Patch free lustre client in tiger

Liang Zhen

2006/02/07

1 Introduction

This HLD describes how to construct patch-free lustre client in OS X Tiger (Darwin 8.x).

Reasons we have to re-design Lustre client:

- Darwin8.x is changed a lot from Darwin7.x because Apple wants to standardize kernel-developing interfaces, and makes kernel package binary-compatible with different OS releasese.
- vfs of Darwin8.x is changed, lock/unlock of fsnode are handled lessly in vfs, namecache is handled in vfs.
- Users of OS X don't like to install a new kernel(patched) when they use Lustre.

2 Distribution

People insterested in:

- Design of Lustre client
- Lustre developing in OS X
- Filesystem development in OS X

Should read this document. Understanding of xnu(kernel of Darwin) and Lustre are assumed.

3 Background Of Darwin8.x

Structure of xnu is not supposed to be introduced here, but following features about xnu and OS X should be noticed:

- locking. Tiger doesn't always lock vnode in vfs while performing operations on it, it's optional for FS. Filesystem can set VFS_TBLTHREADSAFE or VFS_TBLFSNODELOCK to indicate if accessing of the filesystem is thread safe or not. lock_fsnode/unlock_fsnode will be called in VNOP_* by vfs before entering to filesystem backend, if the FS isn't thread safe. No lock will be taken if the filesystem is created with thread-safe setting, by this way, filesystem should decide where to lock/unlock the file-node in vnops.

- name resolution. XNU has standard UNIX namei() interface to resolve path-names. namei() descents through file system tree, calling VNOP_LOOKUP() to resolve single path component. Symlinks are handled through VNOP_READLINK() method. It is possible to determine when last path component is being resolved.

- name cache. XNU VFS is vnode based. All VNOP_* operations take vnode as argument. There is simple name-cache, In Darwin7.x, it is not used by the VFS layer. File system backend is supposed to use it by its own. This provides some advantage for Lustre: all file system operations go through VOP_LOOKUP() stage (even if name/vnode is in name cache). But things are changed in Darwin8.x, vfs uses namecache somehow. VFS: lookup()->cache_lookup_path(), cache_lookup_path() will try to find vnode and revalidate it, the revalidation for local filesystem is fine, but for remote authorisation it's really rough. The revalidation will never call any operation of backend filesystem, it just think the vnode from name-cache is valide if the last VNOP_LOOKUP() is happened in 2 seconds and nobody change it locally. This is really pain for us.

  VNOP_LOOKUP() records a timestamp in vnode, cache_lookup_path() will check it later.

- No robust way to pass per-thread information from VNOP_LOOKUP() to other end-point VNOPs (i.e, VNOP_OPEN, VNOP_CREATE), that means, it will be hard for us to use lookup-intent in xnu, although we can know lookup-intent somehow in VNOP_LOOKUP(): nameidata->componentname->cn_nameiop can be (LOOKUP, CREATE, DELETE, RENAME).

- no way to access file descriptor in filesystem layer(file struct can only be accessed in vfs layer), as we mentioned before, all VNOP_* operations take vnode as argument, we can't access per-file data in vnops.

- Apple will use intel to replace ppc, nobody knows if xnu will still be opensource after intel version released(To prevent people from running OS X in their own machine).

4 Requirements

According to the description of the this project, the requirement include

- Construct a Lustre client for Tiger without patch of xnu or with a small patch.
- The Lustre client has basic filesystem features.
- The Lustre client may have no lookup-intent operations in lookup.
- The Lustre client uses 0-conf.
- Try our best to provide binary-compatiblity with OS X (reduce hacking), just as expecting of Apple.

5 Functional Specification

Tiger client will be built as several sub-modules, some of them have been described in xll_design.lyx by Nikita, here just talking about modules have been changed in Tiger:

- filesystem vfs/vnops framework, interact with BSD vfs system
- abandon or handle lookup-intent.
- name cache
- file IO operations: The key issure for patchless client is about read-ahead.
- page cache: To re-use generic Lustre code in osc and lnet, each platform has to provide implementation of *cfs_page_t* API. This API closely matches Linux page cache interface. To implement it on xnu, intermediate layer (*xll page cache*) is introduced which sits on top of mach vm and provides data-structures and interfaces similar to ones found in Linux. cfs_page_t API is trivially implemented in terms of xll page cache.

5.1 Filesystem VFS/VNOPS framework

This part is similar to what has been done in Panther client, with a few exception:

- VNOP_LOCK/VNOP_UNLOCK is not existed anymore. There is no KPI to call filesystem backend fsnode lock/unlock, vfs provides generic functions lock_fsnode()/unlock_fsnode to protect vnode to be accessed by other threads if the filesystem is not thread-safe. If the filesystem is thread-safe, fsnode should be lock/unlock totally at filesystem backend. Lustre client should be thread-safe in Tiger to get better grained operations, although we can start from thread-unsafe mode. In thread-safe mode, as

fsnode lock/unlock will only be called by filesystem, it's possible to map them to ldlm locks. To get this step we need another desigh document, for now ldlm locks and vnode locks will still be aquired separately (Just like in Panther).

- VNOP_ACCESS should be designed carefully. VNOP_ACCESS is now used by vfs lookup() (In Panther, it's called by filesystem backend in VNOP_LOOKUP), it's always called before each VNOP_LOOKUP(), by vnode_authorize(), to check KAUTH_VNODE_EXECUTE of current vnode, we should handle this carefully to avoid redundant RPCs.

- vfs_mount(): Filesystem should set MNTK_AUTH_OPAQUE and MNTK_AUTH_OPAQUE_ACCES flags during initializing. MNTK_AUTH_OPAQUE is used to indicate authorisation decisions are not made locally, MNTK_AUTH_OPAQUE_ACCESS is set to tell vfs that VNOP_ACCESS is reliable for remote auth, otherwise vfs will attempt to formulate a result based on VNOP_GETATTR data (before all VNOP_LOOKUP).

5.2 lookup-Intent

- No lookup-intent in patchless version, because there is no reliable way to get lookup-intent in current VNOP_LOOKUP() without patch. Nameidata->componentname->cn_nameiop describes operation-intent roughly as (LOOKUP, CREATE, DELETE, RENAME), and it's possible to determine when last component is being resolved, but it's still impossible for VNOP_LOOKUP() to know real lookup intent like: IT_GETATTR, IT_OPEN, IT_CREATE(mknod, create, mkfifo all use CREATE in nameidata), so we will not use lookup-intent for patch-free version, working rpc can only be set by end-point vnops like VNOP_CREATE, VNOP_OPEN, VNOP_LOOKUP() will do nothing more then trying of matching local lock or sending IT_LOOKUP request to lookup.

- If we want lookup-intent, we need to add precise intent-operation to nameidata in xnu-patch. Although there is still no way to pass per-thread data from VNOP_LOOKUP to other end-point vnop in xnu, but xll_session (refer to xll_design.lyx by Nikita) can be abandoned. There are two ways to save/pass intent-lock information:

  - intent-lock status can be passed from VNOP_LOOKUP() to end-point vnop (like VNOP_OPEN) by a hash table maintained by our client module (indexed by thread address or vfs_context address), this can be reused to client for other platforms (like patchless linux client).
  - Patch vfs_context of xnu, add a pointer in the structure and save intent-lock information in it while lookup. It's very easy way, because vfs_context is always be passed to filesystem backend in all vnops,

and it's exactly per-thread data because most time it's just staying in stack of current thread.

5.3 Name cache

We can do nothing to namecache because there is no callback entry for filesystem backend in xnu. We can't jump over the rough revalidation of namecache in vfs unless we create/use our own namecache. Things we can do are:

- complain in Darwin-kernel-list, it will take some time to get feedback, we will have to wait and see.
- Patch xnu, it's easy to create a patch for this, we'll not add callback entry for revalidate, but we can break cache_lookup_path() with set a flag and let VNOP_LOOKUP() take over namecache searching, revalidating, and real-lookup, just like pather.
- Hack without patch xnu. Purpose of hack is same with patching: break cache_lookup_path() called in lookup(), let VNOP_LOOKUP() to take over everthing. We will take this way for now, but still hope Apple can provide a better interface for us to get over the problem in the recent future.

5.4 read-ahead

Llite in Linux uses file->private_data to save read-ahead context of each file descriptor, as we know, xnu(BSD) vfs is vnode based, there is no way to access per-file data like file descriptor in vnops, so it's not easy for xnu to use read-ahead algorithm from llite.

However, read-ahead is only for improving of read performance, it's fine for us to design another way to implement read-ahead.

There can be some choices for us to deal with read-ahead:

5.4.1 One read-ahead context for one vnode

Like the way used by cluster IO of xnu, create one read-ahead context for each vnode, only one thread can own the read-ahead context(by holding the lock associated with it), other threads can't own the context will run without read-ahead, this allows multiple readers to run in parallel and since there's only one read ahead context, there's no real loss in only allowing one reader to have read-ahead enabled.

- Each time reading from different offset(by different reading from threads or by lseek), read-ahead context will be reset
- It's fine for single thread reading, but not very helpful when a lot of threads are trying to read the same file, because only one of them can issure read-ahead and read-ahead context is always reset.

5.4.2 No read-ahead context

It's the simplest way to use read-ahead, read-ahead window always starts from next page of current read, length is min(permited_length, end), no context is loaded/saved in read-ahead.

5.4.3 Per-process readahead context in LRU list

A LRU list for saving read-ahead context is added to fsnode (xnode), readers can get a matched read-ahead context or create a new read-ahead context from the LRU list. Read-ahead context will be freed only if it's at the end of the LRU list, and it hasn't been accessed for long time or there are too many read-ahead contexts in the filesystem. It's possible that a read-ahead context is freed even if there is still task wants to use it, but it's not hurting as a new context will be allcated when it's required. We have no way to allocate/free read-ahead context precisely in xnu, because threads in kernel can read/write file by vn_rdwr() just after getting vnode by namei(), without open.

By this way, we can take read-ahead algorithm used by llite.

5.5 Page cache

Page cache of xll is well designed and re-usable for Tiger, we are not going to design & implement another one. For more information, please read xll_design.lyx by Nikita.

6 Use cases

The lustre client will be used with 0-conf in Tiger, that means user can mount/umount and use it just like the way they use nfs.

6.1 Intent handle

No intent handle in no-patched version.

6.2 read-ahead

read-ahead context is updated before read_page(), and issured asynchronizingly after reading of page.

6.3 page cache

Please refer to xll_design.lyx by Nikita.

6.4 Shared functions for both Linux and XNU

There are only few functions like lustre_process_log can be shared between Linux and xnu, because vfs frameworks of two OS are too different(VFS of xnu is based on vnode), and xnu will use our own page cache.

6.5 Xnu should be adapted to new KPIs

Apple provided a limted set of KPIs and structures, we can't access kernel services as free as before.

7 Logic Specifications

7.1 Intent handle

- VNOP_LOOKUP() always use IT_LOOKUP as locking request, even in the last step of nameing resolve, because we don't know what exactly user wants to do while name resolving.
- The lock is always released before exiting VNOP_LOOKUP, nothing can be carried to end-point vnops.
- The working request can only be sent in end-point vnops.
- VNOP_ACCESS is called by vfs all the time while name resolving, be careful with it, otherwise a lot of un-expected rpcs will be sent.

7.2 Name cache

We need to explain how xnu is implemented on this before discussing our hack:

```
lookup()
{
   ......
   cache_lookup_path(...);
   ......
   if (nd->ni_vp != NULL)
      goto found;
   ...
   VNOP_LOOKUP(...);
   ...
found:
   ....
}
cache_lookup_path(...)
{
   ......
```

```
        while (1) {
            ...
            if (remote_auth && (now_second - vp->auth_timestamp) > 2)
               break;
            ...
        }
    }
    VNOP_LOOKUP(...)
    {
        vp->auth_timestamp = now_second;
        ...
        vp->vnop_lookup(...)
        ...
    }
```

VNOP_LOOKUP() updates timestamp in vnode each time before calling of vnop_lookup provided by filesystem backend , cache_lookup_path() checks the timestamp in the next calling of lookup() if the vnode is still in namecache, it thinks it's valide while last VNOP_LOOKUP() is called in 2 seconds. So, the thing we can do is using an old time to replace vp->auth_timestamp set by VNOP_LOOKUP() in vp->vnop_lookup().

```
    xll_lookup(...)
    {
        ...
        vp->auth_timestamp -= 2;
        ...
        cache_lookup(...)
        if revalidate(...)
           return
        ...
        real_lookup(...)
    }
```

By this way, loop in cache_lookup_cache() will be broken by condition ((now_second - vp->auth_timestamp) $>$ 2), and this change wouldn't affect anything else because vp->auth_timestamp is never used by other functions. The only thing makes us unhappy is we have to hack out define of vnode, because xnu hasn't exported it to kext users.

7.3 Read/Write

- Get extent lock of required range
- if reading, glimpse size of fsnode
- issure write/read/read-ahead request

- Put extent lock of required range

Logic of readdir is very similar to read of regular file, but with different lock.

7.4 Read-ahead

- Aquire read-ahead context from fsnode
- update read-ahead context
- issure read-ahread
- Release read-ahead context.

8 State Management

N/A

9 Alternatives

N/A

10 Focus of Inspection

No lookup-intent while rename resolving.

11 Inspection Summary