CompSci131

Parallel and Distributed Systems

Prof. A. Veidenbaum

Today's topics

- Virtualization
- Code migration

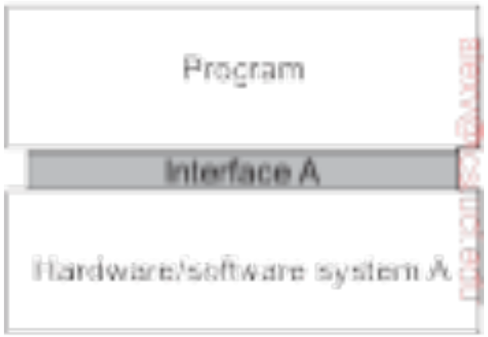
- Reading assignment:
  - Today: 3.2, 3.5
  - Next time: 4.2-4.3

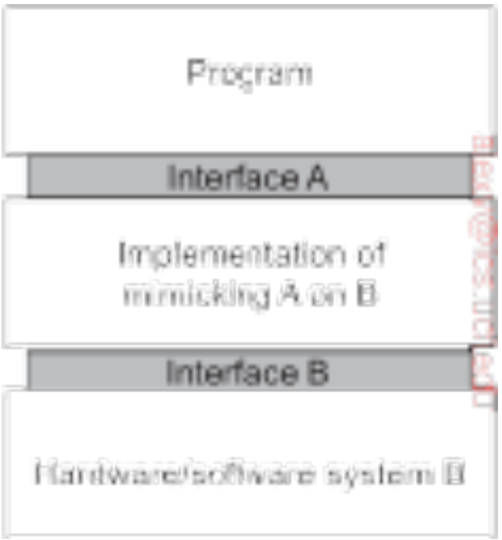
Last Lecture Covered

- **SMP parallel programming**

Virtualization

- DS virtualization helps port sftw to new systems
  - E.g. provide a new interface to old sftw
  - More general, higher level than wrappers
  - Also helps share resources
  - So IBM created a “hypervisor” and could run either OS on it




Program
Interface A
Implementation of mimicking A on B
Interface B
Hardware/software system B
alexv@ics.uci.edu

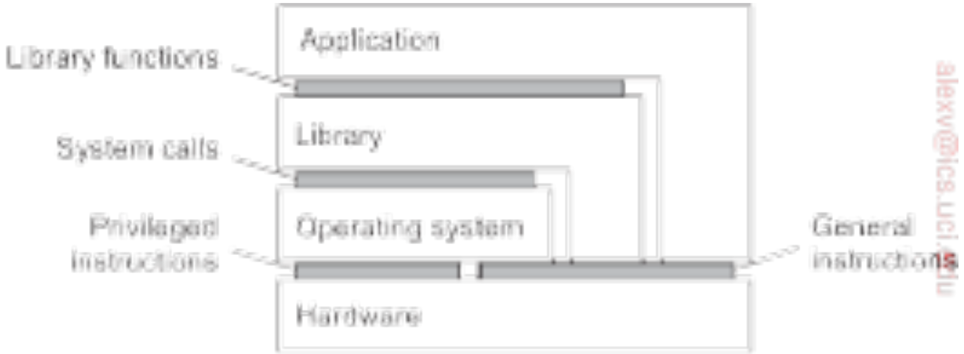
- Networking is pervasive, so can be on remote system

(Resource) Virtualization

- Already saw an OS ‘virtualizing’ a single processor
  - For a process
  - But different OS’s do it differently
- IBM had to maintain backward compatibility and run the old OS on new hardware in the 80’s
  - They also had a new OS
  - So they created a “hypervisor” and could run either OS on it
- The hypervisor is ‘a private OS’ on top of raw hrdw
  - Intel’s term=Virt. Machine Monitor (VMM)
    - Intel provides VM hardware extensions (VMX)
- Will see what processor support is required

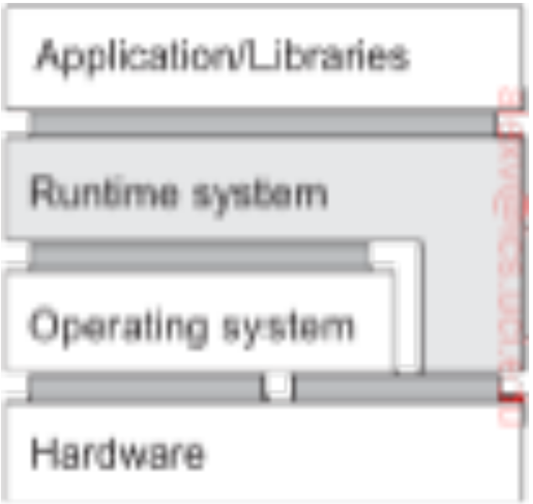
Types of Virtualization

- Computers generally provide 4 types of interfaces
  - E.g. provide a virtual machine
- These interfaces are at 3 different levels
  - Hrdw/Sftw level – instruction set architecture (ISA)
    - User and priveledged instructions
  - OS level – system calls
  - API level – usually library calls
    - May hide some system calls



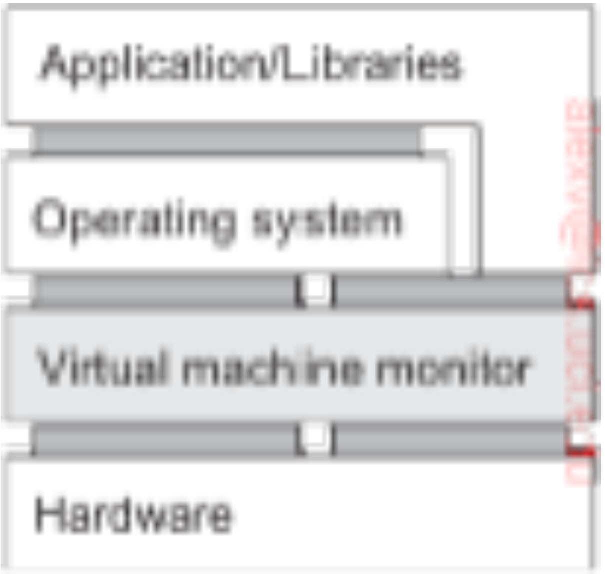
Types of Virtualization

- Virtualization mimics these interfaces
- Two ways of doing it
  1. A runtime system providing an abstract ISA to apps
     - » Java virtual machine - interpreted
     - » Windows apps running on Linux – emulated
       - Quite hard for Windows sys calls
     - » This is virtualization for a single process



Types of Virtualization

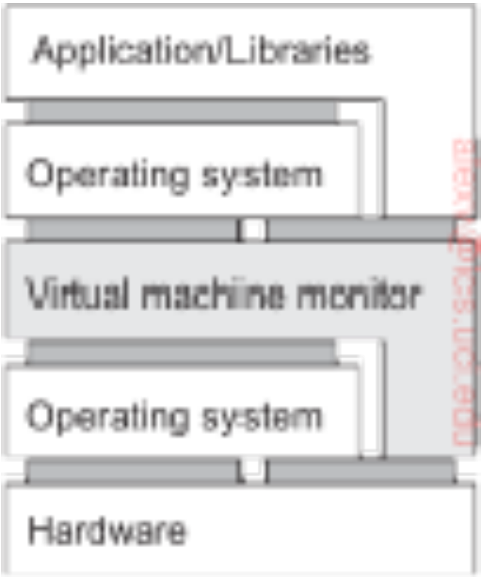
2. A system layer shielding the original hrdw
   - » Offers the hrdw interface to layers above
     - Same as underlying hrdw OR different hrdw
   - » Known as “native VMM” ‘cause it runs on original hrdw
     - Can still clearly see the 4 interfaces



– Note that the VMM interface can be used by multiple apps simulatenaously

Types of Virtualization

- The native VMM has to provide access to and manage various resources
  - E.g. storage, networking, etc through drivers
- But the OS already does it, so why re-implement?
- Another virt. type is known as a hosted VMM



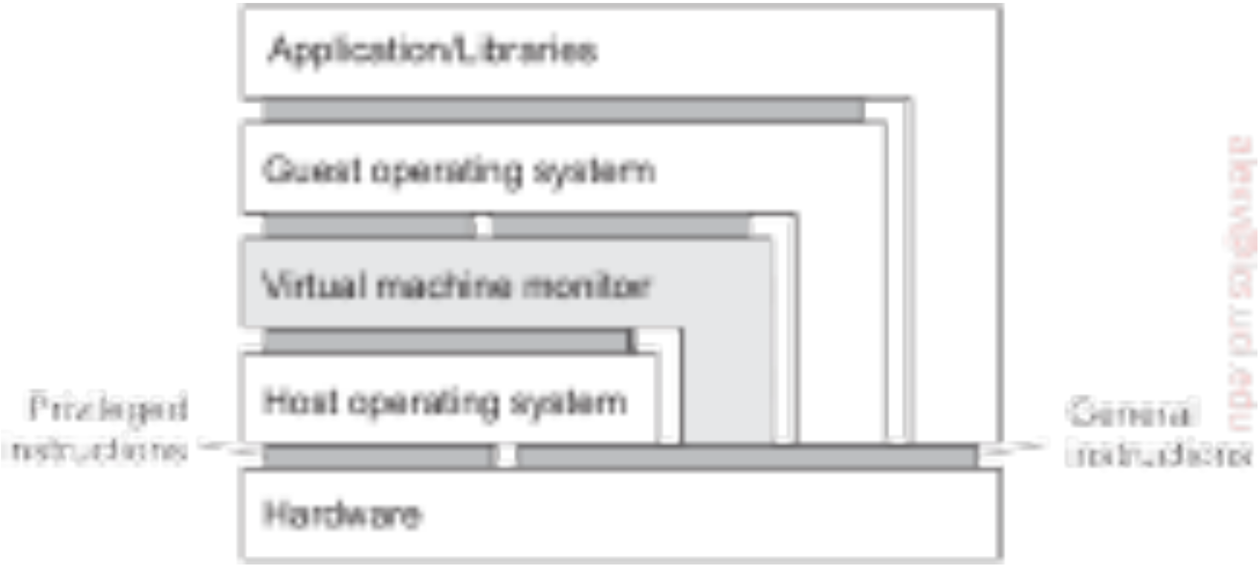
Trusted OS

Partially priveledged

Virtualization Performance

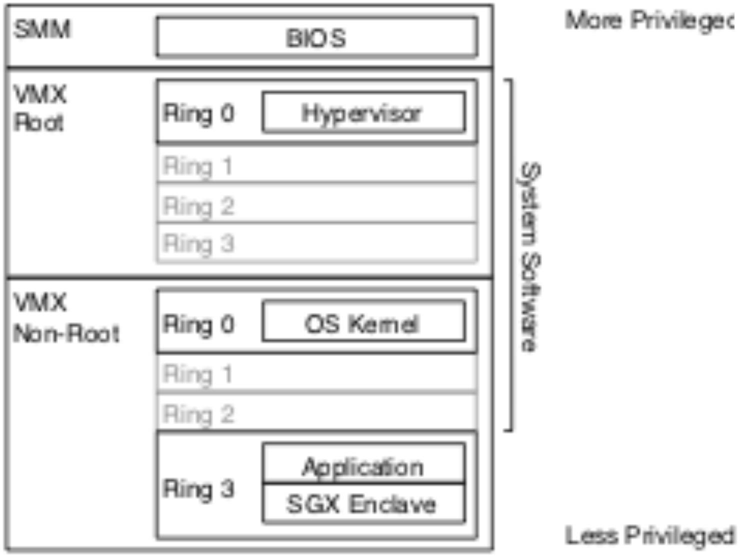
- **Modern virtualized systems are highly efficient**
  - **They run as much code as possible <u>natively</u>**

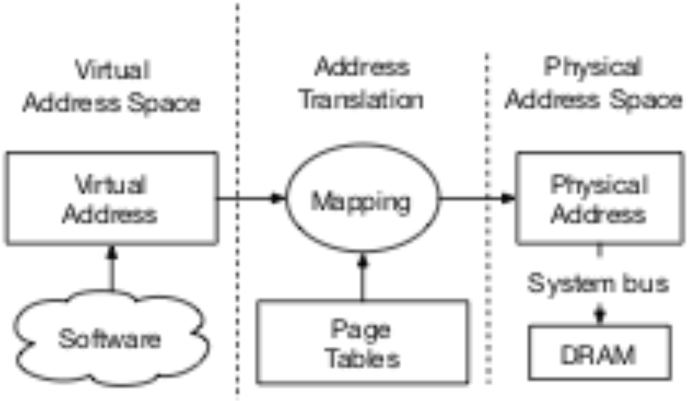
    Directly on underlying hrdw for most non-priveldged instructions

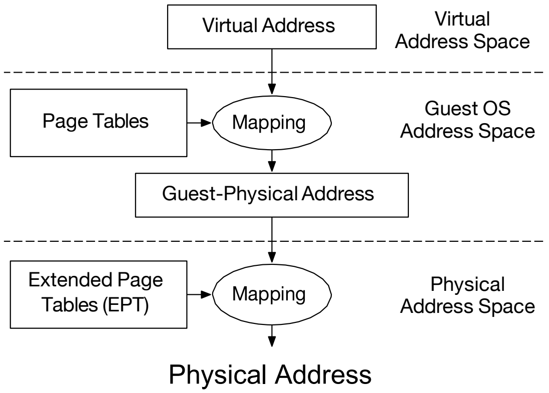


Processor Virtualization

- Hypervisor divies up hardware resources to guest OS’s: page frames in DRAM, access to I/O devices,…







An example of system virtualization

- Recall the concept of Infrastructure-as-a-Service
  - Aka IaaS, is built on virtualization
- Consider Amazon EC2
  - It allows one to create a number of virtual servers
    - Using preconfigured machine images (AMIs)
      - An OS kernel with a number of services
    - One such image is LAMP
      - Linix kernel, Apache Web server, MySQL, PHP libraries
    - A launched AMI is an EC2 instance
  - Each instances gets 2 IP addresses: public & private
- A customer cannot tell
  - Where his code is running
  - Who, if anyone, is she sharing the EC2 resources
  - What fraction of a CPU, etc does she get

Code migration

- It is sometimes desirable to move programs or objects
  - Traditionally done as process migration
    - » Move a process from one DS node to another
    - » Process is a running program!
      - A very difficult and costly move, must be sure it is worth it
        - Usually to get better performance due to load balancing

- Today the goal is to get better loading of each machine
  - Allows data centers to serve more customers -> get more $$$
  - Done by migrating complete virtual machines + their apps

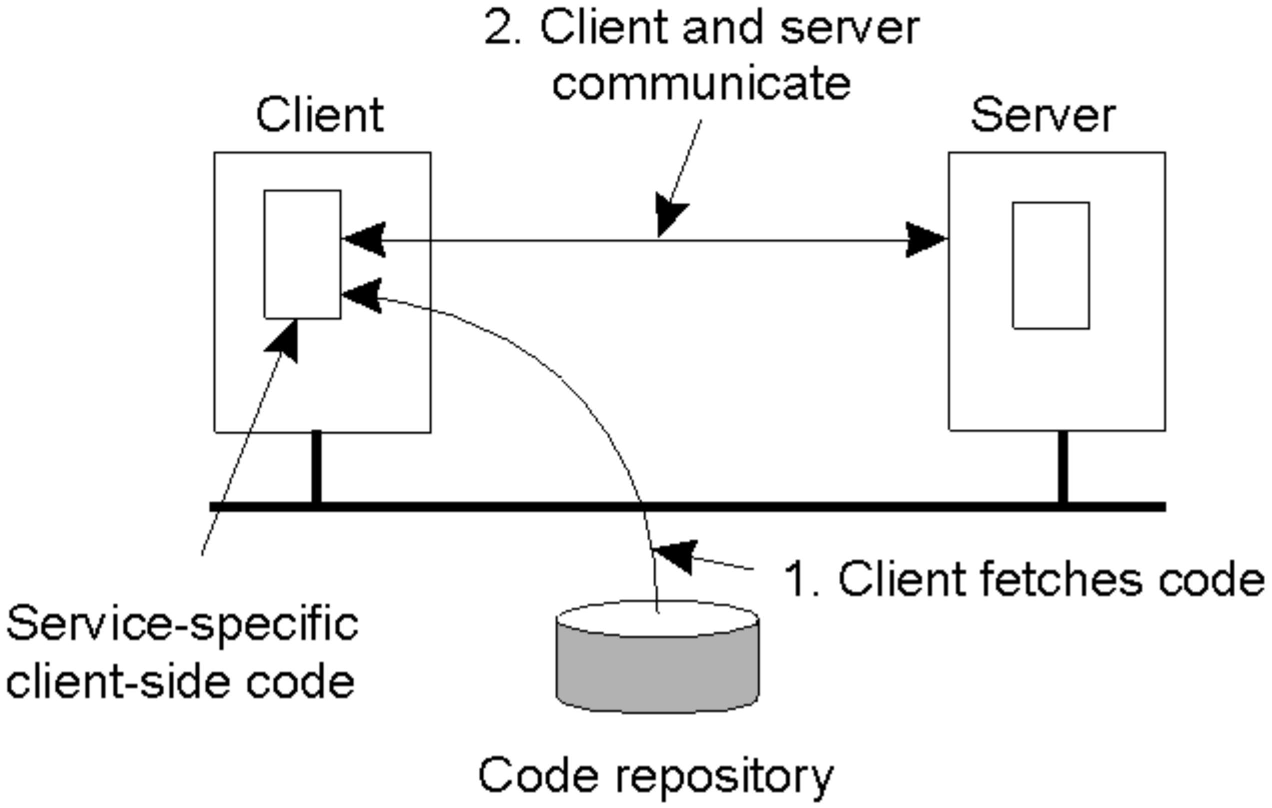
- At the same time, network delays may dominate
  - So it may be worth moving programs to data
    - » E.g. when extensively using a large database

Advantages of code migration

- Allows load consolidation
- Can improve load balancing and thus performance
  - Or exploit parallelism and create many copies to do work
- Reduce communication cost
- Flexibility in design
  - don't need to decide where application runs
  - Can configure clients when they run, not at design time
    - Dynamic client configuration

Dynamic Client Configuration

- **The principle of dynamically configuring a client to communicate to a server. The client first fetches the necessary software, and then invokes the server.**
- **Allows client server protocols to be changed transparently**


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

Models for Code Migration

- Models assume a process consists of
  - – A Code Segment
  - – A Resource Segment
  - – An Execution Segment (state, including data)

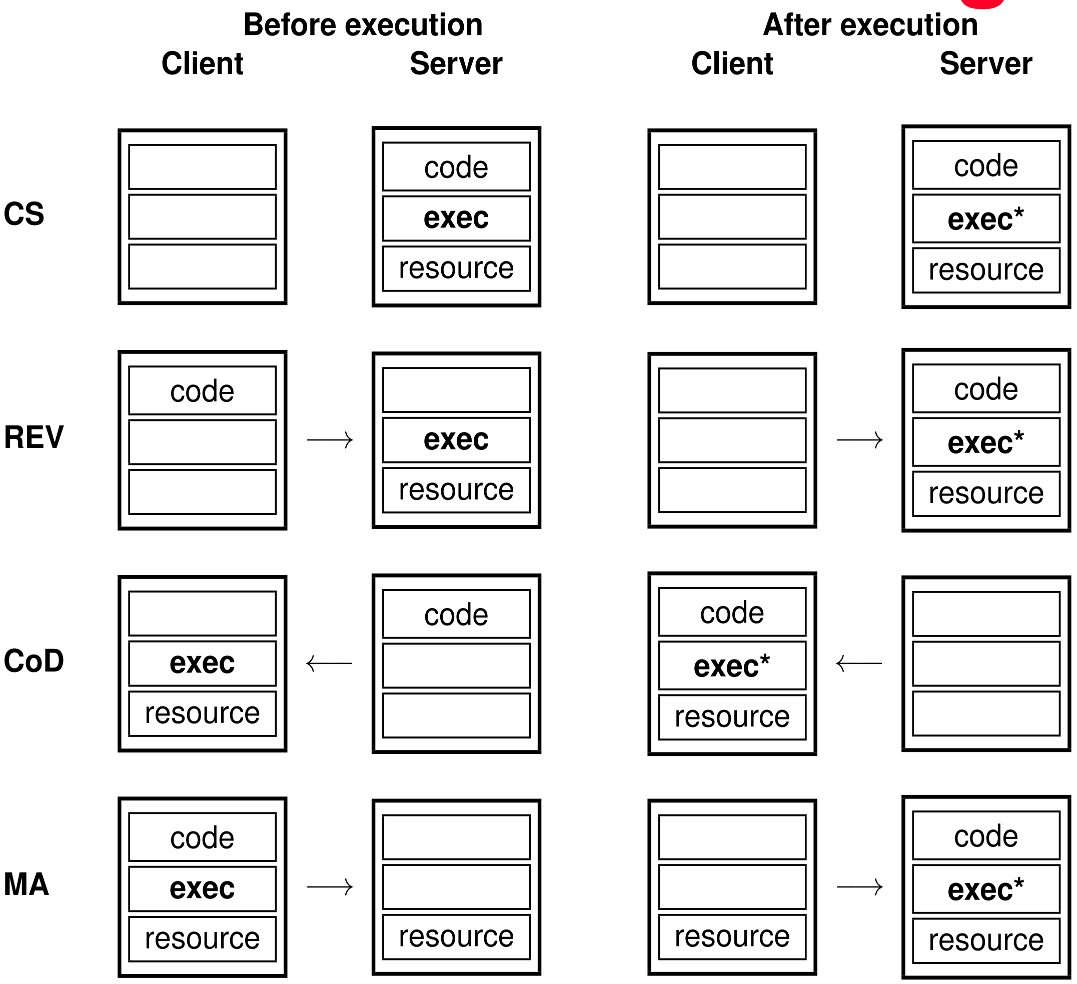
- Weak mobility migrates only the code segment
  - – (Re)starts the program from initial state
    - » Advantage: simplicity
      - Example - Java applets

- Strong mobility also migrates execution segment
  - – Allows a running process to be stopped, moved, restarted

Mobility characteristcs

- Sender-initiated
  - Send code to a server
    - » E.g. send a program to a compute server
- Client-initiated (or receiver-initiated)
  - Browser loads an applet
    - » This is harder! Why?
- Lastly, can run as same process or a new one
  - New process is expensive but provides protection
- For strong mobility can also clone a process
  - An exact copy
  - Runs in parallel with the original process

Alternatives for code migration



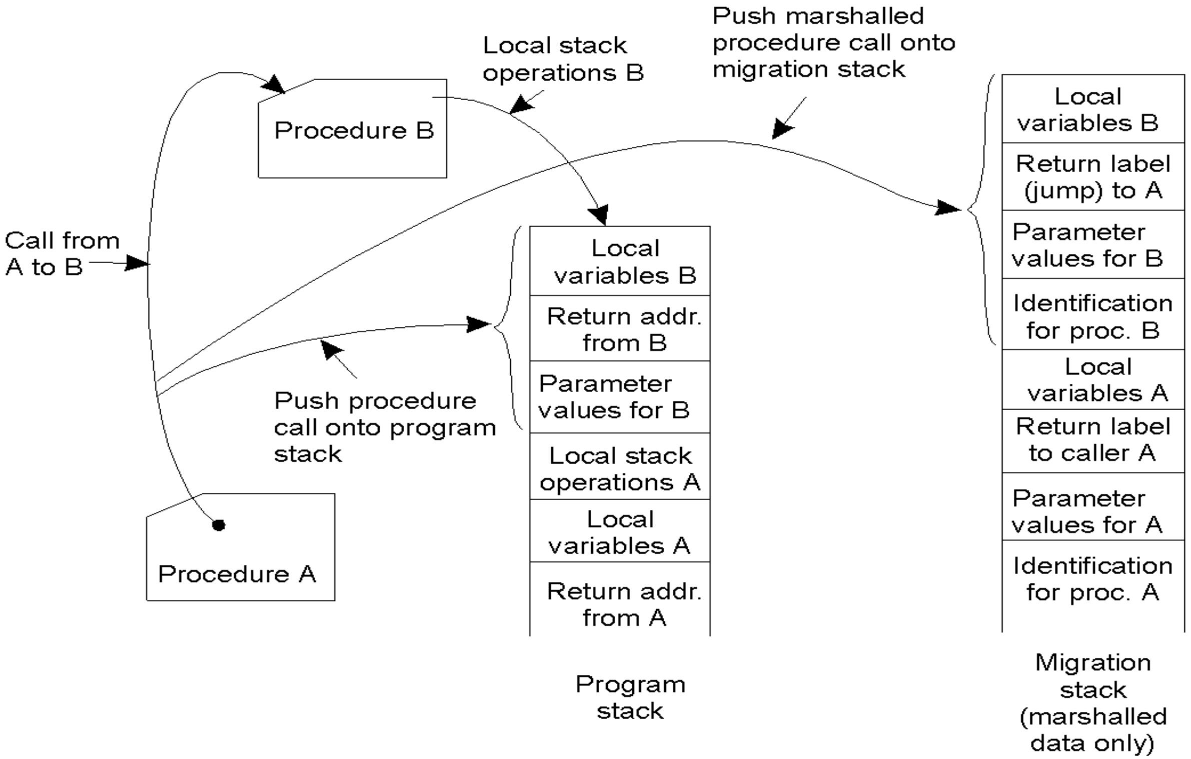
	Before execution			After execution		
	Client		Server	Client		Server
CS			code **exec** resource			code **exec*** resource
REV	code	→	**exec** resource		→	code **exec*** resource
CoD	**exec** resource	←	code	code **exec*** resource	←	
MA	code **exec** resource	→	resource	resource	→	code **exec*** resource

CS: Client-Server
CoD: Code-on-demand

REV: Remote evaluation
MA: Mobile agents

Migration in heterogeneous systems

- Requires executing code on different types of system
- Hard for strong mobility, main problem is migrating the execution segment
- Can use a migration stack to migrate an execution segment, for C and Java
  - Restricted to specific points in the program - function calls
    - » Using machine independent execution stack (migration stack)
      - Maintained in parallel with a regular stack



- **Best supported by a compiler that generates both normal and migration code/stack for C**

- **Another solution is using VMs (e.g. Java),**
  - **An intermediate language, compiler-generated**
  - **A restricted set of libraries**
  - **Protection of local resources**
  - **Interpretation or JIT**