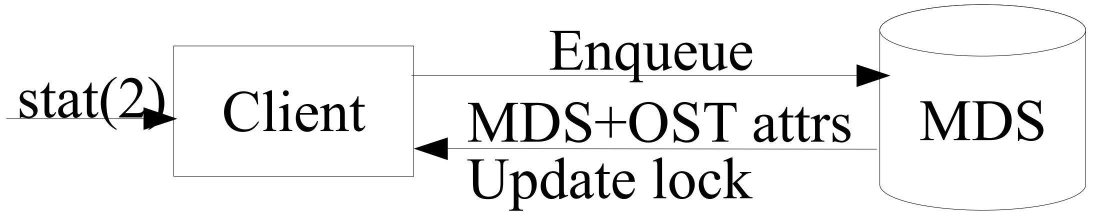
Size On MDS (SOM)

Vitaly Fertman
2009-11-10

1. Introduction and terminology.



Cache OST attributes on MDS

IOEpoch (IOEp) – period of time when an IO can be initiated for an inode through interaction with OST somewhere in the cluster.

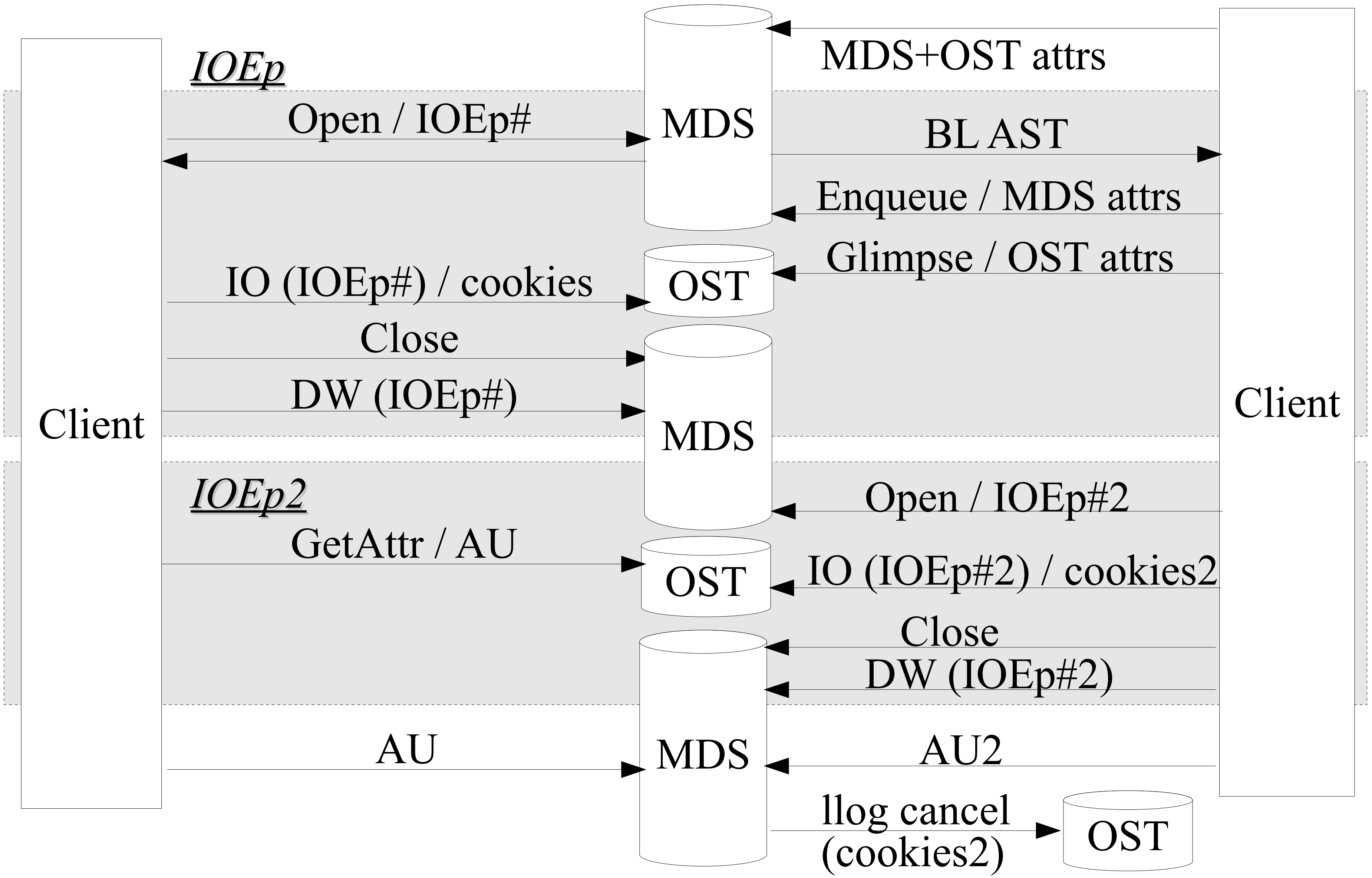
DONE_WRITING (DW) – a special RPC sent by client notifying MDS the client closes IOEp.

IOEp Holder – is a client which has IOEp opened.

SOM Attribute Update (AU) – a special RPC which contains all the needed info to rebuild SOM cache on MDS.

SOM LLOG record – a llog record created on OST as a flag that SOM cache on MDS is not valid and needs to be updated

2.1 IOEpoch life-cycle.



2.2 IOEpoch life-cycle. Optimizations.

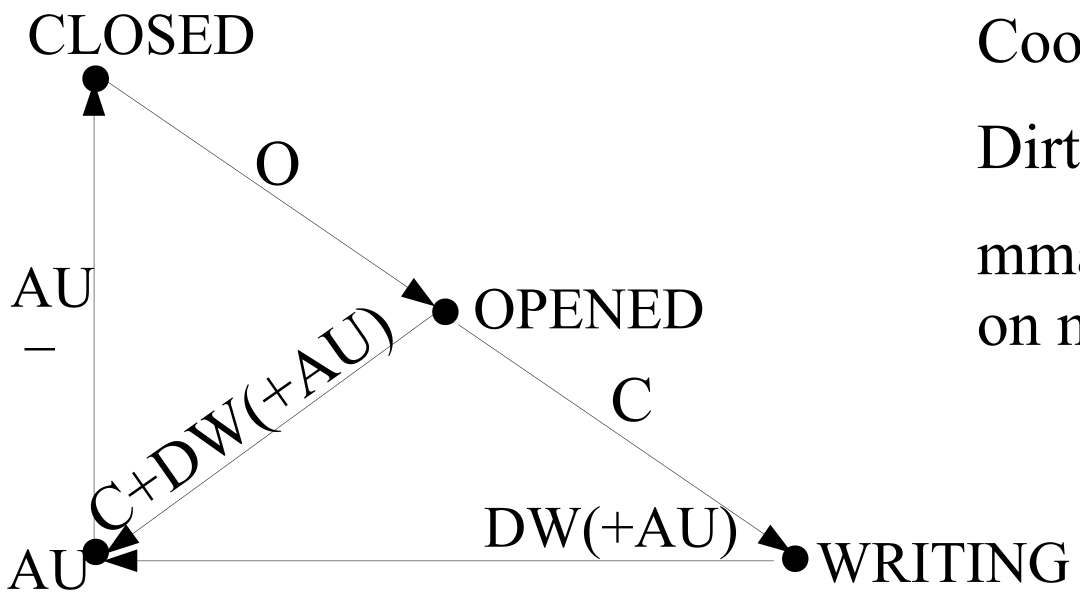
Goal: no extra RPC for at least exclusive IOEp holder

- DW can piggyback on CLOSE
- AU can piggyback on DW, which can piggyback on CLOSE
    - Client has all EOF locks;
    - Client is exclusive IOEp holder (it hopes so);
    - MDS believes only to exclusive IOEp holder AU (on CLOSE,DW)
- SOM_CHANGE flag informs MDS file is changed;
- OPENLOCK cache, better chances DW will piggyback on CLOSE
- Enqueue gets llog cookies from OST
- Postpone DW for truncates (idea from Andreas)

3. OST object state graph.

- •2 states: Quiescent and Changed
- •IO: Q->C:
    - •llog record is created as a part of the same transaction;
    - •llog cookie is assigned to inode;
    - •llog cookie->inode hash is populated for quick inode search;
- •OST replies cookies when asked (Write, Punch, GetAttr);
- •llog_cancel or object desotry: C->Q
- •1 llog record per file per IOEp: llog record has IOEp#
- •MDS failover: llog record has MDS object FID
- •OST failover: llog record has OST object ID&GR

4. Client IOEp holder state graph.



Cookies are assigned to IOEp

Dirty pages list on inode

mmap: pages get to dirty list on munmap

Truncate: no WRITING state

Utime (set-in-past): no WRITING state

1 IOEp holder in WRITING state at a time

O: open   C: close DW: done_writing AU: attribute update

5. MDS IOEp holder state graph.

$$\begin{tikzcd}
\text{SOM-ENABLED} \arrow[rd, "O"'] & & \\
& \text{OPENED} \arrow[lu, bend right, "C+DW(+AU)"'] \arrow[ld, "C+DW"'] \arrow[rd, "C"'] & \\
\text{AU} \arrow[uu, "AU"] & & \text{WRITING} \arrow[ll, "DW"'] \arrow[lluu, bend right=60, "DW(+AU)"']
\end{tikzcd}$$

[C,DW]->SOM-ENABLED:
AU is valid or
no change or
not the last IOEp holder

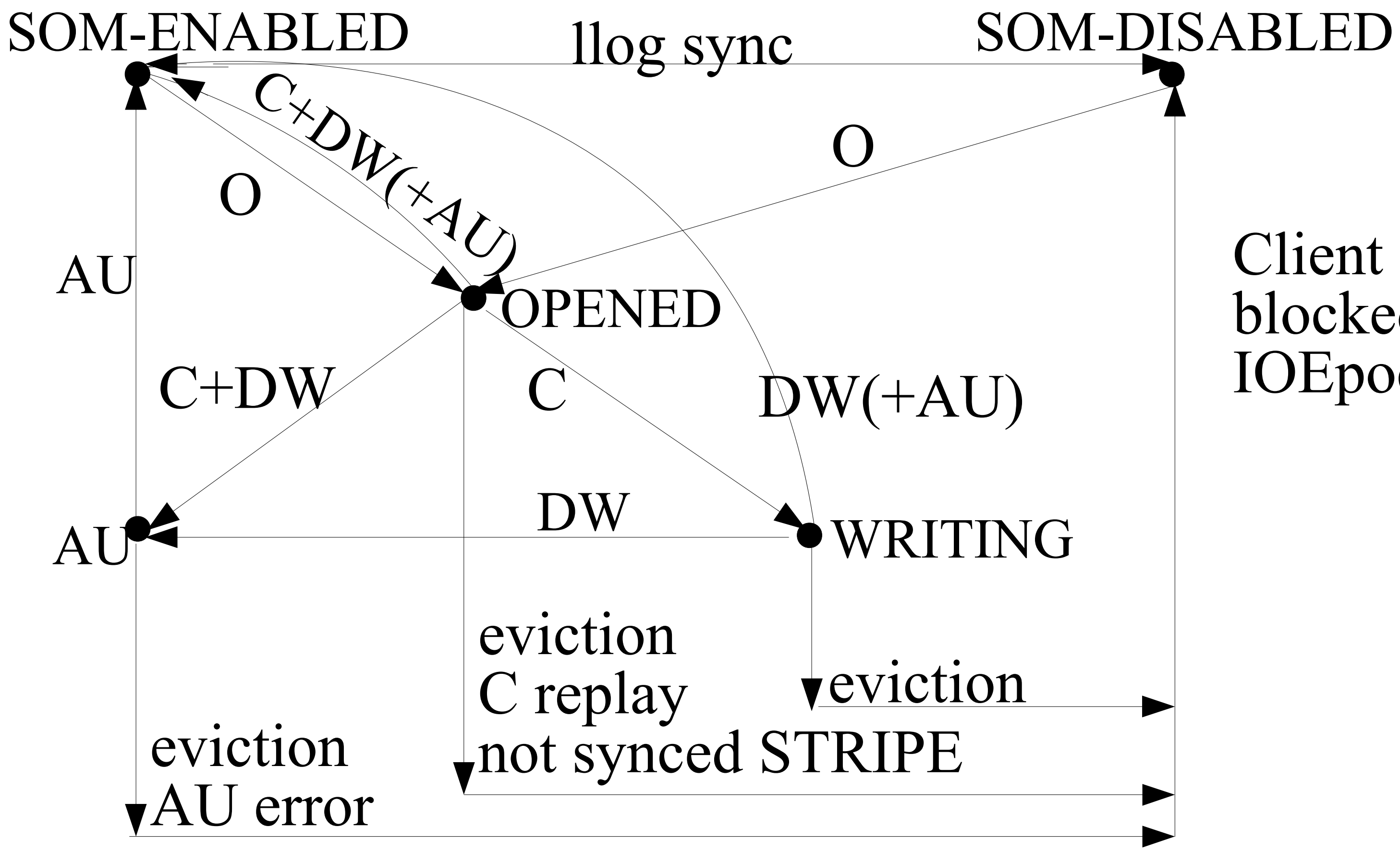
AU (on C, DW) is valid for Exclusive IOEp holder only

Exclusive IOEp holder detection: CHANGED flag

1 IOEp holder in AU state per IOEp

O: open C: close DW: done_writing AU: attribute update

6.1. MDS IOEp holder state graph. Recovery.



Client ops are not blocked by OST ops: IOEpoch window

O: open C: close DW: done_writing AU: attribute update

6.2. Recovery problems.

Client eviction or client disappeared on MDS failover.

Client is still alive and generate IO, MDS has no control over it:
•existent data cache under extent locks;
•lockless IO (write, truncate);
•new enqueue and new data cache under new locks;

IO could be different stage:
•new syscall; RPC in-flight; new RPC creation;
•RPC in resend list; data under extent lock;

Goal: block any IO from lost client since SOM cache is rebuilt.

Later eviction detection by client.

6.3. Recovery solutions. MDS failover.

MDS tells OST the hard limit IOEp on MDS-OST sync

OST lets IO happen of not older IOEp

OST marks extent PW locks as “IO-allowed” on MDS-OST sync

OST lets locked IO happen of older IOEp if it is “IO-allowed”

OST returns error to clients for not allowed IO

SOM is disabled on file until MDS-OST sync is done

OST lets all the IO happen until MDS-OST sync is done

Client get new generated IOEp on OPEN replay

Client drops IO

- a further optimization could be done

6.4. Recovery solutions. Client eviction.

MDS knows the files involved

1st solution: store IOEp# in EA on OST on next AU from OST

2nd solution (Alex): MDS asks MGS to evict the client from all the OST
•SOM is disabled for these files until MGS reports success;
•client needs to detect MDS eviction before OST one;
•client drops old IO once OST eviction is detected;
•client blocks new IO until files are re-opened;
Problems:
•Client eviction from MGS is not ready;
•File re-opening, including truncates, is not ready;
•IO blocking is not ready (probably solved through LOVEA);
•4th node interaction is involved (MGS, OST failures);

6.5. Recovery solutions. Client eviction. Cont'd.

3rd solution: disable SOM for problem files temporary:

MDS tells OST largest unused IOEp from time to time

OST blocks IO of unused IOEp

Evicted client does IO until OST is told about some larger IOEp limit

MDS does not need to remember the list of files not evicted clients

MDS does not need to ensure all the OST are connected

MDS needs to detect SOM is disabled for a file on next OPEN

•disable SOM for a particular IOEp# on disk;

Requirements:

•Do not close IOEp on CLOSE (no DW) if not all the OST are synced

7.1 Interoperability.

MDS downgrade loses all the SOM cache

MointID region [start;end] and mointID in LMA
•optimization: get rid of mountID in LMA

MDS downgrade
•OST zeros SOM llog on MDS-OST sync
•OST creates no llog record anymore, applies IO with no limitations

MDS downgrades first: SOM incompatible mount option on OST

•OST drops in on MDS connection once detects no SOM support

Client downgrade: cannot connect to SOM-enabled servers

MDS upgrade before OST, client is able to connect to both
•no valid cache on MDS;
•MDS cannot connect to OST, SOM is disabled for involved files;
•no SOM cache on MDS – no need in llog on OST;

MDS upgrade (continued)

7.2 Interoperability. MDS upgrade.

Many use cases due to multiple upgrades & downgrades
•file is opened on SOM-enabled lustre before downgrade&upgrade
•file is opened on SOM-disabled lustre before upgrade
•file is opened again after replay on SOM-enabled lustre
•The same file could be opened in all these ways;

Goal to minimize it to a common use case, preferably including recovery use cases.

1st solution: Files are always re-opened with new generated IOEps;
2nd solution: Files are always re-opened with NO IOEp;

•file cannot be opened with & w/out IOEp at the same time;
•state graph on MDS is simple;
•interoperability does not introduce new problems