The Overhead of Tasks in Ciel

Ross Lagerwall

University of Cambridge

March 12, 2013

Background

Ciel is an execution engine which supports distributed data-flow programs.

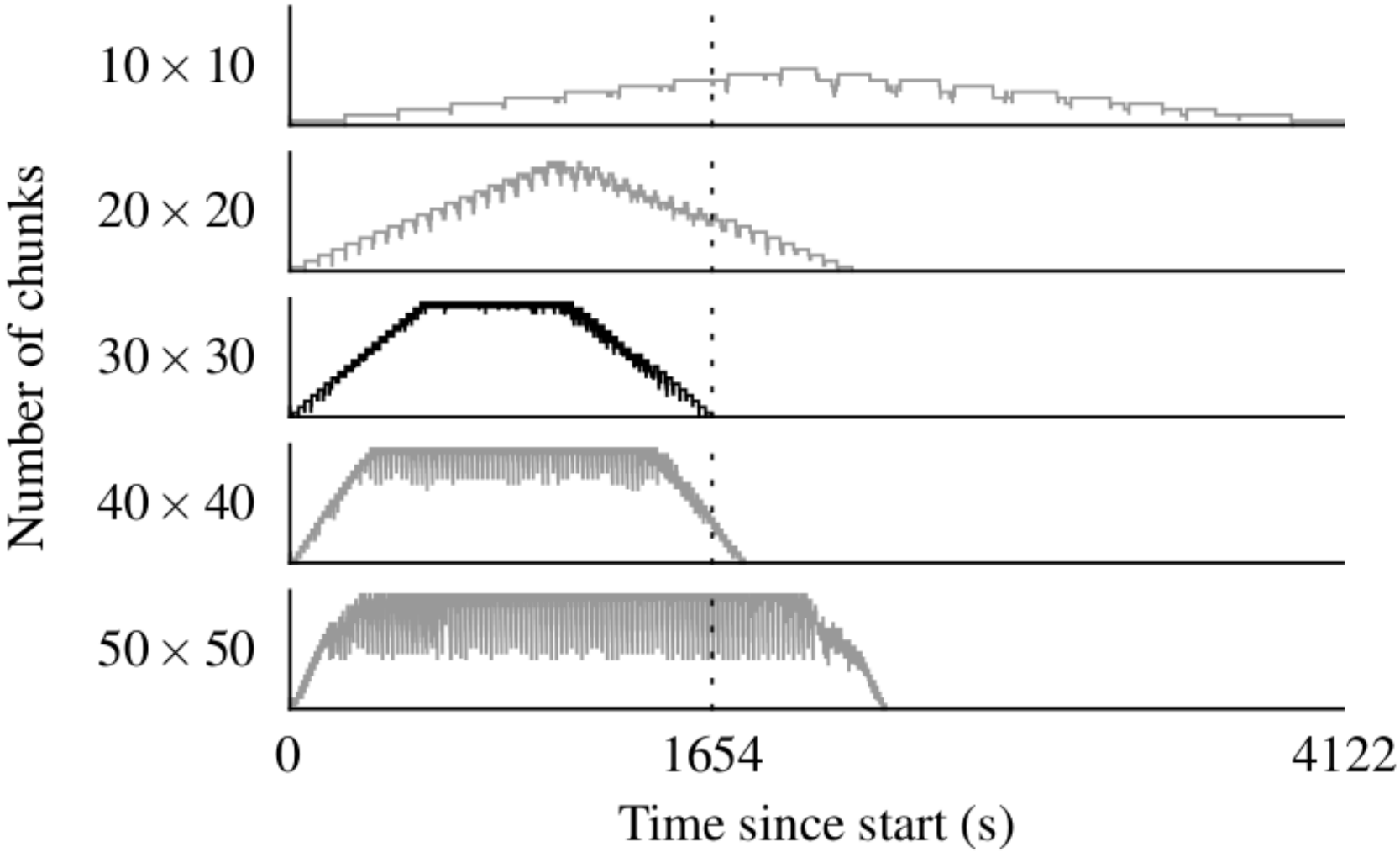
It supports dynamically spawning tasks which increases the kinds of algorithms that it can work with.

This flexibility comes at a cost:

- The Ciel task overhead is substantial which means that techniques such as batching need to be used.

The result is that the programmer needs to be conscious of Ciel when writing their algorithm.

Motivating Example



Benchmark

To do this work, develop a basic example which runs:

- Empty tasks
- Two at a time
- Such that each task depends on the combined result of the previous two

This can be used as a worst case benchmark.

Fixes

The workers and the master each run a web server (Lighttpd).

To start a task on a worker, the master makes a POST request to the worker.

Lighttpd redirects the POST request to the CGI handler (CherryPy).

CherryPy decodes the request and then executes the task.

Requests are JSON-encoded.

Too many steps and unnecessary latency: optimize!

Use a single TCP connection between the master and worker.

Fixes

This same TCP socket can be reused in the opposite direction.

Initial tests showed that it gets a ~20% improvement.

Can also look at using a binary protocol instead of the JSON encoding.

Fixes

Replacing pickle with cPickle achieved an ~10% speedup.

I will continue to look at:

- The way that the worker talks to the actual process doing the computation.
- The way that requests are sent to retrieve data from the block store and store data in the blockstore.

Conclusion

Ciel has some performance limitations for task overhead.

These mean that the programmer needs to think in terms of the target system.

There is scope for optimizations which improve the situation.