## 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): Robert Bradshaw

**Date & Time of Event:** 1:30 p.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 <u>all</u> 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 <u>all</u> notes from media provided (blackboard, overhead, etc.)

(✓) Gather <u>all</u> 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, DSAGE, task farm, factorisation, bottleneck, firewalls.

2. Please summarize the lecture in 5 or less sentences.
There is more than one way to skin a cat in parallel. We look into the strengths and weaknesses of a few different parallelisation models.

***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)***

1:30 p.m.
Loosely Dependent Parallel Processes Bradshaw

Can be seen as a continuation of the DSAGE talk.

Two opposite ends of the spectrum: Massively Parallel or Task Farm.

Massively Parallel: MPI/shared memory, Master and slaves, constant communication
Task Farm: Occasional network access, controller and workers, intermittent communication

Integer Factorization:
Trial Division (small primes)
Quadratic Sieve ("reasonably" sized primes)
Elliptic curve methods (probabilistic, dominated by size of smallest factor)
All of these methods are embarassingly (proudly) parallelizable.

DSAGE implementation:
1 worker does Qsieve, others do ECM. If an ECM worker factors the number, the sieve is killed and restarted with a nonprime factor. The ECMs work on factoring the factors as well.
Trial division fits in as a quick check at the beginning.

The workers are SAGE instances, themselves, and have DSAGE. One can be a Worker/Controller so you can save a controller, go offline, come back, and see the results.

Communication Bottleneck
All communicatoin passes through server and client
Current implementation is extremely course-grained (workers only listen for kill signals)

Worker-to-Worker
Pros:
Can open up a wider range of problems, such as periodically sharing boundary data
Cons:
Firewalls, etc.

Inter-process communication can be done in DSAGE, but it needs more work.

Notes, black background → see schedule page