CS 550:
Advanced Operating Systems

Code and Process Migration

Ioan Raicu
Computer Science Department
Illinois Institute of Technology

CS 550
Advanced Operating Systems
February 22nd, 2011

Outline

- Code and process migration
  - Motivation
  - How does migration occur?
  - Resource migration
- Distributed scheduling

Motivation

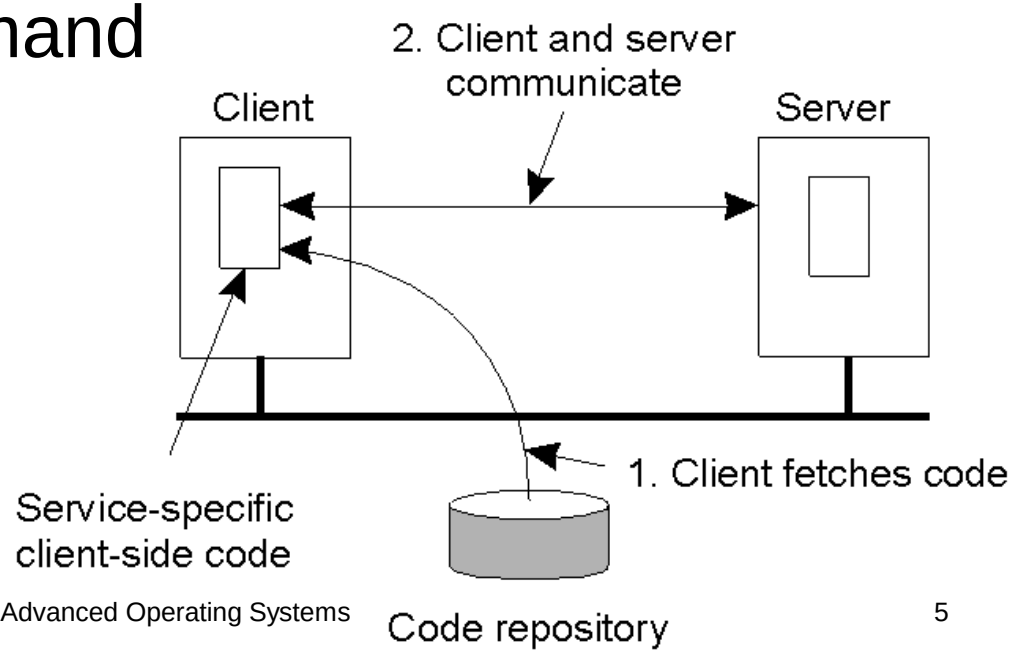
- Key reasons: performance and flexibility
- Process migration (aka *strong mobility*)
  - Improved system-wide performance – better utilization of system-wide resources
- Code migration (aka *weak mobility)*
  - Shipment of server code to client
  - Ship parts of client application to server instead of data from server to client
  - Improve parallelism

Motivation

- Performance
  - The overall system performance can be improved if processes are moved from heavily-loaded to lightly-loaded machines
  - Exploit parallelism, e.g., searching for information in the web through the development of mobile agent, that moves from site to site
  - fault tolerance, e.g., moving from failure-prone to failure-free machines

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
- 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

Distributed Scheduling

- *Challenge*: multiple processing nodes=> scheduling not only is performed locally on each node but also globally across the system
- *Distributed scheduling* (aka load balancing): potentially useful for performance improvement or system utilization
- An Example: Work Stealing

Design Issues

- The measure of load
  - Must be easy to measure
  - Must reflect performance improvement
- Types of policies
  - Static vs. Dynamic vs. Adaptive
- Types of task transfers
  - Preemptive vs. non-preemptive
- Major components
- Three algorithms

Components

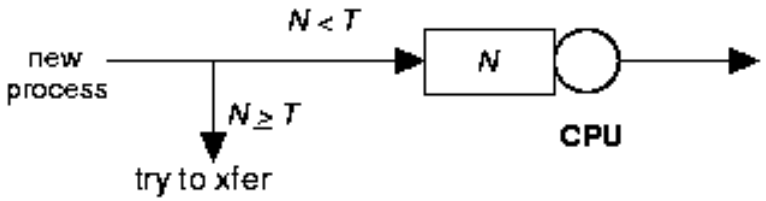
- *Transfer policy*: when to transfer a task?
  - Threshold-based policies are common and easy
- *Selection policy*: which task to transfer?
  - An easy approach
- *Location policy*: where to transfer the task?
  - Polling, random, nearest neighbor
- *Information policy*: when and from where?
  - Demand driven
  - Time-driven
  - State-change-driven

Three Algorithms

- *Sender-initiated*:
  - distribution initiated by an overloaded node.
- *Receiver-initiated*:
  - Distribution initiated by lightly loaded nodes.
- *Symmetric*:
  - Initiated by both senders and receivers.

Sender-initiated Algorithm

- Transfer policy: use threshold



- Selection policy: newly arrived task
- Location policy: three variations
  - Random:
  - Threshold:
  - Shortest:

Sender-initiated Algorithm

- Information Policy: demand-driven.
- Stability: can become unstable at high loads, why?

Receiver-initiated Algorithm

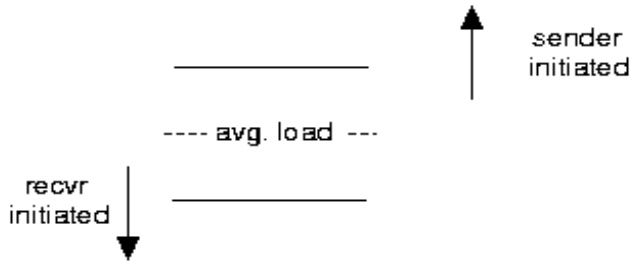
- Transfer policy: If departing task causes load < *T*, find a task from elsewhere
- Selection policy: newly arrived or partially executed task
- Location policy:
  - Threshold:
  - Longest/heaviest:

Receiver-initiated Algorithm

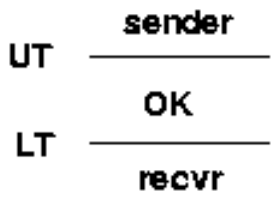
- Information policy: demand-driven.
- Stability: Not unstable since lightly loaded nodes initiate the algorithm
- Drawbacks:
  - ?

Symmetric Algorithm

- Nodes act as both senders and receivers: combine previous two policies without change
  - Use average load as threshold



- Improved symmetric policy: exploit polling information
  - Two thresholds: *LT, UT, LT <= UT*
  - Maintain sender, receiver and OK nodes using polling info
  - Sender: poll first node on receiver list …
  - Receiver: poll first node on sender list …



Comments on Distributed Scheduling

- If a system never gets highly loaded, which one is better?
- In case of high loads, which one is better?
- In case of widely fluctuating loads, which one is better?

Example: Work Stealing

- Discussion

Summary

- Code and process migration
  - Motivation
  - How does migration occur?
  - Resource migration
- Distributed scheduling
  - Sender-initiated
  - Receiver-initiated
  - Symmetric
- Readings:
  - Chpt 3 of AST

Questions