Course Title: Parallel and Distributed Computing

Prerequisite: CSc 3210 Computer Organization and Programming and CSc 3320 System Level Programming (C/Unix programming) - C/C++ programming and Discrete math should be brushed up!

Class Time and Place: TT 10:00-11:45; Aderhold Learning Center 31

Instructor: Sushil K. Prasad; Room 717, 25 Park Place Email sprasad@gsu.edu.

Office Hours: TT 9:00-9:50 am; 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 architectures; 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 (i) Just upgraded to 20-node Linux cluster with heterogeneous multicore and GPGPUs and infiniband interconnect and/or (ii) a 24-processor Origin-2000 hypercube-interconnect computer.


Grading:

<table>
<thead>
<tr>
<th>Section</th>
<th>CSC 4310</th>
<th>CSC 6310</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attendance and Class Participation</td>
<td>5%</td>
<td>5%</td>
<td></td>
</tr>
<tr>
<td>Programming, Home Assignments and Quizzes</td>
<td>40%</td>
<td>30%</td>
<td></td>
</tr>
<tr>
<td>Test 1</td>
<td>15%</td>
<td>15%</td>
<td>Sep 23</td>
</tr>
<tr>
<td>Test 2</td>
<td>20%</td>
<td>20%</td>
<td>Oct 23</td>
</tr>
<tr>
<td>Test 3</td>
<td>20%</td>
<td>20%</td>
<td>Dec 4</td>
</tr>
<tr>
<td>Term Project</td>
<td>5% (optional)</td>
<td>10%</td>
<td>Due Dec 2</td>
</tr>
</tbody>
</table>

CSc 6310 students will have additional requirements in home works, test questions and/or projects. 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. All assignments and projects must be completed to pass the course. Assignments are due before the class; late submissions do not earn any points.

Academic Honesty: Work submitted for grading must be student’s own. Collaborative assignments will be specified by the instructor. D2L employs sophisticated checks against Web resources. Plagiarism, in my professional opinion, may result in grade ‘F’ for all involved, and will also risk further action. Refer to the university catalogs for policy on plagiarism. THIS IS MOST SERIOUS!

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
Choose a problem from the topics below (alternative topics with possibility of innovation are encouraged), 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 (total of four programs). You need to get your algorithms approved before working on it.

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: Sept 30
- Ist Algorithm Implemented and writeup: Oct 28

Submit the following: A 3-5 page single-spaced project report (hard copy) containing the following sections:

1. your problem,
2. chosen algorithms,
3. associated data structures,
4. underlying communication pattern for distributed program,
5. amount of read and write contention and synchronization overheads for the shared memory program,
6. parallel time complexity for each implementation with a break down of communication/synchronization time and of computation time,
7. timing experiment details (description of what parameters have been varied in what range, and how many runs have been used for performance data),
8. performance of each program with reference to the plots for (a) execution time and for (b) speedup as number of processes varies, and
9. your conclusions containing your interpretation of the performance of these programs, their limitations, and possible future improvements.

- 2nd Algorithm Implemented and writeup (including a comparison section with the first algorithm): Dec 02

Writeup 3-5 page as above, and a comparison section.

- Presentation: Last couple of classes.

**Suggested problems and associated algorithms:**

2. Fast Fourier Transform: hypercube topology
4. Shortest Paths in a Graph
5. Traveling Sales Person
7. Concurrent Priority Queues:
8. Graph Coloring

**Research Oriented Topics:**

9. GIS computation on cloud/cluster/multi-cores
10. Parallel Priority Queue on GPUs/Multicores
11. Parallel/Distributed Quad or R-Trees for GIS computation
12. Parallel Parsing of GML (Geographic Markup Language) files
13. P2p search algorithms
14. Distributed Vertex cover algorithms
15. Distributed graph coloring algorithms
16. Distributed Scheduling Algorithmic on Sensor Network (simulation or implementation)
17. MPI to CUDA program porting/translation
18. Parallelizing a serial program involving iterative computation (iC2mpi platform).
19. Collaborative distributed applications over heterogeneous Devices (SyD platform).
20. Distributed workflows over web services for data and tool integration (BondFlow platform).
21. Database computation on GPUs.
22. Data aggregation on sensor networks
23. Efficient and relevant search on the web
24. Web service composition
25. Distributed workflows over web services
26. Power Efficient Algorithmic Techniques
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 8 on shared memory, 1 to 8 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 shared memory)
\( k \) = 2000, 4000, 8000, 16000, and 32000.

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 \( 2^{20} \) 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 \( 2^{20}/p \) cycles (thus performing equivalent amount of work).

For parallel heap and calendar queues, perform only \( 2^{20}/(p \times r) \) cycles, because each cycle will delete \( r \) items. In these two data structures, vary \( r \) as 100, 1000, and 10000. The following plots would also need suitable \( r \) values.

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 \( p \) (\( n = 2^{10}, 2^{15}, \) and \( 2^{20}; k=1, 1000, 4000 \)). 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 \( p \) (\( n = 2^{10}, 2^{15}, \) and \( 2^{20}; k=1, 1000, 4000; \) total of nine curves)
3. \( S_8 \) v/s \( k \): \( S_8 \) is the best speedup obtained using 8 processes on SGI. Use \( S_8 \) for MPI/PVM. (\( n = 2^{10}, 2^{15}, \) and \( 2^{20} \)).
4. \( S_8 \) 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 on its parent.
3 Books and Other Pointers

7. Ian Foster, Designing and Building Parallel Programs, Addison-Wesley Pub. (Available online on Web at www.mcs.anl.gov/dbpp)
8. K. M. Chandy and Taylor, An Introduction to Parallel Programming, Jones and Bartlett Pub.
10. Lester, The Art of Parallel Programming, Prentice-Hall
11. Chandy and Misra, Parallel Program Design, (Addison Wesley),
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 High Performance Computing (HiPC)
3. International Conference on Parallel Processing (ICPP)
4. ACM Symposium on Parallel Algorithms and Architectures (SPPA)
5. Supercomputing (SC)
6. Symposium on the Frontiers of Massively Parallel Computing
7. Euro-Par
8. Intl Conf. on Distributed Computing Systems (ICDCS)
9. CC-GRID
TCPP  Sign up for Newsletter: tcpp.computer.org

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:
   Google Scholar
   ACM’s Digital Library - free access through GSU library
   IEEE’s Digital Library - free access through GSU library
   www.researchindex.com
1. (a) (6 points) Draw and label an 8-processor hypercube network. Argue that its diameter is $O(\log p)$ if $p$ is number of processors.
   Note: Graduate Students need to formally prove the latter.
(b) (8 points) Illustrate four steps of Foster’s Parallel Program Design technique using either (a) simulation of cooling of a rod with end points dipped in ice using 8 partitions of rod over 4 processors interconnected as a path, or (b) a reduction tree for adding 16 numbers over a hypercube computer with 4 processors.
   Note: Graduate students need to do both (a) and (b).
(c) (8 points) Give one way to implement producer-consumer shared-memory synchronization for prime-finding sieve method wherein $P_0$ produces the prime markers in the range $2, \sqrt{n}$ and all $p$ processes consume them to mark in their subranges. Give three ways to improve the performance of the basic sieve program.
(d) (6 points) Given $T^*_1$ of $O(n)$ and $T_p$ of $O(n/p + \log p)$, calculate the range of optimality for this algorithm.

2. Given $n$ positive integers in an array $A[0..n-1]$ in the shared memory.
Write a shared-memory program for finding the maximum value in $A$, by giving either pseudo-codes or C/C++ codes for the worker process for shared memory program. Assume $p \leq n/2$ processes and that the answer should be in a shared variable $max$. For this, exact syntax of parallel programming primitives is not needed, but the key parameters need to be identified.

(a) (5 points) Give the code for each process to find the start and end indices of the array $A$ to process individually. Ensure that the indices are correct even if $n$ is not divided evenly by number of processes $p$.
(b) (7 points) Give rest of the code.
Write a MPI program to solve Graph Clique problem using master-slave load balancing framework. Input is the number of nodes n, probability of an edge p, and clique size K. Your program will do exactly the following:

1. $P_0$ generates a random Boolean matrix of size nXn with probability p for each edge $\{i, j\}$ and send to all others.

2. $P_0$ will act as the work pool manager and whenever other processors finish, they send their result to $P_0$ and $P_0$ in turn send more work to them or tell them to terminate. The basic algorithm is to try all subsets of nodes and determine if any subset forms a clique of size k. For this, $P_0$ will use a counter from 0 to $2^n - 1$ where each value represents a subset.

3. For ensuring enough granularity of work, $P_0$ will send a range (start..end) to a slave, and slave will check for each corresponding set and return i as soon as it corresponds to clique of size k or more. Graduate students will also create a variation where maximum clique is found, and slaves check for number of nodes in a subset and proceed to verify clique only if it exceeds the current max clique size (i.e., not by simply running the previous program for k=n, k=n-1, etc.). They also experiment with variable range to ensure best speedup which balances communication overhead and load imbalance.

Email overall algorithm design including control loop algorithm/pseudo-code of master and slave ensuring termination (30%), source codes adequately documented and corresponding to your algorithm design/pseudocode (50%), sample runs (5%), and timing/speedup plots (15%). Graduate students submit 3 plots including $T_p$, $S_p$, and granularity. Try different number of nods and probabilities. Undergraduates can get 20% bonus for graduate students’ portion.
6 Sample Test 3

CSc 4310/6310: Introduction to Parallel and Distributed Computing
Fall, 2010
Test 3

Notes:

1. Time = 100 minutes. Undergraduates answer any four-fifth answer will carry bonus marks. Answer briefly and to the point.

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

3. When working out an algorithm, give snapshot of the relevant data after each major step.

4. 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 syntax) and insert adequate comments.

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

6. Good luck!

Questions

1. (a) (7 points) Write a cuda kernel Merge (A,B,C) which merges two input arrays A and B into C, assuming that each ith item in A and B have a cuda thread and BinarySearch(X,y) returns the rank of y in array X. If Merge() is to be called upon to perform Mergesort, what synchronization technique/primitive one can employ if cuda threads are (i) within one multiprocessor, (ii) multiple multiprocessor?

   (b) (3 points) What is the main advantages of this parallel mergesort algorithm compared to the usual merge sort algorithm based one one-processor merging. What is a disadvantage?

2. (a) (7 points) Write an MPI program segment to carry out scatter on a hypercube, assuming processor 0 has an array A[0..p-1].

   (b) (3 points) Find the computational and communication complexity of this algorithm.

3. (a) (7 points) Workout hyperquicksort on the following input on a hypercube of dimension 2: 12 19 4 23 8 6 38 10 8 9 16 36

   (b) (3 points) What are two advantages of PSRS algorithm over hyperquicksort?

4. (a) (7 points) State Cannons algorithm for matrix multiplication.

   (b) (3 points) Find the computational and communication complexity of this algorithm.

5. (a) (7 points) How would you implement insert and min-delete operations on a heap data structure on a shared memory machine such that up to O(log n) processors can concurrently access it?

   (b) (3 points) If each heap node can accommodate some r items, how does a min-delete operation ensure that it moves down only one branch of the Parallel heap?
7 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 3210 Computer Organization course completed:
   Semester   Instructor   Grade
Prerequisite CSC 3320 C and Unix Computer courses completed:
   Semester   Instructor   Grade
Descrete Math course completed:
   Semester   Instructor   Grade

Where and When other Computer courses completed:

Your expectations of this course:

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