Harnessing the power of supercomputers using the PanDA Pilot 2 in the ATLAS Experiment

Paul Nilsson[1,*], *Alexey* Anisenkov[2,3], *Doug* Benjamin[4], *Wen* Guan[5], *Tomas* Javurek[6], and *Danila* Oleynik[7,8]

[1]Brookhaven National Laboratory, Physics Department, United States
[2]Budker Institute of Nuclear Physics, Russia
[3]Novosibirsk State University, Russia
[4]Argonne National Laboratory, United States
[5]University of Wisconsin-Madison, Department of Physics, United States
[6]CERN, European Laboratory for Particle Physics, Switzerland
[7]University of Texas at Arlington, Department of Physics, United States
[8]Joint Institute for Nuclear Research, Russia

Abstract. The unprecedented computing resource needs of the ATLAS experiment at LHC have motivated the Collaboration to become a leader in exploiting High Performance Computers (HPCs). To meet the requirements of HPCs, the PanDA system has been equipped with two new components; Pilot 2 and Harvester, that were designed with HPCs in mind. While Harvester is a resource-facing service which provides resource provisioning and workload shaping, Pilot 2 is responsible for payload execution on the resource. The presentation focuses on Pilot 2, which is a complete rewrite of the original PanDA Pilot used by ATLAS and other experiments for well over a decade. Pilot 2 has a flexible and adaptive design that allows for plugins to be defined with streamlined workflows. In particular, it has plugins for specific hardware infrastructures (HPC/GPU clusters) as well as for dedicated workflows defined by the needs of an experiment. Examples of dedicated HPC workflows are discussed in which the Pilot either uses an MPI application for processing fine-grained event level service under the control of the Harvester service or acts like an MPI application itself and runs a set of job in an assemble. In addition to describing the technical details of these workflows, results are shown from its deployment on Titan (OLCF) and other HPCs in ATLAS.

1 Introduction

In the coming High-Luminosity Large Hadron Collider (HL-LHC) era, both ATLAS [1] and CMS [2] experiments are expected to produce up to an order of magnitude more data compared to the previous data taking. Storage and computing needs continue to grow at a much higher rate than what the flat budgets allow for. However, at the same time, the IT

* Corresponding author: Paul.Nilsson@cern.ch

landscapes, computing infrastructures and funding models are changing. The national science programmes are consolidating computing resources and encourage using cloud services as well as HPCs. ATLAS has been using heterogeneous resources for many years, and has successfully integrated multiple grids, clouds, and HPCs into its workflow management system. But heterogeneous resources are not always tailored for ATLAS workloads. New technologies have therefore recently been introduced to address and overcome some of the more challenging limitations while at the same time reducing operational manpower.

1.1 Paradigm shift

ATLAS has used PanDA [3] as the workload management system to control all the resources. Generic factories running on the PanDA servers submit Pilot wrapper scripts to the batch systems. When such a wrapper occupies a slot, it will download and launch the actual Pilot code. The Pilot asks the PanDA server for a suitable job, which is then downloaded, executed and monitored throughout its lifetime. Relevant metrics are reported back to the server at regular intervals.

The Server-Pilot paradigm worked well for non-stop running on the grid with 250k+ cores for well over a decade, but on heterogeneous resources such as HPCs it worked less well. Different edge services and operational policies at the different HPCs led to major challenges with an already aging Pilot architecture, which led to different Pilot versions running on the grid and HPCs. There were also too many manual interventions needed to effectively fill the available CPU resources. To overcome these challenges, new technical solutions were needed and in 2016, the PanDA team launched the Harvester [4] and Pilot 2 [5] projects.

2 Harvester in the PanDA system

Harvester is a resource-facing service between the PanDA server and Pilots, and is used for resource provisioning and workload shaping. It is a lightweight stateless service that is running on a VObox or on an edge node on HPC centers to provide a uniform view for various resources. Harvester has a modular design for different resource types and workflows, and has a coherent implementation for HPCs. It provides better resource monitoring, timely optimization of CPU allocation among various resource types and removal of batch-level partitioning. The Harvester service enables a tight integration between the PanDA system and the resources needed for new workflows.

3 Pilot 2

Pilot 2 (henceforth referred to as “Pilot”) is a complete rewrite of the original PanDA Pilot [6] which was used in ATLAS for over a decade. Its architecture follows a component based approach which evolves the system according to modern functional use-cases, to facilitate coming feature requests from the PanDA users. It also has improved system flexibility and has enabled a clear workflow control.

ATLAS requires that all payloads are executed in containers. This is achieved either by launching the entire Pilot in a container or by having the Pilot execute the payload in a container, which is more relevant since different payloads may require different container images which is only known after the Pilot has downloaded a job from the server. The container images are currently stored on CVMFS and DockerHub.

While any part of the code can be imported by an external user, some functionality can be accessed by simplified Pilot APIs that have been streamlined for external use to bypass complex function calls or data structures that otherwise would need to be defined. For example, the data API is used by Harvester for file transfers on and off HPCs.

The Pilot supports different internal workflows such as normal grid mode, HPC mode and a stage-in mode (in development). Additional workflows can be designed, with all user specific details (e.g. ATLAS) kept in plugins and relevant code can be pulled in from the Pilot API, which means that the Pilot is a powerful package that can be tailormade for any user as well as for new workflows. The HPC workflows are described in more detail in the next section.

3.1 HPC workflows

Depending on the type of the HPC, the Pilot may either run payloads on the worker nodes like on any grid site, or in a simplified mode via a plugin. Grid-like execution is used when an HPC provides external connectivity on the worker nodes and enables access to external software areas. Limited connectivity execution is used when the worker nodes do not have outbound network access, in which case a plugin may be needed. On the Titan supercomputer at OLCF, a plugin was developed where the Pilot acts like an MPI application under the control of Harvester and runs a set of jobs in an assembly. In this case, Harvester takes care of prestaging input data, stage-out of output (using the Pilot data API) and handles all communications with the PanDA server. The Pilot intercommunicates with Harvester via the shared filesystem and reads the job definitions from pre-placed files and declares outputs for later processing by Harvester. Results from Titan, which is now retired, is discussed below. For Summit, the new supercomputer at OLCF, as well as for other HPCs, the approach is different. The main workflow is limited connectivity execution and HPC specifics are still being put in plugins (minor details only), and with the Pilot running in full mode.

Another new workflow in active development is to support the new Raythena [7] tool, which is based on Ray [8], and will be used for running ATLAS Event Service [9] on HPCs. In this case, a Ray driver is in charge of communications with Harvester using the shared file system. A Ray actor communicates with the Pilot which is running the actual payload using HTTP. With the Pilot running in full mode on the worker node, we get all the benefits and functionalities provided by the Pilot, especially job monitoring and error handling.

4 HPCs in ATLAS

ATLAS was able to utilize about half a billion hours of walltime usage on HPCs during 2017/2018 and 0.34 billion hours during 2019 on 20+ HPCs (Figure 1). Pilot 2 was used either directly with Harvester (on especially US HPC resources) or in combination with the ARC Control Tower [10] on grid-like HPC resources, and other HPCs that do not use Harvester directly.

During the production phase (2016-2019), 1.4 billion events were simulated on Titan in backfill mode (Figure 2). A tailormade version of the original PanDA Pilot was used between 2016-mid 2018, while the new Pilot 2 version was used from mid 2018 until Titan was retired in 2019. Support for multiple HPCs are in active development (e.g. on Summit and NERSC with successful test jobs already run).

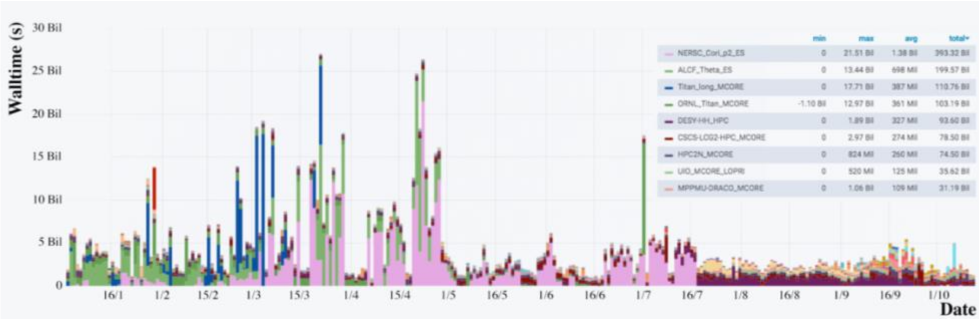

Fig. 1. Walltime consumption in seconds during 2019 for jobs running on all HPCs used in ATLAS (list of HPCs is cut).

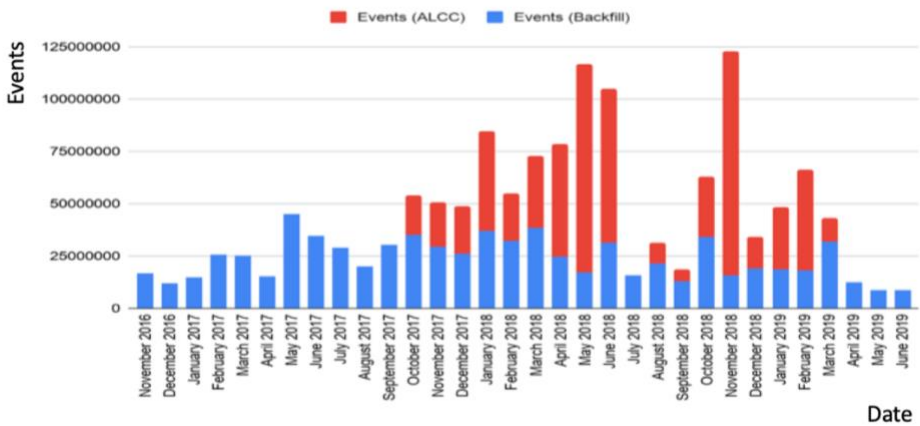

Fig. 2. Events produced on the Titan supercomputer between 2016 and 2019 (as part of the Advanced Scientific Computing Research Leadership Computing Challenge (ALCC)). Red means allocation mode (using static and long running jobs) and blue means backfill mode (with dynamic shaping of job sizes).

4.1 ATLAS and the next generation HPCs

The next generation of HPCs are evolving from pure computational facilities to resources for extreme data processing; e.g. Big Data, High Performance Data Analysis and Artificial Intelligence (Machine and Deep Learning). The architectures for these HPCs will be very different from the usual grid sites. The primary data storage will be large shared file systems. They will have burst buffers; fast NVMe, SSD dedicated storage for data processing to cope with I/O intensive payloads; nodes with over 100 cores with limited memory (1 GB per core or less); and limited outbound throughput on the nodes. Most of the computing power will be provided though heterogeneous architectures mixing CPUs and GPUs or FPGA accelerators (US and EU).

ATLAS C/C++ code was originally designed and developed for single x86 CPU cores, but has been modified for multicore CPUs. The code can be recompiled for other CPU platforms as well, such as ARM and Power9. However, to use the large number of flops available on GPUs at the HPC centers, some of the kernel code needs to be reengineered and new algorithmic code must be developed.

5 Conclusions

Compared to present ATLAS levels, an order of magnitude more data is expected in the coming HL-LHC era. Innovation is ongoing to further improve the PanDA system with a strong focus on supporting workflows on HPCs. ATLAS has been using HPCs in production for many years, primarily for Monte Carlo simulation. ATLAS was able to utilize almost a billion hours of walltime usage on HPCs during 2017-2019. To handle this huge increase in usage, numerous innovations and improvements needed to be done. The new tools Harvester and Pilot 2 were designed with HPCs in mind. Harvester is a resource-facing service between PanDA Server and Pilots for resource provisioning and workload shaping, while Pilot 2 is a complete rewrite of the original PanDA Pilot which was used in the ATLAS experiment for over a decade, and is responsible for executing payloads on the worker nodes. Pilot 2 has so far been used on 15 HPCs with Harvester/ARC Control Tower and on five with local Harvester. Fully optimized use of existing and future HPC facilities is a long-term goal which ATLAS is consistently working towards.

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of High Energy Physics, under contract number DE-SC0012704, and was funded in part by the Russian Ministry of Education and Science under contract No 14.Z50.31.0024.
This research used resources of the National Energy Research Scientific Computing Center, a U.S. Department of Energy Office of Science User Facility operated under Contract No. DE-AC02-05CH11231.
This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.
This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

References

1. ATLAS Collaboration, JINST **3** S08003 (2008)
2. CMS Collaboration, JINST **3** S08004 (2008)
3. T. Maeno et al., J. Phys. Conf. Ser. **331** 072024 (2011)
4. T. Maeno et al., 23rd International Conference on Computing in High Energy and Nuclear Physics, EPJ Web of Conferences, Volume **214** 03030 (2019)
5. P. Nilsson et al., 23rd International Conference on Computing in High Energy and Nuclear Physics, EPJ Web of Conferences, Volume **214** 03054 (2019)
6. P. Nilsson et al., J. Phys.: Conf. Ser. **513** 032071 (2014)
7. M. Muskinja et al., These proceedings (2019)
8. P. Moritz et al., Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation, 561 (2018)
9. P. Calafiura et al., J. Phys.: Conf. Series **664** 092025 (2015)
10. M. Ellert et al., Future Generation Computer Systems **23** n.2 219-240 (2007)