Distributed Processing, Client/Server, and Clusters

Chapter 13

Client/Server Computing

- Client machines are generally single-user PCs or workstations that provide a highly user-friendly interface to the end user.
- Each server provides a set of shared user services to the clients.
- The server enables many clients to share access to the same database and enables the use of a high-performance computer system to manage the database.
Client/Server Applications

- Basic software is an operating system running on the hardware platform
- Platforms and the operating systems of client and server may differ
- These lower-level differences are irrelevant as long as a client and server share the same communications protocols and support the same applications
Client/Server Applications

- Actual functions performed by the application can be split up between client and server
- Optimize platform and network resources
- Optimize the ability of users to perform various tasks
- Optimize the ability to cooperate with one another using shared resources
Database Applications

- The server is a database server
- Interaction between client and server is in the form of transactions
  - the client makes a database request and receives a database response
- Server is responsible for maintaining the database

Figure 13.3 Client/Server Architecture for Database Applications
(a) Desirable client/server use

Figure 13.4 Client/Server Database Usage

(b) Misused client/server

Figure 13.4 Client/Server Database Usage
Classes of Client/Server Applications

- **Host-based processing**
  - not true client/server computing
  - traditional mainframe environment

![Host-based processing diagram]

- **Server-based processing**
  - server does all the processing
  - user workstation provides a graphical user interface

![Server-based processing diagram]
Classes of Client/Server Applications

- Client-based processing
  - all application processing done at the client
  - data validation routines and other database logic function are done at the server

Classes of Client/Server Applications

- Cooperative processing
  - application processing is performed in an optimized fashion
  - complex to set up and maintain
Three-Tier Client/Server Architecture

- Application software distributed among three types of machines
  - User machine
    - thin client
  - Middle-tier server
    - Gateway
    - Convert protocols
    - Merge/integrate results from different data sources
  - Backend server
File Cache Consistency

- File caches hold recently accessed file records
- Caches are consistent when they contain exact copies for remote data
- File-locking prevents simultaneous access to a file

Figure 13.7 Distributed File Caching in Sprite
Middleware

- Set of tools that provide a uniform means and style of access to system resources across all platforms
- Enable programmers to build applications that look and feel the same
- Enable programmers to use the same method to access data

Figure 13.8 The Role of Middleware in Client/Server Architecture
Figure 13.9 Logical View of Middleware [BERN96]

Figure 13.10 Example of Middleware Functionality

Running on the Novell network are applications, middleware, Novell Network, and IBM OS/2

Running on the TCP/IP LAN are applications, middleware, TCP/IP, and UNIX

Running on the DECnet and PCs are applications and middleware

VAX running VMS

Terminal server

Bridge
Distributed Message Passing

- Message passed used to communicate among processes
- Send and receive messages as used in a single system OR
- Remote procedure calls

Basic Message-Passing Primitives

![Diagram of message passing with sending and receiving processes](image)

*Figure 13.12 Basic Message-Passing Primitives*
Reliability Versus Unreliability

- Reliable message-passing guarantees delivery if possible
  - Not necessary to let the sending process know that the message was delivered
- Send the message out into the communication network without reporting success or failure
  - Reduces complexity and overhead

Blocking Versus Nonblocking

- Nonblocking
  - Process is not suspended as a result of issuing a Send or Receive
  - Efficient and flexible
  - Difficult to debug
- Blocking
  - Send does not return control to the sending process until the message has been transmitted
  - OR does not return control until an acknowledgment is received
  - Receive does not return until a message has been placed in the allocated buffer
Remote Procedure Calls

- Allow programs on different machines to interact using simple procedure call/return semantics
- Widely accepted
- Standardized
  - Client and server modules can be moved among computers and operating systems easily

Figure 13.13 Remote Procedure Call Mechanism
Client/Server Binding

- Binding specifies the relationship between remote procedure and calling program
- Nonpersistent binding
  - logical connection established during remote procedure call
- Persistent binding
  - connection is sustained after the procedure returns

Synchronous versus Asynchronous

- Synchronous RPC
  - Behaves must like a subroutine call
- Asynchronous RPC
  - Does not block the caller
  - Enable a client to invoke a server repeatedly so that it has a number of requests in the pipeline at one time
Object-Oriented Mechanisms

Clusters

- Alternative to symmetric multiprocessing (SMP)
- Group of interconnected, whole computers working together as a unified computing resource
  - illusion is one machine
  - system can run on its own
Benefits of Clusters

- Absolute scalability
  - Can have dozens of machines each of which is a multiprocessor
- Incremental scalability
  - Add new systems in small increments
- High availability
  - Failure of one node does not mean loss of service
- Superior price/performance
  - Cluster can be equal or greater computing power than a single large machine at a much lower cost

Clusters

- Separate server
  - Each computer is a separate server
  - No shared disks
  - Need management or scheduling software
  - Data must be constantly copied among systems so each is current
Clusters

- Shared nothing
  - Reduces communication overhead
  - Several servers connected to common disks
  - Disks partitioned into volumes
  - Each volume owned by a computer
  - If computer fails another computer gets ownership of the volume

Clusters

- Shared disk
  - Multiple computers share the same disks at the same time
  - Each computer has access to all of the volumes on all of the disks
Operating System Design

Issues

- Failure management
  - Highly available cluster offers a high probability that all resources will be in service
    - No guarantee about the state of partially executed transactions if failure occurs
  - Fault-tolerant cluster ensures that all resources are always available

- Load balancing
  - When new computer added to the cluster, the load-balancing facility should automatically include this computer in scheduling applications

- Parallelizing Computation
  - Parallelizing compiler
  - Parallelized application
  - Parametric computing
Cluster Computer Architecture

- Cluster middleware services and functions
  - Single entry point
  - Single file hierarchy
  - Single control point
  - Single virtual networking
  - Single memory space
  - Single job-management system

Cluster Computer Architecture

- Cluster middleware services and functions
  - Single user interface
  - Single I/O space
  - Single process space
  - Checkpointing
  - Process migration
Clusters Compared to SMP

- SMP is easier to manage and configure
- SMP takes up less space and draws less power
- Clusters are better for incremental and absolute scalability
- Clusters are superior in terms of availability

Windows 2000 Cluster Service

- Cluster Service
  - Collection of software on each node that manages all cluster-specific activity
- Resource
  - Item managed by the cluster service
- Online
  - Online at node when it is providing service on that specific node
- Group
  - Collection of resources managed as a single unit
Sun Cluster

- Major components
  - Object and communication support
  - Process management
  - Networking
  - Global distributed file system
Figure 13.17  Sun Cluster Structure

Figure 13.18  Sun Cluster File System Extensions
Beowulf and Linux Clusters

Key features

- Mass market commodity components
- Dedicated processors (rather than scavenging cycles from idle workstations)
- A dedicated, private network (LAN or WAN or internetted combination)
- No custom components
- Easy replication from multiple vendors

Beowulf and Linux Clusters

Key features

- Scalable I/O
- A freely available software base
- Using freely available distribution computing tools with minimal changes
- Returning the design and improvements to the community
Figure 13.19 Generic Beowulf Configuration