Code and Process Migration

- Motivation
- How does migration occur?
- Resource migration
- Agent-based system
- Details of process migration

Motivation

- Key reasons: performance and flexibility
- Process migration (aka strong mobility)
  - Improved system-wide performance – better utilization of system-wide resources
  - Examples: Condor, DQS
- Code migration (aka weak mobility)
  - Shipment of server code to client – filling forms (reduce communication, no need to pre-link stubs with client)
  - Ship parts of client application to server instead of data from server to client (e.g., databases)
  - Improve parallelism – agent-based web searches
Motivation

• Flexibility
  – Dynamic configuration of distributed system
  – Clients don’t need preinstalled software – download on demand

Migration models

• Process = code seg + resource seg + execution seg
• Weak versus strong mobility
  – Weak => transferred program starts from initial state
• Sender-initiated versus receiver-initiated
• Sender-initiated (code is with sender)
  – Client sending a query to database server
  – Client should be pre-registered
• Receiver-initiated
  – Java applets
  – Receiver can be anonymous
Who executes migrated entity?

- Code migration:
  - Execute in a separate process
  - [Applets] Execute in target process
- Process migration
  - Remote cloning
  - Migrate the process

Models for Code Migration

- Mobility mechanism:
  - Strong mobility
  - Weak mobility
  - Sender-initiated mobility
  - Receiver-initiated mobility
  - Execute at target process
  - Execute in separate process
  - Migrate process
  - Clone process
Do Resources Migrate?

- Depends on resource to process binding
  - By identifier: specific web site, ftp server
  - By value: Java libraries
  - By type: printers, local devices
- Depends on type of “attachments”
  - Unattached to any node: data files
  - Fastened resources (can be moved only at high cost)
    - Database, web sites
  - Fixed resources
    - Local devices, communication end points

Resource Migration Actions

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

- Actions to be taken with respect to the references to local resources when migrating code to another machine.
- GR: establish global system-wide reference
- MV: move the resources
- CP: copy the resource
- RB: rebind process to locally available resource
Migration in Heterogeneous Systems

- Systems can be heterogeneous (different architecture, OS)
  - Support only weak mobility: recompile code, no runtime information
  - Strong mobility: recompile code segment, transfer execution segment
    [migration stack]
  - Virtual machines - interpret source (scripts) or intermediate code [Java]

Agents

- Software agents
  - Autonomous process capable of reacting to, and initiating changes in its environment, possibly in collaboration
  - More than a “process” – can act on its own
- Mobile agent
  - Capability to move between machines
  - Needs support for strong mobility
  - Example: D’Agents (aka Agent TCL)
    - Support for heterogeneous systems, uses interpreted languages
Software Agents in Distributed Systems

<table>
<thead>
<tr>
<th>Property</th>
<th>Common to all agents?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autonomous</td>
<td>Yes</td>
<td>Can act on its own</td>
</tr>
<tr>
<td>Reactive</td>
<td>Yes</td>
<td>Responds timely to changes in its environment</td>
</tr>
<tr>
<td>Proactive</td>
<td>Yes</td>
<td>Initiates actions that affects its environment</td>
</tr>
<tr>
<td>Communicative</td>
<td>Yes</td>
<td>Can exchange information with users and other agents</td>
</tr>
<tr>
<td>Continuous</td>
<td>No</td>
<td>Has a relatively long lifespan</td>
</tr>
<tr>
<td>Mobile</td>
<td>No</td>
<td>Can migrate from one site to another</td>
</tr>
<tr>
<td>Adaptive</td>
<td>No</td>
<td>Capable of learning</td>
</tr>
</tbody>
</table>

- Some important properties by which different types of agents can be distinguished.

Agent Technology

- T

[Diagram of agent technology]

Intra-platform communication

Inter-platform communication

Agent program

Agent's endpoint

Management component

Directory service

ACC

Agent platform
Agent Communication Languages

(1)

<table>
<thead>
<tr>
<th>Message purpose</th>
<th>Description</th>
<th>Message Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>INFORM</td>
<td>Message sends an intention of the sender of a message, along with the description of the actual message content.</td>
<td>Proposition</td>
</tr>
<tr>
<td>QUERY-IF</td>
<td>Query whether a given proposition is true</td>
<td>Expression</td>
</tr>
<tr>
<td>QUERY-REF</td>
<td>Query for a given object</td>
<td>Proposal specifics</td>
</tr>
<tr>
<td>CFP</td>
<td>Ask for a proposal</td>
<td>Proposal</td>
</tr>
<tr>
<td>PROPOSE</td>
<td>Provide a proposal</td>
<td>Proposal</td>
</tr>
<tr>
<td>ACCEPT-PROPOSAL</td>
<td>Tell that a given proposal is accepted</td>
<td>Proposal ID</td>
</tr>
<tr>
<td>REJECT-PROPOSAL</td>
<td>Tell that a given proposal is rejected</td>
<td>Proposal ID</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Request that an action be performed</td>
<td>Action specification</td>
</tr>
<tr>
<td>SUBSCRIBE</td>
<td>Subscribe to an information source</td>
<td>Reference to source</td>
</tr>
</tbody>
</table>

Agent Communication Languages

(2)

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>INFORM</td>
</tr>
<tr>
<td>Sender</td>
<td>max@<a href="http://fanclub-beatrix.royalty-spotters.nl:7239">http://fanclub-beatrix.royalty-spotters.nl:7239</a></td>
</tr>
<tr>
<td>Receiver</td>
<td>elke@iiop://royalty-watcher.uk:5623</td>
</tr>
<tr>
<td>Language</td>
<td>Prolog</td>
</tr>
<tr>
<td>Ontology</td>
<td>genealogy</td>
</tr>
<tr>
<td>Content</td>
<td>female(beatrix),parent(beatrix,juliana,bernhard)</td>
</tr>
</tbody>
</table>

- A simple example of a FIPA ACL message sent between two agents using Prolog to express genealogy information.