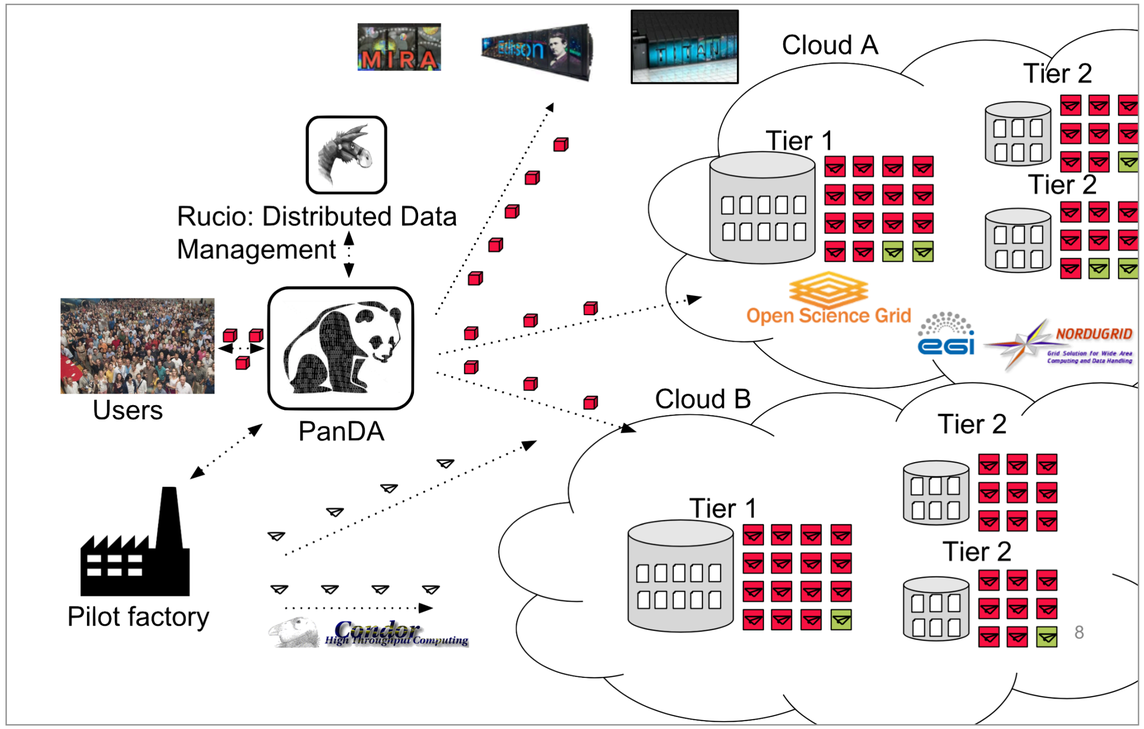
Harvester at OLCF

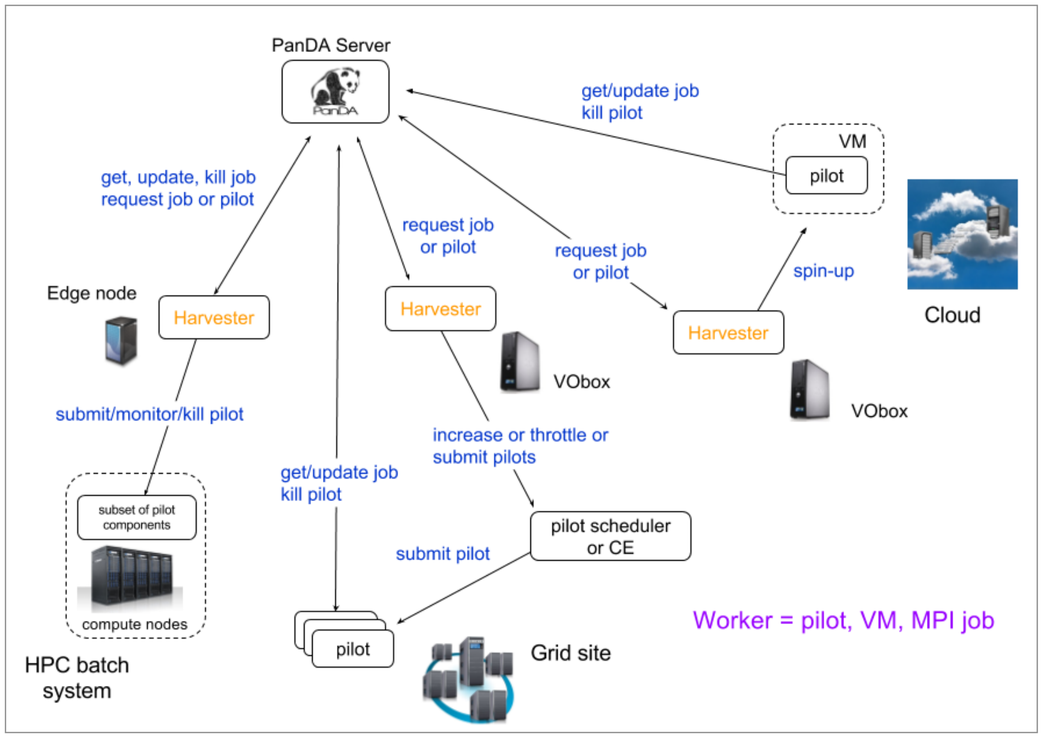
Overview

- Remainder: what is PanDA Harvester
- Harvester for HPC
- Harvester & Pilot
- OLCF solution
- Current status
- Nearest plans

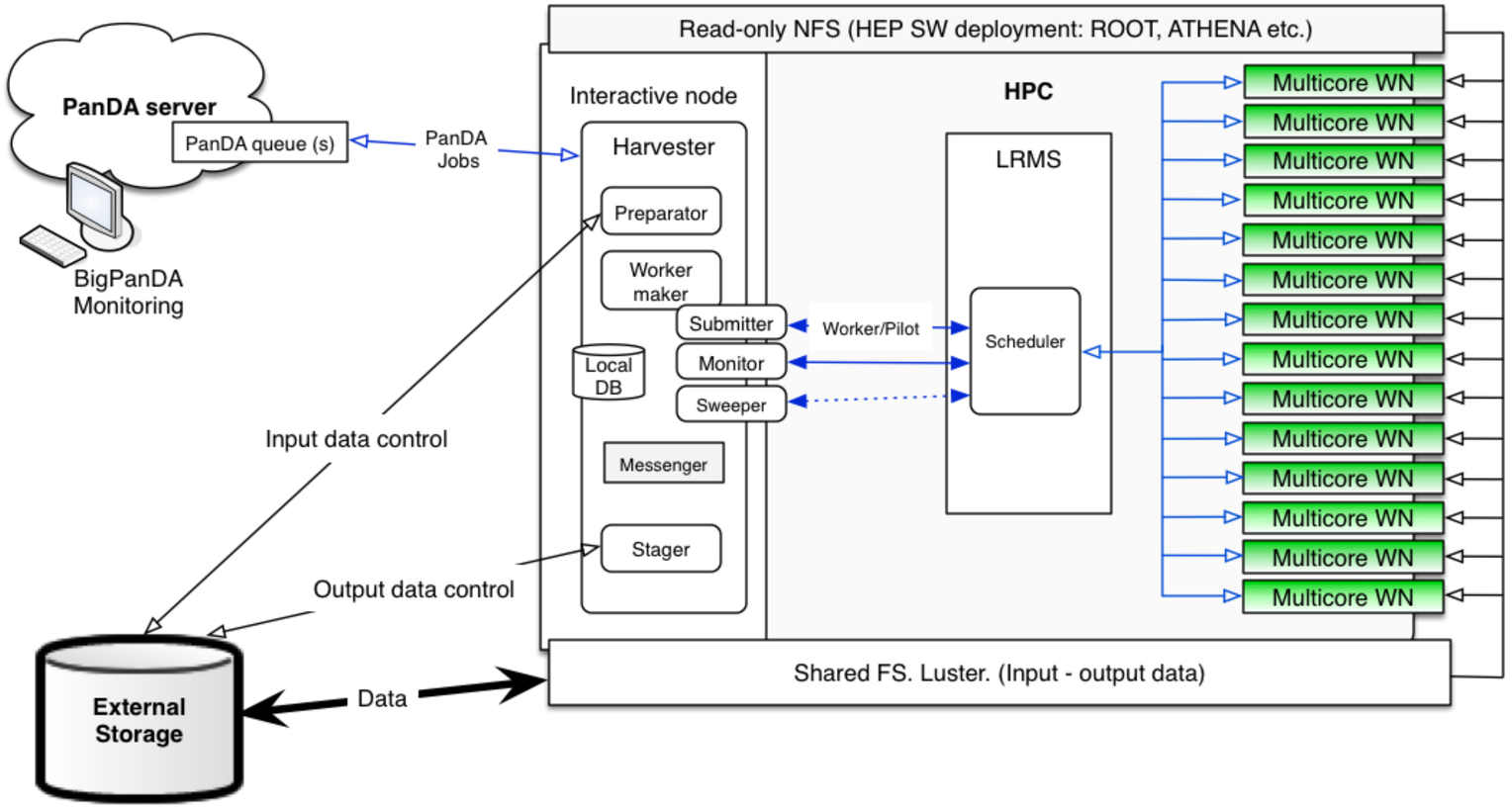
Remainder: what is PanDA Harvester

- *Harvester is a resource-facing service between the PanDA server and collection of pilots for resource provisioning and workload shaping. It is a lightweight stateless service running on a VObox or an edge node of HPC centres to provide a uniform view for various resources.*





Harvester for HPC



- Harvester is the service which manage a set of agents
- Launching of each agent is initiated by the state of PanDA jobs through Harvester core component
- Agents works simultaneously, and have no direct depending
- Agents interacts with third-party services through plug-ins
- Each plugin configurable
  - One Harvester instances may serves a set of PanDA queues
- Each PanDA queue may be configured independently (use own set of plugins)
  - Each PanDA queue may by served with different workflow

Agents & interfaces

- **Preparator:** prepare (stage in) data for jobs
- **WorkMaker:** setup pilot(s)
- **Submitter:** submit pilot(s) to the batch system
- **Messenger**: interface which responsible for interaction between pilot(s) and Harvester
- **Stager:** stage out data
- **Monitor:** monitor pilot(s) through scheduler
- **Sweeper:** kill pilots and cleanup sandboxes (working directories)

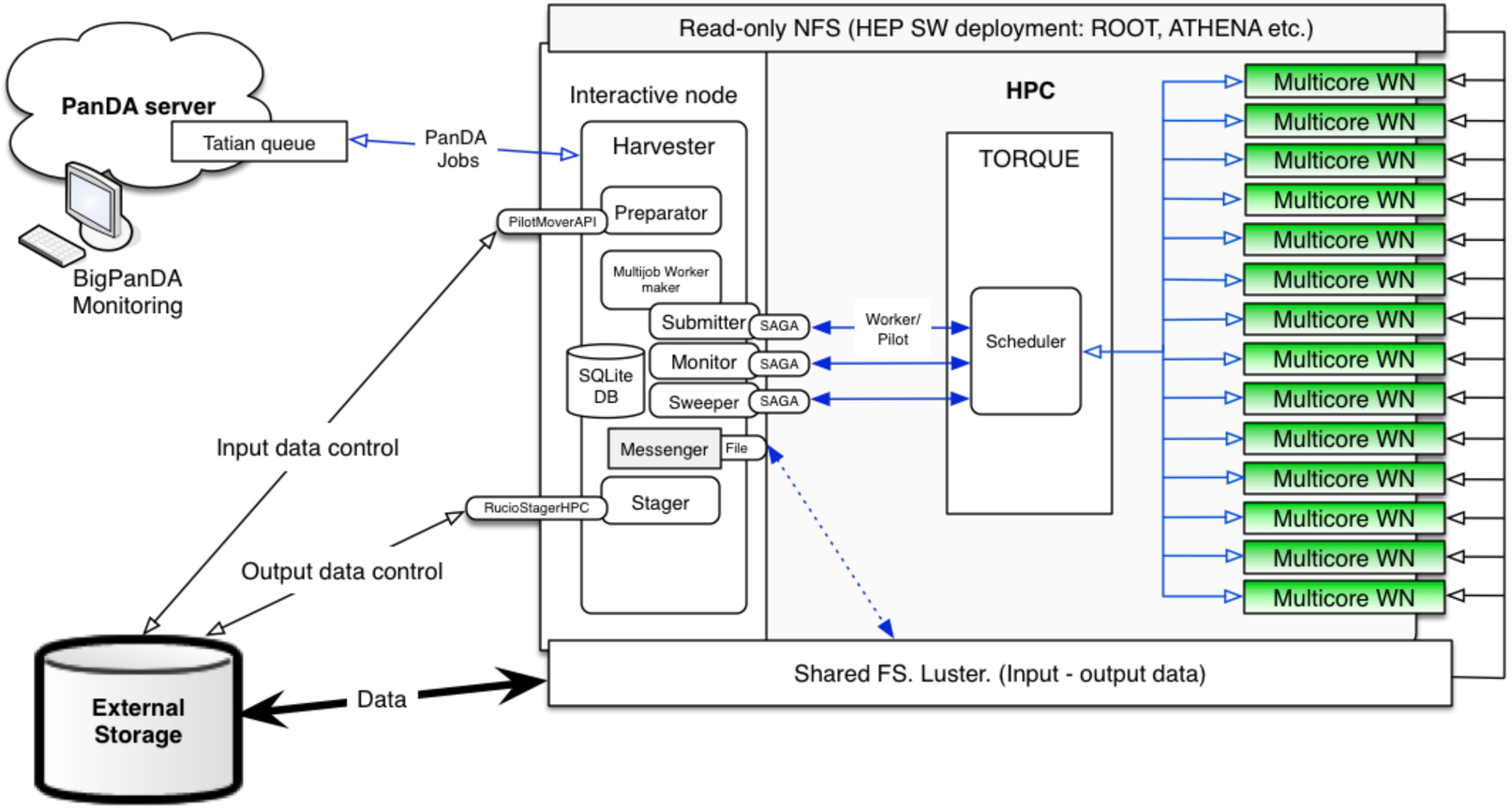
Deployment and configuration

- Harvester can be deployed by pip from github
  - With or without root privileges (through «virtualenv")
  - Most of dependencies will be installed automatically
- Configuration can be placed remotely and loaded via http(s)
  - Sensitive information like database password should be stored only in local config files
  - System config files are read only when the harvester instance is launched
  - Queue config files are read every 10 min so that queue configuration can be dynamically changed during the instance is running

Harvester & Pilot (2.0)

- Pilot is an application which supports execution of payload on working node
- PanDA Pilot is a complex application to support execution of payload on a grid working nodes:
  - It already support two workflows: 'regular job’ and ‘Event service job’
  - Communicate with PanDA server, perform stage in of input data, setup environment for payload, manage of execution of payload, perform stage out of data, analyze results of execution etc.
- However execution of payload on HPC require only some parts of functionality of PanDA Pilot and uses much simple, but different, workflows
  - Workflows can be different for different HPC
- Common functionality of PanDA Pilot available through API

OLCF solution



- Harvester deployed under virtualenv on dtn38
- SAGA plugin for Submitter, Monitor, Sweeper
- Plugin for Pilot movers for Preparator
- Multijob worker maker
- File plugin for Messenger
- Rucio plugin for Stager

Pilot for Titan

- Evolution of MPI wrapper which was used by Multijob Pilot
  - Common (simple) version in place
- Workflow for ATLAS payload more complicated and includes next steps:
  - environment setup, by calling special setup script with proper parameters
  - moving of input data and part of configuration data to RAM disk of WN
  - setup of working directory in RAM disk of WN
  - cleanup of working directory
  - publish results of execution (job report)
  - copy of working directory to Luster for future processing
- A lot of functionality used through Pilot API

Current status

- Harvester and additional components (SAGA, Pilot 2.0, Rucio clients) were successfully deployed in OLCF and launched under virtualenv
- Configured dedicated PanDA queues which used for testing
- Developed a set of plugins: Pilot Mover preparator, SAGA submitter, SAGA monitor, SAGA sweeper, MultoJob worker maker
- Implemented simple pilot application, more advanced version in process of testing
- Passed functional testing with generic jobs, testing with ATLAS jobs in progress

Plans

- Short term plan (next 2 month)
  - Pass pre-production testing and move ATLAS production through ALCC allocation to Harvester at OLCF
- Mid-term plan (next 4 month)
  - Extend Harvester with modules which will allow to shape payload to available backfill and move all current production to Harvester
- Long term plan:
  - Adapt Yoda in OLCF and integrate facility with ATLAS Event service