CS-735/835: Introduction to Parallel and Distributed Programming

Michel Charpentier
August 31, 2014

1 Catalog Description

Programming with multiple processes and threads on distributed and parallel computer systems. Introduces programming tools and techniques for building applications on such platforms. Course requirements consist primarily of programming assignments.

2 Attributes

• This course is one of the CS electives designated as *implementation intensive*.

• This course can be combined with a CS696W module to satisfy a *Writing Intensive* requirement.

• This course can be taken as CS735H by honors students.

3 Outcomes

• **methodologies of software development:** multi-threaded programming, client-server paradigm, performance issues.

• **fundamental computer science algorithms:** concurrent algorithms, synchronization, non-blocking algorithms.

• **concepts of programming languages:** concurrency support at the language and library levels, memory models and non-sequential consistency, serialization and remote method invocation.

4 Topics

• **Basic concepts of concurrent and distributed programming:**
  – parallelism, threads, scheduling, nondeterminism
  – blocking operations, deadlocks, livelocks, timeouts
  – atomicity, synchronization, data races
  – memory models, memory barriers, visibility
  – serialization, remote procedure call

• **Client-server paradigm:**
  – sockets, socket servers
  – Remote Method Invocation

• **Synchronizers:**
  – locks, semaphores, latches, barriers, conditions, futures
  – blocking and bounded queues

• **Concurrent data structures:**
  – *synchronized vs concurrent*, amount of parallelism
– atomic operations, client-side locking
– non-blocking algorithms

• **Executor services:**
  – tasks, futures, thread pools, failures, cancellations, shutdown
  – parallelizing recursive computations
  – timers and scheduled executors

• **Engineering concurrent programs:**
  – correctness, thread-safety
  – state, invariants, sharing, locality, mutability, immutability,
  – decomposition, tasks, execution services
  – thread-safe and concurrent data structures
  – interrupts, poison pills, cancellation, termination, abortion, non-interruptible blocking

• **Performance:**
  – Amdahl’s law
  – contention, thread creation, parking/unparking, I/O
  – throughput, responsiveness
  – resource management

• **Programming language support (Java):**
  – `synchronized` blocks, `final` fields, `volatile` fields
  – `wait`, `notify`, `notifyAll`
  – `java.lang.ThreadLocal`
  – `java.util.concurrent` (usage and implementation)
  – `java.net`, `java.io.Serializable`, `java.rmi`

• **Advanced topics:**
  – `compare-and-set`, non-blocking algorithms
  – Java Memory Model
  – `java.util.concurrent.lock`
  – `java.util.concurrent.atomic`
  – building custom synchronizers

5 Evaluation

Five programming assignments (50%), one project (30%) and one exam (20%).

6 Textbooks

Required:


Additional (for students new to Java):


Reference: