HLD of mmap support in Lustre

February 7, 2008

1 Requirements

Distributed mmap functionality in Lustre including:

- execution of files.
- shared mappings between clients for read/write SHARED mappings.
- correct handling of PRIVATE mappings.

Current situation (before mmap patch): access pages via mmap path are not protected by ldlm extent locks.
(Note: all 'lock' in this article means ldlm extent lock)

2 Functional Specification

We need to cover the page accessing not only from read/write path, but also from mmap path. To achieve this goal:

- Override mmap() operation with our own ll_file_mmap().
- Override vm_operation with our own ll_file_vm_ops.

```
static struct vm_operations_struct ll_file_vm_ops = {                         .nopage
        .open           = ll_open,
        .close          = ll_close,
#if (LINUX_VERSION_CODE >= KERNEL_VERSION(2,5,0))
        .populate       = ll_populate,
#endif
};
```

- In ll_nopage(), acquire lock before page cache installation, then drop it after installation finished.
- Clean PTEs mapping to lustre file on lock revocation. Which would generate pagefault on futrue memory accessing.

- In read/write path, lock both the read/write region and mmapped region orderly to avoid deadlock. Introduce ll_tree_lock structure to achieve this.
- Never drop the lock covering a mmaped file to LRU, these locks should be put in LRU as soon as the file is unmapped.

3 Use Cases

3.1 case 1: running a file on lustre

1> Kernel mmap the bin file as PRIVATE | DENYWRITE, loader mmap the share library as PRIVATE | READABLE (for code segment) or PRIVATE | READWRITE (for data segment).

2> ll_file_mmap was called to setup the vm_ops of mapped vma as ll_file_vm_ops.

3> The process try to read the mmapped address, which arose pagefault (no page).

4> Kernel enter do_no_page(), it calls ll_nopage to acquire lock, install page, then drop lock. Then it setup PTE of the page.

5> On the lock revocation, clear PTEs of the lock covering range (except the copied page's PTE), then truncate the covering pages.

6> Next access to the unprotected address will induce pagefault again.

3.2 case 2: shared mmap write (or read)

1> Client A and Client B calls sys_mmap to mmap the same file as SHARED WRITABLE.

2> Each sys_mmap calls ll_file_mmap to setup the vm_ops of mapped vma as ll_file_vm_ops.

3> Client A write to the portion of mmaped buffer, which induce no page fault, kenerl enter do_no_page().

4> do_no_page() calls ll_nopage to acquire lock, install page, drop lock.

5> Client B write to the mmaped buffer, which induce no page fault and lock acquisition too.

6> The lock on Client A was revoked, and the PTEs was cleared.

3.3 case 3: private mmap write (or read)

The different of private mmap with shared mmap is that:

1> 2.4: The PTEs of private mmap region will not be cleared on lock revocation. Because their pages maybe copied pages.

2> 2.6: The PTEs of copied pages will not be cleared on lock revocation.

4 Logic Description

The changes in llite module:

- Add ll_tree_lock structure and tree lock functions.
- Add ll_file_vm_ops structure and ll_file_mmap() operation.
- Add functions to clean PTEs of specified range.
- modify ll_file_read()/ll_file_write() functions.
- modify ll_pgcache_remove_extent() function.

The changes in ldlm module:

- Add ldlm_cli_join_lru(). Which can join/split all unused locks of the specified resource to/from LRU.
- Add lock flag LDLM_FL_NO_LRU for indicating if the this lock should be in LRU.

The changes in lov module:

- Add lov_join_lru().

The changes in osc module:

- Add osc_join_lru().

The changes in obd module:

- Add obd_join_lru() operation.

5 State Management

5.1 The LRU of unused locks

- The unused lock which covering mmap file will not be drop in LRU;
- As soon as there is one portion of file be mmapped, all the unused locks of this file would be split from the LRU.
- As soon as all the the portions of file be unmapped, all the unused locks of this file would be added into the LRU

5.2 Page mapping

- If the page is mapped to some inode, the page mapping should be set as the inode's i_mapping.
- If the page is newly copied page, the page mapping should be set as NULL.
- If the page has been swapped, the page mapping should be swapper_space.

6 Protocol, API's, Disk format

No configuration changing, no disk format changing. Need a new operation obd_join_lru() for obd device.

7 Scalability and performance

- To avoid canceling locks for running file under lock pressure, we introduce obd_join_lru facilities.

8 Recovery

Imagine the scenario like that: A client is evicted while a process is runing, after recovery, the running binary has been changed by other clients, thus the process run into an unpredictable condition now. So we should check if the binary file is changed before reading page on nopage fault, if it changed, just terminate the process instantly.

9 Alternatives

N/A

10 Focus for inspection

- Serveral known races: change inode size without i_sem hold; lock only covering page cache installation but not covering the PTE setup phase.
- If there is any case missed in pagefault path?
- If the vm operations: open(), close() are proper for track mapping files?