



CLIO TOI

Nikita Danilov

Senior Staff Engineer

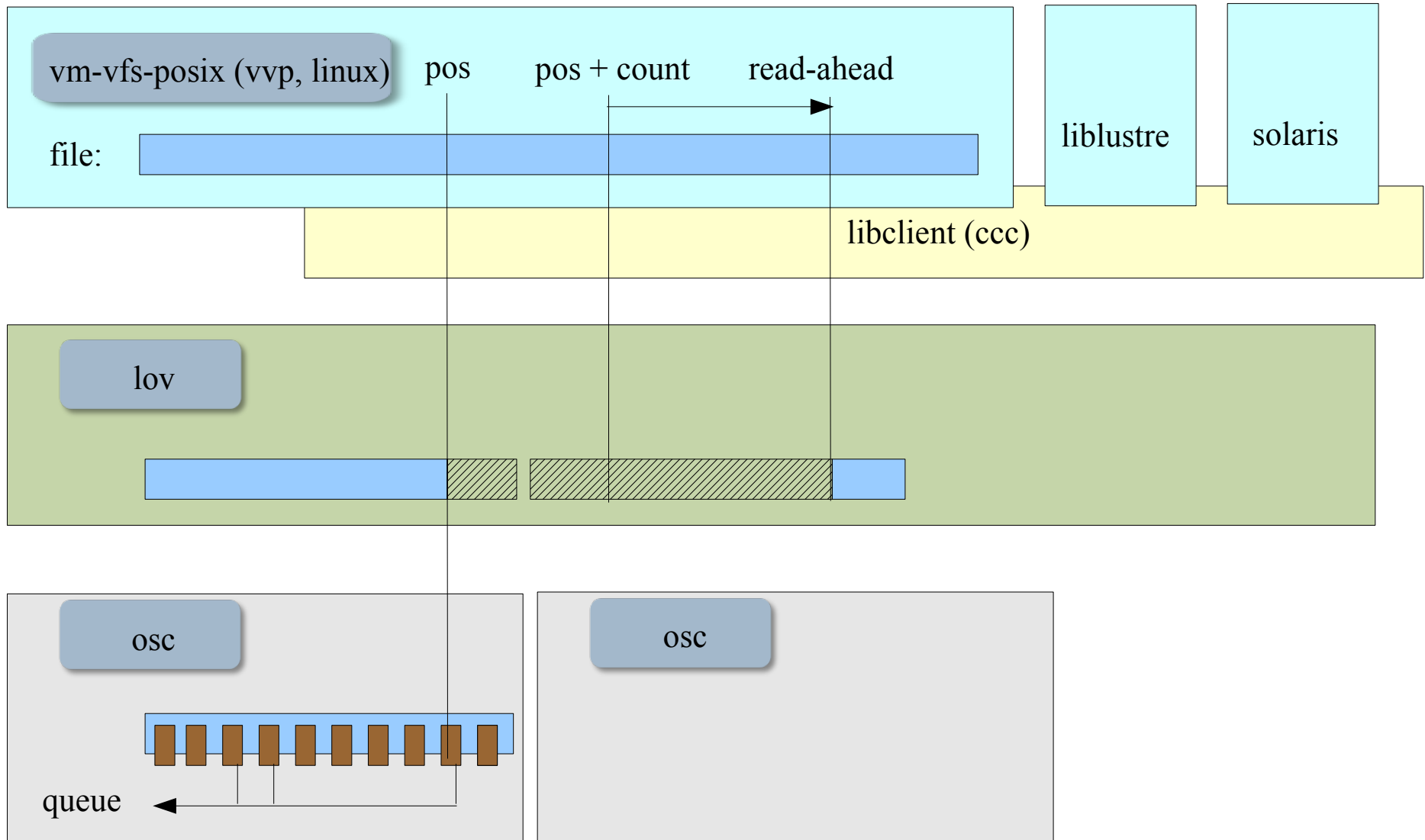
Lustre Group



Layers

user space

```
sys_read(fd, buf, count);
sys_write(fd, buf, count);
fstat(fd, &statbuf);
```



FILE	DECLARATIONS	LINES		
llite/vvp_dev.c	30	559	470	404
llite/vvp_object.c	7	144	114	74
llite/vvp_lock.c	3	90	69	29
llite/vvp_page.c	32	555	462	377
llite/vvp_io.c	30	991	845	696
lov/lov_dev.c	23	515	432	391
lov/lov_object.c	31	651	561	441
lov/lov_lock.c	24	784	677	558
lov/lov_page.c	10	227	187	141
lov/lov_io.c	32	851	740	658
osc/osc_dev.c	17	253	204	163
osc/osc_object.c	14	219	176	133
osc/osc_lock.c	47	1559	1342	929
osc/osc_page.c	20	388	320	264
osc/osc_io.c	17	411	338	231

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

cl_object

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

```
struct lu_object_header {
    unsigned long loh_flags;
    atomic_t      loh_ref;
    struct lu_fid loh_fid;
    __u32         loh_attr;
    hlist_node    loh_hash;
    list_head     loh_lru;
    list_head     loh_layers;
};
```

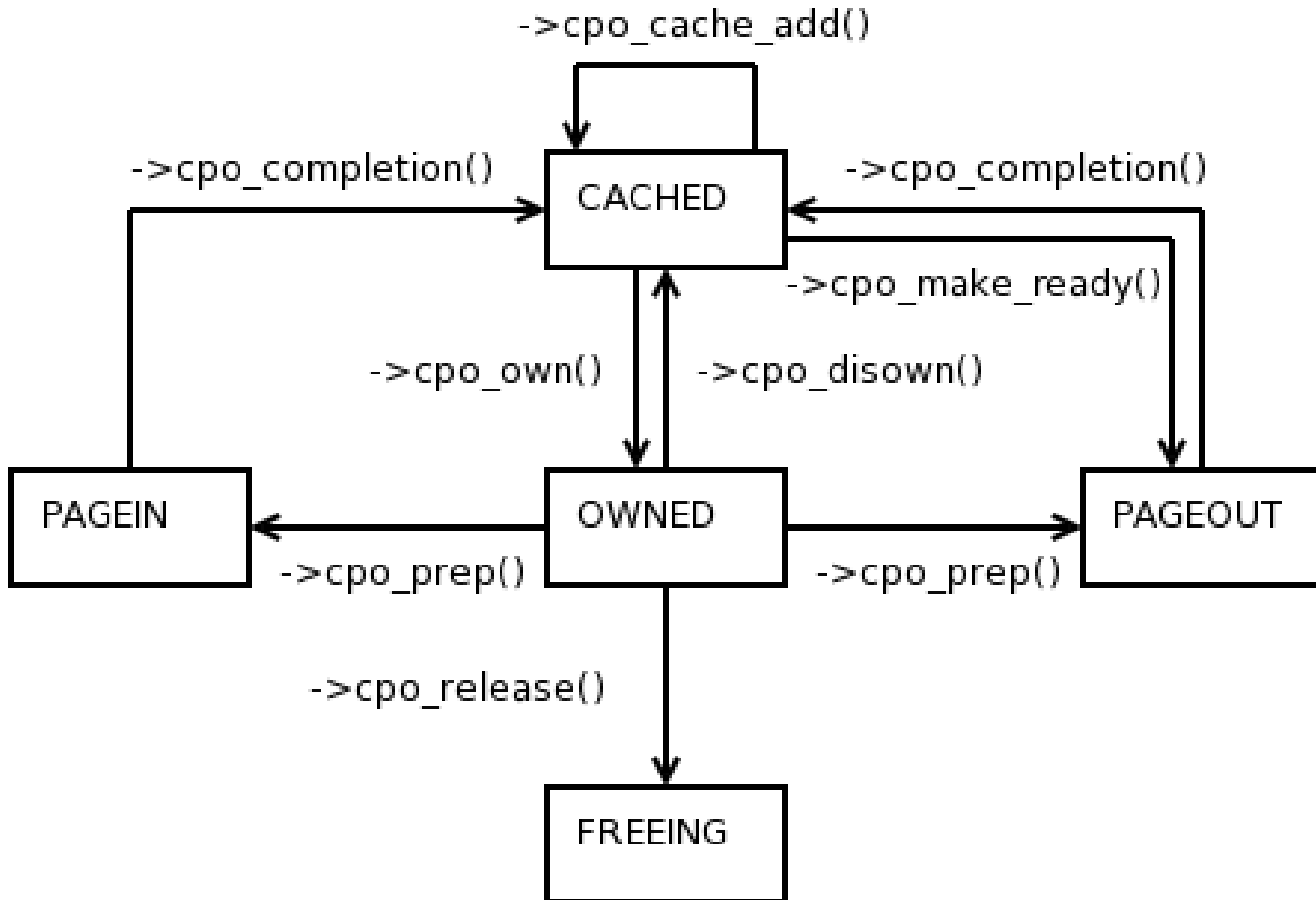
```
struct cl_object_header {
    lu_object_header coh_lu;
    spinlock_t       coh_page_guard;
    spinlock_t       coh_lock_guard;
    radix_tree_root  coh_tree;
```

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

```
struct cl_page {
    atomic_t          cp_ref;
    struct cl_object  *cp_obj;
    pgoff_t          cp_index;
    list_head         cp_layers;
    cl_page           *cp_parent;
    cl_page           *cp_child;
    enum cl_page_state cp_state;
    list_head         cp_batch;
    struct mutex       cp_mutex;
    list_head         cp_flight;
    int               cp_error;
    enum cl_page_type  cp_type;
    struct cl_io       *cp_owner;
}
```

cl_page state machine



Page operations

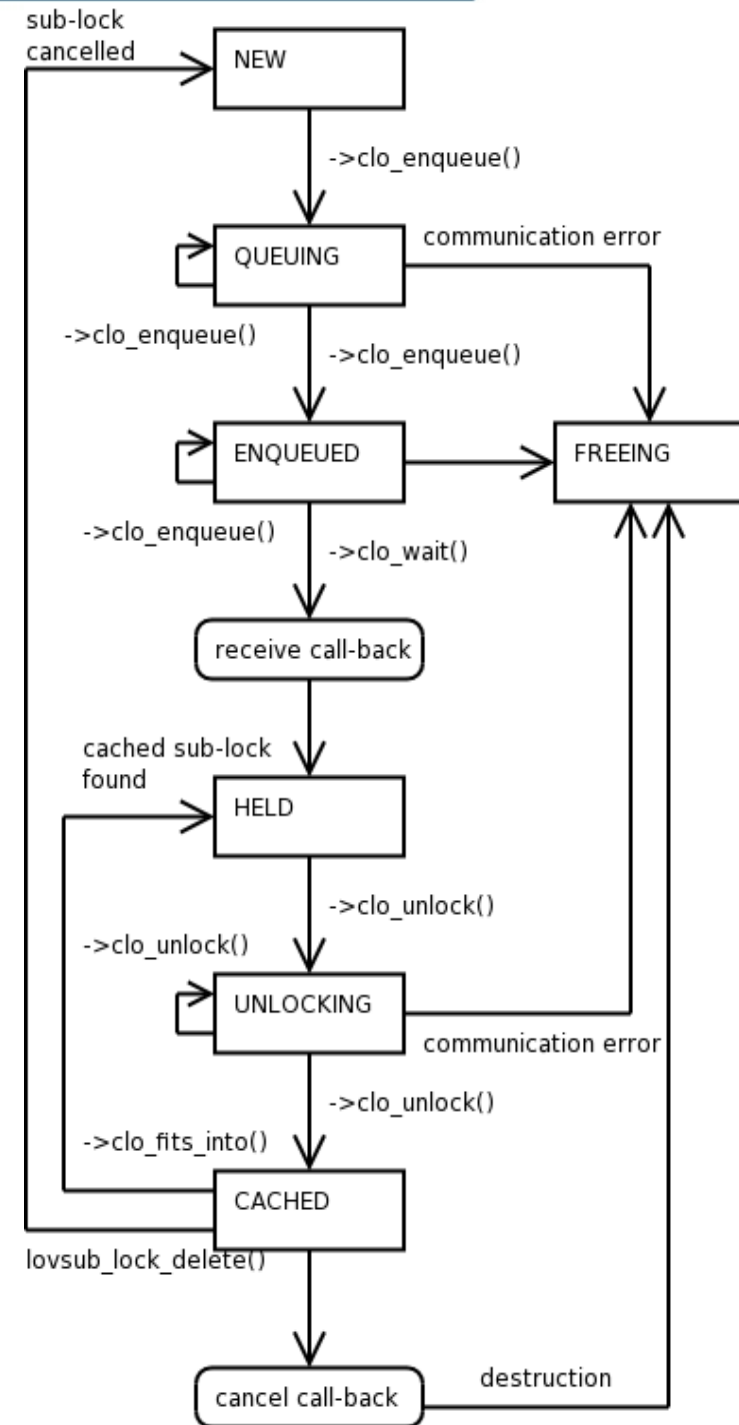
```
struct cl_page_operations {  
    cfs_page_t    (*cpo_vmpage)(...);  
    void          (*cpo_own)(...);  
    void          (*cpo_disown)(...);  
    void          (*cpo_assume)(...);  
    void          (*cpo_unassume)(...);  
    void          (*cpo_export)(...);  
    int           (*cpo_unmap)(...);  
    int           (*cpo_is_vmlocked)(...);  
    void          (*cpo_discard)(...);  
    void          (*cpo_delete)(...);  
    void          (*cpo_fini)(...);  
    int           (*cpo_is_under_lock)(...);  
};
```

cl_lock

```

struct cl_lock {
    atomic_t
    struct list_head
    struct list_head
    struct cl_lock_descr
    enum cl_lock_state
    cfs_waitq_t
    struct mutex
    cfs_task_t
    int
    int
    int
    int
    cll_ref;
    cll_layers;
    cll_linkage;
    cll_descr;
    cll_state;
    cll_wq;
    cll_guard;
    *ccl_guarder;
    cll_depth;
    cll_error;
    cll_holds;
    cll_users;

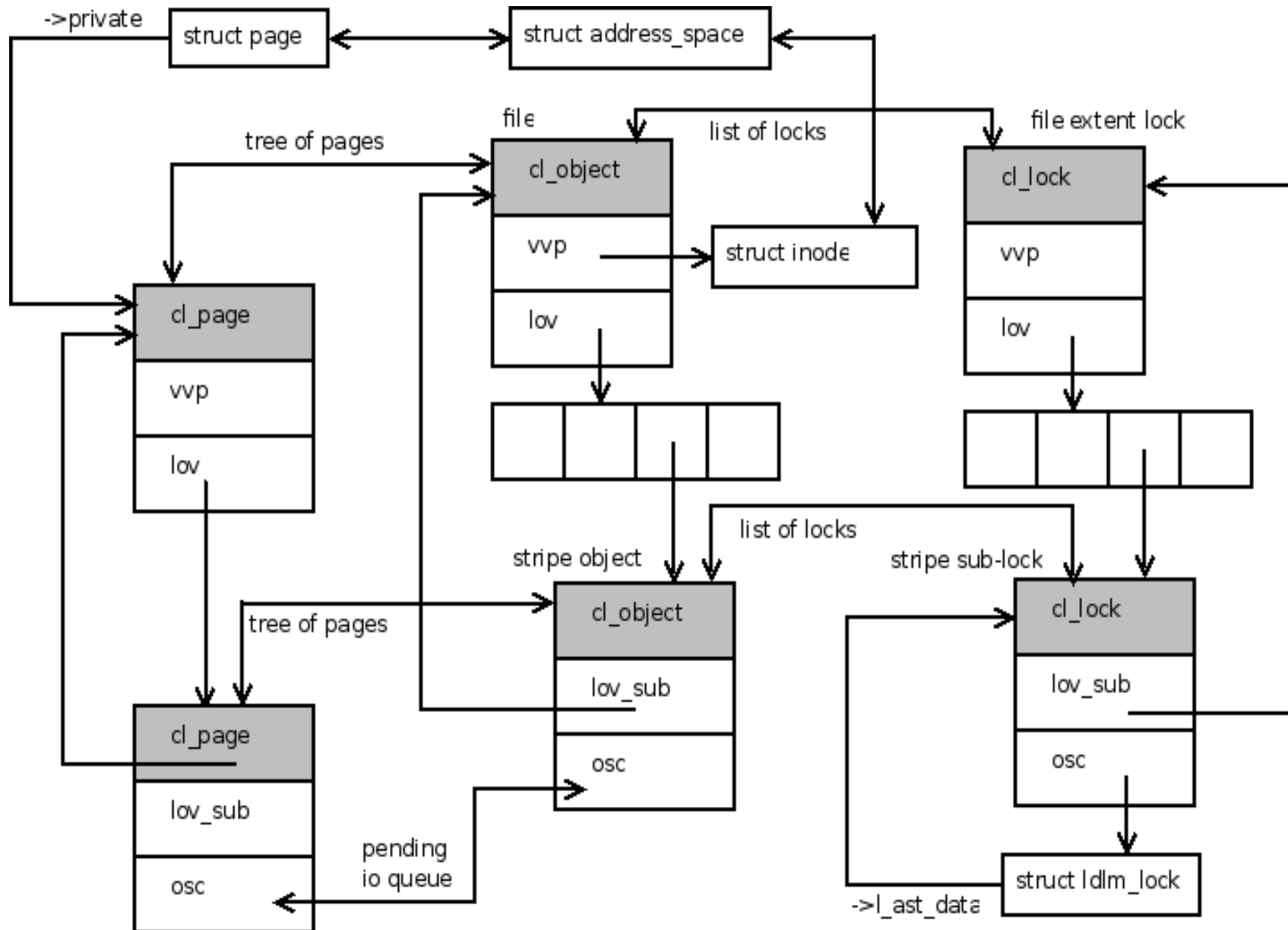
```



Lock operations

```
struct cl_lock_operations {  
    int          (*clo_enqueue)(...);  
    int          (*clo_wait)(...);  
    int          (*clo_unuse)(...);  
    int          (*clo_use)(...);  
    void         (*clo_state)(...);  
    int          (*clo_fits_into)(...);  
    void         (*clo_cancel)(...);  
    unsigned long (*clo_weigh)(...);  
    int          (*clo_closure)(...);  
    int          (*clo_modify)(...);  
    void         (*clo_delete)(...);  
};
```

Files, locks, pages.



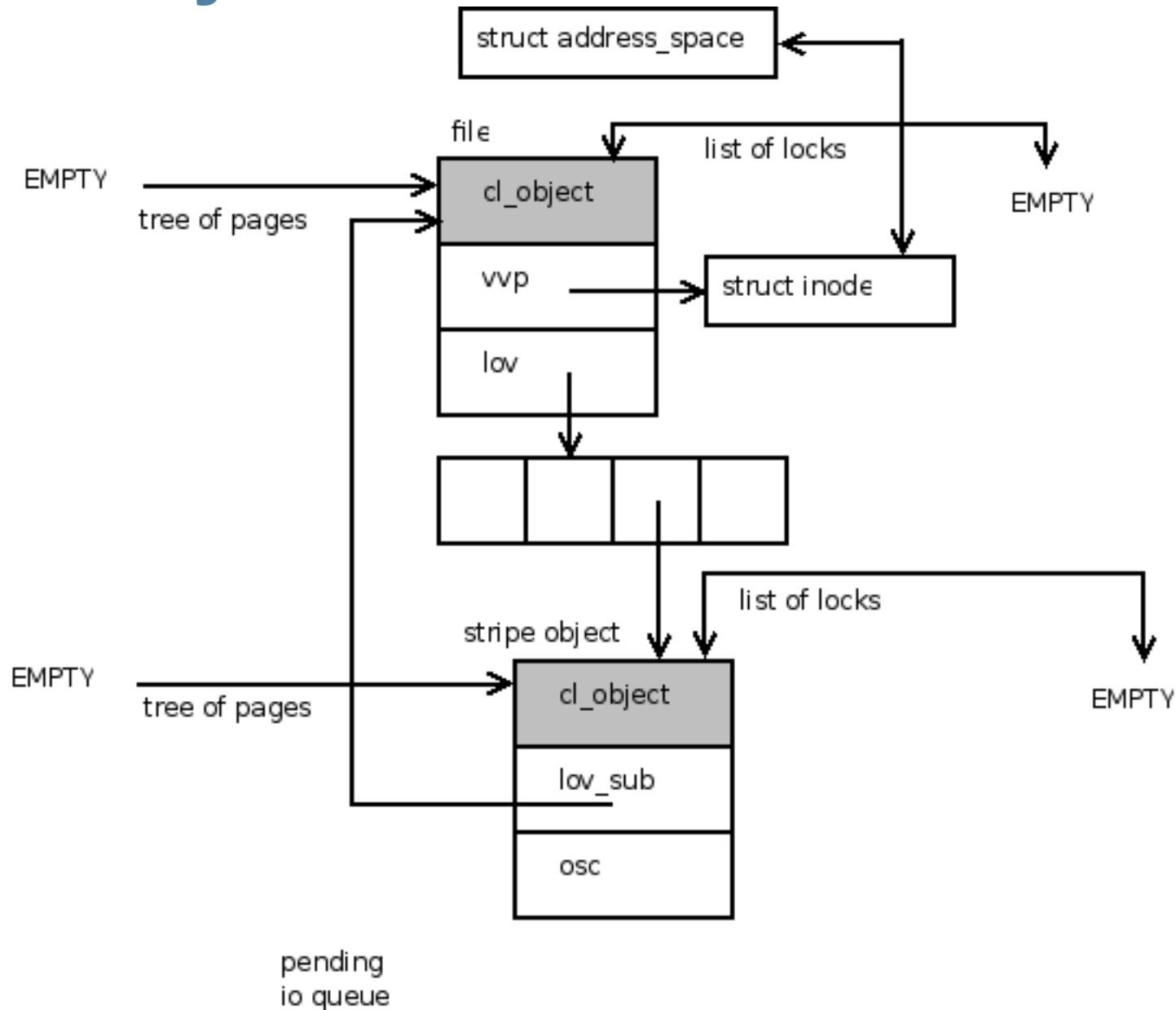
Use cases

- Inode creation
- First read of a file
- Sequential write

Inode creation (regular file)

- ll_lookup_it(), ll_lookup_it_finish(), ll_prep_inode():
lustre_md md;
- ll_iget(), cl_inode_init() (lclient);
- cl_object_find():
 - > htable_lookup(); lu_object_alloc():
 - vvp_object_init();
 - lov_object_init():
 - lov_init RAID0(), for-each-stripe:
cl_object_find(stripe):
 - htable_lookup(); lu_object_alloc():
 - lovsub_object_init();
 - osc_object_init().

Object layers

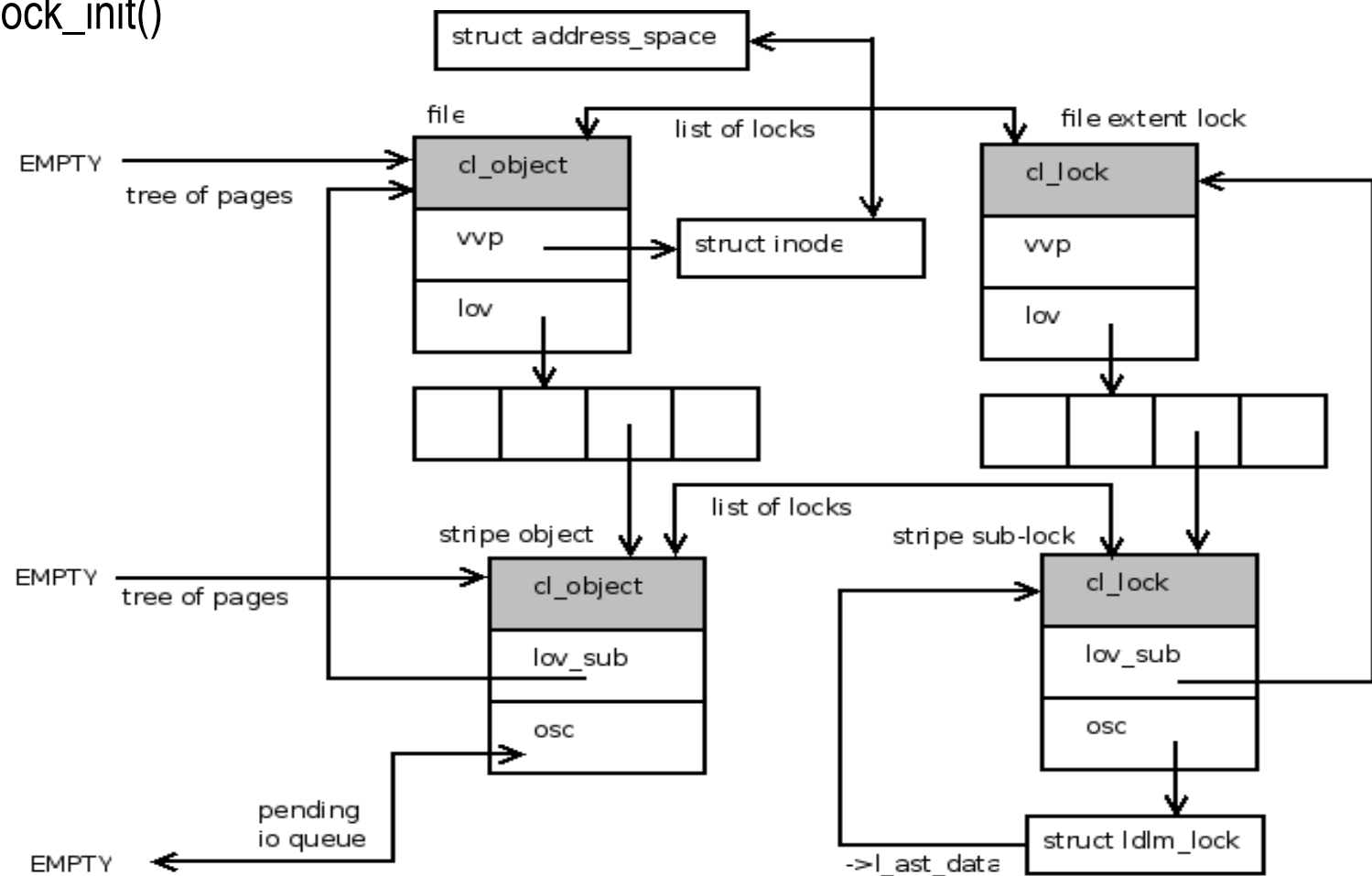


File read (uncached)

- ll_file_read(), ll_file_readv(), ll_file_io_generic(), cl_io_init():
 - > for-each-object-layer: obj_slice->coo_io_init():
 - vvp_io_init() (count == 0, tally, lli_write_sem);
 - lov_io_init(): for-each-stripe: allocate sub-io.
- cl_io_loop(): cl_io_iter_init():
 - > for-each-io-layer: io_slice->cio_iter_init():
 - vvp: nothing
 - lov_io_rw_iter_init(): clip to stripe; init sub-io
- cl_io_lock():
 - > for-each-io-layer: io_slice->cio_iter_lock():
 - vvp_io_read_lock(): ... cl_io_lock_add(), mmap.

File read: create lock (continued)

- cl_io_lock(): for-each-lock: cl_lock_find():
 - > cl_lock_lookup(); cl_lock_alloc():
 - ccc_lock_init();
 - lov_lock_init(): for-each-stripe: cl_lock_find():
 - osc_lock_init()



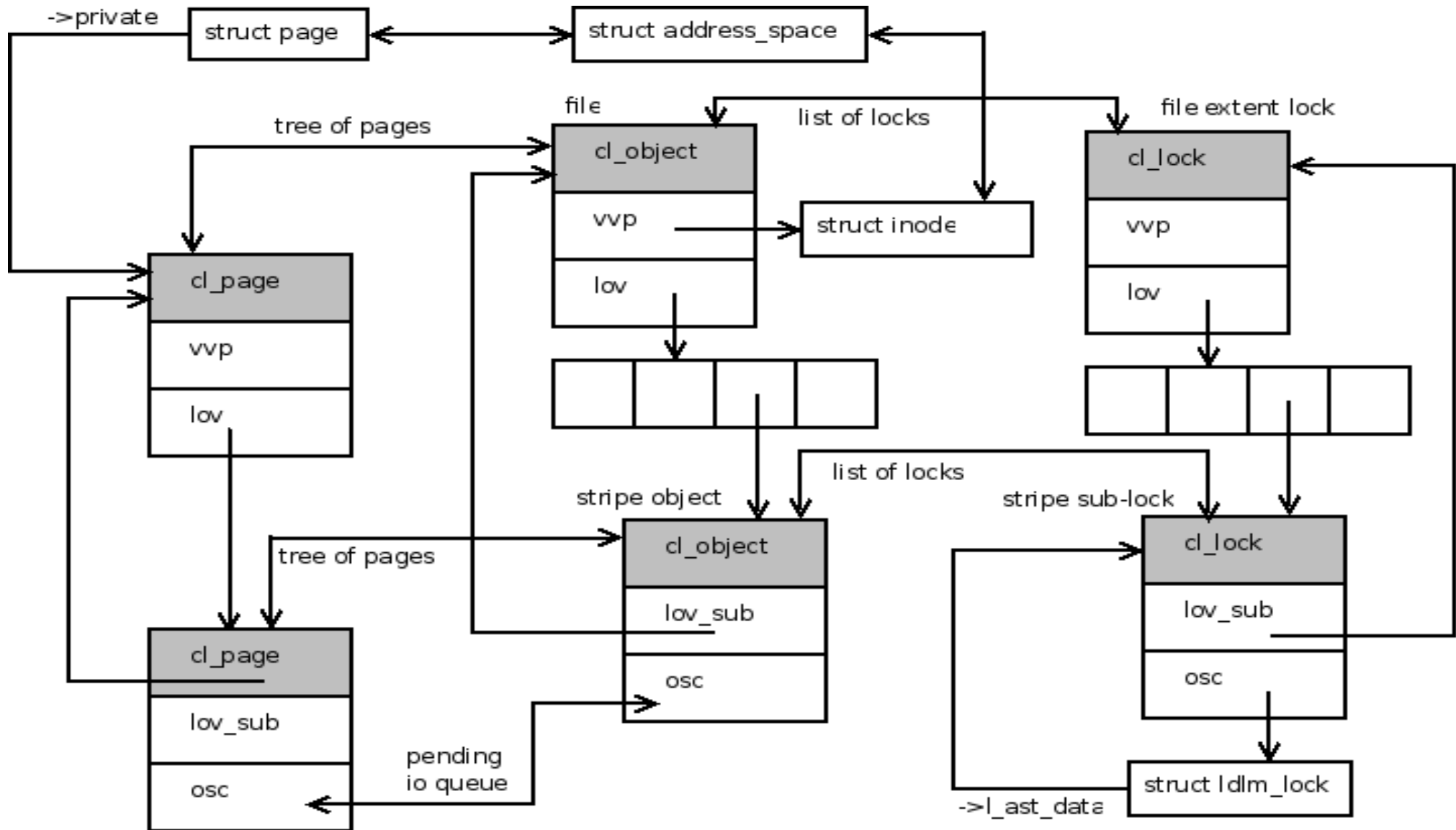
File read: enqueue lock (continued)

- `cl_io_lock(): cl_lock_request():`
 - > `cl_enqueue(): cl_enqueue_try():`
 - for-each-lock-layer: `lock_slice->clo_enqueue():`
 - vvp: nothing;
 - `lov_lock_enqueue():` for-each-stripe:
`cl_enqueue_try():`
 - `osc_lock_enqueue(): osc_enqueue_base():`
LDLM
 - > `cl_wait():` wait until lock state changes....
 - `osc_lock_upcall(): cl_lock_signal():` wakeup.

File read (continued): page creation

- `cl_io_start()`:
 - > for-each-io-slice: `io_slice->cio_start()`
 - `vvp_io_read_start()` (old `ll_file_read()`):
 - `ccc_prep_size()` (short read)
 - read-ahead setup
 - `lustre_generic_file_read()`:
 - for-each-page:
 - `page->readpage(): ll_readpage():`
 - `cl_page_find()`:
 - `cl_page_lookup(); cl_page_alloc()`:
 - for-each-object-layer: `slice->coo_page_init()`
 - `vvp_page_init(): vmpage->private = page;`
 - `lov_page_init(): cl_page_find()`:
 - `osc_page_init(): osc_prep_async_page()`

A file, a lock and a page



File read (continued): page read

- ll_readpage(): cl_io_read_page():
 - > for-each-io-layer: io_slice->cio_read_page():
 - vvp_io_read_page(): update read-ahead; add page to queue;
 - lov: nothing.
 - > cl_io_submit_rw():
 - > for-each-io-layer: io_slice->cio_submit():
 - vvp: nothing.
 - lov_io_submit(): splice pages into queues:
 - for-each-stripe: cl_io_submit_rw():
 - osc_io_submit(): for-each-page: submit page.

File read (finished)

- Pages submitted;
- RPC completes:
 - > `cl_page_completion()`:
 - for-each-page-layer:
 - `page_slice->cpo_completion()`
 - `vvp_page_completion`: `unlock_page()`.
- Page lock completed, `generic_file_read()` proceeds to the next page.

Write

- `ll_file_write()`, `ll_file_writev()`, `ll_file_io_generic()`, `cl_io_init()`:
 - > for-each-object-layer: `obj_slice->coo_io_init()`:
 - `vvp_io_init()` (count == 0, tally, `lli_write_sem`);
 - `lov_io_init()`: for-each-stripe: allocate sub-io.
- `cl_io_loop()`: `cl_io_iter_init()`:
 - > for-each-io-layer: `io_slice->cio_iter_init()`:
 - vvp: nothing
 - `lov_io_rw_iter_init()`: clip to stripe; init sub-io
- `cl_io_lock()`:
 - > for-each-io-layer: `io_slice->cio_iter_lock()`:
 - `vvp_io_write_lock()`: ... `cl_io_lock_add()`, mmap.

Write (continued): cached lock found

- `cl_io_lock()`: for-each-lock: `cl_lock_find()`:
 - > `cl_lock_lookup()`: `cl_lock_ext_match()`, `cl_lock_fits_into()`
- Finds and returns lock
- Lock is already held.

Write (continued): page cached

- `cl_io_start()`:
 - > `vvp_io_write_start()`:
 - `lustre_generic_file_write()`: for-each-page:
 - `page->prepare_write()`
 - `page->commit_write()`: `cl_io_commit_write()`:
 - `vvp_io_commit_write()`:
`cl_page_cache_add()`:
 - `osc_page_cache_add()`:
`osc_queue_async_io()`
 - ...
- Pages are in cache.
- `write(2)` system call finishes.

Write (finished): RPC formed

- OSC decides to send an RPC: `osc_check_rpcs()`:
 - > `osc_send_oap_rpc()`: for-each-page-in-rpc:
 - `cl_page_make_ready()`:
 - `vvp_page_make_ready()`: try-lock page
 - add page to `cl_req`.
 - > RPC is sent.
- RPC completes: `osc_completion()`: `cl_page_completion()`:
 - > `vvp_page_completion()`: unlock a page.

Clio, Greek: κλειώ, /ˈklaɪou/, n:

the muse of history also known as the Proclaimer. The name is from the root κλει-, meaning *recount* or *make famous*.

Nikita.Danilov@sun.com

