10

Summary

Uwe R. Zimmer - The Australian National University

Summary

Concurrency – The Basic Concepts

- **Forms of concurrency**
- **Models and terminology**
  - Abstractions and perspectives: computer science, physics & engineering
  - Observations: non-determinism, atomicity, interaction, interleaving
  - Correctness in concurrent systems
- **Processes and threads**
  - Basic concepts and notions
  - Process states
- **Concurrent programming languages:**
  - Explicit concurrency: e.g. Ada, Chapel
  - Implicit concurrency: functional programming – e.g. Haskell, Caml

Summary

Mutual Exclusion

- **Definition of mutual exclusion**
- **Atomic load and atomic store operations**
  - … some classical errors
  - Decker's algorithm, Peterson's algorithm
  - Bakery algorithm
- **Realistic hardware support**
  - Atomic test-and-set, Atomic exchanges, Memory cell reservations
- **Semaphores**
  - Basic semaphore definition
  - Operating systems style semaphores

Summary

Communication & Synchronization

- **Shared memory based synchronization**
  - Flags, condition variables, semaphores,
    conditional critical regions, monitors, protected objects.
  - Guard evaluation times, nested monitor calls, deadlocks,
    simultaneous reading, queue management.
  - Synchronization and object orientation, blocking operations and re-queuing.
- **Message based synchronization**
  - Synchronization models
  - Addressing modes
  - Message structures
  - Examples

Summary

Non-Determinism

- **Non-determimism by design**:
  - Benefits & considerations
- **Non-determinism by interaction**:
  - Selective synchronization
  - Selective accepts
  - Selective calls
- **Correctness of non-deterministic programs:**
  - Sources of non-determinism
  - Predicates & invariants

Summary

Data Parallelism

- **Data-Parallelism**
  - Vectorization
  - Reduction
  - General data-parallelism
- **Examples**
  - Image processing
  - Cellular automata

Summary

Scheduling

- **Basic performance scheduling**
  - Motivation & Terms
  - Levels of knowledge / assumptions about the task set
  - Evaluation of performance and selection of appropriate methods
- **Towards predictable scheduling**
  - Motivation & Terms
  - Categories & Examples

Summary

Safety & Liveness

- **Liveness**
  - Fairness
- **Safety**
  - Deadlock detection
  - Deadlock avoidance
  - Deadlock prevention
- **Atomic & Idempotent operations**
  - Definitions & implications
- **Failure modes**
  - Definitions, fault sources and basic fault tolerance

Summary

Distributed Systems

- **Networks**
  - OSI, topologies
  - Practical network standards
- **Time**
  - Synchronized clocks, virtual (logical) times
  - Distributed critical regions (synchronized, logical, token ring)
- **Distributed systems**
  - Elections
  - Distributed states, consistent snapshots
  - Distributed servers (replicates, distributed processing, distributed commits)
  - Transactions (ACID properties, serializable interleavings, transaction schedulers)

Summary

Architectures

- **Hardware architectures - from simple logic to supercomputers**
  - logic, CPU architecture, pipelines, out-of-order execution, multithreading, ...
- **Data-Parallelism**
  - Vectorization, Reduction, General data-parallelism
- **Concurrency in languages**
  - Some examples: Haskell, Occam, Chapel
- **Operating systems**
  - Structures: monolithic, modular, layered, µkernels
  - UNIX, POSIX

Summary

Exam preparations

Helpful

- **Distinguish** central aspects from excursions, examples & implementations.
- **Gain** full understanding of all central aspects.
- Be able to **categorize** any given example under a general theme discussed in the lecture.
- **Explain** to and **discuss** the topics with other (preferably better) students.
- Try whether you can **connect** aspects from different parts of the lecture.

Not helpful

- Remembering the slides word by word.
- Learn the Chapel / Unix / Posix / Occam / sockets reference manual page by page.