第3讲 进程与虚拟化

§3.1 进程与线程

§3.2 虚拟化

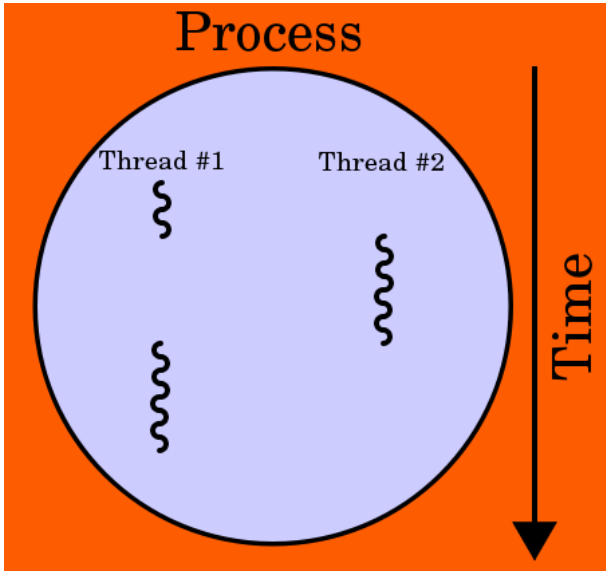
§3.3 客户-服务器

§3.4 计算迁移

§3.1进程与线程

- 何为进程?
- 何为线程?
- 操作系统如何管理进程?

初始化
分配资源
调度
销毁

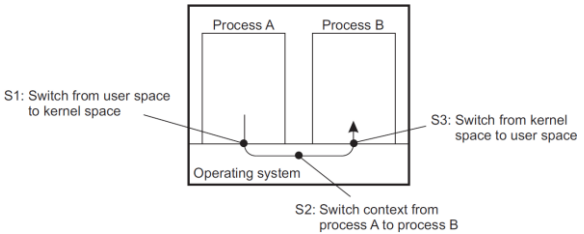


进程与线程

- 在物理处理器上用软件创建虚拟处理器：
- 线程：
  - 最小的可执行一系列指令的软件处理器；
  - 终止线程当前执行，保存线程上下文，用于以后执行；
- 进程：
  - 包含多个线程的软件处理器，线程需要在进程的上下文中执行。
- 上下文
  - 处理器上下文：栈指针、地址寄存器、程序计数器等；
  - 进程上下文：线程上下文、MMU寄存器值、TLB等；
  - 线程上下文：处理器上下文、状态等。

多线程

- 避免不必要的阻塞：
  - 单线程的进程在进行I/O操作的时候会被阻塞；
  - 在多线程的进程中，操作系统可以将CPU切换到进程的另外一个线程；
- 更好地发挥并行性：
  - 具有多线程的进程可以在多核或多处理器CPU上并行执行；
- 避免进程上下文切换：
  - 进程切换需要在内核空间进行。



分布式系统多线程

- 多线程客户端
  - 客户端派生多个线程，每一个线程负责一个调用；
  - 如果调用由不同的服务器，将会得到线性加速；
- 例子：Web浏览器
  - 从服务器下载HTML页面时，利用多线程分别负责内容获取：如text、image，及内容展示；
  - 随着文件的到达，浏览器将这些文件展示出来；

多线程并行度

Thread-level parallelism: TLP

Let c_i denote the fraction of time that exactly i threads are being executed simultaneously.

$$TLP = \frac{\sum_{i=1}^{N} i \cdot c_i}{1 - c_0}$$

with N the maximum number of threads that (can) execute at the same time.

Practical measurements

A typical Web browser has a TLP value between 1.5 and 2.5 $\Rightarrow$ threads are primarily used for logically organizing browsers.

分布式系统多线程

- 多线程服务器
  - 提高服务能力
    - 增加一个线程的代价低于一个进程
    - 单线程服务很难在多处理器系统中实现 scale-up（纵向扩展）
    - 与多线程客户端相呼应，通过并行响应请求隐藏网络延迟
  - 改善程序结构
    - 使用简单容易理解的阻塞调用可以简化整体结构
    - 多线程的程序的控制流简单、代码数量较少，容易理解

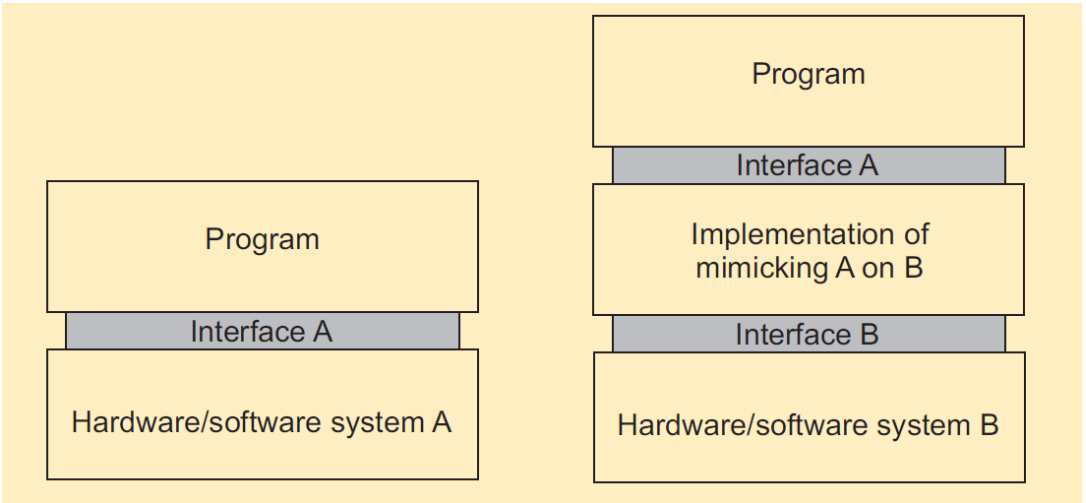
模型	特征
多线程	并行，使用会导致阻塞的系统调用
单线程进程	非并行，使用会导致阻塞的系统调用
有限状态机	并行，使用非阻塞系统调用

图 3.4　构建服务器的 3 种方式

单线程：对阻塞任务进行状态保存，待OS告知完成后再取回处理

§3.2 虚拟化

- 资源虚拟化是一个宽泛的概念
  - 多进程、多线程实现了CPU的多路复用，可以看作是一种虚拟化
  - 编程接口都可以看作是虚拟化的体现



虚拟化的主要作用

- 应用程序适配：硬件比软件变化快
- 资源需求多样化：不同应用需要不同的环境
- 需要灵活的可移植性和代码迁移
- 失效和攻击隔离

虚拟化技术

- 计算机系统的四种接口
- 软件-硬件接口：指令集架构
  - 特权指令：允许操作系统执行的执行；
  - 通用指令：可以被任何程序执行的指令；
- 操作系统接口：系统调用
- 应用程序接口：库函数调用

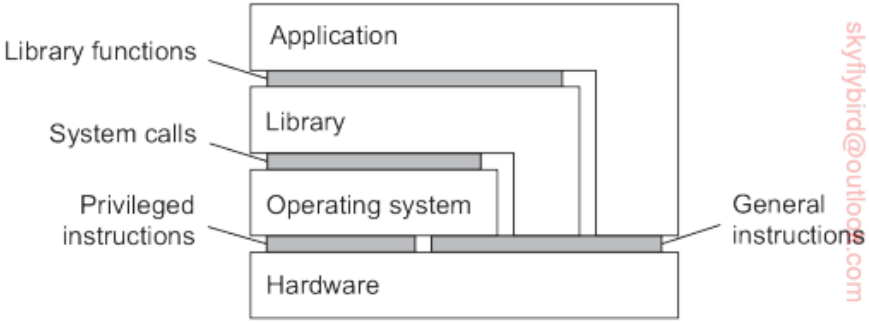


Figure 3.7: Various interfaces offered by computer systems.

虚拟化技术

(a) 进程虚拟机 (Process VM)，如JVM

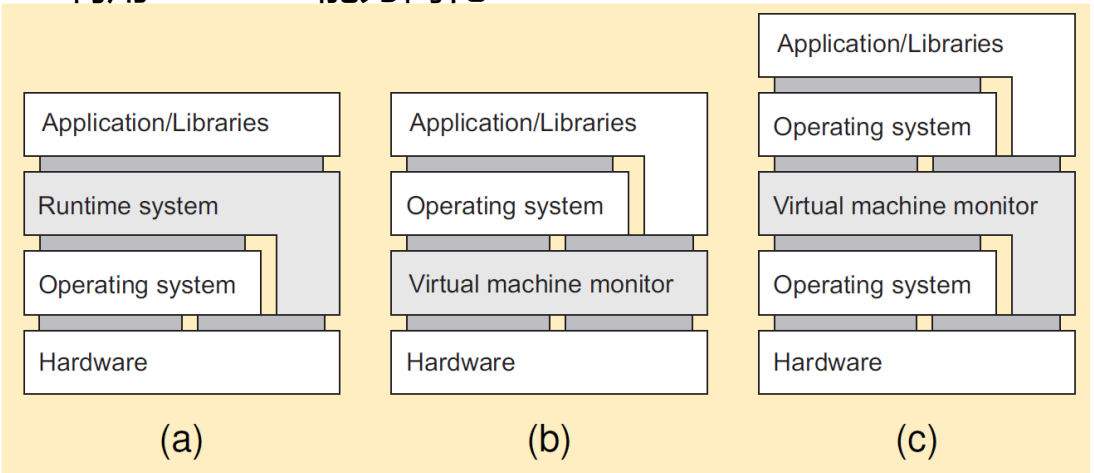
- 分离的指令集合，实际是运行在OS之上的解释器

(b) 原生虚拟机监控器（Native VMM)

- 底层指令集，允许多样化的Guest OS

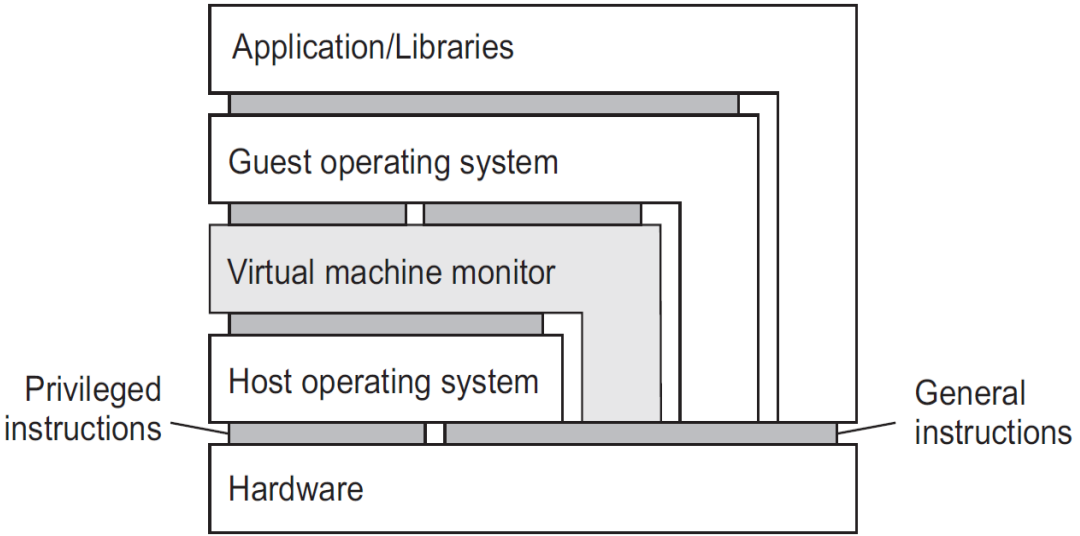
(c) 主机虚拟机监控器 (Hosted VMM)

- 利用Host OS能力简化VMM



虚拟机指令执行与性能

- 现在的VM性能优异，比较接近物理机水平
  - 大部分指令（VMM、OS、App）直接被硬件执行，无需解释、模拟



虚拟机指令执行与性能

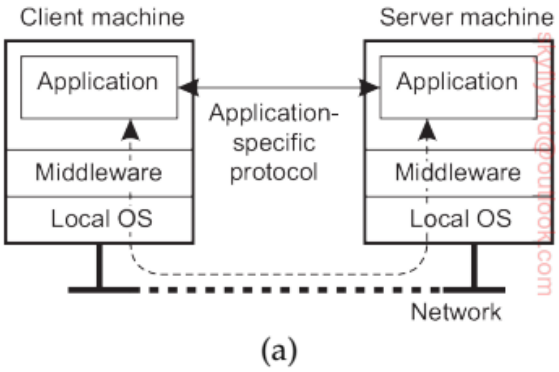
- 特权指令：用户模式下执行时会系统陷入的指令
  - 非特权指令：其他所有指令
- 特殊/敏感指令：
  - 控制敏感性指令：可能影响到机器配置的指令（如：寄存器重定位或者中断表）
  - 行为敏感性指令：指令效果受上下文影响，如：可能改变寄存器值也可能不
- “可虚拟化”
  - 所有的敏感指令都属于特权指令
  - 不是所有系统能满足：如X86，解决：
    - 指令模拟、指令包装、修改GuestOS等

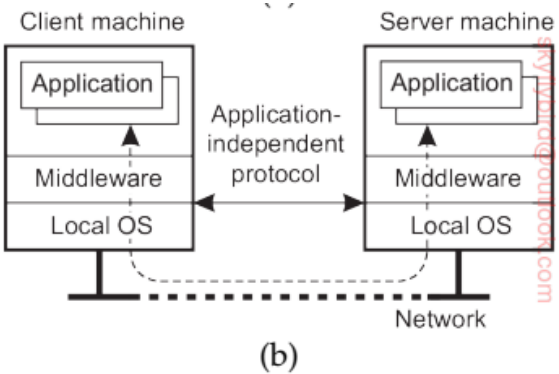
分布式系统虚拟化

- 云计算是虚拟化在分布式系统中的应用
  - **Infrastructure-as-a-Service** (**IaaS**) covering the basic infrastructure
  - **Platform-as-a-Service** (**PaaS**) covering system-level services
  - **Software-as-a-Service** (**SaaS**) containing actual applications

§3.3 客户与服务器

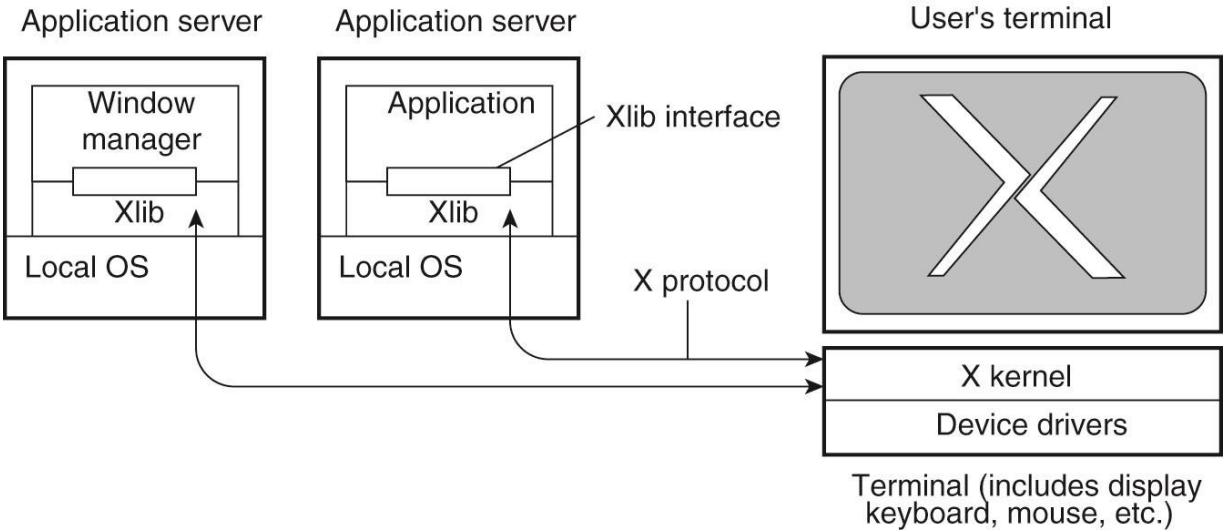
- 客户端（Client）
  - to provide the means for users to interact with remote servers
- 网络化用户接口
  - App specific vs. App independent





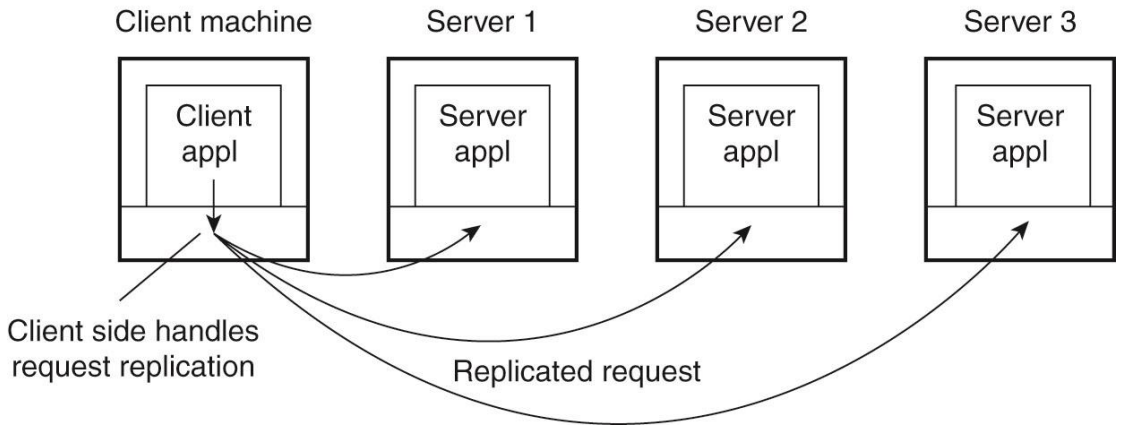
The X Window

- one of the oldest and still widely used networked user interfaces



客户端实现透明性

- 访问透明性：客户端拥有用于 RPC访问的存根
- 位置/迁移透明性：客户端记录服务器的实际位置
- 副本透明性：客户端存根多个副本的调用
- 故障透明性：经常放在客户端（用于屏蔽服务器和通信问题）



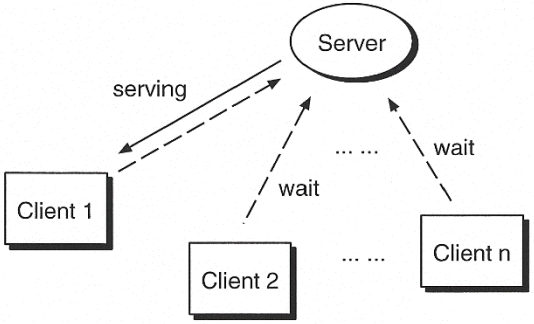
服务器

- 服务器是实现特定服务功能的进程
- 基本模式：接收-处理-回复
- General design issues
  - Server behavior: iterative vs. concurrent
  - State of client: stateful vs. stateless
  - Server discovery: where is it?

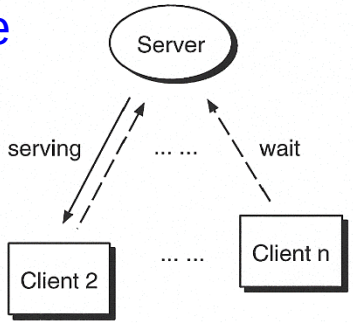
迭代服务器与并发服务器

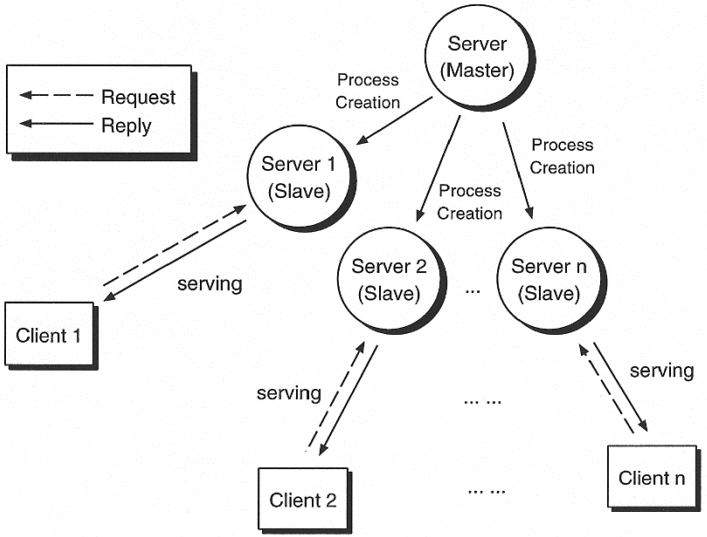
- Iterative server迭代服务器:
  - One single server process provides the service for client.
  - Requests from multiple clients handled one by one sequentially.
  - Advantages:
    - Easy to program; no concurrent access problem; good if processing time for a request is very short.
  - Disadvantages:
    - May block other requests and cause a long response time.
- Concurrent server并发服务器:
  - Server process creates a child process/thread to handle each request.
  - It then returns to wait for a new request.
  - Multiple requests are handled concurrently.
  - Advantages:
    - faster response time without blocking
  - Disadvantages:
    - Harder to program: concurrent access to shared data/resource.

迭代服务器与并发服务器



Iterative server





Concurrent server

有状态与无状态

- Stateful server: load is on server.
  - Server saves information about status of clients.
  - Server can remember what clients have done before.
  - This reduces the size of request messages
    - Since both client and server share their context.
  - Lost messages could put client and server out of synchrony（可靠性是问题）
- Stateless server: load is on client. (RESTful)
  - Server maintains no information about status of clients.
  - All requests start from a "*fresh*" state and are self-contained.
  - Request messages need to contain sufficient information for the context.
  - Straightforward handling procedures upon failures.

Server Discovery

- How does a client find a server?
- Need to know the *end-point* or *port* and the host address
  - Statically assigned like well known servers like HTTP on port 80
  - Look-up service provided by a special directory server
  - Using a superserver that selects on multiple ports and forks off the appropriate server when a request comes

Server Discovery

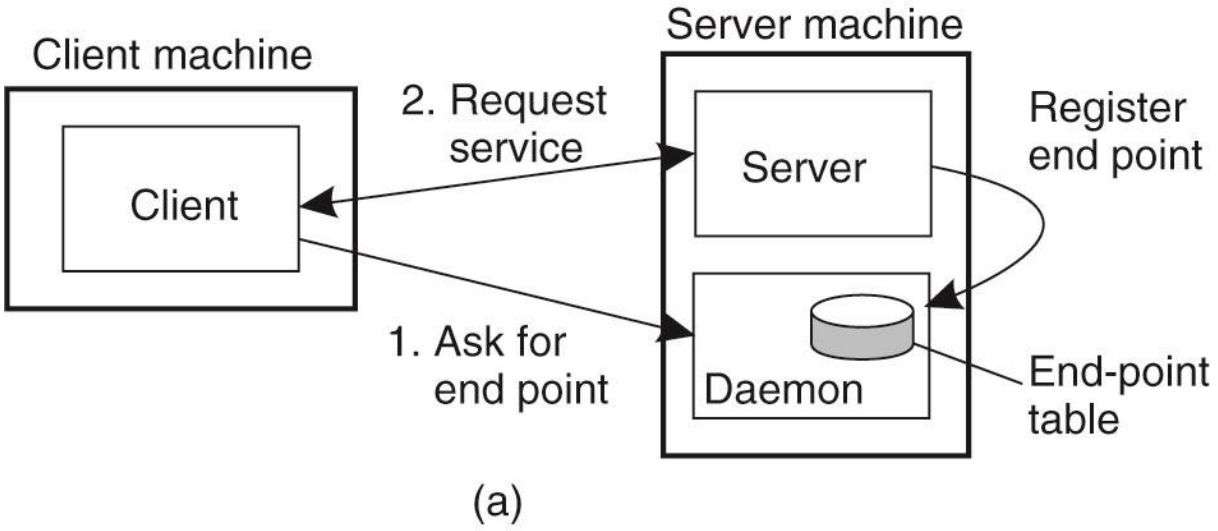
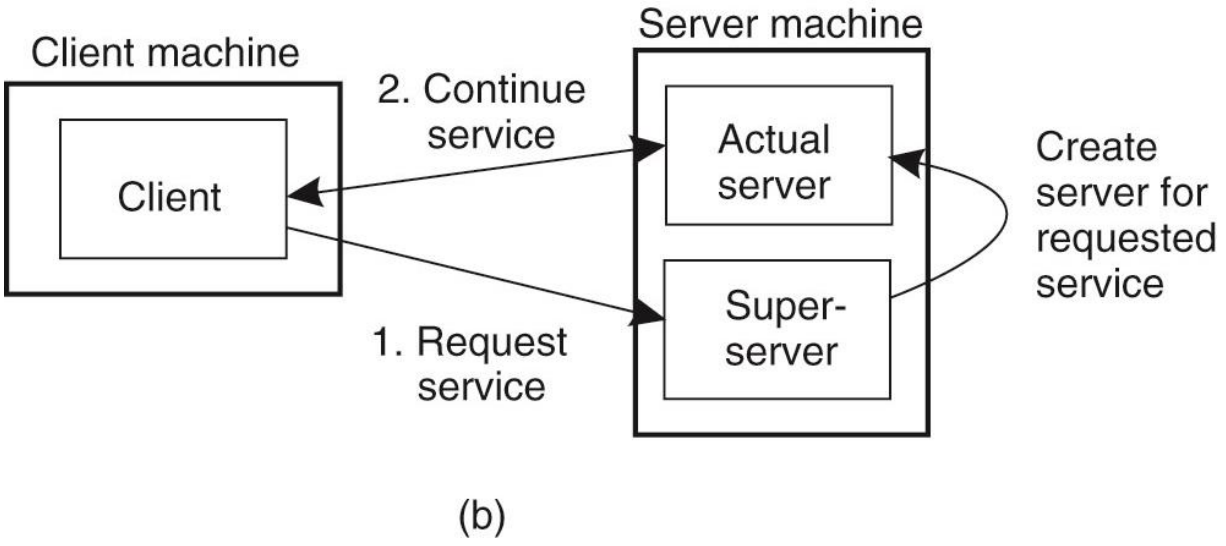


Figure 3-11. (a) Client-to-server binding using a daemon.

Server Discovery

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



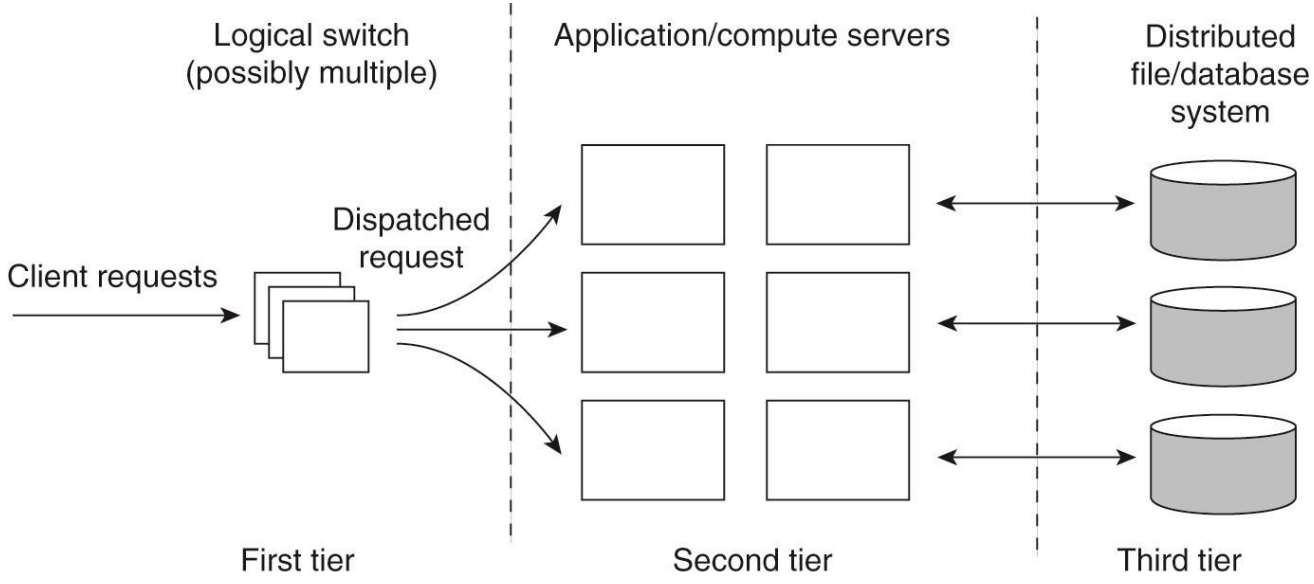
(b)

服务器中断

- 在服务器已经接收了服务器请求后，如何中断服务器的运行?
  - 如：中止某个文件的处理
- 简单方法：用户强行关闭客户端程序
- 复杂方法：带外数据（out-of-band data）机制
  - 单独的线程/进程+单独连接用于紧急信息处理
  - 相同连接但是单独的数据包优先级，如TCP urgent data+OS signal

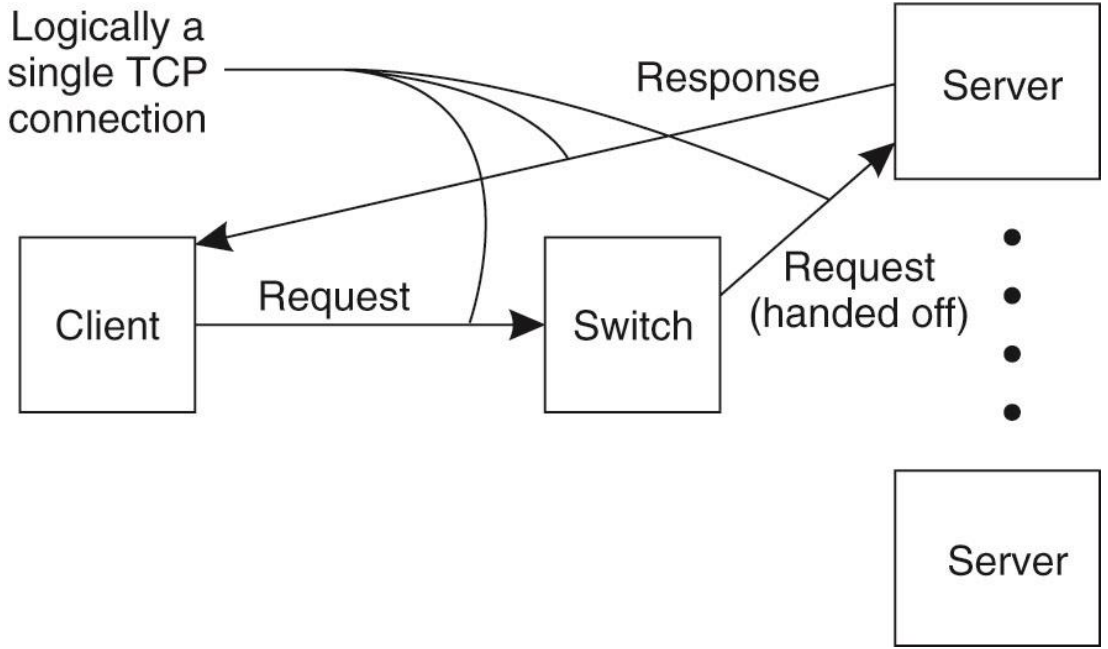
Server Clusters

Figure 3-12. The general organization of a three-tiered server cluster.



Server Clusters

Figure 3-13. The principle of TCP handoff (requires IP forwarding and IP rewriting).

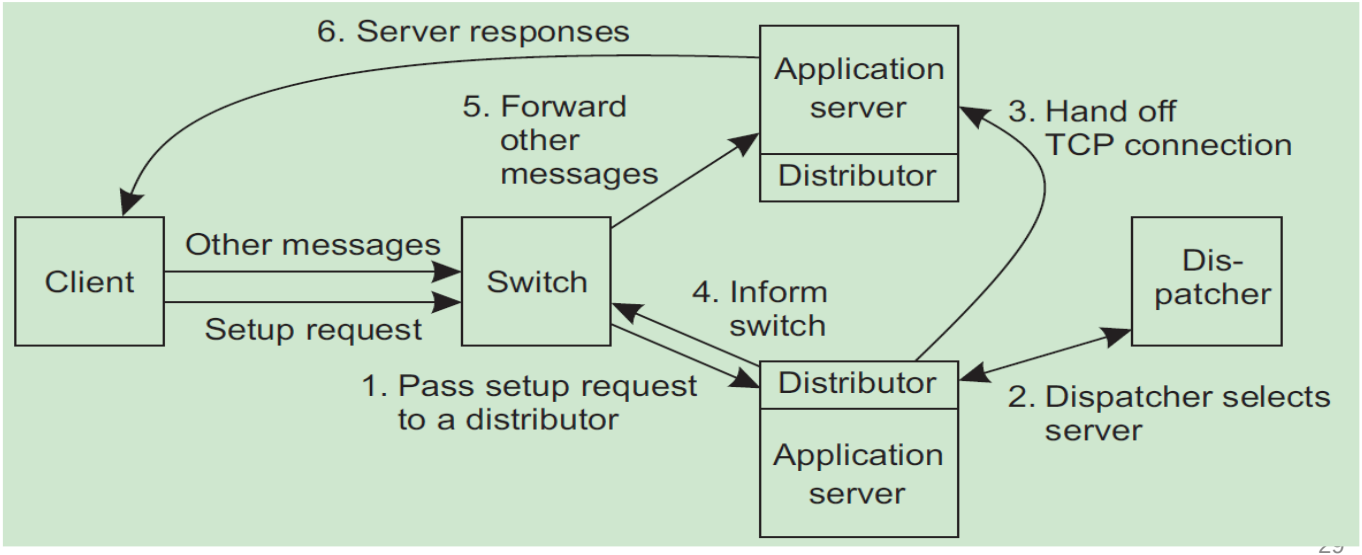


Server Clusters

- 传输层分发策略
  - Round-robin: need to track connection-based requests
  - Based on service type (e.g., port)
  - Based on server loads
- 基于请求内容的分发策略
  - E.g., http request content at Web server
- 混合策略：传输层+请求内容

传输层+请求内容混合分发

- Switch passes new request to arbitrary distributor;
- The distributor passes the request to a dispatcher;
- The dispatcher inspects the payload to decide on the best application server.



Wide-area Server Clusters

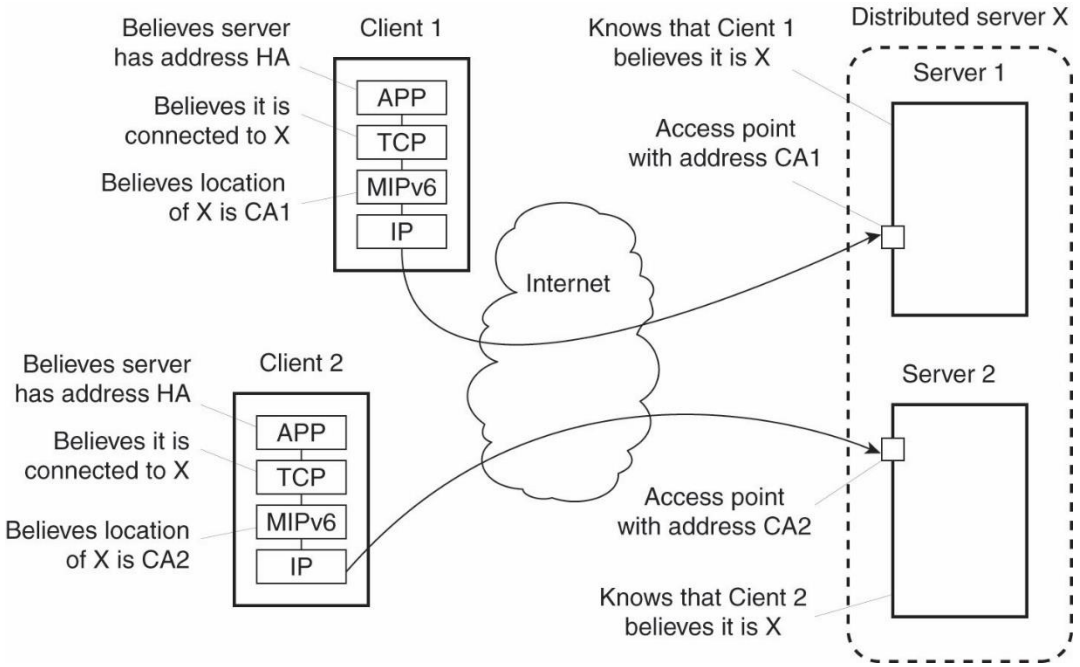
- Internet 范围的服务器集群
  - 如地理分布的云数据中心
- 可能由不同组织/机构拥有和管理
- Request dispatching
  - 基于DNS：DNS服务器登记同域名的多个主机地址
    - 通常基于locality来分发
    - 如果request经由proxy则locality可能丧失

Wide-area Server Clusters

- A *distributed server* is a special kind of WA clusters
  - a possibly dynamically changing set of machines,
  - with also possibly varying access points,
  - but which nevertheless appears as a single, powerful machine
- Key Point: *stable access point* for a distributed sever
  - can be implemented using mobility support for IPv6 (MIPv6)
  - *home network*: where a mobile node normally resides
  - *home address (HoA):* stable address for a node in home network
  - *care of address (CoA)*: the address to forward packet to
  - *route optimization*

Distributed Servers

Route optimization in a distributed server via MIP.



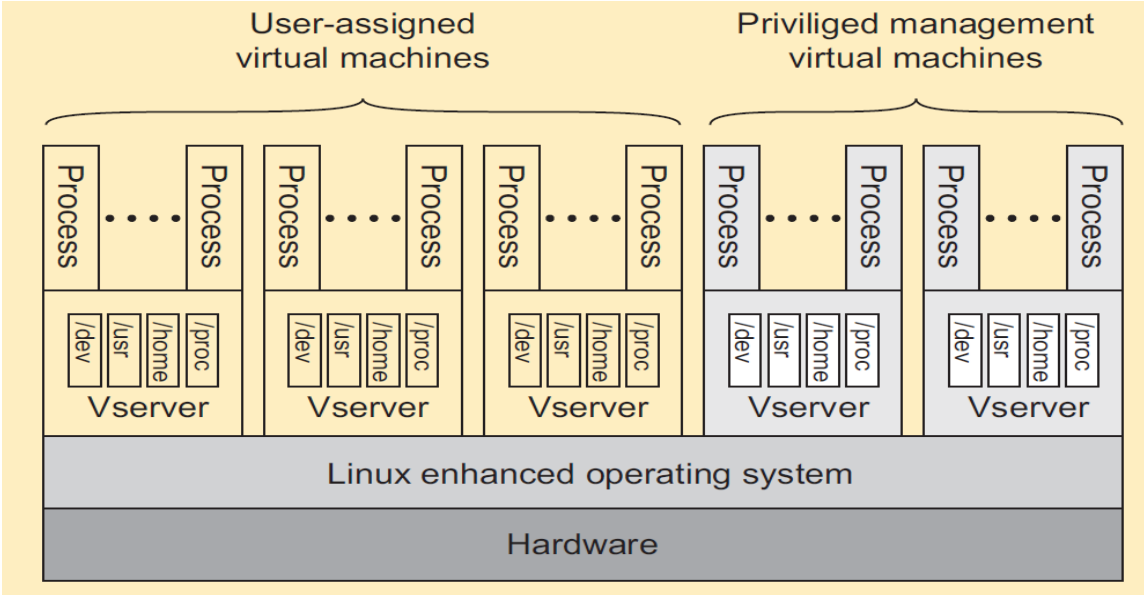
Managing Server Clusters

- Individual
  - An administrator manually logs in to each server to manage them.
  - Or they use shell scripts or simple tools to manage a collection of servers
- Integrated
  - An management interface is provided at one system to collectively manage the cluster of servers
- Still problem for large clusters

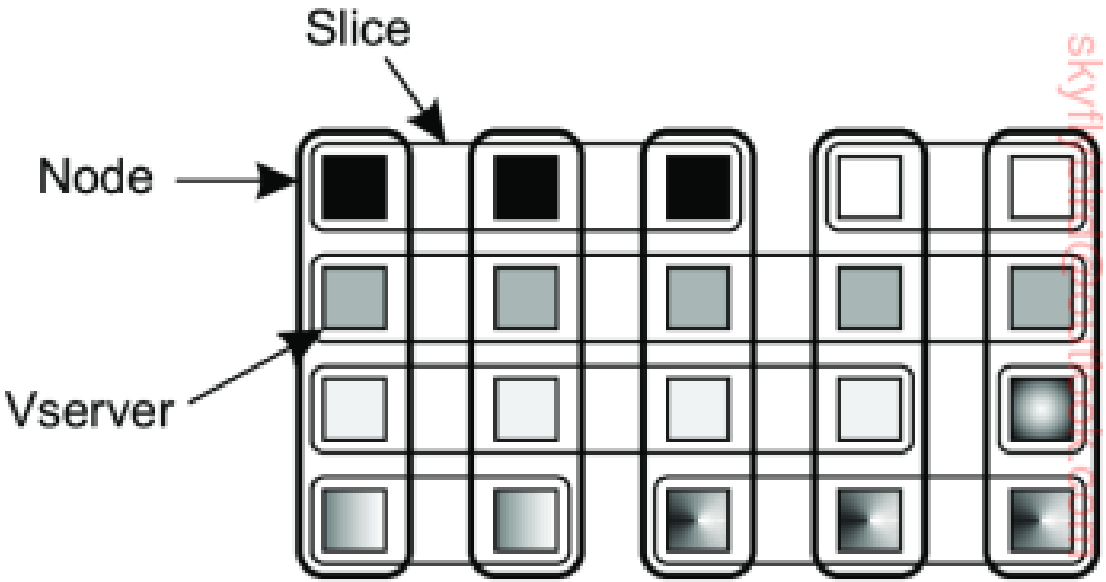
Large server cluster farms are increasing rapidly!

PlanetLab: Example Server Clusters

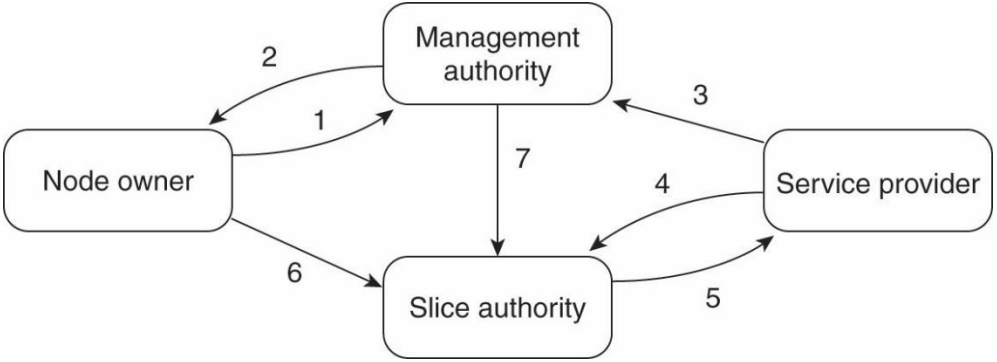
- 不同的组织贡献机器，共享用于各种实验
- Linux enhanced with VMM
- Vservers for users



PlanetLab Slicing



PlanetLab Management



(1) 结点拥有者把它的结点加入一个管理授权者的管理域，如果合适的话可能会限制使用方式。

(2) 管理授权者提供必需的软件用于增加一个结点给 PlanetLab。

(3) 服务提供者向管理授权者注册，信任它会提供良好行为的结点。

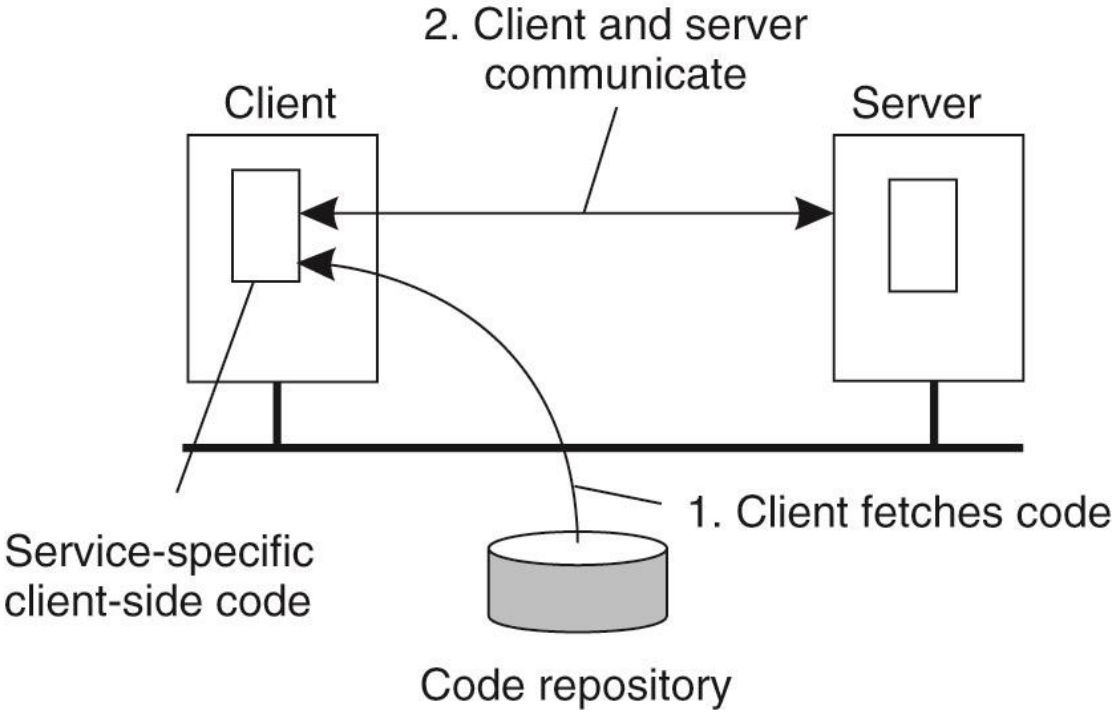
(4) 服务提供者联系切片授权者在一组结点上创建一个切片。

(5)切片授权者需要认证服务提供者。

(6) 结点拥有者为切片授权者提供切片创建服务用以创建切片。它实质上让切片授权者可以使用它的资源。

(7) 管理授权者把切片创建的任务委托给切片授权者。

§3.4 Code Migration



Purpose of Code Migration

- Load balancing
  - From high load node to low load node
- Communication reducing
  - Move code to data side
- Dynamic config.
  - On-demand protocol/lib installation
- Mobile agents
  - Parallel processing
  - Data side computing

Code Migration Models

- Three segments:
  - code segment: 指令集合
  - resource segment：外部资源指针
  - execution segment：执行状态量（数据、栈等）
- *Weak mobility* vs. *Strong mobility*
  - Only the code segment transferred, e.g. JSP
    - Some initialization data, predefined starting point
  - Both code and execution segments transferred.
- *Sender-initiated* vs. *receiver-initiated*
  - Uploading code to a server
  - Downloading code from a server

Code Migration Models

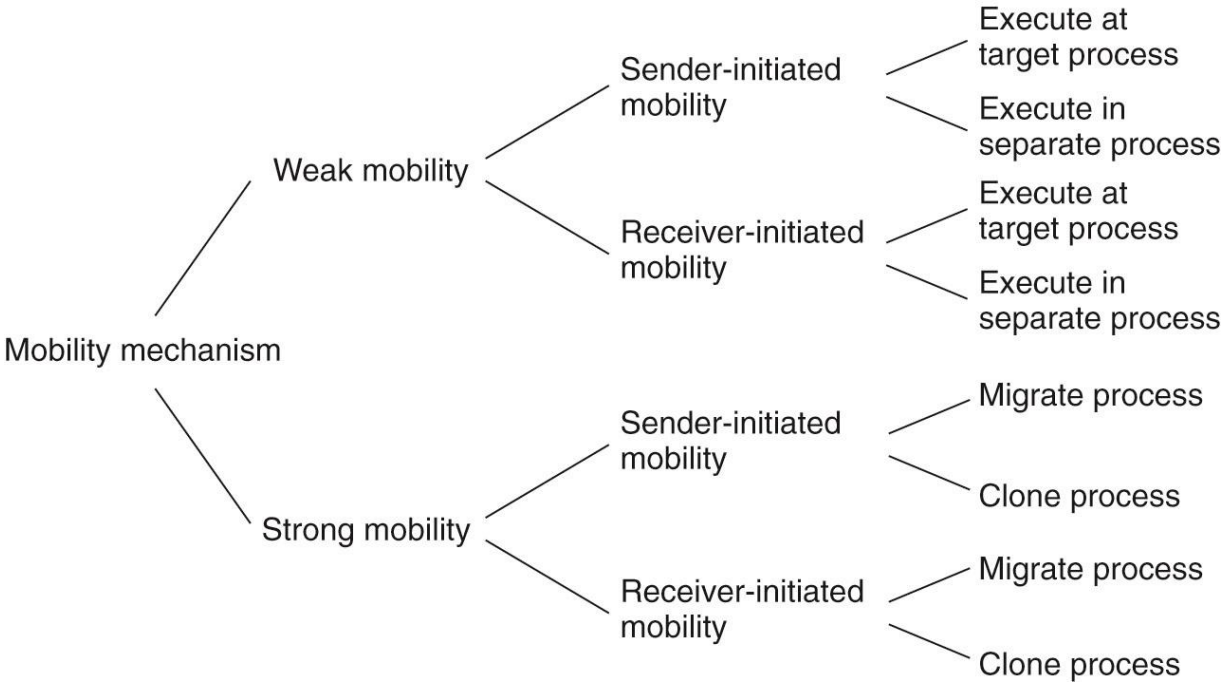
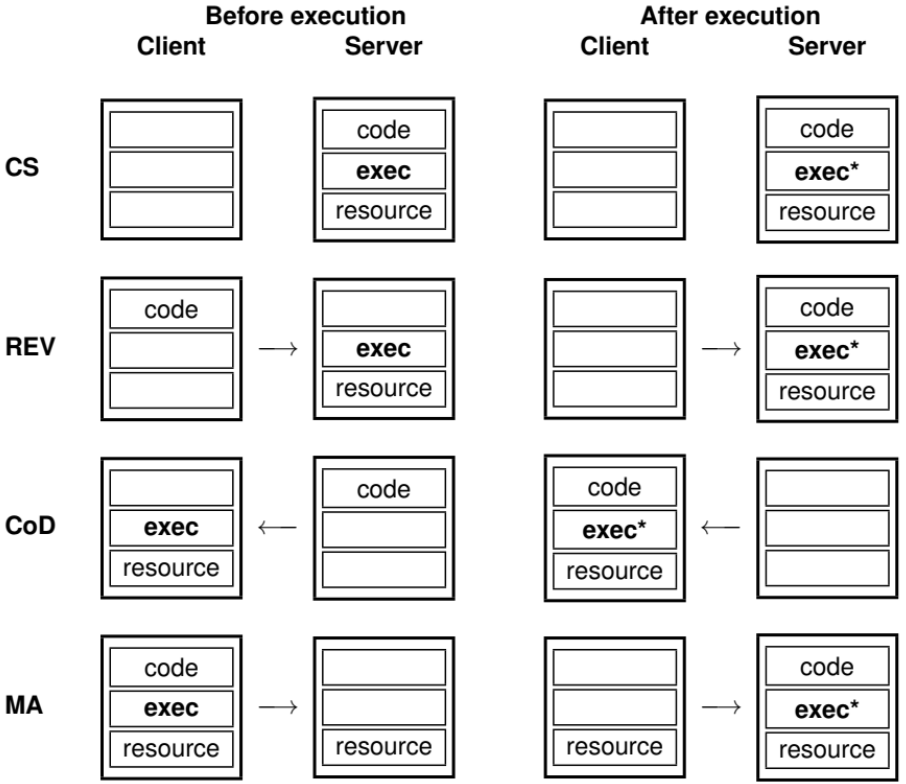


Figure 3-18. Alternatives for code migration.

Code Migration Paradigms

- 四种样式

CS: Client-Server
CoD: Code-on-demand
REV: Remote evaluation
MA: Mobile agents



Migration and Local Resources

Process-to-resource binding	Resource-to-machine 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 MV,CP)	RB (or GR,CP)	RB (or GR)

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.

异构系统中的迁移

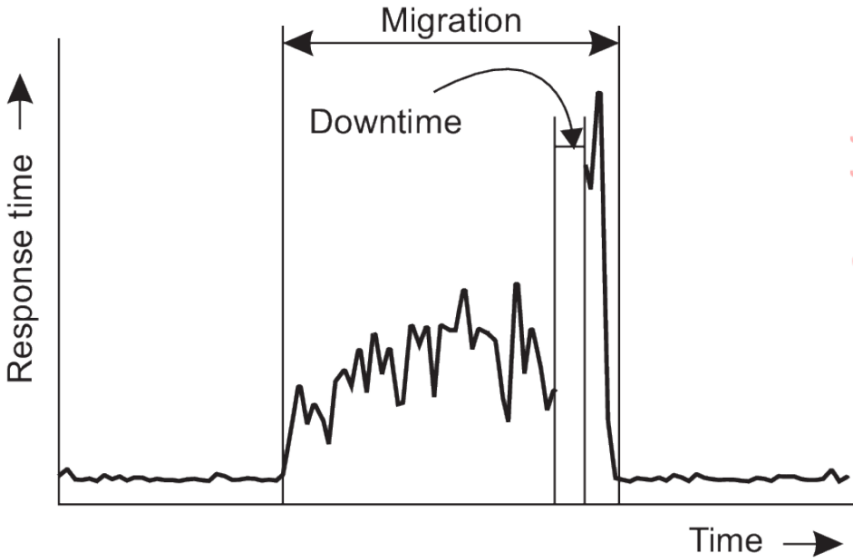
- 主要问题
  - 目标机器可能不适合执行迁移后的代码
  - 进程/线程/处理器的上下文的定义比较依赖于硬件、操作系统和运行时系统
- 仅有的解决方案：
  - 在不同的平台上抽象机器的实现
  - 解释型语言，拥有自己的VM
  - 虚拟机监控器

虚拟机迁移

- 静态迁移
  - 停止当前的虚拟机，迁移内存，然后重新启动虚拟机；
- 动态迁移（live migration）
  - 将内存内容推送到新的机器，在迁移过程中重新发送被修改过的页面；
  - 让新的虚拟机按需拉取内存页面：在新的虚拟机上立即创建进程，并且按需拷贝内存页面；
  - 可能需要重复多次内存复制（dirty during migration）

虚拟机迁移的性能问题

- 一次完整的虚拟机迁移可能需要几十秒或更长
  - 如果需要下载镜像则会更耗时
- 迁移期间，可能有几秒钟的服务完全不可用的状态



Summary

- 多线程
- 虚拟化技术
- 客户端
- 服务器
  - 有状态vs.无状态
  - 服务器发现
  - 请求分发
  - 服务器集群
- 代码迁移
  - 作用、方式
  - 异构系统迁移

Homework Questions

1. 请分析一下进程虚拟机，如JVM，与操作系统虚拟机，包括原生虚拟机和主机虚拟机，各有什么特点?
2. 请分析讨论：基于目录服务器和基于超级服务器的两种服务器发现方法各有什么优缺点?