CS 267 - Applications of Parallel Computers - Spring 1995
Class Project

This document contains suggestions for class projects. As with the homework, these are to be done in small interdisciplinary teams. You must submit a proposal to me by Thursday April 6, so we have time to talk about it and I can approve it. In a discussion section on Tuesday April 11, your team will present a short summary of your proposal, goals, plan, and schedule to the rest of the class. On Tuesday, May 9, we will have a poster session in which students will put slides describing their projects up on posters for a 2 hour period, during which the public will be invited to visit and ask questions. (In previous semester I had students give formal talks, but I think a poster session encourages more interaction, so we will try it this way this semester.)

This year, I also want you to submit a URL for your project. These will be put on the class home page for the edification of future generations of CS267 students as well as the whole WWW community.

Below is a short list of possible class projects. You are welcome to suggest your own, and to ask me or any other faculty member for ideas. At the end the projects from 1993 and 1994 are listed, so you can get some idea of previous activities. I have most, if not all, of the writeups for previous years' projects in my office, and you are welcome to look at them.

1 Extensions to ScaLAPACK.

We discussed ScaLAPACK in class. It is a library of routines for numerical linear algebra being constructed here and at the University of Tennessee. We released version 1.0 at the end of February; there have been 16,000 accesses to it on the Web, so it is widely distributed. There are a number of extensions and additions we would like to make. One or more of them could constitute an interesting project.

- *ScaLAPACK – Split-C interface.* We would like to be able to call ScaLAPACK from Split-C. First, this requires changing the internals of the underlying message passing system used by ScaLAPACK (the BLACS, or Basic Linear Algebra Communication Subroutines) to use Generic Active Messages, on which Split-C is based. Second, it required building an interface so that spread arrays can be passed to ScaLAPACK. Both use quite similar data layouts, so this should be doable efficiently.

- *A guaranteed accurate symmetric eigensolver for ScaLAPACK.* The current routine in ScaLAPACK for finding eigenvalues and eigenvectors of symmetric matrices makes certain compromises to guarantee speed, at the expense of occasionally getting inaccurate vectors (in which case a warning is given!). We have several plans to provide a routine with guaranteed accuracy. The easiest (and slowest) one is based on an algorithm called QR iteration. I expect this to be quite easy to build of existing routines in LAPACK in ScaLAPACK. You should also do performance modeling and testing to see how much slower the QR routine is than the existing ScaLAPACK routine, and how the accuracy compares.

- *Singular Value Decomposition for ScaLAPACK.* This problem is closely related to the symmetric eigenvalue problem, and with algorithms analogous both to the current ScaLAPACK eigensolver, and to the QR algorithm just proposed. The project would produce an SVD analogous to the current ScaLAPACK routine. I expect many existing ScaLAPACK routines could be used. Performance modeling would be done as well.

- *Other ScaLAPACK routines.* I can propose a large number of other similar projects as well.

2 Monte Carlo Simulation.

This problem is offered by Prof. Andrew Gelman of the Statistics Dept. (gelman@stat, 642-9959). Please see him for details.

Background. An important problem in statistical computation is Monte Carlo simulation, and a commonly-used method is Markov chain, or "random walk" simulation, in which a computer program simulates a random walk through a probability distribution. The idea is to run the random walk long enough so that it has time to take a tour through most of the distribution. A basic idea in implementation is to simulate several parallel random walks, so that you can (1) go through several parts of the distribution at once, and (2) compare the paths of the several random walks to assess convergence. When the random walks are close to convergence to their ergodic distribution, their paths will have similar statistical properties.

Our current set-up. We currently have a program written in "serial/parallel": it starts m random walks, then cycles through them, performing one step in each random walk, until each walk has gone through n steps. Typical values are m=10, n=10000. In some implementations, the program pauses occasionally (perhaps every 1000 steps) to (a) alter the random walk algorithm, and (b) assess convergence. (Output from the random walks up to time t can be used to alter the random walk algorithm and improve the efficiency of the simulations.)

Our goal. We want a parallel set-up in which each of m processors is continually running a random walk. The parallel processors can be in a single machine or, perhaps more usefully, on a network of workstations. The random walks should be run asynchronously, if possible (so that if one of the processors is busy, the others don't have to wait for it). There should also be a processor used as a controller. The controller gives the m processors starting points for their walks; the controller also supplies them with some parameters that govern the random walk algorithms (an example of a parameter is delta, the length of a jump in each step of the walk). The m processors are writing their results into some shared memory, and occasionally the controller reads the memory and uses the results to (a) decide how to alter the random walk algorithms, and (b) assess convergence (i.e., decide whether to stop). The results from (a) are fed back into the random walks, and so forth.

The technical difficulties seem to be: (1) setting up the asynchronous processes with the shared memory, and (2) having the controller be able to read this information without interfering with the processors that are writing it. I'll supply a serial program with all the algorithms so that the issue for you is to parallelize it.

I think that parallel processing for this family of algorithms will be useful to a lot of applied statisticians. Of course, I expect this will result in a publication in a statistics journal (I don't know what gets published in C.S. journals) and will lead to many other applications.

3 Using and extending Multipol.

These problems are offered by Professor Kathy Yelick, 2-8900, yelick@cs.berkeley.edu.

- You can try to improve the "pruning" performance of your TSP implementation by using another distributed data structure from Multipol, like a hash table or trie, to keep track of paths visited so far, to better prune paths which contain those paths.

- You can do a logic level device simulator (logic plus gate delays) using the event graph in Multipol.

- Multipol contains an octtree data structure, which is used in the Fast Multipole Method (FMM) or Barnes-Hut solution of the N-body problem. An alternative octtree, which caches remote references, is used by John Salmon for his N-body problem. Create a similar Multipol oct-tree, and compare the performance on the rudimentary N-body solver of Arvind Krishnamurthy.

- Implement a sophisticated N-body solver using an octtree. Use it for Sharks and Fish 1+2 or 6, or "bouncing balls with gravity."

- Implement an existing PIC ("particle in cell") code for semiconductor device simulations in Split-C and Multipol.

- Add a sparse matrix class to Multipol, including the basic operations (BLAS and level-0 preconditioning) mentioned during the talk by Roldan Pozo.

- Add an unstructured graph data structure to Multipol, including operators to partition it, and iterators. Use to to parallelize a finite difference application on an irregular grid (EM3D). Also see if graph partitioning can be used to reorganize a locally stored part of a graph to better exploit local memory hierarchy.

4 Adaptive Poisson Solver.

This problem is offered by Professor Phil Colella of Mechanical Engineering, colella@ce.berkeley.edu. It is a serial implementation of an adaptive mesh refinement scheme for solving Poisson's equation on irregular domains with sharply varying coefficients in the equation, leading to steep gradients in the solution. (Andrey Zege, zege@rose.lbl.gov, has signed up for this.)

5 Sparse Cholesky.

Sparse Cholesky is an algorithm for solving a linear system of equations $Ax = b$ where A is sparse and symmetric positive definite. It is basically LU factorization, where positive definiteness means that $U = L^T$, and any pivot order can be used. There are a number of complicated data structures involved to exploit the sparsity and ensure that a minimum amount of storage and minimal floating point operations are performed. There are a number of ways to layout data, and detect and schedule parallelism. Only recently have reasonable speedups been attained for this problem. We have a pretty good parallel implementation for which we have a number of optimizations that could be implemented and measured. I expect this could be a very good implementation if the optimizations are successful.

6 Parallelizing a Global Climate Model.

GATOR stands for Gas-Aerosol-Transport-Radiation-Chemistry Model, and is a model of the earth's atmospheric chemistry. It is part of a larger Earth Systems Model of the earth's climate, including the ocean (http://http.cs.berkeley.edu/ ssmith/GATOR.html). The ultimate goal is an integrated system including

- satellties collecting large quantities of sensor data (up to 1 Tbyte/day),
- numerical models using this data to predict the climate up to 50 years in the future,
- visualization systems to permit scientists to explore this data, and
- a data base to store all the satellite and simulation data.

The data base system is being built by the Sequoia project in the CS Division. There are a number of issues involved in building this system:

- The computational bottleneck in GATOR is solving the ODEs of chemical kinetics, for about 90 chemical species participating in about 200 reactions, at every grid cell into which the atmosphere is divided. The current code was written and optimized for a Cray C90, which has a much higher memory bandwidth than the cache-based microprocessors of which most parallel computers are built. As a result, the floating point performance is very poor. It may be possible to significantly increase performance by using techniques like prefetching, which we used on the IBM RS6000/590 to accelerate linear algebra.
- GATOR needs to exchange data with several other components: atmospheric dynamics, ocean dynamics, the data base, and the visualization system. All these systems operate on different grids of data, on different machines, possibly in geographically distant locations, connected over high speed networks. In order to design this complicated system, we need detailed performance models of the components, and the networks and format converters connected them, to identify bottleneck and fix them before they are built into the architecture.

(Oleg Zakharov, oleg@savanna.berkeley.edu, and Balazs Kralik, kralik@tiger.berkeley.edu, have signed up for this.)

7 Adapting Rigorous Electromagnetic Scattering Algorithms to the IBM SP-2.

This project is offered by Professor Andy Neureuther, neureuth@eecs, 2-4590, 510 Cory. Please see him for further details.

The projects are of interest to Prof. Neureuther for application in the IC and disk head industry and depending on its success there is a possibility it could evolve into a summer job and/or support during the academic year.

The projects are based on extending the PhD work of 1) Alfred Wong on finite-difference time-domain methods in a program TEMPEST developed on CM-2 and CM-5 architectures and ported to a CRAY-YMP and 2) Michael Yeung (finishing PhD in May 1995) on Fast Multipole Methods running on an IBM R/S 6000. Publications on these projects available from professor Neureuther on request. (Thesis of A. Wong, TEMPEST userguide, Paper by Michael Yeung, paper by Bob Socha.)

Porting to the IBM SP2

TEMPEST

The IBM SP2 architecture offers both the horsepower for CPU intensive tasks and the convenience of a workstation like environment. It should be straightforward to adapt any of the SAMPLE simulators to a given processor. The challenge will be in exploring ways to utilize multiple processors efficiently. For TEMPEST which is a massively parallel code the communication in sharing results at each time step between processors will be an issue. Improvements may be possible by utilizing the Collective Communications Library (CCL) to group information passing, and by using planes of interface nodes. Also overlapping these planes in buffer interface regions might be advantageous in reducing the frequency of communication calls.

Work on the basic algorithms in TEMPEST is also needed. Several potential improvements in the absorbing boundary conditions have been formulated and are available for testing and tuning to the IBM SP2 architecture in three-dimensions. Fundamental improvements in the discretization for curved material boundaries is also needed.

FAST MULTIPOLE METHODS

The IBM SP2 is attractive for simulation developed on workstations using both nonrigorous (physical optics) and rigorous (boundary integral) approaches based on a code which implements a new extension of the Fast Multipole Method to electrodynamics in 3D. An initial test showed that on the IBM SP2 the code runs 2x slower than on an IBM RISC/6000 model 540 where it takes 2 hours. The code is based on grouping panels which represent the surface in to sets and evaluating their multipole expansions which would nicely parallelize. The rigorous solution is carried out by GMRES methods and their may be existing work on such iterative methods on the IBM SP2 which could be utilized.

FUTURE POSSIBILITY OF WORK ON COMPUTATIONAL GEOMETRY

The SAMPLE group codes involve many computational geometry aspects which might also be of interest on parallel machines. Each surface of a material in the wafer topography is represented by about 5,000 triangles in a data structure with connectivity. These surfaces advance in time and auxiliary Oct Tree sort and is used to efficiently identify when a patch of one surface is about to crash into another surface or cross over itself. A zipper algorithm for finding surface intersections and clipping off or making a swallow tail of the unwanted region is also used. In taking the intersection many small triangles are created and a suite of triangle queries is used to identify thin/small triangles and regularize them without destroying intersections lines, surface ridges etc. The Oct Tree sort takes a few seconds on the RISC/6000 model 540 while the regularizations takes several minutes. Since this advance-intersect-regularize sequence must be performed 30 to 300 times in a single process simulation, their parallelization on the IBM SP2 is of interest.

Background on Rigorous EM Simulation

Topography scattering effects are world class electromagnetic scattering problems which require the very best in modern simulation approaches. Regions as large as 15 wavelengths on a side or over 3,000 wavelengths cubed in volume can contain materials with high and complex refractive indices and must be illuminated with projected images from optical systems with partially coherent illumination. Fortunately several approaches simulating reflective notching problems have been developed at Berkeley under support of the Semiconductor Research Corporation. One of these approaches is to use a finite-difference methods in the time-domain in a program called TEMPEST which has recently been extended to three-dimensions [1] and shown to be applicable to gate formation [2]. TEMPEST take several minutes of CPU time on a CM-5 machine. With assistance from Los Alamos National Laboratory TEMPEST has been shown to be portable to single instruction machines such as the CRAY-YMP. It would be very interesting to port TEMPEST to the emerging architectures such as that of the IBM-SP2.

A second approach is to utilize a breakthrough in the extension of the formulation of Fast Multipole Methods (FMM) to three-dimensional electromagnetic scattering [3]. This new formulation is being incorporated into both a rigorous boundary integral method with iterative solution and a much faster approximate method based on physical optics methods. The FMM takes about 2 hours on a stand alone IBM RISC/6000 model 540 and appears to run 2 times slower on the IBM SP2 architecture on campus.

All three methods are likely applicable to topography scattering effects and which is most effective will likely vary with the detail of the geometry, materials and scattering phenomena.

Background on the Physical Applications

Reflective notching caused by electromagnetic scattering from previously existing device features is a problem common to patterning the magnetic material of the pole tip for disk-heads and in forming MOS gate linewidths as they cross through the bowel shaped region of tightly layed out devices. Technology Computer-Aided-Design (TCAD) tools can be applied to these problems. This involves linking together the use of SAMPLE for aerial images, TEMPEST and Fast Multipole Methods (FMM) for electromagnetic scattering from nonplanar substrate features, and SAMPLE-3D for resist development effects.

These unanticipated contributions of exposure energy are an additional source of unwanted linewidth variation which is highly dependent on small changes in the topography and the alignment of the feature with respect to the topography. In some cases sever changes in linewidth occur and the problem is referred to as one of reflective notching. For example in making disk heads a fine pole tip must be patterned in a permalloy layer as close as possible to an insulated conducting coil which the permalloy layer must also cover. The sloping rise of the permalloy as it ascends the coil creates a nearly geometrical lateral reflection which results in loss of loss of pattern fidelity of the fine pole tip [4]. Similar nearly geometrical lateral scattering effects occur from a blanket polysilicon layer as it rises from the active MOS device area to coat underlying field oxide regions. Here elliptical bowl like structures are formed which can not only scatter but focus unwanted light into nominally unexposed regions [5]. More typically though a thinning of the linewidth is seen near the end of the gate polysilicon as it approaches and crosses up onto the field oxide.

As feature sizes are reduced compared to the height of topography features and the lithography must be performed with less process latitude to achieve nearly the fundamental theoretical limits, topography scattering must be considered as part of the overall design for manufacturability. The contribution of reflective notching to the linewidth variation budget must be understood and traded-off with other more traditional sources of variations such as inter-field tool variations, resist processing variations, etc. While the ultimate measure of these effects is experimental, process simulation offers the potential of systematic assessment and a physical mechanism based framework for characterizing contributions.

References

[1] A.K. Wong, and A.R. Neureuther, "Examination of Polarization Effects in Photolithographic Masks Using Three-Dimensional Simulation," Proc. SPIE Vol. 2197, pp. 521-528, 1994.

[2] R. Socha, A. Wong, M. Cagan, Z. Krivokapic, and A.R. Neureuther, "Effects of Wafer Topography on the Formation of Polysilicon Gates," Proc. SPIE Vol. 2440, to appear, 1995.

[3] M.S. Yeung and A.R. Neureuther, "Three-Dimensional Reflective-Notching Simulation using Multipole Accelerated Physical-Optics Approximation," Proc. SPIE Vol. 2440, to appear, 1995.

[4] J.S. Gau, "Photolithography for Integrated Thin-Film Read/Write Heads," Proc. SPIE VOl. 1088, pp. 504-514, 1989.

[5] M.P. Karnett and M.C. Sarnoff, "Optimizations and Characterizations of Single Layer Resist," Proc. SPIE Vol. 1088, pp. 324-338, 1989.

8 CS 267 Class Projects - 1994

9-9:25. "The How and Why of Using CMSSL with Split-C", Steve Steinberg

9:25-9:50. "LU Decomposition on Distributed Memory Multiprocessors Using Split-C", Eun-Jin Im, Bijal Modi, and Quyen Nguyen

9:50-10:15. "A Parallel Implementation of Cuppen's Divide and Conquer Algorithm for the Symmetric Tridiagonal Eigenvalue Problem", Allen Chen and Kim Man Liu

10:15-10:40. "Direct Methods for Solving the Static Analysis Problem in Civil Engineering", Frank McKenna, Kevin Gong, and Mark Cushey

10:40-10:55. Break

10:55-11:20. "Optimization and Parallelization of a Model of Carcinogenesis", Melody Ivory and Drew Roselli

11:20-11:45. "Parallelizing a Mixed-Volume Algorithm for Counting Roots of a Polynomial System of Equations", Michael Mitzenmacher and Ethan Bernstein

11:45-12:10. "Parallelization of SmartPath, An Automated Highway System Simulator", Farokh Eskafi and Siddartha Devadhar

12:10-1:10. Lunch

1:10-1:35. "Parallel Radiosity Computation in Computer Graphics", Laura Downs

1:35-2:00. "A Parallel Implementation of the Projection Method for the Incompressible Navier-Stokes Equations of Fluid Flow", Satoshi Asami, Andrew Yu, and Rick Propp

2:00-2:25. "Cartesian Mesh Gas Dynamics on the CM5", Matt Bettencourt, Dan Martin, and Jeff Bilmes

2:25-2:40 Break.

2:40-3:05. "A Parallel Algorithm for Physical Mapping of Chromosomes Using Unique Probes", Richard Tang and Deborah Weisser

3:05-3:30. "A Split-C Implementation of Concurrent B-trees", Clifford Mather

3:30-3:55. "Parallel Applications on a NOW: a Trace-driven Study", Remzi Arpaci, Todd Postma, and Amin Vahdat

3:55-4:20. "Architectural and Performance Comparison of the CM-5 and Meiko CS-2", Cedric Krumbein and Sven Meier

9 CS 267 Class Projects - 1993

9-9:30. "Parallelizing the Perfect Phylogeny Problem in Genetics", Jeff Jones

9:30-10:00. "A Parallel Global Router for Standard Cells in Electronic CAD", Lok T. Liu

10:00-10:30. "On Porting a Magnetic Field Calculation from Fortran 77 to the MasPar MP-2 and to a network of workstations using PVM", Ruth Hinkins

10:30-10:40. Break

10:40-11:10. "Modeling gain narrowing and gain saturation in a chirped pulse" laser amplifier", Rich Martin, Paul Epp

11:10-11:40. "Performance Measurements of Monte Carlo Simulation of a Nuclear Reactor on the CM-5", Sanjeev Arora, Steve Slater, Joel Fine

11:40-12:10. "Parallelizing an Atmospheric Chemical Tracer Model", Suzanne Mercer-Paley, Howard Robinson

12:10-1:10. Lunch

1:10-1:40. "A parallel MPEG Encoder for Video Compression", Szu-Tsung Cheng, Sun-Inn Shih

1:40-2:10. "Parallel alpha-beta game tree search", Michael Strittmatter, Vassilis Papavassiliou

2:10-2:40. "pRAY: An Efficient Scene-Caching Parallel Ray Tracer for Computer Graphics", Steve Von Worley, Bob Jenkins

2:40-2:50. Break

2:50-3:20. "Porting LU Decomposition to the CM5 Vector Units," Douglas Hauge, Nisha Talagala, David Jeon

3:20-3:50. "An Effecient Parallel Tridiagonal Eigenvalue solver on CM5—with Laguerre Iteration", Ren-Cang Li, Huan Ren

3:50-4:20. "Parallelizing the Solution of the Singular Value Decomposition", Da Cheng, Steve Luna, Phil Liao

4:20-4:50. "Recursively building a distributed tree structure on the CM-5", Bendict Gomes, Geoffrey Pike, James Akao