Status of the Linux NFS client

- Introduction - aims of the Linux NFS client
- General description of the current status
- NFS meets the Linux VFS
  - Peculiarities of the Linux VFS vs. requirements of NFS
- Linux NFS read/write code
- The development patches
- Future developments?

Introduction

Aims of the Linux NFS client

'Prime directive':

- Fully compliant client implementation of NFSv2/v3 with NLM locking over both UDP, TCP
- Optimal performance.
  - Particular emphasis on caching
- Minimal change to the Linux VFS - modularity.

Secondary goals:

- Support for diskless workstations (a.k.a. NFSroot)
- Support for different authentication schemes.
- Support for layered filesystems (cachefs, etc.).

General description of status

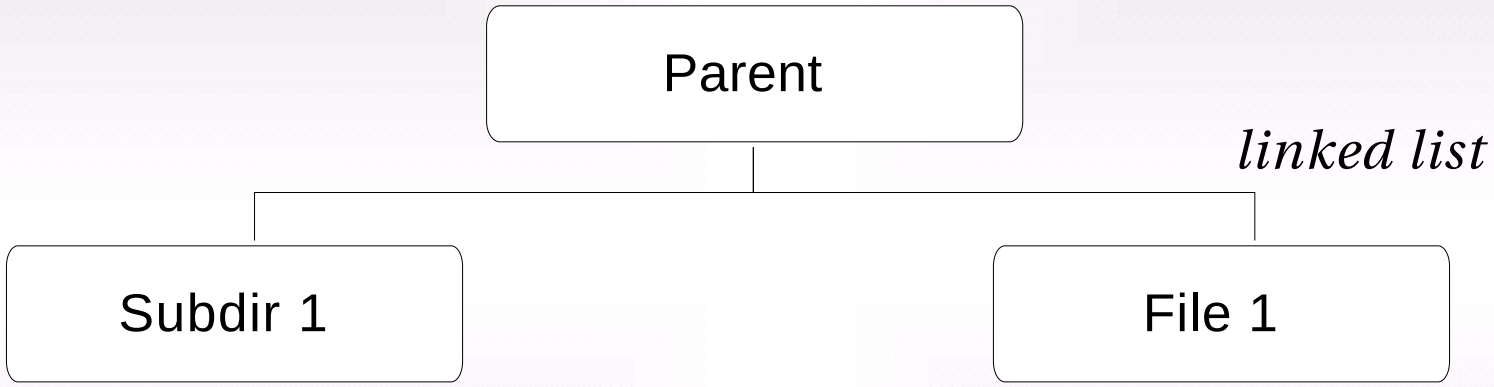
Current NFS features implemented in kernel 2.4.18

- RPC version 2
  - Transport over IPV4 using both UDP and TCP supported
  - AUTH_NULL + AUTH_SYS authentication. No support for AUTH_DES or AUTH_KERB yet...
- NFS version 2
  - Full implementation. No features missing.
- NFS version 3
  - Almost complete. All features except READDIRPLUS, ACCESS (see later slide), and exclusive CREATE implemented.
  - Fail Connectathon test against servers that don't return attributes in CREATE - otherwise all tests passed so far...
- NLM versions 1, and 4
  - Full implementation including reboot recovery.

NFS meets the Linux VFS

Linux implementation features - The dcache

- File paths are represented in the Linux dentry cache (dcache) which is completely managed by the VFS



- dcache is completely separate from data & metadata caches.
- Path is automatically broken down into single elements.
- Mount point traversal, symlink traversal etc. all performed at VFS level.
  - Filesystem gets called back when looking up an uncached path element, or when revalidating a cached one.

NFS meets the Linux VFS

The dcache

- Fast getcwd() can be handled fully by the VFS. No need to call back NFS subsystem
- Problem: dcache is a static structure, hence client and server path information may differ.
  - Renames on the server are only reflected when looking up a new path.
    - Implies that getcwd() and chdir("..") can sometimes give 'unexpected' results.
  - If a directory in which an existing process is working gets moved from one location to another, you might end up aliasing the directory trees.

NFS meets the Linux VFS

VFS services - File data caching

- File metadata saved in the 'inode' cache.
  - Full 64-bit metadata available to the filesystem via private fields
  - However, some data is 32-bit only at the VFS & user level
    - inode number (a.k.a. fileid)
    - (a|c|m)time
- File data is cached in a 'unified buffer/page cache'.
  - Data neutral - caches raw READDIR data, symlink data, regular file data,...
  - Individual pages are tagged by means of an 'unsigned long' index
    - Gives 44-bit address space on i386 (32-bit index + 12-bit page size).
    - Minimizes use of slow (on 32-bit systems) 64-bit arithmetic within the kernel

NFS meets the Linux VFS

The darker side of the Penguin - known NFS problems

- The stable Linux kernel (currently linux 2.4.18) does not implement close-to-open semantics properly:
  - Cached attributes are sometimes not revalidated on open(). Problem affects open("."), open("..") and is due to the dcache assuming it doesn't have to revalidate those dentries
  - Inefficiency due to use of LOOKUP in situations where GETATTR would suffice.

- NFSv3 ACCESS call is not implemented correctly. Need caching support in order to make progress. Currently only call server for the following cases:
  - Check for root squashing
  - Check if server has some ACL that overrides the case when standard UNIX permission bits deny access.

NFS meets the Linux VFS

known NFS problems

- Readdir currently does not respect RFC1813 with respect to 'dtpref'. Linux never issues requests with sizes greater than PAGE_CACHE_SIZE (due to limitations of the page cache API). On most 32-bit platforms PAGE_CACHE_SIZE = 4k.

- User-land libc implementation relies heavily on being able to seek() the READDIR stream. It also mixes 64-bit and 32-bit readdir system calls. Leads to nasty incompatibilities against certain server platforms. Kernel 'hack' is available on my beta-test site that 'fixes' problem for known servers, but problem really needs to be solved in libc.

Linux NFS read/write code

Features

- Has support for both synchronous and delayed reads/writes.
- All read/writes are done through the page cache
  - No support for any form of uncached read/writes.
  - I/O access to the page cache is serialized by a per-page bit-lock.
    - => VFS supports read/write to single pages only to avoid deadlocks.
    - => NFS client subsystem must do its own clustering of pages in order to achieve > PAGE_SIZE read/writes. (This can of course not be done for synchronous writes.)
- Byte range POSIX locks via the NLM protocol
  - includes support for delayed writes.

Linux NFS read/write code

Delayed writes

- For delayed writes:
  - Full support for NFSv3 server-side write caching (a.k.a. unstable writes).
  - Support for coalescing several contiguous requests into a single RPC call. Maximum value of wsize is currently 32k.
  - Only one one request allowed per page - flush out older requests that are not contiguous and/or have incompatible credentials.
    - => writes into the same page will be fully serialized when doing byte-range locking.
  - Limit of 256 cached/pending read+write requests per mount. Limit required in order to regulate memory footprint.

Linux NFS read/write code

Delayed reads and misc other features

- For delayed reads:
  - Coalesce requests from contiguous page ranges. Maximum rsize = 32k
  - Generic readahead is supported via the standard VFS interface. Users can either call the 'sys_readahead()' system call in order to manage their own readahead, or allow VFS to manage it automatically.

- Other read/write features/bugs that are peculiar to Linux:
  - The Linux shared mmap() interface does not flush out data on close()/munmap() (ordinary writes do!)

The development patches

Available for beta-testing

- Represent a collection of patches to the kernel source code, that are written by others + myself. Not guaranteed to make it into the kernel.
- All NFS client patches available from web-site

  http://www.fys.uio.no/~trondmy/src
- Apply to stock kernel from ftp.kernel.org (Patches are NOT guaranteed to apply to pre-patched kernels from the various Linux distributors)
- Current highlights:
  - Fix NFS close-to-open problem
  - NFSv3 READDIRPLUS implementation
  - Implement O_DIRECT file read/writes.
  - Finer grained SMP locking
- Problems should be reported to me (trond.myklebust@fys.uio.no) and/or the Linux NFS mailing-list NFS@list.sourceforge.net.

Future developments

Suggestions?

- **Secure RPC (the beginnings of a backport from the NFSv4 codebase has been developed by Andy Adamson & myself during this Connectathon!)**
- **Proper treatment of credentials in a BSD-like scheme.**
- **NFS over IPV6.**
- **NFSv4 (see Andy Adamson's talk)**

- **Documentation?**