Projects

- Topic:
  - choose from my suggestions or
  - define your own project
- On your own or in groups of two
- Pick a project by March 28
- Presentations: May 5 and May 12
- Final reports: May 16

Projects

Two options:

- seminar-based (no group work)
  - study a set of coherent papers
  - summarize in a report (6 pages)
  - presentation at the end of the semester
- implementation-based (groups of up to 2 people)
  - solve a specific problem related to concurrent programming
  - summarize in a report (4 pages)
  - presentation at the end of the semester

Projects

Two options:

- seminar-based (no group work)
  - study a set of coherent papers
  - summarize in a report (6 pages)
  - presentation at the end of the semester
- implementation-based (groups of up to 2 people)
  - solve a specific problem related to concurrent programming
  - summarize in a report (4 pages)
  - presentation at the end of the semester

Project Suggestion 1: Performance Analysis of Concurrent Programs

1. Pick a problem with at least three-four different solutions
   a. Lock implementations
   b. Data structures: queues, stacks, sets…

2. Examine the performance of the solutions in different settings:
   a. small number of threads vs large number of threads
   b. 2 cores, small amount of memory (laptop) vs. many cores, large memory/cache (server)
   c. different usage models
   d. input that generates little vs. input that generates lots of contention

3. Find a hybrid solution that works well in a particular setting

Project Suggestion 2: Performance/Conciseness Evaluation of Concurrent Programming Paradigms

1. Pick a problem or algorithm with a non-trivial concurrent solution

2. Implement the algorithm using different concurrency paradigms
   a. traditional shared-memory concurrency
   b. software transactional memories
   c. actors

3. Compare performance and implementation complexity of the different solutions

Project Suggestions 3 (challenging): Implement Scala Library for Higher-Order Concurrent Programming

- Study the higher-order concurrent programming model provided by Concurrent ML
- Implement this model in a Scala library
  - build on top of the Akka library or
  - directly on the JVM

Project Suggestions 4 (challenging): Verification of a concurrent data structure

1. Pick an implementation of a concurrent data structure: a stack, a queue, a set, ..
2. Pick a verification tool: for example: Chalice
3. Prove that the implementation is linearizable