Operating Systems
Spring, 1999
Dr. Dennis Kafura

Computer Science 5204

Course Overview
Concurrency

1. Sources, Systems, and Issues
2. Problems and Solutions
Sources of Concurrency

multiple physical processors
Sources of Concurrency

network

node

node
Sources of Concurrency
Motivations for Exploiting Concurrency

• performance
• simplicity
• availability
• controllability
• autonomy
• asynchrony
• required
Contemporary Systems

• **Fixed computation-transportable data**
  • sockets
  • remote procedure call
  • object based
    • CORBA (OMG)
    • RMI (Java)
    • Com/Dcom (MS)

• **Transportable computation-transportable data**
  • metacomputing
    • Globus
    • Legion
  • mobile code/mobile objects/agents
    • Voyager
    • Aglets
Major Topics in Course

• **Synchronization and Coordination**

  • safety vs. liveness
    • safety: insuring consistency of system
    • liveness: insuring progress of systems activity

  • conservative vs. optimistic:
    • conservative: refuse to perform any action unless the system's consistency can be guaranteed

    • optimistic: perform actions with the expectation of their successful completion and be prepared to recovery to a consistent state if they cannot be completed
Major Topics

• Theory/Foundations
  – proofs
  – algebraic models

• Scheduling

• Fault Tolerance/Recovery

• Protection/Security

• Languages and Tools:
  • programming languages: Java
  • modeling/analysis languages: CCS
Consistency and Synchronization
Consistency and Synchronization

• characterize interactions which may jeopardize the safety of a concurrent system

• describe the basic forms of synchronization required to control these interactions

• identify classical concurrent programming problems

• illustrate how the different forms of synchronization are programmed in Java
Interference

Definition: Interference refers to the loss of consistency in an object shared among concurrent activities due to the unsynchronized interleaving of action by these concurrent activities on the shared object.

A point object represents the screen coordinates of an object displayed on the screen (e.g., The cursor)

Suppose there are concurrent threads that alter the point’s coordinates and other concurrent threads that examine the point’s coordinates.
Interference?

Point
Interference?
Interference?
Bernstein’s Condition

Consider two threads, T1 and T2, which have read sets and write sets denoted by \( R_1, R_2 \) and \( W_1, W_2 \), respectively. Then T1 and T2 interfere if:

\[
(R_1 \cap W_2) \cup (R_2 \cap W_1) \cup (W_1 \cap W_2) \neq \emptyset
\]
Forms of Synchronization

- mutual exclusion - preventing concurrent access to shared objects to preserve the consistency of the object

- condition synchronization - blocking attempted operations on a shared object until that object is in a state where the operation will preserve the consistency of the object
Classical Problems

Exclusive Use

inc()
dec()

Counter

Producer - Consumer

Producers

Buffer

Consumers
Classical Problems

Reader-Writer

Writer → Database

Reader

Reader

Dining Philosophers

philosopher

philosopher

philosopher
Java as a Concurrency Programming Language

• Language:
  • language concepts for threads and synchronization
  • platform independent

• Libraries for basic network programming
  • sockets
  • Remote Method Invocation

• Used to implement distributed systems
  • Aglets
  • Voyager