Chapter 18 – Distributed Systems and Web Services

Outline

Chapter 18 – Distributed Systems and Web Services

Outline (cont.)

Objectives

- After reading this chapter, you should understand:
  - characteristics and examples of networked and distributed file systems.
  - types, benefits and examples of clustering.
  - the peer-to-peer distributed computing model.
  - grid computing.
  - Java distributed computing technologies.
  - Web services and the Microsoft .NET and Sun ONE platforms.

18.1 Introduction

- File sharing
  - Accessing files stored on file servers using a distributed file system is similar to accessing files stored on the user's local computer
- Distributed file systems
- Clustering
  - Takes advantage of distributed systems and parallel systems to build powerful computers
- Peer-to-peer distributed computing model
  - Used to remove many central points of failure in applications like instant messengers
- Grid computing
  - Exploits unused computer power to solve complex problems

18.2 Distributed File Systems

- Networked file systems
  - Allow clients to access files stored on remote computers
- Distributed file systems
  - Special examples of networked file systems that allow transparent access to remote files

18.2.1 Distributed File System Concepts

- Distributed file server
  - Can be either stateful or stateless
  - Stateful system
    - The server keeps state information of the client requests so that subsequent access to the file is easier
  - Stateless system
    - The client must specify which file to access in each request
- Distributed file systems provide the illusion of transparency
  - Complete file location transparency means that the user is unaware of the physical location of a file within a distributed file system
  - The user sees only a global file system

18.2.1 Distributed File System Concepts

- Many distributed systems implement client caching to avoid the overhead of multiple RPCs
  - Clients keep a local copy of a file and flush modified copies of it to the server from time to time
  - Because there are multiple copies of the same file, files can become inconsistent
- Distributed file systems are designed to share information among large groups of computers
  - New computers should be able to be added to the system easily
  - Distributed file systems should be scalable
- Security concerns in distributed file systems:
  - Ensuring secure communications
  - Access control

18.2.2 Network File System (NFS)

- NFS versions 2 and version 3
  - Assume a stateless server implementation
  - Fault tolerance easier to implement than with a stateful server
  - With stateless servers, if the server crashes, the client can simply retry its request until the server responds
- NFS-4
  - Stateful
  - Allows faster access to files
  - However, if the server crashes, all the state information of the client is lost, so the client needs to rebuild its state on the server before retrying

18.2.2 Network File System (NFS)

- Client-side caching is essential to efficient distributed file system operation
  - NFS-4 extends the client-caching scheme through delegation
    - Allows the server to temporarily transfer the control of a file to a client
    - When the server grants a read delegation of a particular file to the client, then no other client can write to that file
    - When the server grants a write delegation of a particular file to the client, then no other client can read or write to that file
    - If another client requests a file that has been delegated, the server will revoke the delegation and request that the original client flush the file to disk

18.2.2 Network File System (NFS)

Figure 18.1 NFS architecture.

18.2.3 Andrew File System (AFS)

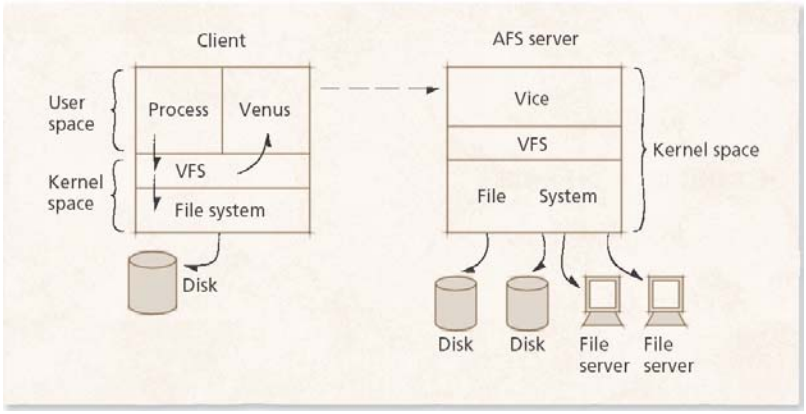
- AFS overview
  - Global file system
  - Appears as a branch of a traditional UNIX file system at each workstation
  - Completely location transparent
  - Provides a high degree of availability
    - Files are always available regardless of the user's location
  - Based on the client-server model
  - Relies on RPCs for communication

18.2.3 Andrew File System (AFS)

- AFS implementation
  - Vice governs access to distributed files
  - Venus is a user process that runs on each client and interacts with Vice to access files
  - AFS uses callbacks to notify clients that files are no longer valid
    - Client must invalidate its file and request the most recent version
  - Provides a high degree of availability
    - Files are always available regardless of the user's location
  - File identifiers (fids) and volumes improve AFS efficiency
  - Cells enable multiple sites to implement custom access control

18.2.3 Andrew File System (AFS)

Figure 18.2 AFS structure.

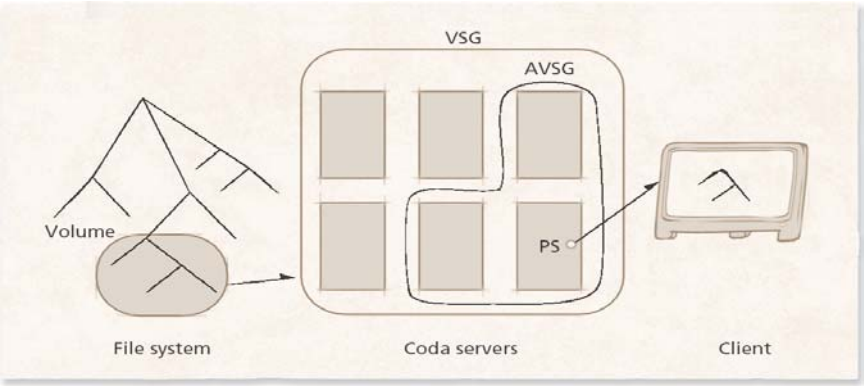


18.2.4 Coda File System

- Coda
  - Developed to address AFS's lack of fault tolerance
  - Uses local nonvolatile storage (e.g., disks) as file caches
  - Uses callbacks to keep clients up to date
  - Volumes
    - Logical pieces of the file system
    - Replicated physically across multiple file servers
  - Volume storage group (VSG)
    - Servers that hold the same volume
  - Available volume storage group (AVSG)
    - Members of the VSG with which the client can connect and communicate
    - Coda uses AVSGs to implement fault tolerance

18.2.4 Coda File System

Figure 18.3 Coda volume structure.

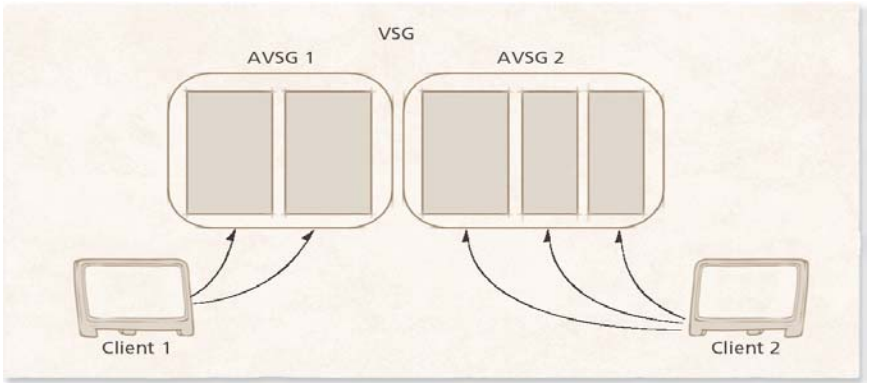


18.2.4 Coda File System

- Addressing consistency in Coda
  - If a directory is modified, the Coda servers deal with the problem
    - Each member of the VSG locks the volume
    - A leader is chosen to process the update
  - For files
    - When connected to Coda, clients cache files so they can be accessed when disconnected (hoarding stage)
    - When disconnected, clients enter the emulation stage where all file requests are serviced from the cache, if the file is resident (error otherwise)
    - When reconnected, file updates are sent to the server asynchronously (reintegration stage)

18.2.4 Coda File System

Figure 18.4 File consistency issues in mutually exclusive AVSGs.

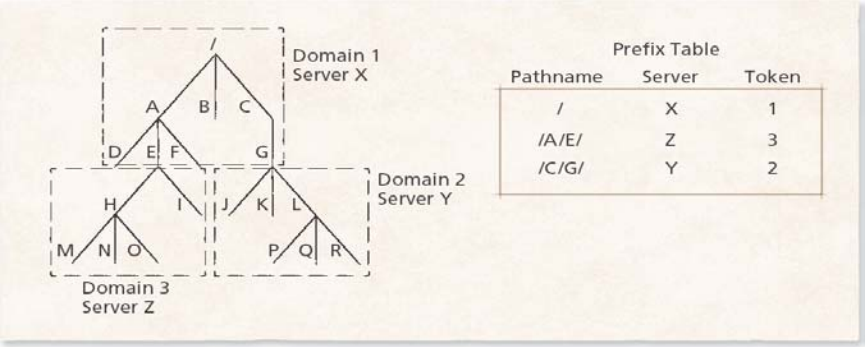


18.2.5 Sprite File System

- Sprite file system characteristics
  - Emulates a UNIX file system
  - Every client has the exact same view of the hierarchy
  - Goes a step further in UNIX file system emulation by allowing transparent remote access to I/O devices (which are represented as files in UNIX)

18.2.5 Sprite File System

Figure 18.5 Sprite file system domains.

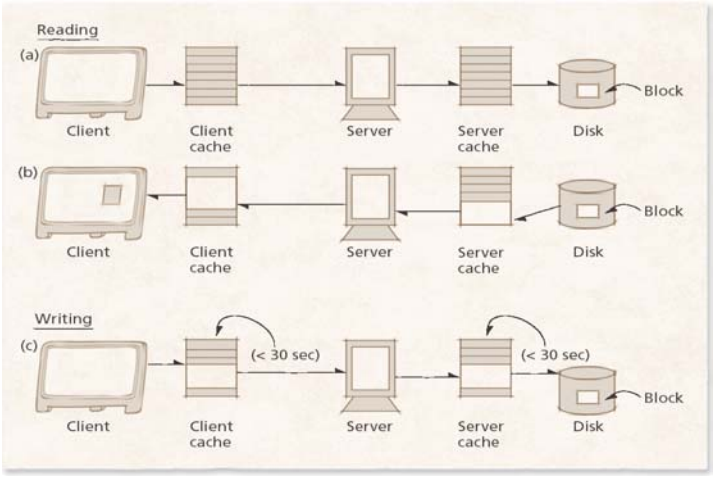


18.2.5 Sprite File System

- Both the client and the server maintain a cache
  - To open a file, the client first checks its cache, then makes a request to its backing store (which is generally a server)
  - If the server is unable to satisfy the request from its cache, it reads the data from disk
  - Both caches retain a copy of the data when it is returned to the client
  - When a client writes a block, it writes into its cache
  - Sprite flushes updated pages to the server every 30 seconds

18.2.5 Sprite File System

Figure 18.6 Reading and writing in the Sprite file system.



18.3 Multicomputer Systems

- Multicomputer systems
  - Do not share a common memory or bus (although they may share a virtual memory)
  - Each processing unit has access to its own resources
  - Independent units are connected in a network to operate cooperatively to form a multicomputer system
- Distributed processing
  - Makes large computations feasible
  - Power comes at the cost of simplicity
  - Must take into account synchronization, mutual exclusion and deadlock
  - Networking and the lack of shared memory make managing such issues more complex

18.4 Clustering

- Clustering
  - Interconnecting nodes (single-processor computers or multiprocessor computers) within a high-speed LAN to function as a single parallel computer
- Cluster: Set of nodes that forms the single parallel machine
  - Enables multiple computers to work together to solve large and complex problems
    - For example, weather prediction

18.4.1 Clustering Types

- High-performance clusters
  - All nodes in the cluster work to improve performance
- High-availability clusters
  - Only some of the nodes in the cluster are active while others serve as backups
  - If working nodes or their components fail, the backup nodes immediately start running and take over the jobs that were being executed on the failed nodes, without interrupting service
- Load-balancing clusters
  - A particular node works as a load balancer to distribute the load to a set of nodes so that all hardware is utilized efficiently

18.4.2 Clustering Benefits

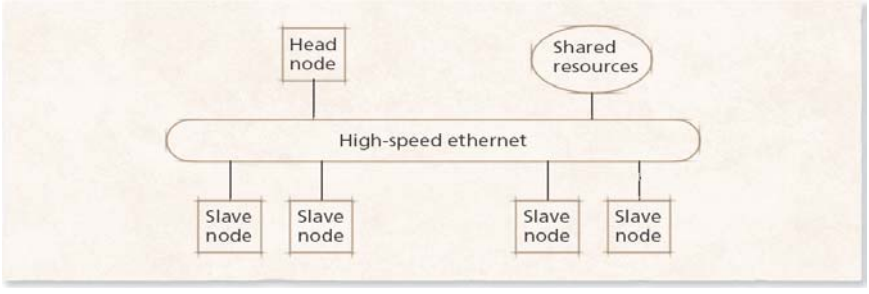
- Economically interconnects relatively inexpensive components
  - Reduces the cost for building a clustered system compared to a single parallel computer with the same capability
- High performance
  - Each node in the clustering system shares the workload
- Fast communications among nodes in a cluster
  - Faster than those in unclustered distributed computing systems due to the high-speed LAN between nodes

18.4.2 Clustering Benefits

- Can provide replication of resources across the nodes
  - The failure of any one computer will not affect the availability of the rest of the system's resources
- Scalability
  - A cluster is able to add or remove nodes (or the components of nodes) to adjust its capabilities without affecting the existing nodes in the cluster
  - Better scalability than multiprocessors
- Reliability and fault tolerance by providing backups for the services and resources

18.4.3 Clustering Examples

- Beowulf
  - Linux systems interconnected with high-speed Ethernet
  - Head node (also called a master node) acts as a server
    - Distributes the work load
    - Controls access to the cluster
    - Handles the shared resources
  - All other nodes in the cluster are often called slave nodes



18.4.3 Clustering Examples

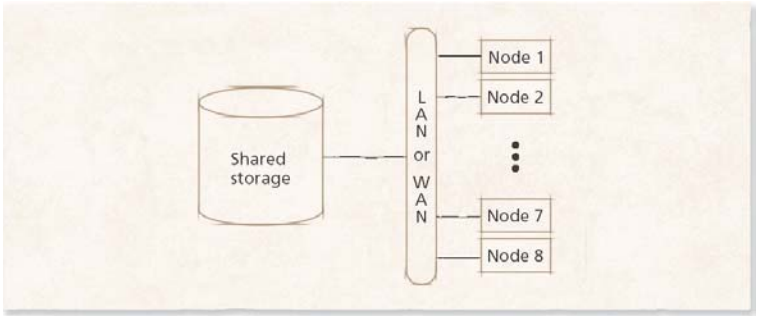
Figure 18.7 Typical Beowulf cluster.

18.4.3 Clustering Examples

- Windows Server 2003
  - High-availability cluster
    - Nodes can be connected either in a LAN or a WAN
    - All nodes may share a storage device or each node may have a local storage
  - Load-balancing cluster
    - Usually the nodes are interconnected with high-speed Ethernet
    - Does not require sharing storage, because each node can do its job independently

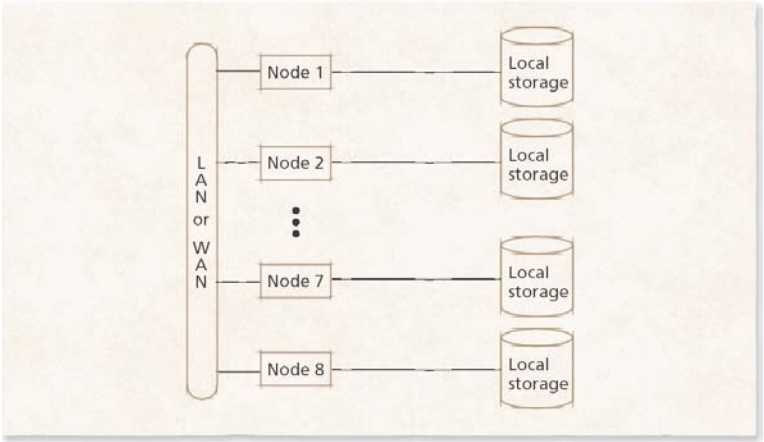
18.4.3 Clustering Examples

Figure 18.8 High-availability cluster with shared storage.



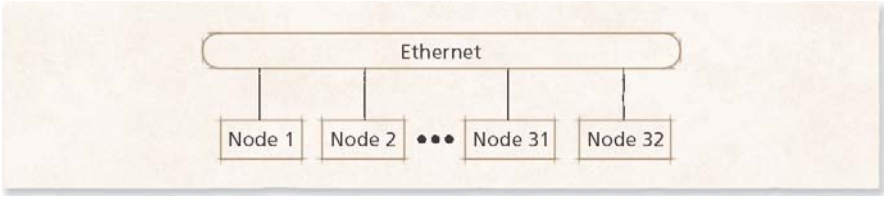
18.4.3 Clustering Examples

Figure 18.9 High-availability cluster with local storage.



18.4.3 Clustering Examples

Figure 18.10 Load-balancing cluster with 32 nodes.



18.5 Peer-to-Peer Distributed Computing

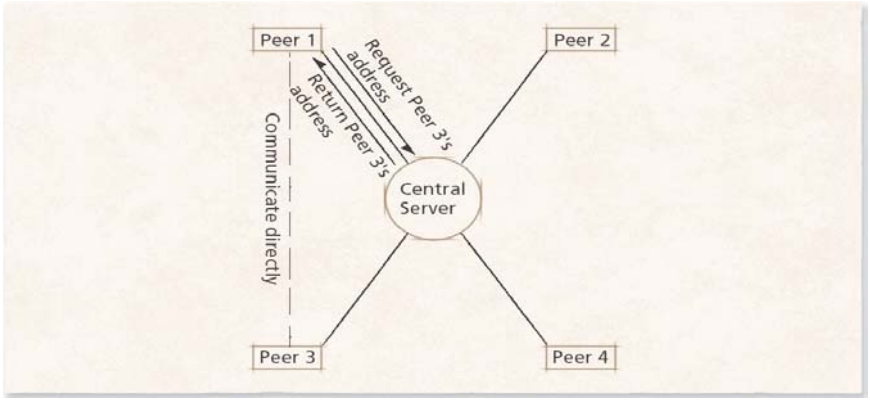
- Peer
  - A single computer in a P2P system
  - Each performs both client and server functions
- P2P applications
  - Distribute processing responsibilities and information to many computers
  - Reclaim otherwise wasted computing power and storage space
  - Eliminates many central points of failure

18.5.1 Client/Server and Peer-to-Peer Applications

- P2P (peer-to-peer) applications
  - Instead of segregating computers by function, all computers act as both clients and servers
  - Each peer has the ability to discover and communicate with other peers
  - Peers may share resources (such as large multimedia files) with others

18.5.2 Centralized vs. Decentralized P2P Applications

- Centralized P2P application
  - Uses a server that connects to each peer
  - Exemplify the client/server relationship

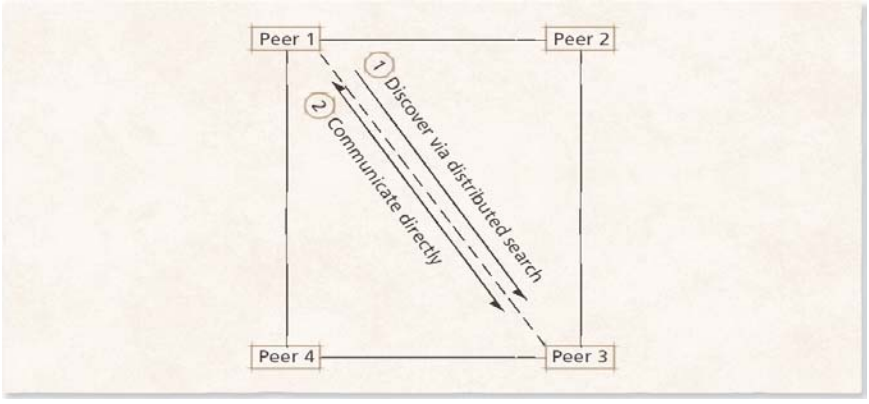


18.5.2 Centralized vs. Decentralized P2P Applications

Figure 18.11 Centralized P2P instant-messaging application.

18.5.2 Centralized vs. Decentralized P2P Applications

- Decentralized P2P application
  - Also called a pure P2P application
  - Does not have a server
    - Does not suffer from the same deficiencies as applications that depend on servers



18.5.2 Centralized vs. Decentralized P2P Applications

Figure 18.12 Pure P2P instant-messenging application.

18.5.2 Centralized vs. Decentralized P2P Applications

Figure 18.13 Common P2P applications.

Distributed Application	Description
Gnutella	A P2P technology used to share documents on the Internet. Gnutella does not use any servers. There is no authentication, and peers search for files via a distributed search mechanism.[59] (Section 18.5.3, Peer Discovery and Searching, overviews this mechanism.)
KaZaA	A file sharing application that is a hybrid between Gnutella and centralized applications. A server authenticates all users. Certain peers serve as search hubs, which catalog the files of peers connected to them. Searches are distributed to each search hub, which then responds with results that allow direct connections for file transfers.[60]
Groove	A P2P system that allows users to communicate, collaborate and share documents on the Internet and intranet. Groove provides secure communication because users are authenticated and private data is not shared with third parties.[61]
Freenet	Decentralized P2P technology that allows users to share documents on the Internet without fear of censorship. Freenet does not have a central server to govern access. Instead, access to Freenet is anonymous. Documents stored in Freenet are encrypted to improve protection.[62]
Instant messenging	P2P application that enables users to send short text messages and files to one another. Most instant messengers use servers that authenticate all users and route messages between peers.

18.5.3 Peer Discovery and Searching

- Peer discovery
  - The act of finding peers in a P2P application
  - Pure P2P applications often suffer slow peer discovery and information searching due to the lack of a central server
- Distributed searches
  - Make networks more robust by removing single points of failure, such as servers
  - Not only can peers find information in this way, but they can search for other peers via distributed searches

18.5.4 JXTA

- A set of standard, low-level platform and language-independent protocols
- Promotes interoperability among peer-to-peer applications
- Provides a foundation on which developers can build any type of P2P application

18.5.4 JXTA

Figure 18.14 JXTA low-level protocols.

Protocol	*Function*
Peer discovery	Peers use this protocol to find other entities in the JXTA network by searching for advertisements.
Peer resolver	Peers that help a search process (e.g., send and process requests) implement this protocol.
Peer information	Peers obtain information about other peers via this protocol.
Peer membership	Peers use this protocol to learn about the requirements of groups, how to apply for membership, how to modify their membership and how to quit a group. Authentication and security are implemented through this protocol.
Pipe binding	Peers can connect pipes to one another, via advertisements, through this protocol.
Endpoint routing	Peer routers implement this protocol to provide other routing services to other peers (e.g., tunneling through a firewall).

18.6 Grid Computing

- Grid computing
  - Links computational resources that are distributed over the wide area network (such as computers, data storages and scientific devices) to solve complex problems
  - Resources are distributed
  - Users can access these resources transparently

18.6 Grid Computing

- Grid computing (cont.)
  - Emphasizes public collaboration
    - Individuals and research institutes coordinate resources that are not subject to a centralized control to deliver qualities of service
  - Grid computing has the same advantage as clustering
    - High performance, achieved by cost-effectively using spare computer power and collaborating resources
  - However, grid computing requires advanced software to manage distributed computing tasks efficiently and reliably

18.7 Java Distributed Computing

- Java
  - Used widely to implement distributed systems
  - Tools for developing distributed systems
    - Java's RMI
    - CORBA
    - Servlets
    - JavaServer Pages
    - Jini
    - JavaSpaces
    - JMI

18.7.1 Java Servlets and JavaServer Pages (JSP)

- Java servlet
  - Extends the functionality of a server, most frequently a Web server
  - Commonly used when a small portion of the content sent to the client is static text or markup
  - Some servlets do not produce content at all
    - They perform a task on behalf of the client (such as a database query), then invoke other servlets or JSPs to provide a response

18.7.1 Java Servlets and JavaServer Pages (JSP)

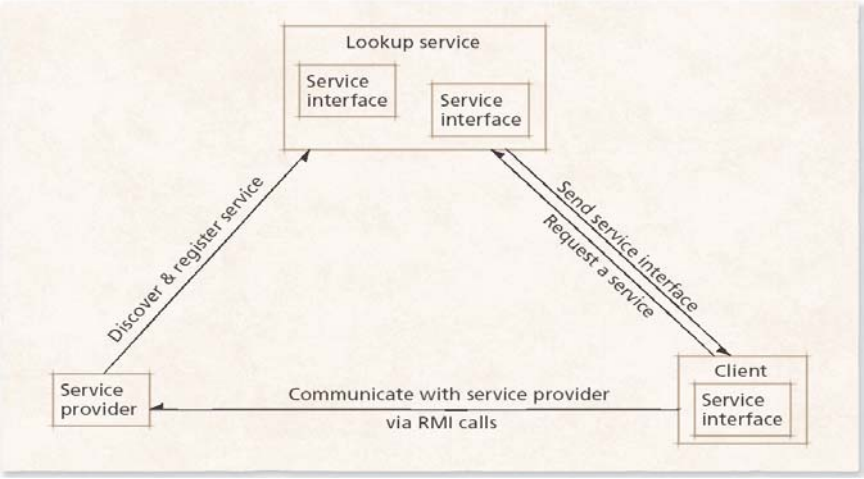
- JavaServer Pages
  - Allow Web-page programmers to create pages that use encapsulated Java functionality and even to write scriptlets (Java code embedded in a JSP) of actual Java code directly in the page
  - Generally used when most of the content sent to the client is static text and markup and only a small portion of the content is generated dynamically with Java code

18.7.2 Jini

- Jini
  - Framework for building reliable and fault-tolerant distributed systems with existing Java technologies
  - Extends the idea of providing services beyond computer-based networks and into home-based networks

18.7.2 Jini

Figure 18.15 Jini architecture.

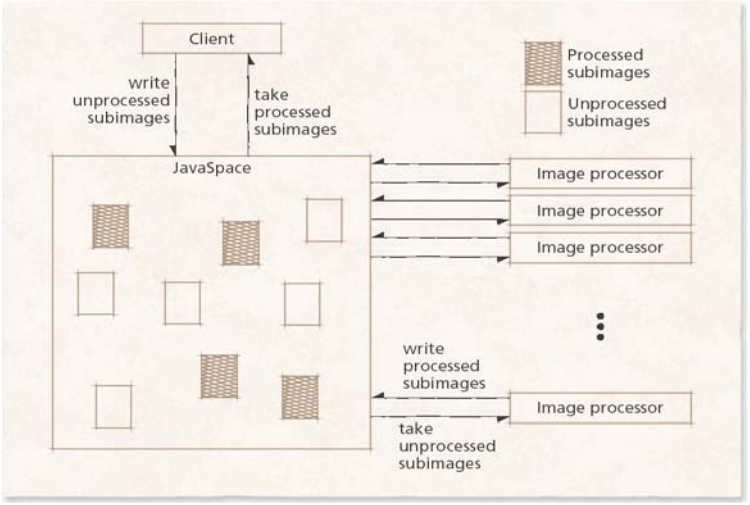


18.7.3 JavaSpaces

- Javaspaces
  - Jini service that implements a simple, high-level architecture for building distributed systems
  - Provides distributed, shared storage (and shared memory) for Java objects
  - Enables Java objects to communicate, share objects and coordinate tasks using the storage
  - Any Java-compatible client can put shared objects into the storage

18.7.3 JavaSpaces

Figure 18.16 Distributed image-processing application using JavaSpaces.

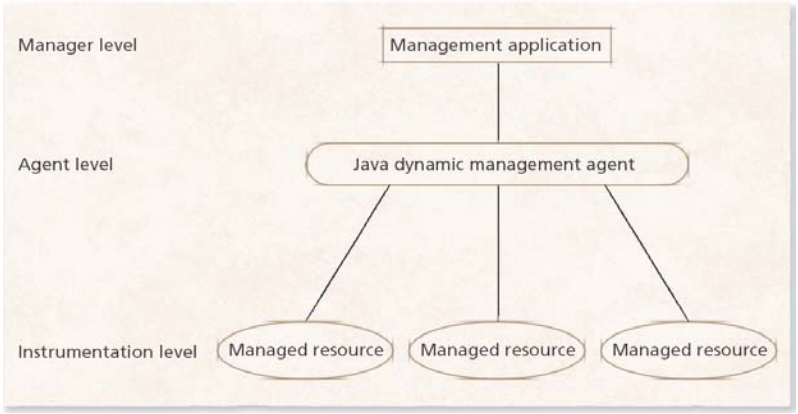


18.7.4 Java Management Extensions (JMX)

- JMX
  - Component framework that enables developers to build automated, intelligent and dynamic network-management solutions
  - Defines a three-level management architecture
    - Instrumentation level
      - Makes any Java-based object manageable so that the management application can access and operate these objects
    - Agent level
      - Provides services for exposing the managed resources
    - Manager level
      - Gives management applications access to resources created in the instrumentation level
      - Operates these resources via the JMX agents

18.7.4 Java Management Extensions (JMX)

Figure 18.17 JMX's three-level management architecture.



18.8 Web Services

- Web services
  - Encompass a set of related standards that can enable computer applications to communicate and exchange data via the Internet
  - Technology and platform independent
  - Improve collaborative software development by allowing developers to create applications by combining code written in any language on any platform

18.8 Web Services

- Web services (cont.)
  - Promote modular programming
    - Each specific function in an application can be exposed as a separate Web service
    - Individuals or businesses can create their own unique applications by mixing and matching Web services that provide the functionality they need
    - Less error prone
    - Promotes software reuse

18.8.1 Microsoft's .NET Platform

- The .NET initiative
  - Includes the Visual Studio .NET integrated development environment
  - Enables programmers to develop Web services in a variety of languages, including:
    - C++
    - C#
    - Visual Basic .NET
  - However, .NET technologies are available only for Windows 2000 and XP
- Extend the concept of software reuse to the Internet by allowing developers to reuse software components that reside on other machines or platforms

18.8.2 Sun Microsystems and the Sun ONE Platform

- Sun ONE
  - Model for software development
    - Critical business information and applications are available at any time to any type of device, including cell phones and PDAs
  - Incorporates support for open standards
    - XML, SOAP, UDDI and WSDL,
    - Helps ensure high levels of interoperability and system integration