Introduction to Concurrency

Forms of concurrency

- Literally: running together in space, as parallel lines; going on side by side, as proceedings occurring together; or, as events or circumstances, existing or arising together; compound, associated (from medieval Latin *concurrentium*).

- Technically: "concurrent" is usually defined as:
  - Two events are considered concurrent if their strict temporal sequence (i.e. one event has fully terminated before the other one starts up), then these two events are considered concurrent.

- "Concurrent" is technically defined negatively as:
  - If there is no observer who can identify two events occurring in strict temporal sequence, i.e. one event has fully terminated before the other one starts up, then these two events are considered concurrent.

A computer scientist’s view on concurrency

Terminology for physically concurrent machines architectures:

- Multiple instruction, multiple data (MIMD)
- Single instruction, multiple data (SIMD)
- Multiple instruction, single data (MISD)
- Single instruction, single data (SISD)

- A computer scientist’s view on concurrency

Add physical/real concurrency

A computer scientist’s view on concurrency

- Parallel Machines & distributed operating systems
- Pipelined processors
- Distributed operating systems
- Embedded processors
- Multi-processors or computer networks

Each computer system can be modeled by the following four aspects:

- Multiple physical, coupled, dynamical systems
- Physical, coupled, dynamical systems
- Multiple physical, coupled, dynamical systems
- Physical, coupled, dynamical systems

- Multiple physical, coupled, dynamical systems
- Physical, coupled, dynamical systems
- Multiple physical, coupled, dynamical systems
- Physical, coupled, dynamical systems

- Multiple physical, coupled, dynamical systems
- Physical, coupled, dynamical systems
- Multiple physical, coupled, dynamical systems
- Physical, coupled, dynamical systems

- Multiple physical, coupled, dynamical systems
- Physical, coupled, dynamical systems
- Multiple physical, coupled, dynamical systems
- Physical, coupled, dynamical systems
Models and Terminology

The concurrent programming abstraction

Atomic operations:
Correctness proofs / designs in concurrent systems rely on the assumptions of:
• Atomic operations (detailed discussion later)

1. Processes
2. Threads
3. Control-flows

Address spaces:
Processes:名誉 applies to and are sufficient for concurrent systems?

1 CPU per task, CPU per thread

© 2020 Uwe R. Zimmer, The Australian National University page 179 of 758 (chapter 1: “Introduction to Concurrency” up to page 202)
Introduction to Concurrency

Process states
- Ready: the task is ready to run, but not yet admitted for execution
- Not Ready: the task is not yet considered by any dispatcher
- Running: the task is holding a CPU and executing
- Blocked: ready to run, but waiting for admission
- Waiting: ready to run, but waiting for admission
- Scheduling: ready to run, but suspended
- Terminated: the task has finished

Process management
- Scheduling and suspending can cause race for independent variables

Concurrent programming languages
- Support for concurrency
- Support for concurrent management (task inclusion, ...) of the child process to the creating process (the 'parent' process)
- Support for communication (message passing, shared memory), ...
- Support for protection (data, memory, devices, ...) of the child's task to the parent's task

Languages with implicit concurrency: e.g. functional programming
- Implicit concurrency in some programming schemes
- QuickSort in a functional language (here: Haskell):