Distributed File Systems

Distributed File Systems

- File System : provides persistent storage through files
- File : named objects that exist from explicit creation till explicit destruction
- DFS : distributed implementation of a file system
  - files, file servers, and users are all dispersed around the network

Main goal: DFS should look like a centralized file system to an user

Design Goals

- Network transparency
  - same set of file operations for both local and remote files
  - uniform naming scheme for local and remote files
  - similar access time for local and remote files
- High availability
  - Files should not become unavailable for “small” failures
- Scalability
  - no. of users, file servers, files handled etc.

Design Issues and Mechanisms

- Naming scheme & name resolution
  - Location transparency vs. location independence
  - single global namespace or not
  - mounting
  - pathname translation
  - hints

- Caching
  - client and server cache
  - cache in disk or memory?
  - granularity of cached data
  - write policy
  - cache consistency

- Sharing semantics
  - UNIX semantics
  - Session semantics
  - Transaction semantics

- Fault tolerance
  - stateful vs. stateless service
  - file replication

NFS (Network File System)

- Server exports parts of a filesystem to clients (specified in /etc/exports file in Linux, command “exportfs” actually exports)
- Clients “mount” server exported namespace at specific mount points in its namespace tree (specified in /etc/fstab in Linux; can be mounted separately by mount command also)
- Same server filesystem can be mounted at different mount points at different clients, so no single global name space
- mounts can be cascaded

- Naming
  - location transparent (host id needed only during mount)
  - pathname translation
    - look up each component in the pathname recursively
    - lookup call made to server when a mount point is crossed
    - cascading mounts can involve multiple servers during translation
    - Also, each component requires a separate RPC call to server. Costly!

- Caching
  - Strictly not part of NFS protocol, but used in practice for performance
    - file attributes cached along with file block.
    - cached file block used subject to consistency checks on file attributes
  - 8 Kb blocks for v2, negotiable for v3
- sharing semantics
  - No clear semantics because of caching details
- Fault tolerance
  - stateless approach – resilient to client, server, and network failure