BERKELEY LAB
Bringing Science Solutions to the World

ATLAS EXPERIMENT

U.S. DEPARTMENT OF ENERGY
Office of Science

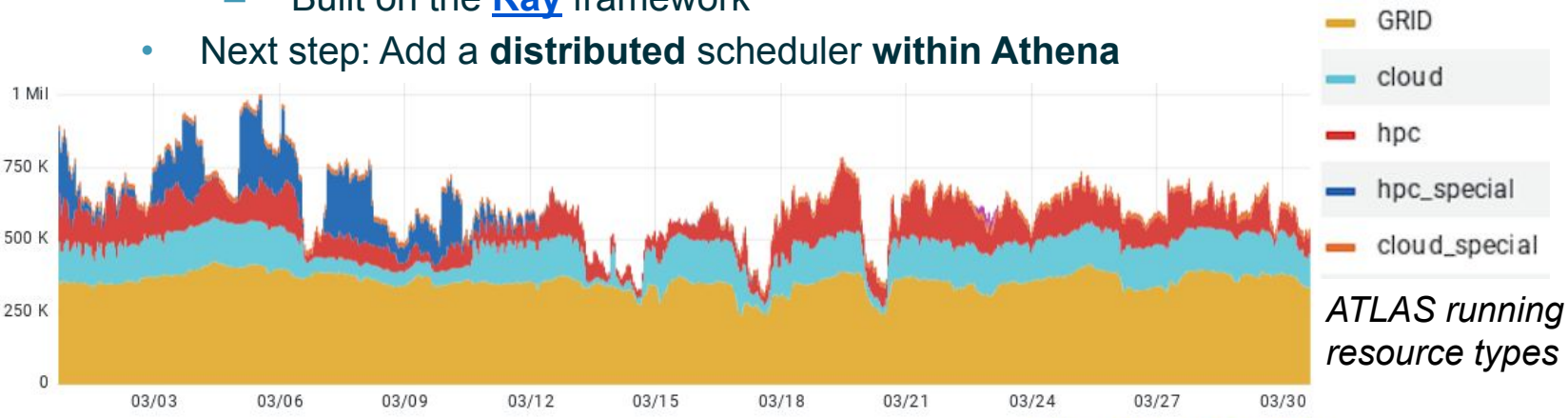
Towards a distributed heterogeneous task scheduler for the ATLAS offline software framework

Julien Esseiva, Beojan Stanislaus
Paolo Calafiura, Xiangyang Ju, Charles Leggett, and Vakho Tsulaia
For the ATLAS Collaboration

CHEP 2023 - 05.11.2023

HPC Computing in ATLAS

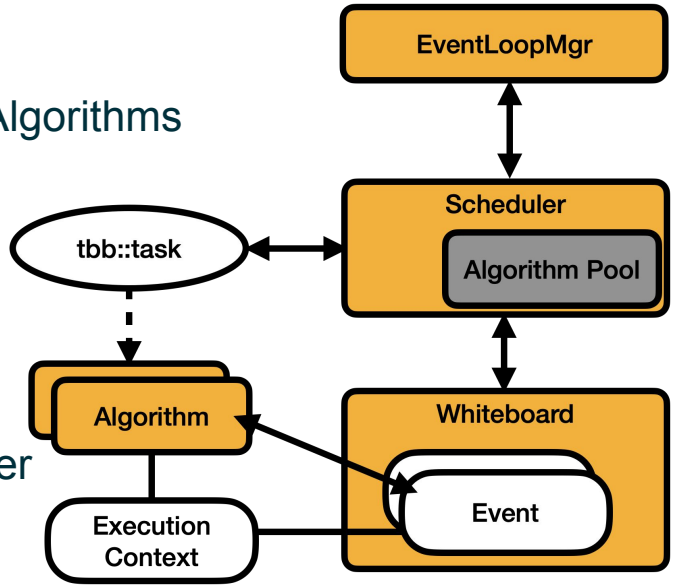
- Significant fraction of jobs running on HPCs
- **Grid-like jobs** → Single node
  - long initialization time and load unbalance
- **Raythena**, a task farm for HPCs is an intermediary step
  - Built on the **Ray** framework
- Next step: Add a **distributed** scheduler **within Athena**



ATLAS running jobs by resource types for March 2023

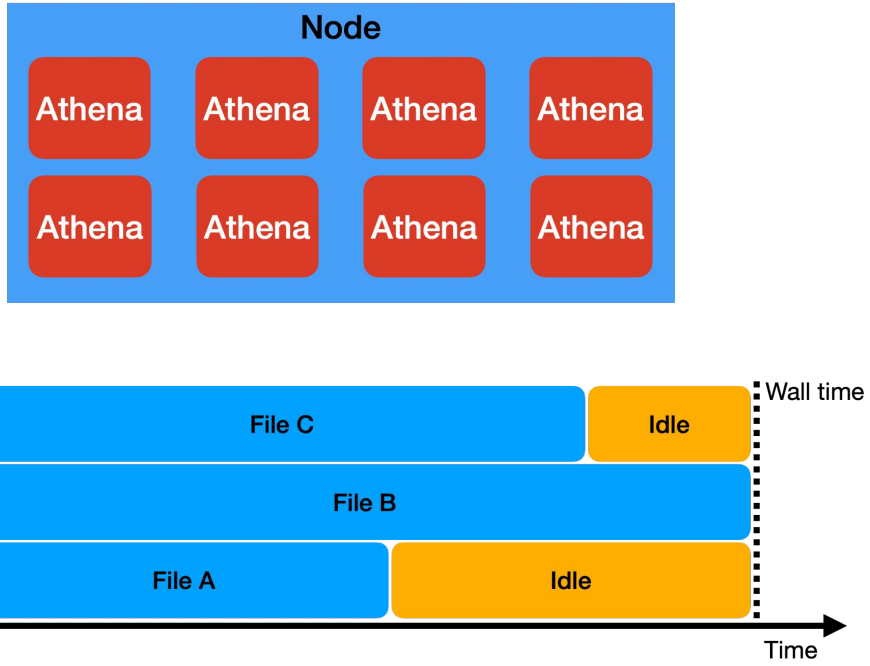
Atlas Event Processing Framework

- **Athena** is based on the **Gaudi** framework with ATLAS extensions
- **AthenaMT** is the multithreaded version of Athena
  - Task scheduler is the central component
- Scheduler solves control / data flow dependencies of Algorithms
  - Uses **TBB** to parallelize work
  - Maximize throughput
- Whiteboard architecture (StoreGate)
  - Algorithms read/write in a **shared data store**
- Each event is a **separate execution graph**
  - Event loop manager sends events to the scheduler
  - Graph can be traversed differently for each event

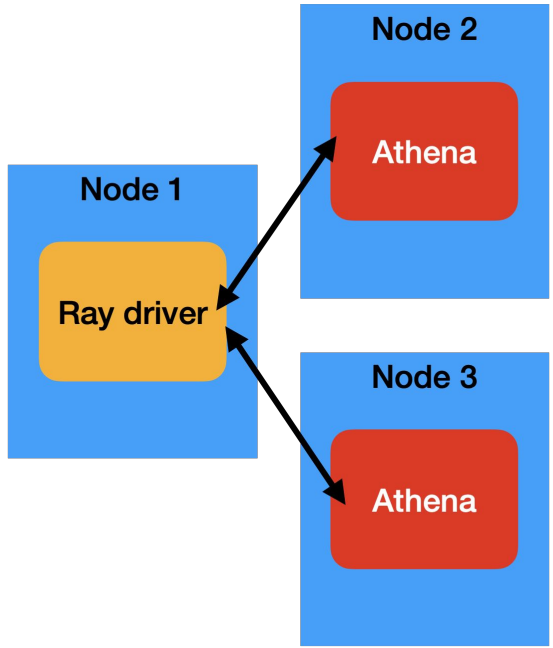


Athena on HPCs Today

32cores / Athena process, assigned one file each



In Production: File-level parallelism

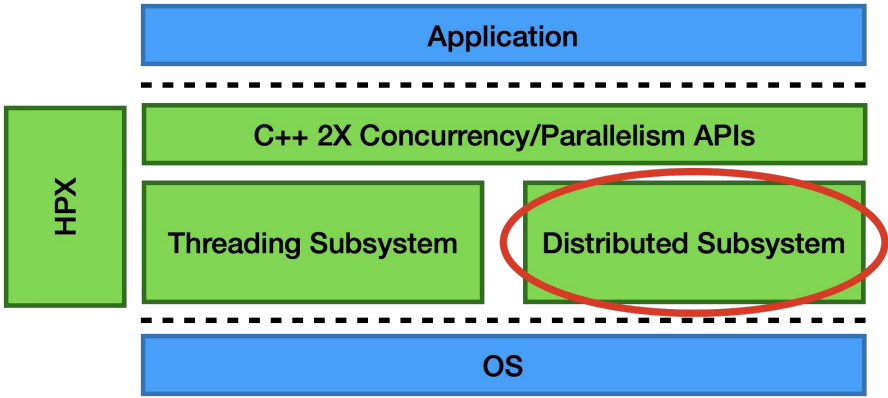


256cores / Athena process. Events are streamed to Athena as needed

Raythena: Event-level parallelism

Towards a Distributed Athena/Gaudi Scheduler: HPX

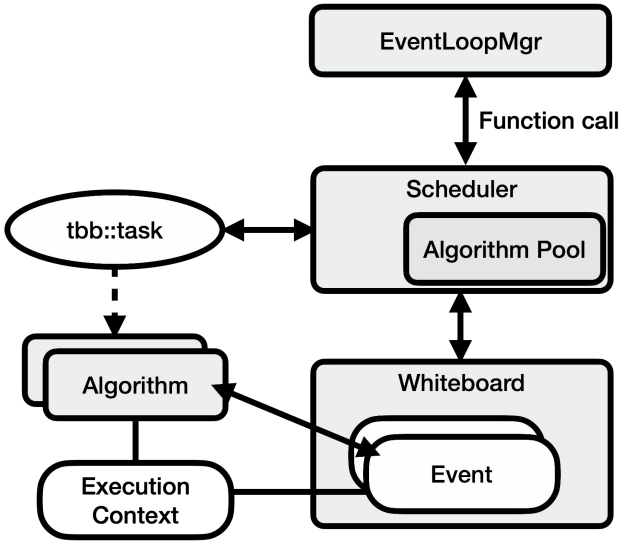
- C++ library for concurrency and parallelism
- Conform to the standard with extensions
  - We are interested in **distributed** capabilities
- **Uniform API** for inter and intra node scheduling
- CUDA support
  - Wrap kernel execution in a future
- Terminology
  - ***Actions*** → remotely callable function
  - ***Components*** → remote objects
  - ***Localities*** → remote processes

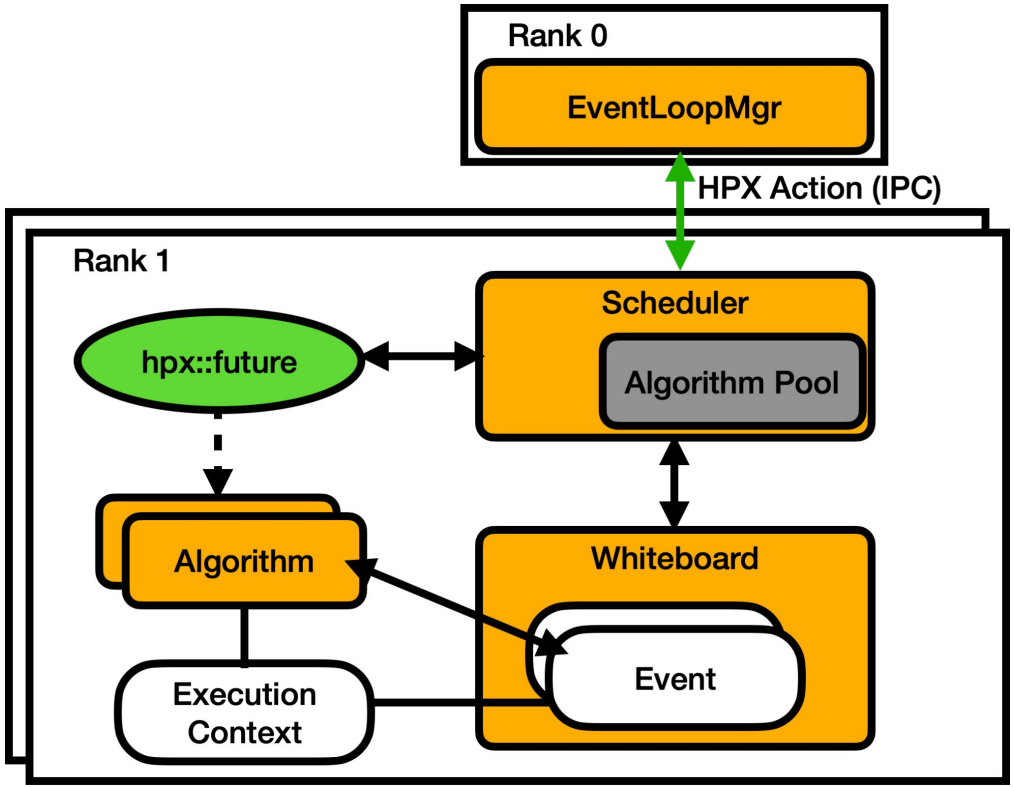


HPX architecture[1]

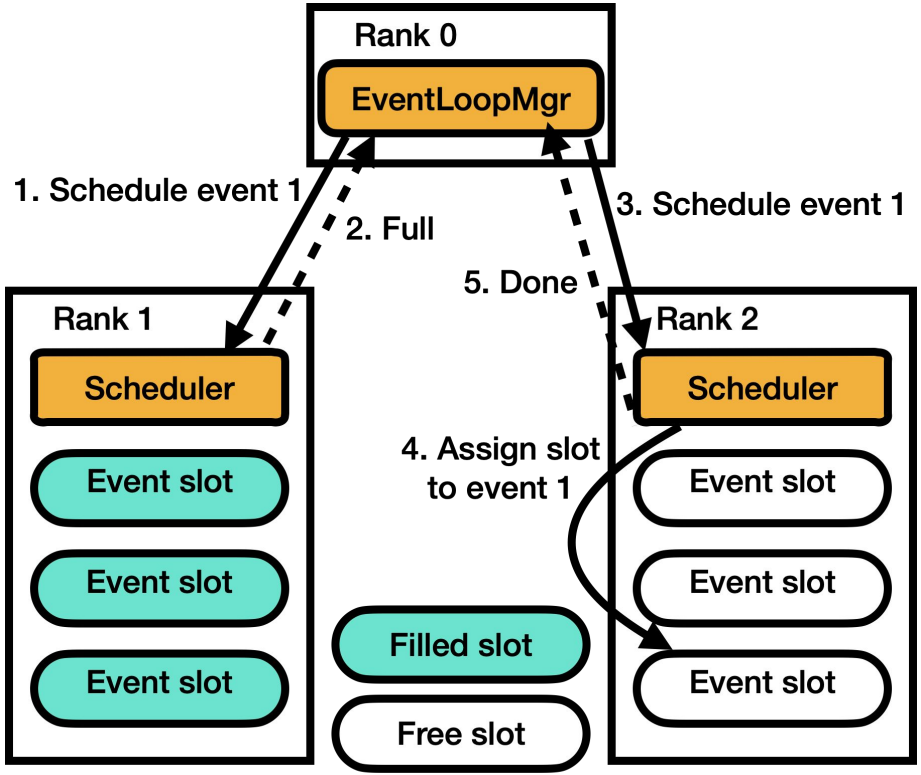
[1] derived from https://doi.org/10.21105/joss.02352

Prototyping HPX-Gaudi Integration





Task distribution strategy



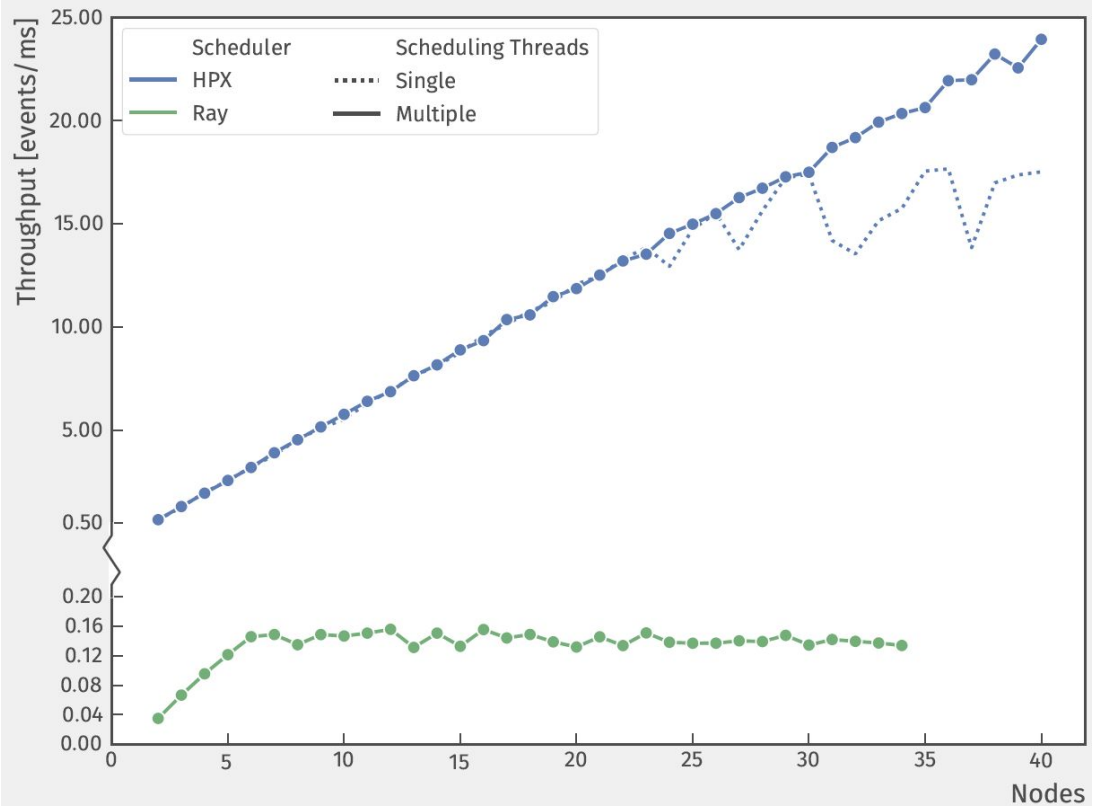
Inter-process event scheduling

Implementation Challenges

- We took care of a lot of extra technicalities:
  - Sequencing for starting / stopping HPX runtime
    - Scheduler needs to have the HPX runtime setup
    - When the event loop on the controller ends, notify workers to stop their HPX runtime
  - Build issues with MPI
    - Gaudi uses OpenMPI
      - HPX crashes with OpenMPI and shared queuing
    - Need to override OpenMPI with MPICH
      - Then NERSC shifter overrides MPICH with CRAY-MPICH

Standalone prototype performance

- Presented at ACAT 2022
- Perlmutter + MPI
  - 128 threads / node
- Weak scaling
  - 200ms / event
  - 1280 events / node
- CPU cruncher



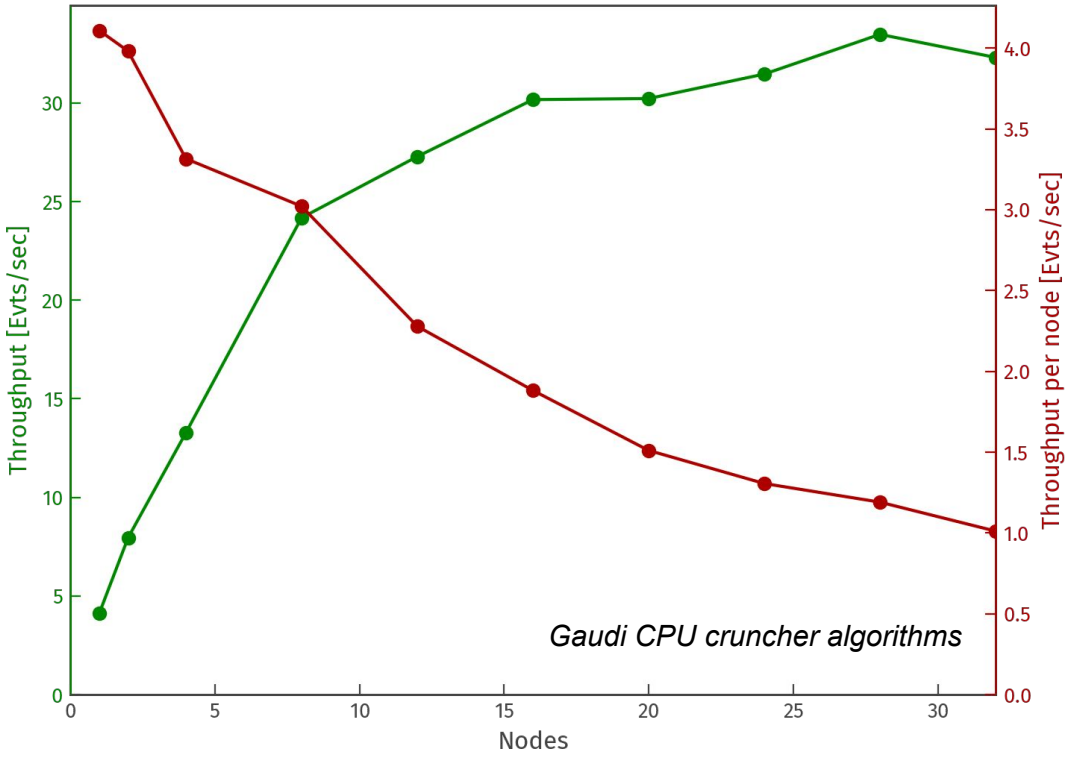
Evaluating Prototype Performance - Setup

- We're interested in the event processing **throughput** scaling with the number of nodes
- Measurements were done on **Cori** at NERSC
  - KNL partition, 128 threads / node
  - HPX TCP backend
- Simplified workload
  - **CPU cruncher** algorithm
  - No I/O
  - 4 Algorithms/event, ~15s/event
- Increase problem size as we increase nodes (**weak scaling**)
  - 3200 events/node, at least 12800 events

Prototype Performance: First Results

Issues we are exploring:

- Tight scheduling budget:
  - 10 nodes → 1280 threads → Granularity of a task is ~15s
    - Schedule one event every **~10 ms**
    - Including network communication
- It takes **30ms** to do the scheduling
  - HPX/network/scheduling



Performance - More Issues to Explore

- Currently using TCP instead of MPI
  - Most likely the main reason for the 30ms overhead
- We saw scaling issues when scheduling from a single thread in the standalone prototype
- Load balancing:
  - task duration is not very consistent
  - Are we overcommitting the machine?

Summary and Outlook

- Prototype integration of HPX within Gaudi can **schedule events across nodes**[1]
- Gaudi **task scheduler is still needed**, can't replace everything with HPX
  - HPX schedules events on nodes and user threads on OS threads
  - Gaudi schedules Algorithms on user threads
- Next steps
  - **Identify and fix scaling bottlenecks**
  - Integrate into Athena
  - Test performance with real workflow
  - Test distributed I/O with ROOT
  - Run heterogeneous workflows

[1] https://gitlab.cern.ch/hpxgaudi/Gaudi/

Backup

Computing in ATLAS

- Mainly using the Worldwide LHC Computing Grid
- PanDA coordinates and submits jobs to individual sites
  - ATLAS distributed workflow management system
  - Considers many factors: resources, availability, proximity to data
- Single node, multicore jobs
  - Athena - ATLAS simulation, reconstruction and data analysis software framework - does the scheduling within a node

HPX-Gaudi Integration

- Where is HPX used
  - IPC interface
  - Thread pool
- Event loop on rank 0 is also a worker
- HPX init / finalize is done in ThreadPoolSvc
- Mutex shared by all actions
- Only exchange event ID
  - Each process reads data from disk

HPX-Gaudi Integration

- We started with the less intrusive implementation
- Keep the current scheduler
  - Use HPX as threading / distributed abstraction
- Controller / worker architecture
- Gaudi event loop / scheduler are not reentrant
  - Event scheduling on workers is serialized
- Keep the processing of an event on the same node
  - Not sharing event data across nodes
- Actions are free functions with references to eventloop and scheduler
  - declared as friend, avoid the need for HPX components

HPX-Gaudi Integration