Shared and Parallel File Systems

Ioan Raicu
Computer Science Department
Illinois Institute of Technology

CS 595
Hot Topics in Distributed Systems: Data-Intensive Computing
September 29th, 2010

Filesystems Overview

- System that permanently stores data
- Usually layered on top of a lower-level physical storage medium
- Divided into logical units called “files”
  - Addressable by a *filename* (“foo.txt”)
  - Usually supports hierarchical nesting (directories)
- A file *path* joins file & directory names into a **relative** or **absolute** address to identify a file (“/home/aaron/foo.txt”)

Shared/Parallel/Distributed Filesystems

- Support access to files on remote servers
- Must support concurrency
  - Make varying guarantees about locking, who “wins” with concurrent writes, etc...
  - Must gracefully handle dropped connections
- Can offer support for replication and local caching
- Different implementations sit in different places on complexity/feature scale

Timeline

- 1980~1990: NFS
- ~2000: PVFS
- ~2002: GPFS
- ~2003: Lustre
- ~2003: GFS
- ~2006: Sector
- ~2007: HDFS

NFS: Network File System

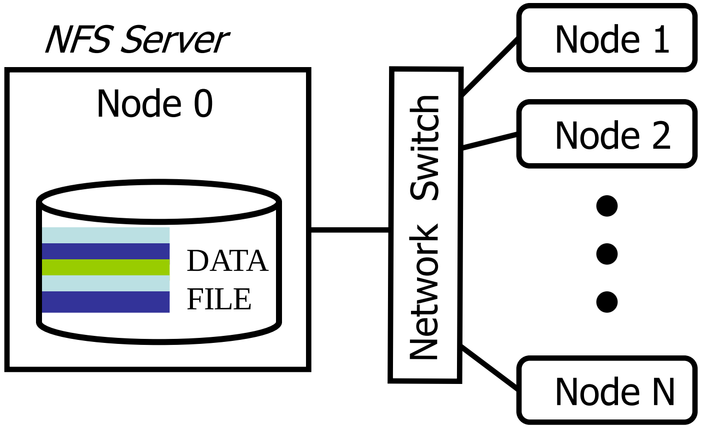
- First developed in 1980s by Sun
- Presented with standard UNIX FS interface
- Network drives are *mounted* into local directory hierarchy

NFS Protocol

- Initially completely stateless
  - Operated over UDP; did not use TCP streams
  - File locking, etc., implemented in higher-level protocols
- Modern implementations use TCP/IP & stateful protocols

NFS Architecture

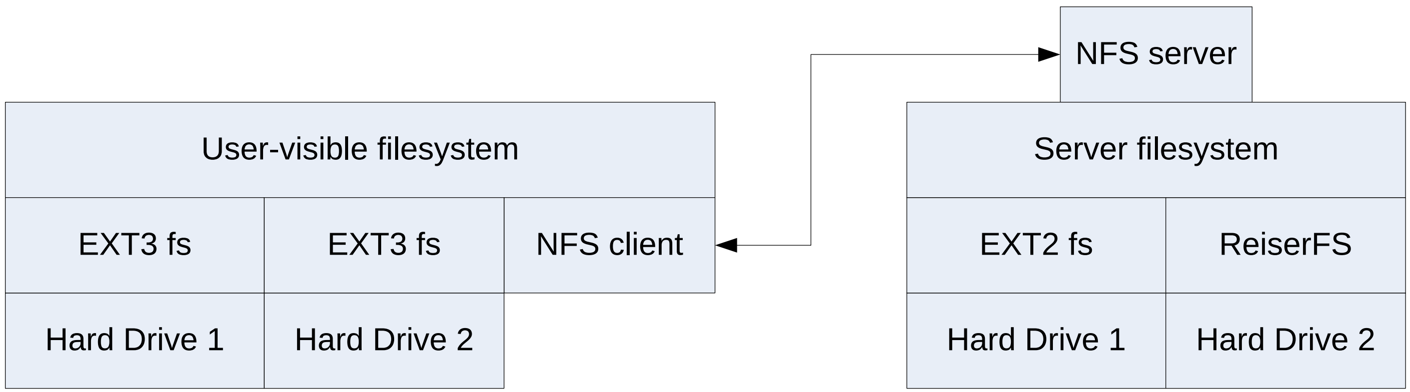
- Client/server system
- Single server for files



Each cluster node has dual-processor Pentium Linux, HD, lots of memory

NFS: Server-side Implementation

- NFS defines a *virtual file system*
  - Does not actually manage local disk layout on server
- Server instantiates NFS volume on top of local file system
  - Local hard drives managed by concrete file systems (EXT, ReiserFS, ...)
  - Other networked FS's mounted in by...?



NFS Locking

- NFS v4 supports stateful locking of files
  - Clients inform server of intent to lock
  - Server can notify clients of outstanding lock requests
  - Locking is lease-based: clients must continually renew locks before a timeout
  - Loss of contact with server abandons locks

NFS Client Caching

- NFS Clients are allowed to cache copies of remote files for subsequent accesses
- Supports *close-to-open* cache consistency
  - When client A closes a file, its contents are synchronized with the master, and timestamp is changed
  - When client B opens the file, it checks that local timestamp agrees with server timestamp. If not, it discards local copy.
  - Concurrent reader/writers must use flags to disable caching

NFS: Tradeoffs

- NFS Volume managed by single server
  - Higher load on central server
  - Simplifies coherency protocols
- Full POSIX system means it “drops in” very easily, but isn’t “great” for any specific need

PVFS Overview

- NFS not sufficient for high-performance computing workloads
- At the time, other solutions either non-existent, or did not run in Linux clusters
  - GPFS (proprietary on some IBM machines)
  - Lustre (not yet)
  - GFS (proprietary to Google)

PVFS Access

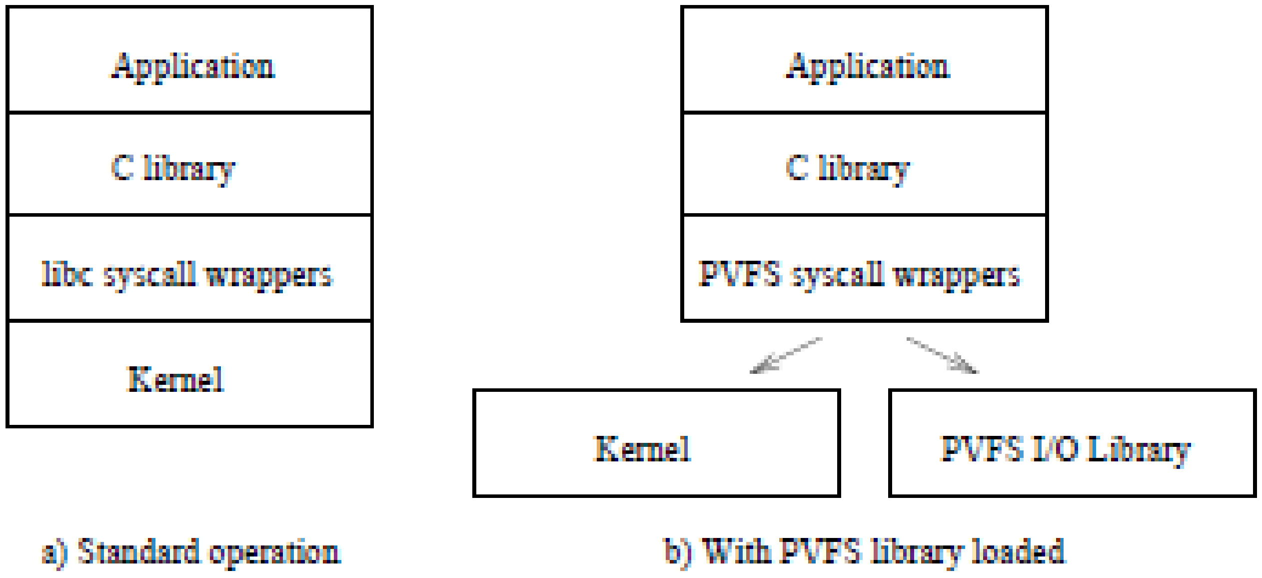
- Native PVFS Library
  - User space implementation
- Trapping I/O System calls
  - Allows applications to run without recompiling
  - Has limitations related to multi-process applications (e.g. exec causes file descriptor state to be lost)
  - Also requires high maintenance
- VFS Kernel Module
  - A module specific for PVFS, similar to NFS module

Native PVFS API example

```
#include <pvfs.h>

int main() {
int fd, bytes;
   fd=pvfs_open(fn,O_RDONLY,0,NULL,NULL);
   ...
   pvfs_lseek(fd, offset, SEEK_SET);
   ...
   bytes_read = pvfs_read(fd, buf_ptr, bytes);
   ...
   pvfs_close(fd);
}
```

Trapping System Calls



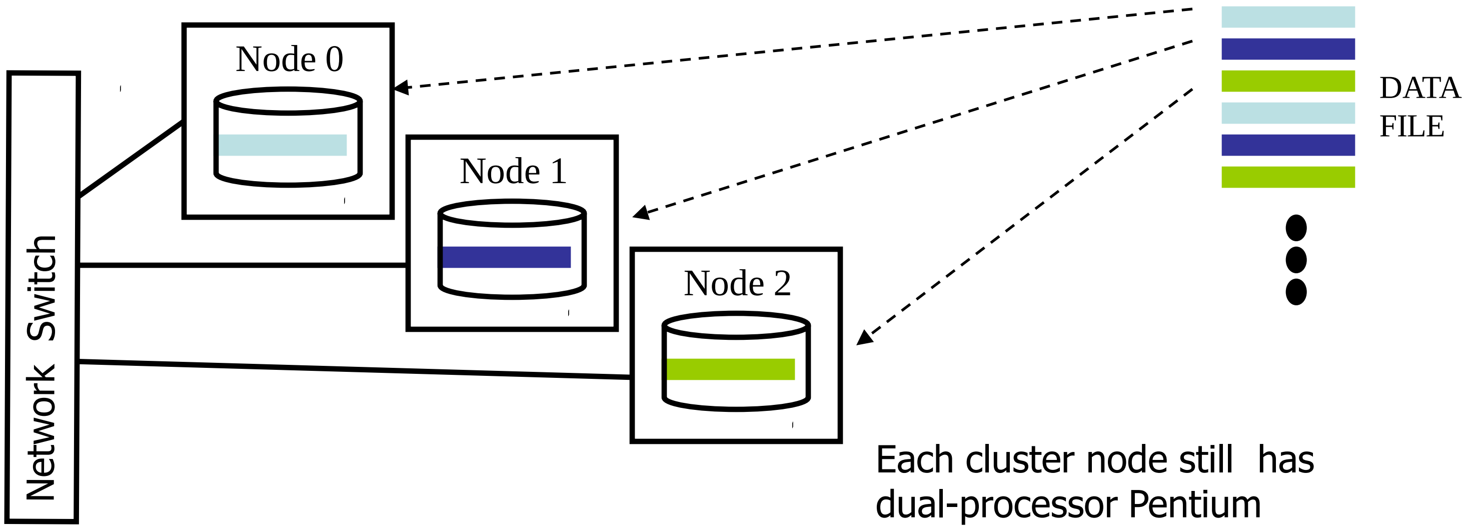
PVFS Architecture

- One node is a manager node
  - Maintains metadata information for files

- Configuration and usage options include:
  - Size of stripe
  - Number of I/O servers
  - Which nodes serve as I/O servers
  - Native PVFS API vs. UNIX/POSIX API

PVFS Architecture

- Also a client/server system
- Many servers for each file
- Fixed sized stripes in round-robin fashion



Each cluster node still has dual-processor Pentium Linux, HD, lots of memory

GPFS Overview

- GPFS had been used for years on IBM machines
- GPFS has been used on some of the largest supercomputers, including Linux-based ones
- GPFS aims for POSIX access semantic in a parallel file system
- All nodes have the same view
- Use distributed locking protocols

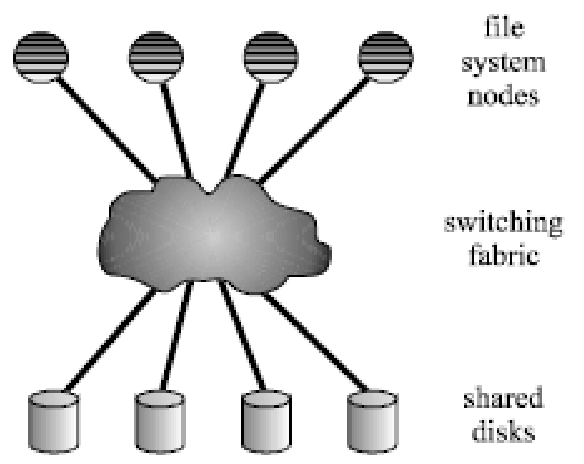


Figure 1: Shared Disk Environment

GPFS Details

- Parallel data and metadata access
- Data striping across disks
- General Large File System Issues
  - Data stripping and allocation, pre-fetch, and write-behind
  - Large directory support
  - Logging and recovery

GPFS Managing Consistency

- Locking management
  - Distributed locking
  - Centralized management
- GPFS distributed lock manager
- Parallel data access
  - Byte range locks
- Synchronizing access to file metadata
- Allocation maps
  - Managing free space
- Centralized token manager scaling

GPFS Fault Tolerance

- Node failures
  - Use recovery logs from shared disks
- Communication failures
  - Heartbeat messages
- Disk failures
  - RAID
  - Replication

Lustre Overview

- Also has a distributed lock manager
  - But more limited than that of GPFS
  - Intent locking
    - Switch between different strategies based on concurrency level
- Object-based vs. Block-based
  - Object-based protocols can help in locking and allocation of metadata
  - Lustre is backwards compatible with block-based storage
- Client caching metadata

Questions

?