Thread Usage in Nondistributed Systems

- Figure 3-1. Context switching as the result of IPC.

Thread Implementation

- Figure 3-2. Combining kernel-level lightweight processes and user-level threads.

Multithreaded Servers (1)

- Figure 3-3. A multithreaded server organized in a dispatcher/worker model.
Multithreaded Servers (2)

- Figure 3-4. Three ways to construct a server.

<table>
<thead>
<tr>
<th>Model</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threads</td>
<td>Parallelism, blocking system calls</td>
</tr>
<tr>
<td>Single-threaded process</td>
<td>No parallelism, blocking system calls</td>
</tr>
<tr>
<td>Finite-state machine</td>
<td>Parallelism, nonblocking system calls</td>
</tr>
</tbody>
</table>

The Role of Virtualization in Distributed Systems

- Figure 3-5. (a) General organization between a program, interface, and system. (b) General organization of virtualizing system A on top of system B.

Virtualization

- Virtualization is to extend or replace an existing interface to mimic the behavior of another system.
- IBM 370 mainframe, VMM (Virtual Machine Monitor) - 1970s
  - Support multi users by one VM per user
  - Support different operation systems
- After 1990s,
  - To provide legacy interface on new hardware platforms
  - To provide uniformity over a heterogeneous collection of servers connected by networks
  - To provide a high degree of portability and flexibility

Interfaces of Computer Systems at Different Levels

- An interface between the hardware and software consisting of machine instructions
  - that can be invoked by any program.
- An interface between the hardware and software, consisting of machine instructions
  - that can be invoked only by privileged programs, such as an operating system.
- An interface consisting of system calls as offered by an operating system.
- An interface consisting of library calls
  - generally forming what is known as an application programming interface (API).
  - In many cases, the aforementioned system calls are hidden by an API.
Architectures of Virtual Machines (a)

A process virtual machine, with multiple instances of (application, runtime) combinations.
- JVM
- Cywin

Architectures of Virtual Machines (b)

(b) A virtual machine monitor, with multiple instances of (applications, operating system) combinations.
Example: VMware

Networked User Interfaces (1)

- Figure 3-8. (a) A networked application with its own protocol.
Networked User Interfaces (2)

(b) A general solution to allow access to remote applications.

Example: The XWindow System

The basic organization of the X Window System.

Client-side transparencies

- Distribute transparency
- Access transparency
  - Use client-side stub from an interface definition of the server
- Location transparency
- Migration transparency
- Relocation transparency
- Replication transparency
- Failure transparency
- Concurrency and persistence transparency handled by servers.

Client-Side Software for Distribution Transparency

Transparent replication of a server using a client-side solution.
Servers

- Binding issues
  - When, how to bind
- Stateless servers
  - Soft state
- Stateful servers
  - Temporary session state vs. permanent state
  - Where to keep state information
    - Cookies - client side

General Design Issues (1)

- Figure 3-11. (a) Client-to-server binding using a daemon.

General Design Issues (2)

- Figure 3-11. (b) Client-to-server binding using a superserver.

Server Clusters (1)
**Server Clusters (2)**

- Figure 3-13. The principle of TCP handoff.

**Distributed Servers**

- Figure 3-14. Route optimization in a distributed server.

**Managing Server Clusters**

- Figure 3-15. The basic organization of a PlanetLab node.

**PlanetLab (1)**

- PlanetLab management issues:
  - Nodes belong to different organizations.
    - Each organization should be allowed to specify who is allowed to run applications on their nodes,
    - And restrict resource usage appropriately.
  - Monitoring tools available assume a very specific combination of hardware and software.
    - All tailored to be used within a single organization.
  - Programs from different slices but running on the same node should not interfere with each other.
Figure 3-16. The management relationships between various PlanetLab entities.

- A node owner puts its node under the regime of a management authority, possibly restricting usage where appropriate.
- A management authority provides the necessary software to add a node to PlanetLab.
- A service provider registers itself with a management authority, trusting it to provide well-behaving nodes.

Figure 3-17. The principle of dynamically configuring a client to communicate to a server. The client first fetches the necessary software, and then invokes the server.

- Relationships between PlanetLab entities:
  - A service provider contacts a slice authority to create a slice on a collection of nodes.
  - The slice authority needs to authenticate the service provider.
  - A node owner provides a slice creation service for a slice authority to create slices. It essentially delegates resource management to the slice authority.
  - A management authority delegates the creation of slices to a slice authority.
**Models for Code Migration**

- Weak mobility
  - Mobility mechanism
  - Sender-initiated mobility
  - Receiver-initiated mobility

- Strong mobility
  - Migration process
  - Clone process

**Migration in Heterogeneous Systems**

Three ways to handle migration (which can be combined)
- Pushing memory pages to the new machine and resending the ones that are later modified during the migration process.
- Stopping the current virtual machine; migrate memory, and start the new virtual machine.
- Letting the new virtual machine pull in new pages as needed, that is, let processes start on the new virtual machine immediately and copy memory pages on demand.

**Migration and Local Resources**

<table>
<thead>
<tr>
<th>Process-to-resource binding</th>
<th>Unattached</th>
<th>Fastened</th>
<th>Fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>By identifier</td>
<td>MV (or GR)</td>
<td>GR (or MV)</td>
<td>GR</td>
</tr>
<tr>
<td>By value</td>
<td>CP (or MV,GR)</td>
<td>GR (or CP)</td>
<td>GR</td>
</tr>
<tr>
<td>By type</td>
<td>RB (or MV,CP)</td>
<td>RB (or GR,CP)</td>
<td>GR</td>
</tr>
</tbody>
</table>

- GR Establish a global systemwide reference
- MV Move the resource
- CP Copy the value of the resource
- RB Rebind process to locally-available resource

Figure 3-19. Actions to be taken with respect to the references to local resources when migrating code to another machine.