Threads

CS3100

Kenneth Sundberg

Thread of Execution

- A stream of instructions executed in a sequentially consistent manner
- A process may have many threads of execution
- Provides
  - Performance
  - Responsiveness

User Threads v. Kernel Threads

- Threads can be managed by OS
- Threads can be managed by User Library

Speedup

- Compares the wall time of a parallel implementation to the wall time of the *best* sequential implementation
- Speedup can not be better than linear
  - Hardware effects
  - Incorrect sequential implementation
  - Search artifacts

Efficiency

- The fraction of time that a processor is usefully employed
- Speedup / p

Amdahl's Law

- Speedup has an asymtotic limit
- $S(n) = \frac{1}{(1-P)+\frac{P}{n}}$
- $\lim_{n\to\infty} S(n) = \frac{1}{1-P}$

std::thread

Data Race

- When two or more threads of execution access the same memory and at least one of them is a writer there is a data race
- Data Races (Race Conditions) are EVIL$^{\text{TM}}$

const and copy

- No race conditions if there are no writers
- No race conditions if each thread has its own copy

Critical-Section Problem

- If two or more threads must access the same memory, and one of them must write
- Then they must not use the memory at the same time
- This conflicted access is the a *critical section*

Critical-Section Solutions

- Mutual Exclusion - Only one thread in the section at a time
- Progress - Selection of which process can enter a critical section cannot be postponed indefinitely
- Bounded Waiting - Once a thread has requested to enter the critical section there is a bound on how long it must wait to enter

Atomic Swap Solution

- Given an atomic swap solve the Critical-Section problem

Mutex

- std::mutex
- std::lock_guard – RAII object for holding a mutex

Atomics

- std::atomic<T>

Textbook sections covered:

- Section 4.4 (frame 7)
- Section 5.2 (frame 10)