EPJ Web of Conferences **214**, 03047 (2019) https://doi.org/10.1051/epjconf/201921403047
CHEP 2018

The Future of Distributed Computing Systems in ATLAS: Boldly Venturing Beyond Grids

Fernando Barreiro[1,*], *Doug* Benjamin[2], *Taylor* Childers[2], *Kaushik* De[1], *Johannes* Elmsheuser[3], *Andrej* Filipcic[4], *Alexei* Klimentov[3], *Mario* Lassnig[5], *Tadashi* Maeno[3], *Danila* Oleynik[1], *Sergey* Panitkin[3], *Torre* Wenaus[3] *on behalf of the ATLAS collaboration*

[1]1University of Texas at Arlington, USA

[2]Argonne National Laboratory, USA

[3]Brookhaven National Laboratory, USA

[4]Jozef Stefan Institute, Slovenia

[5]European Center for Nuclear Research, Switzerland

Abstract. Since 2010 the Production and Distributed Analysis system (PanDA) for the ATLAS experiment at the Large Hadron Collider has seen big changes to accommodate new types of distributed computing resources: clouds, HPCs, volunteer computers and other external resources. While PanDA was originally designed for fairly homogeneous resources available through the Worldwide LHC Computing Grid, the new resources are heterogeneous, at diverse scales and with diverse interfaces. Up to a fifth of the resources available to ATLAS are of such new types and require special techniques for integration into PanDA. In this talk, we present the nature and scale of these resources. We provide an overview of the various challenges faced, spanning infrastructure, software distribution, workload requirements, scaling requirements, workflow management, data management, network provisioning, and associated software and computing facilities. We describe the strategies for integrating these heterogeneous resources into ATLAS, and the new software components being developed in PanDA to efficiently use them. Plans for software and computing evolution to meet the needs of LHC operations and upgrade in the long term future will be discussed.

1 Motivation

The ATLAS [1] experiment at the Large Hadron Collider (LHC) has an ambitious programme moving towards the High Luminosity (HL) LHC era in the coming decades. Data volumes will increase at higher energy and luminosity, causing the storage and computing

* Corresponding author: barreiro [at] uta.edu

needs to grow at a much higher pace than the flat budget technology evolution (see Figure 1).

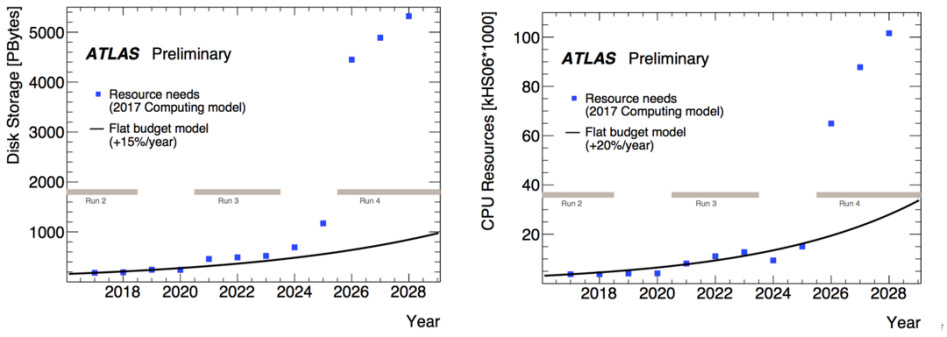

Fig. 1. Estimation of ATLAS computing needs vs predictable increase on a flat budget*

At the same time IT landscapes, computing infrastructures and funding models change. National science programs are consolidating computing resources and encouraging the usage of national HPCs. Outsourcing to Cloud Computing providers is also becoming a feasible solution for institutes with smaller computing clusters.

ATLAS is not new to using various heterogeneous resources and has successfully integrated Grid, HPC and Clouds. As shown in Figure 2 for a random week of 2018, the Grid is still the largest contributor, but the contribution of other resources is already significant. However, heterogeneous resources are not always tailored for ATLAS workloads and have been adapted independently by different ATLAS collaborators. Based on the accumulated experience, it is now a good moment to harmonize the adaptations and overcome some of the most challenging limitations while reducing the operational manpower.

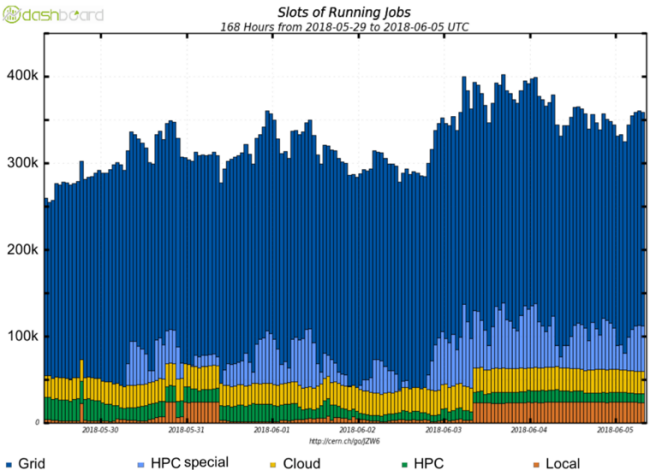

Fig. 2. Breakdown of ATLAS computing resources by type. The plot illustrates the important contribution of HPC and Cloud Computing (at the order of 100k cores)

*https://twiki.cern.ch/twiki/bin/view/AtlasPublic/ComputingandSoftwarePublicResults

2 Development

2.1. Revised PanDA architecture: Server - Harvester - Pilot

During the first decade of operation, ATLAS has relied on PanDA[2] as the workload management system to control all resources. Generic factories would submit the PanDA pilot[3] to the batch systems. The pilot would occupy the slot, ask PanDA server for a job, monitor the job through its lifetime and report back relevant metrics to the server.
This approach was satisfactory on the Grid, but as the heterogeneity of resources grew, the number of specialized factories and different solutions grew over the years. Direct dialog with the PanDA infrastructure and advanced features requiring direct PanDA integration were not supported.

The Harvester [4] implementation is filling this gap by providing the implementation of a generic core and only requiring resource specific plug-ins. This approach reduces drastically the development and integration time for new resources, allowing to make better use of short-lived offerings. In the general case, where Harvester can manage the resources centrally, the new resource can be configured into the existing infrastructure and avoids the proliferation of different services that need to be operated.

To date, Harvester has been interfaced to multiple Cloud resources (Google Compute Engine, OpenStack), several US DOE HPCs (Titan[5], Theta[6], Cori[7]) and is being rolled out on Grid sites (all tested CEs except Globus Toolkit 5 [8] are supported).
The next sections will highlight integration examples beyond the Grid with some key IT players and explain some of their complexities.

2.2. High Performance Computers

2.2.1. HPC background and requirements

HPCs, and Supercomputers as their most powerful representatives, are large computing infrastructures, architected to execute massively parallel applications, whilst following strong design constraints like security, energy consumption and cooling. Each HPC is designed independently and has its own set of architectures and restrictions. Coming up next is a list of important aspects that need to be considered for each integration:

- **Deployment models**: Harvester has versatile deployment models, adapting to the security and load constraints of each HPC. As such, Harvester can run on the edge node in a very lightweight mode, or be a more heavyweight central service.
- **Different operating systems**
- **Different software installation options**: typically, the ATLAS Software is distributed through the Grid via CVMFS [9], a distributed, read only file system. The usual solutions on HPCs are customized CVMFS setups, containers with pre-installed (usually Monte Carlo) ATLAS Software or manual installation on the distributed file systems of the HPC.
- **Processor architectures:** ATLAS Software is currently compiled and tested for x86 CPU architectures. HPCs are increasingly moving towards different architectures. One example is Summit, the new Supercomputer at Oak Ridge National Laboratory, coming live in 2018 with IBM Power9 processors. Therefore, it is crucial to have an ATLAS Software compilation methodology [10]. In addition, HPCs are having an increasing presence of GPU co-processors, e.g. NVIDIA Volta V100s in the case

of Summit. Given the effort required of porting ATLAS Software to GPUs and the lack of manpower, these co-processors currently remain idle. In order to remain attractive HPC users and fully utilize ATLAS HPC allocations, making use of the GPUs will become more and more important, either by porting the software or implementing new workflows.

- **Nodes without disk:** Most HPCs provide diskless nodes on top of a shared file system, e.g. Lustre. Concurrent file access can become a bottleneck in situations, where hundreds of nodes read the ATLAS software simultaneously during startup. Those I/O operations need to be optimized and, where possible, moved to solutions like RAM disks.
- **Data management:** There is not always a gridFTP-compatible Storage Element available at the HPCs and a large fraction of the HPC architectures block any external connectivity from the nodes, preventing remote read/write to a Storage Element on the grid. Data management solutions therefore need to provide alternatives such as asynchronous download to/upload from the shared file system or third-party transfers either through FTS [11] or Globus Online [12]. Additional security constraints like 2-factor authentication make it harder to converge on a single solution.

2.2.2. HPC vs ATLAS workloads

HPC allocations are usually awarded by millions of node-hours over a period of months. In order to make use of the allocation, users have to submit multi-node requests to reserve a fraction of the HPC. HPC internal scheduling algorithms prioritize large jobs and fill the gaps with smaller workloads to optimize the usage of their infrastructures. As shown in Figure 3, the Titan scheduling policies determine the maximum allowed wall time depending on the size of the job: large jobs are allowed to run longer. Depending on the HPC utilization, queueing times can be long, e.g. several weeks. There will be gaps between the large requests, potentially available as backfill and allowed to execute for short periods, e.g. 2 hours.

Bin	Min Nodes	Max Nodes	Max Walltime (Hours)	Aging Boost (Days)
1	11,250	–	24.0	15
2	3,750	11,249	24.0	5
3	313	3,749	12.0	0
4	126	312	6.0	0
5	1	125	2.0	0

Fig. 3. Example for Titan (Oak Ridge National Laboratory) scheduling policies. The maximum walltime increases with the number of the nodes in the slot. Large slots also get an aging boost (as if they would have been waiting longer in the queue) to prioritize them.

On the contrary, ATLAS workloads are loosely coupled. An ATLAS job needs 1-16 cores (parts of a node) and 2-4 GB RAM/core. It executes for several hours to process few hundred events in a file. Hence multiple ATLAS jobs need to be combined into a single HPC multi-node workload.

ATLAS multi-node submissions need to maximize the efficiency: all nodes in the slot need to be active during the slot's lifetime. Efficiency penalties occur when submitting together jobs of different durations, or one of the jobs misbehaving and runs longer than expected (see Figure 4).

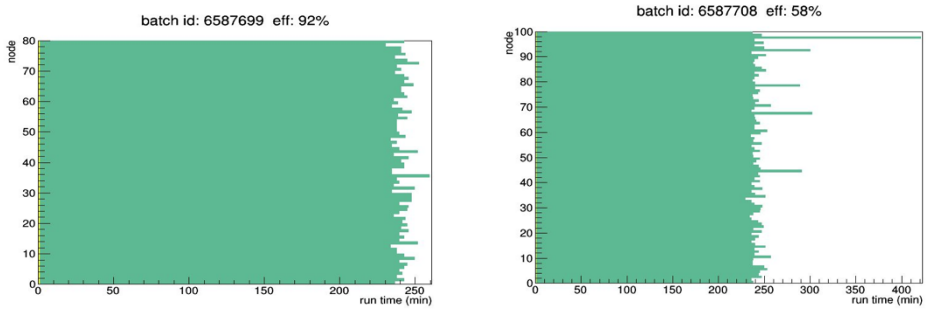

Fig. 4. Examples of jobs with roughly same duration (left) vs problematic job holding up the slot (right).

The aforementioned conditions are important for the selection of ATLAS jobs submitted to HPCs. During first R&D phases tasks were manually selected and tailored for HPCs, but it is important to automate this step and reduce the operational load. The solution being commissioned uses Jumbo Jobs. Jumbo jobs package together multiple related jobs and manage the Jumbo Job at a event-level granularity. Multiple consumers can pull event ranges from the Jumbo Job and process these in parallel. The ATLAS consumer developed for HPC is called Yoda [13] (see schematic on Figure 5 a). Yoda works together with Harvester, pulls the event ranges and feeds these to the lower ranks through the Message Passing Interface standard [14].

Figure 5 b shows the execution of ATLAS event-level jobs inside a node across the 128 cores. The nodes are kept busy continuously. The only inefficiency occurs to the jobs that didn't complete immediately before returning the node.

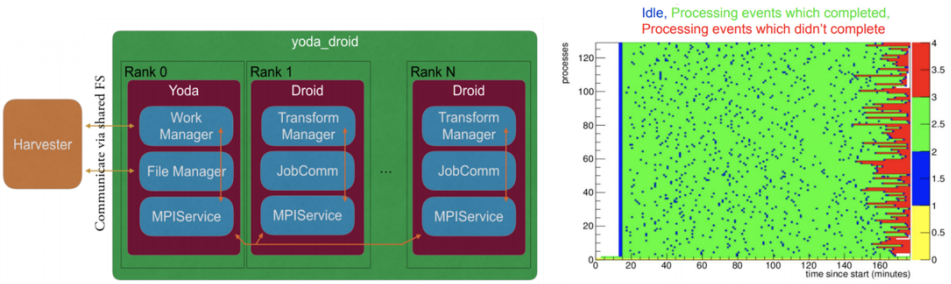

Fig. 5. a) Yoda schematic. b) Example of event level processing inside a node

2.2.3. HPC status and results

Harvester and specific plugins have been implemented for US DOE HPC (Theta, Cori, Titan) usage. The allocations available to ATLAS on these HPCs have been successfully and timely exploited. While some initial customization for each HPC is unavoidable, Harvester has been equipped with plugins to handle the data management (stage-in and stage-out of data to the HPC), communicating with the HPC batch interface and handle advanced workloads through the usage of Yoda.

2.3. ATLAS-Google project [15]

2.3.1. Integration model

Beginning of 2018 ATLAS and Google have launched a common project with a first goal to fully integrate both data and workload management with Google Cloud Platform. Rucio, the ATLAS data management layer, can submit third party transfers between Grid storage elements and Google Cloud Storage (GCS), directly download/upload files and bookkeep the data in the cloud. PanDA can manage the Virtual Machine (VM) lifecycle in Google Compute Engine (GCE) through the Harvester resource manager.

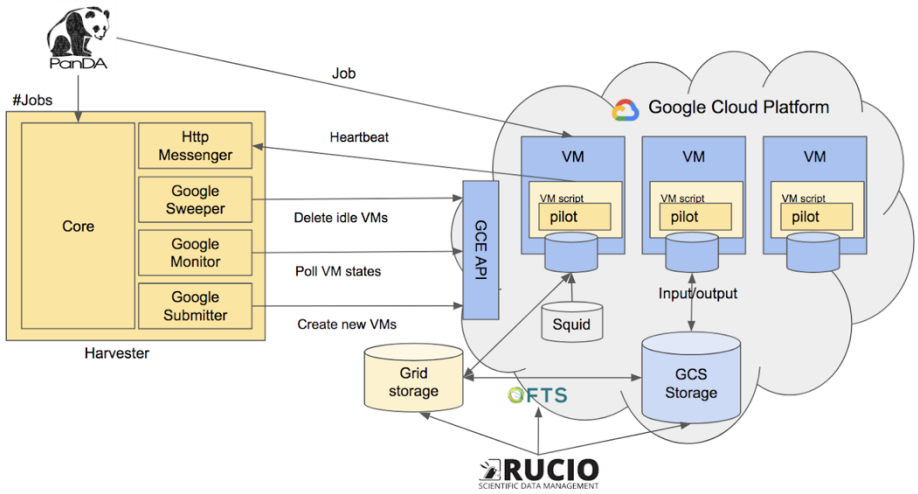

Fig. 6. Schematic overview of the PanDA and Rucio integration with Google Cloud Platform

We have implemented Harvester plugins to interact with the GCE API and boot, monitor and delete VMs. We rely on standard CernVM4 [16] images that are contextualized at boot time to run the pilot and download a job. Input and output data can be read from/written to either GCS or Grid Storage.

2.3.2 Results on GCP

During the first phase we operated a cluster of 120 cores, seamlessly integrated with PanDA and running standard Monte Carlo simulation jobs. The size of the cluster is not bound by technical reasons and could be scaled up significantly. We evaluated both normal with close to zero failure rate and preemptible [17] VMs with almost 20% failure rate (see Figure 7). Considering that the cost of preemptible VMs is 80% lower, the cost/event ratio is an attractive option. Event level workflows would allow to optimize the lost wall clock time, by only losing the currently being processed event range.

With the possibility to natively use GCS storage, we also executed more I/O intensive workloads for a few pioneer analysis users. A study for achievable data transfer rates and cost needs to be explored further.

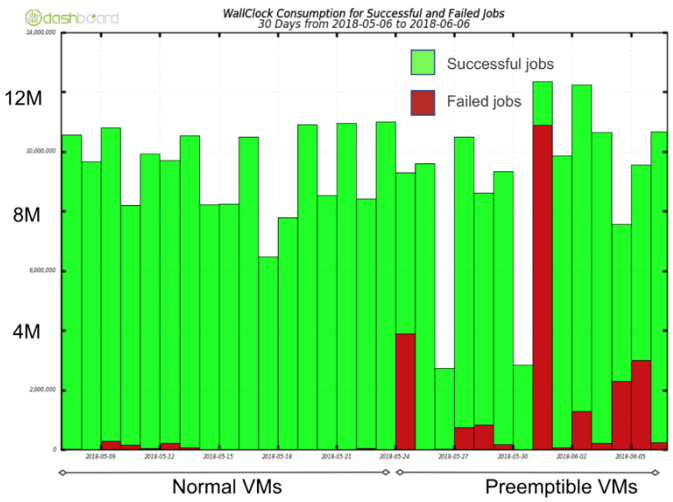

Fig. 7. Wall clock consumption for successful and failed jobs, highlighting the increased failure rate when moving to Preemptible VMs

3 Conclusions

Harvester, through its powerful core and extendable plugin architecture, has proven extreme versatility during its first two years of usage. It provided ATLAS with a single solution to exploit the allocations on completely different top ranked US DOE HPCs. The integration with Yoda allows for advanced workflows and will increase the usage efficiency in the allocations.

In the case of Google Cloud Platform, a promising proof of concept was setup within few weeks by implementing the corresponding plugins. While in the past ATLAS was limited to run I/O light workflows on the cloud, the accompanying integration of Rucio with GCP made it possible to execute different types of workloads on a scalable manner.

Acknowledgements

This research used resources of the National Energy Research Scientific Computing Center (NERSC), 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 2008 The ATLAS Experiment at the CERN Large Hadron Collider *J. Inst.* **3** S08003
2. K. De et al. J. Phys. Conf. Ser. **664** 062035 (2015)
3. P. Nilsson et al. J. Phys. Conf. Ser. **513** 032071 (2014)
4. T. Maeno et al Harvester: an edge service harvesting heterogeneous resources for ATLAS Journal of Physics: Conference Series (pre press) (2018)
5. ALCF Theta http://www.alcf.anl.gov/
6. OLCF Titan https://www.olcf.ornl.gov/titan/
7. NERSC Cori https://www.nersc.gov/
8. Globus Toolkit 5 release notes http://toolkit.globus.org/toolkit/about.html
9. J. Blomer et al. J. Phys.: Conf. Ser. **898** 062031 (2017)
10. A. Undrus et al ATLAS Software Installation on Supercomputers Journal of Physics: Conference Series (pre press)(2018)
11. A. Ayllon et al Journal of Physics: Conference Series **513** 032081 (2014)
12. Globus Online https://www.globus.org/
13. V. Tsulaia J. Phys. Conf. Ser. **664** 092025 (2015)
14. Message Passing Interface https://www.mpi-forum.org/docs/
15. M. Lassnig et al The Data Ocean Project Journal of Physics: Conference Series (pre press)(2018)
16. CernVM4 http://cernvm.cern.ch/portal/release_4.1
17. Preemptible VMs in GCE https://cloud.google.com/preemptible-vms/