Global heterogeneous resource harvesting:
The next-generation PanDA pilot for ATLAS

Paul Nilsson (1), Alexey Anisenkov (2,3), Daniel Drizhuk (4), Wen Guan (5), Mario Lassnig (6), Danila Oleynik (7), Pavlo Svirin (1) for the ATLAS Experiment

(1) Brookhaven National Laboratory, Upton, USA (2) Budker Institute of Nuclear Physics, Novosibirsk, Russia (3) Novosibirsk State University, Novosibirsk, Russia (4) National Research Centre "Kurchatov Institute", Moscow, Russia (5) University of Wisconsin-Madison, Chamberlin Hall Madison, USA (6) CERN, Geneva, Switzerland (7) University of Texas at Arlington, USA / Joint Institute of Nuclear Research, Russia

The Production and Distributed Analysis system (**PanDA**), used for workload management in the **ATLAS Experiment** for over a decade, has in recent years expanded its reach to diverse new resource types such as **HPCs**, and innovative new workflows such as the event service. PanDA meets the heterogeneous resources it harvests in the **PanDA Pilot**, which has embarked on a next-generation reengineering to efficiently integrate and exploit the new platforms and workflows. The new modular architecture is the product of a year of design and prototyping in conjunction with the design of a completely new component, **Harvester**, that will mediate a richer flow of control and information between Pilot and PanDA. While the traditional task of the Pilot is to execute payloads on a grid worker node, the introduction of Harvester makes it simpler to approach complex systems like HPCs. In those cases, some Pilot responsibilities can be moved to Harvester and others can be made available to Harvester via Pilot APIs.

Modular architecture

The architecture of the new Pilot version is based on plugins, which help it to be highly customizable. Usage of plugins makes it simpler to add support for new users of the system. Furthermore, with the modular approach it is simpler to add new data copying tools and different types of execution methods, including various supercomputers and containers.

Extended documentation

All the Pilot components as well as usage and APIs are fully documented in the code as well as in an extended additional usage documentation provided for all exposed interfaces. The information is open and is accessible through the project repository documentation page.

Highly configurable

The new Pilot is configurable on several levels. It fetches configurations from the environment, from global configs, and several other locations. It takes into account which environment it is running in, the experiment or user it serves, the task or queue to which it was assigned, as well as many other parameters.

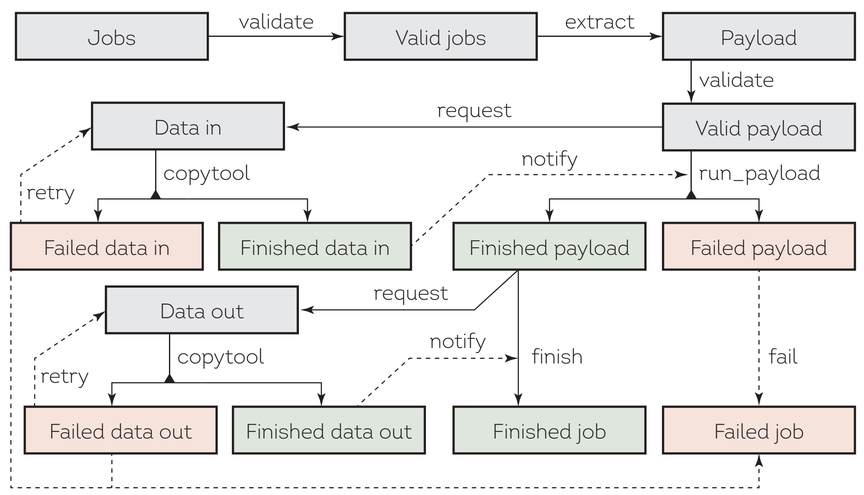
Container support

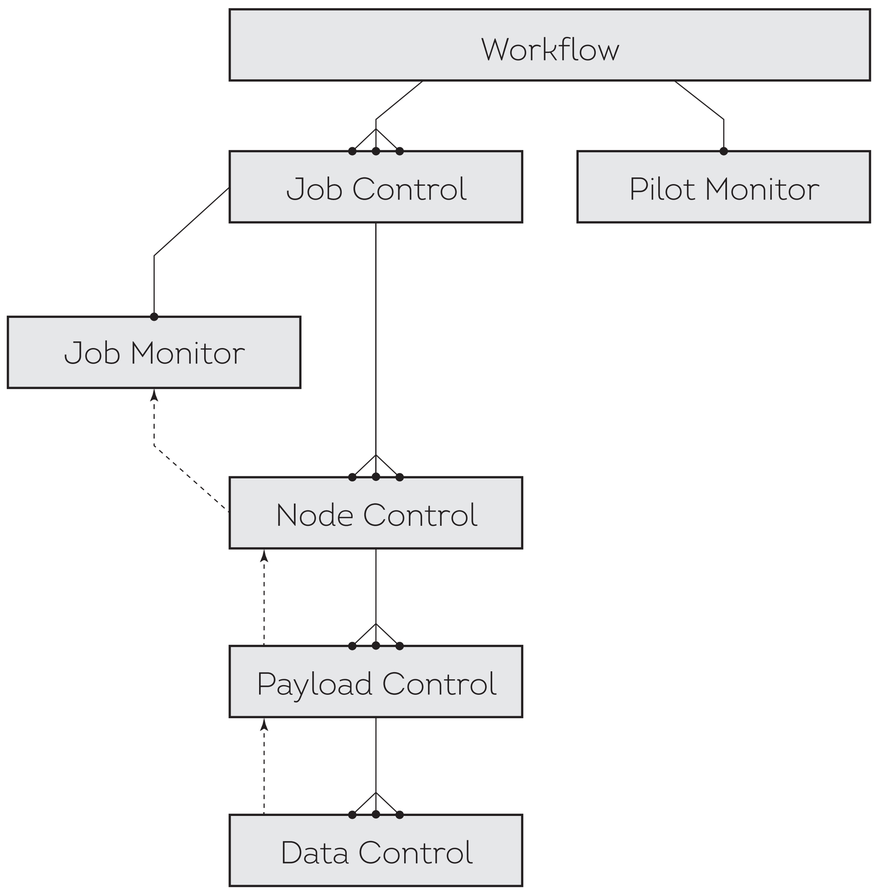
The new Pilot has support for containers on multiple levels: as a method of executing payloads and related file transfers, as well as its own environment where the Pilot and all subprocesses are executed within a single container. It provides users with an easy way of deploying an indefinite number of tasks in the exact environment they need.

Easy to extend

The plugins API enables an easy way to develop new workflows and new applications of the Pilot, provide new monitoring, task executing or management technologies. Plugins are simple to use, extendable and easy to deploy. There is no need for users to modify the core Pilot code or its main components since the plugin mechanism enables high-level tweaking of the workflow for specific cases or environments.

Internal workflow





APIs

APIs for third-party applications are provided. A **Data API** contains the interface to the copy tools supporting synchronous and asynchronous stage-in/out using rucio, xrdcp, gfal-copy, locally defined copy tools, and more. The **Communicator API** provides functions for interacting with external servers or services such as the PanDA server, the ARC Control Tower and Harvester. Finally, the **Environment API** has an interface to the job execution environment used on HPCs.

Monitoring

The Pilot has extendable monitoring using plugins. A monitor can be created for the resources the task is operating with and recent changes can be followed via the task manager without waiting for the task to finish. It can monitor the health of the task, its resource usage, or trace any arbitrary parameter and forward it to the PanDA server, which in turn makes the information available to the end user via the PanDA Monitor.

Resource harvesting

The Pilot may interact with another new PanDA product, **Harvester.** This is a service between the PanDA server and HPCs that provides resource provisioning and workload shaping. It enables more intelligent and dynamic task matching with the resources, simplifying the operator and user view of a PanDA site but internally utilizing various kinds of information about the resource and a site's capabilities to boost the performance of tasks as well as sites.

Supercomputer support

The modular architecture and abstraction layer through plugins allow the new Pilot to work with different computational infrastructures including massively parallel systems or supercomputers through interfaces to batch systems and executors.
Another PanDA product, **Harvester,** provides an efficient usage of computing resources in general and especially on HPCs. Harvester on HPCs uses Pilot functionality through the Pilot API, and allows for more complicated workflows than the Pilot can do alone.