Integration of Titan supercomputer at OLCF with ATLAS production system

F. Barreiro Megino, K. De, S. Jha, A. Klimentov, T. Maeno, P. Nilsson, D. Oleynik, S. Padolski, S. Panitkin, J. Wells, T. Wenaus
on behalf of the ATLAS Collaboration

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

Outline

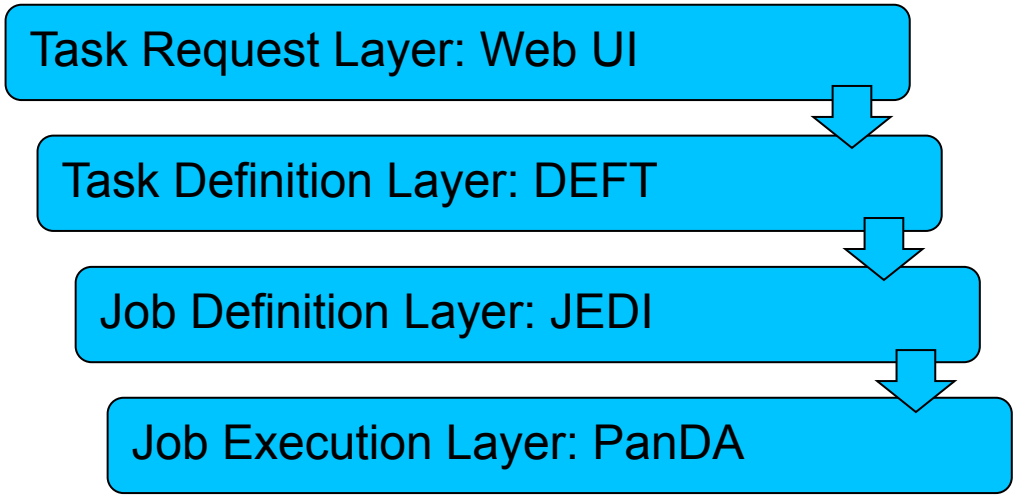
- Introduction and motivation
- ATLAS ProdSys and PanDA workload management system (WMS)
- PanDA setup for integration with Titan
- Results
- Summary

ATLAS and Supercomputers

- Current pace of research and discovery is limited by ability of the ATLAS computing Grid to generate Monte-Carlo events - ”Grid luminosity limit”
  - Currently O(100k) cores available to ATLAS worldwide, ¾ dedicated to MC production.
  - Still not enough CPU power !
  - Many physics simulation requests have to wait for many months
- Supercomputers are rich source of CPUs
- ATLAS initiated R&D project aimed at integration of supercomputing and HPC resources into ATLAS distributed computing
- DOE ASCR supported project aimed at integration of PanDA WMS with Titan supercomputer at OLCF is part of this effort

ATLAS Production System

- Production system is a layer which connects distributed computing and physicists in a user friendly way
- Database Engine for Tasks (**DEFT**): is responsible for definition of the tasks, **chains** of tasks and also task groups (**production request**), complete with all necessary parameters
  - It also keeps track of the state of production requests, chains and their constituent tasks
- Job Execution and Definition Interface (**JEDI**): is an intelligent component in the PanDA server to have capability for **task-level** workload management.
  - Key part of it is '**Dynamic**' job definition, which optimizes usage of resources.
  - Dynamic job definition in JEDI is also crucial for use of multi-core nodes, HPC's, etc
- **PanDA WMS** is job execution layer for the Production System
  - Resource brokerage
  - Job submissions and resubmissions

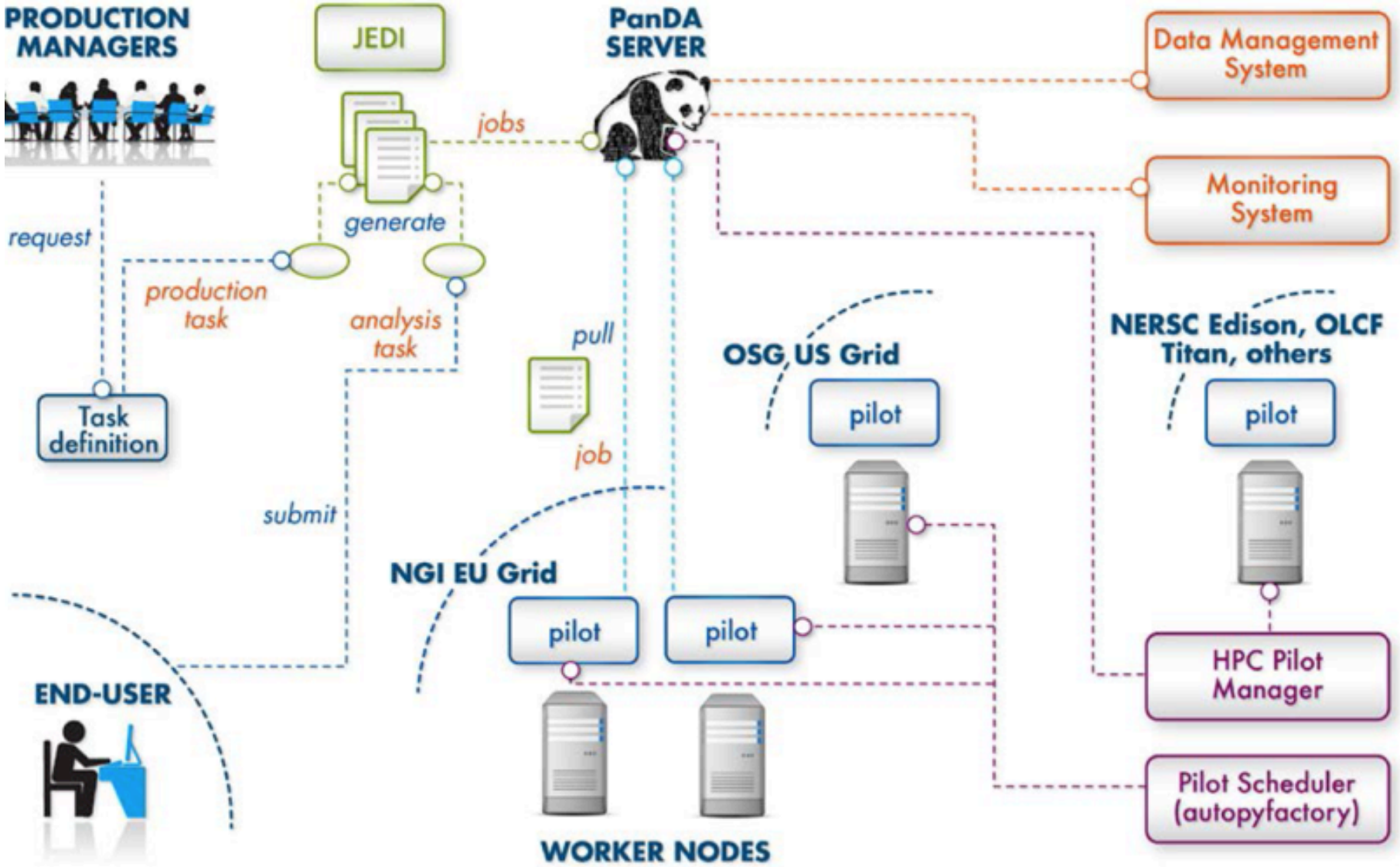


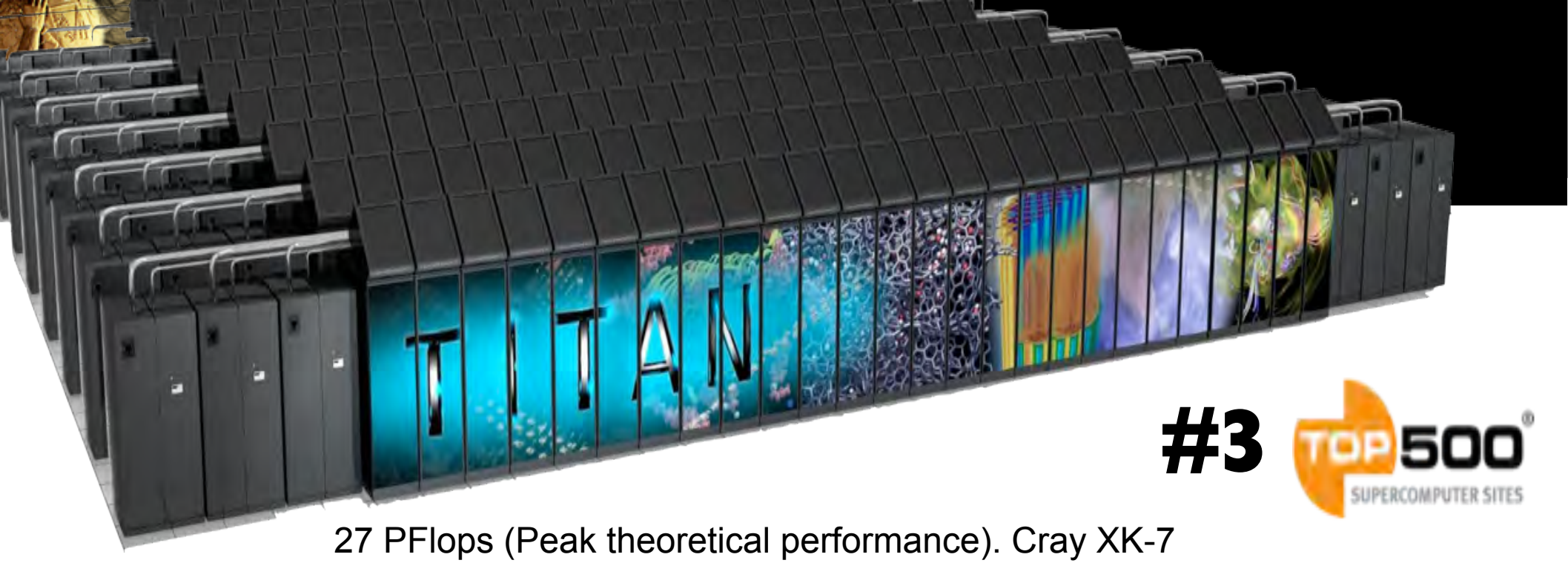
PanDA WMS in ATLAS

- ATLAS uses PanDA Workload Management System (WMS) to run jobs on WLCG
- PanDA - **P**roduction **an**d **D**ata **A**nalysis WMS
- Goal: An automated yet flexible WMS which can optimally make distributed resources accessible to all users
- Adopted as the ATLAS wide WMS in 2008 (first LHC data in 2009) for all computing applications
- Modular, extensible design, Pilot based WMS
- Currently PanDA successfully manages O(10E2) sites, O(10E5) cores, O(10E8) jobs per year, serving O(10E3) users per year
- Current scale: ~25M jobs completed per month
- PanDA is exascale WMS now - since 2013 more than Exabyte of data is being processed every year.

For more details about PanDA see talk by T. Maeno – “PANDA for ATLAS Distributed Computing in the Next Decade”

PanDA Workload Management System





#3 TOP500 SUPERCOMPUTER SITES

27 PFlops (Peak theoretical performance). Cray XK-7
18,688 compute nodes with GPUs

299,008 CPU cores

AMD Opteron 6200 @2.2 GHz (16 cores per node)

32 GB RAM per node

NVidia TESLA K20x GPU per node

32 PB disk storage (center-wide Luster file system)

>1TB/s aggregate FS throughput

29 PB HPSS tape archive

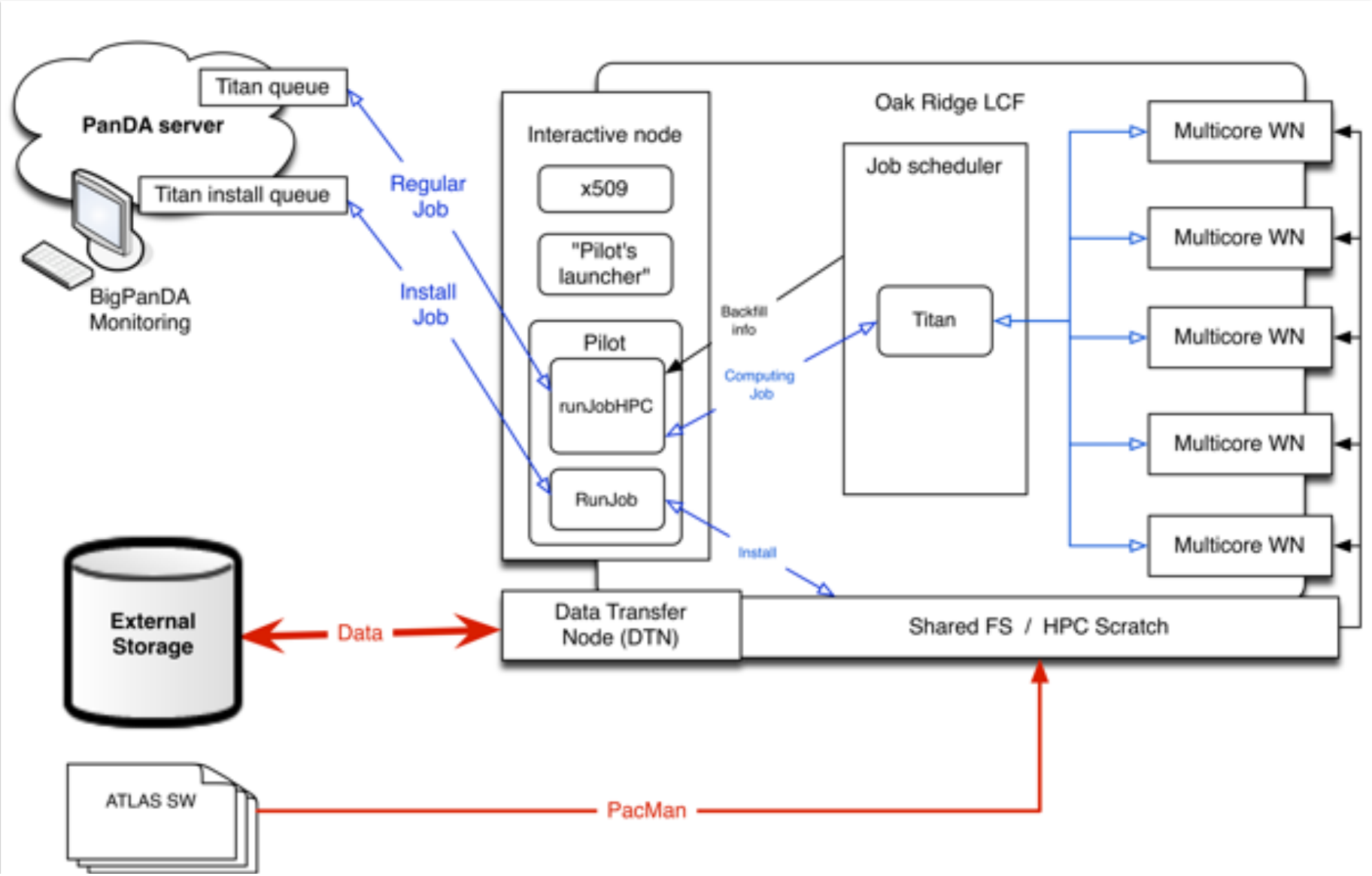
Some Titan features that affect integration with PanDA

- Highly restricted access. One-time password interactive authentication
  - No portals, gatekeepers, VO boxes. Pilot needs to run on Titan's interactive nodes – login nodes or data transfer nodes
- No network connectivity from worker nodes to the outside world
  - Pilot can not run on worker nodes, needs a new mechanism for batch workload management
- Limit on number of submitted jobs in batch queue per user and limit on number of running jobs per user
  - Sequential submissions of single node jobs is not an option
  - Have to use MPI in some form!
- Specialized OS (SUSE based CNL) and software stack
- Highly competitive time allocation. Geared toward leadership class projects and very big jobs
  - Creates opportunity for backfill. Estimated backfill capacity ~300M hours/year

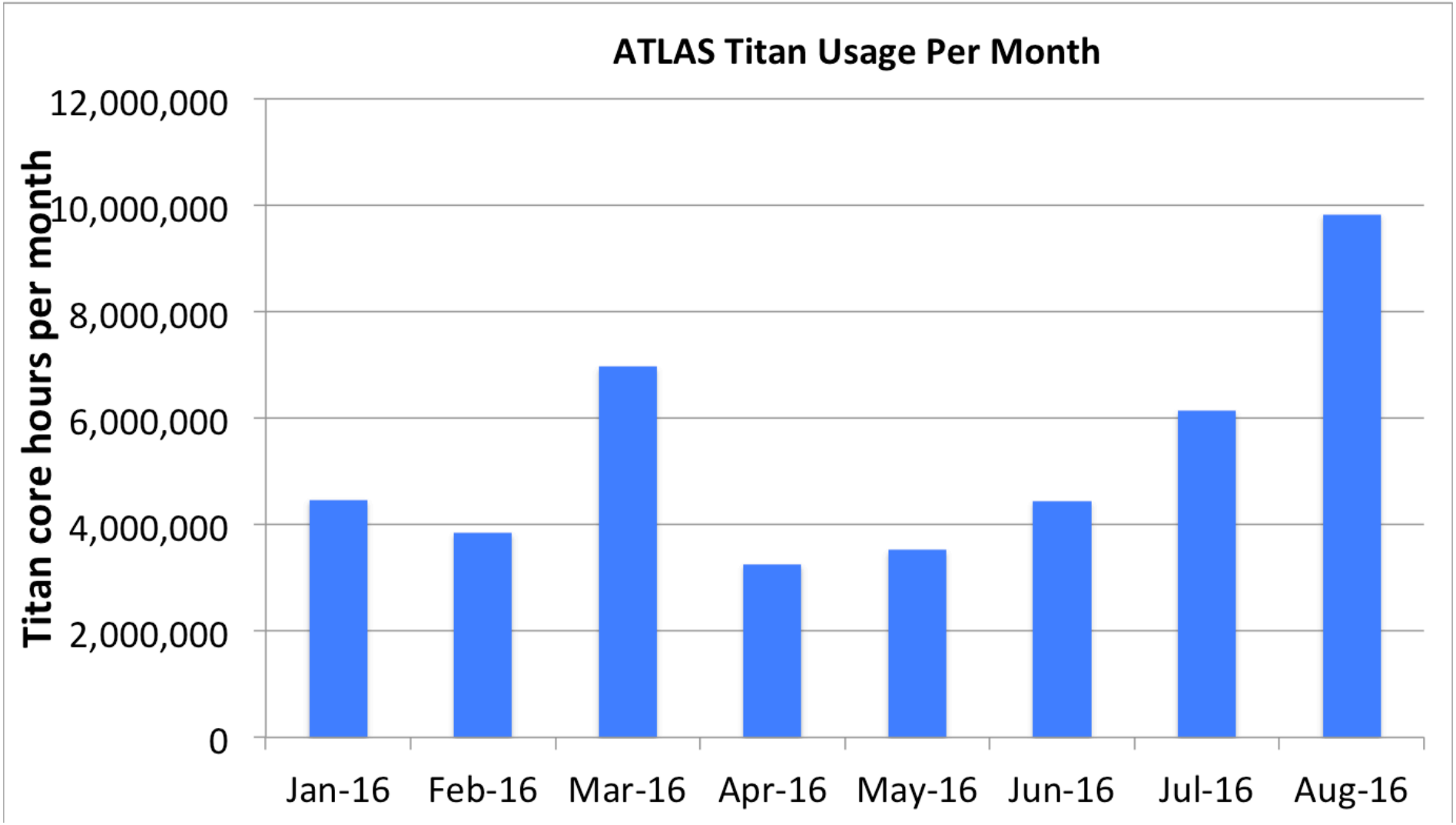
PanDA setup on Titan

- Main idea - try to reuse existing PanDA components and workflow logic as much as possible
- Modified PanDA pilot runs on Titan's front end nodes, in user space
- All connections to PanDA servers at CERN or Amazon EC2 are initiated from the front end nodes by PanDA Pilot over HTTPS
- For local HPC batch interface use SAGA-Python (Simple API for Grid Applications) framework by Rutgers U. group
  - http://saga-project.github.io/saga-python/
- Custom light-weight Python MPI wrapper scripts for running (single node) workloads in parallel on multiple multi-core WN
- Software is installed/ported in advance on Titan's shared file system
- Added capability to PanDA pilot to collect **unused resources (backfill)** on Titan.
  - Our project is running without allocation on Titan since November 2015

PanDA setup on Titan

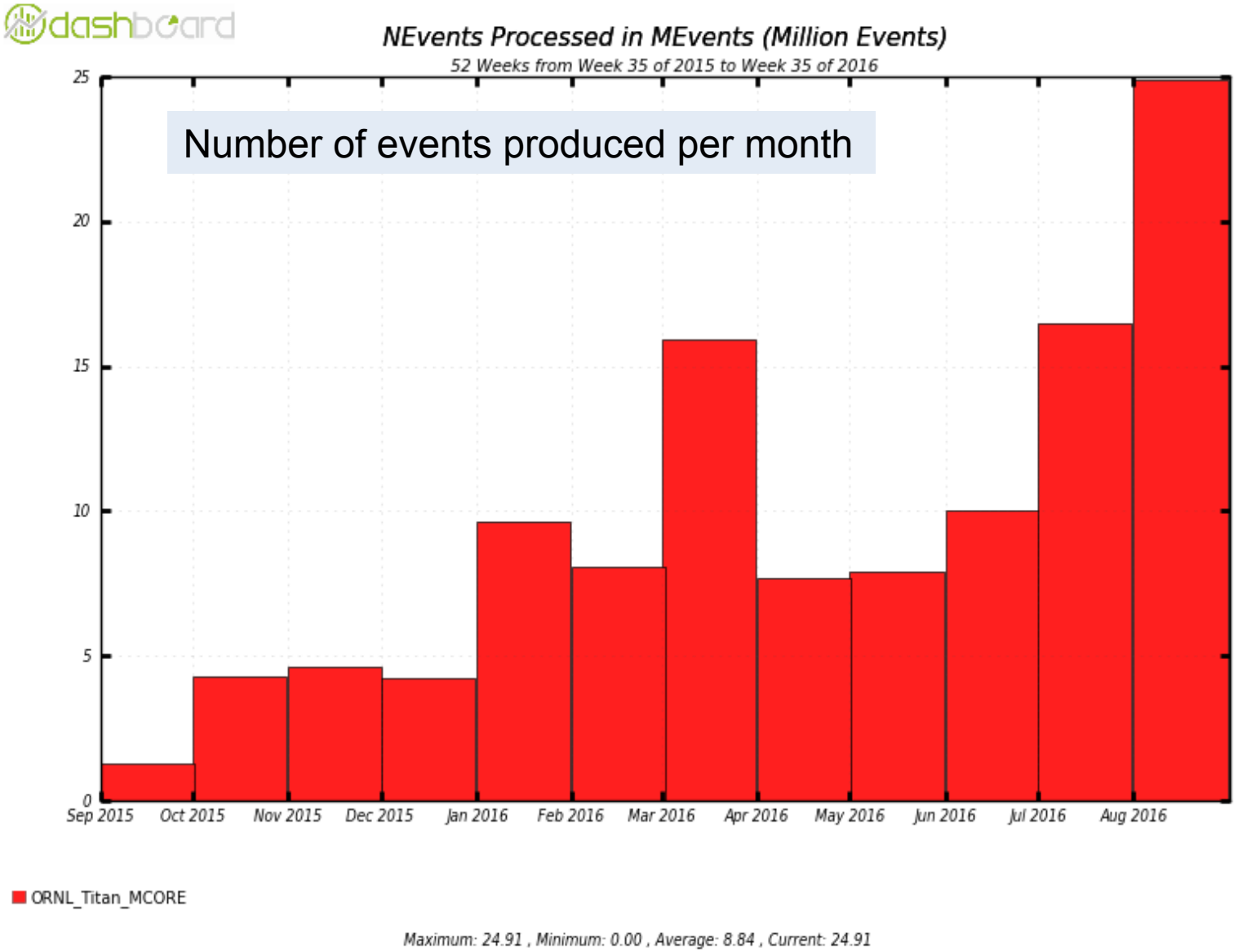


ATLAS production running on Titan in 2016



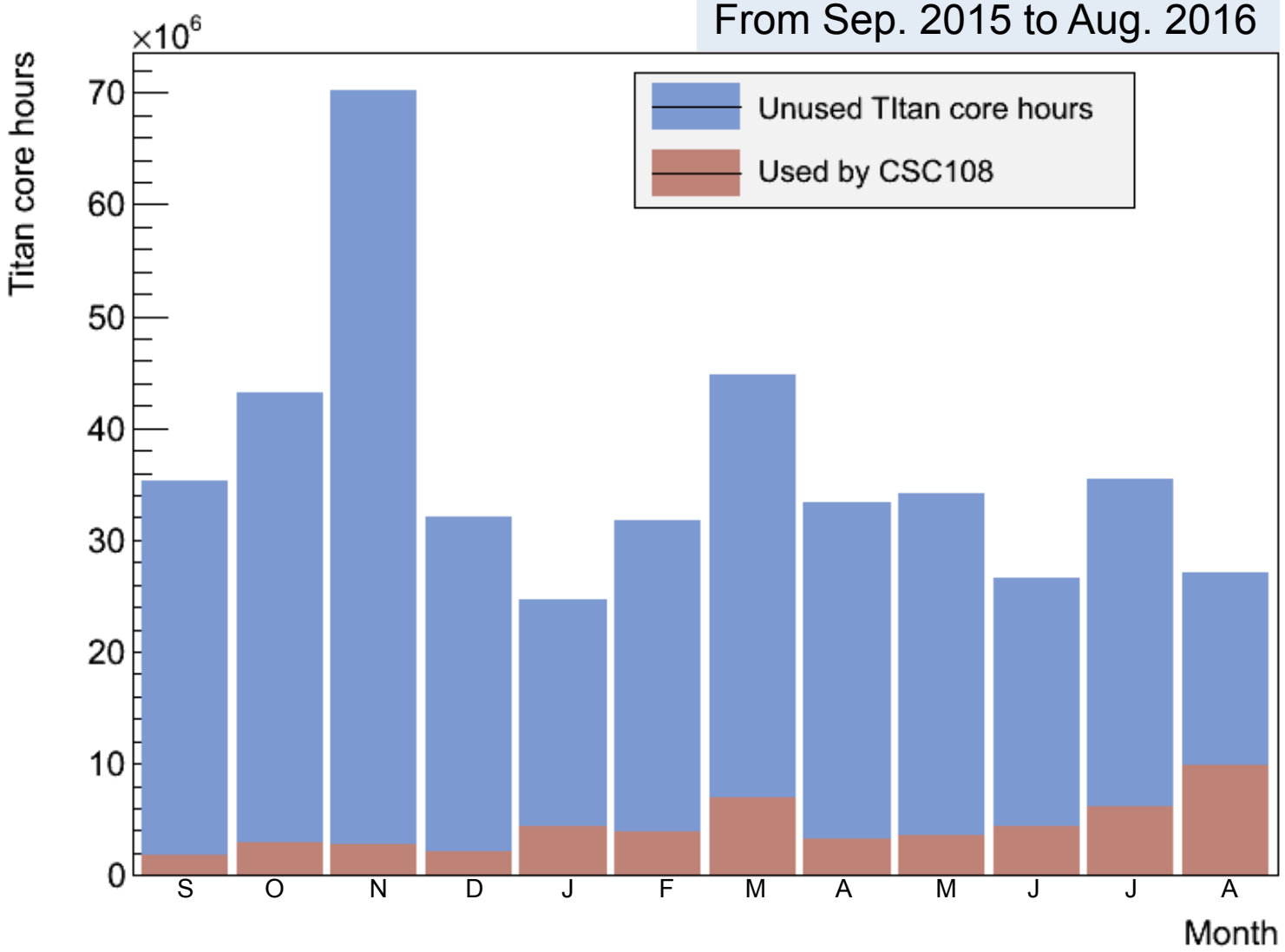
Pure opportunistic backfill mode, no project allocation, ATLAS Geant4 simulations

ATLAS simulations on Titan



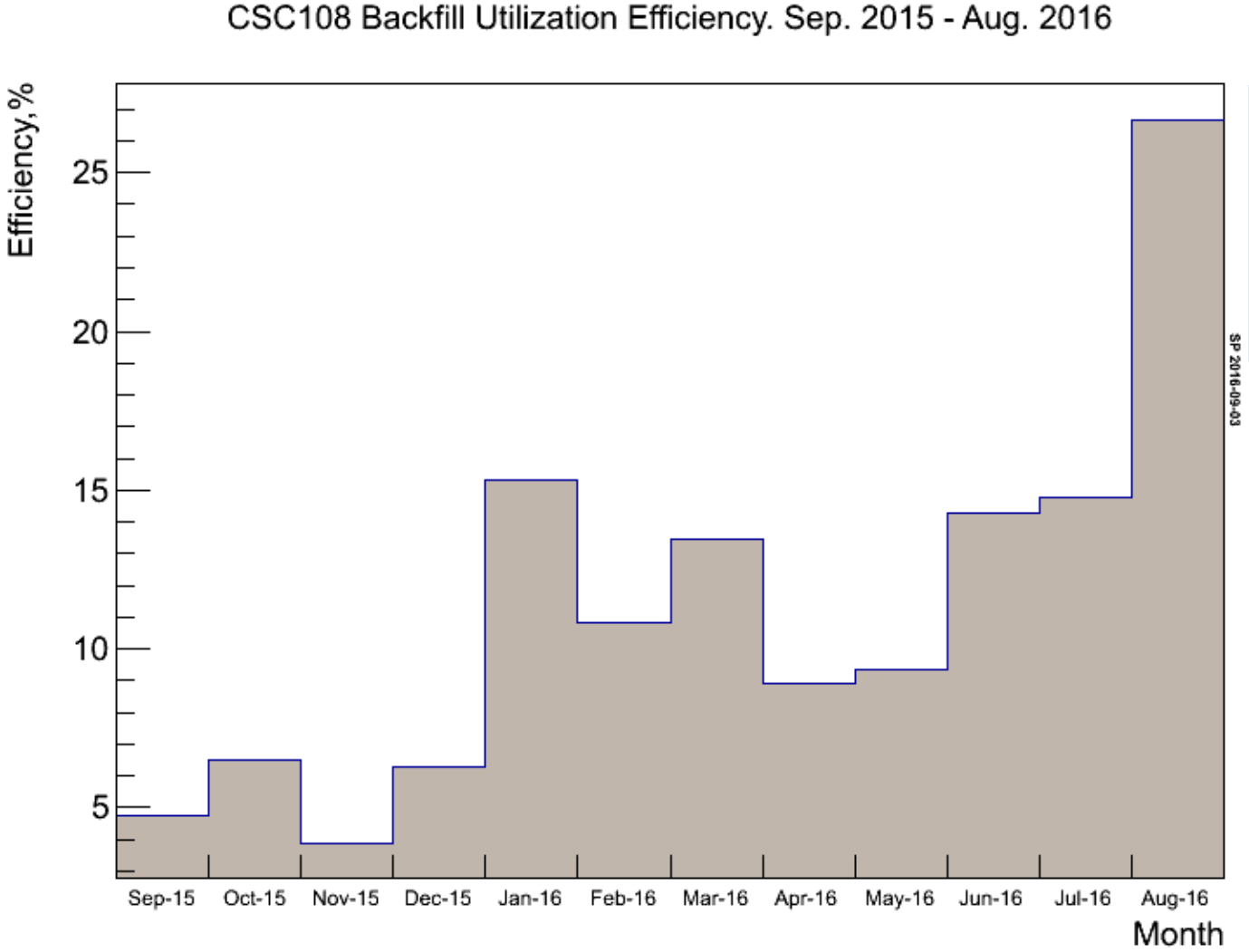
ATLAS CPU consumption vs backfill on Titan

From Sep. 2015 to Aug. 2016



- CSC108 – BigPanDA project on Titan for ATLAS and ALICE

Backfill Utilization Efficiency



- ~10% average efficiency
- ~27% utilization efficiency reached in August 2016
- Up to 2% of total Titan capacity

- Fraction of otherwise unused resources on Titan utilized by ATLAS from Sep 2015 to Sep 2016
- CSC108 – BigPanDA project on Titan for ATLAS and ALICE

Summary

- We completed integration of Titan supercomputer at OLCF with PanDA and ATLAS production system in 2015, as a part of US DOE ASCR funded research project - “BigPanDA”
- Tasks are submitted via ATLAS Production System. Job submission and data movement are fully automatic, with transparent data stage-in/out from/to ATLAS Grid
- In 2015 Titan was officially validated by the ATLAS to run Geant4 simulations
- Since June 2015 we are running ATLAS production tasks on Titan continuously
- Average Titan core hours collection per month: ~4M hours
- Pure backfill operation, running multiple multi-job pilots (currently up to 76800 cores)
- From September 2015 to September 2016, ATLAS project consumed ~52M Titan core hours, ~1.5M detector simulation jobs were completed, ~114M events processed
- We have shown that we can improve overall Titan utilization by ~2%, while consuming up to ~27% of otherwise unutilized resources, all without negatively impacting Titan’s operation and other users on Titan
- ALICE is working on integration of Titan with their production system via PanDA
- In July 2016 DOE ASCR has funded BigPanDA for another 2 years, to expand operations on Titan

Backup Slides

Key Features of PanDA

- Pilot based job execution system
  - Pilot manages job execution on local resources, as well as data movement for the job
  - Payload is sent only after pilot execution begins on CE
  - Minimize latency, reduce error rates
- Modular design
- Central job queue
  - Unified treatment of distributed resources
  - SQL DB keeps state - critical component
- Automatic error handling and recovery
- Extensive monitoring
- HTTP/S RESTful communications
- GSI authentication
- Use of Open Source components
- Workflow is maximally asynchronous

Backfill Enabled Pilot

- Typical LCF facility is ran on average at ~90% occupancy
  - On a machine of the scale of Titan that translates into ~300M unused core hours per year
- Anything that helps to improve this number is very useful
- We added to PanDA Pilot a capability to collect, in near real time, information about current free resources on Titan
  - Both number of free worker nodes and time of their availability
- Based on that information Pilot can define job submission parameters when forming PBS script for Titan, thus tailoring the submission to the available resource.
  - Takes into account Titan's scheduling policies
  - Can also take into account other limitations, such as workload output size, etc
  - Modular architecture, adaptable to other HPC facilities

MPI wrapper for workloads

- In order to use Titan efficiently we have to use MPI
- We utilize light-weight Python MPI wrapper, specific to each workload type
- Uses mpi4py Python module
- The wrapper is launched on Titan by PanDA Pilot as MPI job of arbitrary size
- Then each wrapper instance knows its MPI rank and serves as “mini-Pilot”
  - Sets up Titan specific environment – like loading appropriate modules, environment, etc
  - Sets up workload specific environment
  - Creates working directory, copies necessary files to $PWD, creates symlinks, etc
  - Manipulates necessary input files for each rank to ensure uniqueness of every job output (random seeds, input file lists, etc)
  - Launches actual workload as sub-process and waits until it finishes
  - Performs necessary clean up of working directory or post-processing, if needed
- The wrapper allows to run simultaneously, arbitrary single-threaded or multi-threaded, non-MPI workloads on multiple multi-core worker nodes on Titan