Course Title: Parallel and Distributed Computing

Prerequisite: CSc 2311 Principles of Computer Programming II (C++ Programming)

Class Time and Place: TR 10:00-11:40 a.m.; Adelphi Learning Center 224

Instructor: Sushil K. Prasad; Room 1453, 34 Peachtree; Email matskp@panther.gsu.edu (do not leave phone messages; do not use sprasad@cs.gsu.edu).

Office Hours: TR 9:00-9:50 a.m. Meetings at other hours by appointment. Office hours may be cancelled occasionally due to academic meetings.

Content: This course is intended to cover various aspects of parallel and distributed processing and algorithm design, with an emphasis on hands-on programming. Topics may include: Taxonomy of parallel computers; Shared-memory vs. message-passing architectures; Computation models and Performance metrics; Parallel/distributed algorithm design – basic techniques; Parallel/distributed programming techniques and issues: partitioning, load balancing, synchronization, task scheduling, message overheads, etc.; Parallel/distributed algorithms for sorting, matrices, etc.; Debugging, Profiling, and Performance enhancements of parallel and distributed programs; Some material will be covered by reading assignments.

Students will gain experience in parallel and distributed programming on state-of-the-art 24-processor Silicon Graphics Origin-2000 high performance hypercube-interconnect computer, and possibly on a network of Unix/Linux workstations/cluster and multiprocessors running Message Passing Interface (MPI) and/or Parallel Virtual Machine (PVM) packages.


Attendance: You may be dropped if you have more than two unauthorized absences. Students are responsible for all the material covered or assigned (whether or not in the text). Nov 23-27 are Thanksgiving holidays. Last class is on Dec 9 (Thu).

Withdrawals: The last date to withdraw is Oct 15.

Grading:

<table>
<thead>
<tr>
<th>Assignment</th>
<th>CSc 4310</th>
<th>CSc 6310</th>
<th>Dates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming and Other Home Assignments</td>
<td>35%</td>
<td>35%</td>
<td></td>
</tr>
<tr>
<td>Test 1</td>
<td>20%</td>
<td>15%</td>
<td>Sep 21</td>
</tr>
<tr>
<td>Test 2</td>
<td>20%</td>
<td>20%</td>
<td>Oct 21</td>
</tr>
<tr>
<td>Test 3</td>
<td>25%</td>
<td>20%</td>
<td>Dec 9</td>
</tr>
<tr>
<td>Term Project</td>
<td>bonus 5%</td>
<td>10%</td>
<td>Due Dec 2</td>
</tr>
</tbody>
</table>

Final grades will be relative to the class performance (to be calculated separately for CSc 4310 and CSc 6310). There will be no make up test given except for medical emergencies supported by proper documentations.

Academic Honesty: Work submitted for grading must be student’s own. Plagiarism, in my professional opinion, may result in grade ‘F’ for all involved, and may also risk further action. Refer to the university catalogs for policy on plagiarism.

Disclaimer: The course syllabus provides a general plan for the course; deviations may be necessary.

A teacher can never truly teach unless he is still learning himself. A lamp can never light another lamp unless it continues to burn its own flame. The teacher who has come to the end of his subject, who has no living traffic with his knowledge but merely repeats his lesson to his students, can only load their minds, he cannot quicken them.

Rabindranath Tagore, Indian Poet
Nobel Laureate in Literature, 1913
2 Term Project

Term Project Topics
August 23, 2004

Choose a problem from the topics below, and two non-trivial algorithms for this problem. For each algorithm, write (i) a shared-memory program, and (ii) a local memory program to implement them. You need to get your algorithms approved before working on it.

Submit the following:

1. A 3-5 page single-spaced project report (hard copy) containing the following sections:
   (a) your problem,
   (b) chosen algorithms,
   (c) associated data structures,
   (d) underlying communication pattern for distributed program,
   (e) amount of read and write contention and synchronization overheads for the shared memory program,
   (f) parallel time complexity for each implementation with a break down of communication/synchronization time and of computation time,
   (g) timing experiment details (description of what parameters have been varied in what range, and how many runs have been used for performance data),
   (h) performance of each program with reference to the plots for (a) execution time and for (b) speedup as number of processes varies, and
   (i) your conclusions containing your interpretation of the performance of these programs, their limitations, and possible future improvements.

Students may also be asked to present their problem description, algorithm(s) employed, program description, and performance plots to the class using 5-10 minutes.

Due Dates:
Initial half-page writeup containing problem statement, and a brief description of your chosen algorithms: Oct 4.
1st Algorithm Implemented and writeup: Nov 4.
2nd Algorithm Implemented and Writeup (including a comparison section with the first algorithm): Dec 2.
Presentation: Last couple of classes.

Suggested Problems and associated algorithms:

2. Matrix Multiplication: Block oriented algorithm on mesh topology, row-column oriented on ring/hypercube topology (Quinn’s).
3. Fast Fourier Transform: hypercube topology
5. Shortest Paths in a Graph: Moore’s algorithm
6. Traveling Sales Person: Mohan’s Algorithm, Quinn’s Algorithm on multicomputers
8. Concurrent Priority Queues:
9. Graph Coloring
Timing Experiments on Parallel Priority Queues - Sample for other projects as well

The following timing measurement should be carried out on the parallel priority queue projects.

No of processes \( p = 1 \) to 6 on Hydra, 1 to 6 on MPI/PVM

Size of the priority queue \( n \) from \( 2^{10} \) to \( 2^{20} \) in steps of even powers of 2.

Maximum Think Time, \( k = 0, 10, 100, 1000, 2000, 4000 \) (on SGI)

Use approx. hold model: Each delete produces zero, one, or two items with equal probability. Try to combine the insert followed by a delete operation, if possible on the data structure.

Sequentially, perform \( 2^{10} \) cycles of insert-delete-think by deleting one item, thinking, and inserting the new items repeatedly. This time, \( T_1 \), should be using the best sequentially optimized code.

Main() will initialize a priority queue of size \( n \), whose time is not counted.

In parallel, using \( p \) processes, each process performs \( 2^{10}/p \) cycles (thus performing equivalent amount of work).

For parallel heap and calendar queues, perform only \( 2^{10}/(p * r) \) cycles, because each cycle will delete \( r \) items.

Use exponentially distributed priority increments as given in the SPDP95 paper by Prasad and Sawant. For calendar Queue, use real values for priorities, for others, use integral priorities. Calendar Queue performs poorly on integral priorities.

Plots to be submitted:

1. \( T_p \) versus \( p \) (\( n = 2^{10}, 2^{15}, \text{ and } 2^{20}, \text{ } k = 1, 1000, 4000 \)). So, you will have a total of nine curves on the same plot for all combinations of \( n \) and \( k \).
2. \( S_p \) v/s \( p \) (\( n = 2^{10}, 2^{15}, \text{ and } 2^{20}, \text{ } k = 1, 1000, 4000; \text{ total of nine curves } \))
3. \( S_6 \) v/s \( k \): \( S_6 \) is the best speedup obtained using 6 processes on SGI. Use \( S_6 \) for MPI/PVM. (\( n = 2^{10}, 2^{15}, \text{ and } 2^{20} \)).
4. \( S_6 \) v/s \( n \): (\( k = 1, 1000, 4000 \)).

Other timings should be submitted in a tabular fashion.

Some data structure specific information are as follows.

**Concurrent Heap:** Rao and Kumar, 1988. Compare two versions: (i) locks per level (ii) one general processor performing all the delete-think-insert cycles, others dedicated to one or more levels for processing the delete and insert update processes as they sink down from root. Plots would, therefore, have both kinds of implementations for direct comparison.

**Concurrent Skewed Heap:** Have a lock per node. This is the only version needed. Since SGI has only up to 4000 locks, allocate all the locks in the main() in an array LOCK[]. Also, have a variable next_lock set to 0. Assign a lock to each new node created by setting its lock index to next_lock, and atomically incrementing next_lock. When the limit is reached, reset next_lock to zero.

**Distributed Calendar Queues:** Have a calendar queue per process. Input \( r \) and the bucket interval \( \Delta r \).

In each cycle, find how many buckets \( j \) starting from the current bucket are eligible to be deleted from. Idea is to delete from \( j \) earliest buckets such that total number of items deleted would be roughly \( r \) top priority items over the entire set of calendar queues. More to be discussed.

In each cycle, perform up to \( r \) iterations of delete-think-insert. Then, send the insert items destined for other processes (one message per processes), and receive the same from others (empty message o.k.).

Vary \( \Delta r \) as appropriate set of intervals sizes (smaller \( \Delta r \) will yield more precision) such that the average number of items per bucket in roughly \( x\% \) of \( r \), for \( x \) varying from 1\% to 8\% in steps of powers of two.
**Concurrent Splay Tree:** Use bottom-up splaying. First implement concurrent insertion and min-deletes on a binary search tree without splaying. Introduce splaying later. Will need parent pointer. Will need four locks for either zig-zag or zig-zig rotations (node, parent, grandparent, and grand-grandparent).

For searching for insert, need a single lock on the current node. For searching for min-delete, need two locks: one on the current node, and one its parent.

### 3 Books and Other Pointers

**Some Reference Books for Parallel Computing**

12. Lewis, El Rewini, *Intro to Parallel Computing*, (Prentice Hall),
13. Moldovan, *Parallel Processing - From Applications to Systems*, (Morgan KAUFMAN)

**Some Conferences and Journals related to Parallel Computing**

**Journals:**

1. IEEE Transactions on Parallel and Distributed Systems
2. Journal of Parallel and Distributed Computing
3. Parallel Computing
4. Parallel Algorithms and Applications
5. International Journal of Parallel Programming
6. Parallel Processing Letters
Annual Conferences:
1. IEEE International Parallel and Distributed Processing Symposium (IPDPS)
2. International Conference on Parallel Processing (ICPP)
3. International Conference on High Performance Computing (HiPC)
4. ACM Symposium on Parallel Algorithms and Architectures (SPPA)
5. Supercomputing
6. Symposium on the Frontiers of Massively Parallel Computing

High Performance Wire Service: Send email to trial@hpcwire.tgc.com for a FREE 6-week subscription to HPCWIRE news service on high performance computing happenings.

Some web sites of interest

MPI Standard: http://www-unix.mcs.anl.gov/mpi/
PVM web home page:
   http://www.epm.ornl.gov/pvm/pvm_home.html

Online computer literature:
   www.researchindex.com
   ACM’s Digital Library - free access through GSU library
   IEEE’s Digital Library (bibliography only) - free access through GSU library
4 Sample Test 3

CSc 4310/6310: Introduction to Parallel and Distributed Computing
Fall 2003
Test 3

Notes:

1. Time = 100 mins. Attempt all questions. Each question is worth 10 points, except the first is worth 2 points each part.

2. When presenting parallel programs, begin with a description of the basic strategy in a couple of sentences. You may ignore the header files and the codes for input/output. Concentrate on the logic for partitioning, scheduling, messaging, and synchronization, and the main logic for accomplishing the task. Use the correct library calls (without worrying much about the

3. Present all algorithms at an abstract level (i.e. do not give pseudo codes). State an algorithm step-wise, and focus on the main logic for each step. You might want to use examples to illustrate your methods.

4. Be neat and precise. Partial marks will be given. Closed books and notes.

5. To obtain your grades, send an email to that effect. You may also collect your projects and other remaining assignments and obtain a copy of your final exam answer sheet early next semester.

1. Answer true or false for each of the following. Give a sentence-long justification.

   (a) A binary tree has a dilation-1 embedding on a mesh.

   (b) A butterfly network pattern cannot be employed to implement a barrier.

   (c) Mapping level-0 hexes to hypercube nodes in Battlefield Management Simulation ensured a dilation-1 embedding.

   (d) As evidenced by almost linear speedup on Ncube-II for fairly large number of processors, even though the combat zones vary dynamically and are located in narrow and scattered regions in Battlefield Management Simulation, the computation involved gets adequately distributed among several processors by using a static wrapped around mapping of the level-0 hex \((i, j)\) to the grid location \((i \text{ mod } r, j \text{ mod } c)\) of a \(r\)-by-\(c\) grid, and then using reflected binary Gray encoding to map the 2-D grid onto a hypercube.

   (e) Brent’s scheduling principle may be applied to reduce the number of processors to obtain cost-optimality only when the total work performed by the algorithm does not exceed \(O(T^1_f)\).

   (f) Not all of the following algorithms can be implemented on a \(p\)-processor hypercube in \(O(\log p)\) time: One-to-all broadcast, all-to-one reduction, prefix sum on \(p\) items.

   (g) Odd-even transposition sort of \(n\) items can be implemented on a \(n\)-processor linear array (1-D mesh) in \(O(\sqrt{n})\) time.

   (h) A pthread signal is lost if not caught by an already waiting pthread process.

2. Work out the bitonic sort algorithm on the following sequence of alphabets, assuming an ASCII order: 
   E P I Q A L C Z

3. Write the shared-memory main() and the worker() functions for multiplying a \(n\)-by-\(n\) matrix \(A\) with a \(n\)-by-1 vector \(B\) to obtain \(C\) on SGI \(p\)-processor multiprocessor. Focus on how each process finds its work given an arbitrary \(p\) number of processors.

4. Write a PVM/mpi program using \(p\) processes to find the sum of squares of \(n\) numbers. Focus on sending out proper portions to the slaves, collecting the results back, and calculating the final sum.

5. Carry out either parallel sorting by regular sampling algorithm or hyper-quick sort algorithm using 4 processors on the following 20 integers. Show the output of each major step, and the source and destination of the partitions as the algorithm progresses.

7 8 12 6 9 2 25 9 1 30 3 57 5 6 83 9 27 4 72 5
5 Student Information

CSc 4310/6310 Student Information

Name:

Telephone numbers where you can be reached:

email:

Optional – Company working for (Address):

Major and Minor

Class (circle one): Fr So Jr Sr M.S. Ph.D.

Prerequisite CSc 2311 C++ course completed:

Semester Instructor Grade

Where and When other Computer courses completed:

Your expectations of this course:

Any computation-bound problems you think you have which might benefit from parallel/distributed programming: 