lustre 1.8.x

Recovery

- All the nodes in the cluster are involved
- Each node may fail
- Connection between some may fail
- LLOG on OST: keep a track of changes



Recovery: client eviction



- Eviction: close files, close IOEpochs: disable SOM
- SOM is disabled until new IOEpoch starts
- AU for new IOEpoch re-enables SOM

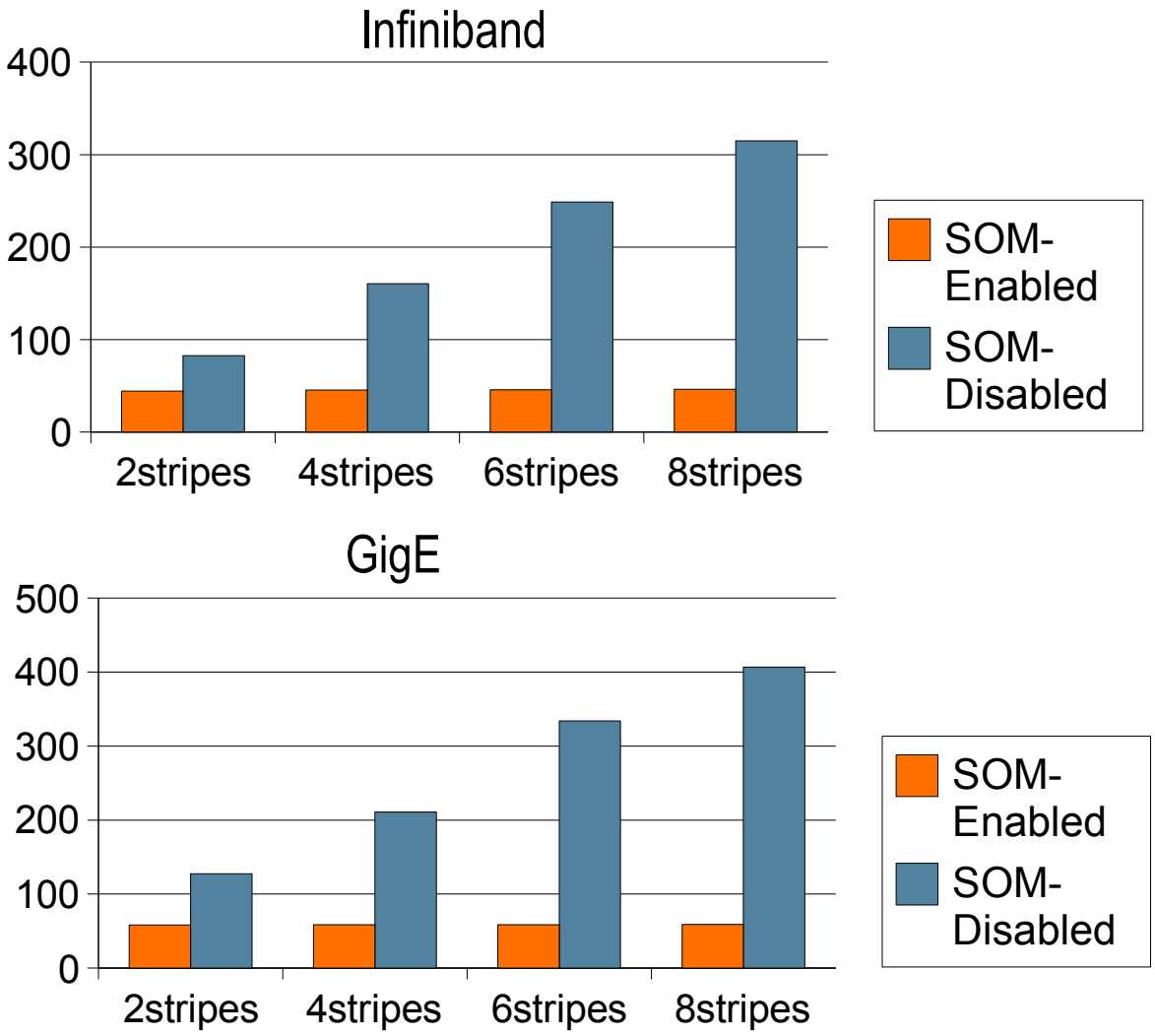
Recovery: MDS failover

- Client recovers the inode state only (file opened, IOEpoch opened) on replay.
- MDS relies on synchronization with OST:
  - MDS reads OST llogs
  - MDS marks inodes as SOM-disabled
- SOM is re-enabled within next IOEpoch lifecycle.

Recovery: OST failover

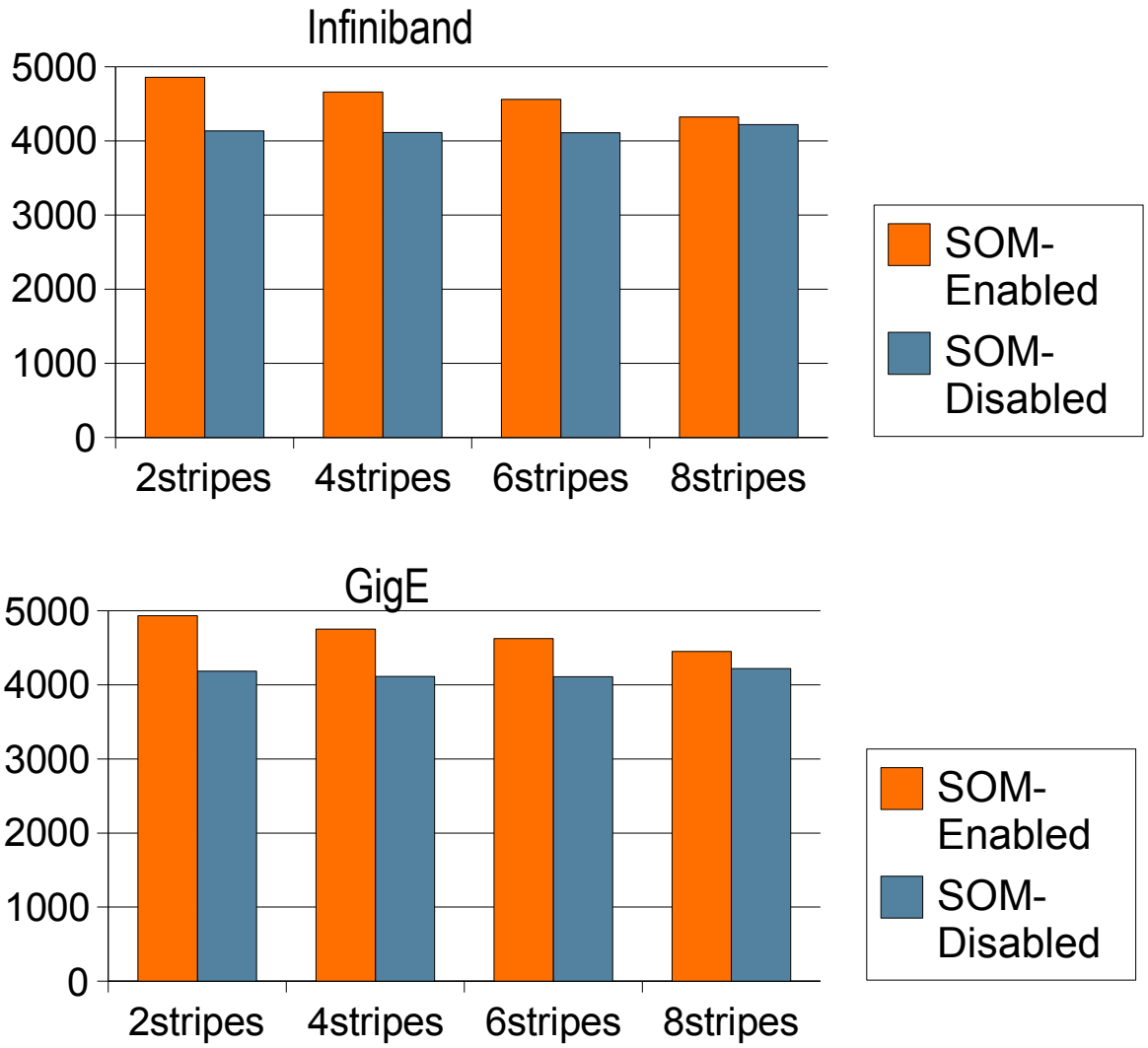
- Client forwards an absent OST error to MDS in Attribute Update
  - MDS marks inode as SOM-disabled
- SOM is disabled if any OST in stripe is not active;
- MDS synchronizes with OST on OST failover:
  - reads OST llogs
  - marks inodes as SOM-disabled

Performance:ls-l



- 100K files in one directory;
- ls -l is run in parallel from 4 clients;
- Speedup: 2stripes x2 8stripes x7
- results are very similar for both networks;

Performance:write



- 100K files in one directory;
- overwrite files sequencially in parallel from 4 clients;
- Slowdown: 2stripes 17%, 8stripes 3%
- results are very similar for both networks;

Future optimizations

- OPENCACHE
  - avoid DoneWriting rpc.
- Exclusive IOEpoch holder
  - avoid GETATTR to OST and SETATTR to MDS.
- Getattr from MDS on recovery
  - avoid SOM disabled on inodes for a long time

Vitaly.Fertman@sun.com