10/23 CS240 - NFS Lease

Announcements

For next class (Tuesday 10/28)

1. Read: The Design and Implementation of a Log-Structured File System
2. Submit answers to reading questions (see course schedule) before class

Looking ahead:

Lab 1 is due end of day November 2nd (Sunday, next week)

Paper backgrounds

- NFS - 1985 Summer USENIX Proceedings
  - Industry paper - light on related work, marketing?
  - Worst is better example?
- Leases - 1989 Symposium on Operating Systems Principles (SOSP)
  - Simple but cool idea
  - Analytic model and evaluation using trace-driven simulation

File System Review

- Layers
  - System call
  - File System
    - Inodes point to blocks, indirect blocks, double indirect blocks
    - Block cache
    - `sync, fsync`
  - Device Driver
  - Disk device
- Workstation cluster (distributed systems)
  - `rcp /usr/home/mendel/foobar olive:/usr/home/mendel/foobar`
  - Single system image
    - Centralized server-based, custom FS code on both client and server, RPC
    - Apollo, lots of academic research systems
    - Consistency
      - Close-to-open consistency
  - ND - Sun's network disk - What's the sharing problem?

UNIX File System Semantics

- Hierarchical single name space
  - File system per disk stitched together by `mount`
    - Most other systems exposed disks
      - DOS: `C:\AUTOEXEC.BAT`
      - VMS: `NODE42::DISK1:[USERS]FILE.TXT;1`
- Files are array of bytes
  - Files frequently used for program communication
- Simple access control - 9 bits in inode, `rwx` for bits for user#, group#, other
- Weirdness:
  - Can remove an open file and keep using the file
  - Can change permission after opening and keep using the file
  - Superuser access (uid==0)
  - Can change current user id (`setuid`())

NFS Design Goals

- Independence
  - Operating system - SunOS/UNIX, new systems calls (`getdirentries?`)
  - Machine - Motorola 68020 vs x86? (XDR?)
- Crash Recovery
  - Client crash - like local machine failing (i.e. 30 second writeback, `sync`, fsync)
  - Server crash - Can't lose data
- Transparent Access
  - Local/Remote - all programs should work both local and remote (leverage `mount`)
  - Unix Semantics
- Reasonable Performance
  - 80% of local machine, match ND

NFS Design

- NFS **Protocol** - Sun RPC
  - Stateless - What did it mean for the server?
  - Idempotent?
  - **Opaque** fhandle and readdir **cookie**
- Client side - Wedge in new interfaces into kernel
  - C++ abstract classes before they were a thing - C function pointers
  - Virtual file system (VFS) - one per mount point - `umount, root, statfs, sync`
    - Useful for adding other file systems (MSDOS, CDROM, …)
  - Virtual Inode (vnode) - one per active inode - lookup, open, close, create, rwdr, …
  - Late binding:
    - `NODE42::DISK1:[USERS]FILE.TXT;1` **vs** `mount("NODE32:/DISK1")`

What did it take for an admin to set up sharing with NFS?

- Server side: Add file system to `/etc/exports`
- Client side: `mount my-server:/disk/u1 /home/u1`
- Mount options:
  - hard?
  - soft?
  - Which one would you like for editing an important file?
  - Later NFS versions added `intr` vs. `nointr`

What NFS RPCs are preformed?

- `cat dir/subdir/file`
  - `cat` does `open("dir/subdir/file")` , why is there not a `NFSopen("dir/subdir/file")RPC`
- `cd dir/subdir; cat file`
  - `stat RPC?`
- `echo message > file`

NFS server failure

- Client: `hard` mount - loop in kernel retrying
- Handles:
  - SunRPC - UDP (unreliable datagram protocol)
    - Request (RPC call) is lost
    - Reply (RPC response) is lost
      - Reply cache? State?
  - Server crashes and reboots
    - Reply cache wiped. How is this handled?
      - `remove, rename, link, mkdir, rmdir`

Why was the generation number in inode needed?

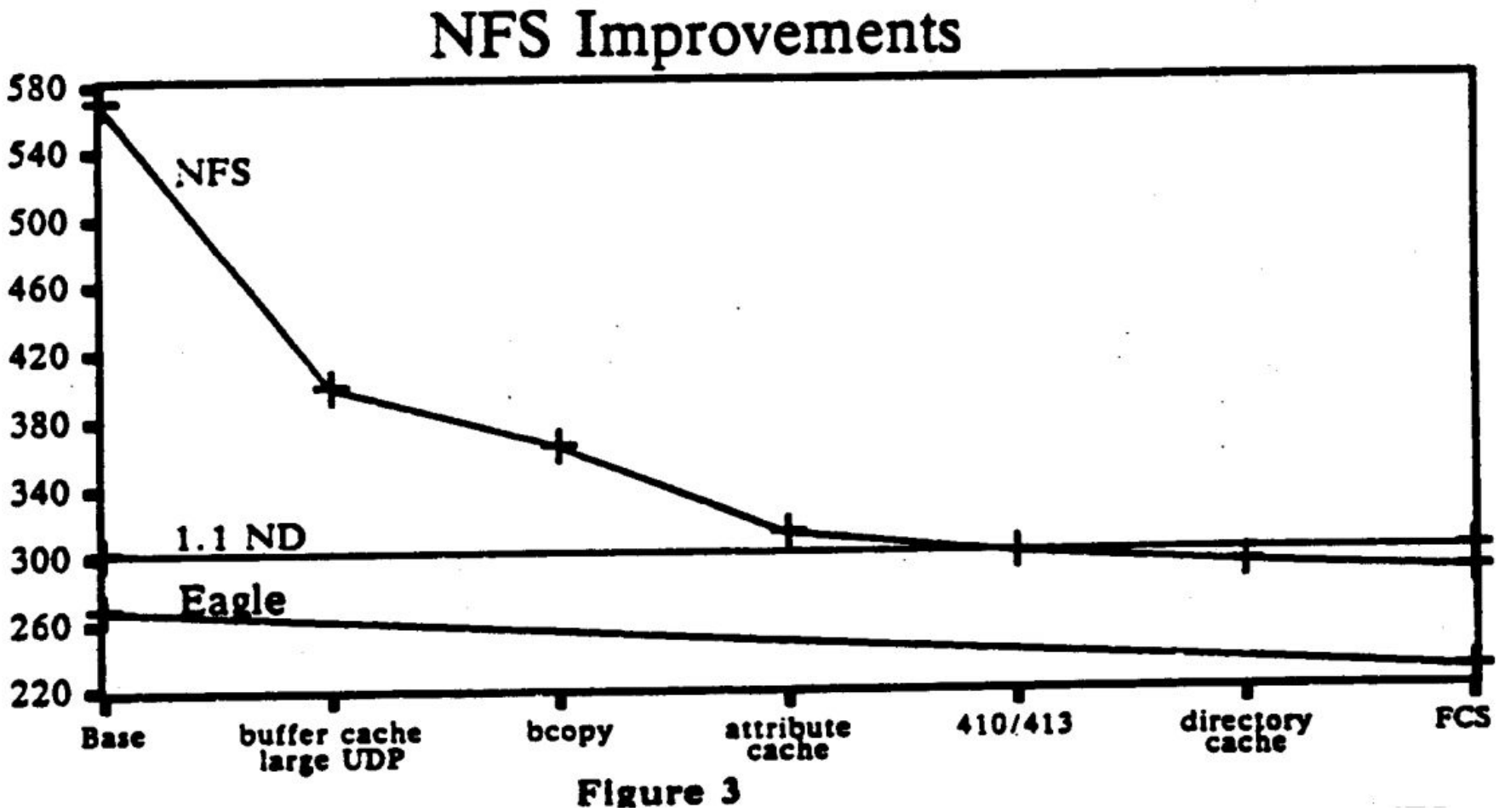
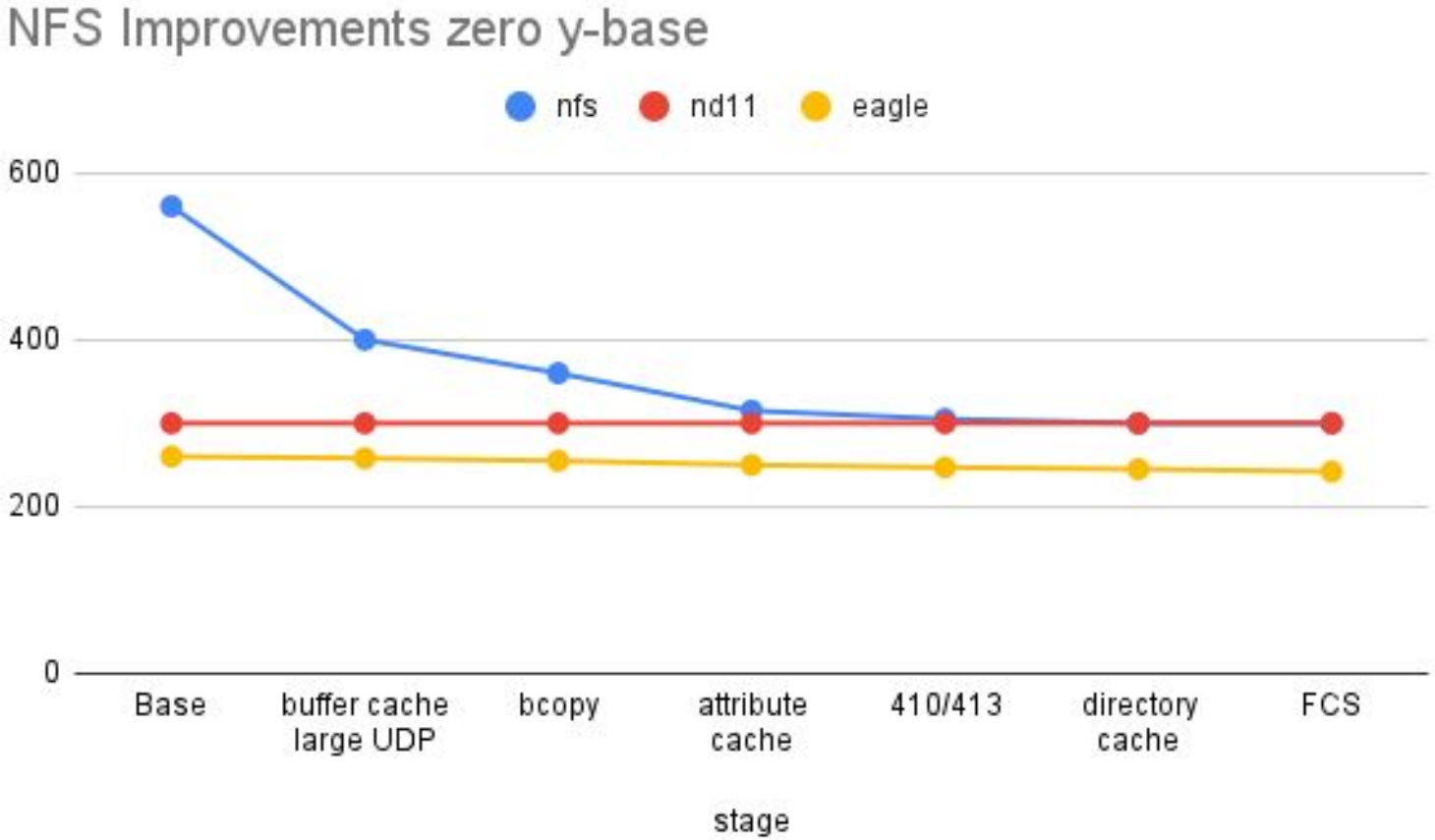

NFS Improvements
580
540
500
460
420
380
340
300
260
220
NFS
1.1 ND
Eagle
Base
buffer cache large UDP
bcopy
attribute cache
410/413
directory cache
FCS

Figure 3


NFS Improvements zero y-base
nfs
nd11
eagle
600
400
200
0
Base
buffer cache large UDP
bcopy
attribute cache
410/413
directory cache
FCS
stage

Single System Image

- NFS approximation
  - Private root file systems
    - Home directories all in NFS
  - Same mount commands on all client machines
  - Same user and group files on all client machines
- Limitations
  - Wonky consistency - need to wait before accessing file written on another machine
  - Process ids and root file system still per machine

Security implications

- Trusted computing base (TCB)

Machine & OS independence, let's ask DEC with VMS

```
DISK1:[USERS:MENDEL]FILE.TXT;1
```

- Pathnames different, had version numbers, case-insensitive, case-preserving
- Record based files vs bytes arrays
- Full access control lists (ACLs)
- Quotas
- File locking
- Little endian VAX

Did handle other Unix-like operating systems well

Leases

- Prior consistency approaches:
  - Reliable broadcast to invalidate read-only copies
  - Check for updates on every read
- Compare leases with a simple consistency mechanism
  - Centralized server tracking client accesses. Many readers, single writers
  - Convert to leases - Benefits?
- Write-through vs Write-back?
- What is false-sharing in leases?
- What happens when a client fails?

Evaluation

- Analytic model
- Trace-driven simulation

Time Skew

- Leases
  - Server ahead of client
  - Client ahead of server
- NFS
  - Modify time set from server's clock
  - Needed to patch `ranlib, ld` and `emacs` since they compared with client clock