Problems with old client IO path

• Old code with a lot of obscurities;
• Inter-layer assumptions: lov, llite;
• Based on a huge obd-interface;
• Not easily expandable;
• Traditionally many bugs;
• High overhead: allocations, kms;
• Not very portable;
• Different from md server stack;
• Interaction between layers is not clear;
Client IO path clean-up requirements

- Reduce number of bugs in the IO path.
- Build an infrastructure for planned features.
- Clean up old code and interfaces.
- Specify precisely
  > state machines
  > Interactions
  > pre-, post-conditions and invariants.
- Don't lose performance.
CLIO design goals

- clear layering;
- controlled state sharing;
- simplified layer interface;
- real stacking;
- support for:
  > SNS;
  > read-only p2p caching;
  > lock-less IO and ost intents;
- reuse of mdt server layering;
- improved portability.
CLIO restrictions

- Not a complete re-write: only interface cleanup
- No new features, only new interfaces
- Only data IO path. Meta-data are left intact
- No changes to the DLM
- No changes to the recovery
- No changes to the RPC formation logic
Layers

- **vvp**: VFS and VM, converting system calls (read, write) into client api. POSIX operation semantics (short reads, flock, O_APPEND), read-ahead. This is mostly old llite.
- **lov**: raid0 striping: files/stripes, locks/sublocks
- **sns**: raid with parity declustering. raid-frame library (not implemented)
- **osc**: ptlrpc/inet, DLM, RPC formation
sys_read(fd, buf, count);
sys_write(fd, buf, count);
fstat(fd, &statbuf);

Layers

user space

vm-vfs-posix (vvp, linux)

file:

pos
pos + count
read-ahead

liblustre
solaris

libclient (ccc)

lov

queue
Fundamental data-types

- **cl_object** a file and a stripe, based on lu_object (fids, hashing, lru), caches pages, and tracks locks, can be composed of other objects.

- **cl_page** fixed-size chunk of file data, can be part of multiple cl_object's: file, and stripe of this file; uniquely identified by (object, index).

- **cl_lock** a region of a particular object, covered by a cluster lock.

- **cl_io** IO context. Contains IO state. Can be stopped, resumed. Collects locks and owns pages.

- **cl_req** RPC
Layered objects

- Compound object: a header and a sequence of layers
- Layer-private state
- Per-layer operation vectors
- Generic code invokes operations on each layer, delegating behavior

\[ \text{header} \rightarrow \text{slice} \rightarrow \text{slice} \rightarrow \text{slice} \]

\[ \text{lev1 data} \rightarrow \text{op0} = \text{lev1 op0} \rightarrow \text{op1} \]

\[ \text{lev2 data} \rightarrow \text{op0} = \text{lev2 op0} \rightarrow \text{op1} \]

\[ \text{lev3 data} \rightarrow \text{op0} = \text{lev3 op0} \rightarrow \text{op1} \]
**cl_page**

- cached file data
- different sizes of pages (DMU blocks)
- interaction with MM/VM (page locking, memory pressure)
- simple locking model, suitable for Lustre
cl_page state machine
**cl_lock**

- extent read/write lock on file data
- top-lock on a file
- sub-locks on stripes
- based on DLM
cl_lock state machine

- top-lock and sub-locks (N:M);
- caching of top-locks (short IO);
- non-blocking, asynchronous state-machine (parallel IO);
**cl_object**

- top-object: file
- sub-object: stripe
- uniquely identified by fid
- cached
- LRU
- tree of pages
- list of locks
- trivial state machine

![Diagram of cl_object structure](attachment:image.png)
Files, locks, pages.

Diagram of file system structures:
- **cl_page**:
  - llite
  - lov
- **cl_object**:
  - llite
  - lov
- **file extent lock**:
  - llite
  - lov
- **tree of pages**
- **file**
- **list of locks**
- **stripe object**
- **stripe sub-lock**
- **list of locks**
- **struct inode**
- **struct address_space**
- **struct ldlm_lock**
- **pending io queue**

Diagram showing relationships between file system components.
**cl_io**

- high level IO
- context for IO activity
- fast: no dynamic allocation
- state machine
- designed for “parallel IO”
cl_io state machine

- **init**: initialize layers
- **prep**: form iteration
- **lock**: collect locks
- **start**: do IO
- **end**: wait completion
Use cases: file read

- sys_read(fd, buf, count);
- vvp: cl_io_init(); cl_io_prep():
  > lov_io_pre(): clip IO to one stripe
- cl_io_lock():
  > vvp_io_lock(): lock [pos, pos+count-1]
  > sns_io_lock(): lock parity block
  > generic: sort locks, enqueue
- cl_io_start():
  > vvp_io_start(): generic_file_read()
    - vvp_readpage(): read-ahead, cl_submit_io():
      - sns_submit_io(): parity ordering
      - osc_submit_io(): send RPC
Use cases: lock cancellation

- osc lock gets cancel
- find object
- find subtree of pages
- call cancel on each page, up to llite
- much simpler than existing logic

network cancel request
Clio, Greek: ????, /'klaiou/, n: the muse of history also known as the Proclaimer. The name is from the root ????, meaning recount or make famous.

Nikita.Danilov@sun.com