## Note Taker Checklist Form -*MSRI*

**Name :** Rob Stapleton

**E-mail Address/ Phone #:** jrstaple@ncsu.edu

**Talk Title and Workshop assigned to:**
Interactive Parallel Computing in Support of Research in Algebra, Geometry, and Number Theory

**Lecturer** (Full name): Jason Martin

**Date & Time of Event:** 9:00 a.m. Jan 31, 2007

**Check List:**

(✓) Introduce yourself to the lecturer prior to lecture. Tell them that you will be the note taker, and that you will need to make copies of their own notes, if any.

(✓) Obtain all presentation materials from lecturer (i.e. Power Point files, etc). This can be done either before the lecture is to begin or after the lecture; please make arrangements with the lecturer as to when you can do this.

(✓) Take down all notes from media provided (blackboard, overhead, etc.)

(✓) Gather all other lecture materials (i.e. Handouts, etc.)

(✓) Scan all materials on PDF scanner in 2nd floor lab (assistance can be provided by Computing Staff) – Scan this sheet first, then materials. In the subject heading, enter the name of the speaker and date of their talk.

Please do **NOT** use **pencil** or colored pens other than black when taking notes as the scanner has a difficult time scanning pencil and other colors.

---

**Please fill in the following after the lecture is done:**

1. List 6-12 lecture keywords: parallel, interactive, Lapack, MPMPLapack, number theory, linear algebra, modulo n, BLAS

2. Please summarize the lecture in 5 or less sentences.
Often, over Dedekind domains, number theory problems reduce to linear algebra problems. This vastly increases the need for a portable, maintainable, powerful, parallel linear algebra tool.

***Once the materials on check list above are gathered, please scan ALL materials and send to the Computing Department. Return this form to Larry Patague, Head of Computing (rm 214)***

Jan 31, 9:00 a.m.
MPMPLAPACK: A Massively Parallel Multi-Precision Linear Algebra Package.
Jason Martin

Vaporware. Slapped together to try multiple approaches

The number theory computations he was doing would often reduce to linear algebra over Dedekind domains.
This also occurs in algebraic geometry.
There are not many pure math parallel linear algebra tools.

Wants an open source software library that can
- Handle linear algebra for modules over "nice" rings
- Reasonably use all the cores on a desktop (single machine)
- Reasonably use clusters or supercomputers
- Droppable into favorite program
- Portable
- Very, very, very maintainable

What is meant by Linear Algebra:
- Operations on/with elements, blocks, rows/columns, etc.
- BLAS-ish routines
- Higher-level (HNF, SNF, JCF, etc.)
- Lattice reductions, discriminants, etc. when possible
- Sane (computationally) representations for products (wedge, tensor, direect, etc.)
- Anything currently available on systems such as Magma

"Nice" rings are integers, rationals, intergers modulo n, finite fields, polynomial rings over integers or integers modulo n or finite fields, Dedekind domains and their fraction fields, local rings, local fields, ...
- Dedekind domains = rings of algebraic integers

Cohn's algorithms all seem to reduce -eventually- to linear algebra over Dedekind domains

At the pc level, parallel operations should make reasonable use of multicore systems, and should do this even when tasks aren't ridiculously time-intensive (most general applications should make use of the power).
What is being developed should, at any level, easily replace current linear algebra in packages such as Pari and Macaulay 2, and be easily integrated into emerging packages.

Portability:
- Should be Pure ISO C (with wrappers for usability)
- Dependence only on established, well supported libraries.
- 64 bit clean (data structures that live in memory are masked to you until they're moved and you read in the middle of the data structure), byte-order neutral
- Thread safe (all functions re-entrant)
- Simple config and build operations.

A lot of these portability issues will lead to a sub-optimal performance, but this is acceptable for portability.

Maintainability:
- Painfully rigorous coding standards. For example, full test code for every function developed. LAPACK is a good example of such rigorous standards.
- (small-ish) pool of dedicated programmers. Thinking GPL or LGPL license.

There's a large disparity over which license to choose. There's fine

points of any number of licenses, mostly when it comes to funding, or use by companies. Strong feelings in the different camps.

Ideas for the future:
Runtime profiling and tuning.
Sparse / black box support
Fault tolerance

Design decisions so far:
Modular (programming) design with recursive data types
Use pthreads/MPI hybrid approach?
-Allows single machine users to use library even if they don't have MPI. The idea is to have a server with the library that the users can use without needing heavy MPI implementation.
Scalability in the future? MPI and pthreads both have their advantages. MPI scales more easily to large (cluster, heterogeneous, etc.) environments.

The scientific computing persons seem to be pushing for GASNet.
MPI has difficulty passing complicated data structures.

Realistic approach to development:
Start with the domain of the integers.
Get an API specification by/around May 2007
Initial release of "working" code for August 2007

There is are a -lot- of details to consider when designing a large package for multi-platform general use. Architectures are so different across the machines that install-time tuning and the ability to fine-tune is almost paramount.

# MPMPLAPACK: A Massively Parallel Multi-Precision Linear Algebra Package

Jason Martin

jason.worth.martin@gmail.com

James Madison University

# Something slightly off topic

Assembly code GMP optimizations for Core 2 Intel chips (Xeon, Core 2 Duo, Core 2 Extreme, etc.) available on my webpage.

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

Works on Mac OS X and Linux.

# Motivations

- Computer engineer in former life.
- Number theorist now.
- Current research in improving bounds on discriminants of number fields.
- Research hit a road block: exhausted available computational resources.

# Motivations

- Computations in much of number theory reduce to linear algebra over Dedekind domains.
- Also occurs in algebraic geometry.
- Applied mathematicians have lots of powerful parallel linear algebra tools.
- Where are the pure math parallel linear algebra tools?

# What I Think I Want

An open source software library that:

- Handles linear algebra for modules over “nice” rings.
- Can reasonably use all the cores in my desktop.
- Can reasonably scale to use clusters or supercomputers.
- Can be dropped in to my favorite programs.
- Is very portable.
- Is very very very maintainable.

# Linear Algebra

By linear algebra I mean:

- Operations on/with elements, sub-block, rows/coloumns, etc.
- Basic linear algebra (BLAS-ish) routines appropriate to the module and ring.
- Higher-level linear algebra (HNF, SNF, JCF, RCF, etc) when possible

# Linear Algebra

By linear algebra I also mean:

- Lattice reduction, Discriminants, Differents, etc. when possible
- Computationally sane representations for direct/wedge/tensor products.
- Any linear algebra-ish operation currently available on commercial software.

# Nice Rings

When I say "nice" rings, I'm thinking of:

- $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Z}/n\mathbb{Z}$, finite fields
- Polynomial rings over $\mathbb{Z}$, $\mathbb{Z}/n\mathbb{Z}$, finite fields
- Dedekind domains and their fraction fields
- Local Rings, Local Fields

# Parallel Operation

Parallel operation on my desktop:

- Should make reasonable use of multi-core stand-alone systems.
- Should benifit from multi-core even when tasks aren't ridiculously time intensive.

# Parallel Operation

Parallel operation distributed-memory systems:

- Shift from shared to distributed memory should be transparent.

- Should scale to clusters and supercomputers with nothing more than a re-compile.

# Drop-in Functionality

- Should easily replace currently linear algebra in packages such as Pari and Macaulay 2.

- Should be easily integrated into emerging packages (e.g. FLINT, Sage)

# Portability

- Pure ISO C (C++, Python wrappers for usability)
- Dependence only on well established, well supported libraries
- 64 bit clean, byte-order neutral
- Thread safe (all functions re-entrant)
- Simple configuration and build.

# Maintainability

- Good version control
- Painfully rigorous coding standards.
- Documentation for all functions generated from code comments.
- Boundary check and random feed tests included for all library functions.
- Pool of dedicated programmers.

# Things I'm considering

- Runtime profiling.
- Runtime tuning.
- Sparse Object / Black Box support.
- Fault tolerance.

# Design decisions so far

Modular (as in programming) design with recursive data types.

- Object oriented data-method association (yes, you can do it without C++)
- Generalize algebraic routines to ring operations.
- Sacrifice some speed for specific base ring cases for greater flexibility.
- Leave hooks for optimized version of routines for special cases for those who want them.

# Design decisions so far

Use pthreads/MPI hybrid approach?

- Use MPI for distributed memory communications, pthreads for shared memory.
- Personal benchmarking for single machine shows pthreads significantly faster than MPI for some naive tests.
- Allows single machine users to use library even if they don't have MPI.
- Might be a mistake! Perhaps it's best to just use MPI. What does the audience think?

# Design decisions so far

Realistic Development Approach

- First deal with $\mathbb{Z}$.
- Initial development will focus on getting coding infrastructure in place.
- Goal for API specification is May 2007.
- Initial release goal for “working” code is August 2007.

# What I need from you

- What features do you need?
- What should the API include?
- Implementation advice?
- Volunteers??