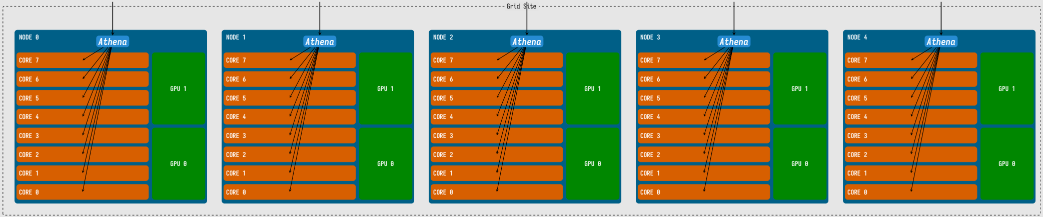
ATLAS Experience with HPX

Beojan Stanislaus

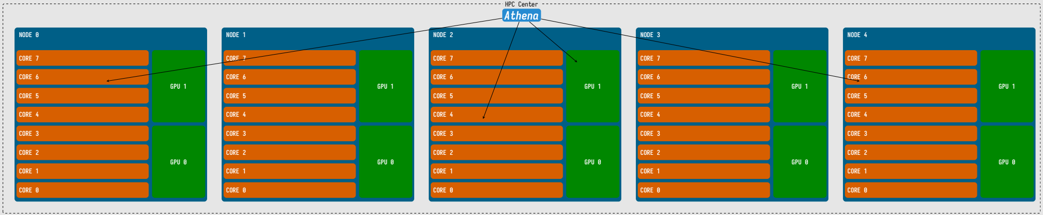
HSF Meeting
29th November 2023

Ultimate Goal

Traditional Grid Sites



HPC Concept



Baby Steps

- Before we can run (dispatching algorithms), need to walk (dispatch events)
- First go at this uses Ray – Raythena

Scheduling with Ray – Raythena

🔗 CHEP 19 Talk

- Use Ray to distribute events over nodes, Athena on each node
- Ray Driver process on one node handling comms with outside world
- Ray Actor process on each worker managing a separate Athena process
  - Feeds events to Athena using the *Event Service* idea already in Athena
- Mostly implemented in 2019, with inefficiencies due to merging output after running
  - Recently improved with on-the-fly merging

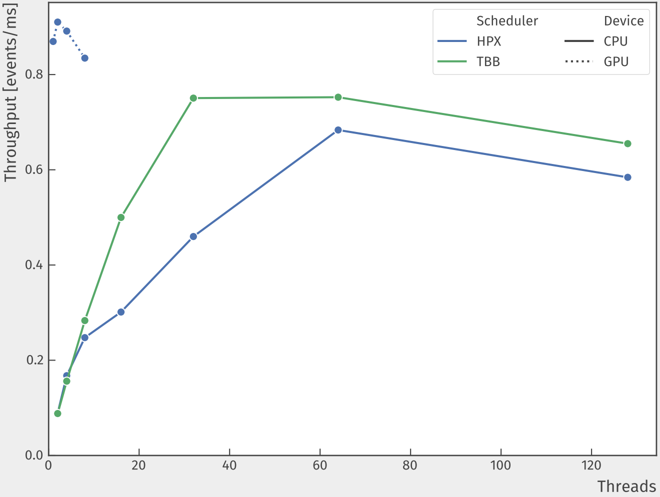
On to HPX

- Want to reduce the number of moving parts
- HPX seemed really promising
  - C++ API so it can be integrated into Athena (Ray is in Python)
  - Built for HPCs – Can handle inter-node and intra-node scheduling
  - Support for GPU acceleration

First Impressions

- First built toy prototype scheduler to compare against TBB flow graph
- Immediately saw a number of issues
  - Scheduling seemed slower than in TBB
  - API was a bit finicky (can't just wrap anything in a future)
  - Built-in CUDA support is too limited to be useable
  - Defaults to one queue per hardware thread

HPX vs TBB



HPX slower than TBB

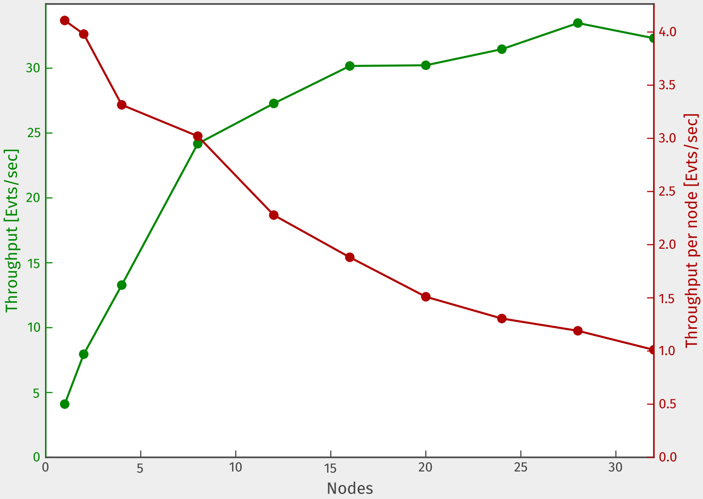
Pushing On

- Pushed on anyway, integrated into Gaudi, then into Athena
- Futures API needed a shim to look like TBB arena API
    - Can't combine HPX and TBB, according to HPX developers
- Work needs to be explicitly launched on a specific node
    - No unified memory space – can't just dispatch algorithms to different nodes
    - End up dispatching events to be scheduled by Athena on each worker

The Fatal Flaw

- Remote launches and local launches have different API calls
- Both end up running on the same thread pool
- Separate queue per hardware thread means you can end up stuck behind slow work
- This interacts *badly* with Athena event loop model (schedule “draining” of slot when we run out):
  - If compute work on event ends up scheduled behind task to “drain” slot, event can never complete
- Even with a global queue, pushing event can take a significant time
  - Delays of 35 ms to push an event – Seriously limits max throughput

HPXGaudi Performance

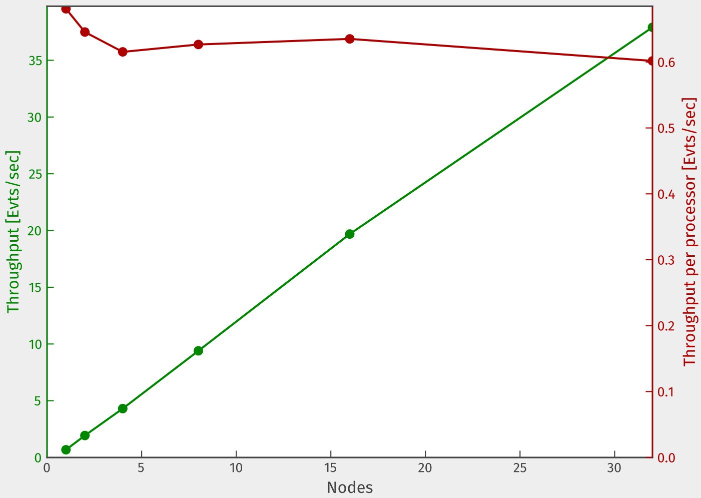


Throughput doesn’t scale

Switching to MPI

- Given up and switched to using MPI (sticking with TBB for local scheduling)
- Already have a working prototype of Athena MPI, now looking to test with grid integration
- Implementation is much cleaner
- Pull model instead of push model
- Roughly 15 µs (round-trip) to *pull* an event

MPI-Gaudi Performance



Near ideal scaling (tested on different system)

Things we never figured out

- Both HPX and TBB show drop in performance with 128 threads
- For some reason performance much better on Cori KNL than Perlmutter
  - Maybe artefact of the way Gaudi CPUCruncher works