D Benjamin, P Calafiura, T Childers, K De, A Di Girolamo, E Fullana, W Guan, T Maeno,
N Magini, P Nilsson, D Oleynik, S Sun, V Tsulaia, P Van Gemmeren, T Wenaus and W Yang

For the ATLAS Collaboration

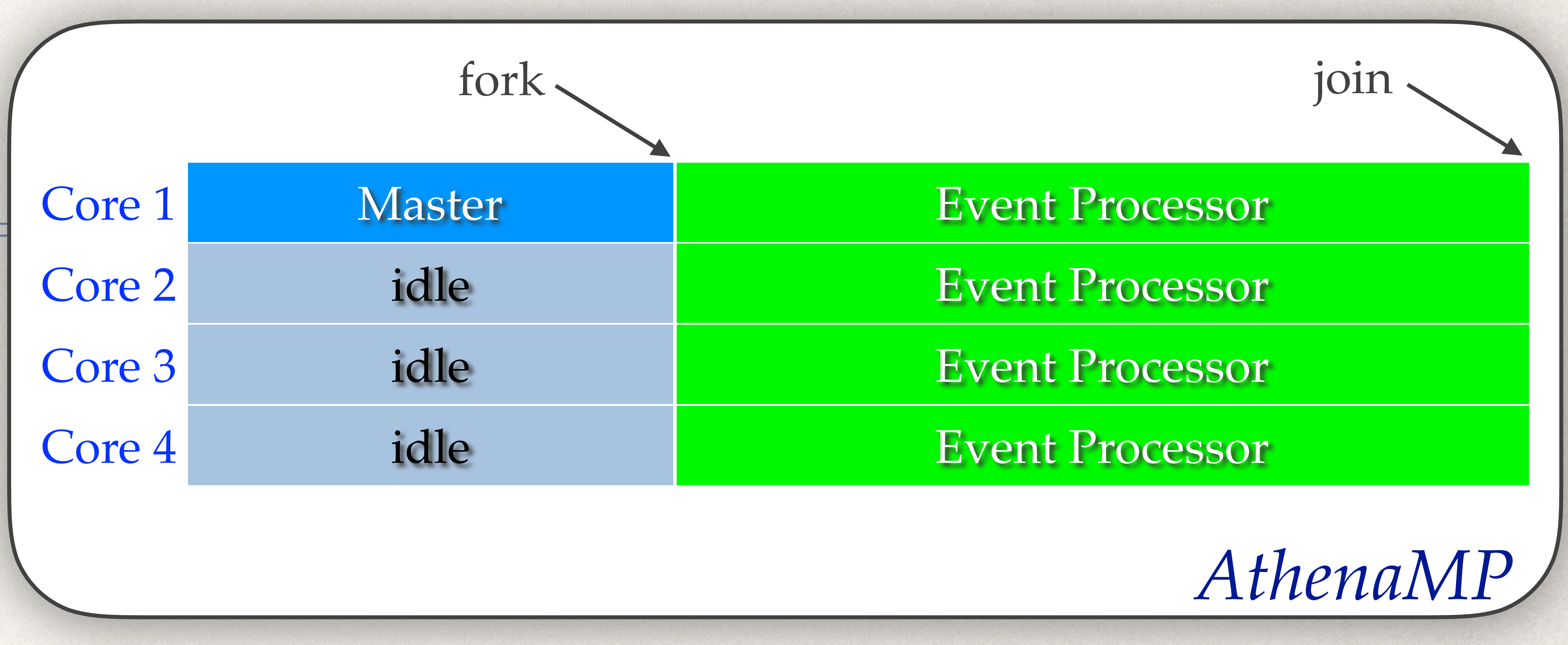
Fine-grained processing towards HL-LHC computing in ATLAS

Exascale Computing for High Energy Physics session @ eScience 2018, Amsterdam

31-October-2018

Glossary

- AthenaMP
  - Multi-process version of the ATLAS reconstruction, simulation and analysis framework Athena.



fork

join

Core 1	Master	Event Processor
Core 2	idle	Event Processor
Core 3	idle	Event Processor
Core 4	idle	Event Processor

AthenaMP

- PanDA
  - Production and Distributed Analysis system. Used by ATLAS for running production workflows on a variety of computing resources (e.g. Grid, HPC, Clouds) worldwide
- Pilot
  - PandDA component. Manages an instance of AthenaMP on a compute node (input stage-in, output stage-out, job monitoring, etc.)

Why fine-grained processing?

- Traditional workflow in ATLAS:
  - Pilot process on a compute node starts an instance of AthenaMP
  - Pilot assigns a fixed number of events to AthenaMP
  - **Pilot waits until AthenaMP is done processing all events**
  - If an error occurs during the processing of some event, the entire instance of AthenaMP is terminated and all event processing outputs produced so far are discarded

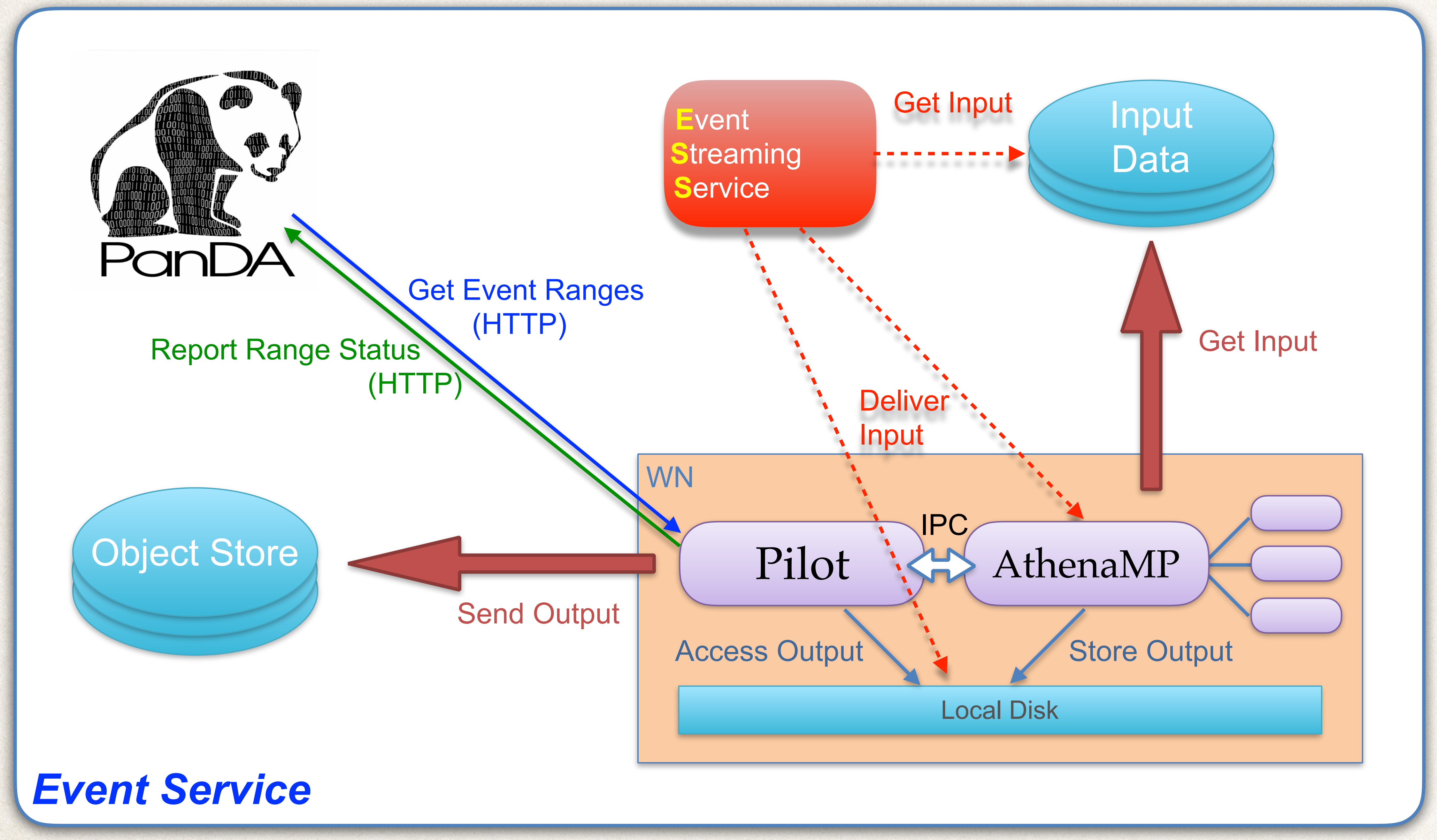
- This behavior is not suited for
  - Opportunistic running (the compute node can be taken away from the job at any time)
  - Running as part of an MPI job on multiple HPC nodes (wasting CPU time on all compute nodes while waiting for the slowest one to finish its task)

Why fine-grained processing? (contd.)

- Fine-grained workflow in ATLAS:
  - Pilot process on a compute node starts an instance of AthenaMP
  - **Pilot delivers chunks of input events ("event ranges") to the running AthenaMP**
  - Outputs of event ranges are saved as soon as they have been produced
  - If an error occurs during the processing of some event range, the range is reported as failed and the processing continues

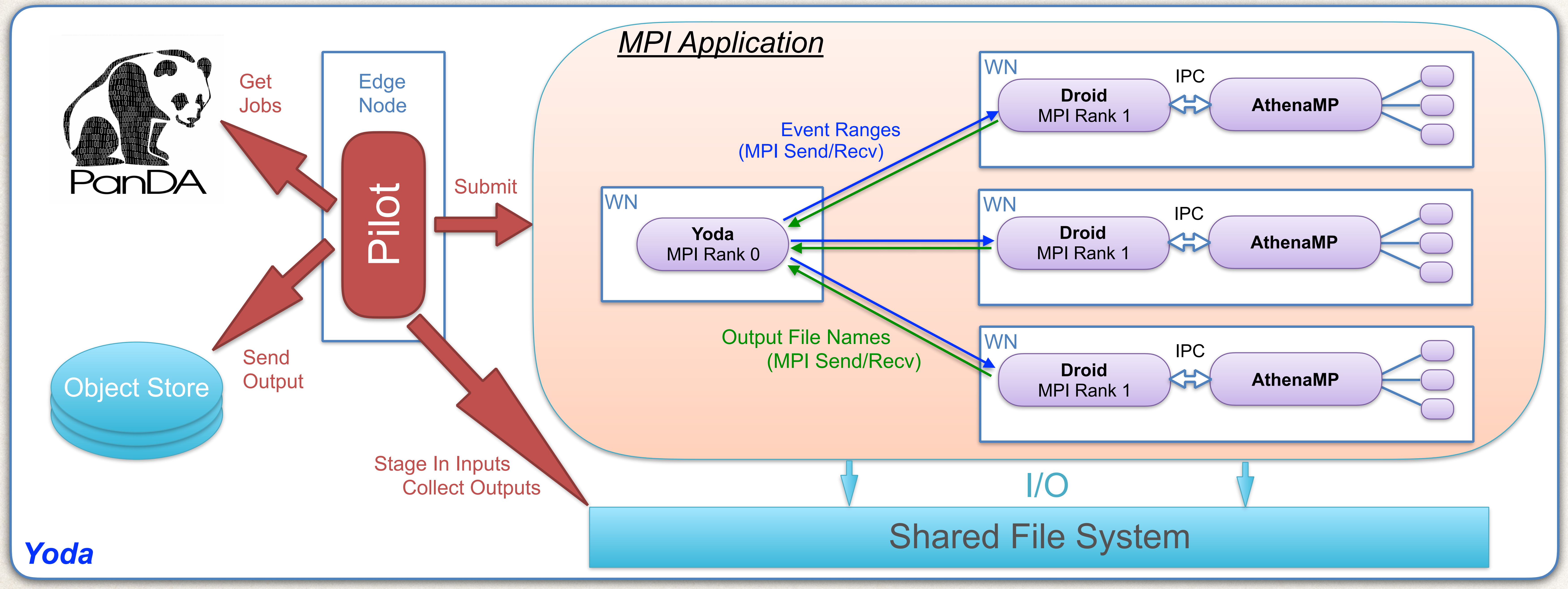
- This behavior is well suited for
  - Opportunistic running (if the compute node vanishes, we lose only those ranges which are currently being processed)
  - Running as part of an MPI job on multiple HPC nodes (by delivering fine-grained inputs at runtime we keep all compute nodes busy for the duration of the job)

Event Service



- The JEDI (Job Execution and Definition Interface) extension to PanDA breaks down production tasks based on optimal usage of available resources
- Pilot communicates with PanDA/JEDI over HTTP
  - Pull new input event ranges
  - Report the status of completed ranges
- AthenaMP writes new output for each completed event range
- Fine-grained outputs are streamed in real-time to Object Stores
- Missing Component: Event Streaming Service. Discussed later in this presentation

Yoda - Event Service on Supercomputers



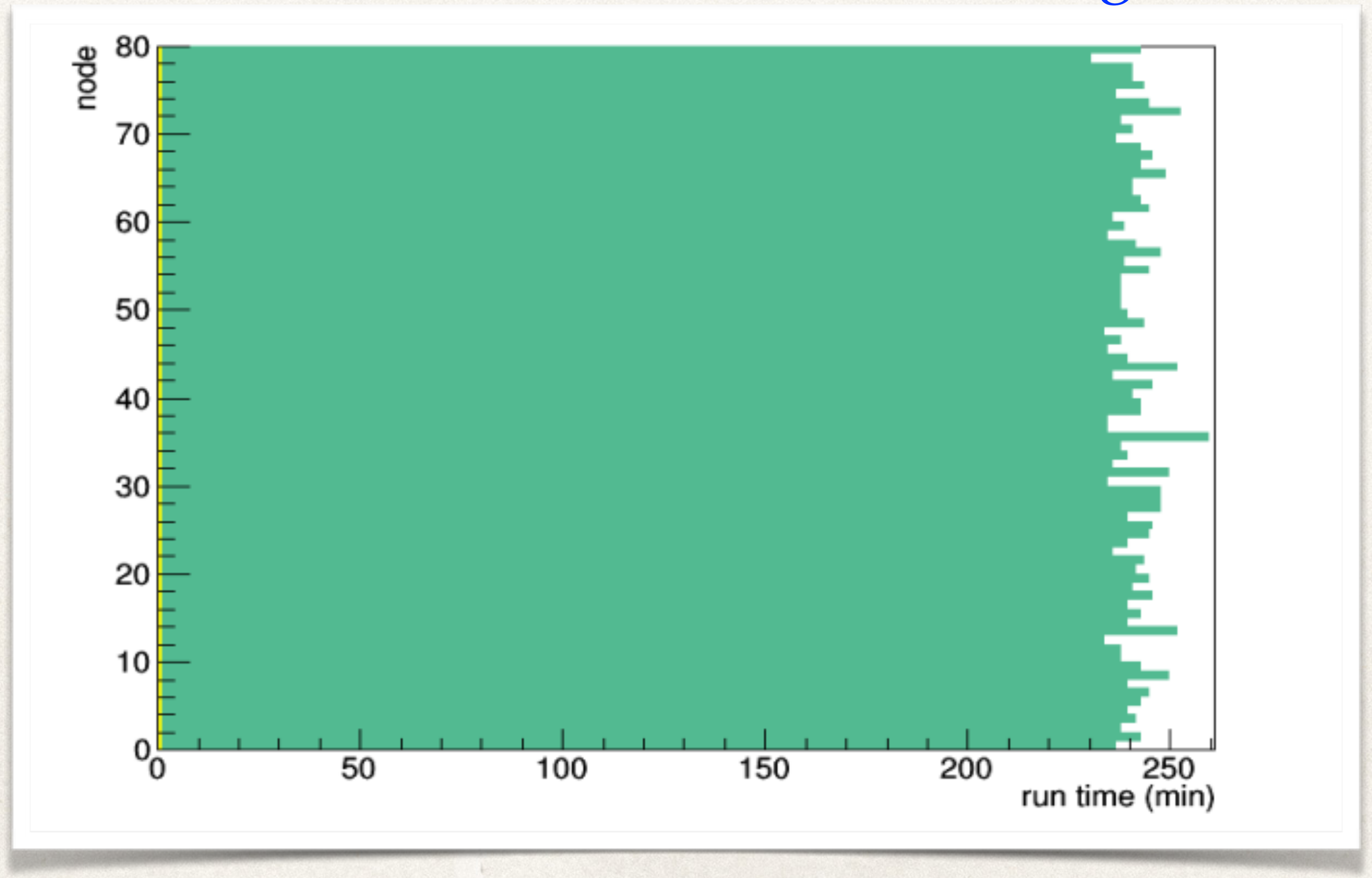
- Event Service on HPC is an MPI application
- MPI ranks in this application are lightweight versions of the conventional Event Service components
  - Yoda - mini JEDI
  - Droid - lightweight Pilot

- Each rank writes many small output files to the disk. Results in high load on the HPC shared file system
  - ➡ We plan to address this problem by implementing specialized MPI ranks for collecting outputs from other ranks and writing them to the disk ("Shared MPI Writer" processes)

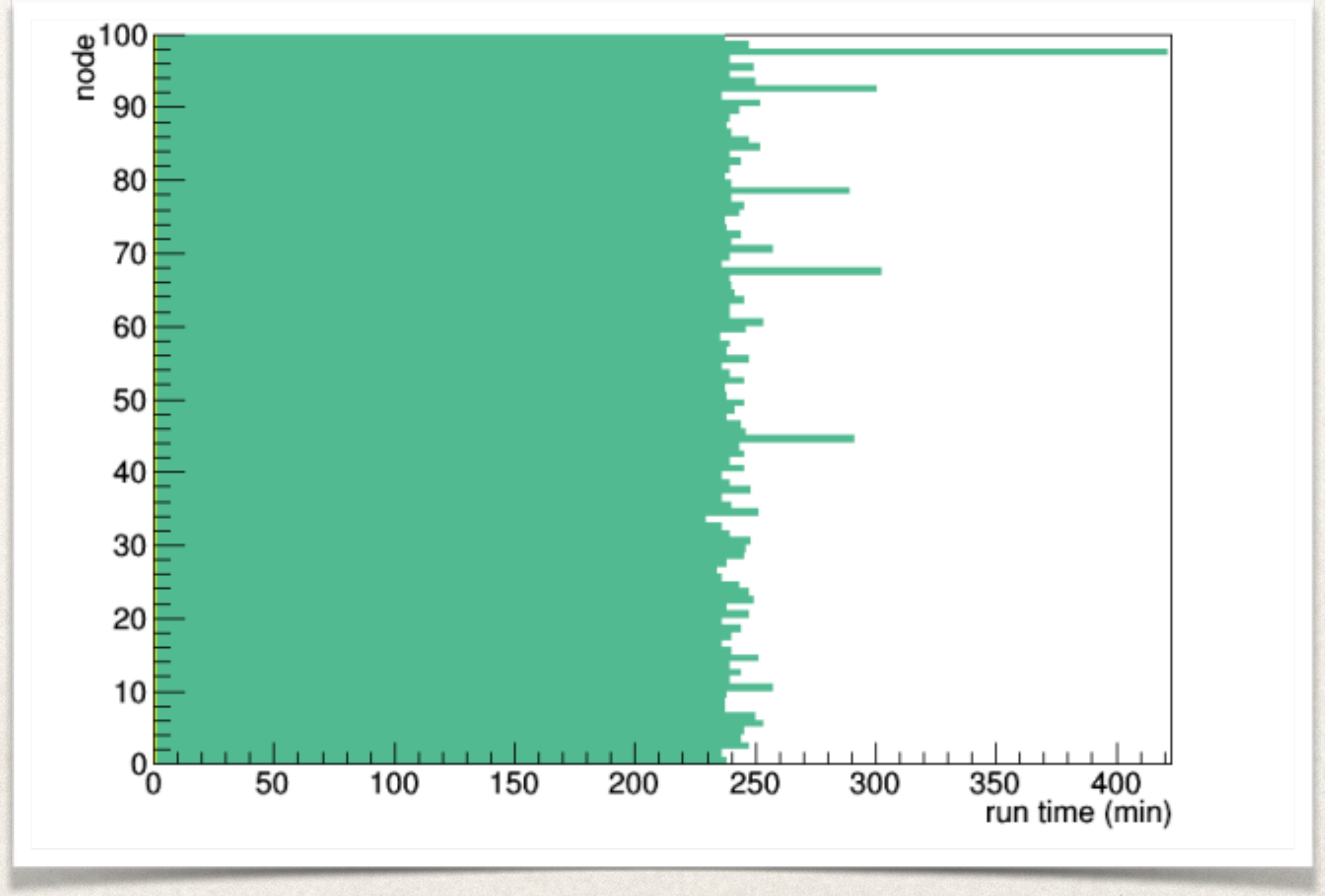
Improved resource utilization

- ATLAS is running conventional Simulation workflows on HPC by combining multiple independent instances of AthenaMP into one MPI submission
- In this approach the MPI job holds on all of its compute nodes until the slowest one is finished
  - Wasted CPU cycles at the end of the job
- The plots below show node utilizations within two such MPI jobs at NERSC (Berkeley, US)

Regular Job



Unfortunate Job



Green: node is busy

White: node is idle

Improved resource utilization (contd.)

- Yoda addresses this problem by constantly streaming input events to the compute nodes until the entire MPI job reaches its wall clock limit
- The plots below show CPU core utilizations within an HPC compute node for conventional and Yoda jobs

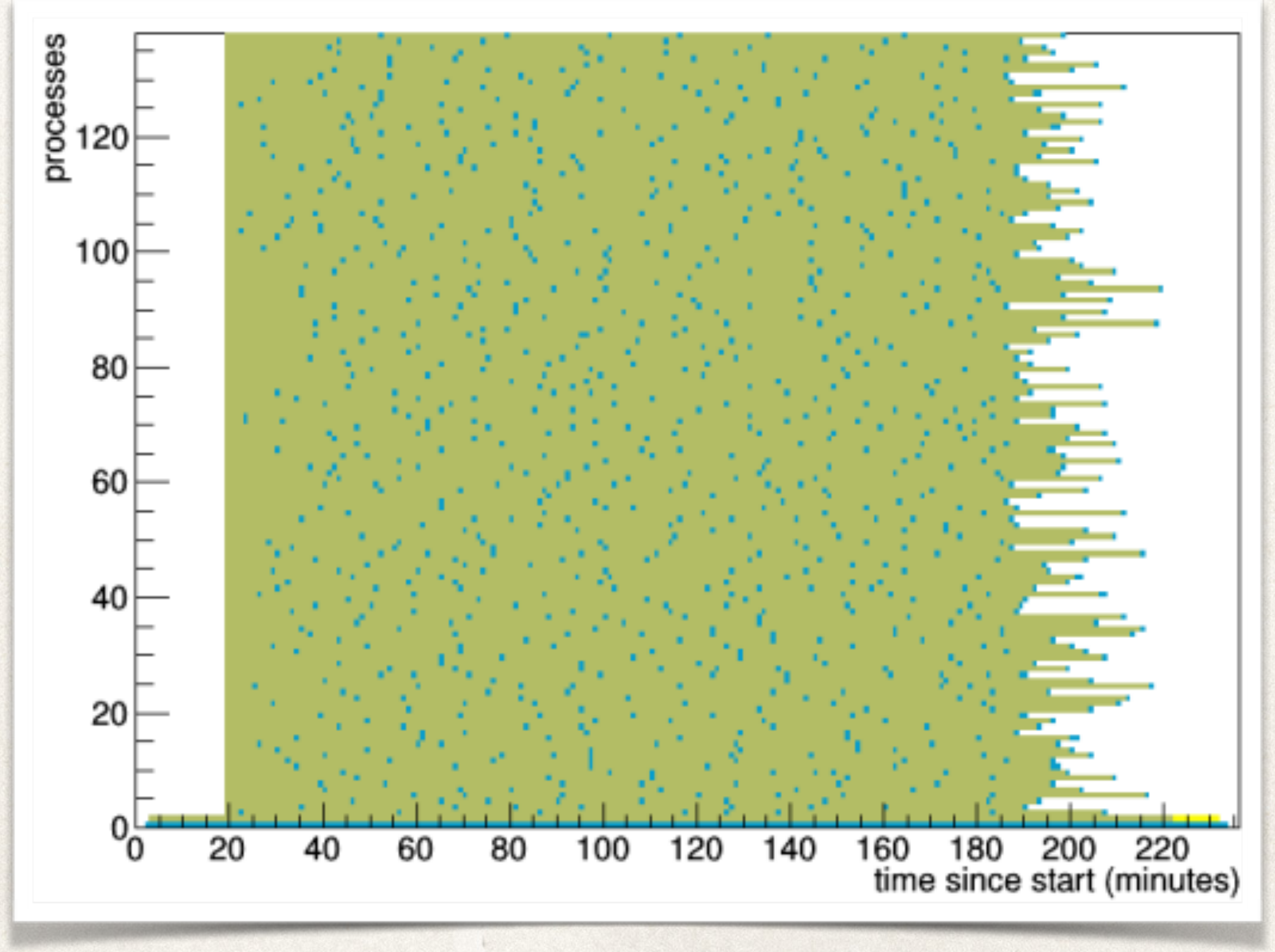
Green: core is processing an event

White: core is idle

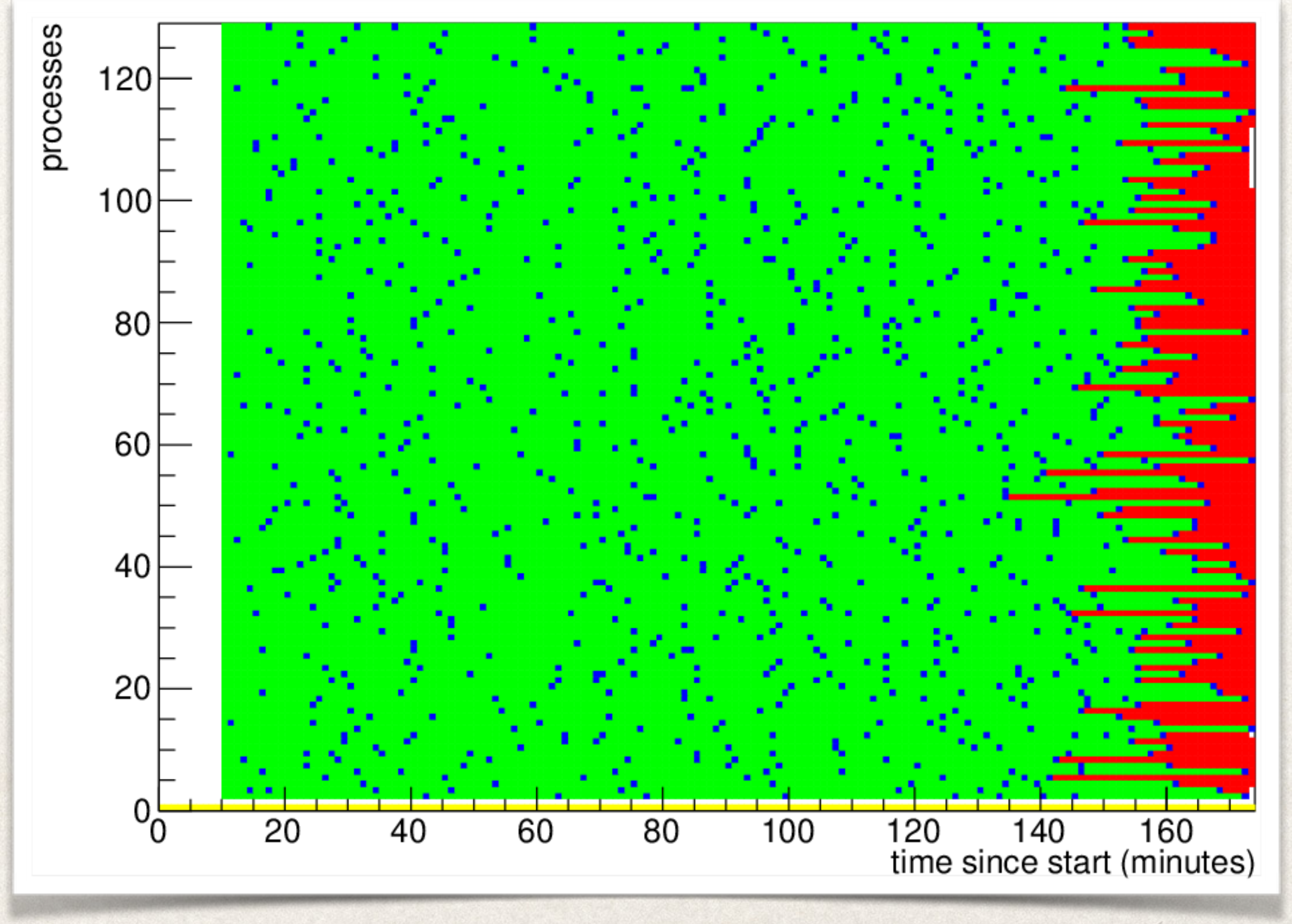
Red: the last interrupted event

Blue: core is waiting for next event

Regular Job



Yoda

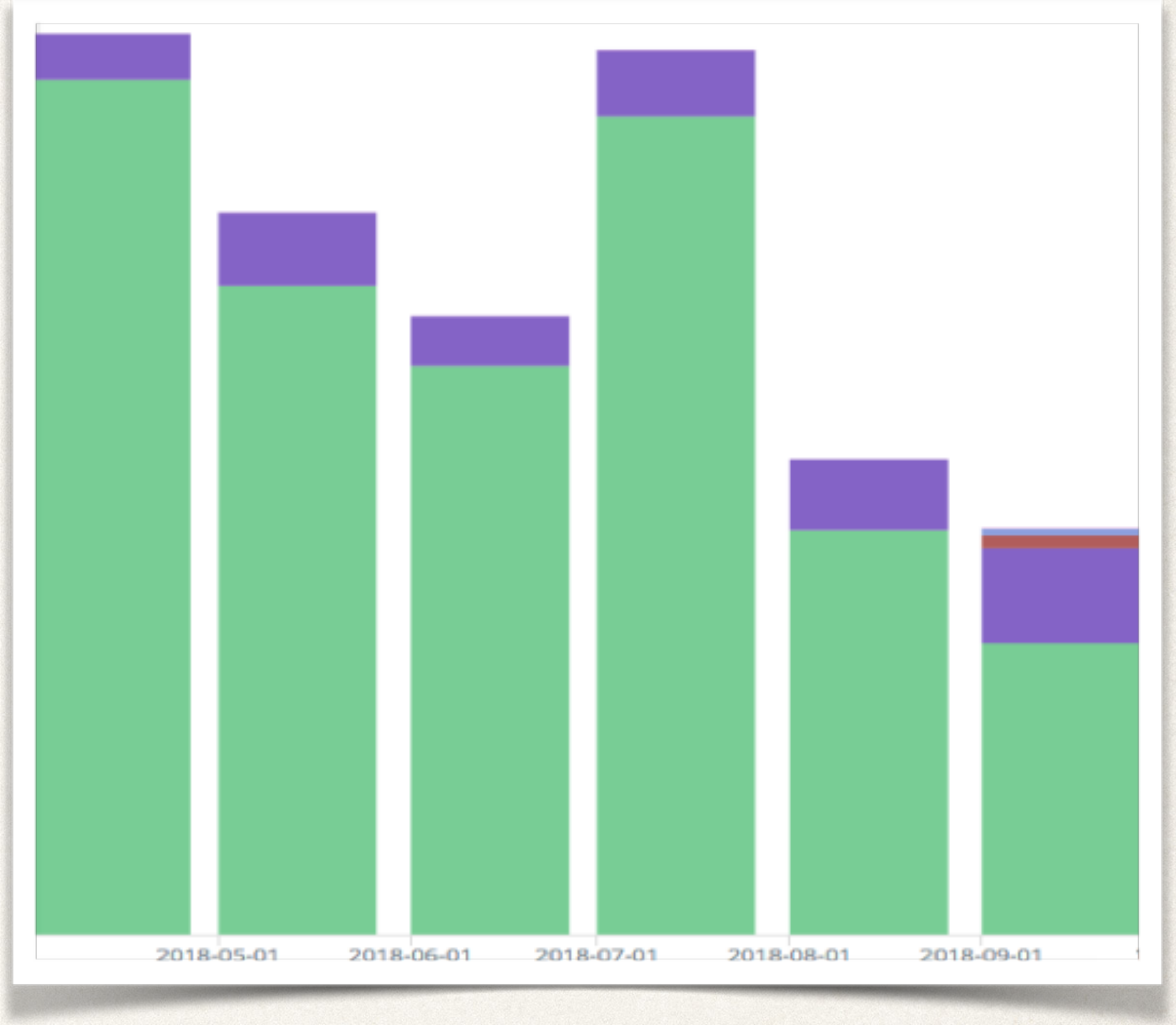


Event Service in ATLAS today

- Today ATLAS is actively using Event Service for running Geant4 Simulation production jobs

- Event Service fraction in the total delivered ATLAS Simulation walltime is increasing

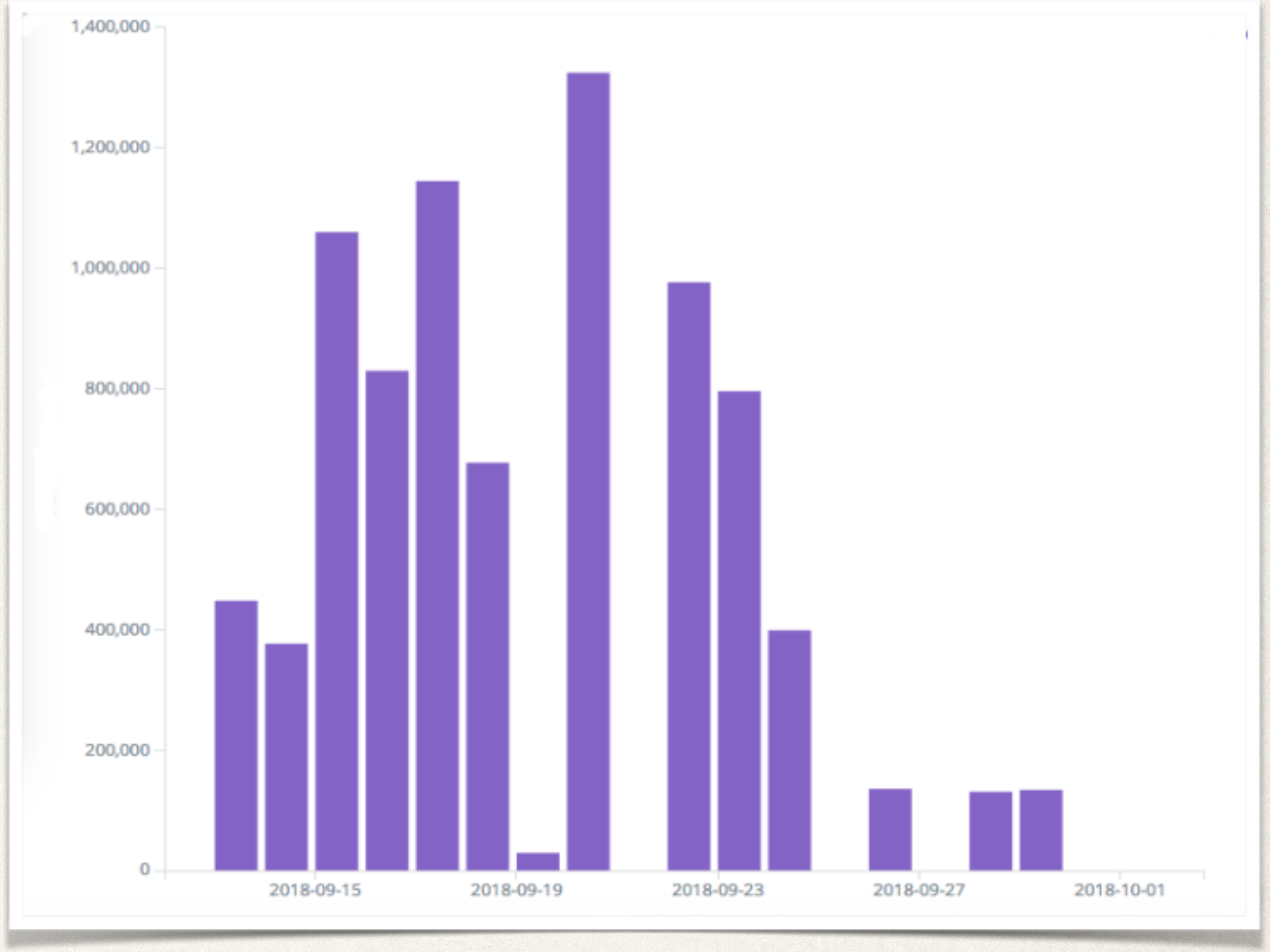
- We are currently evaluating the feasibility of running at least some part of all Simulation tasks with the Event Service for faster turnaround



Walltime delivered by the **Event Service** wrt **Regular Simulation** per calendar month

Yoda in ATLAS today

- Yoda is running in production on several supercomputers
  - Full scale production on Theta (ALCF, Argonne, US) and Cori (NERSC, Berkeley, US)
  - Titan (OLCF, Oak Ridge, US) is running Yoda in backfill mode. Working on ramping up to the full scale
- In the near future Yoda is expected to become a major contributor to the overall ATLAS Geant4 simulation production



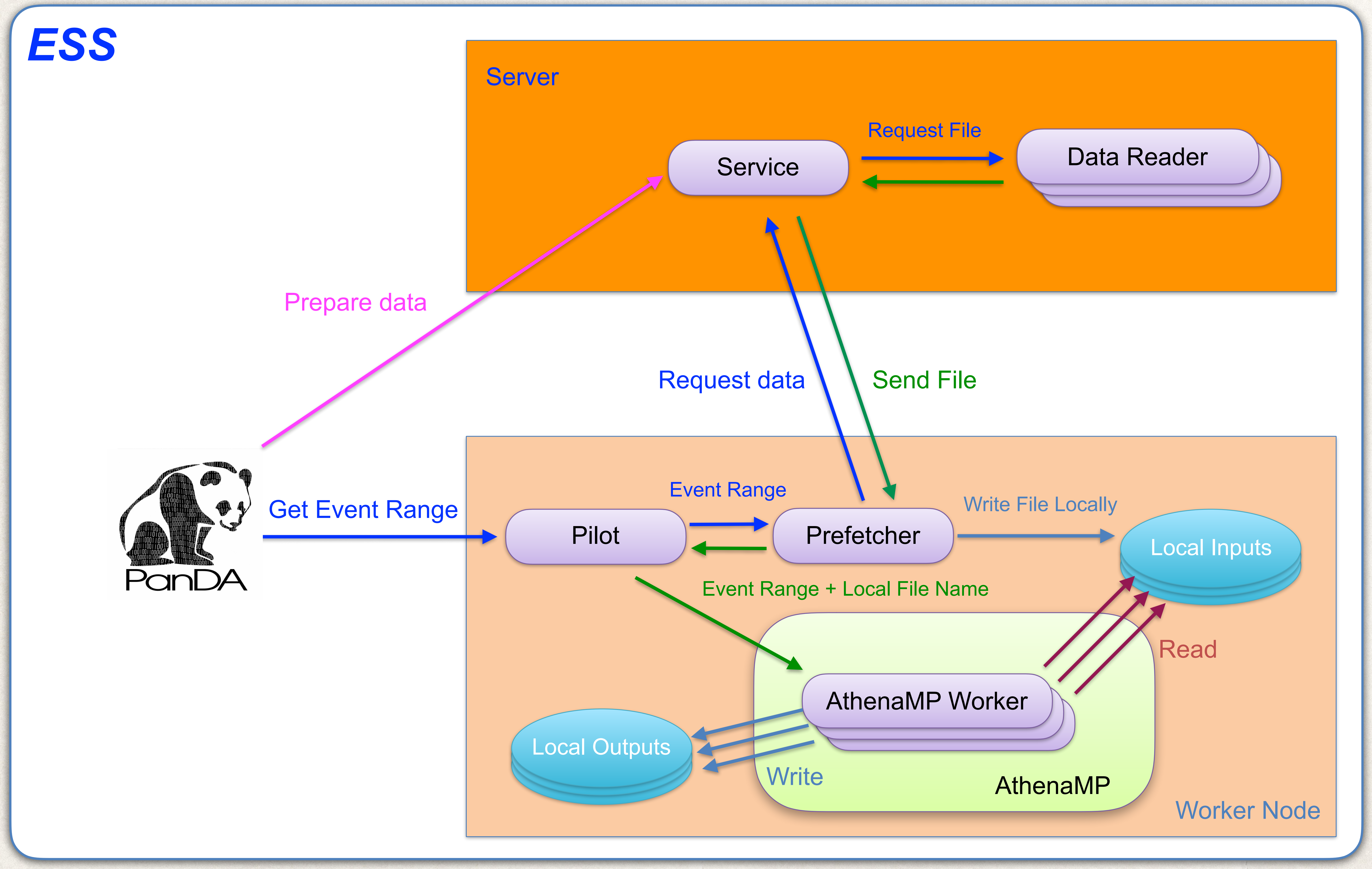
Number of events/day delivered by Yoda in second half of September 2018

From Event Service to Event Streaming Service

- The Event Service can integrate perfectly with a similarly event-level data delivery service - the Event Streaming Service - that responds to requests for "science data objects" by intelligently marshaling and sending the data needed

- The Event Streaming Service can encompass
  - Optimization of data source "close" to the client, like in Content Delivery Networks
  - Knowledge of the data itself sufficient to intelligently filter event data during marshaling
  - Servicing the request via processing on demand rather than serving preexisting data

Prototyping the Event Streaming Service



- Server component currently in the R&D phase
  - Uses knowledge available in the system for preparing required input in advance
- Asynchronous prefetching of fine grained inputs on the compute node done by a specialized process

Summary

- Event Service is our strategy for efficient utilization of the variety of computing resources, in particular supercomputers and opportunistic resources

- Flexible architecture of the Event Service/Yoda has a potential for efficient scaling to hundreds of compute nodes on modern HPC systems

- Next step in the evolution of fine-grained processing in ATLAS - the Event Streaming Service - is currently in an R&D phase