Titles and Abstracts:

Interactive Parallel Computation in Support of Research in Algebra, Geometry and Number Theory

A Workshop at MSRI Jan 29-Feb 2 organized by
Burhanuddin, Demmel, Goins, Kaltofen, Perez, Stein, Verrill, and Weening

February 1, 2007

Bailey: Experimental Mathematics and High-Performance Computing

David Bailey - Lawrence Berkeley Labs (LBL)

`http://crd.lbl.gov/~dhbailey/`

Recent developments in "experimental mathematics" have underscored the value of high-performance computing in modern mathematical research. The most frequent computations that arise here are high-precision (typically several-hundred-digit accuracy) evaluations of integrals and series, together with integer relation detections using the "PSLQ" algorithm. Some recent highlights in this arena include: (2) the discovery of "BBP'-type formulas for various mathematical constants, including pi and log(2); (3) the discovery of analytic evaluations for several classes of multivariate zeta sums; (4) the discovery of Apery-like formulas for the Riemann zeta function at integer arguments; and (5) the discovery of analytic evaluations and linear relations among certain classes of definite integrals that arise in mathematical physics. The talk will include a live demo of the "experimental mathematician's toolkit".

Bradshaw: Loosely Dependent Parallel Processes

Robert Bradshaw - University of Washington

`robertwb@math.washington.edu`

Many parallel computational algorithms involve dividing the problem into several smaller tasks and running each task in isolation in parallel. Often these tasks are the same procedure over a set of varying parameters. Inter-process communication might not be needed, but the results of one task may influence what subsequent tasks need to be performed. I will discuss the concept of job generators, or custom-written tasks that generate other tasks and process their feedback. I would discuss this specifically in the context of integer factorization.

Cohn: Parallel Computation Tools for Research: A Wishlist

Henry Cohn - Microsoft Research

`http://research.microsoft.com/~cohn/`

Cooperman: Disk-Based Parallel Computing: A New Paradigm

Gene Cooperman - Northeastern University

`http://www.ccs.neu.edu/home/gene/`

One observes that 100 local commodity disks of an array have approximately the same streaming bandwidth as a single RAM subsystem. Hence, it is proposed to treat a cluster as if it were a single computer with tens of terabytes of data, and with RAM serving as cache for disk. This makes feasible the solution of truly large problems that are currently space-limited. We also briefly summarize other recent activities of our working group: lessons from supporting ParGAP and ParGCL; progress toward showing that 20 moves suffice to solve Rubik's cube; lessons about marshalling from support of ParGeant4 (parallelization of a million-line program at CERN); and experiences at the SCIEnce workshop (symbolic-computing.org), part of a 5-year, 3.2 million euro, European Union project. Our new distributed checkpointing package now provides a distributed analog of a SAVE-WORKSPACE command, for use in component-based symbolic software, such as SAGE.

Edelman: Interactive Parallel Supercomputing: Today: MATLAB(r) and Python coming Cutting Edge: Symbolic Parallelism with Mathematica(r) and MAPLE(r)

Alan Edelman - MIT

`http://www-math.mit.edu/~edelman/`

Star-P is a unique technology offered by Interactive Supercomputing after nurturing at MIT. Star-P through its abstractions is solving the ease of use problem that has plagued supercomputing. Some of the innovative features of Star-P are the ability to program in MATLAB, hook in task parallel codes written using a processor free abstraction, hook in existing parallel codes, and obtain the performance that represents the HPC promise. All this is through a client/server interface. Other clients such as Python or R could be possible. The MATLAB, Python, or R becomes the "browser." Parallel computing remains challenging, compared to serial coding but it is now that much easier compared to solutions such as MPI. Users of MPI can plug in their previously written codes and libraries and continue forward in Star-P.

Numerical computing is challenging enough in a parallel environment, symbolic computing will require even more research and more challenging problems to be solved. In this talk we will demonstrate the possibilities and the pitfalls.

Granger: Interactive Parallel Computing using Python and IPython

Brian Granger - Tech X Corp.

`http://txcorp.com`

Interactive computing environments, such as Matlab, IDL and Mathematica are popular among researchers because their interactive nature is well matched to the exploratory nature of research. However, these systems have one critical weakness: they are not designed to take advantage of parallel computing hardware such as multi-core CPUs, clusters and supercomputers. Thus, researchers usually turn to non-interactive compiled languages, such as C/C++/Fortran when parallelism is needed.

In this talk I will describe recent work on the IPython project to implement a software architecture that allows parallel applications to be developed, debugged, tested, executed and monitored in a fully interactive manner using the Python programming language. This system is fully functional and allows many types of parallelism to be expressed, including message passing (using MPI), task farming, shared memory, and custom user defined approaches. I will describe the architecture, provide an overview of its basic usage and then provide more sophisticated examples of how it can be used in the development of new parallel algorithms.

Because IPython is one of the components of the SAGE system, I will also discuss how IPython's parallel computing capabilities can be used in that context.

Harrison: Science at the petascale: tools in the tool box

Robert Harrison - Oak Ridge National Lab

`http://www.csm.ornl.gov/ccsg/html/staff/harrison.html`

Petascale computing will require coordinating the actions of 100,000+ processors, and directing the flow of data between up to six levels of memory hierarchy and along channels that differ by over a factor of 100 in bandwidth. Amdahl's law requires that petascale applications have less than 0.001be at least 50the most regular or embarrassingly parallel applications, yet we also demand that not just bigger and better, but fundamentally new science. In this presentation I will discuss how we are attempting to confront simultaneously the complexities of petascale computation while increasing our scientific productivity. I hope that I can convince you that our development of MADNESS (multiresolution adaptive numerical scientific simulation) is not as crazy as it sounds.

This work is funded by the U.S. Department of Energy, the division of Basic Energy Science, Office of Science, and was performed in part using resources of the National Center for Computational Sciences, both under contract DE-AC05-00OR22725 with Oak Ridge National Laboratory.

Hart: Parallel Computation in Number Theory

Bill Hart - Warwick

`http://www.maths.warwick.ac.uk/~masfaw/`

This talk will have two sections. The first will introduce a new library for number theory which is under development, called FLINT. I will discuss the various algorithms already available in FLINT, compare them with similar implementations available elsewhere, and speak about what the future holds for FLINT, with the focus on parallel processing and integration into Pari and the SAGE package.

The second part of the talk will focus on low level implementation details of parallel algorithms in number theory. In particular I will discuss the design decisions that we have made so far in the FLINT library to facilitate multicore and multiprocessor platforms.

If time permits, there will be a live demonstration.

Hida: Moving Lapack and ScaLapack to Higher Precision without Too Much Work

Yozo Hida - UC Berkeley

`http://www.cs.berkeley.edu/~yozo/`

I will be discussing recent developments in Lapack and ScaLapack libraries, along with some recent work on incorporating higher precision into Lapack and ScaLapack.

Khan: Game Theoretical Solutions for Data Replication in Distributed Computing Systems

Samee Khan - University of Texas, Arlington

`sakhan@cse.uta.edu`

Data replication is an essential technique employed to reduce the user perceived access time in distributed computing systems. One can find numerous algorithms that address the data replication problem (DRP) each contributing in its own way. These range from the traditional mathematical optimization techniques,

such as, linear programming, dynamic programming, etc. to the biologically inspired meta-heuristics. We aim to introduce game theory as a new oracle to tackle the data replication problem. The beauty of the game theory lies in its flexibility and distributed architecture, which is well-suited to address the DRP. We will specifically use action theory (a special branch of game theory) to identify techniques that will effectively and efficiently solve the DRP. Game theory and its necessary properties are briefly introduced, followed by a through and detailed mapping of the possible game theoretical techniques and DRP. As an example, we derive a game theoretical algorithm for the DRP, and propose several extensions of it. An elaborate experimental setup is also detailed, where the derived algorithm is comprehensively evaluated against three conventional techniques, branch and bound, greedy and genetic algorithms.

Kotsireas: Combinatorial Designs: constructions, algorithms and new results

Ilias Kotsireas - Laurier University, Canada

ikotsire@wlu.ca

We plan to describe recent progress in the search for combinatorial designs of high order. This progress has been achieved via some algorithmic concepts, such as the periodic autocorrelation function, the discrete Fourier transform and the power spectral density criterion, in conjunction with heuristic observations on plausible patterns for the locations of zero elements. The discovery of such patterns is done using meta-programming and automatic code generation (and perhaps very soon data mining algorithms) and reveals the remarkable phenomenon of crystalization, which does not yet possess a satisfactory explanation. The resulting algorithms are amenable to parallelism and we have implemented them on supercomputers, typically as implicit parallel algorithms.

Leykin: Parallel computation of Grobner bases in the Weyl algebra

Anton Leykin - IMA (Minessota)

leykin@ima.umn.edu

The usual machinery of Grobner bases can be applied to non-commutative algebras of the so-called solvable type. One of them, the Weyl algebra, plays the central role in the computations with D-modules. The practical complexity of the Grobner bases computation in the Weyl algebra is much higher than in the (commutative) polynomial rings, therefore, calling naturally for parallel computation. We have developed an algorithm to perform such computation employing the master-slave paradigm. Our implementation, which has been carried out in C++ using MPI, draws ideas from both Buchberger algorithm and Faugere's F_4. It exhibits better speedups for the Weyl algebra in comparison to polynomial problems of the similar size.

Martin: MPMPLAPACK: The Massively Parallel Multi-Precision Linear Algebra Package

Jason Martin - James Madison University

http://www.math.jmu.edu/~martin/

For several decades, researchers in the applied fields have had access to powerful linear algebra packages designed to run on massively parallel systems. Libraries such as ScaLAPACK and PLAPACK provide a rich set of functions (usually based on BLAS) for performing linear algebra over single or double precision real or complex data. However, such libraries are of limited use to researchers in discrete mathematics who often need to compute with multi-precision data types.

This talk will cover a massively parallel multi-precision linear algebra package that I am attempting to write. The goal of this C/MPI library is to provide drop-in parallel functionality to existing number theory and algebraic geometry programs (such as Pari, Sage, and Macaulay2) while preserving enough flexibility to

eventually become a full multi-precision version of PLAPACK. I will describe some architectural assumptions, design descisions, and benchmarks made so far and actively solicit input from the audience (I'll buy coffee for the person who suggests the best alternative to the current name).

Moreno Maza: Component-level Parallelization of Triangular Decompositions

Marc Moreno Maza - Western Ontario

`http://www.csd.uwo.ca/~moreno/`

We discuss the parallelization of algorithms for solving polynomial systems symbolically by way of triangular decompositions. We introduce a component-level parallelism for which the number of processors in use depends on the geometry of the solution set of the input system. Our long term goal is to achieve an efficient multi-level parallelism: coarse grained (component) level for tasks computing geometric objects in the solution sets, and medium/fine grained level for polynomial arithmetic such as GCD/resultant computation within each task.

Component-level parallelism belongs to the class of dynamic irregular parallel applications, which leads us to address the following questions: How to discover and use geometrical information, at an early stage of the solving process, that would be favorable to component-level parallel execution and load balancing? How to use this level of parallel execution to effectively eliminate unnecessary computations? What implementation mechanisms are feasible?

We report on the effectiveness of the approaches that we have applied, including "modular methods", "solving by decreasing order of dimension", "task cost estimation for guided scheduling". We have realized a preliminary implementation on a SMP using multiprocessed parallelism in Aldor and shared memory segments for data communication. Our experimentation shows promising speedups for some well-know problems. We expect that this speedup would add a multiple factor to the speedup of medium/fine grained level parallelization as parallel GCD/resultant computations.

Noel: Structure and Representations of Real Reductive Lie Groups: A Computational Approach

Alfred Noel - UMass Boston / MIT

`http://www.math.umb.edu/~anoel/`

I work with David Vogan (MIT) on the Atlas of Lie Groups and Representations. This is a project to make available information about representations of semi-simple Lie groups over real and p-adic fields. Of particular importance is the problem of the unitary dual: classifying all of the irreducible unitary representations of a given Lie group.

I will present some of the main ideas behind the current and very preliminary version of the software. I will provide some examples also. Currently, we are developing sequential algorithms that are implemented in C++. However, because of time and space complexity we are slowly moving in the direction of parallel computation. For example, David Vogan is experimenting with multi-threads in the K-L polynomials computation module.

This talk is in memory of Fokko du Cloux, the French mathematician who, until a few months ago, was the lead developer. He died this past November.

Pernet: Parallelism perspectives for the LinBox library

Clement Pernet - University of Waterloo

`cpernet@uwaterloo.ca`

LinBox is a generic library for efficient linear algebra with blackbox or dense matrices over a finite field or Z. We first present a few notions of the sequential implementations of selected problems, such as the system

resolution or multiple triangular system resolution, or the chinese remaindering algorithm. Then we expose perspectives for incorporating parallelism in LinBox, including multi-prime lifting for system resolution over Q, or parallel chinese remaindering. This last problem raises the difficult problem of combining early termination and work-stealing techniques.

Qiang: Distributed Computing using SAGE

Yi Qiang - University of Washington

`http://www.yiqiang.net/`

Distributed SAGE (DSAGE) is a distributed computing framework for SAGE which allows users to easily parallelize computations and interact with them in a fluid and natural way. This talk will be focused on the design and implementation of the distributed computing framework in SAGE. I will describe the application of the distributed computing framework to several problems, including the problem of integer factorization and distributed ray tracing. Demonstrations of using Distributed SAGE to tackle both problems will be given plus information on how to parallelize your own problems. I will also talk about design issues and considerations that have been resolved or are yet unresolved in implementing Distributed SAGE.

Roch: Processor oblivious parallel algorithms with provable performances: applications

Jean-Louis Roch - ID-IMAG (France)

`http://www-id.imag.fr/Laboratoire/Membres/Roch_Jean-Louis/perso.html`

Based on a work-stealing schedule, the on-line coupling of two algorithms (one sequential; the other one recursive parallel and fine grain) enables the design of programs that scale with provable performances on various parallel architectures, from multi-core machines to heterogeneous grids, including processors with changing speeds. After presenting a generic scheme and framework, on top of the middleware KAAPI/Athapascan that efficiently supports work-stealing, we present practical applications such as: prefix computation, real time 3D-reconstruction, Chinese remainder modular lifting with early termination, data compression.

Tonchev: Combinatorial designs and code synchronization

Vladimir Tonchev - Michigan Tech

`tonchev@mtu.edu`

Difference systems of sets are combinatorial designs that arise in connection with code synchronization. Algebraic constructions based on cyclic difference sets and finite geometry and algorithms for finding optimal difference systems of sets are discussed.

Verschelde: Parallel Homotopy Algorithms to Solve Polynomial Systems

Jan Verschelde - UIC

`http://www.math.uic.edu/~jan/`

A homotopy is a family of polynomial systems which defines a deformation from a system with known solutions to a system whose solutions are needed. Via dynamic load balancing we may distribute the solution paths so that a close to optimal speed up is achieved. Polynomial systems – such as the 9-point problem in mechanical design leading to 286,720 paths – whose solving required real supercomputers twenty years ago can now be handled by modest personal cluster computers, and soon by multicore multiprocessor workstations. Larger polynomial systems however may lead to more numerical difficulties which may skew

the timing results, so that attention must be given to "quality up" as well. Modern homotopy methods consist of sequences of different families of polynomial systems so that not only the solution paths but also parametric polynomial systems must be exchanged frequently.

Wolf: Parallel sparsening and simplification of systems of equations

Thomas Wolf and Winfried Neun

twolf@brocku.caneun@zib.de

In a Groebner Basis computation the guiding principle for pairing and 'reducing' equations is a total ordering of monomials or of derivatives for differential Groebner Bases. If reduction based on an ordering is replaced by reduction to minimize the number of terms of an equation through another equation then on the downside the resulting (shorter) system does depend on the order of pairing of equations for shortening but on the upside there are number of advantages that makes this procedure a perfect addition/companion to the Groebner Basis computation. Such features are:

- In contrast to Groebner Basis computations, this algorithm is safe in the sense that it does not need any significant amount of memory, even not temporarily.
- It is self-enforcing, i.e. the shorter equations become, the more useful for shortening other equations they potentially get.
- Equations in a sparse system are less coupled and a cost effective elimination strategy (ordering) is much easier to spot (for humans and computers) than for a dense system.
- Statistical tests show that the probability of random polynomials to factorize increases drastically the fewer terms a polynomial has.
- By experience the shortening of partial differential equations increases their chance to become ordinary differential equations which are usually easier to solve explicitly.
- The likelihood of shortenings to be possible is especially high for large overdetermined systems. This is because the number of pairings goes quadratically with the number of equations but for overdetermined systems, more equations does not automatically mean more unknowns to occur which potentially obstruct shortening by introducing terms that can not cancel.
- The algorithm offers a fine grain parallelization in the computation to shorten one equation with another one and a coarse grain parallelization in that any pair of two equations of a larger system can be processed in parallel. In the talk we will present the algorithm, show examples supporting the above statements and give a short demo.

Yelick: Programming Models for Parallel Computing

Kathy Yelick - UC Berkeley

http://www.cs.berkeley.edu/~yelick/

The introduction of multicore processors into mainstream computing is creating a revolution in software development. While Moore's Law continues to hold, most of the increases in transistor density will be used for explicit, software-visible parallelism, rather than increasing clock rate. The major open question is how these machines will be programmed. In this talk I will give an overview of some of the hardware trends, and describe programming techniques using Partitioned Global Address Space (PGAS) languages. PGAS languages have emerged as a viable alternative to message passing programming models for large-scale

parallel machines and clusters. They also offer an alternative to shared memory programming models (such as threads and OpenMP) and the possibility of a single programming model that will work well across a wide range of shared and distributed memory platforms. PGAS languages provide a shared memory abstraction with support for locality through the user of distributed data types. The three most mature PGAS languages (UPC, CAF and Titanium) offer a statically partitioned global address space with a static SPMD control model, while languages emerging from the DARPA HPCS program are more dynamic. I will describe these languages as well as our experience using them in both numeric and symbolic applications.