Porting PanDA pilot to Oak Ridge Leadership Computing Facilities. Status report.

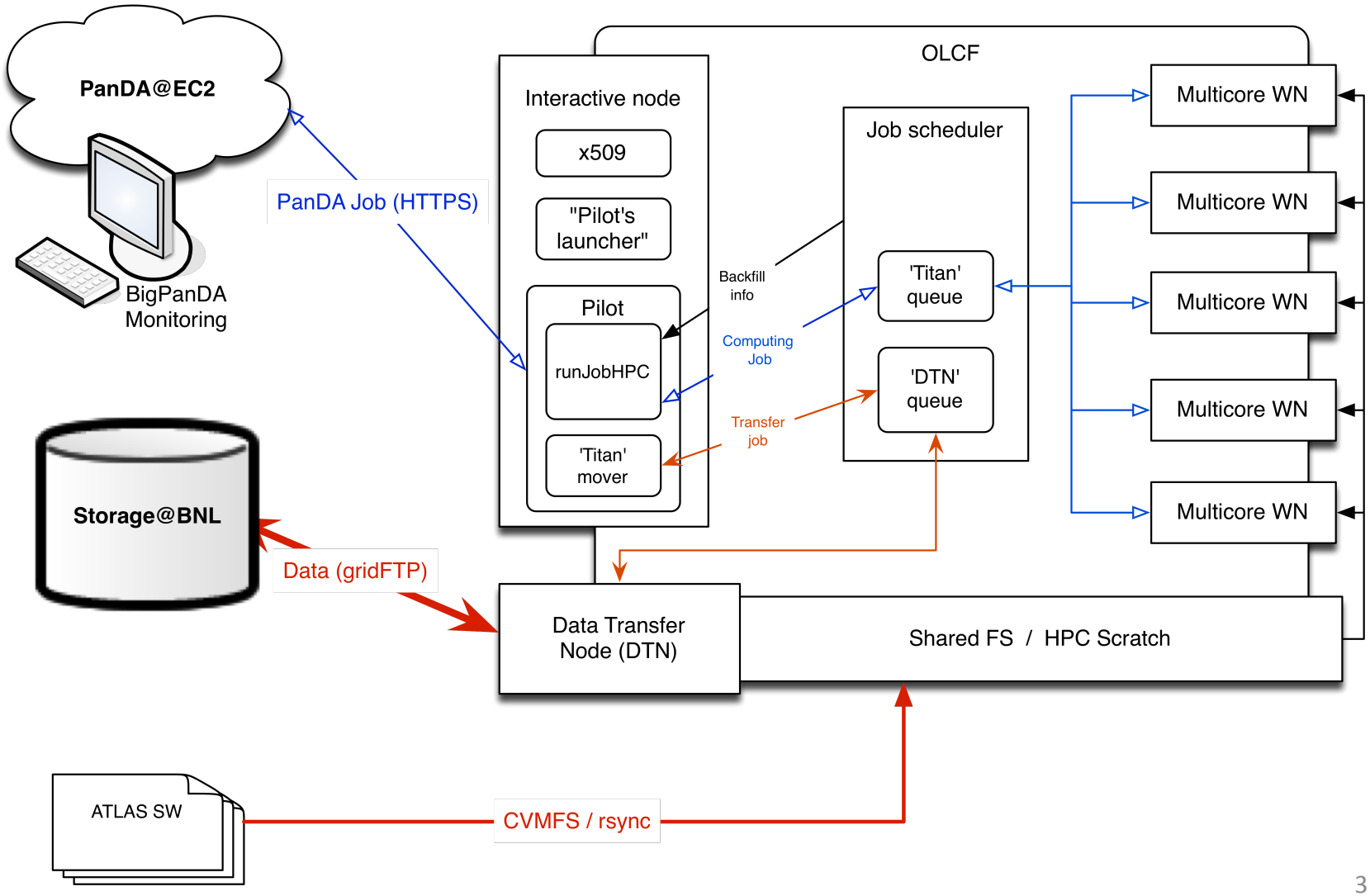
Danila Oleynik, UTA

21 August 2014

Introduction

- PanDA pilot integration with OLCF has two goals: make use of allocation and autofill at Titan, and developing generic solution which can be easily adapted for other HPC facilities.
- In addition to factorization of the pilot, we have identified some common features of HPC's:
  - Restricted/no external access to computing nodes
  - payload execution management only through a local batch system
  - Special treatment of shared file systems

PanDA@OLCF


PanDA@EC2
BigPanDA Monitoring
PanDA Job (HTTPS)
OLCF
Interactive node
x509
"Pilot's launcher"
Pilot
runJobHPC
'Titan' mover
Backfill info
Computing Job
Transfer job
Job scheduler
'Titan' queue
'DTN' queue
Multicore WN
Multicore WN
Multicore WN
Multicore WN
Multicore WN
Storage@BNL
Data (gridFTP)
Data Transfer Node (DTN)
Shared FS / HPC Scratch
ATLAS SW
CVMFS / rsync

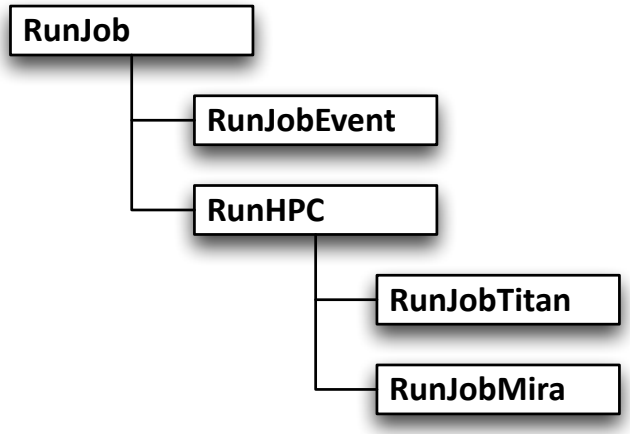
PanDA@OLCF (Details)

- Pilot(s) executes on interactive node (or some edge node)
  - Allow all needful connections with PanDA server
- Pilot interact with local job scheduler to manage job
  - Realized on high level abstraction for supporting wide range of batch systems
- Number of executing pilots should be equal or less of number of available slots in local scheduler
  - Increase efficiency of usage of HPC
- Stage in/out procedures goes through dedicated OLCF facility – Data Transfer Nodes (DTN)
  - Speedup transfers

Pilot changes for Titan

- Minor changes in ATLASexperiment class for compatibility verification (to allow proper startup of pilot)
- Functionality for supporting payload execution process through internal batch system encapsulated in a dedicated class
- Declaration of common, but HPC specific, methods and parameters done in dedicated class.
- NO changes in other pilot components

PanDA Pilot architecture update



- Proper class selection based on schedconfig.catchall
  - E.g. catchall = “HPC_Titan” -> RunJobTitan gets selected

RunJob class

- Base class for supporting payload execution workflow:
  - A lot of common methods which have no depends from computing backend

RunJobHPC class

- Inherited from RunJob
- Support common methods and additional parameters for execution of payload on HPC:
  - Limit on maximum number of allocated nodes (cores)
  - Limit on waiting time (before internal rescheduling)
  - Limit on minimum walltime
  - Set of configuration parameters (better to propagate them through schedconfig)

RunJobTitan class

- Inherited from RunJobHPC
- Provides execution of MPI payloads on Titan
  - SAGA API used as an interface with internal batch manager (PBS) on Titan
  - Instrumented for efficient use of 'backfill' resources:
    - Special function collects information about available resources (number of nodes and availability time) from MOAB
    - PBS job parameters are formed according to available resources and Titan queue policies
    - Introduced PBS wait time limit and retry mechanism

Continuous tests on Titan LCF (July)

- Provided for evaluation of stability of full workflow
- 3 sets by 8 hours
- During testing of backfill algorithm efficiency consumed 146000 core/hours
- In most of cases waiting time less than 5 min.
- Detected IO problems on **huge allocations** (dozens thousands of cores). Mostly related with non optimized IO in payload, cleanup procedure in pilot will be needed optimization for specific architecture.
  - Intensive IO may affect not only payload execution time, but reliability of facility itself
  - Works together with OLCF team for proper solution

Continuous tests on Titan LCF (August)

- Provided for testing of algorithm for internal rescheduling of payload (backfill procedure optimization)
- Wait time limit – 2 min.
- ~ 10 hours without interruptions
- Single stream of pilots
- Consumed about 14,4% of available resources on Titan (2,3% of all Titan resources)

Functional tests at NERSC (Hopper, Edison)

- Same solution as for Titan (with catching backfill resources) was successfully tested on Cray machines at NERSC
  - Minor changes for NERSC policy were needed
  - Most of changes were for 'static' parameters like queue name and partition, number of cores per node etc.
- Compared to Titan, due to different mixture of jobs (different use policies?) at NERSC machines, job backfill will be not as efficient. Many small jobs, fewer free resources.

Next steps

- Review of methods of collecting monitoring information and extending with HPC specific data (number of allocated nodes, state of payload in internal queue, etc)
- IO optimization for OLCF (initially):
  - optimization of cleanup procedure,
  - Proper involving DTN for stage in/out procedures (Titan Mover).
- Testing solution with other HPCs:
  - Supercomputer in Ostrava (Czech National Supercomputing Centre)
  - ARCHER (Edinburgh)