Production Experience with the ATLAS Event Service

D Benjamin, P Calafiura, T Childers, K De, W Guan, T Maeno, P Nilsson, V Tsulaia, P Van Gemmeren and T Wenaus

For the ATLAS Collaboration

CHEP 2016, San Francisco, USA
October 10-14, 2016

Event Service. Concept

- Presented at CHEP 2015 in Okinawa

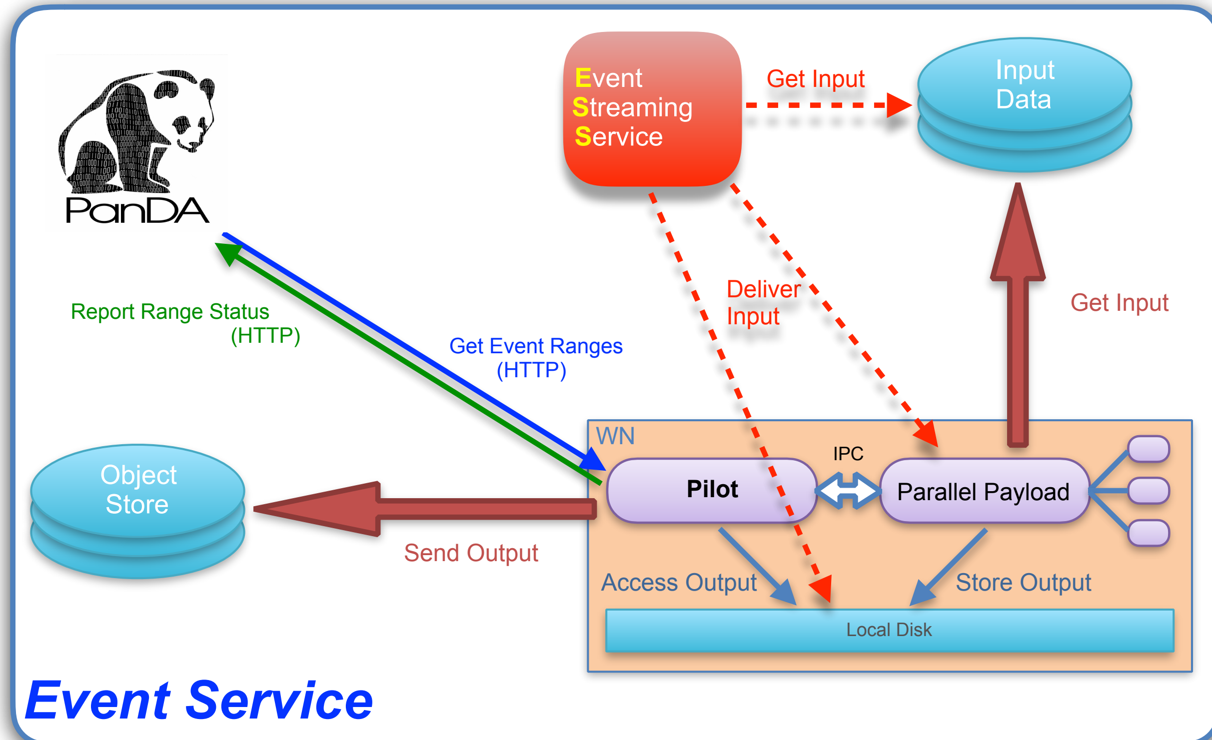
- A fine-grained approach to event processing. Designed for exploiting diverse, distributed and potentially short-lived resources
  - ✓ Quasi-continuous event streaming through worker nodes

- Exploit event processors fully and efficiently through their lifetime
  - ✓ Real-time delivery of fine-grained workloads to running application
  - ✓ Be robust against disappearance of compute node on short notice

- Decouple processing from chunkiness of files, from data locality considerations and from WAN latency

- Stream outputs away quickly
  - ✓ Negligible losses if the worker node vanishes
  - ✓ Minimal demands for the local storage

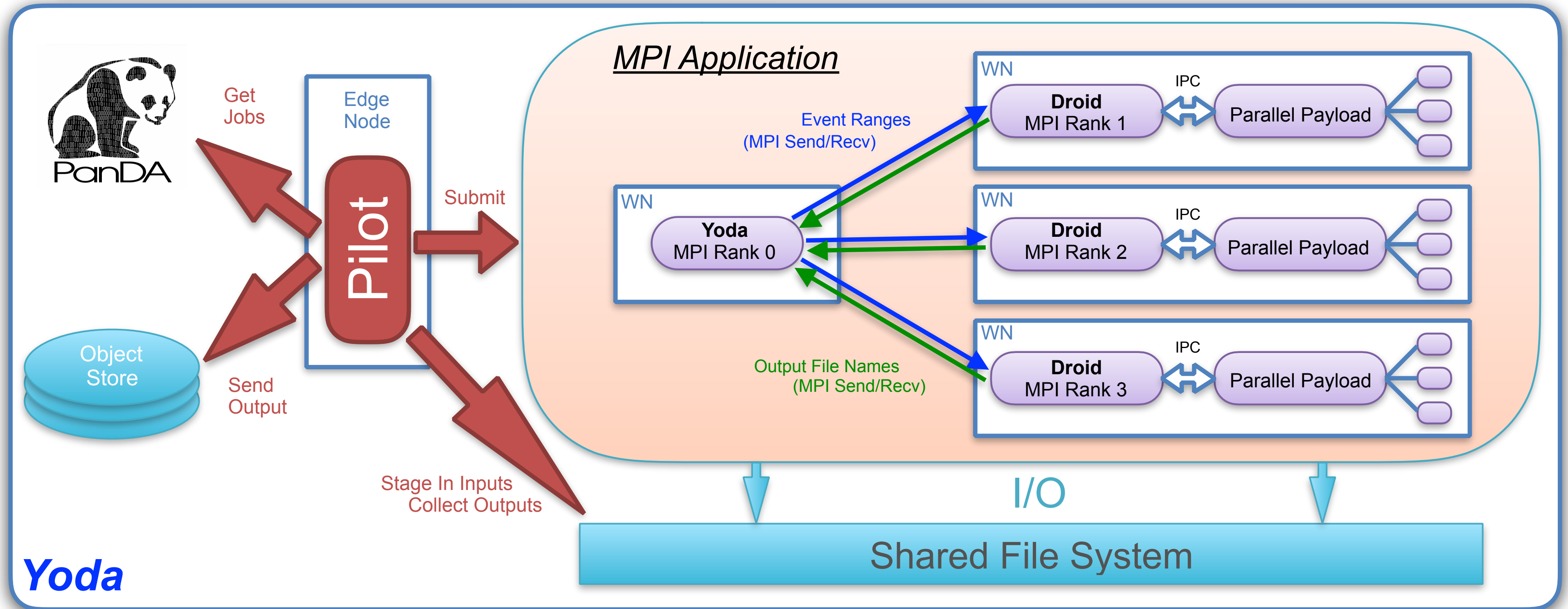
Event Service. Schematic



- Pilot delivers fine-grained workloads to the running payload application in real time
  - ✓ Workload: **Event Ranges**
- Payload application: **process-parallel version of Athena (AthenaMP)**
  - ✓ Serial initialization in the master process
  - ✓ Then fork worker processes
  - ✓ Workers process the events

- Payload directly reads input files for the event data (either local or remote file access)
- Payload uses **Output File Sequencer** for writing intermediate outputs (one per range), which are sent to **Object Stores**
- **Missing Component: Event Streaming Service.** Intelligent asynchronous delivery of the input data to the worker nodes
  - ✓ Presently in early design/prototyping phase

Yoda - Event Service on Supercomputers

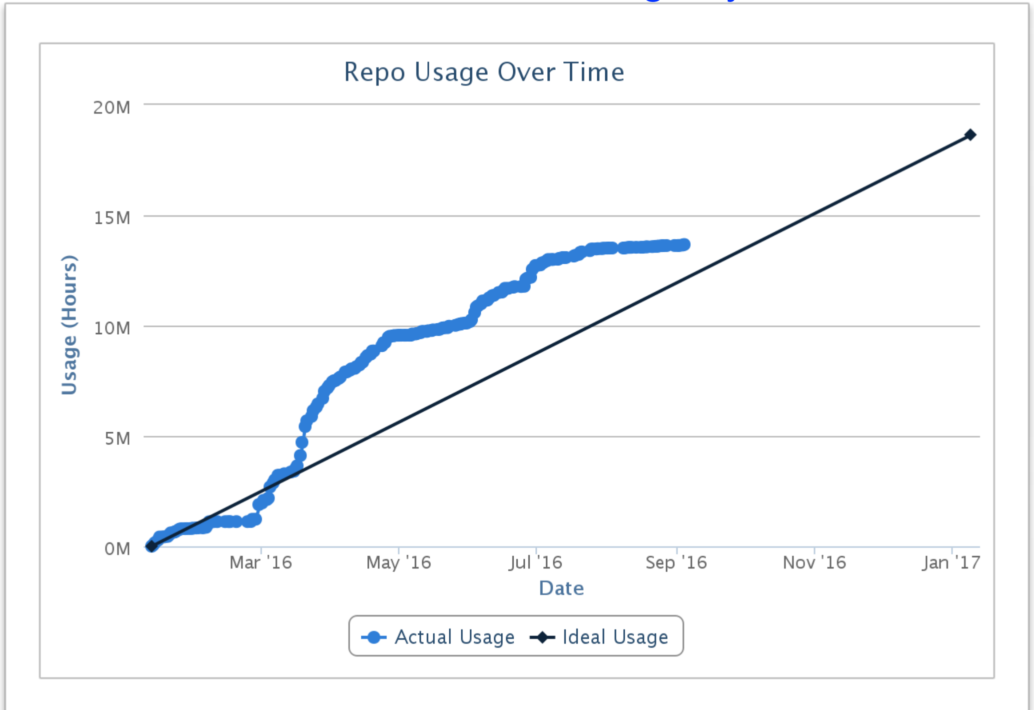


- Lightweight versions of the conventional Event Service components
  - ✓ Yoda - mini **JEDI** (**J**ob **E**xecution and **D**efinition **I**nterface)
- Yoda components communicate with each other over MPI
  - ✓ As opposed to the HTTP-based communication implemented in the conventional Event Service

Commissioning and running in production

- First use-case for the Event Service: ATLAS detector simulation with Geant4
- The supercomputers at **NERSC** (**N**ational **E**nergy **R**esearch **S**cientific **C**omputer Center, LBNL, USA) have been the main platform for the commissioning of the Event Service and for running production workloads
  - ✓ Commissioning activity on the Grid is well underway
- Since late 2015 Yoda has been running ATLAS Simulation production on Edison HPC at NERSC
  - ✓ 14M CPU-hours delivered to ATLAS in 2016

NERSC CPU time allocation usage by Yoda in 2016



CPU efficiency

- During the commissioning phase of Yoda we studied various factors which can have a visible effect on the efficient usage of CPU resources of the compute nodes

- Such factors include

  1. Initialization time of the payload application

  2. Sequential running of several instances of the payload application on a compute node during one MPI submission

  3. Handling of fine-grained outputs produced by the payload application

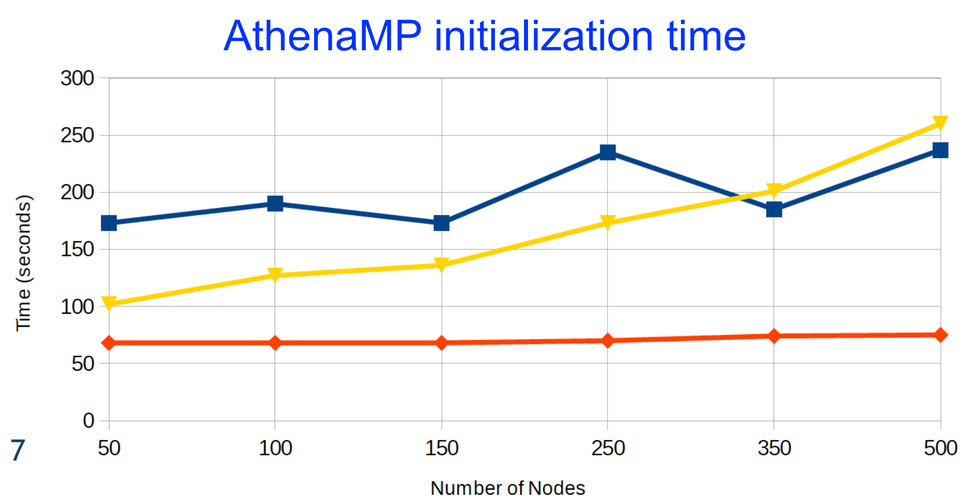
Payload initialization

- The Event Service payload (AthenaMP) reads large number of files from the disk during the initialization step

- Concurrent reading of software installation from the HPC shared file system can lead to a serious performance bottleneck when running on many compute nodes simultaneously

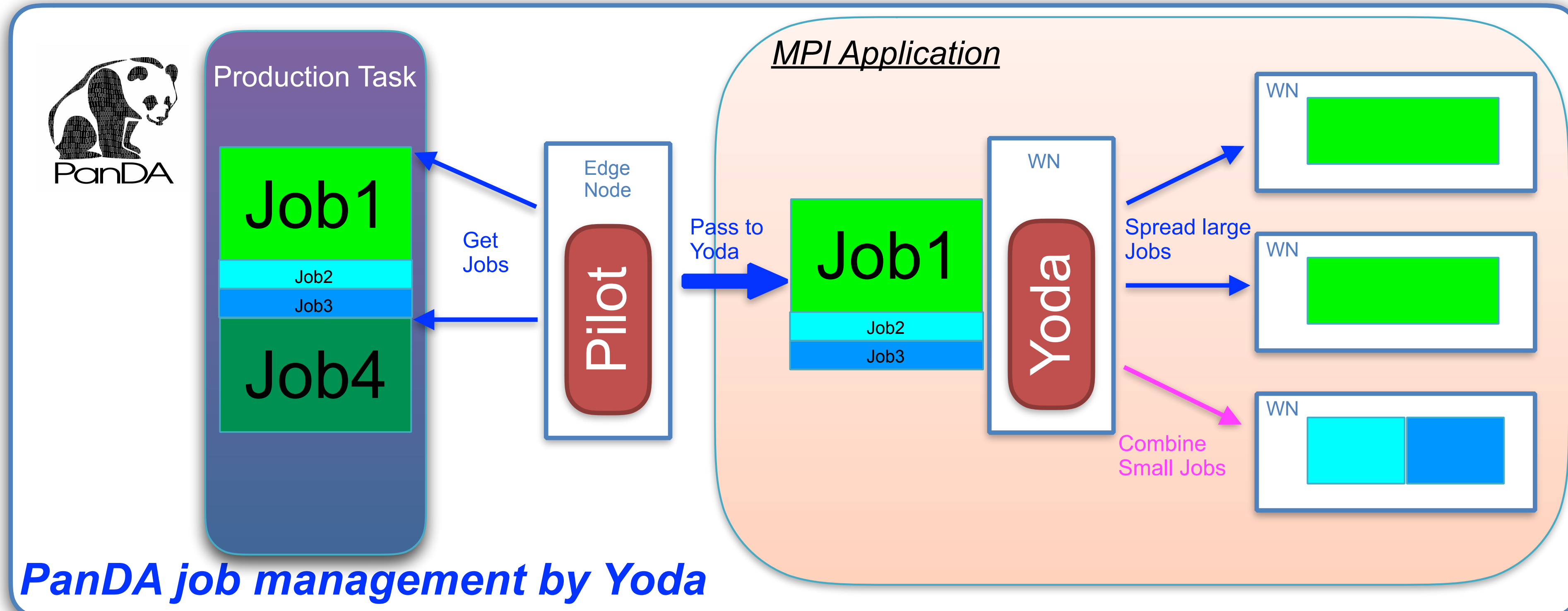
- Solution currently used in production: copy software release into the memory of compute nodes

- On **Cori Supercomputer** at NERSC we also studied the scaling of AthenaMP initialization when installing software releases on different file systems
  - ✓ **Lustre**
  - ✓ **Cori Burst Buffer**
    - See the talk by W Bhimji at CHEP2016
  - ✓ **Shifter**
    - See the talk by L Gerhardt at CHEP2016



PanDA jobs vs MPI jobs

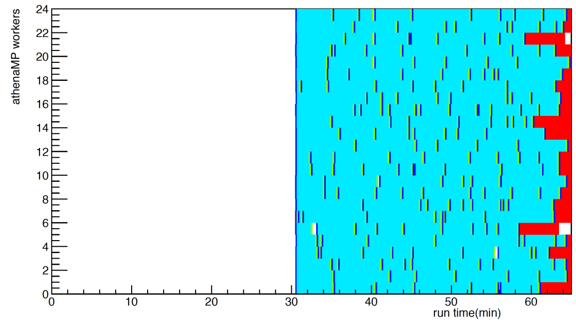
- PanDA jobs are building blocks of PanDA production tasks
  - Thousands of jobs per task
- Yoda combines multiple PanDA jobs into single MPI submission
- If Yoda fails to process all events from some PanDA job during MPI allocation time, then PanDA **generates new job for the leftover events**
  - Hence different number of events in PanDA jobs in the Event Service tasks



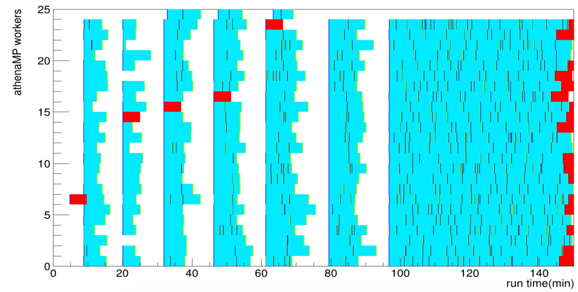
CPU efficiency of HPC compute nodes

- Examples of Yoda compute nodes with different CPU efficiency
  - ✓ **Edison Supercomputer** at NERSC, 24-core nodes

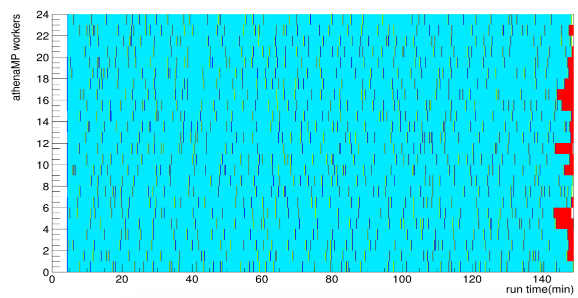
- Example 1. Poor efficiency
  - ✓ One PanDA job …
  - ✓ But very slow initialization



- Example 2. Poor efficiency
  - ✓ Several PanDA jobs on one node



- Example 3. Good efficiency
  - ✓ One PanDA job
  - ✓ Fast initialization



Legend. **White space**: core is idle
Turquoise: core processing an event
Red: event processing started but not finished

V Tsulaia et al, ATLAS, CHEP 2016

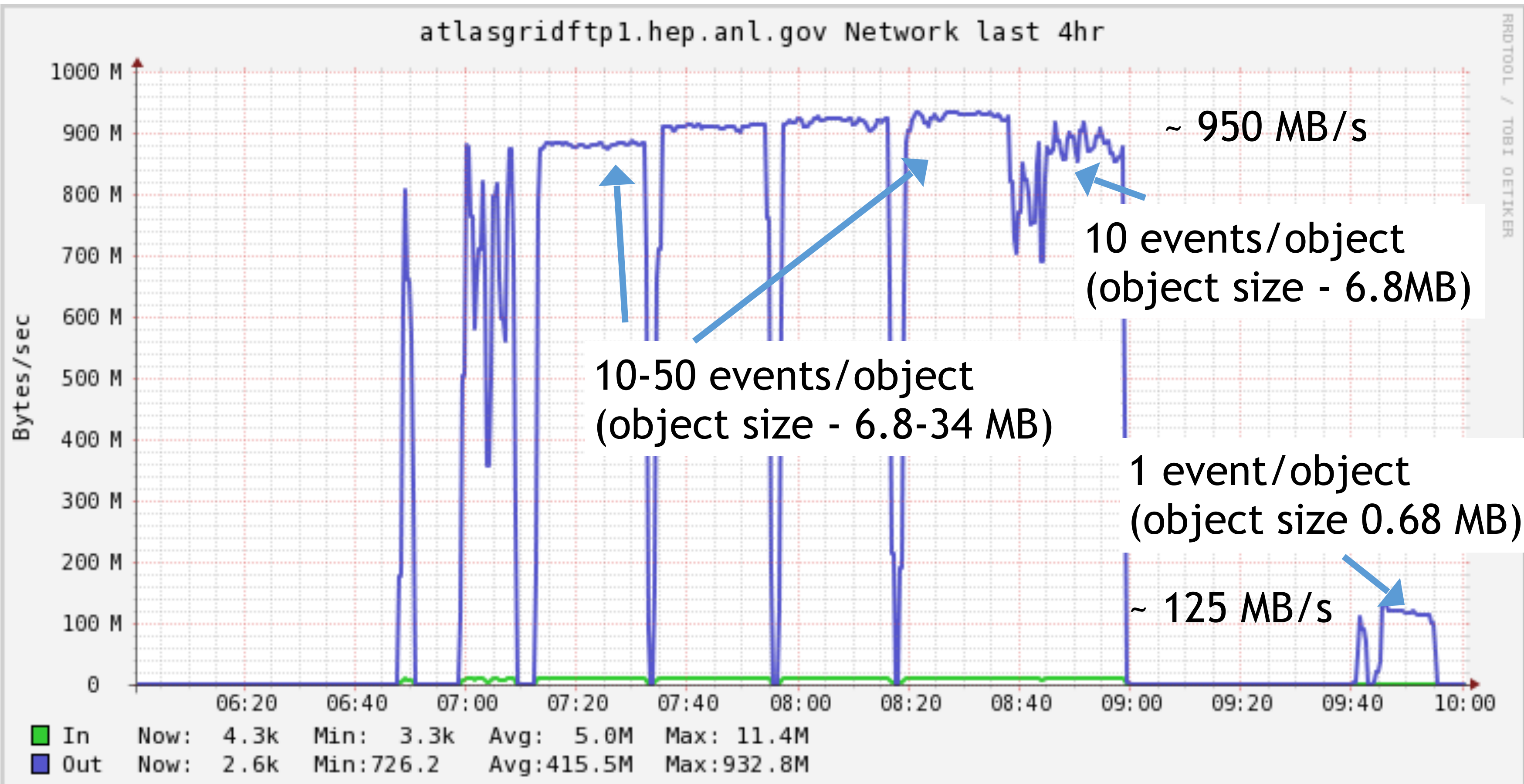
Handling of fine-grained outputs using Object Stores

- The Event Service Payload creates intermediate outputs, which are sent to Object Stores (OS)
  - ✓ Final outputs are produced later by specialized merge jobs
  - ✓ Yoda currently uses OS at BNL

- As part of Yoda commissioning at NERSC we studied the OS performance. Some observations/conclusions:
  - ✓ CEPH OS has no queuing or protection from overloading
  - ✓ When the clients overload CEPH OS various errors can occur
    - Authentication errors
    - Inability to connect to bucket
    - Inability to write object
    - Longer running writes
  - ✓ Client software must have retry and perhaps queuing capabilities. Otherwise we should use a system that can regulate the OS writes (like FTS)

OS Performance. Bandwidth vs Object Size

- Achieved ~7.2GiB/sec writing speed from ANL to BNL

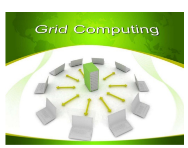


Lessons learned

- Primary causes of sub-optimal usage of HPC compute nodes by Yoda:
  - ✓ Slow initialization of the payload
  - ✓ Combining multiple PanDA jobs into single MPI submission
- Large number of small transfers can saturate Object Stores
  - ✓ Initially Yoda was sending outputs one at a time directly from the compute nodes
  - ✓ Fixed this by asynchronous sending of pre-merged outputs (tar-balls)
- Data stage-out has to be decoupled from the event processing
  - ✓ On HPC use **DTN** (**D**ata **T**ransfer **N**odes) for stage-out
- Prefer few large transfers to the Object Store to many small transfers

Outlook and Future Work

- Avoid fragmentation of PanDA jobs in the Event Service tasks by implementing the new concept of a **Jumbo Job** in PanDA
  - ✓ 1 Jumbo Job = 1 PanDA task
- Implement specialized I/O processes for AthenaMP
  - ✓ **Shared reader:** optimizes data reading on worker nodes, saves memory, also an important step towards the implementation of the Event Streaming Service
  - ✓ **Shared writer:** reduce the number of outputs produced by Event Service payloads
- Design and implement the **Event Streaming Service**
- Extend Event Service functionality to other ATLAS workflows beyond Simulation
  - ✓ Reconstruction, Analysis
- Make Event Service a unified workflow architecture across all ATLAS computing platforms



Grid

Clouds

HPC

Volunteer Computing
ATLAS@Home

Backup

Event Service. Workflow