Final Exam Review CpSc 418 (2012)

1 What kinds of questions could be on the final?

New material since the midterm:

Algorithms:

reduce and scan (we had covered reduce before the midterm, but scan came afterwards).

Sorting networks:

- Given a diagram for a sorting network, show what output it produces for a specified input.
- Given an algorithm in pseudo-code, determine whether or not it corresponds to a sorting network. Justify your answer.

The zero-one principle.

- Be able to state the zero-one principle.
- Given a simple sorting network, use the zero-one principle to show that it sorts correctly.
- Given an incorrect sorting network, construct an input consisting only of zeros and ones that it sorts incorrectly.

Bitonic sorting

- Compare with traditional merge sort.
- Be able to state a big-O formula for how many comparisons are performed by bitonic sort. Be able to give an informal justification of this big-O result.
- Be able to state a big-O formula for the communication cost of bitonic sort. If a bitonic sort is performed with P processors working on blocks of M words, how many messages must be sent? How many words must be sent?

Other sorting networks:

- Bubble sort as a sorting network.

Dynamic programming:

- describe where the parallelism is available in the algorithm
- describe the issues involved in exploiting this parallelism
- formulate a big-O formula for the computation time and communication time when run on a two strings of length N using P processors.

Other algorithms

- I could sketch a parallel implementation of some other algorithm and ask you to figure out the big-O analysis for computation time and communication time.
- The book has described LU decomposition and simultaneous over relaxation (SOR). Those algorithms or similar ones could be reasonable for a question.

Performance: Given an algorithm sketch, identify the key causes of overhead. Be able to justify your answers. Examples from the second half of the term include:

- Determing upper and lower bounds for the time to send batches of messages depending on network topology (see HW3, Q3).
- The analysis of the MPI implementation of dynamic programming (Nov. 1 & 6 lectures).

Correctness

- Safety Properties:
  - Freedom from races and deadlock, termination detection.
  - Able to read a short proof using invariants to prove the correctness of a mutual-exclusion algorithm or a similar algorithm.
  - Can modify an existing proof, or complete a few missing lines in a proof of an invariant.
- Liveness Properties: Can define livelock and starvation and give examples.

Architecture:

- We covered GPUs in second half of the term. Be able to describe key features of a GPU:
  - How does the GPU organize a large number of execution units?
  - Compare and contrast a GPU with a traditional CPU?
  - How does the GPU architecture make software for GPUs different than software for CPUs?
  - What is a data-parallel programming model?

Programming:

- Erlang and MPI are both based on message passing. Compare and contrast the models for messages in these two:
  - broadcasting messages
  - buffering messages
  - matching senders to receivers
  - selecting a message when many have been received.
  - . . .
- pthreads: able to describe basic concepts and constructs of pthreads
  - mutexes, condition variables, and barriers.
  - how does a pthreads make it easier for a programmer to handle weak consistency.
  - able to answer questions about the producer-consumer example from class (there's also an implementation of producer-consumer in the text).

My previous summary of possible midterm questions:

Performance:

- Speed-up: what does it mean that a parallel program is 25 times faster than a sequential program?
- throughput and latency. Be able to explain what they are, how they are related, and how they are different.
- Understand the various kinds of overhead that can prevent parallel programs from realizing ideas speed-up. Be able to give examples of each kind.
- Dependences: Be able to define and give examples of a flow-dependence, an anti-dependence, and an output dependence. (Lin & Snyder, Chapter 3, Parallel Structure → Dependences . . . How Dependences Limit Parallelism, (p. 68-71)).
- Amdahl's law: what is it? Be able to give and use the formula. Also, be able to express the intuition behind it. Be able to give examples of how it applies and of situations where it does not apply.
- Super-linear speed-up: What is super-linear speed-up? Give some examples of things that can cause it?
- What is an "embarrassingly parallel application"?
- From the homework: what have you learned about measuring the execution time of a program? Is it easy to get one, clear number? If not, why not?

Architecture

- Shared-memory multiprocessors
  - What is a snooping cache? The MESI protocol.
  - What's directory-based coherence?
  - What's is sequential consistency?
  - Do most chip multiprocessors provide shared memory?
  - Do most chip multiprocessors guarantee sequential consistency?
- Message passing computers
  - Describe some of the key design-issues for the communications network: latency, bandwidth, robustness.
  - What are some typical network topologies? Name one advantage and one disadvantage for each.
  - Describe some of a small- or medium-sized compute cluster.
  - Describe some of the key features of a large supercomputer.
- Superscalar computers
  - Register renaming. The life-cycle of a register.
  - How renaming handles "false dependencies" (see Lin & Snyder, Chapter 3, Parallel Structure, for a more general discussion of dependences).
  - Branch prediction: what it is. How mispredicted branches are handled.
  - Precise exceptions: what they are. How a superscalar provides precise exceptions.
  - Scaling issues: why not make a super-scalar that issues 200 instructions/cycle?

Programming

- Be able to read a short (10-20) lines of Erlang code and explain what it does (this will **not** use past winners from some "obsfucated Erlang" contest).
- Be able to write up 3-10 lines of Erlang to complete a function.
- Given a sketch for an algorithm, comment on its communication requirements and other efficiency and performance issues.
- Looking over the lectures, readings, and homework, I could include something from the examination of various ways to parallelize matrix multiplication.