“… parallel and distributed computing has moved from a largely elective topic to become more of a core component of undergraduate computing curricula.”

- The Joint Task Force on Computing Curricula Association for Computing Machinery (ACM) IEEE Computer Society
Parallel Programming Education at Rice

For the brave souls among us:

COMP 422: Introduction to Parallel Computing

COMP 522: Multi-Core Computing

COMP 520: Distributed Systems
Parallel Programming Education at Rice

COMP 322: Fundamentals of Parallel Programming
Instructor: Prof. Vivek Sarkar

Taught using HJ-lib parallel programming library

6 Homeworks – 40% of grade
- Code template + text description of assignment
- Graded on a mix of correctness, performance, design, style, and writeup

Homeworks – 40% of grade
- Code template + text description of assignment
- Graded on a mix of correctness, performance, design, style, and writeup
Growing Pains of COMP 322

First offering in Fall 2009, required course in 2012
Growing Pains of COMP 322

Provide more feedback prior to assignment submission.

Improve latency, transparency of the grading process.

This work investigated solving these problems using an auto-grader.
Improving Feedback

Feedback currently takes the form:

- Office hours
- Online Q&A forum (Piazza)
- Catching the teaching staff after class
- Limited automatic tooling

Mostly manual, requires an attentive teaching staff, do not see students taking advantage of tools
Improving Transparency

Rubric-based grading, performance and correctness

Fragile set of scripts to collect performance information for each submission

Parallelize by question across team of graders

Manually inspect logs, code, report

High latency operation
Problems with a capital P

Class has *grown 17x* since initial offering, continues to grow.

Want to improve *feedback* during assignments.

Reduce *grading burden, latency* and *improve transparency*.

“When a student submits their code for grading they should know 80% of their grade”
Proposed Solution

Use an **auto-grader**. Must have:

- User-friendly UI, browser-based solution
- Support for correctness, performance grading of parallel programs
- Disaster recovery/backups
- Integration of third-party tools for parallel programs
- Answer “what speedup should my parallel program be getting?”
Conducted a survey of options: Mooshak, Marmoset, JavaBrat, Codewebs, WebCAT, build from scratch

Eventually chose WebCAT
  • Modularity via plugins
  • Existing Java support
  • Good reviews online, actually being used at universities

Still a lot of unsupported requirements in regards to parallelism and performance, need to construct a system around WebCAT.
System Design

Client

WebCAT

Subversion

Remote Compute Cluster

Leaderboard
System Design

```
finish(() => {
  async(() => {
    ...
  });
  async(() => {
    ...
  });
});
```

- Client
- WebCAT
- Subversion
- Remote Compute Cluster
- Leaderboard

Backup code

Submission
Single-threaded correctness tests, static checking tools like checkstyle, FindBugs
Parallel performance/scalability experiments, data race & deadlock detection, lightweight-java-profiler
Upload results to SVN and leaderboard.
System Design

WebCAT

Leaderboard
Requirements Satisfied

• Support for correctness, performance grading of parallel programs ✓

• Disaster recovery/backups ✓

• Third-party tool integration ✓

• Answer the question “how fast should my code be running?” ✓

• User-friendly UI, browser-based solution. ?
Main outstanding issue: clash between WebCAT and our requirements

- Lots of administrative complexity for a single course
- UI can be confusing without the proper mental model
- Doesn’t support performance testing well
- Core code is hard to read, impossible to build

Will use auto-grading in Spring 2016 offering of COMP 322

Expect it to be a hugely useful tool for students: instant feedback, grading transparency, more teaching staff time available for actual teaching