

Frangipani: A Scalable Distributed File System by  
Chandramohan A. Thekkath, Timothy Mann, and Edward K. Lee  
Digital Equipment Corporation, 1997

AND

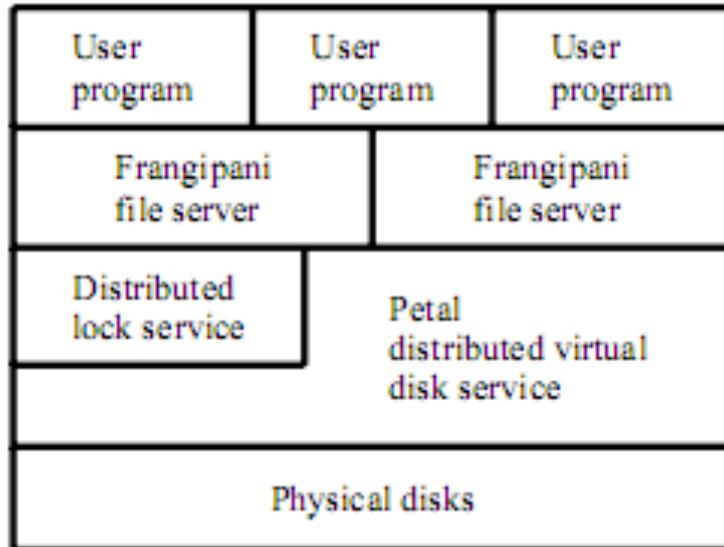
FARSITE: Federated, Available, and Reliable Storage  
for an Incompletely Trusted Environment by  
Atul Adya, William J. Bolosky, Miguel Castro, Gerald Cermak, Ronnie  
Chaiken, John R. Douceur, Jon Howell, Jacob R. Lorch, Marvin  
Theimer, and Roger P. Wattenhofer  
Microsoft Research, 2002

Presented by Fatih Kaya

# Frangipani

- A distributed file system
  - Developed for a cluster of machines
  - Secure environment required
- Simple design that deploys a two layered model
  - File system over distributed virtual disk (Patel)
- Easily manageable
  - Consistent view for all users
  - Straightforward user addition
  - Online backup
  - Recovery with minimum operator intervention

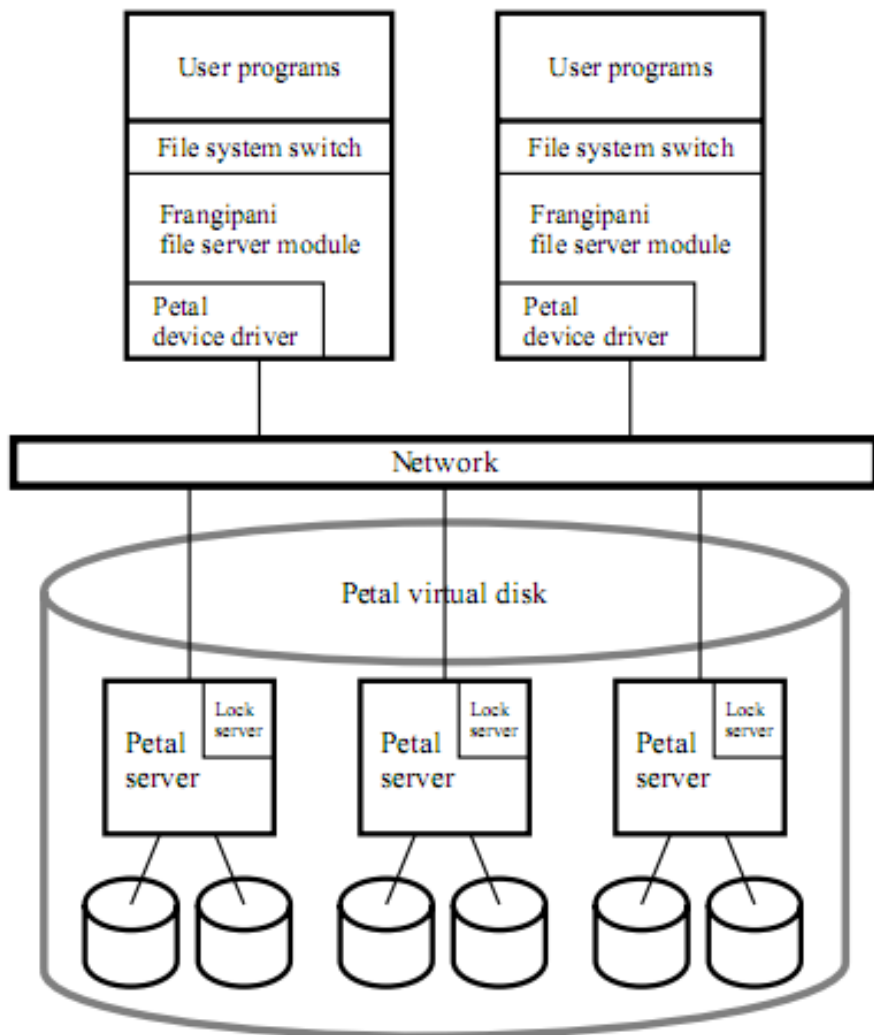
# Layered Model and Patel



Distributed storage system  
Pool of disks  
Replication  
Snapshots

No files  
No synchronization  
No meta-data

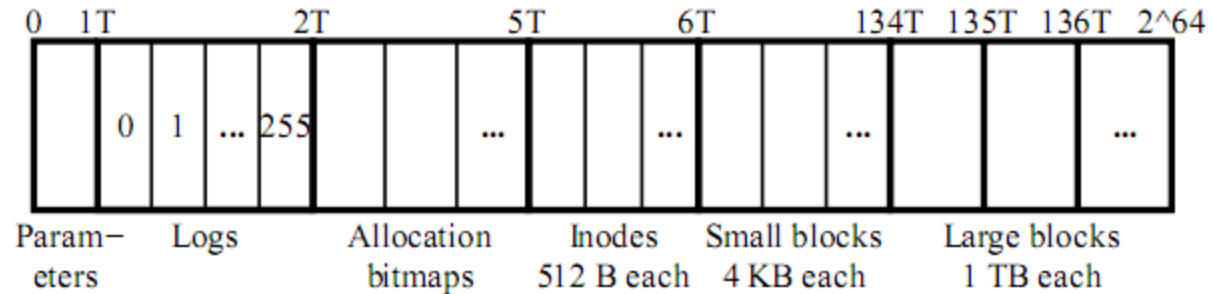
# System Structure



Components and their roles

- \* Frangipani file server module  
implemented in kernel  
communicates with lock servers and Petal
- \* Petal device driver  
interface to the Petal service
- \* Lock server  
provides multiple-reader, single-writer locks
- \* Petal server

# Disk Layout



$2^{64}$  bytes of address space

64 KB chunks for commit/decommit

commits occur only when data is written

6 regions

shared configuration parameters, logs,  
allocation bitmaps, inodes, small data blocks,  
large data blocks

# Logging and Recovery

- Write-ahead redo logging
  - metadata only
  - private log for each Frangipani server on Petal
- Crash detection
  - by client
  - by lock service
- Recovery from logs
- No high-level semantic guarantees
  - improve metadata update performance

# Lock Service

- Multiple-reader, single-writer lock mechanism
  - Read lock allows a server to read data and cache it.
  - Write lock allows a server to read or write data .
  - When a write lock is downgraded or released, the server must flush its dirty data to disk.
- Locks are coarse-grained
  - Lock for each logical segments
    - Each file, directory or symbolic link is one segment.
  - Protects entire file or directory

# Lock Service

- Avoiding deadlock by globally ordering these locks (locks for the same file).
- Acquiring locks in two phases:
  - A server determine what locks it needs. Which file or directory?  
Read lock or write lock?
  - The server sorts the locks by inode address and acquires each lock in turn.
    - Then checks whether any objects identified in phase one were modified while their locks were released. If so, the server releases locks and loops back to phase one.

# Lock Service

- The lock service deal with client failure using **leases**
  - Client obtain a lease together with the lock. If the lease expires, the client either renews the lease or the lock will become invalid.
- Three different implementations:
  - 1) A single, centralized server. All lock states are kept in the server volatile memory. Not fault tolerant.
  - 2) Store the lock state on a Petal virtual disk, so in case of server crash, the lock state can be recovered. Poor performance.
  - 3) A set of mutually cooperating lock servers, and a clerk module linked into each Frangipani server. Result: fully distributed for fault tolerance and scalable performance.

# Administration and Backup

- Adding a new Frangipani server
  - Assign Petal virtual disk
  - Assign lock servers
- Removing a server
  - Do nothing (lock servers and recovery service will take care)
- Backup
  - Take a Petal snapshot and copy to tape

# Performance

- Equipment
  - 7 333 Mhz DEC Alpha computers as Petal servers
  - Each has 9 DIGITAL RZ29 disks, 4.3 GB each
  - Connected to 24 port ATM switch 155 Mbit/s link
- Use of non-volatile ram (NVRAM)
  - Placed in between disks and Petal servers
  - Decreases disk latency
- Comparison with Digital Unix AdvFS
  - Faster than other alternatives like UFS
  - Uses a write-ahead log like Frangipani
  - AdvFS runs on 225 Mhz DEC Alpha with 192 MB RAM and 8 DIGITAL RZ29 disks connected with a 10MB/s bus. Comparable to the built Patel server.
  - Frangipani client uses the same machine without local disks.

# Single Machine Performance

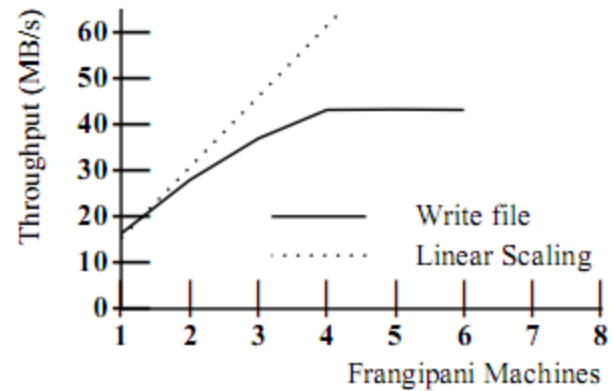
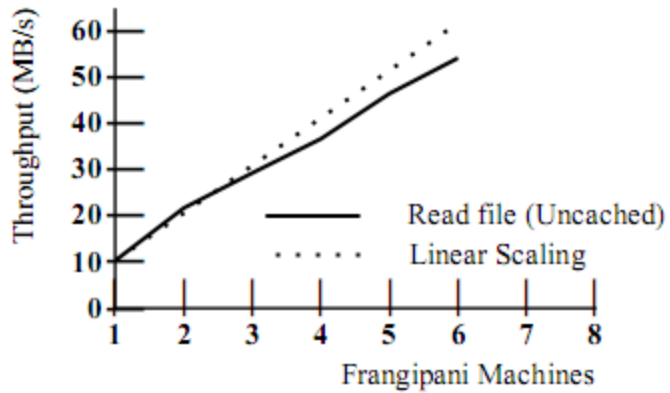
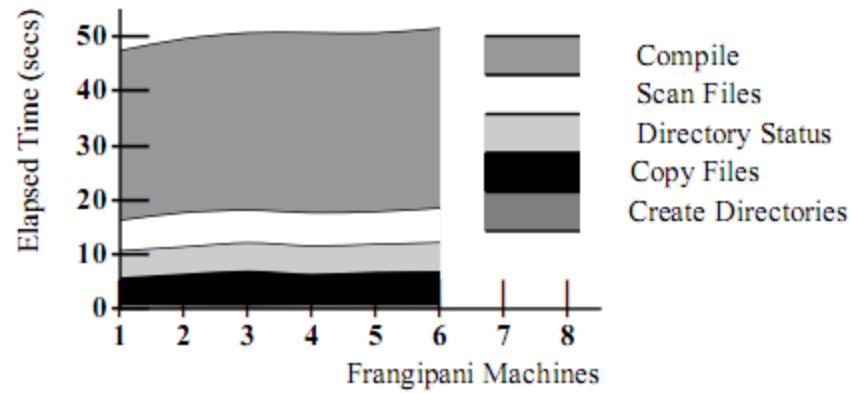
Phase	Description	AdvFS		Frangipani	
		Raw	NVR	Raw	NVR
1	Create Directories	0.69	0.66	0.52	0.51
2	Copy Files	4.3	4.3	5.8	4.6
3	Directory Status	4.7	4.4	2.6	2.5
4	Scan Files	4.8	4.8	3.0	2.8
5	Compile	27.8	27.7	31.8	27.8

Modified Andrew Benchmark with unmount operations

	Throughput (MB/s)		CPU Utilization	
	Frangipani	AdvFS	Frangipani	AdvFS
Write	15.3	13.3	42%	80%
Read	10.3	13.2	25%	50%

Frangipani Throughput and CPU Utilization

# Scaling



# Conclusion

- Pros
  - Simplicity in design
  - Two layered approach brings modularity
  - Has comparable performance to commercial file systems
- Cons
  - Not portable
  - Strict security assumptions
  - Some design choices lack formal justification (i.e. file system parameters in disk layout)

**FARSITE: Federated, Available, and Reliable Storage for an  
Incompletely Trusted Environment by**

Atul Adya, William J. Bolosky, Miguel Castro, Gerald Cermak, Ronnie Chaiken, John R. Douceur, Jon Howell, Jacob R. Lorch, Marvin Theimer, and Roger P. Wattenhofer

Adapted from OSDI02 slides

# FARSITE

- A serverless distributed file system
- Developed for cooperating but not completely trusted clients
- Serverless
  - Runs entirely on client machines.
  - Logically centralized but physically distributed.
- Servers are more powerful, more secured, better maintained, but...
  - Reliant on system administrators
  - Expensive (High-performance I/O, RAID disk, special rooms)
  - Centralized points of failure
  - High-value targets for attacks
- Ensure user privacy and data integrity
- Scalable and efficient
- Easy management and self-configuration

# Target Environment

- Large university or company
  - High-bandwidth, low-latency network
  - Clients are cooperative but not completely trusted
- Rough scale:
  - $10^5$  total machines,  $10^{10}$  total files,  $10^{15}$  total bytes
- Machine availability
  - Lower than dedicated servers, higher than Internet hosts.
  - Uncorrelated machine downtimes
- Workload: High access locality, low persistent update rates, usually sequential but rarely concurrent read/write sharing
- Small but significant fraction of malicious users
- No user-sensitive data persist beyond user logoff or reboot

# Enabling Technologies

- Availability: enough disk space for replicas
  - Low disk costs
  - Unused disk capacity
  - Duplicate files: ~50% space savings
- Privacy: fast crypto
  - Symmetric encryption: 72 MB/sec
  - One-way hashing bandwidth: 53 MB/sec
  - Disk sequential I/O bandwidth: 32 MB/sec
  - Computing RSA signature: 6.5 msec < Rotation time for a 7200-RPM disk
  - Can use strong cryptography for security

# System Architecture

- File system construct:
  - Hierarchical directory namespace
  - Namespace - Logical repository of files (a directory).
  - Namespace root – A virtual file server
  - Created by administrator
- Machine roles:
  - Client – Directly interacts with users
  - Directory group member – Manages file metadata
  - File host – Stores encrypted replicas of data files

# System Architecture (Cont...)

- Directory Group:
  - Manages file metadata for a *namespace*
  - Choice of machines for namespace root assigned by administrator
  - Subtree of namespace can be delegated to subgroup under heavy load.
- Separate data and metadata:
  - Use Byzantine agreement protocol in directory group to protect metadata against malicious clients
  - Replicate and encrypt data in file hosts
  - Stores cryptographically secure hash of data in directory group for validation

# System Architecture (Cont...)

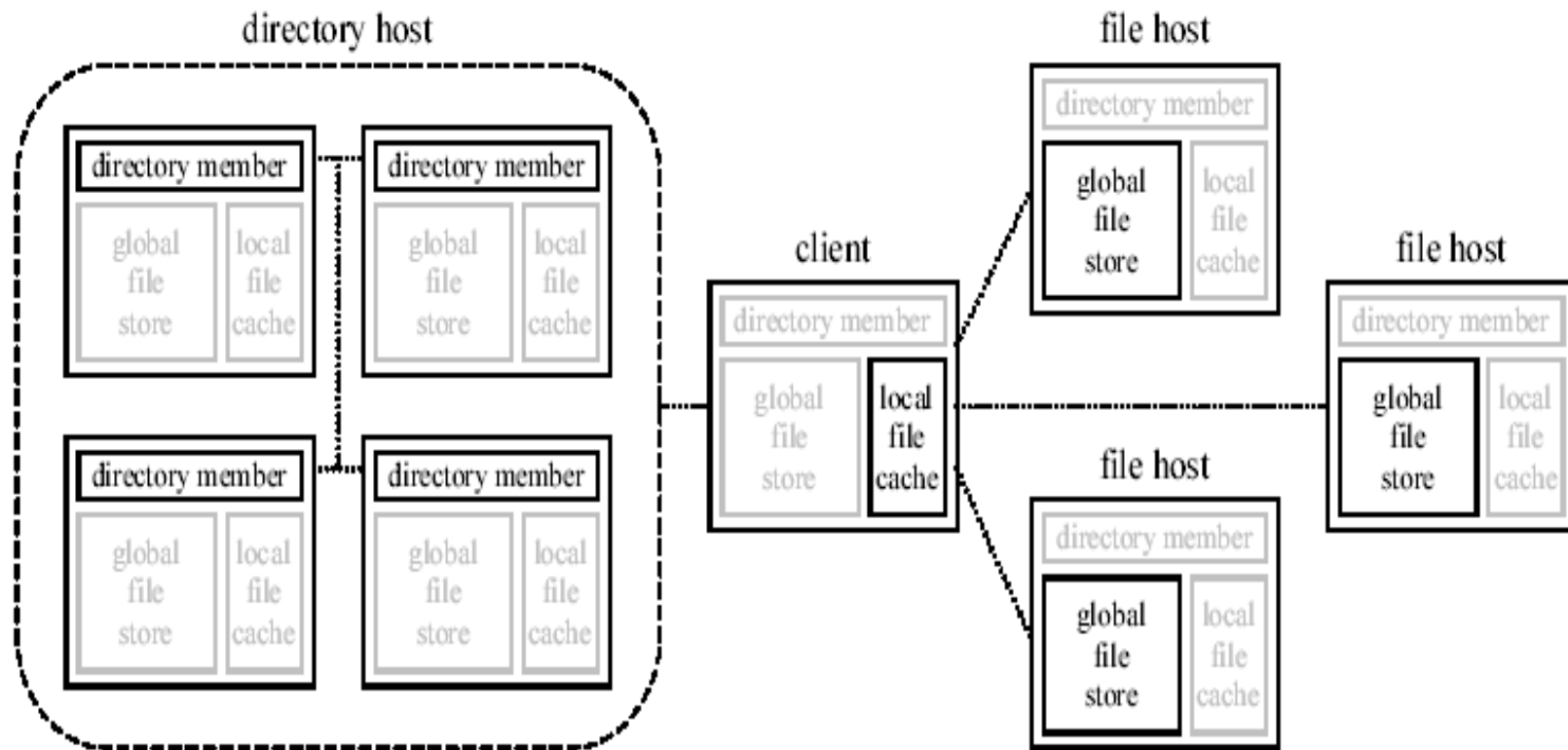


Figure 1. Portion of Farsite system architecture from one client's perspective

# Certificates

- Machine Certificates
  - Associate a machine with own public keys.
  - Establishing the validity of machine.
- Namespace certificates
  - Associate namespace roots to set of managing machines.
  - Administrator grants root certificate.
- User certificates
  - Associate users to their public keys
  - For read/write access control to files.
- Certification authorities (CAs)

# Write Operation

1. Client sends updated hash of file contents to directory group.
2. Directory group verifies user permission to write to the file using user certificate.
3. Once verified, directory group directs file hosts to retrieve new data from clients.
4. Leases on existing clients may be recalled by directory group to satisfy new update request.

# Read Operation

1. Send message to directory group.
2. Directory group proves its authority (recursively to root) using *namespace certificates*.
3. Directory Group grants client:
  - *Lease* on file for a period for local access.
  - One-way hash of file contents for validation
  - List of file hosts storing data.
4. Client retrieves replica from a file host:
  - Validates content using one-way hash
  - Decrypts using private key.
  - Works on local cached copy for lease period.

# File System Features

- Reliability and Availability
- Security
- Consistency
- Scalability
- Efficiency
- Manageability

# Reliability and Availability

- Goals:
  - Long-term persistence
  - Immediate accessibility of file data during request.
- Directories/meta-data maintained via BFT
  - Needs replicas to protect against malicious nodes.
  - $3f + 1$  replicas for tolerating  $f$  faults
- Files replicated via simple replication
  - File replicas to ensure high degree of availability.
  - $f + 1$  replicas to tolerate  $f$  faults
- Data migration & repair
  - Away from unavailable machines
  - Swap machine locations between high/low availability file replicas.

# Security

- Access control
  - Directory groups maintain an access control list (ACL) of public keys of all authorized writers.
- Privacy
  - During file creation, client generates a random file key used to encrypt the file.
  - File key is encrypted with public keys of all authorized readers of the file. Encrypted file keys are stored with file.

# Durability

- FARSITE does not update at once all replicas of a file:
  - Would be too slow.
  - Use a background delayed update mechanism.
- Updates to metadata are written in compressed logs on client's local disk.
- Log is sent to directory group periodically, and at end-of-lease.

# Consistency

- Data consistency
  - Content leases with expiration - Read/write or read-only
  - Single-writer multi-reader semantics
  - Request for conflicting lease triggers a recall of lease by directory group.
  - Single-client serialization of concurrent non-reads access.
  - Inappropriate for large-scale write sharing.
- Other leases
  - Name leases – Allows clients to modify private namespace regions without contacting directory group.
  - Mode leases – Windows File-sharing semantics (read, write, delete, exclude-read, exclude-write, exclude-delete)
  - Access leases – Windows deletion semantics (public, protected, private)

# Efficiency

- Space Efficiency
  - Users share identical files and applications.
  - Detect similar using file-content hashes.
  - Half of all occupied disk space can be reclaimed.
- Time Efficiency (Performance)
  - Local caches (data and pathnames)
  - Delay between file creation/updates and replica creation.
  - Limit leases per file.

# Manageability

- Local-Machine Administrations
  - Removing an old disk is a special case of hardware failure.
  - Supports different versions of software
  - Argues that backup processes are unnecessary Replicas are present anyway.
    - Versioning systems are more effective for archival purposes.

# Evaluation

- Method
  - 5 FARSITE machines
  - 1-hour representative trace from 3-week traces
  - Played at real time (~ 450,000 operations)
  - Measured file operations latency
- Results (untuned code):
  - 20% faster than Microsoft CIFS
  - 5.5x slower than NTFS

# Conclusion

- Scalable, decentralized, network file system.
- Use loosely coupled, unsecured and unreliable machines to establish a secured and reliable virtual file server.
- Synthesis of known techniques: Replication, BFT, leases, namespaces, client caching, lazy updates, cryptography, certificates, etc.