Timely Dataflow with Heterogeneous Systems

eg timely + arrayfire = win?

Nat McAleese

Heterogenous computing

Computing over a variety of hardware

What is ArrayFire?

Free; general-purpose; open-source (kinda)

Targets parallel and massively-parallel architectures including CPUs, GPUs, and other hardware acceleration devices.

Used on devices from low-powered mobile phones to high-powered GPU-enabled supercomputers

And it has **rust** bindings!

What is Timely?

“Timely dataflow is a low-latency cyclic dataflow computational model, introduced in the paper Naiad: a timely dataflow system. This project is an extended and more modular implementation of timely dataflow in Rust.” [0]

[0] https://github.com/frankmcsherry/timely-dataflow

Why is this worthwhile?

We should expect a latency / throughput trade-off - but you never really know ‘till you measure!

It should be legitimately useful for people trying to do compute-bounded dataflow computations.

Work Plan

Run some timely demos ✓

Run some arrayfire demos ✓

Pick benchmarks

Build testbed

Measure

Extend!

```
nat@nat-xps-13 ~/.../naiad-gpu/src $ RUST_BACKTRACE=1 cargo run
    Finished dev [unoptimized + debuginfo] target(s) in 0.0 secs
     Running `/home/nat/Dropbox/part-3-coursework/large_scale/open-source/naiad-gpu/target/debug/naiad-gpu`
ArrayFire v3.5.1 (CPU, 64-bit Linux, build 0a675e8)
[0] Intel: Intel(R) Core(TM) i5-2467M CPU @ 1.60GHz, 3271 MB, Max threads(4)
Info String:
ArrayFire v3.5.1 (CPU, 64-bit Linux, build 0a675e8)
[0] Intel: Intel(R) Core(TM) i5-2467M CPU @ 1.60GHz, 3271 MB, Max threads(4)
Arrayfire version: (3, 5, 1)
Element-wise arithmetic
a =>
[5 3 1 1]
    0.6010     0.7273     0.8704
    0.1195     0.4348     0.2977
    0.2643     0.9311     0.9509
    0.3945     0.1210     0.9859
    0.2261     0.5332     0.1123

sin(a) + 1.5 =>
[5 3 1 1]
    2.0654     2.1649     2.2646
    1.6192     1.9212     1.7933
    1.7612     2.3023     2.3139
    1.8844     1.6207     2.3338
    1.7241     2.0083     1.6120

a(seq(1,3,1), span)
[3 3 1 1]
    0.1195     0.4348     0.2977
    0.2643     0.9311     0.9509
    0.3945     0.1210     0.9859

a(indices, seq(0, 2, 1))
[3 3 1 1]
    0.1195     0.4348     0.2977
    0.2643     0.9311     0.9509
    0.3945     0.1210     0.9859

nat@nat-xps-13 ~/.../naiad-gpu/src $
```

```
nat@nat-xps-13 ~/.../naiad-gpu/src $ cargo run
   Compiling naiad-gpu v0.1.0 (file:///home/nat/Dropbox/part-3-coursework/large_scale/open-source/naiad-gpu)
warning: unused import: `Exchange`
 --> main.rs:4:42
  |
4 | use timely::dataflow::operators::{Input, Exchange, Inspect, Probe};
  |                                          ^^^^^^^^
  |
  = note: #[warn(unused_imports)] on by default

    Finished dev [unoptimized + debuginfo] target(s) in 11.20 secs
     Running `/home/nat/Dropbox/part-3-coursework/large_scale/open-source/naiad-gpu/target/debug/naiad-gpu`
worker 0:       hello 0
worker 0:       hello 1
worker 0:       hello 2
worker 0:       hello 3
worker 0:       hello 4
worker 0:       hello 5
worker 0:       hello 6
worker 0:       hello 7
worker 0:       hello 8
worker 0:       hello 9
nat@nat-xps-13 ~/.../naiad-gpu/src $
```

Planned Benchmarks

Wordcount

Streaming KMeans

MLP training?