



Panel 1




Motivation

- **Distributed system** is collection of loosely coupled processors interconnected by a communications network
- Processors variously called *nodes*, *computers*, *machines*, *hosts*
 - *Site* is location of the processor
- Reasons for distributed systems
 - 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

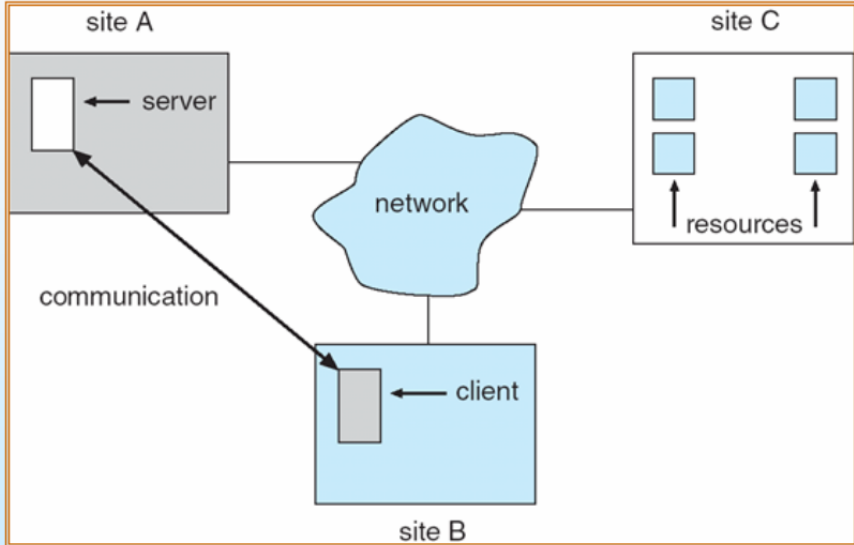


Operating System Concepts – 7th Edition, Apr 4, 2005 16.1 Silberschatz, Galvin and Gagne ©2005


Panel 2



A Distributed System

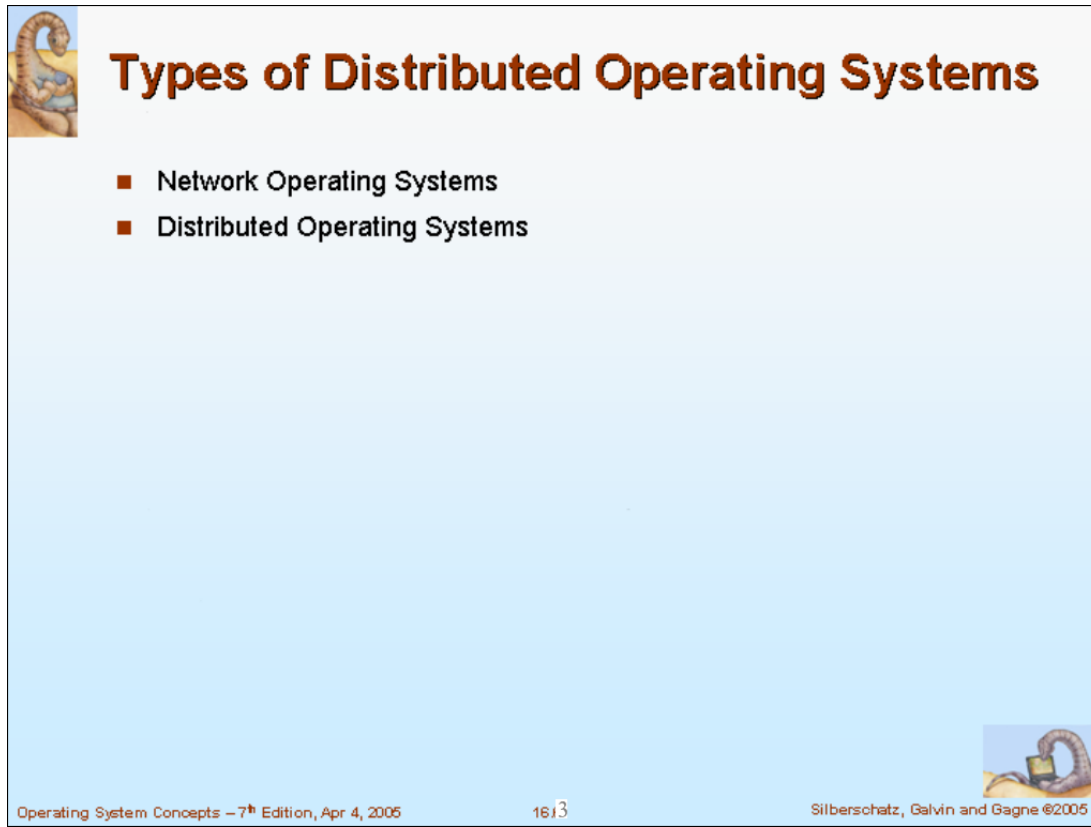


The diagram illustrates a distributed system with three sites: site A, site B, and site C, all connected to a central network. Site A contains a server, site B contains a client, and site C contains resources. Arrows indicate communication between the server and client, and connections to the network.



Operating System Concepts – 7th Edition, Apr 4, 2005 16.2 Silberschatz, Galvin and Gagne ©2005

Panel 3



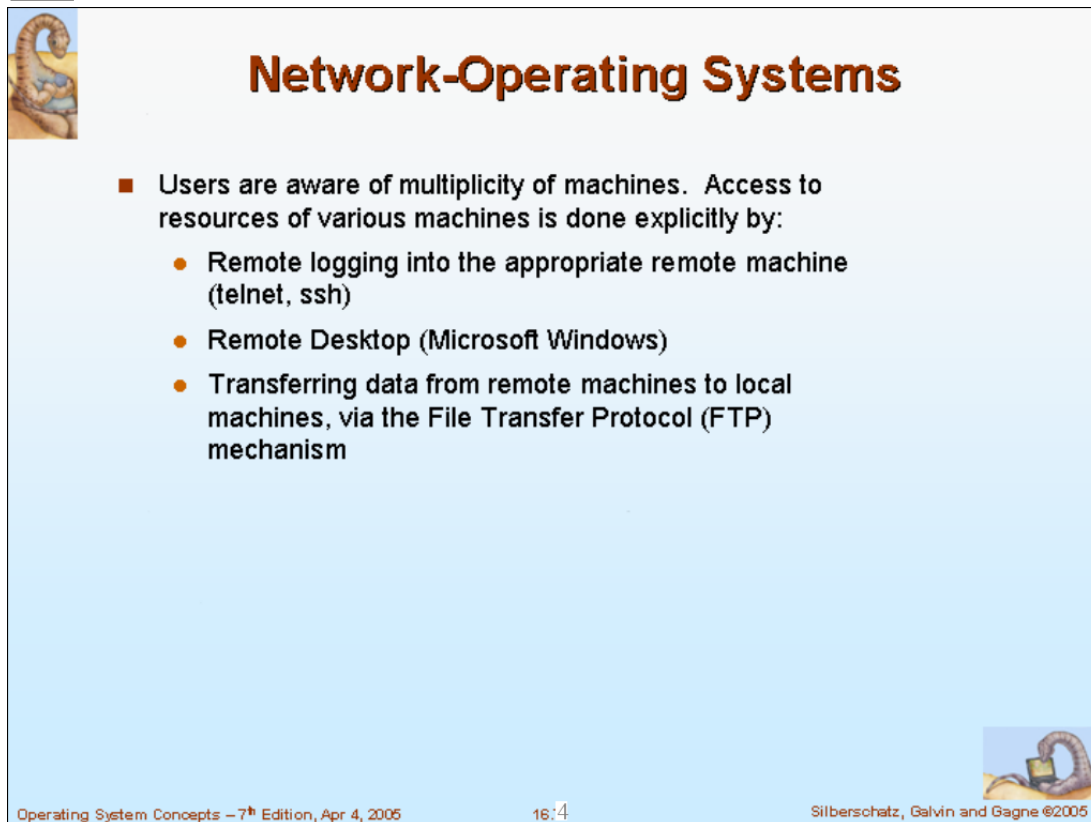
Types of Distributed Operating Systems

- Network Operating Systems
- Distributed Operating Systems

Operating System Concepts – 7th Edition, Apr 4, 2005 16:3 Silberschatz, Galvin and Gagne ©2005

This slide features a light blue background with a decorative illustration of a scorpion in the top-left and bottom-right corners. The title is in a large, bold, brown font. The list items are preceded by brown square bullet points. The footer contains the text 'Operating System Concepts – 7th Edition, Apr 4, 2005', the slide number '16:3', and the authors 'Silberschatz, Galvin and Gagne ©2005'.

Panel 4




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 (telnet, ssh)
 - Remote Desktop (Microsoft Windows)
 - Transferring data from remote machines to local machines, via the File Transfer Protocol (FTP) mechanism

Operating System Concepts – 7th Edition, Apr 4, 2005 16:4 Silberschatz, Galvin and Gagne ©2005


This slide features a light blue background with a decorative illustration of a scorpion in the top-left and bottom-right corners. The title is in a large, bold, brown font. The main bullet point is preceded by a brown square, and its sub-points are preceded by orange circles. The footer contains the text 'Operating System Concepts – 7th Edition, Apr 4, 2005', the slide number '16:4', and the authors 'Silberschatz, Galvin and Gagne ©2005'.

Panel 5




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




Operating System Concepts – 7th Edition, Apr 4, 2005 1615 Silberschatz, Galvin and Gagne ©2005

Panel 6




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




Operating System Concepts – 7th Edition, Apr 4, 2005 1616 Silberschatz, Galvin and Gagne ©2005

Panel 7



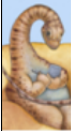
Design Issues

- **Transparency** – the distributed system should appear as a conventional, centralized system to the user
- **Fault tolerance** – the distributed system should continue to function in the face of failure
- **Scalability** – as demands increase, the system should easily accept the addition of new resources to accommodate the increased demand
- **Clusters** – a collection of semi-autonomous machines that acts as a single system




Operating System Concepts – 7th Edition, Apr 4, 2005 16:7 Silberschatz, Galvin and Gagne ©2005

Panel 8




Chapter 17 Distributed-File Systems

- Background
- Naming and Transparency
- Remote File Access
- Stateful versus Stateless Service
- File Replication
- An Example: AFS




Operating System Concepts – 7th Edition, Apr 4, 2005 17:8 Silberschatz, Galvin and Gagne ©2005

Panel 9




Chapter Objectives

- To explain the naming mechanism that provides location transparency and independence
- To describe the various methods for accessing distributed files
- To contrast stateful and stateless distributed file servers
- To show how replication of files on different machines in a distributed file system is a useful redundancy for improving availability
- To introduce the Andrew file system (AFS) as an example of a distributed file system




Operating System Concepts – 7th Edition, Apr 4, 2005 17.9 Silberschatz, Galvin and Gagne ©2005

Panel 10




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 set 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




Operating System Concepts – 7th Edition, Apr 4, 2005 17.10 Silberschatz, Galvin and Gagne ©2005

Panel 11




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




Operating System Concepts – 7th Edition, Apr 4, 2005 17/11 Silberschatz, Galvin and Gagne ©2005

Panel 12



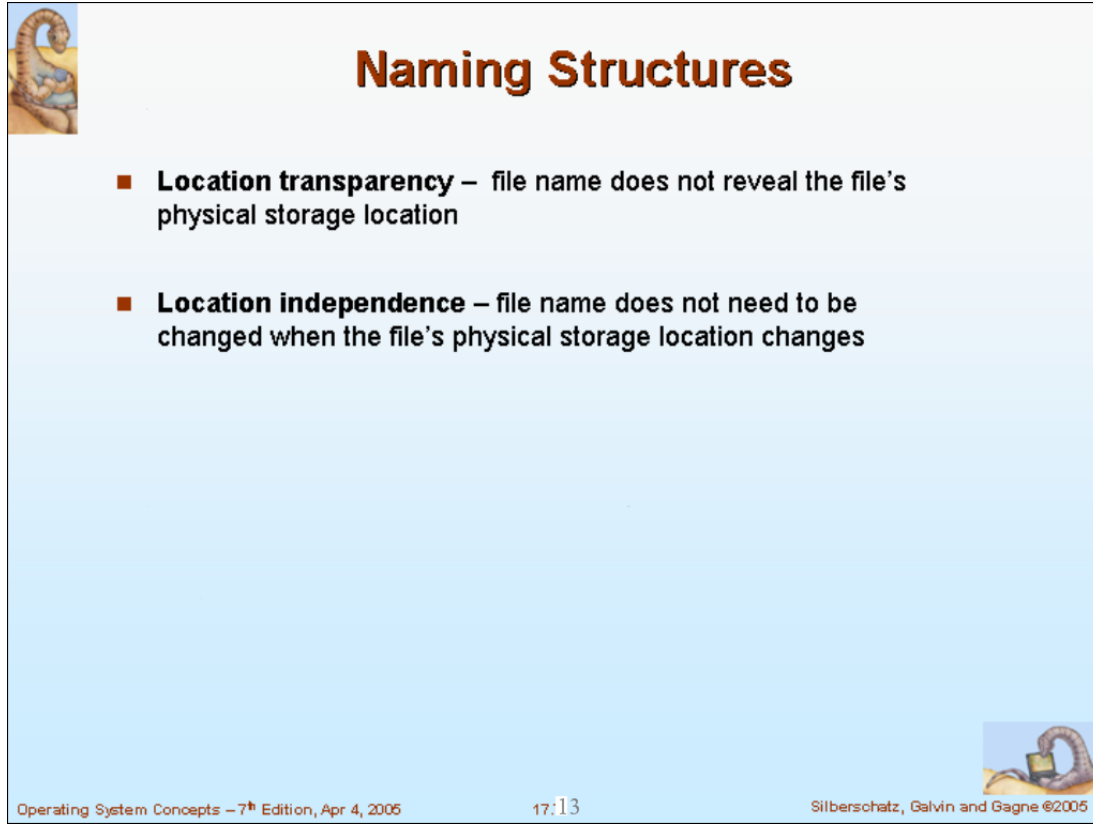
Naming and Transparency

- **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



Operating System Concepts – 7th Edition, Apr 4, 2005 17/12 Silberschatz, Galvin and Gagne ©2005

Panel 13

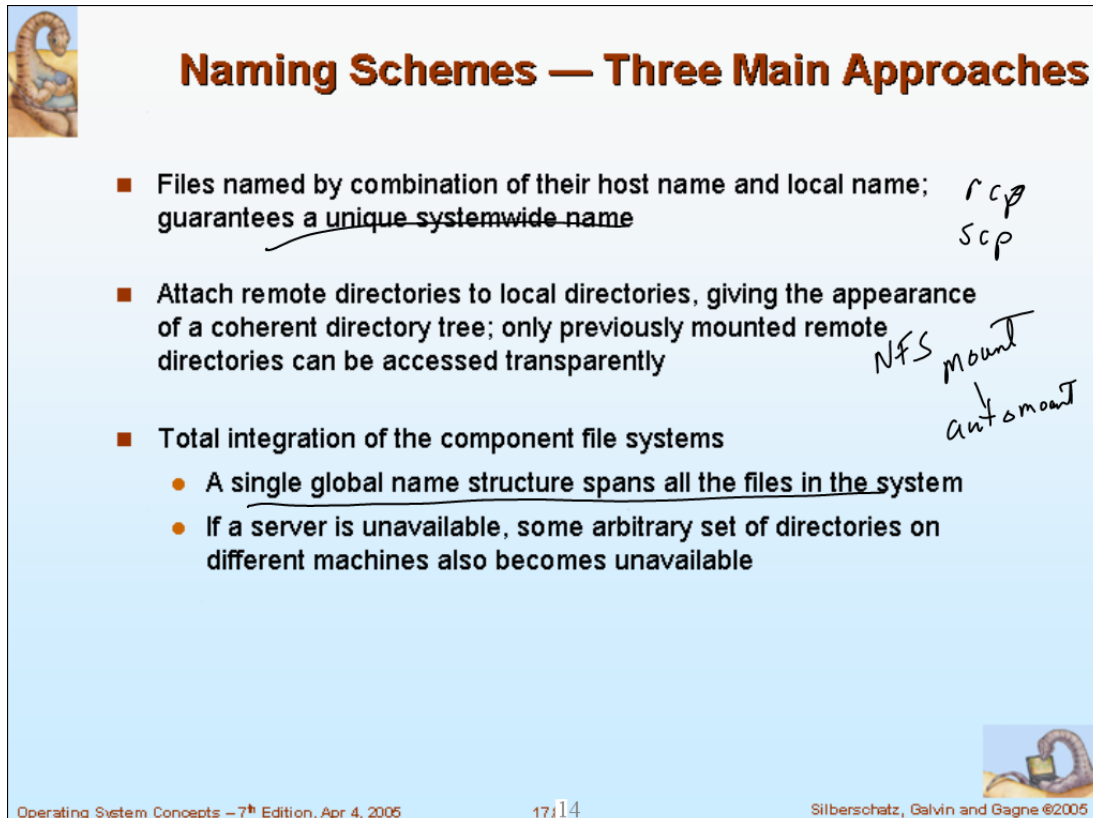


Naming Structures

- **Location transparency** – file name does not reveal the file's physical storage location
- **Location independence** – file name does not need to be changed when the file's physical storage location changes

Operating System Concepts – 7th Edition, Apr 4, 2005 17:13 Silberschatz, Galvin and Gagne ©2005

Panel 14

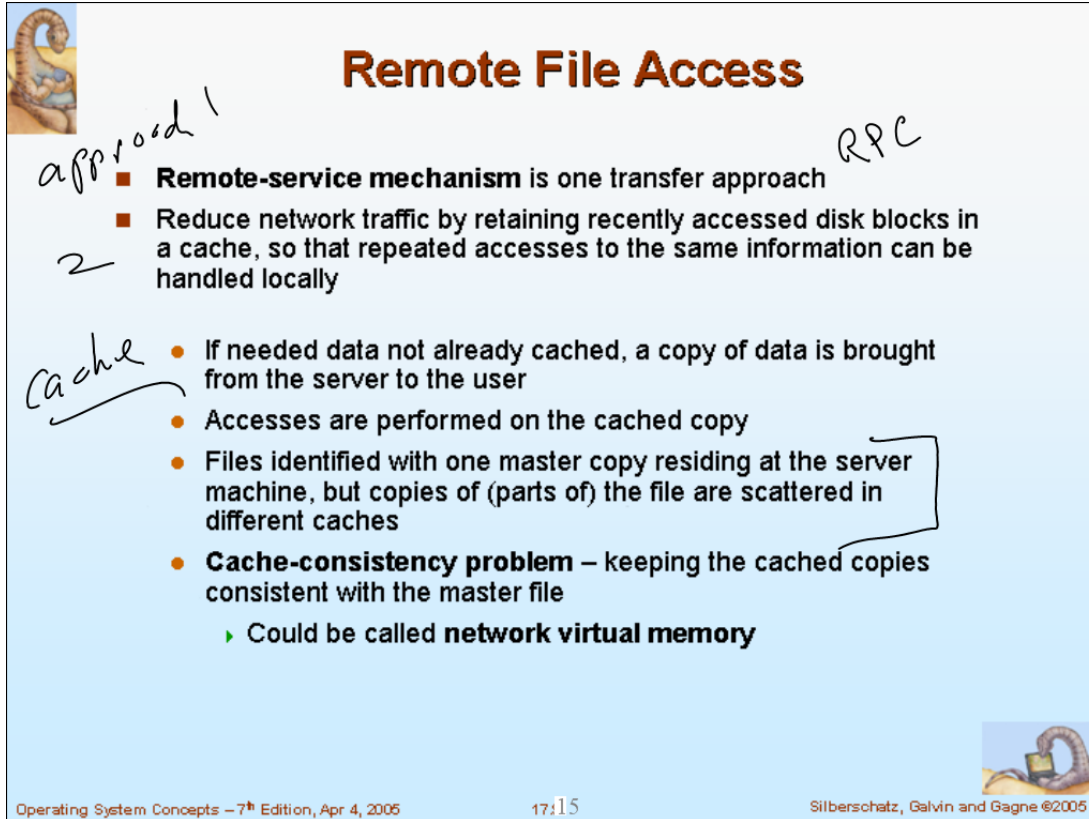


Naming Schemes — Three Main Approaches

- Files named by combination of their host name and local name; guarantees a unique systemwide name *r cp*
scp
- Attach remote directories to local directories, giving the appearance of a coherent directory tree; only previously mounted remote directories can be accessed transparently *NFS mount*
automount
- 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

Operating System Concepts – 7th Edition, Apr 4, 2005 17:14 Silberschatz, Galvin and Gagne ©2005

Panel 15



Remote File Access

approach!

- **Remote-service mechanism** is one transfer approach *RFC*
- Reduce network traffic by retaining recently accessed disk blocks in a cache, so that repeated accesses to the same information can be handled locally

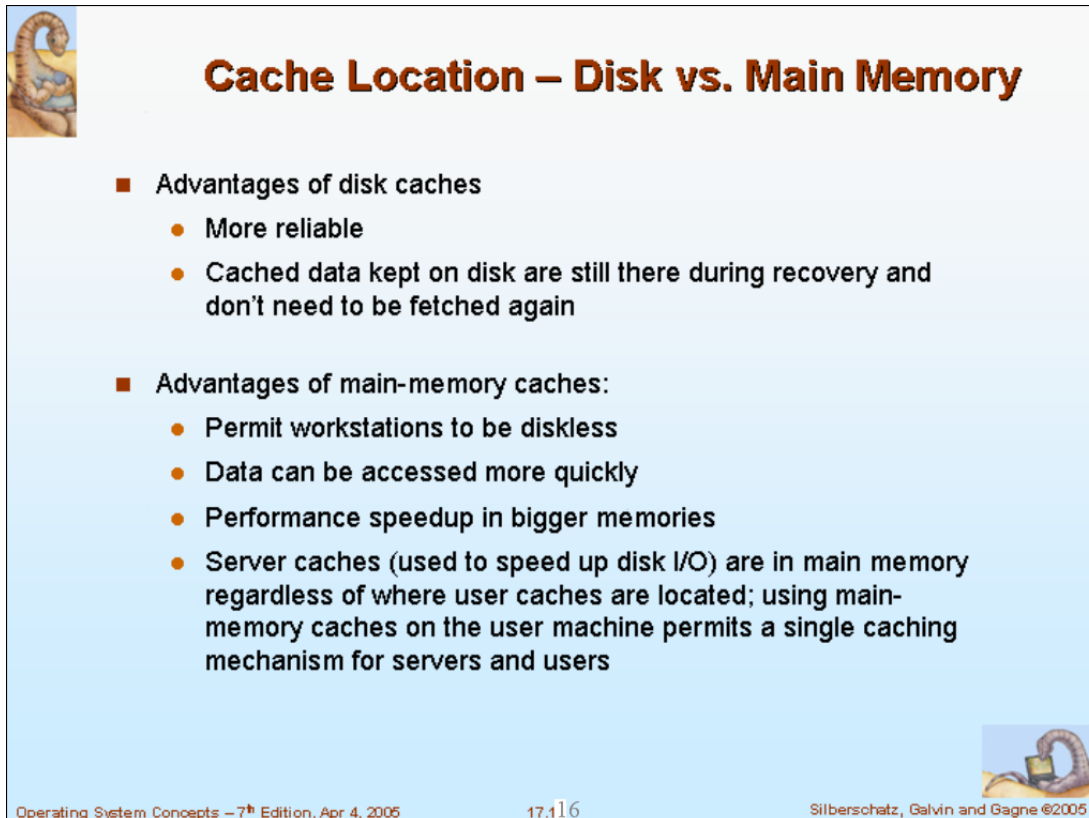
2

Cache

- 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
 - ▶ Could be called **network virtual memory**

Operating System Concepts – 7th Edition, Apr 4, 2005 17.15 Silberschatz, Galvin and Gagne ©2005

Panel 16



Cache Location – Disk vs. Main Memory

- 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

Operating System Concepts – 7th Edition, Apr 4, 2005 17.16 Silberschatz, Galvin and Gagne ©2005

Panel 17



Cache Update Policy

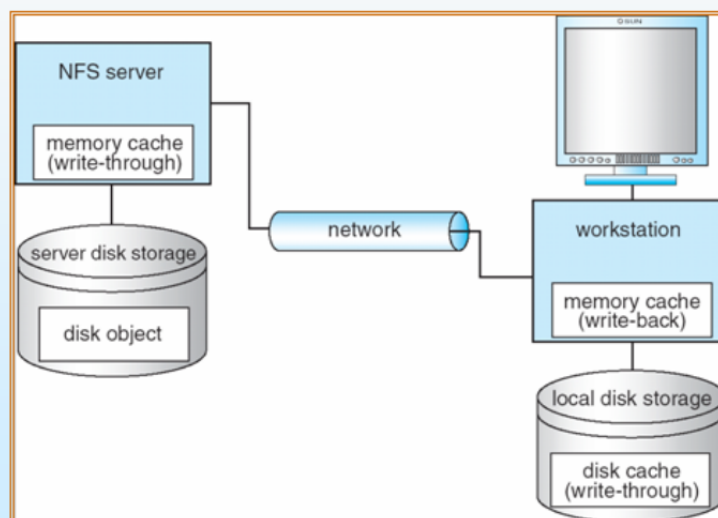
- **Write-through** – write data through to disk as soon as they are placed on any cache
 - Reliable, but poor performance
- **Delayed-write** – modifications written to the cache and then written through to the server later
 - Write accesses complete quickly, some data may be overwritten before they 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



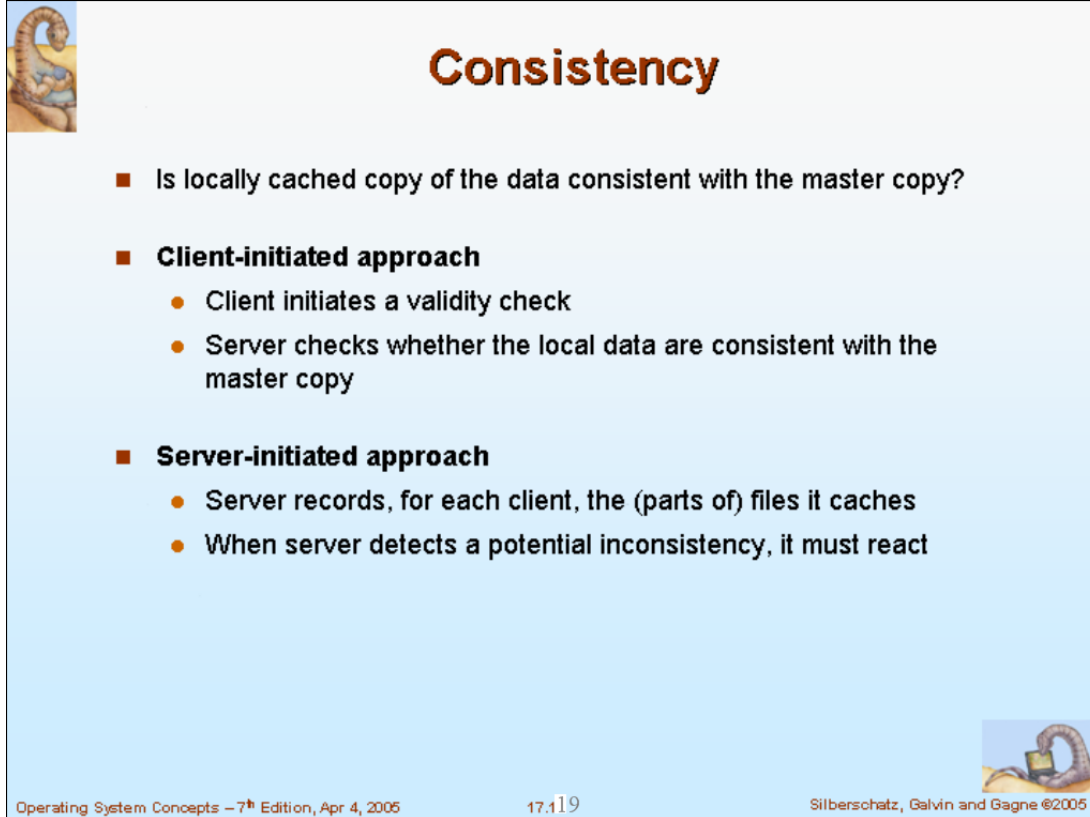
Panel 18



Caches and its Use of Caching



Panel 19

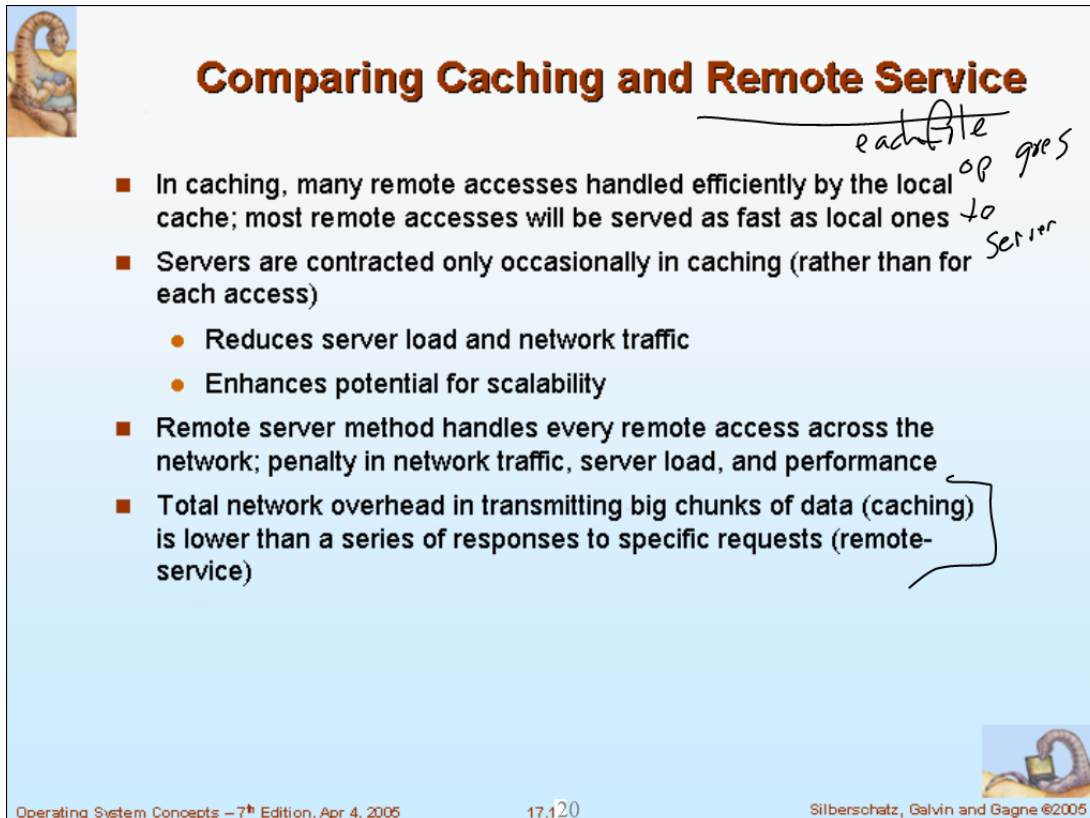


Consistency

- 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

Operating System Concepts – 7th Edition, Apr 4, 2005 17.19 Silberschatz, Galvin and Gagne ©2005

Panel 20




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)

Operating System Concepts – 7th Edition, Apr 4, 2005 17.20 Silberschatz, Galvin and Gagne ©2005


Handwritten notes:
 - Above the first bullet: each file
 - Between the first and second bullets: *op goes to server*
 - A bracket groups the last two bullets.

Panel 21




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 intermachine interface is different from the upper user interface
- In remote-service, the intermachine interface mirrors the local user-file-system interface




Operating System Concepts – 7th Edition, Apr 4, 2005 17.121 Silberschatz, Galvin and Gagne ©2005

Panel 22



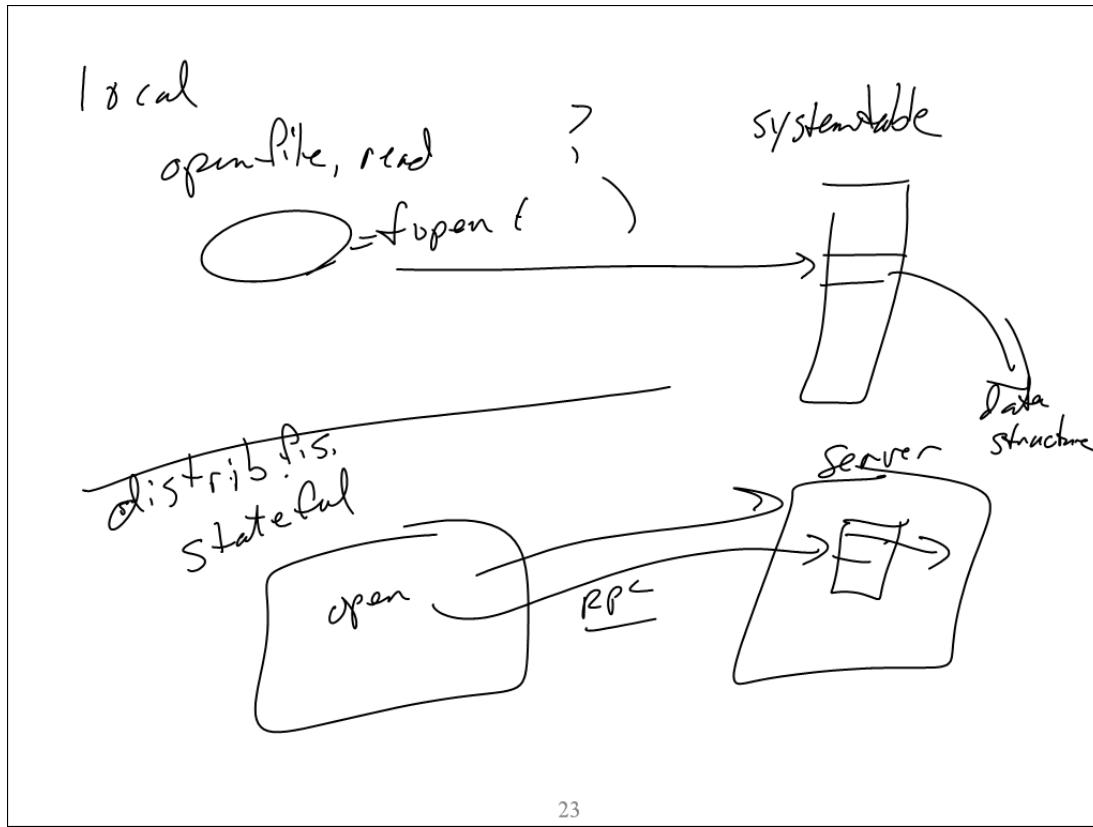
Stateful File Service

- 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



Operating System Concepts – 7th Edition, Apr 4, 2005 17.122 Silberschatz, Galvin and Gagne ©2005

Panel 23



Panel 24



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



Panel 25


NFS stateless

NFS v4 stateful
mount protocol

locking


25

Panel 26



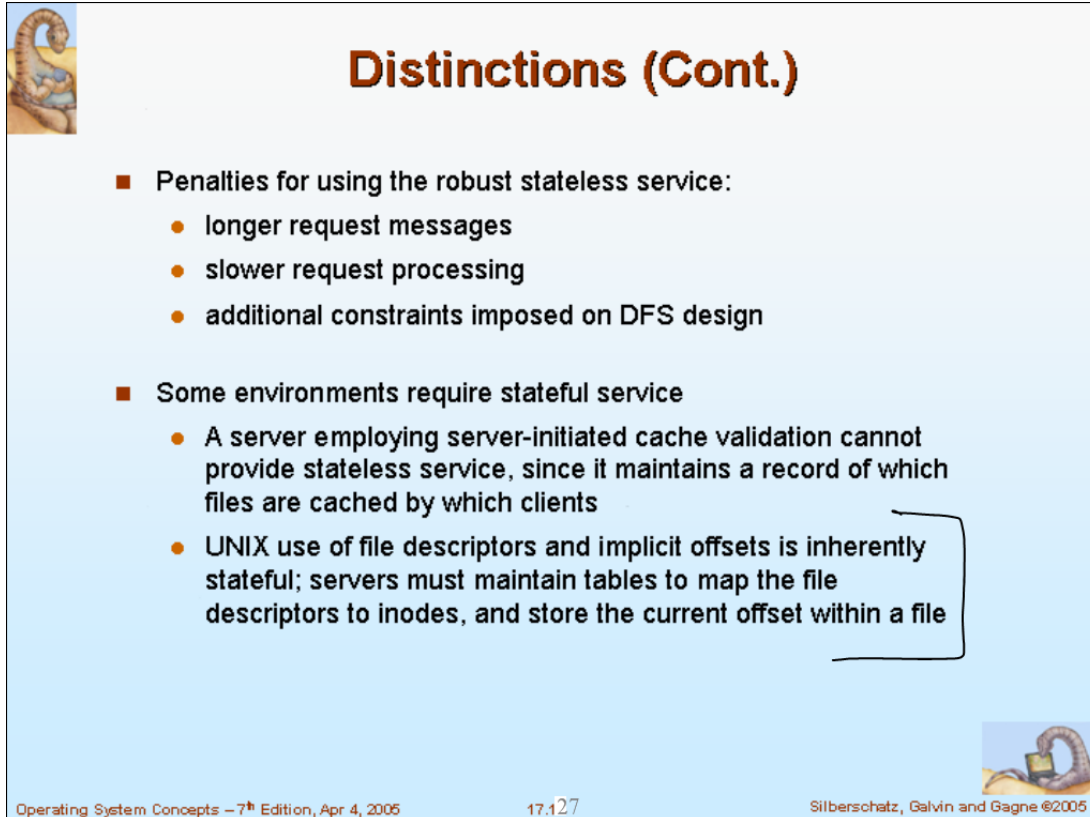
Distinctions Between Stateful & Stateless Service

- 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 failure and recovery are almost unnoticeable
 - ▶ A newly reincarnated server can respond to a self-contained request without any difficulty



Operating System Concepts – 7th Edition, Apr 4, 2005 17.126 Silberschatz, Galvin and Gagne ©2005

Panel 27

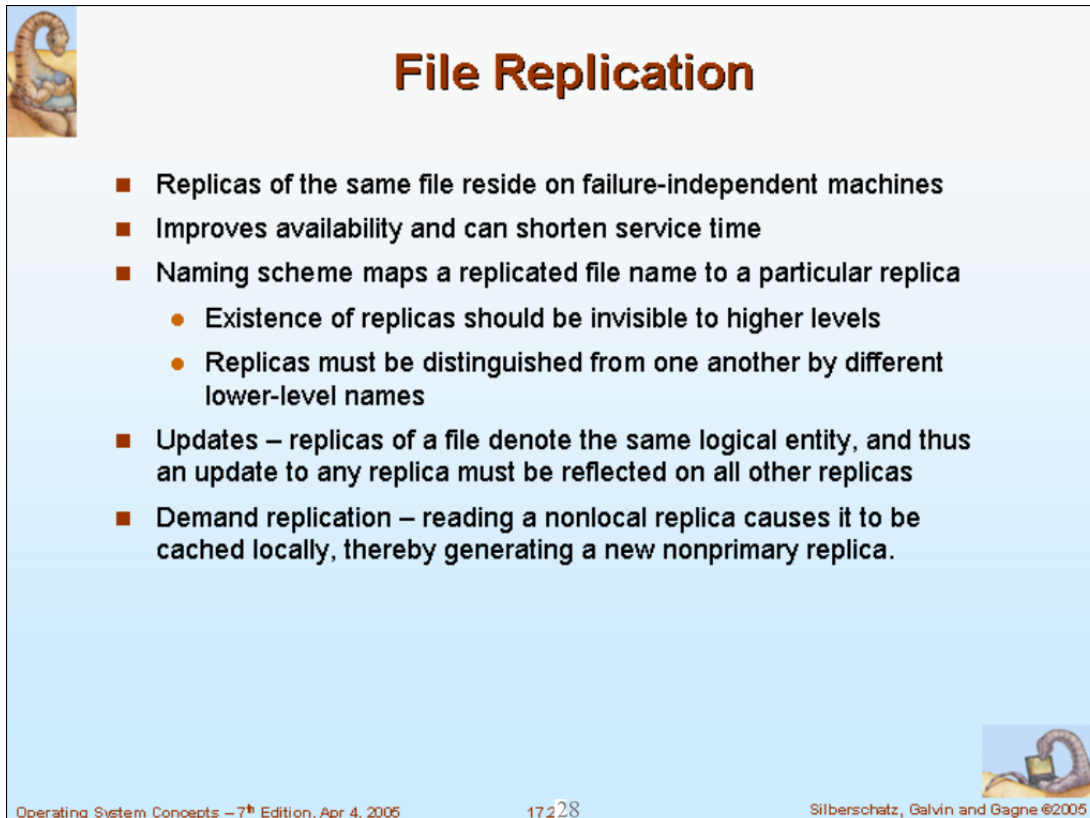


Distinctions (Cont.)

- 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

Operating System Concepts – 7th Edition, Apr 4, 2005 17.127 Silberschatz, Galvin and Gagne ©2005

Panel 28



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 nonlocal replica causes it to be cached locally, thereby generating a new nonprimary replica.

Operating System Concepts – 7th Edition, Apr 4, 2005 17.228 Silberschatz, Galvin and Gagne ©2005

Panel 29



An Example: AFS

- A distributed computing environment (Andrew) under development since 1983 at Carnegie-Mellon University, purchased by IBM and released as **Transarc DFS**, now open sourced as OpenAFS
- AFS tries to solve complex issues such as uniform name space, location-independent file sharing, client-side caching (with cache consistency), secure authentication (via Kerberos)
 - Also includes server-side caching (via replicas), high availability
 - Can span 5,000 workstations



Panel 30

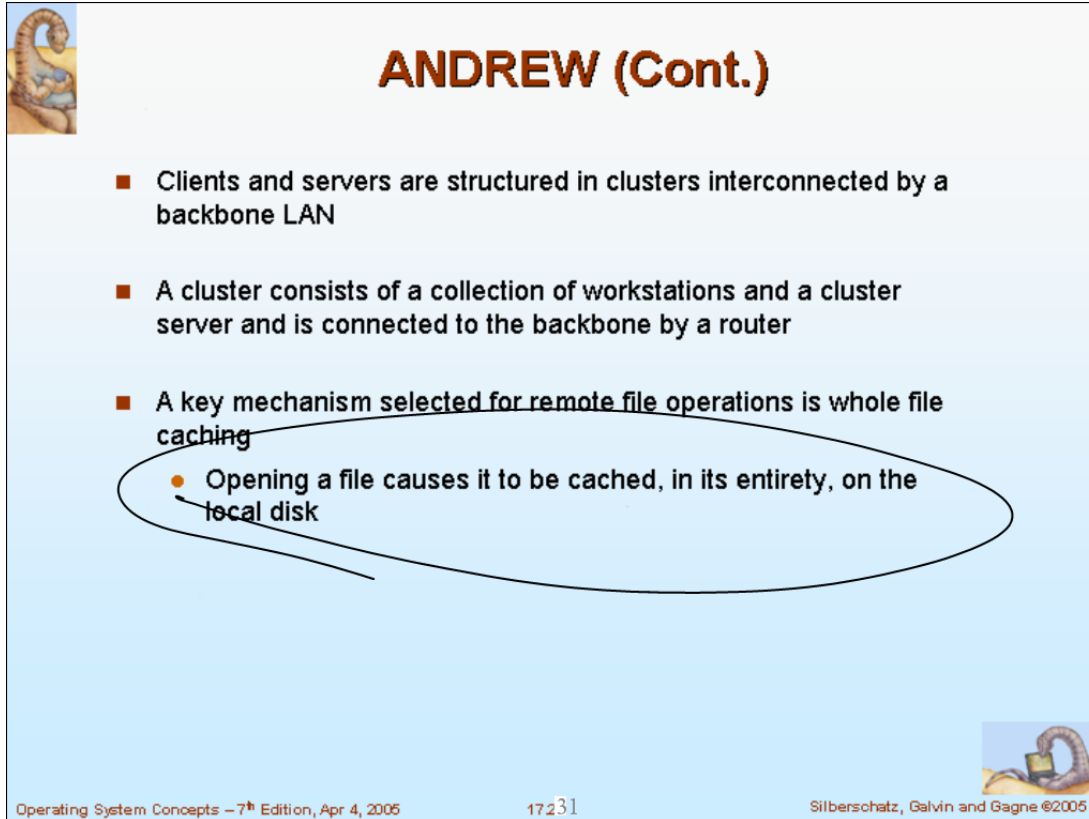


ANDREW (Cont.)

- Clients are presented with a partitioned space of file names: a **local name space** and a **shared name space**
- Dedicated servers, called *Vice*, present the shared name space to the clients as an homogeneous, identical, and location transparent file hierarchy
- The local name space is the root file system of a workstation, from which the shared name space descends *mount*
- Workstations run the *Virtue* protocol to communicate with Vice, and are required to have local disks where they store their local name space
- Servers collectively are responsible for the storage and management of the shared name space



Panel 31

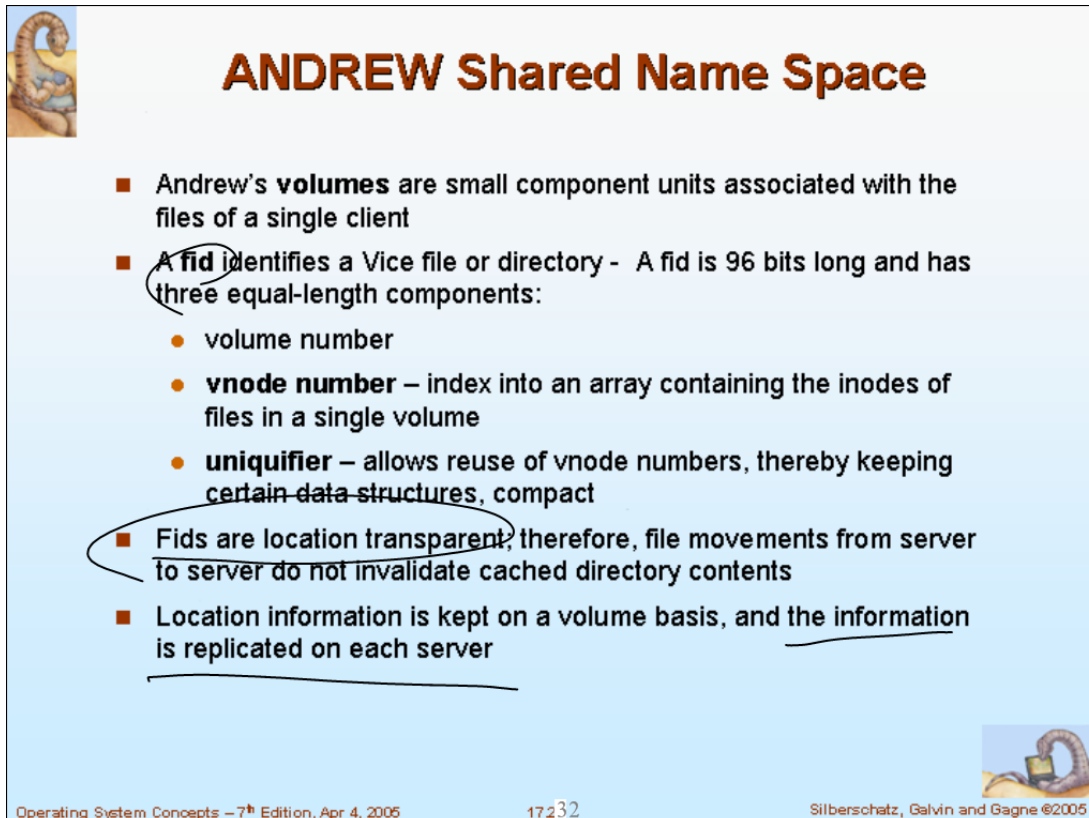


ANDREW (Cont.)

- Clients and servers are structured in clusters interconnected by a backbone LAN
- A cluster consists of a collection of workstations and a cluster server and is connected to the backbone by a router
- A key mechanism selected for remote file operations is whole file caching
 - Opening a file causes it to be cached, in its entirety, on the local disk

Operating System Concepts – 7th Edition, Apr 4, 2005 17231 Silberschatz, Galvin and Gagne ©2005

Panel 32

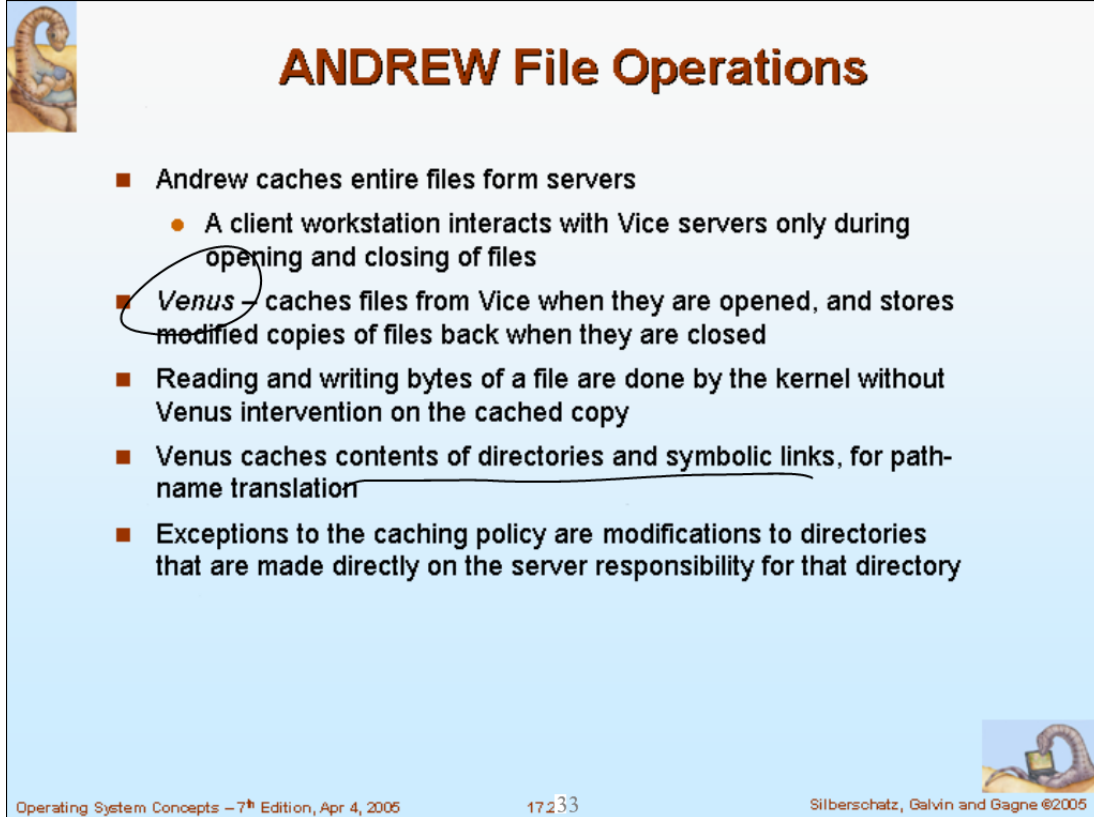


ANDREW Shared Name Space

- Andrew's **volumes** are small component units associated with the files of a single client
- A **fid** identifies a Vice file or directory - A fid is 96 bits long and has three equal-length components:
 - volume number
 - **vnode number** – index into an array containing the inodes of files in a single volume
 - **uniquifier** – allows reuse of vnode numbers, thereby keeping ~~certain data structures~~, compact
- Fids are location transparent; therefore, file movements from server to server do not invalidate cached directory contents
- Location information is kept on a volume basis, and the information is replicated on each server

Operating System Concepts – 7th Edition, Apr 4, 2005 17232 Silberschatz, Galvin and Gagne ©2005

Panel 33

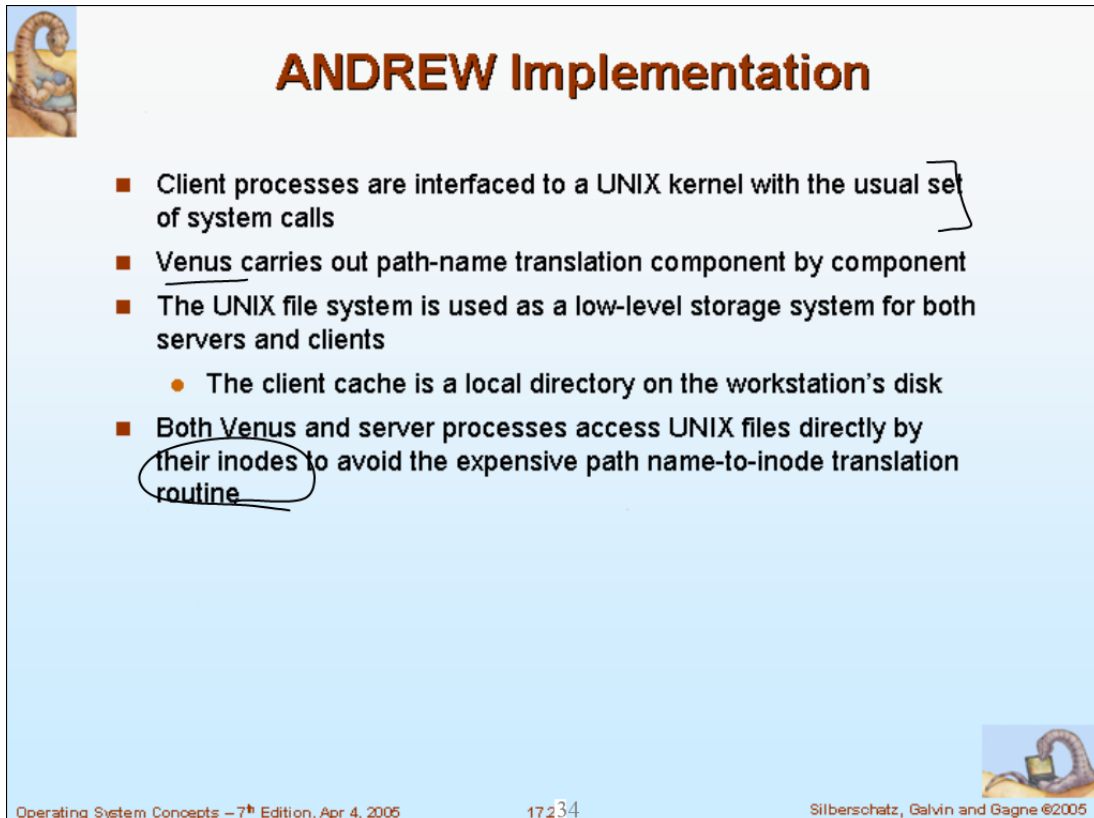


ANDREW File Operations

- Andrew caches entire files from servers
 - A client workstation interacts with Vice servers only during opening and closing of files
- Venus caches files from Vice when they are opened, and stores modified copies of files back when they are closed
- Reading and writing bytes of a file are done by the kernel without Venus intervention on the cached copy
- Venus caches contents of directories and symbolic links, for path-name translation
- Exceptions to the caching policy are modifications to directories that are made directly on the server responsibility for that directory

Operating System Concepts – 7th Edition, Apr 4, 2005 17233 Silberschatz, Galvin and Gagne ©2005

Panel 34



ANDREW Implementation

- Client processes are interfaced to a UNIX kernel with the usual set of system calls
- Venus carries out path-name translation component by component
- The UNIX file system is used as a low-level storage system for both servers and clients
 - The client cache is a local directory on the workstation's disk
- Both Venus and server processes access UNIX files directly by their inodes to avoid the expensive path name-to-inode translation routine

Operating System Concepts – 7th Edition, Apr 4, 2005 17234 Silberschatz, Galvin and Gagne ©2005

Panel 35



ANDREW Implementation (Cont.)

- Venus manages two separate caches:
 - one for status
 - one for data
- LRU algorithm used to keep each of them bounded in size
- The status cache is kept in virtual memory to allow rapid servicing of *stat* (file status returning) system calls
- The data cache is resident on the local disk, but the UNIX I/O buffering mechanism does some caching of the disk blocks in memory that are transparent to Venus



Panel 36

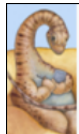


Fig. 17.01

