

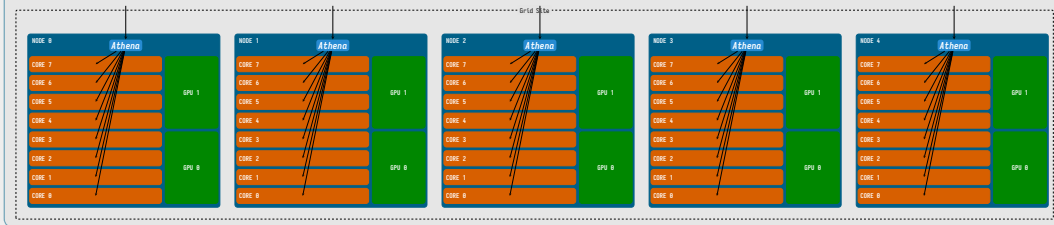
Evaluating HPX as a Next-Gen Scheduler for ATLAS on HPCs

*Paolo Calafiura, Julien Esseiva, Xiangyang Ju, Charles Leggett,
Beojan Stanislaus, and Vakho Tsulaia
on behalf of the ATLAS Collaboration*

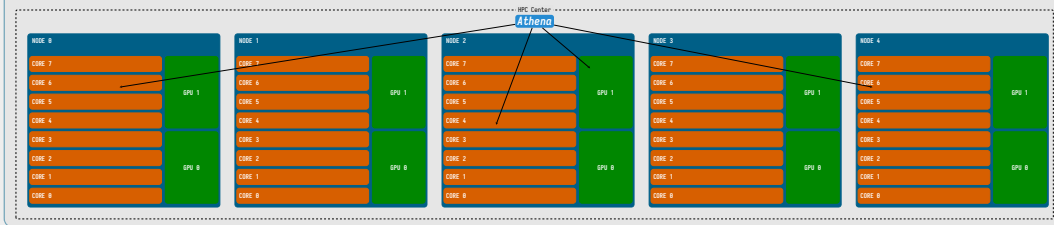
- Processed using the Worldwide LHC Computing Grid
 - Global network of large data centres contributed by institutes / countries
- Coordinated by *PanDA* – splits work and submits jobs into local batch systems
 - Decides where to send work considering proximity to input data
- Submit single node (multicore) jobs, with some single core jobs
- On each node, scheduling handled by custom scheduler within *Athena* framework

- HEP needs High *Throughput* Computing, governments like to build High *Performance* Computing
- HPC centres often
 - Don't like small (few node) jobs
 - Increasingly focused on accelerators (GPUs, FPGAs)
 - Nodes have limited / no network access to outside world
- Want to extend Athena to schedule work over many nodes (and improve scheduling across cores / on accelerators)
- Dream: Ability to exploit “heterogeneous heterogeneous” resources with multiple node types specialized for different tasks

Traditional Grid Sites



HPC Concept



- 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

Next Steps

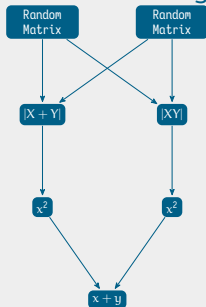
- Replace Athena scheduler with Ray
- HPX identified as potential alternative

Evaluating HPX

- Ray API is in Python, HPX is in C++
 - HPX can be integrated directly into Athena
- HPX can handle both inter-node and intra-node scheduling
 - In theory, also heterogeneous

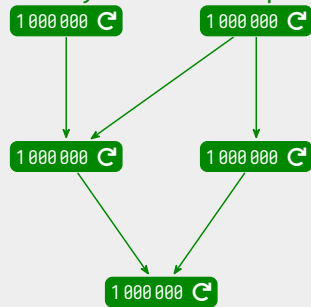
Toy prototype schedulers using different technologies

HPX and TBB on a single node



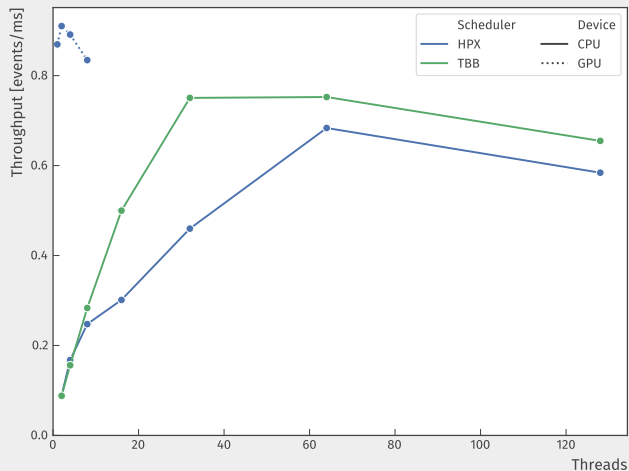
- Matrices are 1000x1000
- TBB means oneTBB and the flow graph API

HPX and Ray across multiple nodes



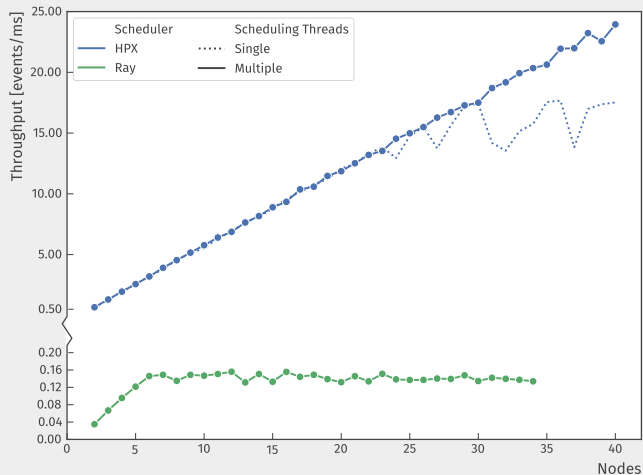
- Busy loops

HPX vs TBB (Single Node)



- GPU throughput ~constant ∴ CUDA serializes
- HPX slower to schedule at higher thread counts (task dependent)
- Both schedulers show unexpected behaviour at 128 threads
 - Investigating, possibly cache related

HPX vs Ray (Multiple Nodes)



- Throughput levels off at around 23 nodes with a single scheduling thread
 - Scheduling thread can't keep up with nodes, resolved with parallel scheduling
- Ray exhibits *much* slower performance
 - Parallel scheduling does not help here

Positives

- Single scheduler across cores and nodes
- In principle, HPX appears capable of handling scheduling needed

Problems Encountered

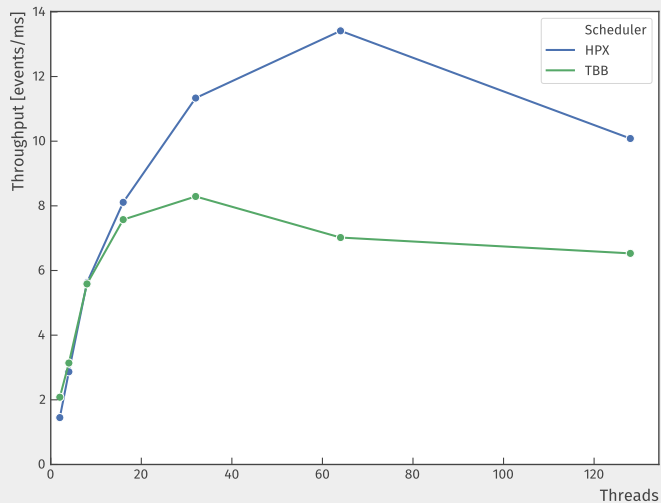
- Worse single-node performance than TBB
- HPX needs compute graph expressed as functions taking and returning futures
 - Appear to be limitations to what can be wrapped in a future
- Built in CUDA support is *very* limited
 - Ended up using CPU tasks calling CUDA directly
- Need to manually throttle submission into HPX to control resource use
- Need to override default queue-per-hardware-thread scheduling policy

Next Steps / Integration into Athena

- Summary: Not a magic bullet
 - Can't say goodbye to Athena scheduler and replace it wholesale
 - Teething pains with larger numbers of nodes
- Nevertheless, viable as a TBB replacement with inter-node scheduling capability
- Would need to use `void` futures or simple `ExecutionContext` futures
 - Combination of HPX limitation and ease of use of API
- Advisable to keep custom scheduler and use HPX as a “threading” layer
 - Need resource control features we currently have

Backup

HPX vs TBB (300x300 Matrices)



HPX faster for smaller problem




Perlmutter CPU Node

```
OS: SUSE Linux 15 SP3 x86_64
Host: HPE_CRAY_EX425 1.6.3
Kernel: 5.3.18-150300.59.87_11.0.78-cray_shasta_c
Uptime: 8 days, 23 hours, 30 mins
Packages: 1238 (rpm)
Shell: zsh 5.6
Terminal: /dev/pts/8
CPU: AMD EPYC 7763 (256) @ 2.450GHz
Memory: 24482MiB / 515316MiB
```

Perlmutter GPU Node

```
OS: SUSE Linux 15 SP3 x86_64
Host: HPE_Cray_EX235n 1.2.1
Kernel: 5.3.18-150300.59.87_11.0.78-cray_shasta_c
Uptime: 9 days, 2 mins
Packages: 1238 (rpm)
Shell: zsh 5.6
Terminal: /dev/pts/8
CPU: AMD EPYC 7763 (128) @ 2.450GHz
GPU: NVIDIA A100 SXM4 40GB
GPU: NVIDIA A100 SXM4 40GB
GPU: NVIDIA A100 SXM4 40GB
GPU: NVIDIA A100 SXM4 40GB
Memory: 20154MiB / 257306MiB
```


Technologies

-  oneAPI TBB
-  HPX
-  Ray