Final Exam Topics and Structure

1 Final Exam Content

The final exam concentrates on conceptual material rather than the details of programming. It might contain questions about material from any chapter of the lecture notes, from Chapter 1 through 10 excluding Chapter 9. There will be nothing about OpenMP. In particular, questions might be drawn from any of these areas:

- Introductory topics such as
  - data dependency graphs,
  - types of parallelism (data, functional)
  - ways of introducing parallelism to programming languages,
  - terminology of parallel computing in general
- Parallel architectures, including
  - network topologies and interconnection networks and their properties
  - parallel computer taxonomy
  - types of parallel computers and multiprocessors
- Parallel algorithm design, including the task/channel model (Foster's design methodology) and its various stages
- Message-passing programming concepts including concepts related to MPI, such as
  - the message passing model,
  - collective and point to point communication
    - types, such as gathering, scattering, send, receive, broadcast
    - understanding the semantics of both including buffering, asynchronous behavior
    - communicator objects
  - reduction
  - barrier synchronization
  - process ranks
- Floyd-Warshall algorithm - know the basic algorithm
- Performance analysis
  - speed-up
  - efficiency
  - overhead
  - total work
  - communication cost, including latency, transfer time, bandwidth
  - Amdahl's law
  - Gustafson-Barsis speed-up
  - Karp-Flatt metric

  - Scalability
  - Isoefficiency function and relation
  - Application of all of the above to parallel programs

- Domain decomposition methods of 2D matrices:
  - row-wise
  - column-wise
  - 2d block

- Monte Carlo methods
  - Independently and identically distributed random variables
  - Uniform distributions - discrete and continuous
  - Monte Carlo approximation and integration - definitions
  - Predicting error - what the theory tells us about sample size and error
  - Pseudo-random number generators
    * types: linear congruential, lagged Fibonacci
    * methods of extending to parallel generators: leapfrogging, sequence splitting, independent sequencing
  - Generating non-uniform variates
    * inversion method
    * acceptance rejection
  - Random walks
  - Markov chain properties - time and space homogeneity
  - Metropolis algorithm
  - Simulated annealing

- Shared memory multiprocessing
  - Threads
    * conceptually
    * representation in a computer system
  - Sharing memory: communication and synchronization using shared variables and the consequences:
    * critical sections
    * mutual exclusion
    * deadlock
    * starvation and fairness
  - fork/join parallelism
  - POSIX Threads model (Pthreads)
    * types of operations supported
    * thread management, such as creation, termination, controlling properties
    * mutexes
    * condition variables
    * barrier synchronization
  - Readers/Writers problem
  - Producer/Consumer problem

2 Final Exam Structure and Form

The final will have several types of questions, such as true/false, multiple-choice, multiple-answer, fill-in-the-blanks. There will be

- questions that require you to know definitions and terminology, such as
  - Is the following graph a hypercube (or a mesh or a butterfly etc)
  - What is the difference between a NUMA computer and a multi-computer?
  - Is a given statement a correct definition of a channel in the task/channel model?
  - Is a given network topology a direct or indirect network?
- questions in which you will need to apply formulas and arrive at numerical or algebraic answers, such as
  - What is the diameter of a given network?
  - What is the bisection width of a given network?
  - What is the maximum speed-up possible with n processors of a particular sequential program?
  - What is the scaled speed-up (Gustafson-Barsis) of a given parallel algorithm?
  - What is the communication overhead of a parallel algorithm,given enough information about it?
  - How about the scalability using the isoefficiency relation?
- questions that expect you to analyze a section of code and state the output or the running time, or some other performance metric,
  - Does a given segment of code or algorithm distribute portions of an array to a set of processes in a load-balanced way?
  - Given a small section of code, does it lead to deadlock or not.
  - Given a small section of code or an algorithm, what is its running time using order notation.
  - Given a small section of code, does it provide mutual exclusion?