Testing PanDA at ORNL

Danila Oleynik

University of Texas at Arlington / JINR

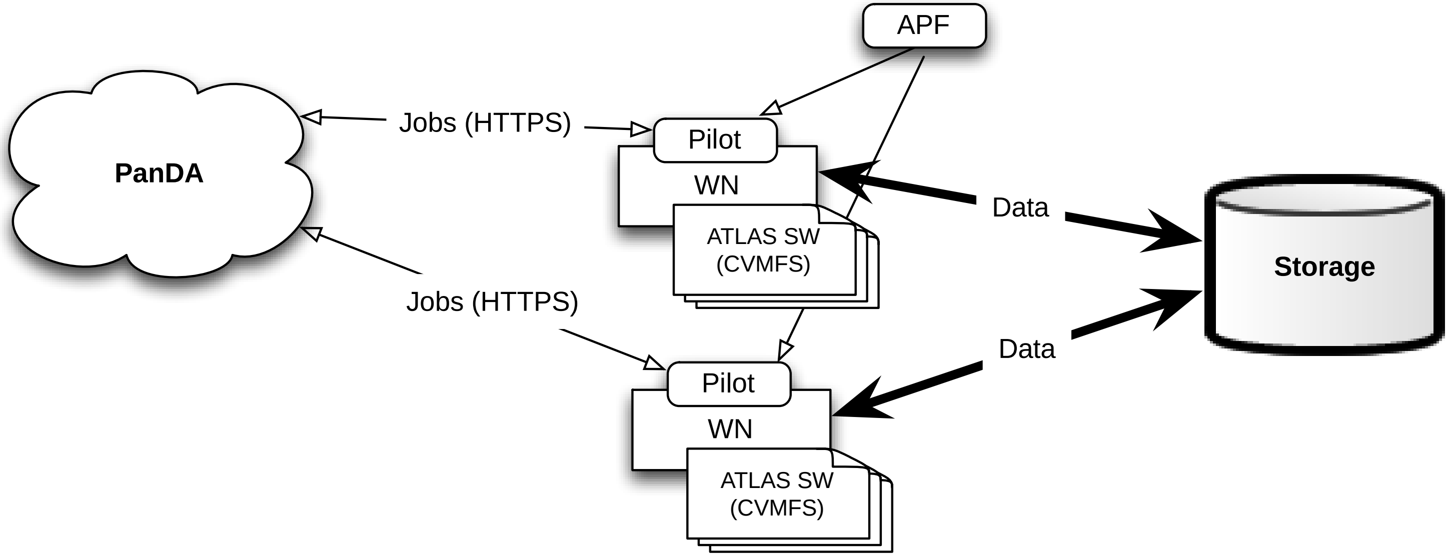
PanDA Workshop @ UTA
3-4 of September 2013

Outline

- Current PanDA implementation
- HPC at ORNL architecture, specialty
- PanDA architecture for Kraken, Titan
- PanDA Pilot modification to cope HPC
- SAGA API
- Initial testing
- In progress: Pilot - SAGA integration.

Current PanDA implementation

- One Pilot per WN
- Pilot executes on same node as job
- SW distribution through CVMFS

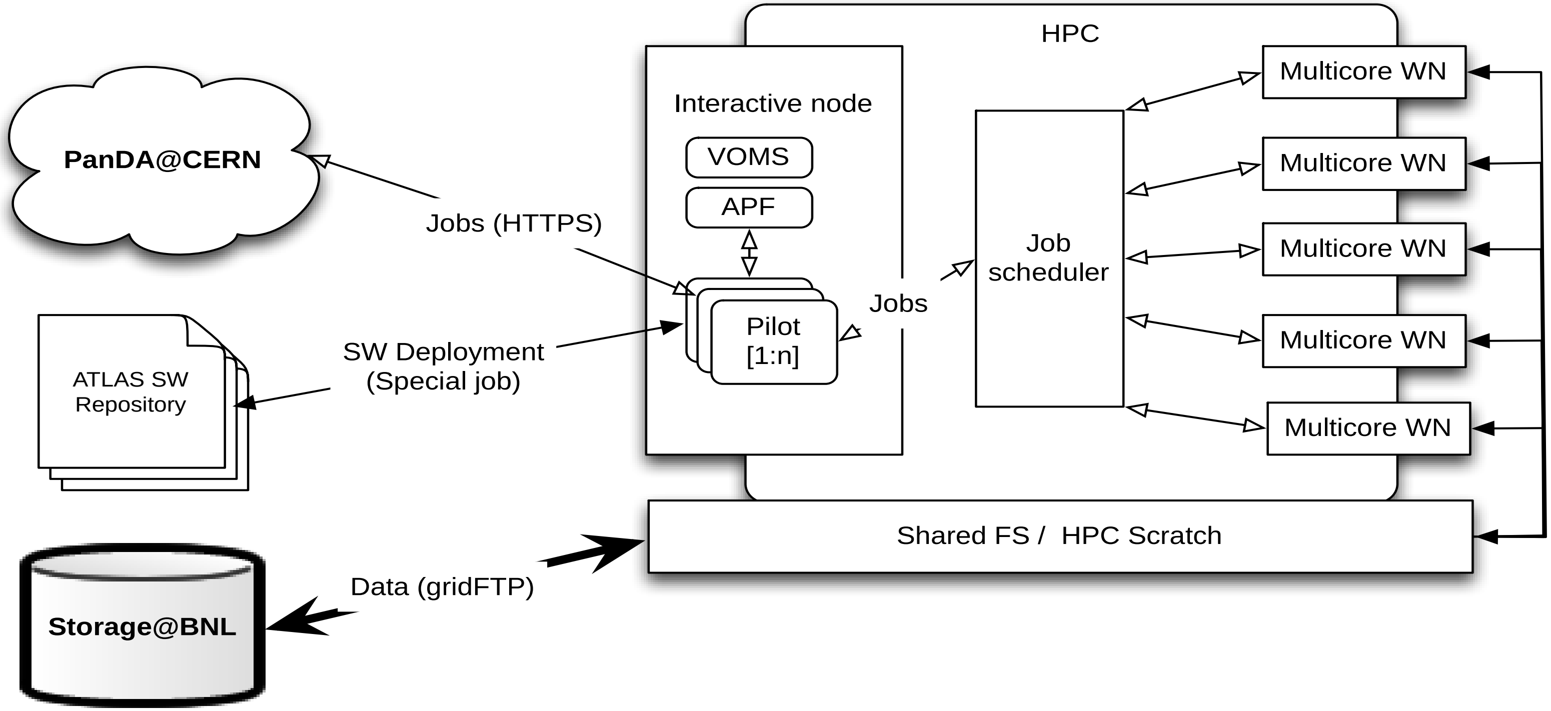


HPC at ORNL architecture, specialty

- Kraken Cray XT5
  - 9408 nodes
  - node: 12 core, 16 GB RAM
- Titan Cray XT7
  - 18,688 nodes
  - node: 16 core, 32 + 6 GB RAM (2GB per core)
- One-Time Password Authentication
- Parallel file system shared between nodes.
- Extremely limited access to worker nodes
- Internal job management tool: PBS/TORQUE
- One job occupy minimum one node (12-16 cores)
- Limitation of number of jobs in scheduler for one user: slots limitation
- Own compilers, interpreters, libraries

PanDA architecture for Kraken, Titan

- Pilot(s) executes on HPC interactive node
- Pilot interact with local job scheduler to manage job
- Number of executing pilots should be equal or less of number of available slots in local scheduler



PanDA Pilot modification to cope HPC

- Main modification of Pilot for working with HPC, is changing in part of Job executing process (runJob class)
- Environment validation procedures should be checked.
- Stage In/Stage Out procedures should be checked.

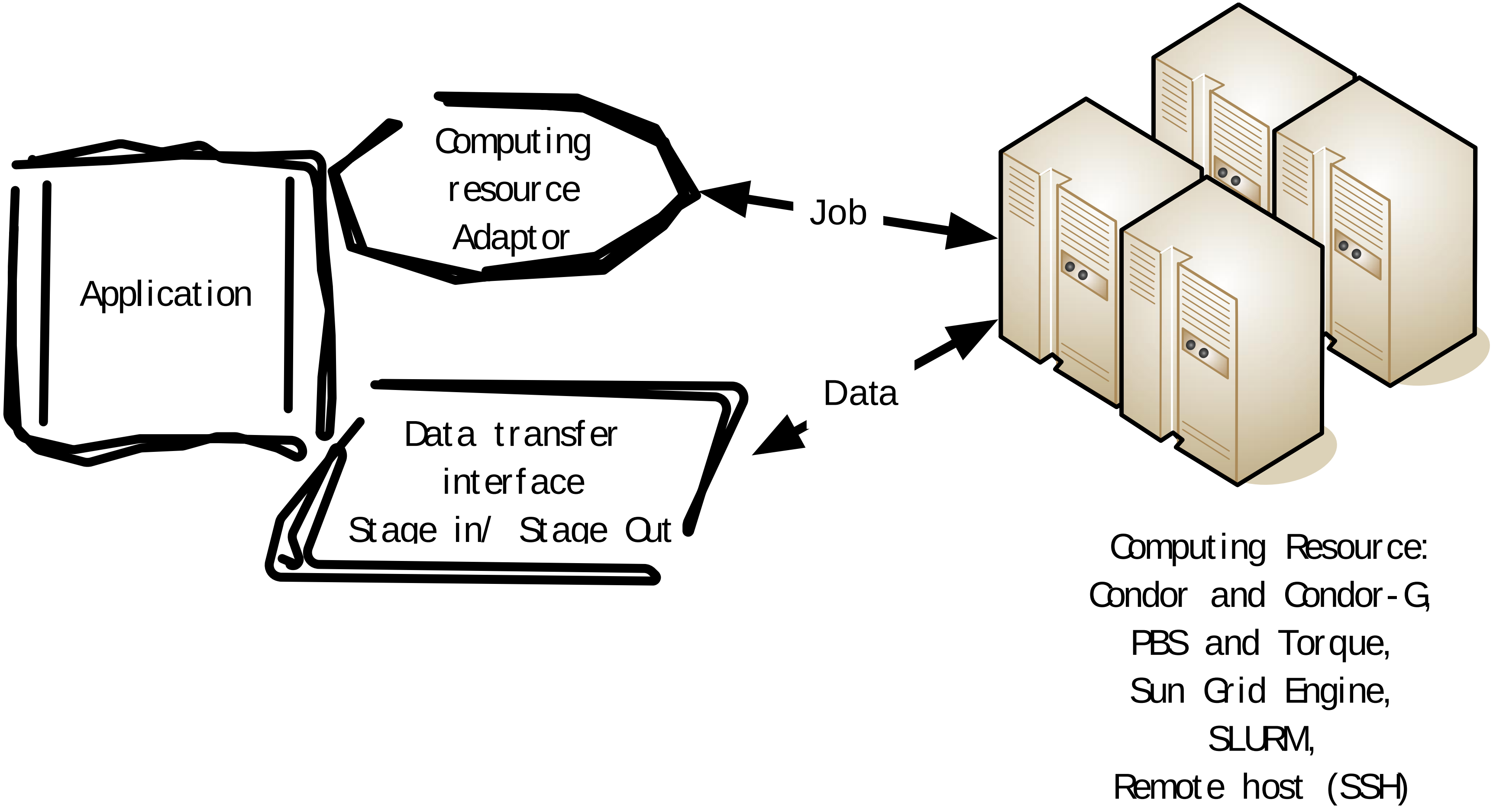
SAGA API

- SAGA (*Simple API for Grid Applications*) defines a high-level interface to the most commonly used distributed computing functionality. SAGA provides an access-layer and mechanisms for distributed infrastructure components like job schedulers, file transfer and resource provisioning services
- Behind the API facade, SAGA-Python implements a flexible *adaptor* architecture. Adaptors are dynamically loadable modules that interface the API with different middleware systems and services.
- Developed and intensive maintained by RADICAL Research at The Cloud and Autonomic Computing Center, Rutgers University.
- http://saga-project.github.io

SAGA API. Supported Middleware

- **Job Submission Systems**
  - SSH and GSISSH
  - Condor and Condor-G
  - PBS/Torque
  - Sun Grid Engine
  - SLURM
- **File / Data Management**
  - SFTP/GSIFTP
  - HTTP/HTTPS
- **Resource Management / Clouds**
  - EC2 (libcloud)

SAGA API. Typical schema



Computing Resource:
Condor and Condor-G,
PBS and Torque,
Sun Grid Engine,
SLURM,
Remote host (SSH)

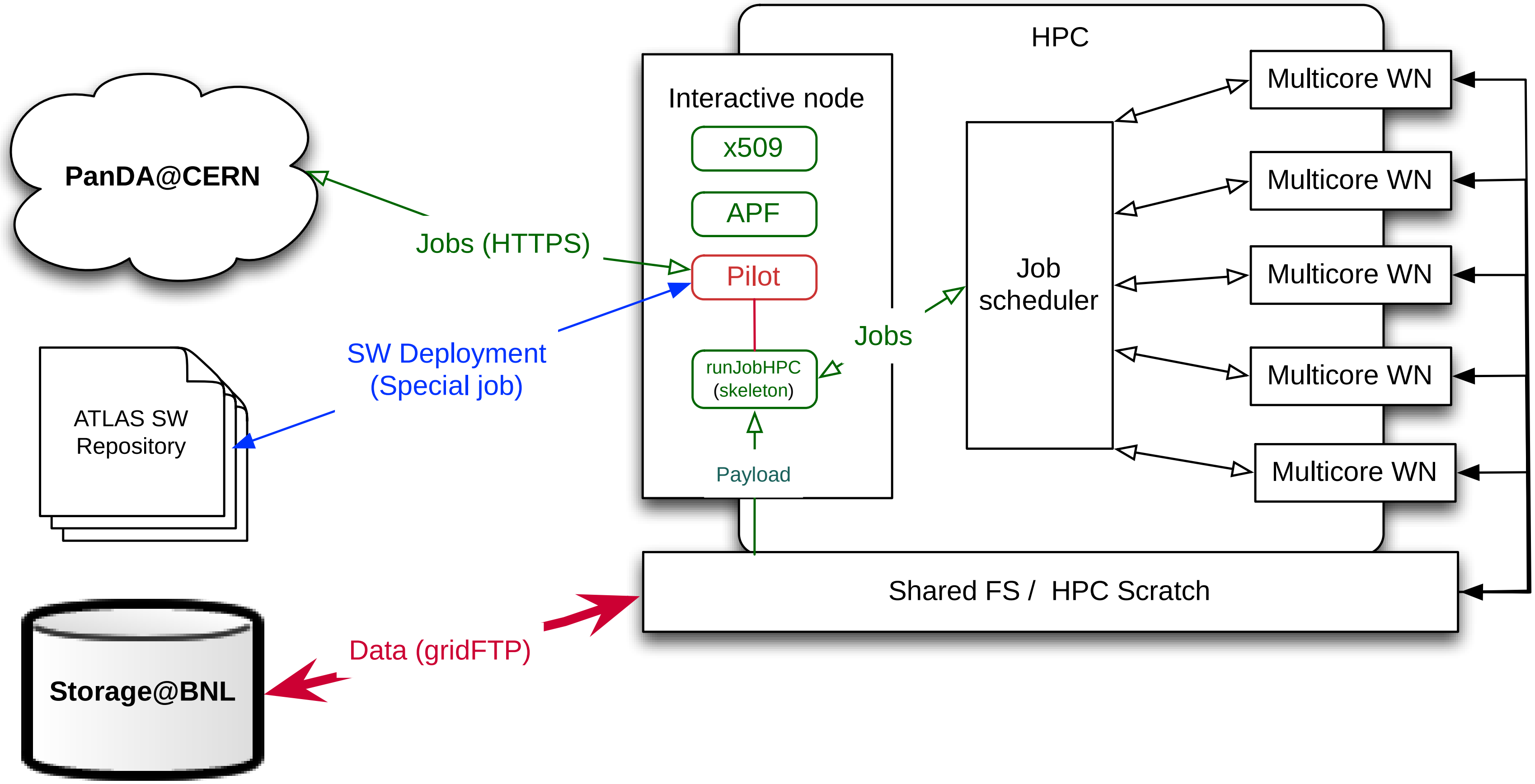
SAGA API. Other features

- Configuration and logging
- Arguments and environment setup for jobs execution in batch processing systems
- Different security contexts:
  - UserPass
  - SSH
  - X.509
  - MyProxy
  - EC2
  - EC2_KEYPAIR

Initial testing

- Initial testing was done for proving that PanDA components will be able to run in HPC environment on interactive nodes
- APF and Pilot was successfully started on HPC
- Outbound https connection was confirmed, so pilots can communicate with PanDA server
- SAGA API was successfully tested on HPC for managing jobs in local job scheduler
- Due to interactive node and worker nodes use shared file-system, we did not need any special internal data-management process
- Connection from Titan FE to Federated ATLAS Xrootd is verified

Initial testing



In progress: Pilot modification

- Though Pilot was run on HPC and communication with PanDA server was established, procedures for checking environment for specific experiment for the moment fails.
  - In process modification of this procedures
- Next steps:
  - Validation of stage in/stage out procedures
  - Validation of simple job execution on HPC head node
  - Development of runJodHPC class, to manage job in PBS/Torque job scheduler.