Parallel Computation Tools for Research

A Wish List

Henry Cohn

`cohn@microsoft.com`

Microsoft Research

Disclaimer

This talk contains no technical content.

My goal is to provoke discussion, ideas, and action.

My perspective is that of a user, not an expert in parallel computation.

Large computations

Increasingly common in mathematics.

Computer-assisted proofs: four color theorem, Kepler conjecture, projective plane of order 10

Tables: Stein’s modular forms database, Sloane’s packing tables

Searches: Great Internet Mersenne Prime Search (GIMPS)

Characteristics

Often embarrassingly parallelizable.

Usually carried out by ad hoc means.

What sorts of tools will enable new projects and avoid duplication of effort?

Three scenarios

Case 1: GIMPS model of recruiting many volunteers (typically otherwise idle cycles).

Case 2: use among collaborators simply for organizing a large-scale computation.

Case 3: use on multicore computers.

Current situation

Most existing tools are designed either for experts or for a specialized purpose.

Volunteer effort within pure mathematics is focused on GIMPS and the like (unserious).

Personal experience shows that setting things up takes real effort for non-experts.

My experience

One of my research interests is packing and energy minimization.

Example: how should one distribute points on a sphere?

Build tables, search for patterns, make conjectures.

Easy to distribute in principle among many processors, but involves a fair amount of bookkeeping and organization.

BOINC

Berkeley Open Infrastructure for Network Computing

Outgrowth of SETI@home.

Probably the best available infrastructure for large-scale volunteer efforts. Good for getting unused cycles or as a screensaver.

Bad choice for inexperienced organizers or smaller projects.

My first attempt

Assume all processors have access to a common file system. Use files to transmit information.

Pro: straightforward to implement, requires no specialized knowledge or software.

Con: can’t scale beyond the common file system.

Have used this approach (with roughly a dozen processors), but it is clearly limited.

Future Plans

Idea: I should develop a more scalable system for my problem.

Future Plans

Idea: I should develop a more scalable system for my problem.

Better idea: I should develop a general-purpose solution for anyone in my position.

Future Plans

Idea: I should develop a more scalable system for my problem.

Better idea: I should develop a general-purpose solution for anyone in my position.

Best idea: William Stein should develop a general-purpose solution for anyone in my position (and distribute it with SAGE).

Goal

Flexible system for automating distributed computation (only when trivially parallelizable).

Key criterion: ease of use comparable to a mainstream computer algebra system. If you have to know what a makefile is, it’s too technical.

“Buy gridMathematica” is not a good solution.

Yi Qiang is working on this (and will give a progress report at this workshop). Keyword: DSage.

Rough architecture

Anybody can set up a problem server (must be trivially easy!). Whoever knows the password can supply it with problems.

Clients download problems and upload answers (or say that they have given up). No communication between clients. Problems time out if client does not respond within some time limit.

Authentication may be required for clients. Each client user may have a unique ID and password.

Usage

May have giant public server at which many mathematicians supply problems and random people can volunteer their computing power for the good of mathematics. Potentially useful but not my personal goal.

I just want to coordinate many diverse computers (at Microsoft, at the University of Washington, at collaborators' universities, at home), and I don't want doing this to be a miniature research project itself.

Difficulties

What problems can occur? How can they be minimized?

Issues for clients

- Malicious code.
- Stability.
- Resource hogging.
- Suspendability.

This is a serious issue for volunteer clients or in a corporate environment.

Virtual machines

I see only one realistic option: the downloaded problems should be run on virtual machines. That solves all these client problems at the cost of some overhead.

Threats to servers

Denial of service is presumably the biggest issue.

Frankly not worth worrying about (risk is low and difficulty of countering it is high).

Tampering with results

Threat models:

- Mischief: anonymous volunteer reports bogus results.
- Sabotage: corrupting a rival’s research calculations.
- Fraud: exaggerating amount of calculation that has been completed or its results (with ability to blame anonymous strangers if fraud is detected).

Role of anonymity

Risks of tampering all depend on anonymity.

Use of anonymous volunteers may be crucial for certain large-scale projects, but it carries small risks. These risks are unavoidable in most cases.

Recommendation: all computations depending on volunteers who are not personally known to the organizers should be labeled as such.

I personally would not rely on such computations in the proof of an important theorem.

Verification

Can one solve tampering threats by verification of results?

We cannot trust that open source clients have not been modified (although only small risk).

One should perform consistency checks and redundant calculations, but keep in mind that this is far from infallible.

Credit

How should credit be awarded for completing distributed computations? GIMPS has dealt with this issue.

Their solution: credit Mersenne prime discoverers, code writers, et al.

This applies only to scenarios like GIMPS (where most users find nothing but occasionally someone hits the jackpot).

Credit recommendations

Use of software such as DSage should be acknowledged in research papers.

Volunteers should always be acknowledged collectively.

Any volunteers whose clients made a critical contribution (such as finding a Mersenne prime) should be acknowledged individually, but need not be included as authors.

Credit recommendations

Any individuals or organizations who contribute a substantial fraction of the total computing power (perhaps 5% or 10%) should be acknowledged individually.

These issues should be addressed before they become a problem, not after. Community consensus should be established and explained in the documentation.

The future

Within ten years the principal application will be massively multicore machines.

We don’t even know what a really good multicore computer algebra system should look like (let alone how to build one).

Too early for concrete design (don’t know memory bandwidth, etc.), but not too early to start thinking.

Multicore

What could one do with 128 cores?

- Parallel algorithms for primitive operations.
- Straightforward parallelization of user code.
- Speculative execution.
- ???

We need new ideas!