Processes – III

- Threads
- Virtualization
- Further Discussions on Clients and Servers
- Code Migration – passing programs instead of data
- Agents
Code Migration

- Transferring programs, even while they are being executed.
- Traditionally, code migrations in DS was process migration – an entire process was moved from one m/c to another for load balancing.
- Currently, in DSs, computing power is a less critical issue than communication costs.
Why Code Migration?

- To process data close to where those data reside – migrate part of the server program to the client
- Improve performance by parallelism – send each copy of a program off to different sites – linear speed up
  - Mobile agents
- Flexibility
  - Traditional approach – partition an application into locations where each part should be executed – multi-tiered architecture
  - Code migration – dynamically configure distributed systems
Dynamic Configuration Through Code Migration

- The server provides client implementation when the client binds to the server
  - The client first fetches necessary software and then calls the server
  - Needs standard for downloading and initializing protocol
  - The client should be able to execute the downloaded code
Dynamic Configuration Through Code Migration

- **Advantages**
  - Clients need not have all the software preinstalled to talk to servers
  - The software can be added (discarded) as (not) necessary
  - Given standard interfaces, the client/server protocol can be changed

- **Disadvantage**
  - Security – how to check the trustworthiness of the downloaded code!
Models for Code Migration

Segments of a Process

- **Code** – the set of instructions that make up the executing program
- **Resource** – references to external resources (files, printers, devices, etc.) needed by the process
- **Execution** – stores the current execution state (private data, the stack, and the program counter) of a process
Alternative Models for Code Migration

Based on transfer of segments of a process

- Code segment only along with some initialization data – weak mobility
  - Transferred program always starts from its initial state – simplicity, Java applets
  - Only target m/c can execute the code – requires portability

- Both code and execution segment – strong mobility
  - A running process can be stopped and subsequently transferred to a different m/c, and resume execution where it left off
  - Powerful than weak mobility, but harder to implement

- Based on initiative for code migration – both for strong and weak mobility
  - Sender initiated – sending a search program on Internet to a Web database server
  - Receiver initiated – Java applet

- Based on the Execution of the Migrated Code – weak mobility
  - Target process
  - Separate process

- Based on process migration – strong mobility
  - Migrated process
  - Cloned process in diff. m/c
Alternatives for Code Migration - Summary

Mobility mechanism

Weak mobility

- Sender-initiated mobility
  - Execute at target process
  - Execute in separate process

- Receiver-initiated mobility
  - Execute at target process
  - Execute in separate process

Strong mobility

- Sender-initiated mobility
  - Migrate process
  - Clone process

- Receiver-initiated mobility
  - Migrate process
  - Clone process
Resource Migration

Segments of a Process

- **Code** – the set of instructions that make up the executing program
- **Resource** – references to external resources (files, printers, devices, etc.) needed by the process
- **Execution** – stores the current execution state (private data, stack, and program counter) of a process

Thus far, we have discussed only the migration of code and data segment

- Code segment only along with some initialization data – weak
- Both code and execution segment – strong

Let us now discuss the issues w.r.t. resource migration
Migration of Local Resources

- Types of process-to-resource binding – in order of strength
  - By identifier – requires a resource uniquely identified by the identifier – referring to URL or FTP server address
  - By value – Not a specific resource, but the value provided by the resource is important – referring to standard libraries in C or Java
  - By type – requires a resource of specific type – references to local devices such as monitors, printers

- In code migration, we need to change the process-to resource binding – we have to bind the resource-to-target m/c
Migration of Local Resources – contd.

- Types of resource-to-m/c binding – in order of increased cost
  - Unattached – the data files related only with the migrated program
  - Fastened – local databases, complete website
  - Fixed – cannot be moved, local devices or communication endpoints
Actions needed w.r.t. the references to local resources to migrate code to another m/c

<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>RB (or GR)</td>
</tr>
</tbody>
</table>

- **GR** - establish a global system wide reference
- **MV** - move the resource
- **CP** - copy the value of the resource
- **RB** - rebind process to locally available resource
Migration in Heterogeneous Systems

- The migrated system should be executed in different target platform - execution segment contains data that is private to the process, its current stack, and the program counter
- One possible solution – avoid having execution that depend on platform specific data such as register values
- Best solution to handle heterogeneity is to use virtual machines
  - Process virtual machines – JVM
  - Virtual machines monitor – allow the migration of processes along with their underlying operating system
Summary

- Code Migration
- What is it?
  - Reason for code migration
  - Dynamic configuration through code migration
  - Alternatives for code migration
- Code Migration Issues
  - Migration of local resources
  - Heterogeneity of systems