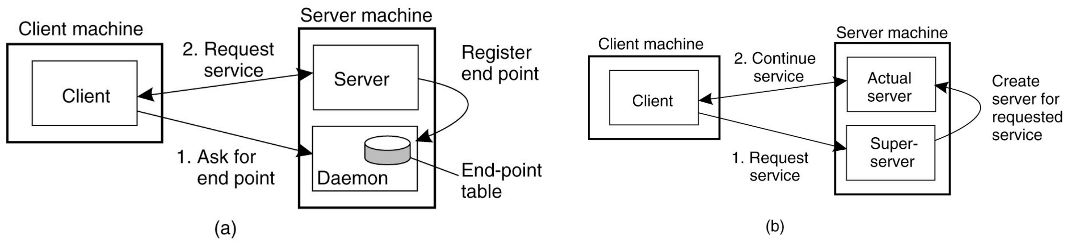
Types of Virtualization

- Emulation
  - VM emulates/simulates complete hardware
  - Unmodified guest OS for a different PC can be run
    - Bochs, VirtualPC for Mac, QEMU
- Full/native Virtualization
  - VM simulates “enough” hardware to allow an unmodified guest OS to be run in isolation
    - Same hardware CPU
  - IBM VM family, VMWare Workstation, Parallels,…

Types of virtualization

- Para-virtualization
  - VM does not simulate hardware
  - Use special API that a modified guest OS must use
  - Hypercalls trapped by the Hypervisor and serviced
  - Xen, VMWare ESX Server
- OS-level virtualization
  - OS allows multiple secure virtual servers to be run
  - Guest OS is the same as the host OS, but appears isolated
    - apps see an isolated OS
  - Solaris Containers, BSD Jails, Linux Vserver
- Application level virtualization
  - Application is gives its own copy of components that are not shared
    - (E.g., own registry files, global objects) - VE prevents conflicts
  - JVM

Server Design Issues

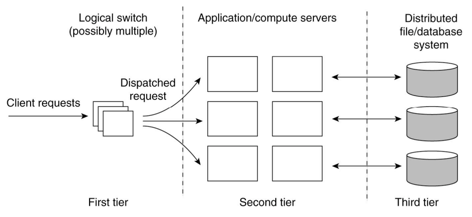


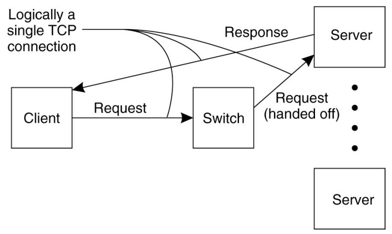
- Server Design
  - Iterative versus concurrent
- How to locate an end-point (port #)?
  - Well known port #
  - Directory service (port mapper in Unix)
  - Super server (inetd in Unix)

Stateful or Stateless?

- Stateful server
  - Maintain state of connected clients
  - Sessions in web servers
- Stateless server
  - No state for clients
- Soft state
  - Maintain state for a limited time; discarding state does not impact correctness

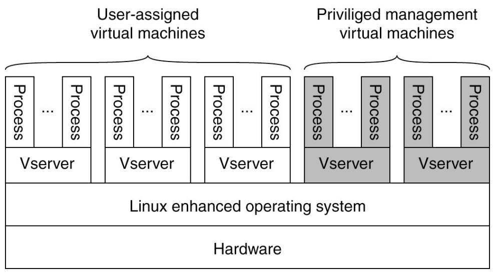
Server Clusters


Logical switch (possibly multiple)
Application/compute servers
Distributed file/database system
Dispatched request
Client requests
First tier
Second tier
Third tier



- Web applications use tiered architecture
  - Each tier may be optionally replicated; uses a dispatcher
  - Use TCP splicing or handoffs

Case Study: PlanetLab


User-assigned virtual machines
Priviliged management virtual machines
Process
...
Process
Vserver
Linux enhanced operating system
Hardware

- Distributed cluster across universities
  - Used for experimental research by students and faculty in networking and distributed systems
- Uses a virtualized architecture
  - Linux Vservers
  - Node manager per machine
  - Obtain a “slice” for an experiment: slice creation service

Code and Process Migration

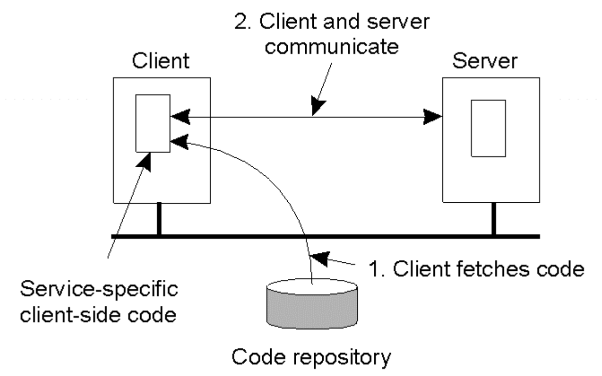
- Motivation
- How does migration occur?
- Resource migration
- Agent-based system
- Details of process migration

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


2. Client and server communicate
Client
Server
1. Client fetches code
Service-specific client-side code
Code repository

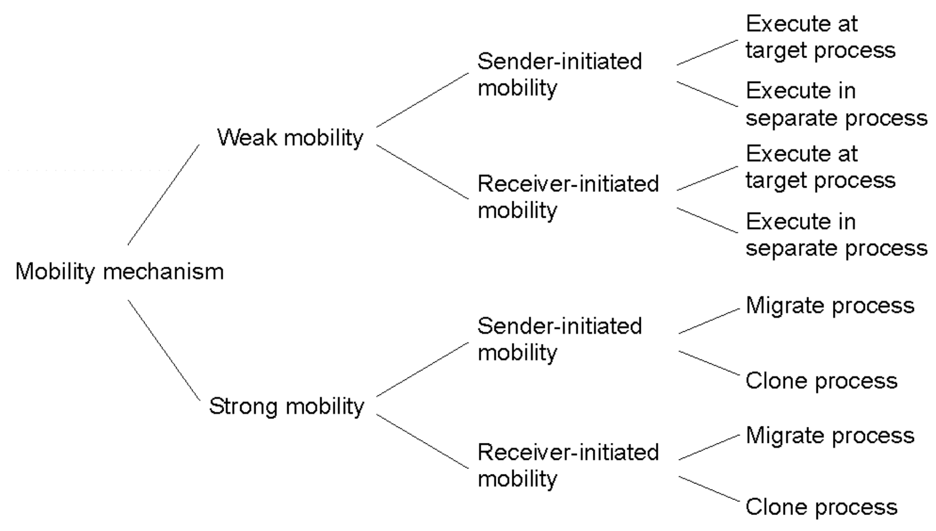
Migration models

- 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
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

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

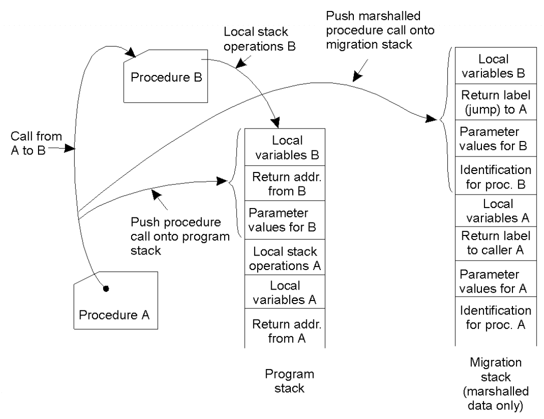
Resource-to machine binding

Process-to-resource binding		Unattached	Fastened	Fixed
	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]



Case study: 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

Case Study: ISOS

- Internet scale operating system
  - Harness compute cycles of thousands of PCs on the Internet
  - PCs owned by different individuals
  - Donate CPU cycles/storage when not in use (pool resouces)
  - Contact coordinator for work
  - Coodinator: partition large parallel app into small tasks
  - Assign compute/storage tasks to PCs
- Examples: Seti@home, P2P backups

Case study: Condor

- Condor: use idle cycles on workstations in a LAN
- Used to run lareg batch jobs, long simulations
- Idle machines contact condor for work
- Condor assigns a waiting job
- User returns to workstation => suspend job, migrate
- Flexible job scheduling policies