CS4513
Distributed Computer Systems

Introduction

Outline
• Overview
• Goals
• Software
• Client Server

The Rise of Distributed Systems
• Computer hardware prices falling, power increasing
  - If cars the same, Rolls Royce would cost 1 dollar and get 1 billion miles per gallon (with 200 page manual to open the door)
• Network connectivity increasing
  - Everyone is connected with fat pipes
• It is easy to connect hardware together
• Definition: a distributed system is
  - A collection of independent computers that appears to its users as a single coherent system.

Definition of a Distributed System
A distributed system organized as middleware.
Note that the middleware layer extends over multiple machines.
Users can interact with the system in a consistent way, regardless of where the interaction takes place.
Note: Middleware may be "part" of application in practice.

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>

Scalability Problems
• As distributed systems grow, centralized solutions are limited
  - Consider LAN name resolution vs. WAN

<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>

• Sometimes, hard to avoid (consider a bank)
• Need to collect information in distributed fashion and distributed in a distributed fashion
• Challenges:
  - geography, ownership domains, time synchronization
Scaling Techniques: Hiding Communication Latency

• Especially important for interactive applications
• If possible, do asynchronous communication
  • Not always possible when client has nothing to do
• Instead, can hide latencies

Scaling Techniques: Distribution

Example: DNS name space into zones
(nl.vu.cs.fluit - z1 gives address of vu gives address of cs)

Scaling Techniques: Replication

• Copy of information to increase availability and decrease centralized load
  • Example: P2P networks (Gnutella +)
    • Distribute copies uniformly or in proportion to use
  • Example: CDNs (akamai)
  • Example: Caching is a replication decision made by client
• Issue: Consistency of replicated information
  • Example: Web Browser cache

Outline

• Overview (done)
• Goals (done)
• Software
  • Client Server

Software Concepts

<table>
<thead>
<tr>
<th>System</th>
<th>Description</th>
<th>Main Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOS</td>
<td>Tight-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>

• DOS (Distributed Operating Systems)
• NOS (Network Operating Systems)
• Middleware

Uniprocessor Operating Systems

• Separating applications from operating system code through a microkernel
  • Can extend to multiple computers
Distributed Operating Systems

- But no longer have shared memory
  - Provide message passing
  - Can try to provide distributed shared memory
- But tough to get acceptable performance

Network Operating System

- OSes can be different (Windows or Linux)
- Typical services: rlogin, rcp
  - Fairly primitive way to share files

Positioning Middleware

- Network OS not transparent. Distributed OS not independent of computers.
  - Middleware can help

Outline

- Overview (done)
- Goals (done)
- Software (done)
- Client Server ←
Clients and Servers

- Thus far, have not talked about organization of processes
  - Again, many choices but most agree upon is client-server
- If can do so without connection, quite simple
- If underlying connection is unreliable, not trivial
  - Resend. What if receive twice?
- Use TCP for reliable connection (most Inet apps)
  - Not always appropriate for high-speed LAN connections or interactive applications

Client-Server Implementation Levels

- Example of an Internet search engine
  - UI on client
  - Processing can be on client or server
  - Data level is server, keeps consistency

Multitiered Architectures

- Thin client (a) to Fat client (e)
  - (d) and (e) popular for NOS environments

Multitiered Architectures: 3 tiers

- Server may act as a client
  - Example would be transaction monitor across multiple databases

Modern Architectures: Horizontal

- Rather than vertical, distribute servers across nodes
  - Example of Web server “farm” for load balancing
  - Clients, too (peer-to-peer systems)