The next generation PanDA Pilot for and beyond the ATLAS experiment

Paul Nilsson[1], Alexey Anisenkov[2], Doug Benjamin[3], Daniel Drizhuk[4], Wen Guan[5], Mario Lassnig[6], Danila Oleynik[7,8], Pavlo Svirin[1] and Tobias Wegner[6], on behalf of the ATLAS Collaboration

[1]Brookhaven National Laboratory (US), [2]Budker Institute of Nuclear Physics (RU), [3]Argonne National Laboratory (US), [4]National Research Centre Kurchatov Institute (RU), [5]University of Wisconsin-Madison (US), [6]CERN - European Laboratory for Particle Physics, [7]University of Texas at Arlington (US), [8]Joint Institute for Nuclear Research (RU)

Abstract

The Production and Distributed Analysis system (PanDA) is a pilot-based workload management system that was originally designed for the ATLAS Experiment at the LHC to operate on grid sites. Since the coming LHC data taking runs will require more resources than grid computing alone can provide, the various LHC experiments are engaged in an ambitious program to extend the computing model to include opportunistically used resources such as High Performance Computers (HPCs), clouds and volunteer computers. To this end, PanDA is being extended beyond grids and ATLAS to be used on the new types of resources as well as by other experiments. A new key component is being developed, the next generation PanDA Pilot (Pilot 2). Pilot 2 is a complete rewrite of the original PanDA Pilot which has been used in the ATLAS Experiment for over a decade. The new Pilot architecture follows a component-based approach which improves system flexibility, enables a clear workflow control, evolves the system according to modern functional use-cases to facilitate coming feature requests from new and old PanDA users.
The paper describes Pilot 2, its architecture and place in the PanDA hierarchy. Furthermore, its ability to be used either as a command tool or through APIs is explained, as well as how its workflows and components are being streamlined for usage on both grids and opportunistically used resources for and beyond the ATLAS experiment.

Introduction

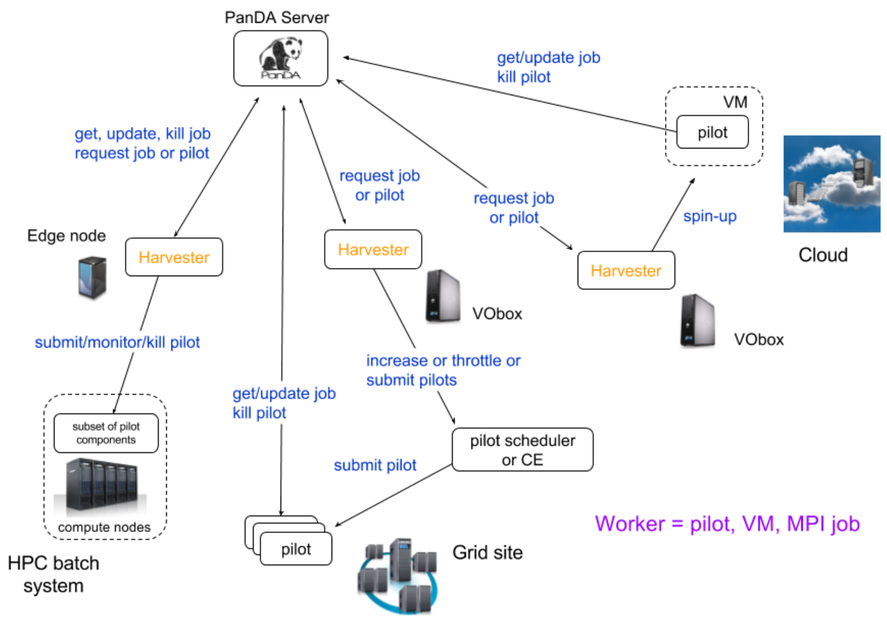
The PanDA Pilot has been used by ATLAS and other experiments for well over a decade. To meet the demands of extending PanDA beyond grids and ATLAS, the original Pilot is being rewritten. After a two-year effort, the Pilot 2 project is now in its final stages of development and has entered testing in the production system.

What does the PanDA Pilot do?

The task of the PanDA Pilot is to monitor and execute work units on a worker node, either on the job or event level. On the job level, the work unit is a payload that a user or production system wants to execute. The payload has certain requirements, e.g. input and output files, that are staged by the Pilot, and needs a working environment (incl. containers) that is setup by the Pilot. On the event level, the Pilot launches and feeds a payload with event ranges (a set of events to be processed) downloaded from a server.

How does the Pilot fit into the PanDA hierarchy?

The PanDA Pilot is executed on the worker nodes on local resources, on grids and clouds, on HPCs and on volunteer computers. It is downloaded and run by wrapper scripts that are sent by Pilot factories to the worker nodes via batch systems. A Pilot interacts with the PanDA server either directly, via a local instance of the ARC Control Tower (a job management framework used on Nordugrid) or with the resource-facing Harvester service (which provides resource provisioning and workload shaping).



Pilot components

The Pilot is component based, with each component being responsible for different tasks. The main tasks are sorted into controller components, such as Job Control, Payload Control and Data Control. There is also a set of components with auxiliary functionalities, e.g. Pilot Monitor and Job Monitor - one for internal use which monitors threads and one that is tied to the job and checks parameters that are relevant for the payload (e.g. size checks). The Information System component presents an interface to a database containing knowledge about the resource where the Pilot is running (e.g. which copy tool to use and where to read and write data).

Pilot APIs

Normally, the PanDA Pilot is used as a command-line tool. In case this is not wanted but some Pilot functionality is still needed, an external user may use relevant functions via Pilot APIs that provide convenient access to internal Pilot modules. E.g. Harvester is using the Data API in production for data transfers on HPCs. Other APIs include the Communicator API (server interactions) and Services API (benchmarking and memory monitoring).

Data API

```
StageInClient()
.transfer()
StageOutClient()
.transfer()
```

Communicator API

```
JobClient()
.get_job()
.get_jobs()
.update_job()
.update_jobs()
.get_event_range()
.update_event_range()
```

Services API

```
Benchmark()
.get_command()
.get_filename()
.get_results()
MemoryMonitor()
[same as Benchmark()]
.get_linear_fit()
```

Pilot workflows

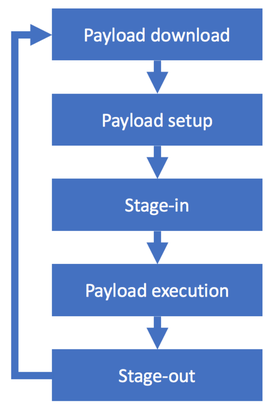
A Pilot workflow determines how a job will be processed. A workflow can consist of multiple steps executed in parallel using threads. Each thread polls a queue until it gets a **job object** to process; after processing, the result is put in another queue for further processing and the thread starts polling its input queue again.

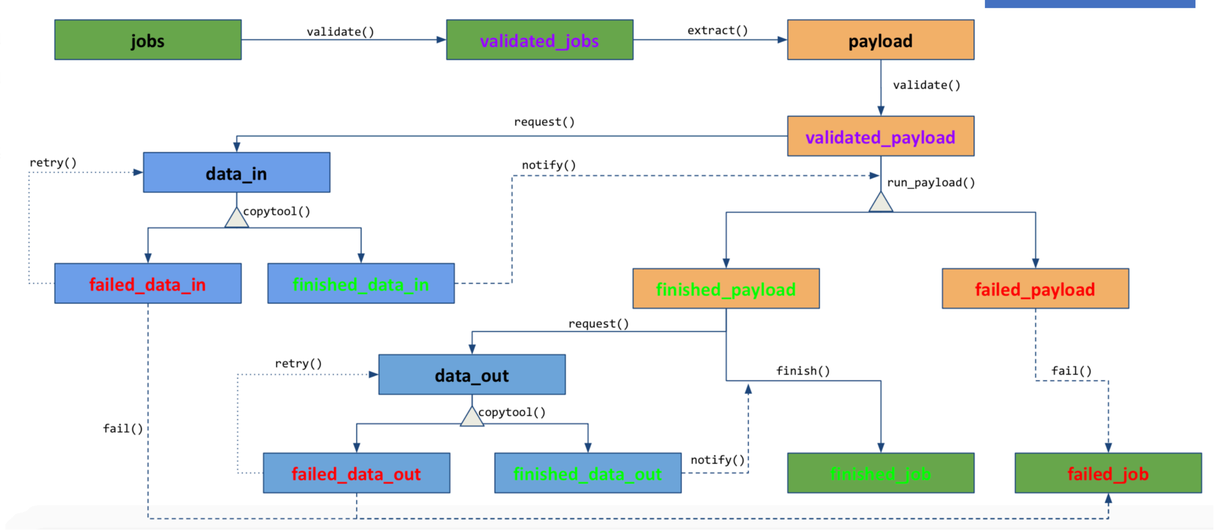
A **job object** is an entity that contains all necessary information about running the payload (e.g. software release version, parameters for payload setup, transfer type of input files, etc).

* Standard workflow

In the standard workflow, the Pilot performs payload download; setup; stage-in; execution; stage-out, along with various verifications, monitoring and server job updates.

The figure below shows the internal generic flow of job objects and how they move through the queues in the Pilot components. Each box represents a queue.





* HPC Pilot workflow

The HPC Pilot refers to a dedicated workflow used on HPCs. When this is selected (via a Pilot option), the normal workflow of the Pilot is skipped in favour of a streamlined workflow that is relevant for HPCs. Resource specific code, such as environmental setup, is kept in plugins.

Generalized workflow on a typical HPC

- Collecting the job definition from a pre-placed JSON file
- Preparing for execution: pre-placement of input data in high speed transient storage associated with the computing node; RAM-disk, Burst Buffer, SSD
- Environment setup; incl. platform specific preparations
- Payload execution
- Publishing of job report in a JSON file, which includes state of payload and other metrics
- Processing of job report
- Transfer of output data to shared file system
- Archiving and storing payload logs to shared file system
- Declaring files for later stage-out

Beyond ATLAS

The PanDA system is currently used by several experiments including ATLAS, Compass, LSST, LQCD, Molecular Dynamics, IceCube. One of the design goals of the Pilot 2 project is to facilitate Pilot development and usage by new users (experiments).

Since the Pilot keeps user specific code in plugins, as well as being a component based system, it is easy to support new workflows. In case the standard workflow (see above) is not relevant for the new user, an entirely new workflow may be implemented that will only use other relevant Pilot modules and functions via the APIs.