

Code, Process, and VM Migration

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



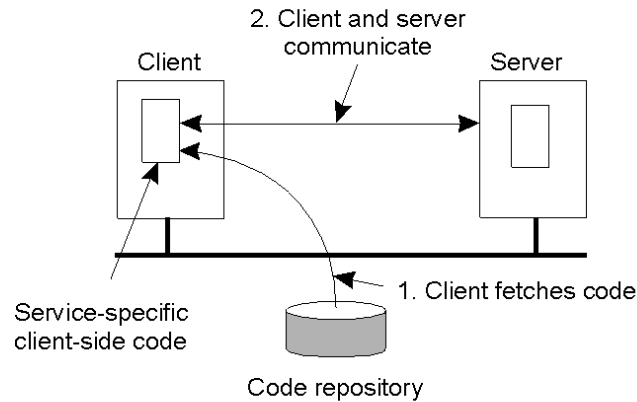
Part 1: Migration Introduction

- 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
 - migration initiated by machine where code resides
 - Client sending a query to database server
 - Client should be pre-registered
- Receiver-initiated
 - Migration initiated by machine that receives code
 - Java applets
 - Receiver can be anonymous

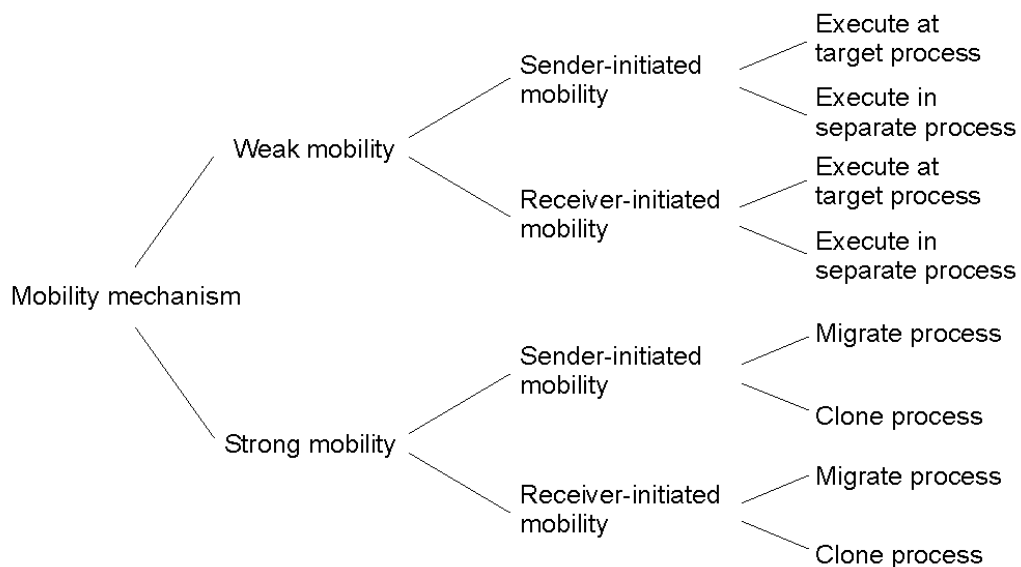


Who executes migrated entity?

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



Models for Code Migration



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

Resource-to machine binding

		Unattached	Fastened	Fixed
Process-to-resource binding	By identifier	MV (or GR)	GR (or MV)	GR
	By value	CP (or MV, GR)	GR (or CP)	GR
	By type	RB (or GR, CP)	RB (or GR, CP)	RB (or GR)

- 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 run time information
 - Strong mobility: recompile code segment, transfer execution segment [migration stack]
 - Virtual machines - interpret source (scripts) or intermediate code [Java]

