# *Distributed Systems*

CS 502
Spring 99
WPI MetroWest/Southboro Campus

## *Network and Distributed–System Structures*

- Network Structures
  - Background
  - Motivation
  - Topology
  - Network Types
  - Communication
  - Design Strategies
- Distributed–System Structures
  - Network-Operating Systems
  - Distributed-Operating Systems
  - Remote Services
  - Robustness
  - Design Issues

1

## *Getting More Work Done*

- Work Harder
  - Reduce Idle Time
  - Eliminate Lower Priority Activities
  - Greater Productivity for the Same Operations
  - *processor speed*
- Work Smarter
  - Find a more efficient algorithm
  - *algorithms*
- Get Help
  - Add additional workers to the task
  - Requires ability to separate subtasks
  - *parallel processing*

2

## *Taxonomy of Parallel/Distributed Systems*

- Covert parallel processing
  - superpipelined, superscaler, VLIW
- Overt parallel processing (MIMD)
  - Tightly Coupled
    - Bus Based Shared Memory Multiprocessors
    - Switched Multiprocessors
  - "Medium" Coupled
    - Switched Multicomputers with private memory
  - Loosely Coupled
    - Private Memory Multicomputers on a network

3

## *Distributed System Definition*

- Lamport – "A *distributed system* is a system where a machine I have never heard of stops me from getting work done"
- Common Definition – A *distributed system* is a collection of computers that do not share memory nor a common clock.

4

## *Site Types*

- Mainframes (IBM3090, etc.)
  - example applications
    - airline reservations
    - banking systems
  - many large attached disks
- Workstations (Sun, HP, IBM)
  - example applications
    - computer aided design
    - office–information systems
    - private databases
  - zero to a few medium size disks

5

## Site Types (Cont.)

- Personal Computers
  - example applications
    - client-side applications
    - office information systems
    - small private databases
  - zero to a few small disks
- Network Computers
  - example applications
    - client-side applications
    - other?
  - no disk



## Potential Benefits

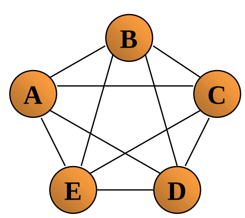
- Resource sharing
  - sharing and printing files at remote sites
  - processing information in a distributed database
  - using remote specialized hardware devices
- Computation speedup – load sharing
- Reliability – detect and recover from site failure, function transfer, reintegrate failed site
- Communication – message passing

## Topology

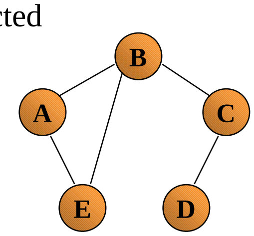
- Sites in the system can be physically connected in a variety of ways; they are compared with respect to the following criteria:
  - **Basic cost**. How expensive is it to link the various sites in the system?
  - **Communication cost**. How long does it take to send a message from site A to site B?
  - **Reliability**. If a link or a site in the system fails, can the remaining sites still communicate with each other?
- The various topologies are depicted as graphs whose nodes correspond to sites. An edge from node A to node B corresponds to a direct connection between the two sites.
- The following six items depict various network topologies.

8

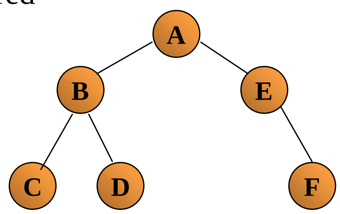
- Fully Connected



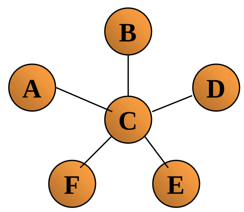

- Partially Connected



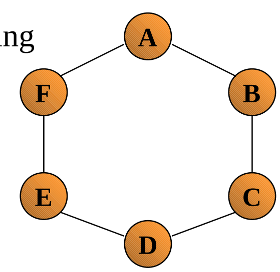
- Tree Structured



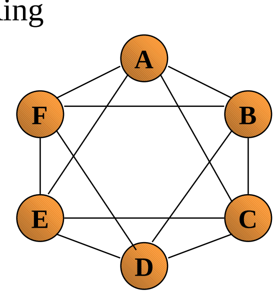

- Star



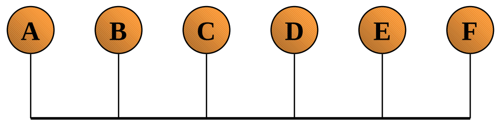

- Single Link Ring



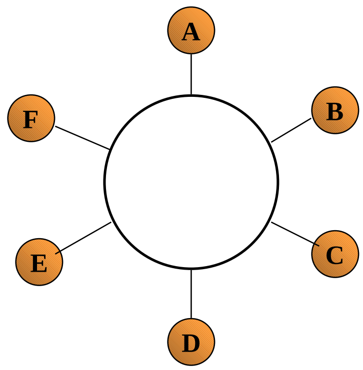

- Double Link Ring



- Linear Bus




- Ring Bus




## *Network Types*

- Local-Area Network (LAN) – designed to cover small geographical area.
  - Multiaccess bus, ring, or star network.
  - Speed – 10 megabits/second, or higher.
  - Broadcast is fast and cheap.
  - Nodes:
    - usually workstations and/or personal computers
    - a few (usually one or two) mainframes
  - Intranet

13

## Network Types (Cont.)

- Wide-Area Network (WAN) – links geographically separated sites.
  - Point-to-point connections over long-haul lines (often leased from a phone company).
  - Speed – 100 kilobits/second.
  - Broadcast usually requires multiple messages.
  - Nodes:
    - usually a high percentage of mainframes
  - Internet
  - Extranet

14

## Network Types (Cont.)

- System Area Network (SAN) – Nodes and I/O Devices connected via high-speed interconnect.
  - Interconnect Examples
    - 100 MBps Switched Ethernet; Gigabit Ethernet
    - FibreChannel
    - ServerNet
    - Intel GigaBit Bus

15

## Communication

The design of a communication network must address four basic issues:

- **Naming and name resolution:** How do two processes locate each other to communicate?
- **Routing strategies**. How are messages sent through the network?
- **Connection strategies**. How do two processes send a sequence of messages?
- **Contention**. The network is a shared resource, so how do we resolve conflicting demands for its use?

16

## Naming and Name Resolution

- Name systems in the network.
- Address messages with the process-id.
- Identify processes on remote systems by <host-name, identifier> pair.
- *Domain name service* (DNS) – specifies the naming structure of the hosts, as well as name to address resolution (Internet).

17

## Routing Strategies

- **Fixed routing**. A path from A to B is specified in advance; path changes only if a hardware failure disables it.
  - Since the shortest path is usually chosen, communication costs are minimized.
  - Fixed routing cannot adapt to load changes.
  - Ensures that messages will be delivered in the order in which they were sent.
- **Virtual circuit**. A path from A to B is fixed for the duration of one session. Different sessions involving messages from A to B may have different paths.
  - Partial remedy to adapting to load changes.
  - Ensures that messages will be delivered in the order in which they were sent.

18

## Routing Strategies (Cont.)

- **Dynamic routing**. The path used to send a message from site A to site B is chosen only when a message is sent.
  - Usually a site sends a message to another site on the link least used at that particular time.
  - Adapts to load changes by avoiding routing messages on heavily used path.
  - Messages may arrive out of order. This problem can be remedied by appending a sequence number to each message.

19

## Connection Strategies

- **Circuit switching**. A permanent physical link is established for the duration of the communication (i.e., telephone system).
- **Message switching**. A temporary link is established for the duration of one message transfer (i.e., post-office mailing system).
- **Packet switching**. Messages of variable length are divided into fixed-length packets which are sent to the destination. Each packet may take a different path through the network. The packets must be reassembled into messages as they arrive.

Circuit switching requires setup time, but incurs less overhead for shipping each message, and may waste network bandwidth. Message and packet switching require less setup time, but incur more overhead per message.

20

## Contention

Several sites may want to transmit information over a link simultaneously. Techniques to avoid repeated collisions include:

- CSMA/CD. Carrier sense with multiple access (CSMA); collision detection (CD)
  - A site determines whether another message is currently being transmitted over that link. If two or more sites begin transmitting at exactly the same time, then they will register a CD and will stop transmitting.
  - When the system is very busy, many collisions may occur, and thus performance may be degraded.
- CSMA/CD is used successfully in the Ethernet system, the most common network system.

21

## *Contention (Cont.)*

- **Token passing**. A unique message type, known as a token, continuously circulates in the system (usually a ring structure). A site that wants to transmit information must wait until the token arrives. When the site completes its round of message passing, it retransmits the token. A token-passing scheme is used by the IBM and Apollo systems.
- **Message slots**. A number of fixed-length message slots continuously circulate in the system (usually a ring structure). Since a slot can contain only fixed-sized messages, a single logical message may have to be broken down into a number of smaller packets, each of which is sent in a separate slot. This scheme has been adopted in the experimental Cambridge Digital Communication Ring.

22

## *Layered Communication Architecture*

The communication network is partitioned into the following multiple layers:

- Physical layer – handles the mechanical and electrical details of the physical transmission of a bit stream.
- Data-link layer – handles the frames, or fixed-length parts of packets, including any error detection and recovery that occurred in the physical layer.
- Network layer – provides connections and routes packets in the communication network, including handling the address of outgoing packets, decoding the address of incoming packets, and maintaining routing information for proper response to changing load levels.

23

## Layered Communication Architecture

- Transport layer – responsible for low-level network access and for message transfer between clients, including partitioning messages into packets, maintaining packet order, controlling flow, and generating physical addresses.
- Session layer – implements sessions, or process-to-process communications protocols.
- Presentation layer – resolves the differences in formats among the various sites in the network, including character conversions, and half duplex/full duplex (echoing).
- Application layer – interacts directly with the users; deals with file transfer, remote-login protocols and electronic mail, as well as schemas for distributed databases.



## Network Operating Systems

- Users are aware of multiplicity of machines. Access to resources of various machines is done explicitly by:
  - Remote logging into the appropriate remote machine.
  - Transferring data from remote machines to local machines, via the File Transfer Protocol (FTP) mechanism.

## *Distributed Operating Systems*

- Users not aware of multiplicity of machines. Access to remote resources similar to access to local resources.
- Data Migration – transfer data by transferring entire file, or transferring only those portions of the file necessary for the immediate task.
- Computation Migration – transfer the computation, rather than the data, across the system.

26

## *Distributed–Operating Systems (Cont.)*

- Process Migration – execute an entire process, or parts of it, at different sites.
  - Load balancing – distribute processes across network to even the workload.
  - Computation speedup – subprocesses can run concurrently on different sites.
  - Hardware preference – process execution may require specialized processor.
  - Software preference – required software may be available at only a particular site.
  - Data access – run process remotely, rather than transfer all data locally.

27

## Remote Services

- Requests for access to a remote file are delivered to the server. Access requests are translated to messages for the server, and the server replies are packed as messages and sent back to the user.
- A common way to achieve this is via the Remote Procedure Call (RPC) paradigm.
- Messages addressed to an RPC daemon listening to a port on the remote system contain the name of a process to run and the parameters to pass to that process. The process is executed as requested, and any output is sent back to the requester in a separate message.
- A port is a number included at the start of a message packet. A system can have many ports within its one network address to differentiate the network services it supports.

28

## RPC Binding

- Client and Server Port addresses must be resolved and bound.
  - Binding information may be predecided, in the form of fixed port addresses.
    - At compile time, an RPC call has a fixed port number associated with it.
    - Once a program is compiled, the server cannot change the port number of the requested service.
  - Binding can be done dynamically by a rendezvous mechanism.
    - Operating system provides a rendezvous daemon on a fixed RPC port.
    - Client then sends a message to the rendezvous daemon requesting the port address of the RPC it needs to execute.

29

## RPC Scheme (Cont.)

A distributed file system (DFS) can be implemented as a set of RPC daemons and clients.

- The messages are addressed to the DFS port on a server on which a file operation is to take place.
- The message contains the disk operation to be performed (i.e., read, write, rename, delete, or status).
- The return message contains any data resulting from that call, which is executed by the DFS daemon on behalf of the client.

30

## Threads

- Threads can send and receive messages while other operations within the task continue asynchronously.
- Pop-up thread– created on “as needed” basis to respond to new RPC.
  - Cheaper to start new thread than to restore existing one.
  - No threads block waiting for new work; no context has to be saved, or restored.
  - Incoming RPCs do not have to be copied to a buffer within a server thread.
- RPCs to processes on the same machine as the caller made more lightweight via shared memory between threads in different processes running on same machine.

31

## DCE Thread Calls

- Thread-management:
  - **create, exit, join, detach**
- Synchronization:
  - **mutex init, mutex destroy, mutex lock, mutex trylock, mutex unlock**
- Condition-variable:
  - **cond init, cond destroy, cond wait, cond signal, cond broadcast**
- Scheduling:
  - **setscheduler, getscheduler, setprio, getprio**
- Kill-thread:
  - **cancel, setcancel**

32

## Robustness

To ensure that the system is robust, we must :

- Detect failures.
  - process
  - site
  - link
- Reconfigure the system so that computation may continue.
- Recover when a failure is repaired.

33

## Failure Detection – Handshaking Approximation

- At fixed intervals, sites A and B send each other an "I-am-up" message. If site A does not receive this message within a predetermined time period, it can assume that site B has failed, that the link between A and B has failed, or that the message from B has been lost.
- At the time site A sends the "Are-you-up?" message, it specifies a time interval during which it is willing to wait for the reply from B. If A does not receive B's reply message within the time interval, A may conclude that one or more of the following situations has occurred:
  - Site B is down.
  - The direct link (if one exists) from A to B is down.
  - The alternative path from A to B is down.
  - The message has been lost.

34

## Reconfiguration

- Procedure that allows the system to reconfigure and to continue its normal mode of operation.
- If a direct link from A to B has failed, this information must be broadcast to every site in the system, so that the various routing tables can be updated accordingly.
- If it is believed that a site has failed (because it can no longer be reached), then every site in the system must be so notified, so that they will no longer attempt to use the services of the failed site.

35

## Recovery from Failure

- When a failed link or site is repaired, it must be integrated into the system gracefully and smoothly.
- Suppose that a link between A and B has failed. When it is repaired, both A and B must be notified. We can accomplish this notification by continuously repeating the handshaking procedure.
- Suppose that site B has failed. When it recovers, it must notify all other sites that it is up again. Site B then may have to receive from the other sites various information to update its local tables.

36

## Design Issues

- Transparency and locality – distributed system should look like conventional, centralized system and not distinguish between local and remote resources.
- User mobility – brings user's environment (i.e., home directory) to wherever the user logs in.
- Fault tolerance – system should continue functioning, perhaps in a degraded form, when faced with various types of failures.
- Scalability – system should adapt to increased service load.
- Large-scale systems – service demand from any system component should be bounded by a constant that is independent of the number of nodes.
- Servers' process structure – servers should operate efficiently in peak periods; use lightweight processes or threads.

37

## Distributed File Systems

- Background
- Naming and Transparency
- Remote File Access
- Stateful versus Stateless Service
- File Replication
- Example Systems

38

## Background

- Distributed file system (DFS) – a distributed implementation of the classical time-sharing model of a file system, where multiple users share files and storage resources.
- A DFS manages sets of dispersed storage devices.
- Overall storage space managed by a DFS is composed of different, remotely located, smaller storage spaces.
- There is usually a correspondence between constituent storage spaces and sets of files.

39

## DFS Structure

- **Service–** software entity running on one or more machines and providing a particular type of function to a priori unknown clients.
- **Server–** service software running on a single machine.
- **Client–** process that can invoke a service using a set of operations that forms its client interface.
- A client interface for a file service is formed by a set of primitive file operations(create, delete, read, write).
- Client interface of a DFS should be transparent, i.e., not distinguish between local and remote files.

40

## Naming and Transparency

- **Naming–** mapping between logical and physical objects.
- **Multilevel mapping** – abstraction of a file that hides the details of how and where on the disk the file is actually stored.
- A *transparent* DFS hides the location where in the network the file is stored.
- For a file being replicated in several sites, the mapping returns a set of the locations of this file's replicas; both the existence of multiple copies and their location are hidden.

41

## Naming Structures

- *Location transparency* – file name does not reveal the file's physical storage location.
  - File name still denotes a specific, although hidden, set of physical disk blocks.
  - Convenient way to share data.
  - Can expose correspondence between component units and machines.
- *Location independence* – file name does not need to be changed when the file's physical storage location changes.
  - Better file abstraction.
  - Promotes sharing the storage space itself.
  - Separates the naming hierarchy from the storage-devices hierarchy.

42

## Naming Schemes – Three Main Approaches

- Files named by combination of their host name and local name; guarantees a unique system-wide name.
- Attach remote directories to local directories, giving the appearance of a coherent directory tree; only previously mounted remote directories can be accessed transparently.
- Total integration of the component file systems.
  - A single global name structure spans all the files in the system.
  - If a server is unavailable; some arbitrary set of directories on different machines also becomes unavailable.

43

## Remote File Access

- Reduce network traffic by retaining recently accessed disk blocks in a cache, so that repeated accesses to the same information can be handled locally.
  - If needed data not already cached, a copy of data is brought from the server to the user.
  - Accesses are performed on the cached copy.
  - Files identified with one master copy residing at the server machine, but copies of (parts of) the file are scattered in different caches.
- *Cache-consistency problem*– keeping the cached copies consistent with the master file.



## Location – Disk Caches versus Main Memory Cache

- Advantages of disk caches
  - More reliable.
  - Cached data kept on disk are still there during recovery and don't need to be fetched again.
- Advantages of main-memory caches:
  - Permit workstations to be diskless.
  - Data can be accessed more quickly.
  - Performance speedup in bigger memories.
  - Server caches (used to speed up disk I/O) are in main memory regardless of where user caches are located; using main-memory caches on the user machine permits a single caching mechanism for servers and users.

## Cache Update Policy

- **Write-through**– write data through to disk as soon as they are placed on any cache. Reliable, but poor performance.
- **Delayed-write** – modifications written to the cache and then written through to the server later. Write accesses complete quickly; some data may be overwritten before they are written back, and so need never be written at all.
  - Poor reliability; unwritten data will be lost whenever a user machine crashes.
  - Variation – scan cache at regular intervals and flush blocks that have been modified since the last scan.
  - Variation – *write-on-close*, writes data back to the server when the file is closed. Best for files that are open for long periods and frequently modified.

46

## Consistency

- Is locally cached copy of the data consistent with the master copy?
- Client-initiated approach
  - Client initiates a validity check.
  - Server checks whether the local data are consistent with the master copy.
- Server-initiated approach
  - Server records, for each client, the (parts of) files it caches.
  - When server detects a potential inconsistency, it must react.

47

## Comparing Caching and Remote Service

- In caching, many remote accesses handled efficiently by the local cache; most remote accesses will be served as fast as local ones.
- Servers are contacted only occasionally in caching (rather than for each access).
  - Reduces server load and network traffic.
  - Enhances potential for scalability.
- Remote server method handles every remote access across the network; penalty in network traffic, server load, and performance.
- Total network overhead in transmitting big chunks of data (caching) is lower than a series of responses to specific requests (remote-service).

48

## Caching and Remote Service (Cont.)

- Caching is superior in access patterns with infrequent writes. With frequent writes, substantial overhead incurred to overcome cache-consistency problem.
- Benefit from caching when execution carried out on machines with either local disks or large main memories.
- Remote access on diskless, small-memory-capacity machines should be done through remote-service method.
- In caching, the lower inter-machine interface is different from the upper user interface.
- In remote-service, the inter-machine interface mirrors the local user-file-system interface.

49

## *Stateful File Service*

- Mechanism.
  - Client opens a file.
  - Server fetches information about the file from its disk, stores it in its memory, and gives the client a connection identifier unique to the client and the open file.
  - Identifier is used for subsequent accesses until the session ends.
  - Server must reclaim the main-memory space used by clients who are no longer active.
- Increased performance.
  - Fewer disk accesses.
  - Stateful server knows if a file was opened for sequential access and can thus read ahead the next blocks.

50

## *Stateless File Server*

- Avoids state information by making each request self-contained.
- Each request identifies the file and position in the file.
- No need to establish and terminate a connection by open and close operations.

51

## Distinctions between Stateful and Stateless Service

- Failure Recovery.
  - A stateful server loses all its volatile state in a crash.
    - Restore state by recovery protocol based on a dialog with clients, or abort operations that were underway when the crash occurred.
    - Server needs to be aware of client failures in order to reclaim space allocated to record the state of crashed client processes ( orphan detection and elimination).
  - With stateless server, the effects of server failures and recovery are almost unnoticeable. A newly reincarnated server can respond to a self-contained request without any difficulty.

52

## Distinctions (Cont.)

- Penalties for using the robust stateless service:
  - longer request messages
  - slower request processing
  - additional constraints imposed on DFS design
- Some environments require stateful service.
  - A server employing server-initiated cache validation cannot provide stateless service, since it maintains a record of which files are cached by which clients.
  - UNIX use of file descriptors and implicit offsets is inherently stateful; servers must maintain tables to map the file descriptors to inodes, and store the current offset within a file.

53

## File Replication

- Replicas of the same file reside on failure-independent machines.
- Improves availability and can shorten service time.
- Naming scheme maps a replicated file name to a particular replica.
  - Existence of replicas should be invisible to higher levels.
  - Replicas must be distinguished from one another by different lower-level names.
- Updates – replicas of a file denote the same logical entity, and thus an update to any replica must be reflected on all other replicas.
- Demand replication – reading a non-local replica causes it to be cached locally, thereby generating a new non-primary replica.

54

## Example Systems

- The Sun Network File System (NFS)
- Locus

55

## *The Sun Network File System (NFS)*

- An implementation and a specification of a software system for accessing remote files across LANs (or WANs).
- The implementation is part of the SunOS operating system (version of 4.2BSD UNIX), running on a Sun workstation using an unreliable datagram protocol (UDP/IP protocol) and Ethernet.

56

## *NFS (Cont.)*

- Interconnected workstations viewed as a set of independent machines with independent file systems, which allows sharing among these file systems in a transparent manner.
  - A remote directory is mounted over a local file system directory. The mounted directory looks like an integral subtree of the local file system, replacing the subtree descending from the local directory.
  - Specification of the remote directory for the mount operation is nontransparent; the host name of the remote directory has to be provided. Files in the remote directory can then be accessed in a transparent manner.
  - Subject to access-rights accreditation, potentially any file system (or directory within a file system), can be mounted remotely on top of any local directory.

57

## NFS (Cont.)

- NFS is designed to operate in a heterogeneous environment of different machines, operating systems, and network architectures; the NFS specification is independent of these media.
- This independence is achieved through the use of RPC primitives built on top of an External Data Representation (XDR) protocol used between two implementation-independent interfaces.
- The NFS specification distinguishes between the services provided by a mount mechanism and the actual remote-file-access services.

58

## NFS Mount Protocol

- Establishes initial logical connection between server and client. Mount operation includes name of remote directory to be mounted and name of server machine storing it.
  - Mount request is mapped to corresponding RPC and forwarded to mount server running on server machine.
  - Export list – specifies local file systems that server exports for mounting, along with names of machines that are permitted to mount them.
- Following a mount request that conforms to its export list, the server returns a file handle—a key for further accesses.
- File handle – a file-system identifier, and an inode number to identify the mounted directory within the exported file system.
- The mount operation changes only the user's view and does not affect the server side.

59

## NFS Protocol

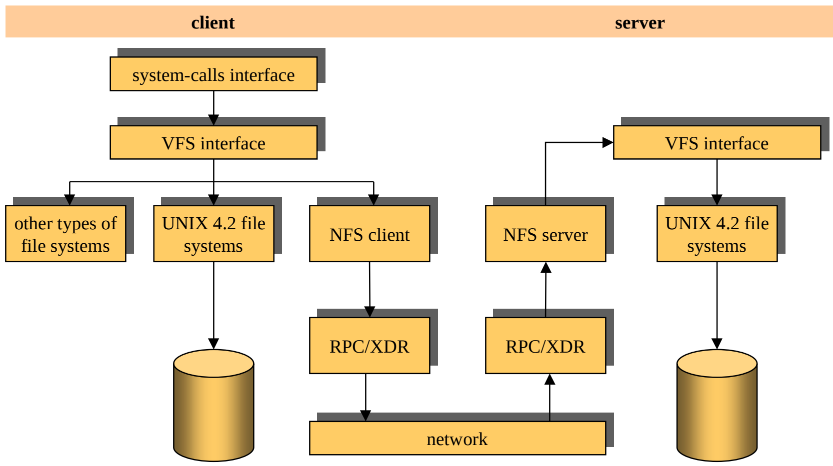
- Provides a set of remote procedure calls for remote file operations. The procedures support the following operations:
  - searching for a file within a directory
  - reading a set of directory entries
  - manipulating links and directories
  - accessing file attributes
  - reading and writing files
- NFS servers are stateless; each request has to provide a full set of arguments.
- Modified data must be committed to the server's disk before results are returned to the client (lose advantages of caching).
- The NFS protocol does not provide concurrency-control mechanisms.



## Three Major Layers of NFS Architecture

- UNIX file-system interface (based on the open, read, write, and close calls, and file descriptors).
- Virtual File System(VFS) layer – distinguishes local files from remote ones, and local files are further distinguished according to their file-system types.
  - The VFS activates file-system-specific operations to handle local requests according to their file-system types.
  - Calls the NFS protocol procedures for remote requests.
- NFS service layer – bottom layer of the architecture; implements the NFS protocol.

## Schematic View of NFS Architecture




62

## NFS Path–Name Translation

- Performed by breaking the path into component names and performing a separate NFS lookup call for every pair of component name and directory vnode.
- To make lookup faster, a directory name lookup cache on the client's side holds the vnodes for remote directory names.

63

## NFS Remote Operations

- Nearly one-to-one correspondence between regular UNIX system calls and the NFS protocol RPCs (except opening and closing files).
- NFS adheres to the remote-service paradigm, but employs buffering and caching techniques for the sake of performance.
- File-blocks cache – when a file is opened, the kernel checks with the remote server whether to fetch or revalidate the cached attributes. Cached file blocks are used only if the corresponding cached attributes are up to date.
- File-attribute cache – the attribute cache is updated whenever new attributes arrive from the server.
- Clients do not free delayed-write blocks until the server confirms that the data have been written to disk.



## LOCUS

- Project at the Univ. of California at Los Angeles to build a full-scale distributed OS; upward-compatible with UNIX, but the extensions are major and necessitate an entirely new kernel.
- File system is a single tree-structure naming hierarchy which covers all objects of all the machines in the system.
- Locus names are fully transparent.
- A Locus file may correspond to a set of copies distributed on different sites.
- File replication increases availability for reading purposes in the event of failures and partitions.
- A primary-copy approach is adopted for modifications.

## *LOCUS (Cont.)*

- Locus adheres to the same file-access semantics as standard UNIX.
- Emphasis on high performance led to the incorporation of networking functions into the operating system.
- Specialized remote operations protocols used for kernel-to-kernel communication, rather than the RPC protocol.
- Reducing the number of network layers enables performance for remote operations, but this specialized protocol hampers the portability of Locus.

66

## *LOCUS Name Structure*

- Logical filegroups form a unified structure that disguises location and replication details from clients and applications.
- A logical filegroup is mapped to multiple physical containers(or packs) that reside at various sites and that store file replicas of that filegroup.
- The <logical-filegroup-number, inode number> (the file's designator) serves as a globally unique low-level name for a file.

67

## *LOCUS Name Structure (Cont.)*

- Each site has a consistent and complete view of the logical name structure.
  - Globally replicated logical mount table contains an entry for each logical filegroup.
  - An entry records the file designator of the directory over which the filegroup is logically mounted, and indication of which site is currently responsible for access synchronization within the filegroup.
- An individual pack is identified by pack numbers and a logical filegroup number.
- One pack is designated as the primary copy.
  - a file must be stored at the primary copy site
  - a file can be stored also at any subset of the other sites where there exists a pack corresponding to its filegroup.

68

## *LOCUS Name Structure (Cont.)*

- The various copies of a file are assigned the same inode number on all the filegroup's packs.
  - Reference over the network to data pages use logical, rather than physical, page numbers.
  - Each pack has a mapping of these logical numbers to its physical numbers.
  - Each inode of a file copy contains a version number, determining which copy dominates other copies.
- Container table at each site maps logical filegroup numbers to disk locations for the filegroups that have packs locally on this site.

69

## LOCUS File Access

- Locus distinguishes three logical roles in file accesses, each one potentially performed by a different site:
  - Using site (US) – issues requests to open and access a remote file.
  - Storage site (SS) – site selected to serve requests.
  - Current synchronization site (CSS) – maintains the version number and a list of physical containers for every file in the filegroup.
    - Enforces global synchronization policy for a filegroup.
    - Selects an SS for each open request referring to a file in the filegroup.
    - At most one CSS for each filegroup in any set of communicating sites.

70

## LOCUS Synchronized Accesses to Files

- Locus tries to emulate conventional UNIX semantics on file accesses in a distributed environment.
- Multiple processes are permitted to have the same file open concurrently.
  - These processes issue read and write system calls.
  - The system guarantees that each successive operation sees the effects of the ones that precede it.
- In Locus, the processes share the same operating-system data structures and caches, and by using locks on data structures to serialize requests.

71

## LOCUS Two Sharing Modes

- A single token scheme allows several processes descending from the same ancestor to share the same position (offset) in a file. A site can proceed to execute system calls that need the offset only when the token is present.
- A multiple-data-tokens scheme synchronizes sharing of the file's in-core inode and data.
  - Enforces a single exclusive-writer, multiple-readers policy.
  - Only a site with the write token for a file may modify the file, and any site with a read token can read the file.
- Both token schemes are coordinated by token managers operating at the corresponding storage sites.

72

## LOCUS Operation in a Faulty Environment

- Maintain, within a single partition, strict synchronization among copies of a file, so that all clients of that file within that partition see the most recent version.
- Primary-copy approach eliminates conflicting updates, since the primary copy must be in the client's partition to allow an update.
- To detect and propagate updates, the system maintains a commit count which enumerates each commit of every file in the filegroup.
- Each pack has a lower-water mark( lwm) that is a commit-count value, up to which the system guarantees that all prior commits are reflected in the pack.

73