Harvester : an edge service harvesting heterogeneous resources for ATLAS

Tadashi Maeno[1*], *Fernando Harald* Barreiro Megino[2], *Doug* Benjamin[3], *David* Cameron[4], *John Taylor* Childers[3], *Kaushik* De[2], *Alessandro* De Salvo[5], *Andrej* Filipcic[6], *John* Hover[1], *FaHui* Lin[7], *Danila* Oleynik[2], on behalf of the ATLAS Collaboration

[1] Brookhaven National Laboratory, NY, USA
[2] University of Texas at Arlington, TX, USA
[3] Argonne National Laboratory, IL, USA
[4] University of Oslo, Oslo, Norway
[5] Sapienza Universita e INFN, Roma I, Roma, Italy
[6] Jozef Stefan Institute, Ljubljana, Slovenia
[7] Academia Sinica, Taipei, Taiwan

Abstract. The Production and Distributed Analysis (PanDA) system has been successfully used in the ATLAS experiment as a data-driven workload management system. The PanDA system has proven to be capable of operating at the Large Hadron Collider data processing scale over the last decade including the Run 1 and Run 2 data taking periods. PanDA was originally designed to be weakly coupled with the WLCG processing resources. Lately the system is revealing the difficulties to optimally integrate and exploit new resource types such as HPC and preemptible cloud resources with instant spin-up, and new workflows such as the event service, because their intrinsic nature and requirements are quite different from that of traditional grid resources. Therefore, a new component, Harvester, has been developed to mediate the control and information flow between PanDA and the resources, in order to enable more intelligent workload management and dynamic resource provisioning based on detailed knowledge of resource capabilities and their real-time state. Harvester has been designed around a modular structure to separate core functions and resource specific plugins, simplifying the operation with heterogeneous resources and providing a uniform monitoring view. This paper will give an overview of the Harvester architecture, current status with various resources, and future plans.

1 Introduction

The Production and Distributed Analysis (PanDA) system [1] has been developed to meet ATLAS [2] production and analysis requirements for a data-driven workload

---

[1*] Corresponding author: tmaeno@bnl.gov

management system capable of operating at LHC [3] data processing scale. PanDA scalability has been demonstrated in ATLAS through the rapid increase in usage over the last decade. PanDA was designed to have the flexibility to adapt to emerging computing technologies in processing, storage, networking and distributed computing middleware. The flexibility has been successfully demonstrated through the past years of evolving technologies adopted by computing centers in ATLAS which span many continents. PanDA performed very well during the LHC data taking. The system had been producing high volumes of Monte Carlo samples and making large-scale diverse computing resources available for individual analysis. The PanDA system used to rely on a server-pilot diagram where the PanDA server centrally manages workload with various granularities, such as task, job, and event, while the pilot executes jobs on compute resources. This model has been working well for the grid with 250,000 jobs concurrently running as underlying resources are not very heterogeneous, but not very well for opportunistic resources, especially for HPCs. Each HPC center has a different edge service and operational policy, leading to an over-stretched architecture of the pilot and incoherent implementation at different HPCs. In addition, too many manual interventions were required to effectively fill available CPU resources at all HPC centers. Although some HPC sites had seamlessly been integrated with the grid through ARC Control Tower (aCT) [4], it was tightly coupled with ARC Compute Element [5] which could not be deployed at all HPC centers, especially at large HPC centers in US.

A new component, Harvester, has been developed to address those issues since December 2016 with wide collaboration of resource and PanDA experts. Harvester is a resource-facing service between the PanDA server and collection of pilots. It is stateless with a modular design to work with different resource types and workflows. The main objectives of Harvester are as follows: First, it should be a common machinery for pilot provisioning on all ATLAS computing resources. Second, it should provide a commonality layer bringing coherence to HPC implementations. Third, it should add a capability to timely optimize CPU allocation among various resource types to remove batch-level static partitioning. Finally, it should integrate the PanDA system and resources more tightly for new advanced workflows. We will present in this paper a brief overview of the Harvester architecture, current status with various resources, and plans for the future.

2 Overview of the Harvester architecture

Figure 1 shows a schematic view of the Harvester architecture. Harvester is a stateless service with a local master database and a central slave database. The local database is used for real-time bookkeeping close to resources, and the central database is periodically synchronized with the local database to provide the resource information to the PanDA server. The PanDA server uses the information together with global overview of workload distribution in order to orchestrate behaviour of Harvester instances. Therefore, communication between Harvester and the PanDA server is bidirectional. Two types of database engines are supported, sqlite3 and MariaDB. Each Harvester instance can be configured to choose a proper database as well as the number of threads, the number of processes, and the number of physical nodes, depending on available runtime environment. Multiple agents are asynchronously running in a Harvester instance to take actions based on transition of job status in the local database. Figure 2 shows how Harvester instances work in the PanDA system. For example, Harvester is supposed to run on edge nodes at HPC centers where CPU and memory usage are strictly limited, and sqlite3 and only one process are used in this kind of resource-limited environment. It is also possible to run Harvester instances outside of the HPC network if HPC centers allow remote access to compute resources. On the other hand, it is possible to have dedicated physical nodes for the grid and cloud, and in this kind of resource-rich environment MariaDB and multiple-processes are used with multiple physical nodes.

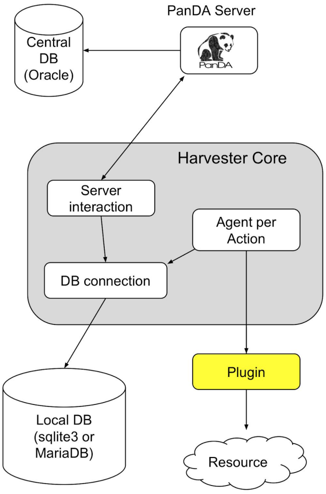

Fig. 1. Schematic view of Harvester architecture.

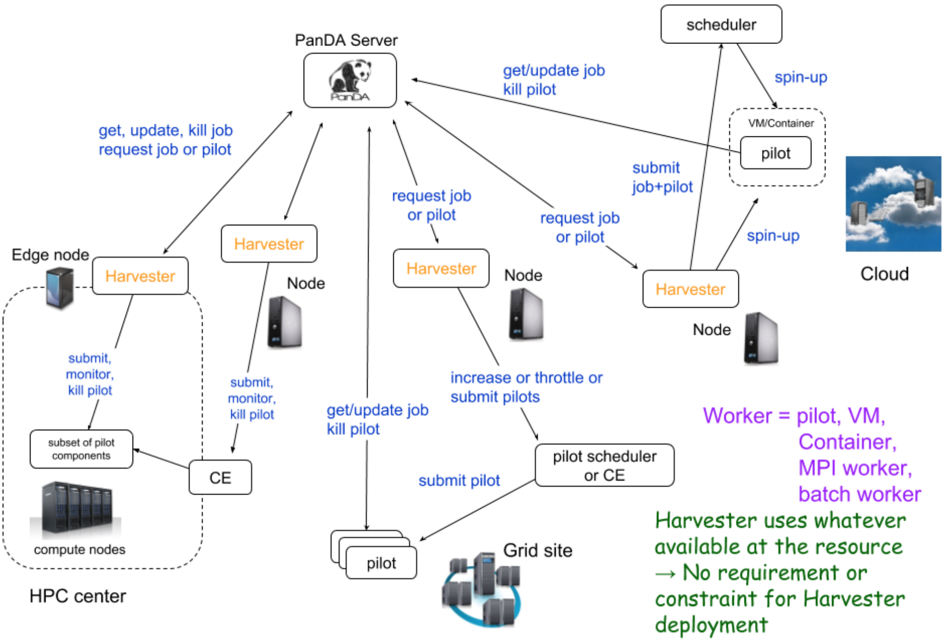

Fig. 2. Harvester instances in the PanDA system.

Harvester accesses resources through plugins which have been developed by resource experts. When Harvester instances run on edge nodes of HPC centers they access compute nodes through local HPC batch systems using HPC submission plugins. Input and output data are transferred with various plugins which use Rucio [6], Globus Online [7], gfal [8], and so on, according to data placement policy at each HPC center. When Harvester instances run outside of HPC network, different sets of plugins are used which access compute nodes through computing elements, SSH, and so on. For the grid, there is a Harvester pool on dedicated physical nodes which are centrally managed and access worker nodes through grid job submission engines like HTCondor [9] and aCT. The same Harvester pool can work for cloud with different sets of plugins which use GCE API [10], Kubernetes API [11], and so on to spin-up virtual machines or containers, or HTCondor to talk to virtual machines which are booted by other services.

3 Current status

3.1 Grid

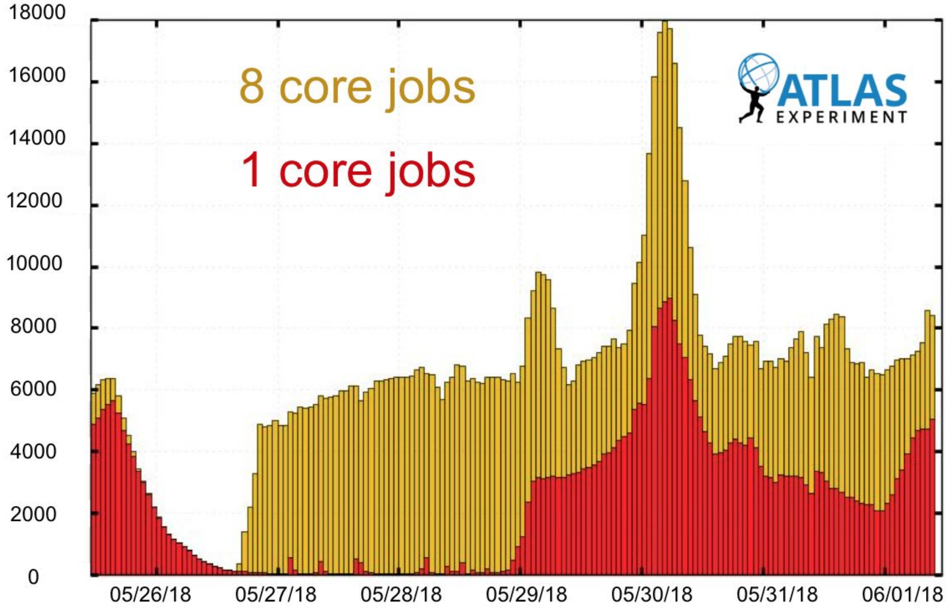

Fig. 3. The number of slots running single core jobs (in red) and multi core jobs (in yellow) at a site.

Migration to use Harvester for large scale production has been completed at CERN, Taiwan, Spanish and Italian sites, and migration at other sites are being scheduled. The runtime test framework for ATLAS offline software has been changed to use Harvester to cope with the intrinsic nature of intermittent workload submission. A mechanism has been developed to dynamically optimize resource partitioning based on current physics needs while getting rid of static batch-level partitioning, which is described in Ref [12]. Figure 3 shows that the mechanism managed to keep the ratio between the number of single core jobs and multi cores jobs at a site. It is planned to have better site description for more optimal resource usage.

3.2 Cloud

Cloud resources at CERN + Leibniz Supercomputing Centre + The University of Edinburgh with 1.2k CPU cores are running with Harvester in production, where virtual machines are booted by HTCondor. There are two major developments ongoing for cloud. The first development is for ATLAS High Level Trigger (HLT) CPU farm with 50k cores, aka Sim@P1, where resource availability widely fluctuates depending on needs for the original HLT usage [13]. Workload should proactively be assigned to the resource for quick ramp up before the resource becomes available, while workload should quickly be released as soon as HLT takes the resource back. The other development is to use native cloud API, such as GCE, EC2, and Kubernetes API. Plugins with GCE API have been successfully demonstrated in the context of the Data Ocean project [14] with GCE, Google Storage + preemptible virtual machines. Figure 4 shows success rate of jobs running with GCE + Harvester, where the resource was reconfigured to use preemptible virtual machines instead of normal virtual machines on 22nd May 2018 to see the effect of switching. Success rate become worse since some jobs were terminated during they were still running.

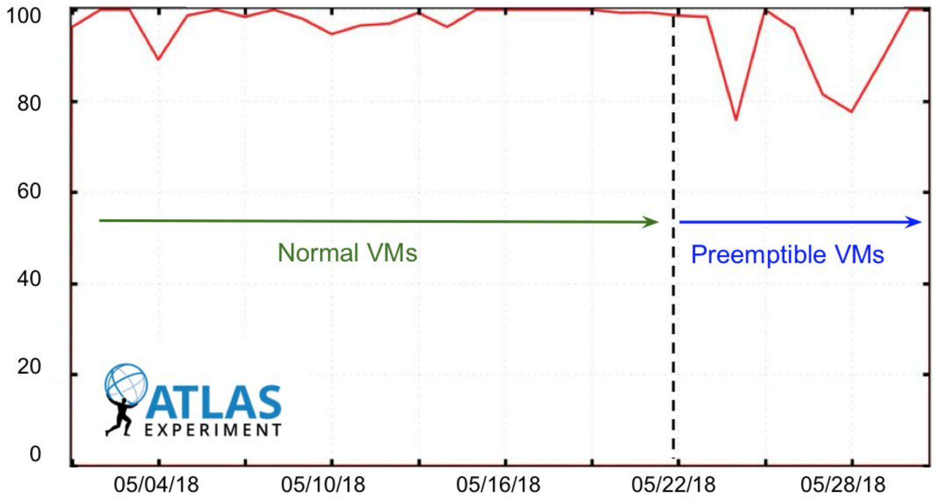

Fig. 4. Success rate of jobs running with GCE + Harvester.

3.3 HPC

Harvester has been running in production at Theta/ALCF [15], Titan/OLCF [16], Cori/NERSC [17] since February 2018 with a mechanism to dynamically combine many PanDA jobs to a single batch submission. Figure 5 shows the number of events processed per week at US HPCs for the last 12 months where there has been a steady increase since Harvester was up and running. The number of events at Cori/NERSC shown in yellow has not increased well since May 2018 because it had consumed all CPU allocation by then. Many development activities are going in parallel: Combination of jumbo payload and event service [18, 19] is going to address difficulties in payload sizing for HPCs. Operational policies at HPC centers drive the need for large payloads, while the system has to be protected against early termination due to preemption and/or inaccurate estimation of execution time. Some HPCs are being integrated to the grid infrastructure with HTCondor or ARC computing elements. A capability to dynamically change the payload size has been developed to feed optimal payloads to HPCs based on real-time information from HPC batch systems. Also, there is an idea to use a data streaming service and local cache service at HPC centers to dynamically deliver data to compute nodes on demand, which should be developed coherently with developments for ATLAS Event Streaming service [20].

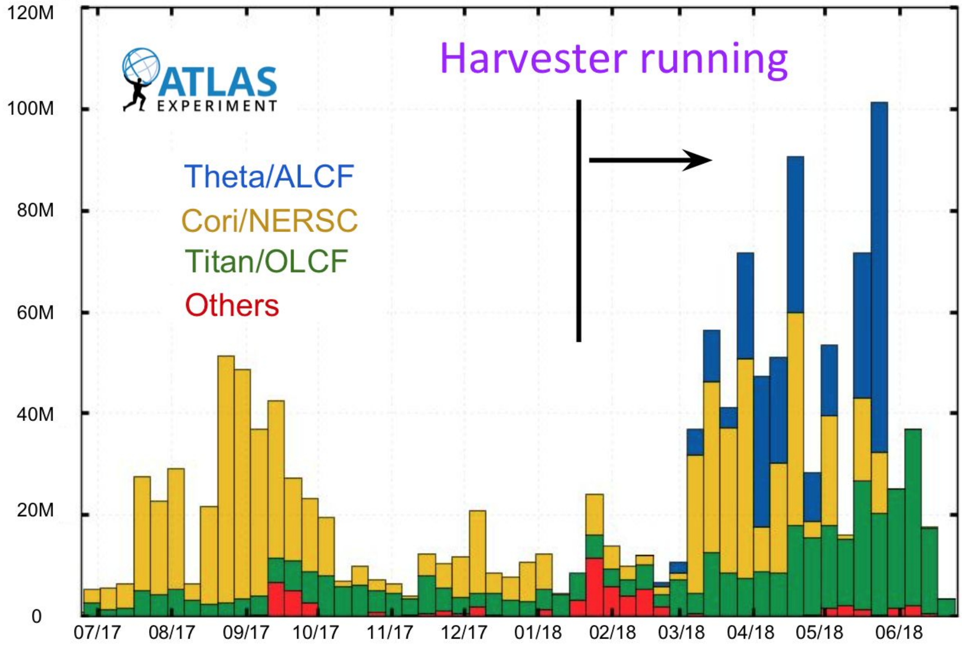

Fig. 5. The number of events processed per week at US HPCs for last 12 months.

4 Beyond ATLAS

Harvester is experiment agnostic. Six Harvester instances have been configured and ready to use for non-ATLAS experiment in BigPanDA project, including one regional instance at Thomas Jefferson Lab [21]. nEDM, LQCD, and LSST payloads have been tested, also with Next Generation Executer [22]. The first LQCD production was successful at BNL. A new plugin has been developed so that Harvester can talk to other workload management systems than PanDA, which will expand Harvester usage more into other experiments.

5 Future plans

New developments and challenges are still coming. The entire ATLAS grid should be migrated to Harvester for production as well as analysis. The mechanism of dynamic resource partitioning should be enhanced to optimize resource allocation between production and analysis. Better site description should be implemented for more optimal resource usage. All HPCs should seamlessly be integrated with the grid resources without any manual interventions. Finally, Harvester usage could be expanded beyond ATLAS.

6 Conclusions

Harvester has been developed since December 2016 with wide collaboration of resource and PanDA experts. Many development activities have been ongoing in parallel for various resources with coherent implementations to meet different requirements. Harvester is already in production for various resources, while there are still a lot of challenges to come.

Acknowledgement

This manuscript has been authored by employees of Brookhaven Science Associates, LLC under Contract No. DE-SC0012704 with the U.S. Department of Energy. The publisher by accepting the manuscript for publication acknowledges that the United States Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

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. K. De et al., J. Phys. Conf. Ser. **664** 062035 (2015)
2. ATLAS Collaboration, J. Inst. **3** S08003 (2008)
3. Evans L and Bryant P (editors), J. Inst. **3** S08001 (2008)
4. J. K. Nilsen et al., J. Phys. Conf. Ser. **664** 062042 (2015)
5. ARC Compute Element [software], available from http://www.nordugrid.org/arc/ce/ [accessed 2018-11-22]
6. Rucio [software], available from https://rucio.cern.ch/ [accessed 2018-11-22]
7. Globus Online [software], available from https://www.globus.org/ [accessed 2018-11-22]

8. gfal [software] Available from https://dmc.web.cern.ch/projects/gfal-2/home [accessed 2018-11-22]
9. HTCondor [software], available from https://research.cs.wisc.edu/htcondor/ [accessed 2018-11-22]
10. GCE API [software], available from https://cloud.google.com/compute/docs/reference/rest/v1/ [accessed 2018-11-22]
11. Kubernetes API [software], available from]https://kubernetes.io/docs/concepts/overview/kubernetes-api/ [accessed 2018-11-22]
12. F. H. Barreiro Megino et al., J. Phys. Conf. Ser. (to be published)
13. F. Berghaus et al., Phys. Conf. Ser. (to be published)
14. M. Lassnig et al., J. Phys. Conf. Ser. (to be published)
15. ALCF Theta, http://www.alcf.anl.gov/
16. OLCF Titan, https://www.olcf.ornl.gov/titan/
17. NERSC Cori, https://www.nersc.gov/
18. T. Wenaus et al., J. Phys. Conf. Ser. **664** 062065 (2015)
19. V. Tsulaia et al., J. Phys. Conf. Ser. **664** 092025 (2015)
20. N. Magini et al., J. Phys. Conf. Ser. (to be published)
21. P. Svirin et al. J. Phys. Conf. Ser. (to be published)
22. M. Turilli et al., arXiv preprint arXiv:1609.03484 (2016)