Remote File Systems

- 14A Remote Data - Architectures
- 14B Remote Data - Security

Remote Data Access: Goals

- Transparency
  - indistinguishable from local files for all uses
  - all clients see all files from anywhere
- Performance
  - per-client: at least as fast as local disk
  - scalability: unaffected by the number of clients
- Cost
  - capital: less than local (per client) disk storage
  - operational: zero, it requires no administration
- Capacity: unlimited, it is never full
- Availability: 100%, no failures or down-time

Remote Data Access: Challenges

- Transparency
  - despite Deutch's warnings
  - creating global file name-spaces
- Security
  - despite insecure networks and heterogeneous systems
- Preserving ACID semantics, Posix consistency
  - despite lack of shared memory and atomic instructions
- Performance
  - despite everything being done with messages
- Reliability and Scalability
  - despite having more parts and modes of failure

Key Characteristics of Solutions

- APIs and Transparency
  - how do users and processes access remote files
  - how closely do remote files mimic local files
- Performance and Robustness
  - are remote files as fast and reliable as local ones
- Architecture
  - how is solution integrated into clients and servers
- Protocol and Work Partitioning
  - what messages exchanged, who does what work

Client/Server Models

- Peer-to-Peer
  - most systems have resources (e.g. disks, printers)
  - they cooperate/share with one-another
- Thin Client
  - few local resources (e.g. CPU, NIC, display)
  - most resources on work-group or domain servers
- Cloud Services
  - clients access services rather than resources
  - clients do not see individual servers

Remote File Transfer

- explicit commands to copy remote files
  - OS specific: *scp(1)*, *rsync(1),* **S3** tools
  - IETF protocols: FTP, SFTP
- implicit remote data transfers
  - browsers (transfer files with HTTP)
  - email clients (move files with IMAP/POP/SMTP)
- advantages: efficient, requires no OS support
- disadvantages: latency, lack of transparency

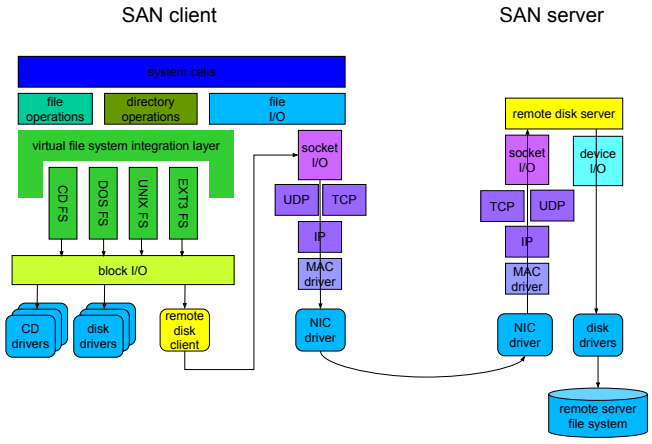
Remote Data Access

- OS makes remote files appear to be local
  - remote disk access (e.g. Storage Area Network)
  - remote file access (e.g. Network Attached Storage)
  - distributed file systems (NAS on steroids)
- advantages
  - transparency, availability, throughput
  - scalability, cost (capital and operational)
- disadvantages
  - complexity, performance, consistency models

Remote Disk Access

- Goal: complete transparency
  - normal file system calls work on remote files
  - all programs "just work" with remote files
- Typical Architectures
  - Storage Area Network (SCSI over Fibre Chanel)
    - very fast, very expensive, moderately scalable
  - iSCSI (SCSI over ethernet)
    - client driver turns reads/writes into network requests
    - server daemon receives/serves requests
    - moderate performance, inexpensive, highly scalable

Remote Disk Access Architecture



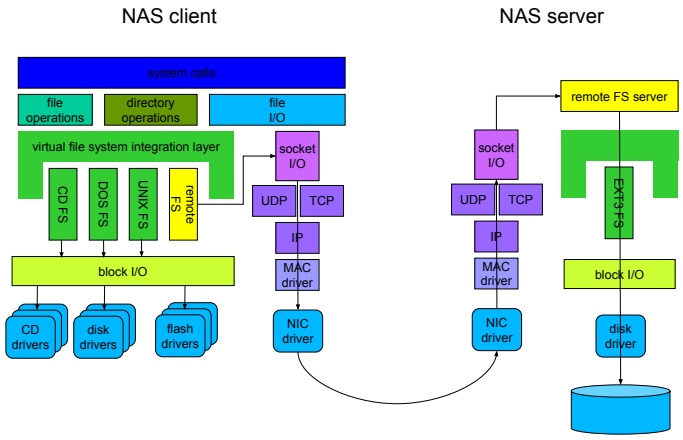
Rating Remote Disk Access

- Advantages:
  - provides excellent transparency
  - decouples client hardware from storage capacity
  - performance/reliability/availability per back-end
- Disadvantages
  - inefficient fixed partition space allocation
  - can't support file sharing by multiple client systems
  - message losses can cause file system errors
- This is THE model for Virtual Machines

Remote File Access

- Goal: complete transparency
  - normal file system calls work on remote files
  - support file sharing by multiple clients
  - performance, availability, reliability, scalability
- Typical Architecture
  - Network Attached Storage Protocols: NFS, CIFS
  - exploits client-side plug-in file systems
    - client-side file system is a local proxy
    - translates file operations into RPC requests
  - server-side daemon receives/process requests
    - translates them into operations on local file system

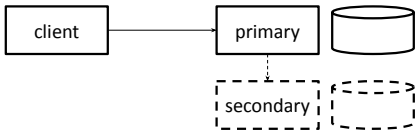
Remote File Access Architecture



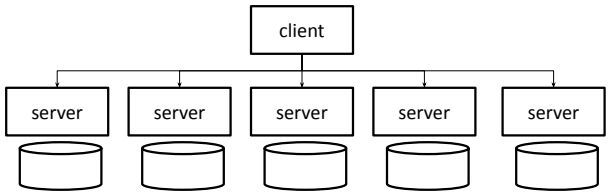
Rating Remote File Access

- Advantages
  - very good application level transparency
  - very good functional encapsulation
  - able to support multi-client file sharing
  - potential for good performance and robustness
- Disadvantages
  - at least part of implementation must be in the OS
  - client and server sides tend to be fairly complex
- This is THE model for client/server storage

Remote Disk/File Access



Distributed File System



(Remote vs. Distributed FS)

- Remote File Access (e.g. NFS, CIFS)
  - client talks to (per FS) primary server
  - secondary server may take over if primary fails
  - advantages: simplicity
- Distributed File System (e.g. Ceph, RAMCloud)
  - data is spread across numerous servers
  - client may talk directly to many/all of them
  - advantages: performance, scalability
  - disadvantages: complexity++

Security: Anonymous access

- all files available to all users
  - no authentication required
  - may be limited to read-only access
  - examples: anonymous FTP, HTTP
- advantages
  - simple implementation
- disadvantages
  - incapable of providing information privacy
  - write access often managed by other means

Peer-to-Peer Security

- client-side authentication/authorization
  - all users are known to all systems
  - all systems are trusted to enforce access control
  - example: basic NFS
- advantages
  - simple implementation
- limitations
  - assumes all client systems can be trusted
  - assumes all users are known to all systems
    - UID mapping between heterogeneous OSs
    - efficiency /scalability of universal user registries

Server Authenticated Sessions

- client agent authenticates to each server
  - session authorization based on those credentials
  - example: CIFS, authenticated HTTPS sessions
- advantages
  - simple implementation
- disadvantages
  - may not work in heterogeneous OS environment
  - universal user registry is not scalable
  - no automatic fail-over if server dies

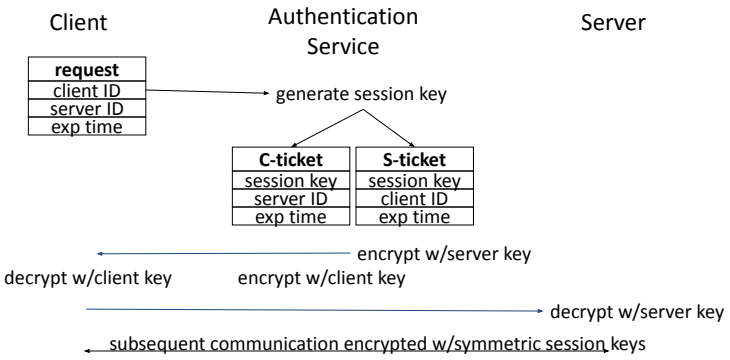
Domain Authentication Service

- independent authentication of client & server
  - each authenticates with authentication service
  - each knows/trusts only the authentication service
- authentication service issues signed "tickets"
  - assuring each of the others' identity and rights
  - may be revocable or have a limited life-time
- may establish secure two-way session
  - privacy – nobody else can snoop on conversation
  - integrity – nobody can generate fake messages

example: KERBEROS

- establishes secure client/server sessions
- based on digital signatures
  - every agent has a secret (symmetric) key
  - keys are known only to agent, and KERBEROS
- request to KERBEROS encrypted w/client key
  - KERBEROS can decrypt it, authenticating requester
- KERBEROS response is two-part work ticket
  - part 1: encrypted with client's key
    - a symmetric session key
    - part 2 (to be forward, by client, to server)
  - part 2: encrypted with server's key
    - client ID, ticket duration,
    - symmetric session key

KERBEROS Work Tickets



Distributed Authorization

- Authentication service returns credentials
  - which server checks against Access Control List
  - advantage: auth service doesn't know about ACLs
- Authentication service returns Capabilities
  - which server can verify (by signature)
  - advantage: servers do not know about clients
- Both approaches are commonly used
  - credentials: if subsequent authorization required
  - capabilities: if access can be granted all-at-once
  - either may have an expiration time

Reading and Assignments

Reading:

- Arpaci C49 … Andrew File System
- ACID semantics

Projects:

- 4C should be nearly complete

Supplementary Slides

Storage Area Networks

- Goals
  - flexibility of local area networking
    - any client can talk to any storage device
  - performance of dedicated disk interfaces
- Typical Architecture
  - giga-bit fibre channel network
    - arbitrated access, very large packet sizes
    - clients access network via an FC SCSI HBA
    - lower cost ethernet (iSCSI) is also becoming popular
  - intelligent non-blocking switches & controllers
    - volume management, caching, mirroring, striping

Rating SANs

- Advantages:
  - decouples client hardware from storage capacity
  - outstanding performance
- Disadvantages
  - very expensive
  - they are still a remote disk solution
    - poorly abstracted for remote file access
    - inefficient allocation, doesn't provide multi-client sharing
- Contemporary Usage
  - they have revolutionized block storage

Network Attached Storage

- enabled by standard file access protocols
  - CIFS, NFS, HTTP, FTP
- a "Storage Appliance"
  - you plug it in, and you start using it
- may provide advanced functionality
  - mirroring (or RAID-5) with automatic recovery
  - snap-shots
- does not expose details of its implementation
  - CPU, OS, file systems, disks