G22.3250-001

Farsite: A Serverless File System

Robert Grimm
New York University

Altogether Now: The Three Questions

- What is the problem?
- What is new or different or notable?
- What are the contributions and limitations?

Distributed File Systems Take Two (late 90s and 00s)

- Goal: Provide uniform file system access across several workstations
  - Single, possibly large organization
  - Computers connected by local network (not Internet)
- General strategy: Serverless file systems
  - Server-based FS's are well administered, have higher quality, more redundancy, and greater physical security
    - But they also are *expensive*!
      - Equipment, physical plant, personnel
  - Alternative: Use already existing workstations
    - Need to reconcile security, reliability, and performance

Farsite Design Assumptions

- High-bandwidth, low-latency network
  - Topology can be ignored
- Majority of machines up and running a majority of the time
  - Generally uncorrelated downtimes
- Small fraction of users malicious
  - Try to destroy or corrupt data or metadata
- Large fraction of users opportunistic
  - Try to read unauthorized data or metadata

Enabling Technology Trends

- Remember: Reconcile security and performance
- Increase in unused disk capacity → Use replication
  - Microsoft studies: Discs grow faster than Windows/Office
    - 1998: 49% of disc space unused
    - 1999: 50% of disc space unused
    - 2000: 58% of disc space unused
- Decrease in cost of cryptographic operations → Provide privacy and security
  - 72 MB/s symmetric encryption bandwidth
  - 53 MB/s one-way hashing bandwidth
  - 32 MB/s sequential I/O bandwidth

Trust and Certification

- Based on public-key certificates
  - Issued by certificate authority or its delegates
    - Important for manageability (e.g., HR and IT departments)
- Structured into three types of certificates
  - Namespace certificates
    - Map FS namespace to set of machines managing metadata
  - User certificates
    - Map user to public key for access control
  - Machine certificates
    - Map machine to public key for establishing physical uniqueness

System Architecture

- Basic system
  - Clients interact with user
  - Directory groups manage file information
    - Use Byzantine fault-tolerance [Castro & Liskov '99]
      - $3f + 1$ replicated state machines tolerate up to f malicious nodes
- Problems with basic system
  - Performance of Byzantine fault-tolerance protocol
  - No privacy of data
  - No access control
  - Large storage requirements (*replicated* state machines!)

System Architecture (cont.)

- Improvements for actual system
  - Cache (and lease) files on clients and delay updates
  - Encrypt data with public keys of all readers
    - Provides privacy and read-access control
  - Check write-access permissions in directory group
  - Store opaque file data (but not metadata) on *file hosts*
    - Use cryptographic hash in metadata to ensure integrity
  - Delegate portions of namespace to different directory groups
- Semantic differences from NTFS
  - Limits on concurrent writers and readers
  - No locking for rename operations

Diggin' Deeper

- Reliability and availability
- Security
- Durability
- Consistency
- Scalability
- Efficiency
- Manageability

Reliability and Availability

- Main mechanism: Replication
  - Byzantine for metadata, regular for file data
- Challenge: Migrate replicas in face of failures
  - Aggressive directory migration
    - After all, need to access metadata to even get to file data
    - Random selection of new group members after short downtime
    - Bias towards high-availability machines
  - Balanced file host migration
    - Swap locations of high-availability files with low-availability files
    - Simulations show 1% of replicas swapped per day

Security

- Access control
  - Based on public keys of all authorized file *writers*
  - Performed by directory group
- Privacy
  - Based on block-level, *convergent* encryption (why?)
    - One-way hash serves as key for encrypting block
    - One-way hash encrypted with symmetric file key
    - File key encrypted with public keys of all file *readers*
- Integrity
  - Ensured by hash tree over file data blocks
    - Tree stored with file, root hash stored with metadata

Durability

- Updates to metadata collected in client-side log
  - Batched to directory group
- What to do after a crash?
  - Obvious solutions are problematic
    - Private key signing of all log entries too expensive
    - Holding private key on machine opens security hole
  - Better solution: Symmetric authenticator key
    - Split into shares and distributed amongst directory group
    - Log entries signed using message authentication code
    - On crash recovery, log pushed to directory group
      - Members reconstruct authenticator key, validate entries

Consistency

- Ensured by leases
  - Content leases control file content
    - Either read/write or read-only
    - Leases, together with batched updates, recalled on demand
      - Optimization allows for serialization through single client (design only)
    - Leases also include expiration times
      - Addresses failures and disconnections
  - Name leases control directory access
    - If named entities do not exist, allow for creation of file/directory
    - If named entities do exist, allow for creation of children
    - Can be recalled on demand

Consistency (cont.)

- Ensured by leases (cont.)
  - Mode leases control Windows file-sharing semantics
    - Implemented by six types: Read, write, delete, exclude-read, exclude-write, exclude-delete
    - Can be downgraded on demand
  - Access leases control Windows deletion semantics
    - Implemented by three types: Public, protected, private
      - Public: Lease holder has file open
      - Protected: No other client gains access without permission by holder
      - Private: No other client gains access
    - Delete operation requires protected/private lease
      - Forces metadata pushback, informs directory group of pending deletion

Scalability

- Hint-based pathname translation
  - Cache pathnames and mappings to directory groups
  - Use longest-matching prefix to start search
    - On mismatch, directory group
      - Provides all delegation certificates → client moves on
      - Informs of mismatch → client checks shorter prefix
- Delayed directory-change notification
  - Windows already allows best-effort notification
  - Farsite directory groups send delayed directory listings through application-level multicast

Efficiency

- Space
  - Replication increases storage requirements
  - But measurements also show that a lot of data is duplicated
    - Convergent encryption makes it possible to detect duplicate data
    - Windows' single instance store coalesces duplicate data
- Time
  - Reflected throughout the system
    - Caching to improve availability and read performance
    - Leasing, hint-based translation reduce load and network round trips
    - Limited leases, hash trees, MAC-logging reduce overheads
    - Update delays reduce traffic

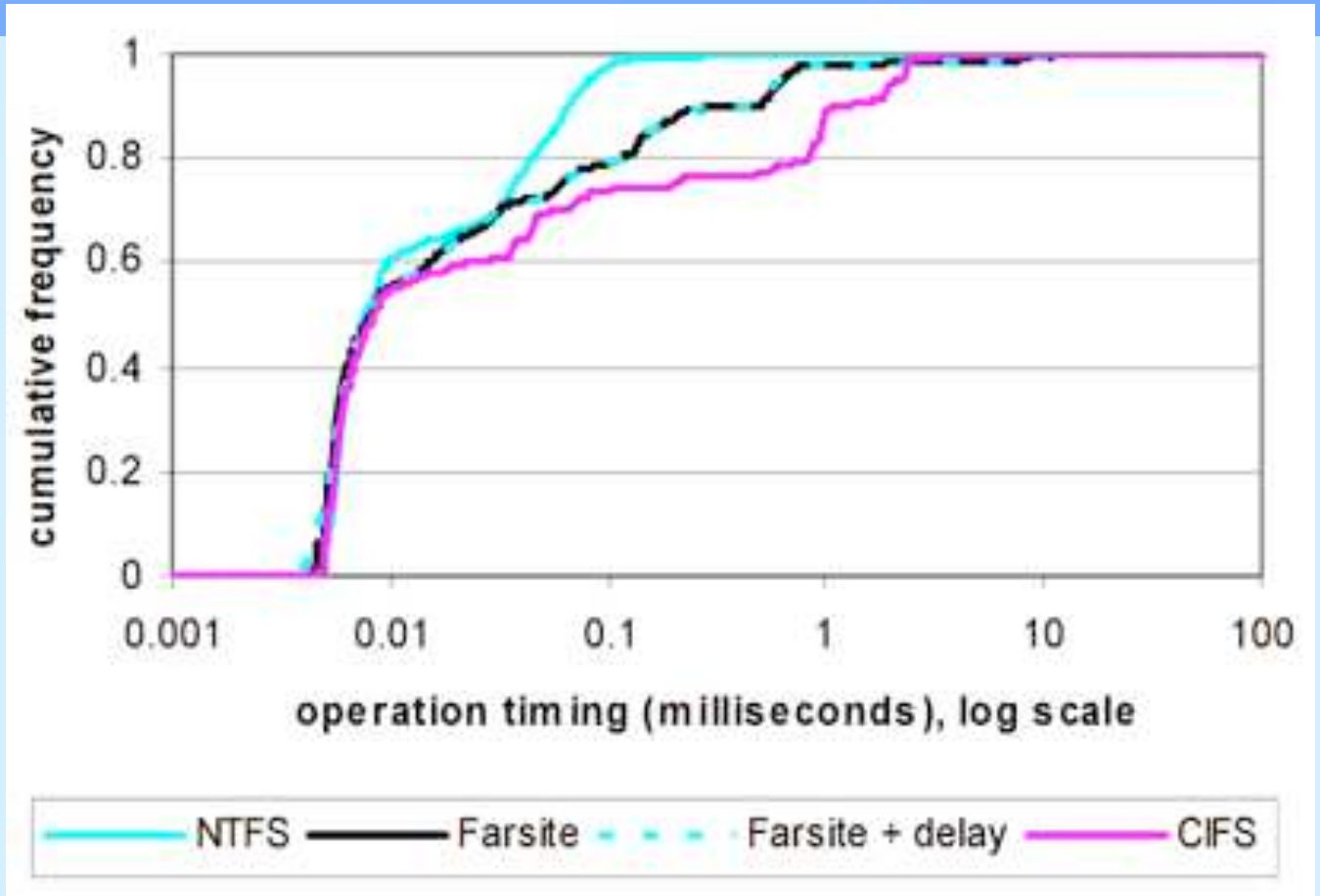
Manageability

- Local machine administration
  - Preemptive migration to improve reliability
  - Backwards-compatible versions to simplify upgrades
  - Potential backup utilities to improve performance and avoid loss of privacy
- Autonomic system operation
  - Timed Byzantine operations
    - Group member initiates time update which causes timed operations to be performed
  - Byzantine outcalls
    - Revert standard model of Byzantine fault tolerance
    - Members prod clients into performing a Byzantine operation

Phhh, That's a Lot of Details

- But how well does it work?
  - Windows-based implementation
    - Kernel-level driver & user-level daemon
    - NTFS for local storage
    - CIFS for remote data transfer
    - Encrypted TCP for control channel
  - Analysis for 10^5 machines
    - Certification: $2x10^5$ certificates per month, less than one CPU hour
    - Direct load on directory groups
      - Limited by leases per file, directory-change notifications
    - Indirect load on directory groups
      - 1 year lifetime → 300 new machines/day → trivial load on root group

Measured Performance Based on Trace Replay, One User



- What do we conclude from this graph?

What Do You Think?