Big PanDA on HPC/LCF Update

Sergey Panitkin, Danila Oleynik

BigPanDA F2F Meeting. March 2014

Outline

- Introduction
- BigPanDA architecture for Titan
- Pilot
  - PanDA Pilot initial changes
  - New features
  - Next steps
- Workloads
  - Current
  - MPI based
- Summary

Current HPC resources for Big PanDA

- Currently we have accounts at:
    - **Oak Ridge Leadership Class Facility (OLCF)**
        - Titan (our own Big PanDA project (CSC108) allocation – 0.5M hours)
        - Kraken (part of NSF XSEDE infrastructure, through UTK allocation)
    - **National Energy Research Scientific Computing Center (NERSC@LBNL)**
        - Hopper, Carver, Edison (through OSG allocation – 1.1M hours)
- We concentrate on ORNL development right now.
    - Great support and interest from OLCF management in Big PanDA
    - Significant CPU time allocation
- Parallel ports to NERSC machines
    - Similar platforms to ORNL - Cray

Titan at ORNL features

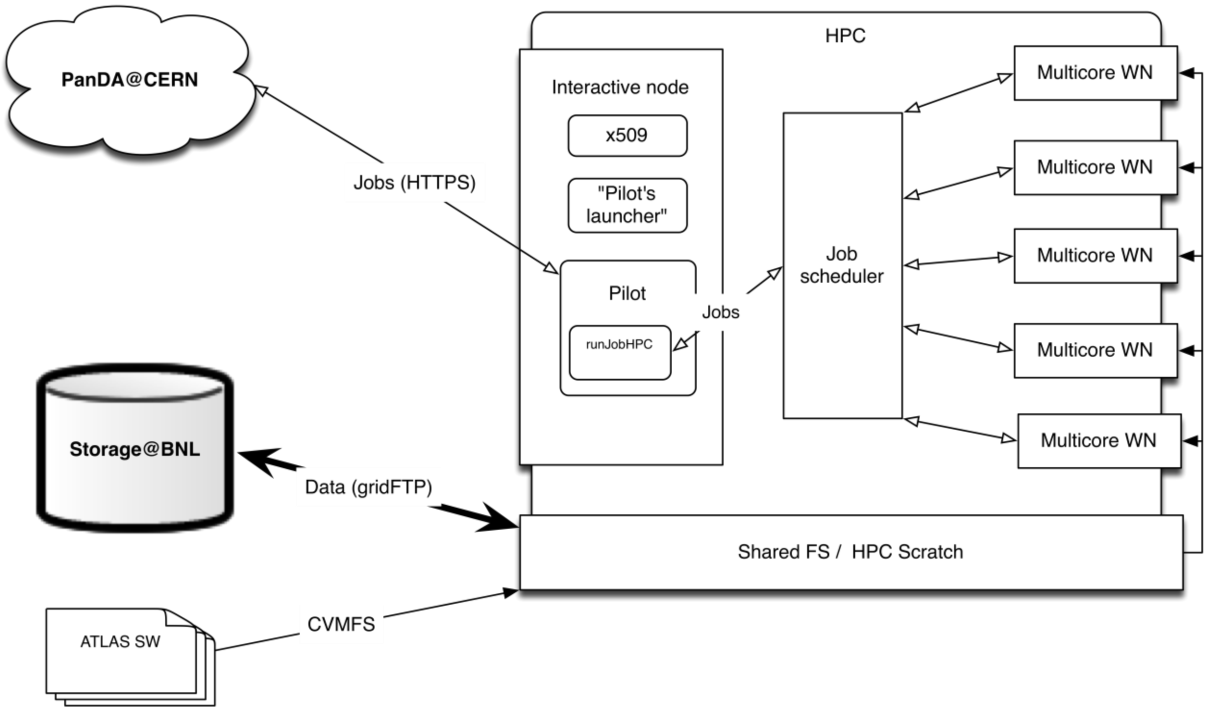
- Titan Cray XK7 (#2 in Top 500)
  - 18,688 nodes with GPUs
  - node: 16 core, 32 + 6 GB RAM (2GB per core)
  - 27 PetaFLOPs theoretical
- Parallel file system shared between nodes, recently upgraded: project workspace 100TB quota (30 PB total capacity)
- 3 types of nodes:
  - Interactive nodes: user interactive login
  - Service nodes: job setup operations, managed through PBS/Torque directives
  - Worker nodes: job executions, managed through ALPS (Application Level Placement Scheduler)
- Special data transfer nodes (high speed stage in/out)
- Highly restricted access:
  - One-Time Password Authentication
  - No network connection with worker nodes
- Limitation of number of jobs in scheduler for one user

PanDA set up on HPC platforms

- Main idea - try to reuse existing PanDA components and workflow logic as much as possible
- PanDA connection layer runs on front end nodes, in user space
- All connections to PanDA server at CERN are initiated from the front end nodes
- “Pull” architecture over HTTPS connections to predefined ports on PanDA server
- For local HPC batch interface use SAGA (Simple API for Grid Applications) framework
  - http://saga-project.github.io/saga-python/
  - http://www.ogf.org/documents/GFD.90.pdf

BigPanDA architecture for Titan

- Pilot(s) executes on HPC interactive node
- Pilot interacts with local job scheduler (PBS) to manage job
- Output transferred to a designated Grid site



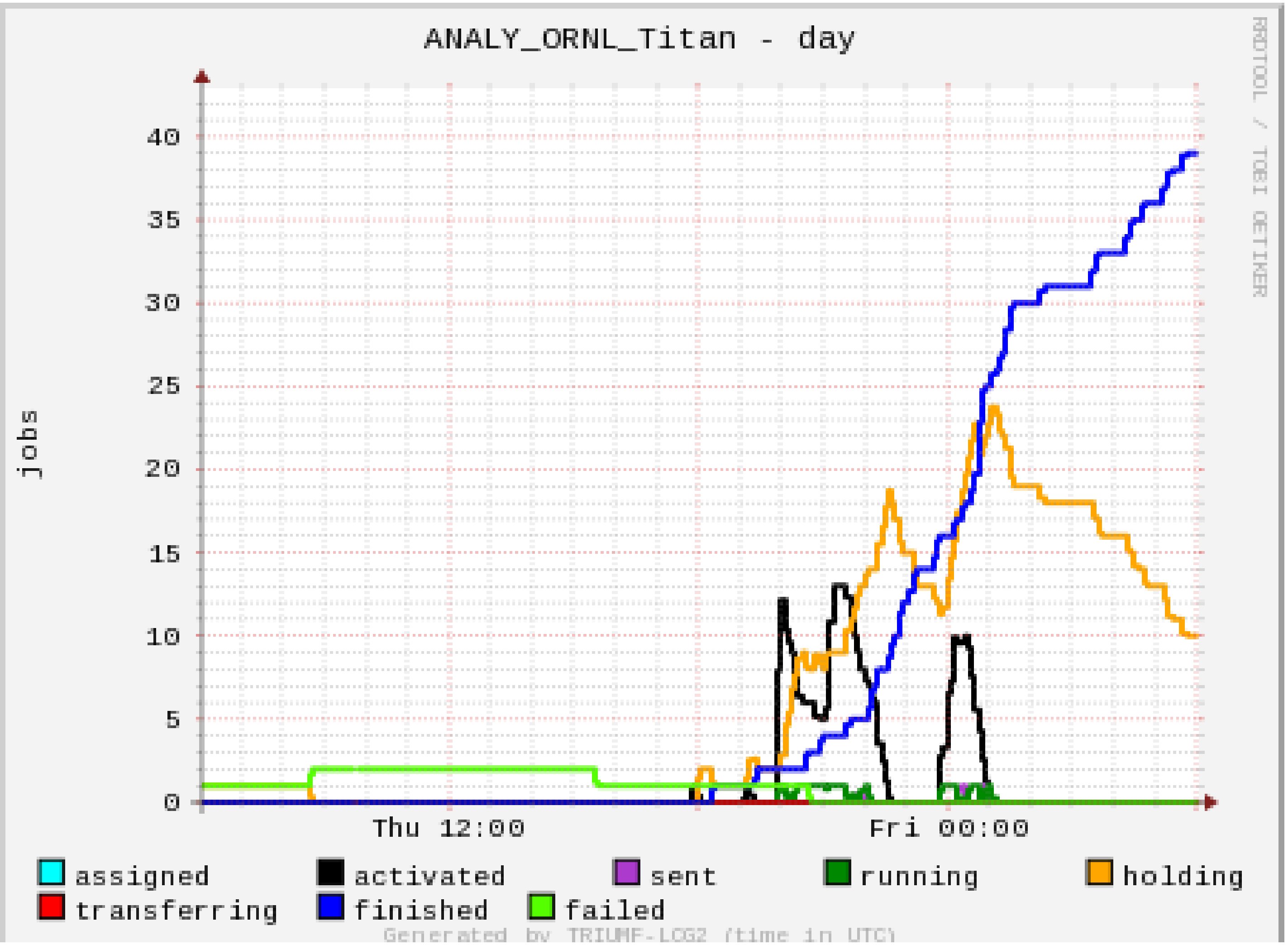
PanDA Pilot initial changes

- Native PanDA pilot was ported to Titan interactive nodes.
  - Correct definition of PanDA queue was needed.
- Main modification was performed for payload execution part: runJobTitan.py module was developed based on runJob.py module.
  - Method, which call payload execution was changed for run and collect results of job execution through PBS;
  - Interface with PBS job manager was implemented by using SAGA API
- Some minor modifications of cleanup procedures was done (subdirectories cleanup).

New features in Pilot

- Proper setup and execution of MPI jobs through ALPS.
- Function for collecting information about available resources for backfill was implemented
- Simple service for Pilots management on Titan was developed.
    - Full PanDA job submission chain on Titan was tested.

PanDA jobs on Titan



Dealing with Transformations

- On a Grid worker node pilot starts a transformation to pull in and set up user payload
- From pilot’s point of view transform is a part of payload.
  - When you submit a job using prun it “wraps/adds” runGen.py transformation script that pilot uses.
  - runGen.py is ~1000 lines of Python code
  - runGen.py needs internet connection (~5 wget), to DDM, to PanDA,,etc
- Problem for HPC application
  - We removed Pilot from worker node space to a place with internet connection
  - Transform still needs to be executed on worker node.
- Can’t use standard grid transforms. Need a substitute of some kind.

New transforms for HPC

- Substitute ATLAS transform with our custom transform script specific to Titan.
  - Sets up Titan specific environment – like appropriate modules, etc
  - Sets up workload specific environment
  - Executes workload
- Right now every workload has it’s own local transform script
- Workloads are precompiled and installed on Titan
- Transforms are installed on Titan
  - Simple python scripts, potentially just shell scripts

Workloads

- Several workloads were ported to Titan
- Root,etc
  - Root based ATLAS analysis
  - Limits setting code (aTGC)
- Event generators
  - SHERPA (v. 2.0.b2 and v. 1.4.3) was ported to Titan and Hopper
  - MadGraph 5 (v. 1.5.12) was ported to Titan and Hopper
  - ALPGEN v 1.4 ported to Titan
  - Simple examples and tutorials for EvGens run
  - Started ATLAS specific ALPGEN test runs on Titan

Limits on aTGC Calculations

- Request from Brian Lindquist (USB) came through ADC to help with his project.
- Limits setting for anomalous triple gauge coupling calculations.
    - CPU intensive
    - Single threaded job takes ~50 hours to calculate one point.
    - Typically 1000 points are needed for one set of parameters.
    - Several sets of parameters are needed for analysis.
    - C++ code
    - Code uses RooFit extension of Root.
    - Can be ran in multi-threaded mode .
- Difficult to run on the Grid. Ideal workload for HPC.
- Converted code to use MPI libraries
- Ran for 50k core-hours run on Carver@NERSC

Need for MPI

- To run effectively on HPC MPI aware workloads are needed
- Use of MPI will allow us to run multiple independent serial jobs as an ensemble, with just one submission at time.
    - Every job knows it’s place in a group and size of the group
- Good for backfill job submission
    - MPI allows to adjust the size of submitted jobs in a natural way.
    - The size of the available "backfillable" gap becomes MPI rank.
- MPI allows to avoid, or at least mitigate, batch queues limits on number of simultaneously submitted tasks

- As a separate note: GPU aware workloads are prime targets for HPC these days.
    - More efficient use of allocated time. Accounting system counts whole node as a node with GPU.
    - It would be great to have such codes in ATLAS.

MPI Workloads

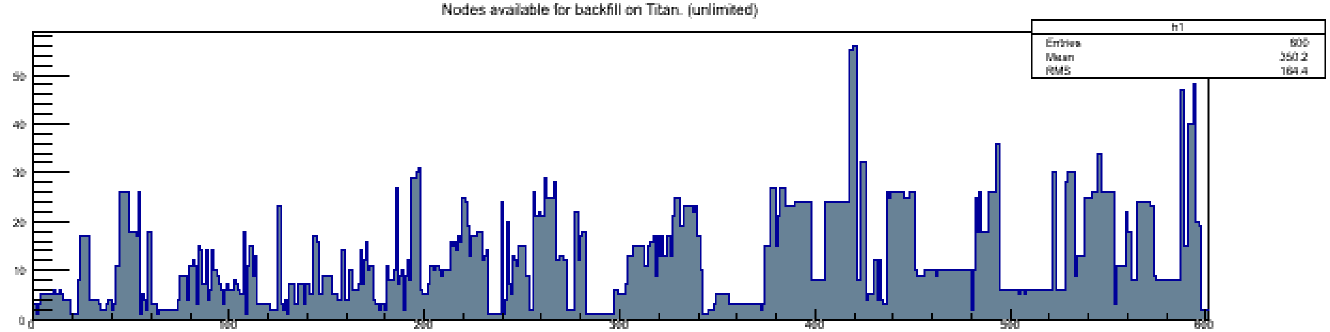
- Workloads with Native MPI support (SherpaMPI, etc)
- Customized ATLAS codes (f.e. like aTGC code or Alpgen@ANL)
- MPI transforms
    - We tested a transform to run a set of ALPGEN jobs as MPI collection
    - In principle this type of transforms can be used for other non MPI jobs
    - Working on running ATLAS Z-tautau-jets Alpgen production on Titan
        - Problem with Alpgen input file definition extracted from ATLAS job definition
        - Very long Alpgen “warm-up” phase (>>24hours) prevents from running this on Titan directly
        - Discussing this with ANL group. Hopefully resolved soon.
        - Issue with random number generation for very large number of events. Limited generator period.
        - Working on more general Alpgen transform for Titan based on ATLAS AlpGenUtil.py

Opportunistic backfill on Titan

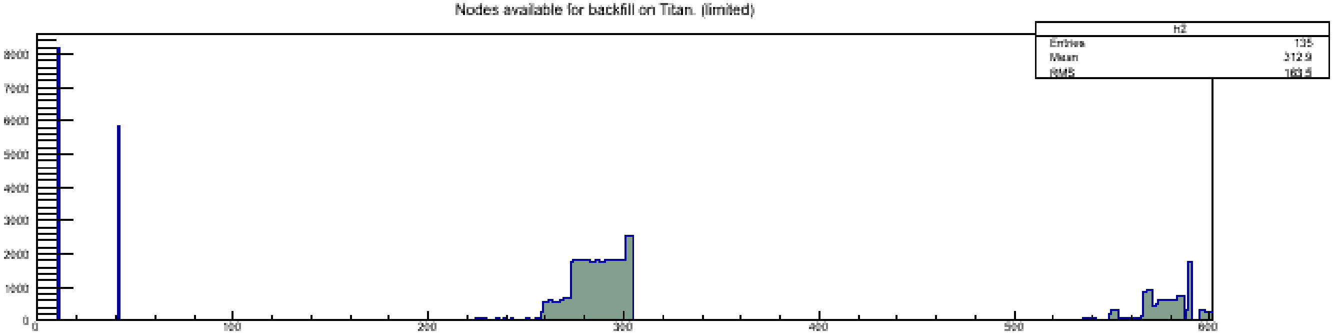
- More details in Danila’s slides
- As a first step a simple algorithm was implemented:
    - Pilot queries MOAB scheduler about unused transient resources
    - Information about available resources returns in a format that includes a number of currently unscheduled nodes and period of their availability
    - Pilot chooses the largest available block of free nodes and generates appropriate job submission parameters, taking into account Titan’s scheduling policy limitations
    - Pilot uses MPI based transform

Titan Backfill 1

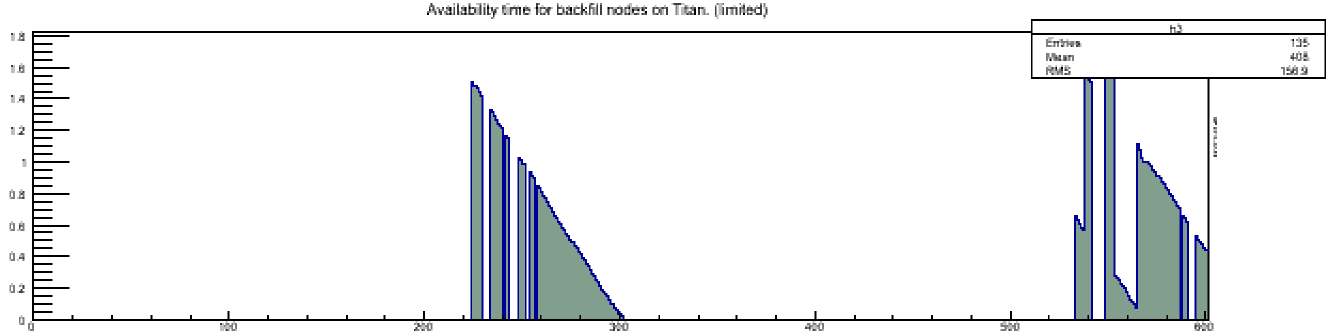
Indefinitely available nodes



Nodes available for a limited time



Availability time estimate



Availability query index

Initial backfill tests on Titan

Submitted	Account	Nodes	Cores	Wait	Walltime limit	Runtime	State	Completed
Mar, 04 16:26	CSC108	**6**	**96**	0.00	1:59:00	0,01	Completed	Mar, 04 16:27
Mar, 04 16:52	CSC108	**185**	**2960**	0.07	5:59:00	0,02	Completed	Mar, 04 16:58
Mar, 04 17:32	CSC108	**608**	**9728**	0.01	11:59:00	0,02	Completed	Mar, 04 17:34
Mar, 04 17:45	CSC108	**578**	**9248**	0.01	11:59:00	0,03	Completed	Mar, 04 17:47
Mar, 04 17:51	CSC108	**1,649**	**26,384**	0.00	11:59:00	0,03	Completed	Mar, 04 17:53
Mar, 04 18:03	CSC108	**636**	**10176**	0.01	11:59:00	0,02	Completed	Mar, 04 18:05
Mar, 04 18:09	CSC108	**740**	**11840**	0.13	11:59:00	0,02	Completed	Mar, 04 18:18
Mar, 04 18:21	CSC108	**577**	**9232**	0.00	11:59:00	0,03	Completed	Mar, 04 18:22
Mar, 04 18:25	CSC108	**596**	**9536**	0.04	11:59:00	0,02	Completed	Mar, 04 18:28

- Jobs submitted through PanDA to Titan
- “Backfill capture” is almost instantaneous!
- No competition for the resource?
- More studies of backfill properties are planned

Next steps

- Additional redesign of Pilots components still needed for:
  - parallel execution of pilots on same worker node
  - Changing of data format for parameters which describe setup and execution of payload (partly done for current PanDA – Titan execution, quite difficult for debug due to dependencies from experiment specifics and different types of jobs
  - Multi HPC site demonstrator in PanDA (Titan, Kraken, NERSC, EOS,…)
  - New Cray XC30 installation became available at ORNL – called EOS
    - 744 nodes, Xeon E5-2670, no GPUs
    - Better scheduling policy limits
  - Need a meeting with Titan folks to discuss backfill status and possibilities
  - Discuss with ALICE (Ken Read) possible workloads to run on Titan as multi-VO demonstrators
  - Take another look at ATLAS software on Titan (cvmfs)

Summary

- Work on integration of OLCF, NERSC machines and PanDA is in progress
- Successful “backfill through PanDA” demonstrator on Titan
- Workloads ports are in progress
  - HEP event generators ported (ALPGEN, Sherpa, Madgraph)
- Conversion of ATLAS code to MPI
  - aTGC limits calculations performed. Direct code conversion to MPI. 50k core hours delivered @NERSC
  - MPI transform for ALPGEN tested
- MPI and GPU aware codes are needed
- Discussion about SUSY parameter scan has started