Lesson 1

Concurrency

Terminology

Concurrency in Systems

Problem Examples

Solution Considerations

11.1.2012 Copyright Teemu Kerola 2012

Chapter 1 [BenA 06]

Concurrency Terminology

- Process, thread
- "Ordinary" program
  - Sequential process, one thread of execution
- Concurrent program
  - Many sequential processes, that may be executed in parallel
  - Web-application, distributed on many systems
- Multiprocessor system, parallel program
  - Many sequential or concurrent processes are executed in parallel
  - Many architectures, no winner yet
- Distributed system, distributed program
  - No shared memory
  - Interconnected systems

Lesson 1

Concurrency Terminology

- Process, thread
- "Ordinary" program
  - Sequential process, one thread of execution
- Concurrent program
  - Many sequential processes, that may be executed in parallel
  - Web-application, distributed on many systems
- Multiprocessor system, parallel program
  - Many sequential or concurrent processes are executed in parallel
  - Many architectures, no winner yet
- Distributed system, distributed program
  - No shared memory
  - Interconnected systems

11.1.2012 Copyright Teemu Kerola 2012

Concurrency at HW-level

- Processor
  - Execute many instructions in parallel
  - Execute many threads in parallel
  - Execute many processes in parallel
- System
  - Many processors/display processors
  - Many I/O devices
- LAN or WAN
  - Many systems (in clusters)
- Internet and other networks
  - Many sub-systems

11.1.2012 Copyright Teemu Kerola 2012

Problem

- Moore’s Law will not give us (any more) faster processors
  - But it gives us now more processors on one chip
- Multicore CPU
- Chip-level multiprocessor (CMP)

11.1.2012 Copyright Teemu Kerola 2012

The Multicore Challenge

- We have a heat-barrier dead-end to develop simple to program single core chips
  - So, we leap to multicore chips in pursuit for ever higher processing power
- Parallel Challenge: how to use these multicore computers efficiently to speed up computing?
  - Concurrent programming
  - We should have launched a parallel programming “Manhattan Project” a long time ago
- Would need now 100’s of millions ($), not 10’s of millions ($) per year for long term funding

11.1.2012 Copyright Teemu Kerola 2012

Lecture 1: Concurrency

**Lecture 1: Concurrency**

### Concurrent Programming at HW-level
- **Machine language code**
  - Many instructions at execution concurrently
  - Logically “one at a time” (von Neumann arch.)
  - At least one “instruction cluster” at a time
  - Program execution may stop/pause after any instruction
- **High level programming language code**
  - Process (thread) switch can occur at any time
  - No “handle” on process switch times (in general)
  - Operating system & external events decide
  - Need to synchronize with other programs
  - Need to communicate with other programs
  - Need to get handle to process switch occurrences
  - Other processes may be in execution at the same time

### Problem Free Concurrency?
- **No problems at all?**
  - Concurrent threads in execution
  - No shared data, no I/O (or private I/O)
  - No communication, no synchronization
- **No shared data, but private data in shared memory**
  - No congestion may be problem
  - Concurrency problem (bus use) solved in HW
  - Slows down execution
  - Communication/synchronization is needed eventually
  - Combine results from concurrent threads

### Concurrency Problems
- **Keep data consistent**
  - Update all fields of shared data
  - Complete writing to a buffer before reading it starts
- **Synchronize with someone**
  - Complete writing before reading starts
  - Give money only after bank card is taken
  - Compile new Java class before execution resumes
  - Do not wait forever, if the other party is dead
- **Communicate with someone**
  - Send a short message to someone
  - Send data to be processed to someone
  - Send 2 GB data for remote processing, wait for result

### Concurrency Examples
- **PlayStation 3**
  - Use effectively Cell processor with 9 cores
  - Two different processor architectures
  - Divide-and-conquer or filtering approach?
- **Desktop PC**
  - Use effectively 4 processors and a graphics adapter to generate graphics for fast moving game
  - Utilize all 4 processors (each with 2 cores?)
  - Control shared access to game data base
    - In memory? In disk?
    - In a file server in Japan?
- **Multithreaded Java program on a multiprocessor system**
  - Access to shared data structures
  - Synchronization between threads
  - Displaying these slides from file server
- **Linux Beowulf 6 node cluster**
  - How to solve weather forecast Hirlam model as fast as possible?
  - How to best distribute data?
  - Solution scalable to 100 or 1000 nodes?
- **Web server**
  - How to serve 1000 or 10000 concurrent requests with 100 file servers
    - Most reads, but some writes to same files?
    - How to guarantee consistent reads with simultaneous writes?
Concurrency Examples

- Operating system
  - How to keep track of all concurrent processes, each with multiple threads?
  - What type of concurrency control utilities should be offered to user programs?
  - Which utilities offered to OS services?
  - How do we guarantee that the system does not “freeze”?
  - How to write an 8-disk disk controller device driver?
  - How do I guarantee, that nothing disturbs an ongoing process switch?

Concurrency Problem Solution Level

- Processor level, i.e., below machine language level
  - HW solutions, automatic, no errors
    - Need to understand, this is where it really happens
- Machine language level
  - Specific (HW) machine instructions for concurrency solutions
    - Clever solutions without specific instructions
    - Need to be used properly, this is where it really happens
- Program level, i.e., programming language level
  - SW solutions, many possibilities for error
    - Solve problem by programming the solution yourself
      - Very error prone
      - Requires priviledged execution mode (usually)
    - Solve problem directly by invoking certain available library services
      - Error prone – may invoke wrong routines at wrong times
    - Solve problem by letting available library service do it all for you
      - Not suitable always – may not fit to your problem well

Library Solutions for Concurrency Problems

- Programming language run-time library
  - E.g., Java thread management
  - Usually within one process (in one system)
  - Any program can use
  - May be implemented directly or with OS-libraries
- Operating systems services (libraries)
  - Any process can use these, not so portable across OS’s
  - Usually only choice between many processes
  - Exception: programming language library that implements its services with OS
  - Only choice between many systems
  - May need privileged execution mode
  - Some services reserved only for OS programs or utilities

Basic Concurrency Problem Types

- Mutex
  - One or more critical code segments, i.e., critical section
    - At most one process executing critical section (of code) at any time
    - I.e., at most one process holds this resource (code) at any time
- Synchronization
- Communication

Basic Concurrency Problems

- Dining philosophers
  - Think-eat cycle
  - Need 2 forks to eat
  - Can take one fork at a time
  - No discussion
  - Question: what protocol to use to reserve forks?
  - Multi-process synchronization
  - Avoid deadlock
  - Avoid starvation
  - Prove correctness

Concurrency Problem Solutions

- Process level, i.e., below machine language level
  - HW solutions, automatic, no errors
    - Need to understand, this is where it really happens
- Machine language level
  - Specific (HW) machine instructions for concurrency solutions
    - Clever solutions without specific instructions
    - Need to be used properly, this is where it really happens
- Program level, i.e., programming language level
  - SW solutions, many possibilities for error
    - Solve problem by programming the solution yourself
      - Very error prone
      - Requires privileged execution mode (usually)
    - Solve problem directly by invoking certain available library services
      - Error prone – may invoke wrong routines at wrong times
    - Solve problem by letting available library service do it all for you
      - Not suitable always – may not fit to your problem well

Basic Concurrency Problems

- Sleeping barber
  - One barber, one barber chair
  - Waiting room with n chairs
  - No customers?
    - Barber sleeps until arriving customer wakes him up
  - Customer arrives?
    - Barber sleeps? Wake him up?
    - Barber busy and empty chairs? Reserve one and wait.
  - o/w leave
  - Question: what protocol for barber & customers?
    - Inter-process communication, synchronization?
    - Avoid deadlock and starvation
Basic Concurrency Problems

- Bakery algorithm
  - Baker, ticket machine
  - Each arriving customer gets a ticket number
  - Customers are served in increasing ticket number order
  - Question: how to implement the ticket machine
    - In distributed system?
    - With/without shared memory?
  - Multi-threaded mutual exclusion
  - Critical section use order?

- Producer-Consumer
  - Bounded shared buffer area
  - Producers insert data items
  - Consumers take data items in arriving order
  - Full buffer?
    - Producer blocks
    - Empty buffer?
    - Consumer blocks
  - Question: protocol for producer/consumer
  - Communication, synchronization
    - Unix/linux "pipe"
  - Avoid deadlock, starvation

System Considerations

- One or more (executing) cores?
- Different threads in same process?
  - Who controls thread switching?
    - Application, language library or OS?
- Different processes in same system?
  - Many threads in each process?
- Different threads/processes in processors grid?
  - No shared memory
- Different threads/processes in distributed system?
  - No shared memory
  - Large communication delays

Solution Considerations

- Solution at application level without HW support
  - Do everything from scratch
- Solution at application level with HW support
  - Use special machine language level instructions or structures
- Solution at operating system level
  - Use utilities in operating system library
- Solution at programming language level
  - Use utilities in programming language library
- Solution at network level
  - Use utilities in some network server
  - Need to understand what really happens

Summary

- Terminology
- Concurrency in systems
- Concurrency problem examples
  - Educational: philosophers, barber, bakery
  - Practical: consumer-producer, readers-writers
- Solution considerations