COP 6611 Advanced Operating System

Introduction

Chi Zhang
czhang@cs.fiu.edu

Outline

- Goals
- Hardware Concepts
- Software Concepts
- The Client-Server Model
Definition of a Distributed System (1)

- A distributed system is:
  A collection of independent computers that appears to its users as a single coherent system.

- Goals
  - Utilizing distributed resources
  - Transparency
  - Open / Extensible
  - Scalable

Definition of a Distributed System (2)

A distributed system organized as middleware. Note that the middleware layer extends over multiple machines.
Transparency in a Distributed System

<table>
<thead>
<tr>
<th>Transparency</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>Hide differences in data representation and how a resource is accessed</td>
</tr>
<tr>
<td>Location</td>
<td>Hide where a resource is located</td>
</tr>
<tr>
<td>Migration</td>
<td>Hide that a resource may move to another location</td>
</tr>
<tr>
<td>Relocation</td>
<td>Hide that a resource may be moved to another location while in use</td>
</tr>
<tr>
<td>Replication</td>
<td>Hide that a resource may be shared by several competitive users</td>
</tr>
<tr>
<td>Concurrency</td>
<td>Hide that a resource may be shared by several competitive users</td>
</tr>
<tr>
<td>Failure</td>
<td>Hide the failure and recovery of a resource</td>
</tr>
<tr>
<td>Persistence</td>
<td>Hide whether a (software) resource is in memory or on disk</td>
</tr>
</tbody>
</table>

Different forms of transparency in a distributed system.

Scalability Problems

- More users / resources
- Geographically scalable
- Administratively scalable

<table>
<thead>
<tr>
<th>Concept</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centralized services</td>
<td>A single server for all users</td>
</tr>
<tr>
<td>Centralized data</td>
<td>A single on-line telephone book</td>
</tr>
<tr>
<td>Centralized algorithms</td>
<td>Doing routing based on complete information</td>
</tr>
</tbody>
</table>

Examples of scalability limitations.
Scaling Techniques (1)

- Hiding communication latencies
  - Asynchronous communication
  - Moving server tasks to clients
- (Server) Distribution
- Replication
  - Increase availability
  - Load balancing
  - Access nearby copy

Scaling Techniques (2)

The difference between letting:

a) a server or

b) a client check forms as they are being filled
Scaling Techniques (3)

An example of dividing the DNS name space into zones.

Hardware Concepts

Homogeneous vs. Heterogeneous
Multiprocessors (1)

A bus-based multiprocessor.

Multiprocessors (2)

a) A crossbar switch
b) An omega switching network
Homogeneous Multicomputer Systems

(a) Grid
(b) Hypercube

System Area Networks (SAN)

Heterogeneous Multicomputer Systems

- Most distributed systems
- Lack a global system view
  - Operating system does not know how many computers in the system
  - Applications are aware of distributedness (without middleware)
  - Applications cannot assume the same performance or services everywhere
Software Concepts

<table>
<thead>
<tr>
<th>System</th>
<th>Description</th>
<th>Main Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOS</td>
<td>Tightly-coupled operating system for multi-processors and homogeneous multicomputers</td>
<td>Hide and manage hardware resources</td>
</tr>
<tr>
<td>NOS</td>
<td>Loosely-coupled operating system for heterogeneous multicomputers (LAN and WAN)</td>
<td>Offer local services to remote clients</td>
</tr>
<tr>
<td>Middleware</td>
<td>Additional layer atop of NOS implementing general-purpose services</td>
<td>Provide distribution transparency</td>
</tr>
</tbody>
</table>

An overview of
- DOS (Distributed Operating Systems)
- NOS (Network Operating Systems)
- Middleware

Uniprocessor Operating Systems

Separating applications from operating system code through a microkernel.
Multiprocessor Operating Systems (1)

- High Performance with Multiple CPUs
  - One thread per CPU
  - Communication through shared memory location
- Protect data against simultaneous accesses
  - Semaphore
  - Monitor

Multiprocessor Operating Systems (2)

```cpp
monitor Counter {
  private:
  int count = 0;
  public:
  int value() { return count; }
  void incr () { count = count + 1; }
  void decr() { count = count - 1; }
}
```

A monitor to protect an integer against concurrent access.
Multicomputer Operating Systems (1)

General structure of a multicomputer operating system

Multicomputer Operating Systems (2)

Alternatives for blocking and buffering in message passing.
Multicomputer Operating Systems (3)

<table>
<thead>
<tr>
<th>Synchronization point</th>
<th>Send buffer</th>
<th>Reliable comm. guaranteed?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block sender until buffer not full</td>
<td>Yes</td>
<td>Not necessary</td>
</tr>
<tr>
<td>Block sender until message sent</td>
<td>No</td>
<td>Not necessary</td>
</tr>
<tr>
<td>Block sender until message received</td>
<td>No</td>
<td>Necessary</td>
</tr>
<tr>
<td>Block sender until message delivered</td>
<td>No</td>
<td>Necessary</td>
</tr>
</tbody>
</table>

Relation between blocking, buffering, and reliable communications.

Distributed Shared Memory Systems (1)

a) Pages of address space distributed among four machines

b) Situation after CPU 1 references page 10

c) Situation if page 10 is read only and replication is used
Distributed Shared Memory Systems (2)

False sharing of a page between two independent processes.

Network Operating System (1)

General structure of a network operating system.
Network Operating System (2)

Two clients and a server in a network operating system.

Introduction to Middleware

- NOS + Middleware
  - NOS: Scalable / Open
    - DOS: Transparent
  - Middleware models
    - Distributed file systems / distributed database
    - Remote procedures calls (RPC)
    - Distributed Objects
  - Middleware services
    - Naming, persistence, transactions …
Positioning Middleware

General structure of a distributed system as middleware.

Middleware and Openness

In an open middleware-based distributed system, the protocols used by each middleware layer should be the same, as well as the interfaces they offer to applications.
Comparison between Systems

<table>
<thead>
<tr>
<th>Item</th>
<th>Distributed OS</th>
<th>Network OS</th>
<th>Middleware-based OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degree of transparency</td>
<td>Very High</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Same OS on all nodes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Number of copies of OS</td>
<td>1</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Basis for communication</td>
<td>Shared memory</td>
<td>Messages</td>
<td>Files</td>
</tr>
<tr>
<td>Resource management</td>
<td>Global, central</td>
<td>Global, distributed</td>
<td>Per node</td>
</tr>
<tr>
<td>Scalability</td>
<td>No</td>
<td>Moderately</td>
<td>Yes</td>
</tr>
<tr>
<td>Openness</td>
<td>Closed</td>
<td>Closed</td>
<td>Open</td>
</tr>
</tbody>
</table>

A comparison between multiprocessor operating systems, multicomputer operating systems, network operating systems, and middleware based distributed systems.

Clients and Servers

General interaction between a client and a server.

Underlying protocols: Reliable or Unreliable?
Processing Level

The general organization of an Internet search engine into three different layers

Multitiered Architectures (1)

Alternative client-server organizations (a) – (e).
Multitiered Architectures (2)

An example of a server acting as a client.

Modern Architectures

An example of horizontal distribution of a Web service.