

ATLAS Simulation on HPC's Large scale fine grain simulation workflows ("Jumbo Jobs") on HPC's



DOUG BENJAMIN
HEP Division ANL

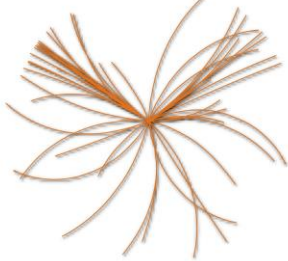
On Behalf of the ATLAS Collaboration

Acknowledgements

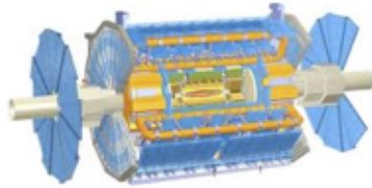
- Thanks to :
 - Miha Muškinja (LBL (US)),
 - Wen Guan (U Wisc (US)), Tadashi Maeno (BNL (US)),
 - Nicolo Magini (Iowa State U (US)), Paul Nilsson (BNL (US)),
 - Danila Oleynik (JINR (RU)), Vakho Tsulaia (LBL (US)),
 - Taylor Childers (ANL (US)), Martina Javurkova (U Mass (US)),
 - Torre Wenaus (ANL (US)), and Harvester team and HPC DevOps

ATLAS activities on LCF HPC's

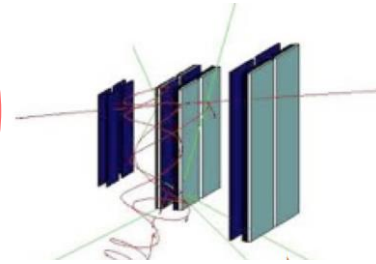
Event
Generation



Simulation



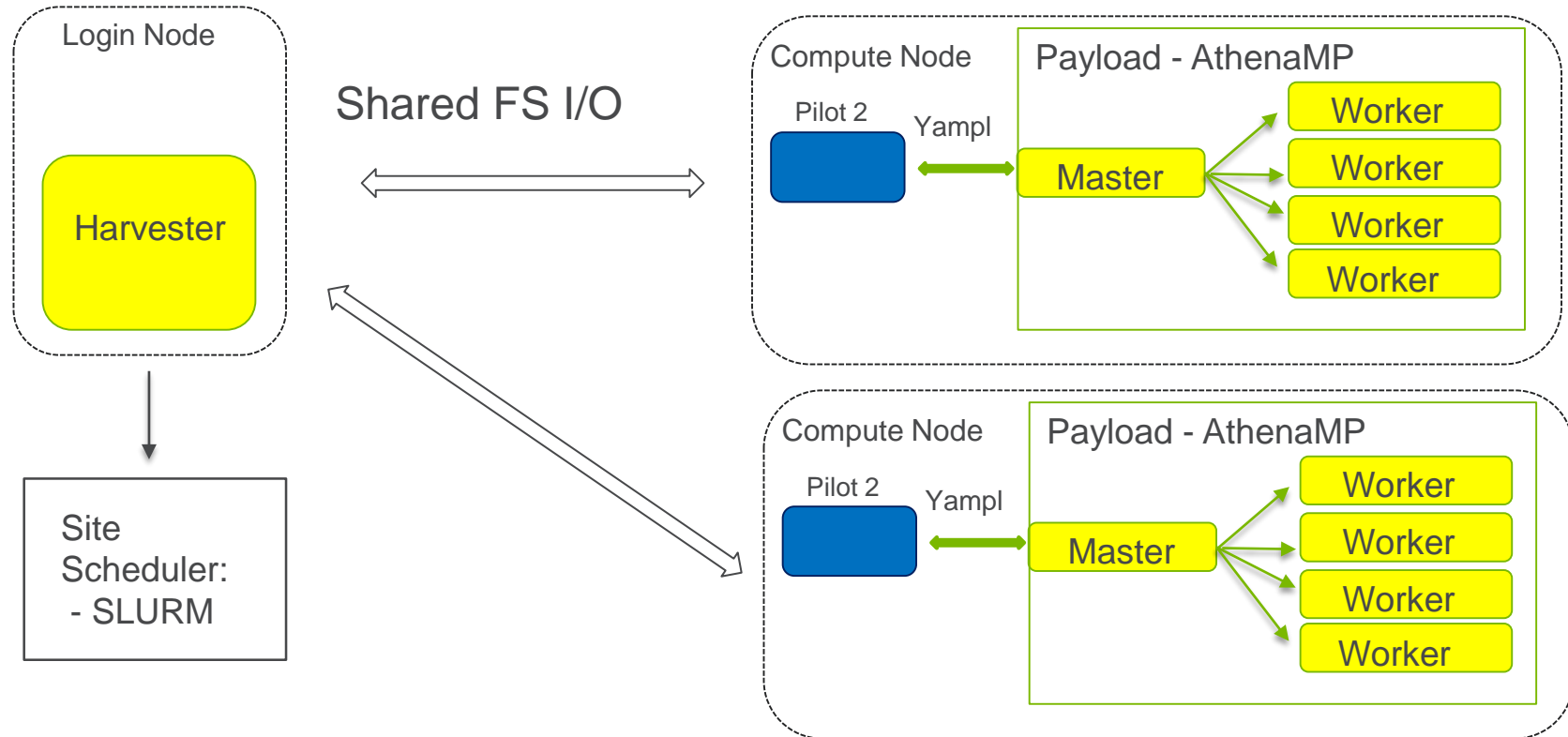
Reconstruction



Fast Chain

New way of running combining all 3.

How ATLAS is currently running Simulation today



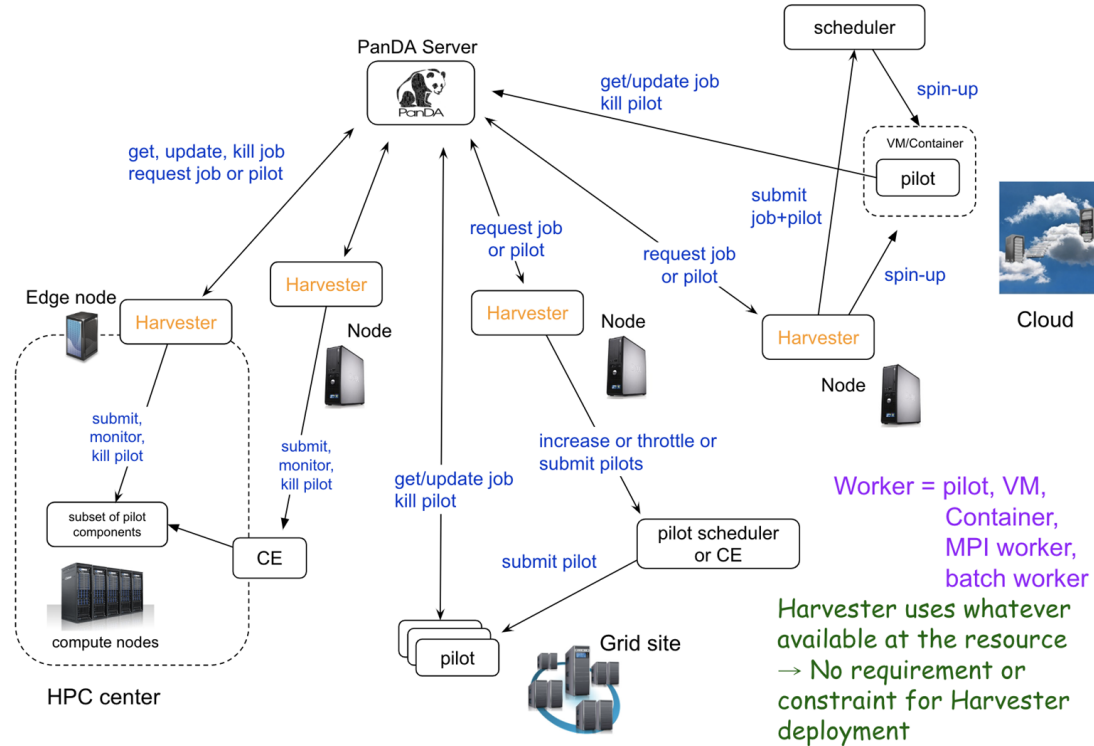
Why Jumbo Jobs? What are they?

- HPC center's WMS (SLURM, Cobalt,.....) limit the number of submission per users.
 - On grid typically one batch submission per 8 cores (or so).
 - For ATLAS means 1 pilot per job and one job per batch slot.
 - On HPC's with limited number of submissions in the queue – need multimode submissions.
 - At NERSC Harvester requests 250 KNL nodes per worker (batch submission)
- Normal ATLAS simulation jobs 1000 events - designed for 8 cores for up to 24 hrs.
- Single ATLAS simulation job for Event Service – sized to fit HPC batch submission.
 - (many 100k events)

Tools need by ATLAS

- In order to run large scale simulation (Geant 4) jobs on leadership class High Performance Computers (HPC) – ie “Jumbo Jobs” – ATLAS needs to use some tools:
 - **Harvester**
 - Service running on edge of HPC center that connects to the ATLAS Workflow Management System - PanDA
 - **ATLAS Event Service (ES)**
 - Method of running simulation – as events (1-10) are simulated they written out to individual files. Each process is instructed what input events to simulate. PanDA keeps track of the simulated events.
 - Fine grain event simulation leads to efficient use of CPU resources.
 - **Yoda/Droid**
 - Software running on compute nodes to pass information between payload – AthenaMP and Harvester/PanDA server

Harvester



Worker = pilot, VM, Container, MPI worker, batch worker

Harvester uses whatever available at the resource
→ No requirement or constraint for Harvester deployment

- ❏ Mediates interactions between a resource and workload/data management
- ❏ Encapsulates resource heterogeneity, presenting uniform interface to workload management and monitoring
- ❏ Incorporates via plugins the particular behaviour and functionality needed by a resource
- ❏ Particularly suited to complex HPCs but also used across the grid
- ❏ Experiment agnostic, with first usage beyond ATLAS recently reported (ASGC)

ATLAS Event Service (ES)

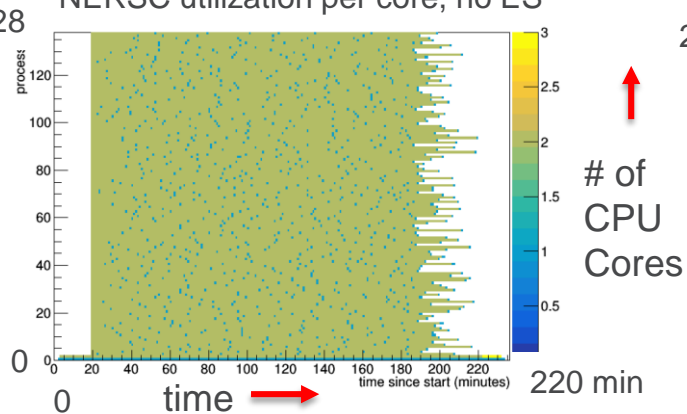
Without event service, each core processes N events. Once a core has finished its allocation, it idles (white)

With event service, each core is allocated events to process until the scheduler slot ends.

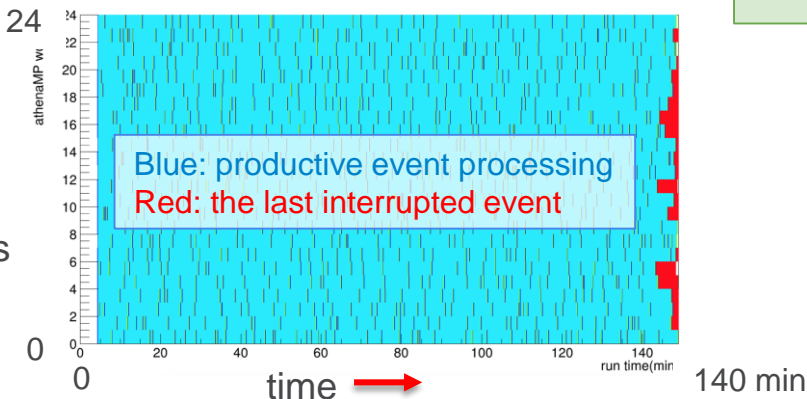
If the job is suddenly killed by preemption, all processed events are preserved except the last few minutes (all are lost in a conventional job)

Efficient
processor and
memory
utilization
In ATLAS
production for
several years

NERSC utilization per core, no ES

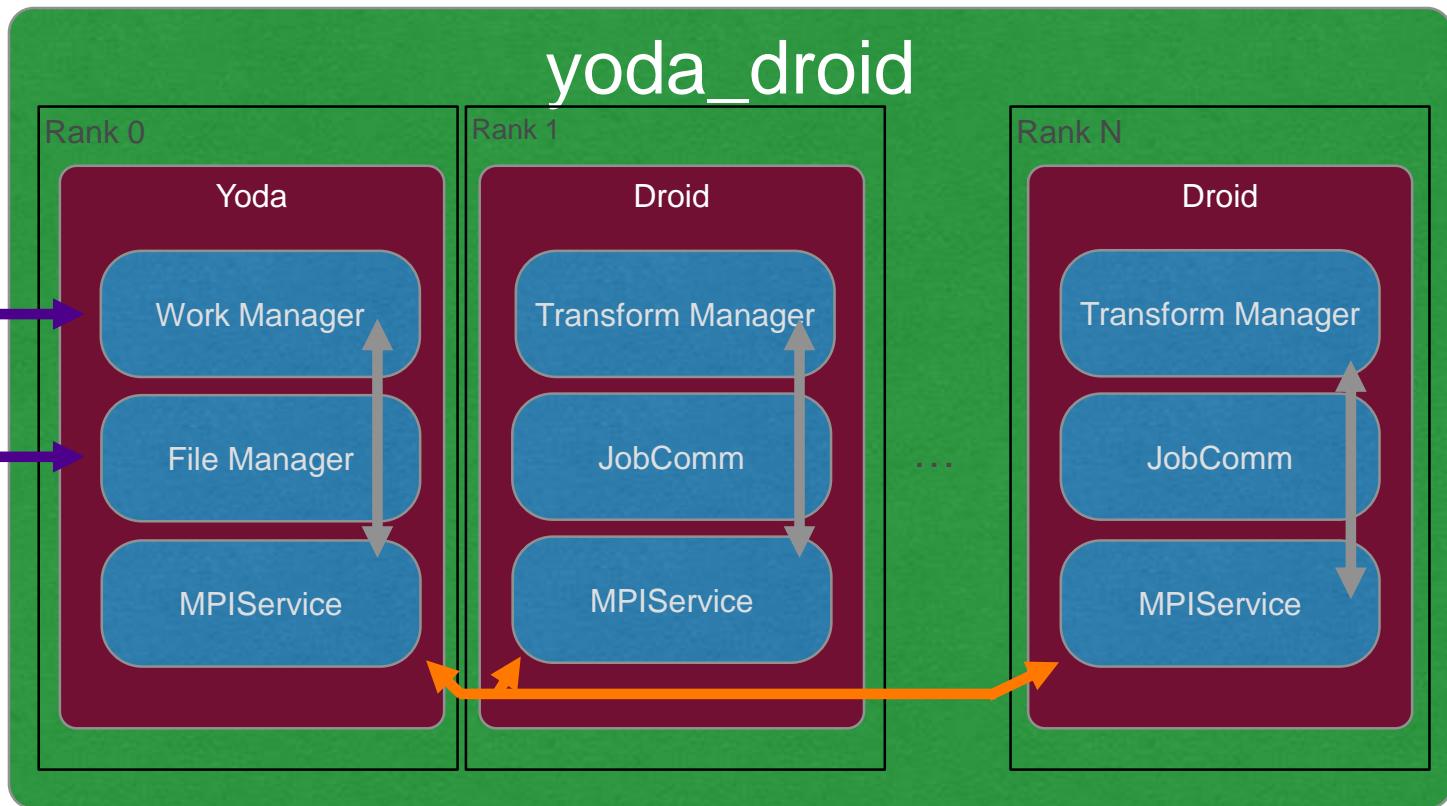


NERSC utilization per core with ES



ES elastically fills all cores for the full job lifetime

Yoda



Yoda
developed
by Taylor
Childers
ANL/ALCF

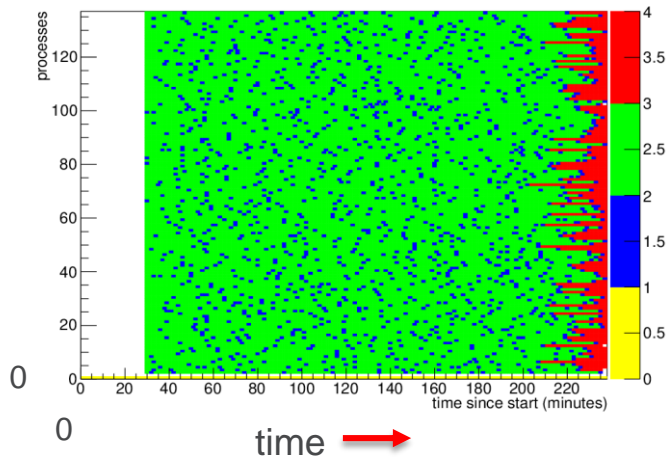
Shared File System →

Queue message →

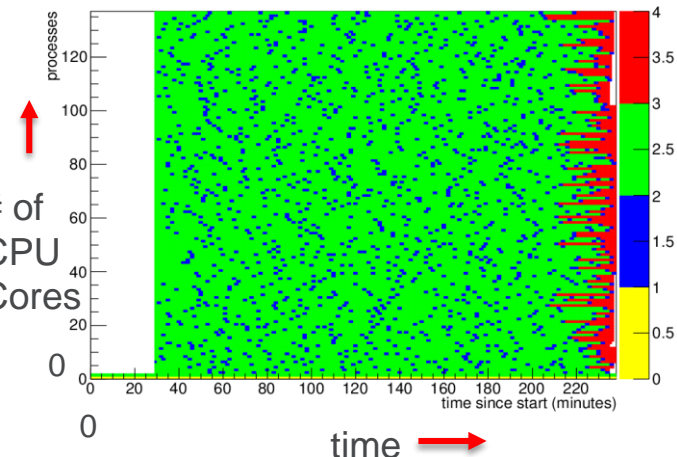
MPI Communication →

NERSC - Cori – Geant 4 Simulation

Efficient
Running



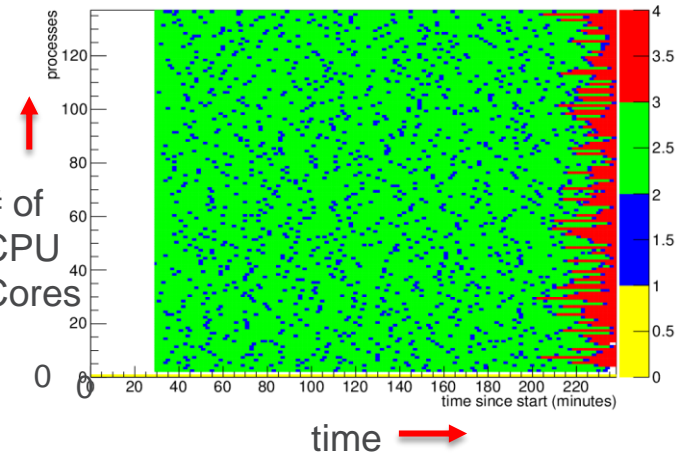
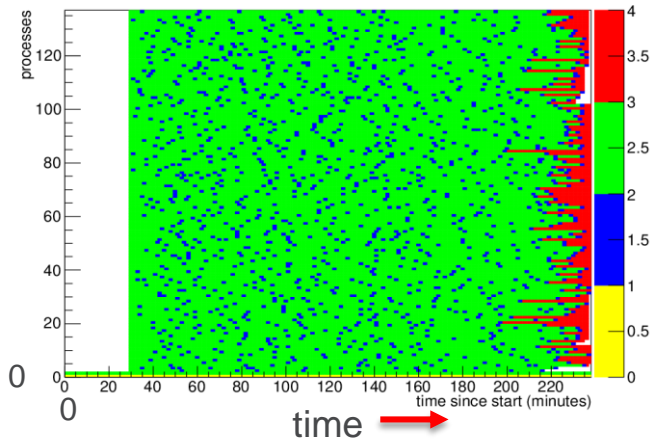
00247



00248

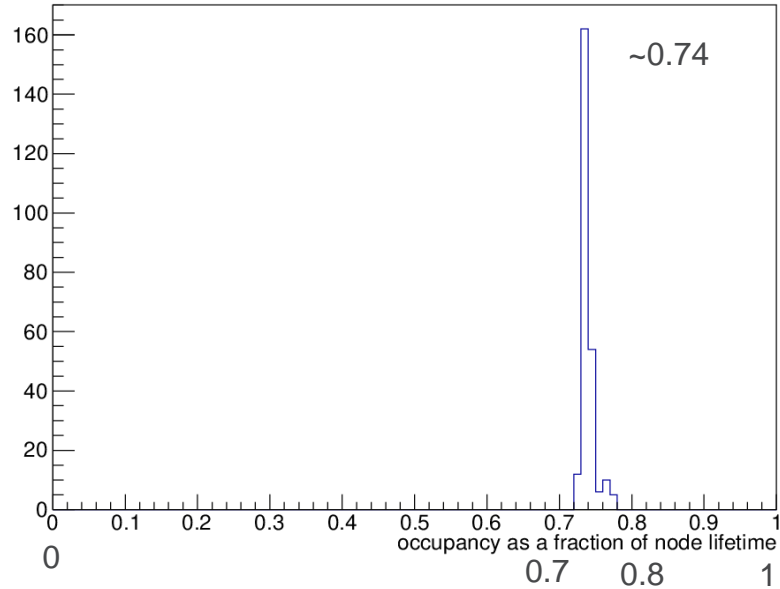
4 Nodes in
typical
Jumbo job

250 nodes
34k Cores

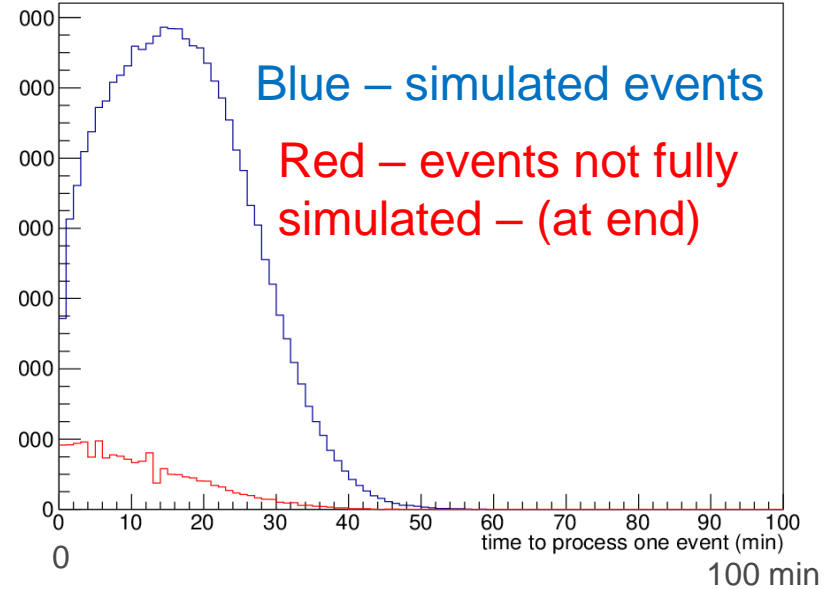


NERSC- CORI KNL nodes

Fraction of Time spent simulating events

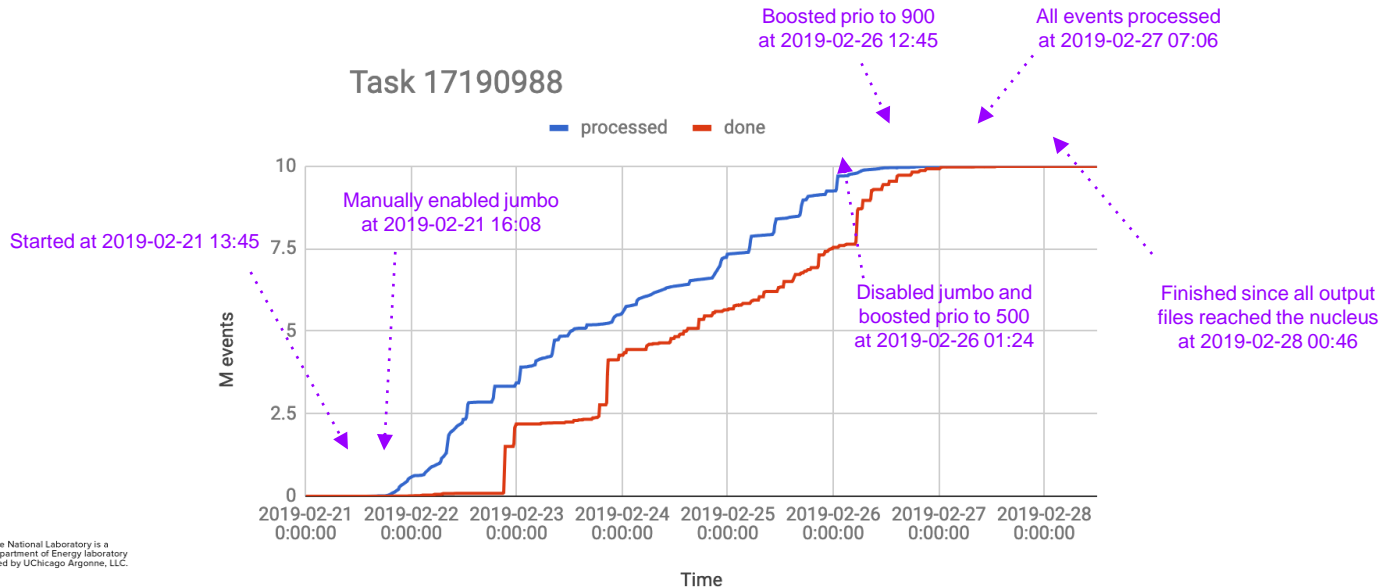
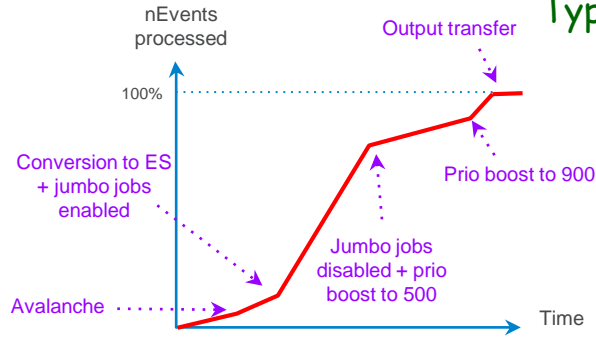


Time spent simulating 1 event

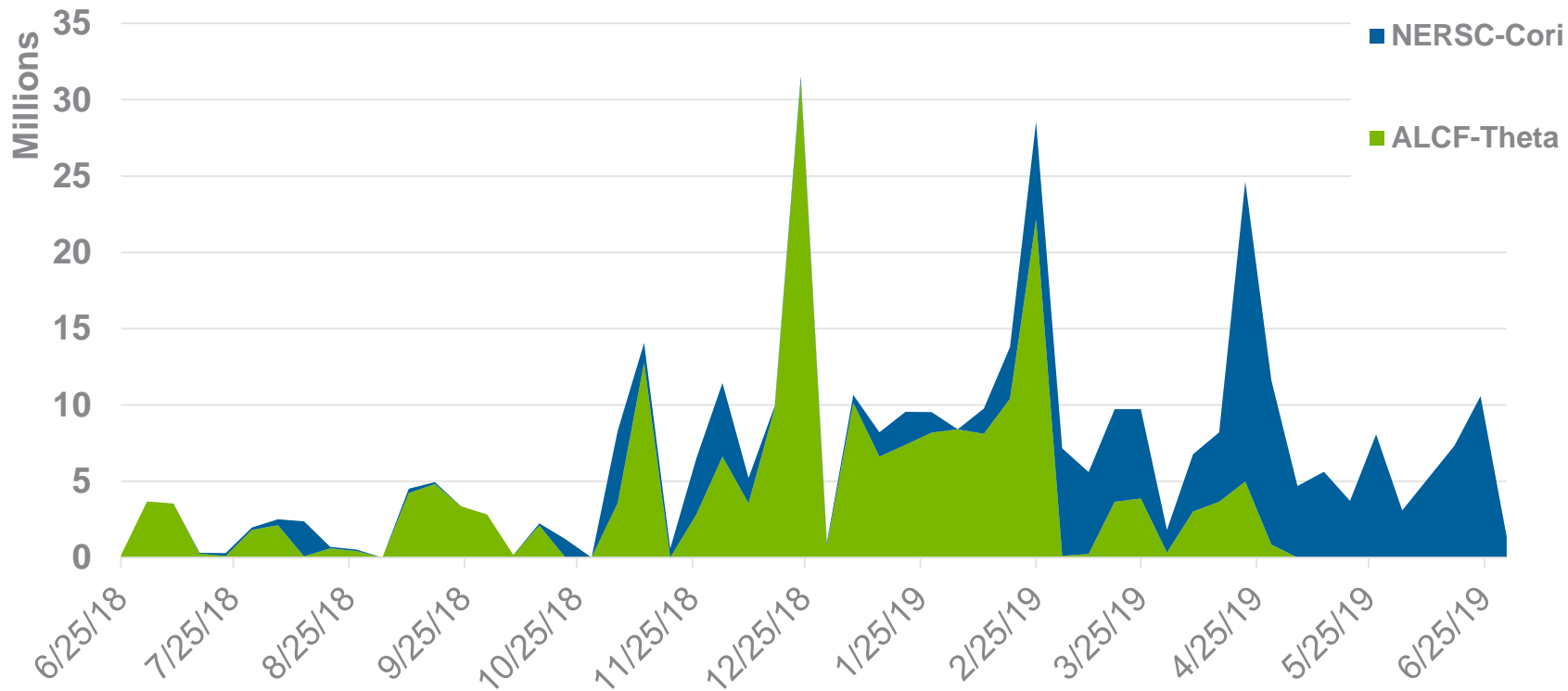


PanDA actions for Jumbo Jobs

Typical event processing with Jumbo jobs



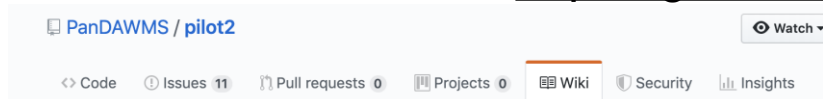
Number of Events Simulated by Week



Next Steps – integration of Pilot 2

- Pilot 2 – the PanDA Pilot has been modernized and rewritten

<https://github.com/PanDAWMS/pilot2>



Pilot workflows

Paul Nilsson edited this page on Feb 19 · 1 revision

The Pilot workflows refer to the particular run mode that the Pilot should use. This includes the Standard workflow and HPC Pilot workflow. The desired workflow is selected by the wrapper script that launches the Pilot, using option `-w workflow` (e.g. `generic_hpc`; default: `generic`) and it determines how the jobs will be processed by selecting the corresponding Pilot module. For the HPC Pilot workflow, the HPC resource name needs to be specified using option `--hpc-resource resource_name` (e.g. `titan`).

The Standard and HPC Pilot workflows are explained in the following sections.

Talk given at this meeting
(Tuesday - 5 – Nov)

Paul Nilsson

<https://indi.to/6jVsf>

HPC workflow

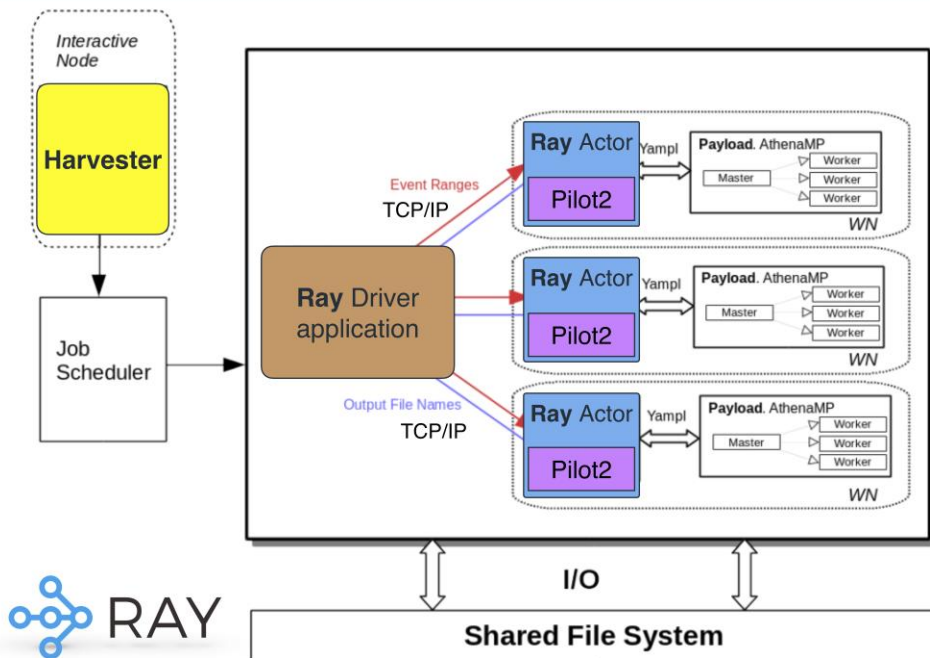
Paul Nilsson edited this page on Jan 25 · 13 revisions

HPC workflow in Pilot 2

The Pilot 2 HPC workflow is a special mode where the application works without a remote connection to PanDA server or other remote facilities. All intercommunications in this case are managed by the Harvester application. Also, in this mode Pilot 2 acts like a simple MPI application, which performs execution of multiple jobs on the computing nodes of the HPC.

Next Generation Event Service on HPC

Raythena workflow



Ray takes care of all communication between nodes and orchestration of workload, the Event Service application becomes more manageable and more modular.



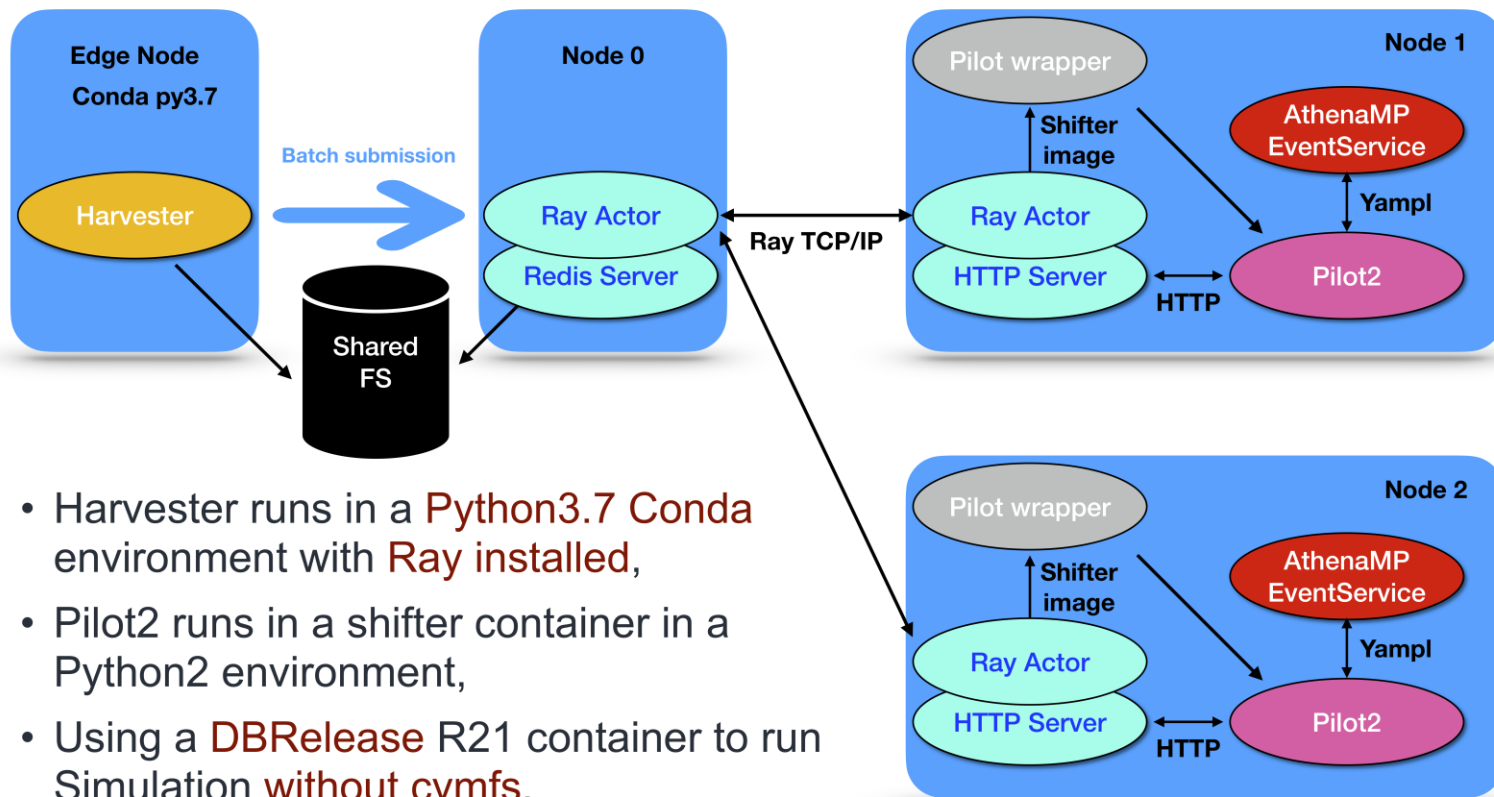
16 April 2020

Miha Muškinja

3

Next Generation Event Service on HPC

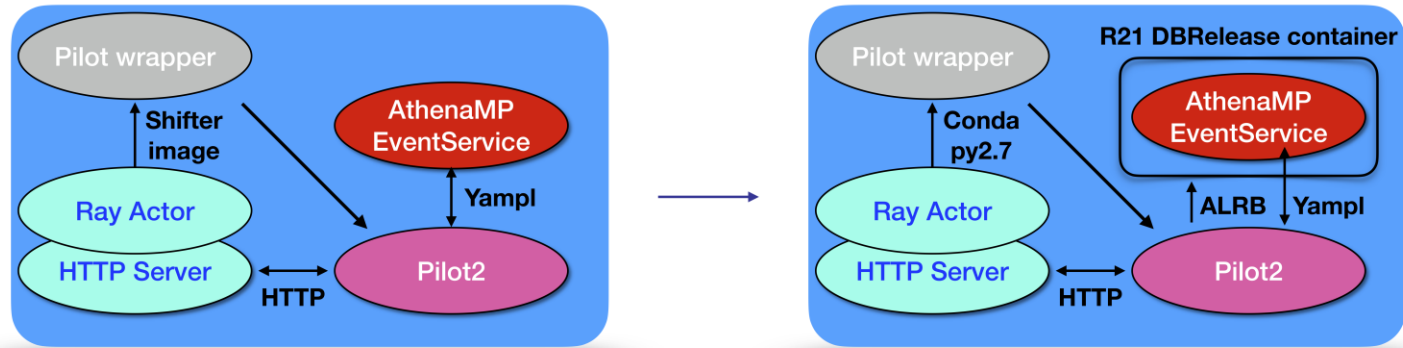
Detailed Raythena Scheme on Cori KNL



- Harvester runs in a **Python3.7 Conda** environment with **Ray** installed,
- Pilot2 runs in a shifter container in a Python2 environment,
- Using a **DBRelease R21** container to run Simulation **without cvmfs**.

Next Generation Event Service on HPC

Update to have pilot launch the container



- With Doug and Paul working on an update where **Pilot2 starts the container with ALRB** instead of running in container itself,
- For Raythena the change is ~trivial, instead of running Pilot2 in a container, we only setup a python2.7 environment for it,
- Container name defined through Panda task parameters.