Distributed File Systems

CS239
Advanced Operating Systems
April 16, 1997
Dr. Richard Guy

Lecture Source Materials

Goscinski, Ch. 13: **File Service**, p. 705-759.

Tanenbaum, Ch. 5: **Distributed File Systems**, p. 245-288.

Singhal, Ch. 9: **Distributed File Systems**, p. 199-235.

Guy, et al: Architecture of the Ficus DFS.

Outline

- Introduction to DFS
- Design Goals
- Architecture
- Implementation
- Trends/Future Issues
- Focus on Replicated File Systems

Introduction

- Primary goal: offer filing services of at least the same generality as those found in centralized file systems. (Goscinski, p. 725)

Design Goals for a DFS

- Application diversity
- Simple semantics
  - simple interface
  - few commands
  - understandable concept of “file”
- Match user requirements
  - enough to support
  - don’t pay for extras

more design goals...

- Consistent naming from anywhere
- Location transparency
- Transparent access (local = remote)
- Data consistency
- Consistent authentication (anywhere)
- Information protection
- User mobility

and more design goals...

- Performance "transparent"
- Availability
- Crash recovery
- High reliability
- Scalability
- Accommodate diverse storage media

Architecture models

- centralized file system
- distributed file system

Centralized FS architecture

DIRECTORY SERVICE
Directory module | Access control module

FILE SERVICE
File module | File access module

BLOCK (DISK) SERVICE
Block module | Device module

CFS modules

- Device: disk I/O and buffering
- Block: access and allocate disk blocks
- File access: read/write file data/attributes
- File: relate file identifiers to particular files
- Access control: check rights for req'd oper.
- Directory: map file text name to file ID.

Distributed FS architecture

NAMING/DIRECTORY SERVICE
Directory | Access control
Placement | Location | Relocation

REPLICATION SERVICE
Data consistency and multiple copy update

TRANSACTION SERVICE
Concurrency ctl

FILE SERVICE
File | File access | Attribute access

BLOCK (DISK) SERVICE
Block | Device | Caching

DFS modules

- Device: disk I/O and buffering
- Block: access and allocate disk blocks
- Caching: results of disk ops and remote ops

DFS modules

- File access: read/write file data/attributes
- File: relate file identifiers to particular files
- Attribute access: read/write file attributes

DFS modules, cont.

- Trans. ops: basic mech. for atomic ops
- Recovery: post-failure recovery mechanism
- Conc. ctl: preserve consistency/atomicity

- Data consistency/multiple copy update: preserve semantics and improve file access performance using replicated file data

DFS modules, cont.

- Access control: check rights for req'd oper.
- Directory: map file text name to file ID
- Placement: select server to hold new file
- Location: find server managing file
- Relocation: migrate file for performance

Disk Service

- File storage
  - data
  - attributes
- Caching
- Unique file identifiers
- File location
- Stable storage

File service

- Mutable and immutable files
  - history of immutable versions
- Operations on data
  - create, delete
  - open, close
  - read, write
- File types

- Operations on attributes
  - read_attribute
  - write_attribute
- Remote access vs. upload/download

Transaction service

- System failure protection
- File sharing semantics
- Transaction tools

File sharing semantics

- unix
- session
- immutable
- transaction

Transaction tools

- begin/end/abort transaction
- failure recovery
- concurrency control
  - locking
  - optimistic with abort/restart
  - timestamps with abort/restart

Replication service

- General approach
  - Master/slave
  - Distributed update control
- Unit of replication

Naming/directory service

- Naming schemes
  - simple hierarchy
  - mounting remote on local (a la NFS)
  - single global name space (a la LOCUS/Sprite)
- Naming transparency
- Protection/Authentication

Implementation

- from Tanenbaum, p.256-272
- File usage observations
- System structure
- Caching
- Replication
- Lessons learned

File Usage (c. 1981, Unix timesharing)

- Most files are small (<10K)
- Reading >> writing
- Read/write is sequential; random access rare
- Most file have short lifetime
- File sharing is unusual (non-system)
- Average process uses few files
- Distinct file classes with different properties

System structure

- Client/server vs peer
- File server distinct from directory server
- Statelessness...

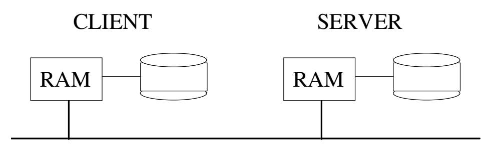
Stateless vs stateful

Stateless Server Advantages	Stateful Server Advantages
Fault tolerance	Shorter request messages
No open/close needed	Better performance
No server space for tables	Readahead possible
No limits on |open files|	Idempotency easier
Client crashes irrelevant	File locking possible

Caching

- where to cache…
- how to cache: cache consistency

Where to cache



Cache consistency

- write through
- delayed write
- write on close
- centralized control

Replication

- update protocols
  - primary copy
  - voting
  - optimistic
- File sharing semantics
  - unix
  - session
  - transaction

Lessons learned

- Workstations have cycles to burn
- Cache whenever possible
- Exploit the usage properties (!?)
- Minimize systemwide knowledge/change
- Trust the fewest possible entities
- Batch work where possible

Six trends/issues in DFSs

- from Tanenbaum (p. 279-285)
- hardware
  - price dropping on old hardware (no disks??)
  - optical disks (slow, huge, random)
  - very fast fiber optics

more trends...

- scalability
  - server bottlenecks--> multiple servers/partitions
  - broadcasts
  - nonlinear management techniques
  - strict semantics hard to support
  - large file system tree hopeless?

and more...

- wide-area networking
  - scale
  - heterogeneity
  - application shifts
  - net bandwidth too low

and more

- mobile users
- fault tolerance
- multimedia

Case studies of Replicated FSs

- Coda
- Ficus
- Rumor

Coda

- Major goals
- Naming and location
- Caching and replication
- Cache coherence
- Disconnected operation
- Replica management

Major goals

- Scalability
- Constant data availability
  - replicated servers
  - client caching
- Graceful integration for portables

Naming and location

- long unique file identifier
- volume location database

Caching and replication

- Volume replication database
- Volume storage group
- Client caching with callback
- Optimistic update strategy

Cache coherence

- Client cache manager must notice
  - enlargement of accessible VSG
  - shrinkage of accessible VSG
  - lost callbacks
- Client is in one of three modes
  - connected
  - discnnected
  - reintegrating

Disconnected operation

- allows uncoordinated update
- detect inconsistencies/propagate updates
- detect conflicts

Replica management

- uses Locus/Ficus version vectors
  - essentially a logical update vector timestamp
  - detects single and concurrent upates

Ficus

- Goals
- Design overview

Goals

- Scalability via peer-to-peer solutions
- Transparency
  - location
  - name
  - replica
- Automatic file/directory update management
  - Automatic directory conflict resolution
  - Limited automatic file conflict resolution

Design Overview

- Stackable layers architecture
  - exploits Sun's VFS concept/implementation
  - runs inside kernel, or main kernel/user-level server
    standard UFS as storage
- Implements replication services as layer(s)
  -
  - standard NFS as remote access substrate
  - base replication service is layer pair above UFS/NFS

- Naming/directory services at Logical (upper) layer
- Storage management at Physical (lower) layer
  - basic design "flaw" example

Rumor

- Ficus services via user-level tools

Summary