Concurrency, intro

Thomas Plagemann

Slides from Tore Larsen
(University of Tromsø)

Why is concurrency tricky

- We use threads or processes as powerful concept to keep threads of control apart, looking at sequential processes instead of arbitrarily interleaved thread of execution
- Yet, in reality interleaved, any process may be interrupted at any time
- Subtle, hard-to-identify, programming errors including race-conditions are introduced

What remedies do we need?

- Low level mechanisms to temporarily prohibit the interleaving of threads, removing the vulnerability of being interrupted
- Higher-level programming constructs to make programming
  - more convenient
  - less error-prone
  - … and other requirements, incl. Efficiency

Low level mechanisms

- Atomic read and atomic write, memory coherency (…hang on)
- Turning interrupts off
- Atomic read-write operations

High level mechanisms

- Semaphors
- Mutexes
- Monitors
- Message passing

Progress

- This week
  - Motivation plus low-level stuff
- Next week(s)
  - The higher level constructs

Two basic issues

- Protecting control flows in the multitasking environment
- Memory coherency
  - Quality of computer system we're using
  - "A read from any given address always returns the value of the latest write to that address"
  - Guaranteed by HW for single CPU systems and shared memory multiprocessors (SMPs)

Two issues (cont.)

- Memory coherency
  - Provide for memory coherency. The machine must make sure that all of the processing nodes have an accurate picture of the most up-to-date memory (for example: if Processor A has a piece of dirty data in its cache and has not written it back to memory, all other processors must know this).
  - HW guarantee comes at a cost and/or performance penalty, particularly for SMPs. We should try to reduce the incurrence of performance penalties
- Consistency
  - Rules for allowing memory references to be reordered, that may lead to observed differences in memory state by multiple processors.

Literature

- A lot exists!
- Early/first(?) textbook devoted to concurrency was Per Brinch Hansen “The Architecture of Concurrent Programs”
    - http://web.syr.edu/~pbhansen/html/book2.html
- Ben-Ari “Principles of Concurrent and Distributed Programming,” is an old title that many still find useful
    - http://www.amazon.com/exec/obidos/tg/detail/-/013711821X/qid=1063188412/sr=1-4/ref=sr_1_4/104-5762746-9774342?v=glance&s=books
- Fred Schneider, “On Concurrent Programming” (strong on theory aspects)
    - http://www.springerny.com/detail.tpl?cart=989601891383187&ISBN=0387949429