Advanced Topics in Distributed Systems

Dr. Ayman A. Abdel-Hamid

Computer Science Department
Virginia Tech

Processes
(Based on Ch3 in Distributed Systems: Principles and Paradigms, 1/E or 2/E)

Outline

• Processes and Threads
• Thread Implementations
• Threads in Distributed Systems
• Virtualization
• Clients and Servers design issues
• Code Migration
Process versus Thread 1/2

- Why a finer granularity is needed?
- The need for threads?
  - Blocking system call
  - Exploit parallelism
  - Large applications developed as a collection of cooperating programs
    - Can use IPC (Interprocess communication) but faces overhead of context switching because of kernel intervention
  - Easier to structure applications as a collection of cooperating threads

Process versus Thread 2/2

![Diagram showing context switching as the result of IPC]

Figure 3-1. Context switching as the result of IPC
Thread Implementations 1/2

- User-level threads
- Kernel-level threads
- Hybrid approaches
  - Lightweight processes (LWP)

Thread Implementations 2/2

Figure 3-2. Combining kernel-level lightweight processes and user-level threads
Threads in Distributed Systems 1/2

- Multithreaded clients
- Multithreaded servers

Figure 3-3. A multithreaded server organized in a dispatcher/worker model.
Virtualization 1/4

Virtualizing system A on top of system B

Virtualization 2/4

Interfaces offered by Computer Systems
Figure 3-7. (a) A process virtual machine, with multiple instances of (application, runtime) combinations.

Figure 3-7. (b) A virtual machine monitor, with multiple instances of (applications, operating system) combinations.
Clients Anatomy 1/2

- Users interact with remote servers
- Networked user interfaces
  - Application-specific protocol
  - Application-independent protocol
    - Client machine only a terminal
    - No need for local storage
- Thin-client computing (versus thick client computing)

Clients Anatomy 2/2

Diagram showing client-server interaction with application-specific and application-independent protocols.
Servers Anatomy 1/3

- Iterative Server
- Concurrent server
- How a client contacts a server?
  - Well-known endpoints (ports)
  - Lookup the endpoint
    - through a daemon listening to a well-known endpoint
- A separate server for each service?
  - A single superserver
  - When contacted, forks a process to service request

Servers Anatomy 2/3

Diagram showing the process of client-server interaction with a daemon and a superserver.
Servers Anatomy 3/3

- Out-of-band data
  - Through a separate control endpoint
  - Through the same data endpoint
- Stateless server
  - Soft state
- Stateful server
  - Session state versus permanent state

Code Migration

- Traditionally Process migration
  - Load balancing and distribution
  - Migrate parts of client to server
  - Migrate parts of server to client
- Exploit parallelism through the use of mobile agents
- Dynamically configure distributed systems
- How about execution status of migrated processes?
- Migration in heterogeneous systems
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

Process Segments: Code segment (instruction), Execution segment (execution state), and Resource segment (external resources)