The future of PanDA in ATLAS distributed computing

K De [1], A Klimentov [2], T Maeno [2], P Nilsson [2], D Oleynik [1], S Panitkin [2], A Petrosyan [3], J Schovancova [1], A Vaniachine [4], T Wenaus [2], on behalf of the ATLAS Collaboration

[1] University of Texas at Arlington, TX, USA
[2] Brookhaven National Laboratory, NY, USA
[3] Joint Institute for Nuclear Research, Dubna, RU
[4] Argonne National Laboratory, IL, USA

tmaeno@bnl.gov

Abstract. Experiments at the Large Hadron Collider (LHC) face unprecedented computing challenges. Heterogeneous resources are distributed worldwide at hundreds of sites, thousands of physicists analyse the data remotely, the volume of processed data is beyond the exabyte scale, while data processing requires more than a few billion hours of computing usage per year. The PanDA (Production and Distributed Analysis) system was developed to meet the scale and complexity of LHC distributed computing for the ATLAS experiment. In the process, the old batch job paradigm of locally managed computing in HEP was discarded in favour of a far more automated, flexible and scalable model. The success of PanDA in ATLAS is leading to widespread adoption and testing by other experiments. PanDA is the first exascale workload management system in HEP, already operating at more than a million computing jobs per day, and processing over an exabyte of data in 2013. There are many new challenges that PanDA will face in the near future, in addition to new challenges of scale, heterogeneity and increasing user base. PanDA will need to handle rapidly changing computing infrastructure, will require factorization of code for easier deployment, will need to incorporate additional information sources including network metrics in decision making, be able to control network circuits, handle dynamically sized workload processing, provide improved visualization, and face many other challenges. In this talk we will focus on the new features, planned or recently implemented, that are relevant to the next decade of distributed computing workload management using PanDA.

ATL-SOFT-PROC-2015-047
17 May 2015

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 management system capable of operating at LHC [3] data processing scale. PanDA has a highly scalable and flexible architecture. PanDA scalability has been demonstrated in ATLAS through the rapid increase in usage over the past seven years, and is expected to easily meet the growing needs over the next decade. PanDA was designed to have the flexibility to adapt to emerging computing technologies in processing, storage, networking and distributed computing middleware. This flexibility has also been successfully demonstrated through the past years of evolving technologies adapted by computing centers in ATLAS which span many continents. PanDA performed very well during the LHC Run 1 data taking

period. The system has been producing high volume Monte Carlo samples and making large-scale diverse computing resources available for individual analysis, while being actively evolved to meet the rapidly changing requirements for analysis use cases. There are about 150 thousand jobs concurrently running in the system and more than 5 million jobs are processed in total per week.

However, in spite of great successes there are strong motivations for new developments. First, it is desirable to use pledged resources more efficiently. Second, an intelligent workload partitioning mechanism is required to utilize opportunistic resources based on their dynamic characteristics. Third, workload and bookkeeping should be handled both with coarse and fine granularities. Fourth, the system needs to be aware of the network to intelligently leverage network resources. Finally, user interfaces must be improved to effectively diagnose and optimize the system. We will present in this paper a brief overview of the major aspects of PanDA systems evolution for LHC Run 2, as well as plans for the future.

2. An overview of system evolution for LHC Run 2

2.1. Dynamic partitioning of workload

Two components have been newly developed since Run 1 concluded. The Database Engine for Tasks (DEFT) [4] handles production requests and tasks. The Job Execution and Definition Interface (JEDI) dynamically splits workloads for optimal usage of resources and automatically merges outputs. JEDI manages workloads at task, job, file, and event levels. The primary change in the workflow is that tasks are now partitioned to jobs based on the dynamic state of available resources; that is, jobs are an implementation detail of getting tasks done, which is distinct from traditional workload systems in which the job is the atomic unit.

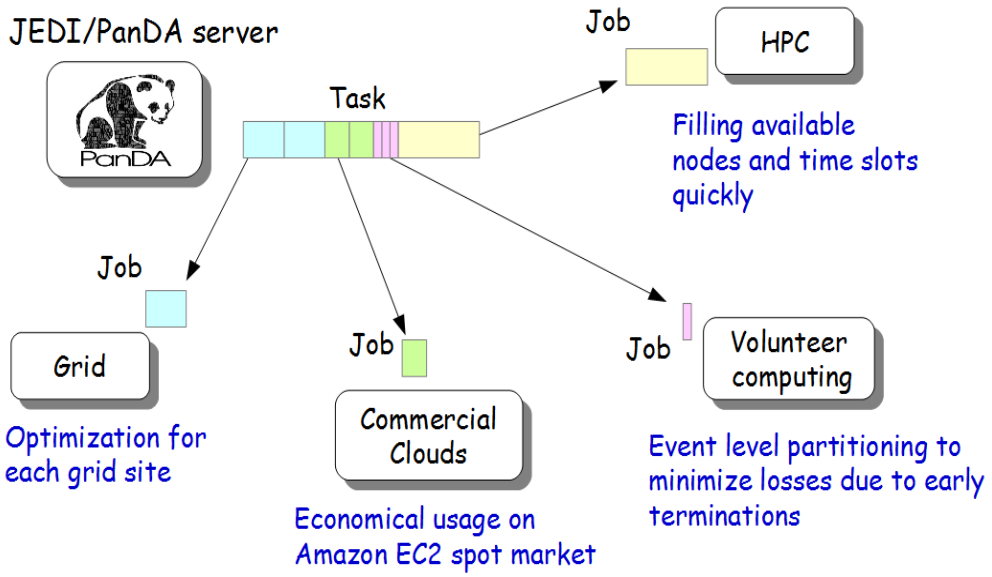

Figure 1. Workload partitioning for pledged and opportunistic resources.

Figure 1 shows how workloads are partitioned for pledged and opportunistic resources. Given a task may process of order one million events, tasks must be partitioned to jobs in an optimal way for dispatching the work to processing resources. In the old system, partitioning was static; for example, one job was created per one thousand events. JEDI now partitions workloads dynamically based on

PanDA's current knowledge of available resources and their distinct characteristics. For HPC resources, large chunks are assigned to quickly fill available nodes and time slots. For commercial clouds like Amazon EC2 [5], workloads are partitioned appropriately for resources that may be economically obtained on the spot market, namely with fine granularity, robust against sudden termination. Similarly for volunteer computing, workloads are partitioned with fine granularity at the event level using the Event Service [6,7] to minimize losses due to sudden termination of volunteer computing slots. Dynamic partitioning is useful for pledged grid resources as well since jobs are optimized based on each site's particular characteristics, such as wall time limit and memory size.

Other benefits arise in addition to optimal resource usage. Users don't need to have detailed knowledge of each computing resource, good especially for heterogeneous resources which remain common. Also, job parameters are self-optimized by using a scouting mechanism which generates a small number (~10) of jobs with minimal input chunks for each task in order to collect accurate job metrics, such as average execution time and memory consumption. Job parameters are then optimized using the job metrics for the rest of the inputs. Finally, client applications are simplified and speeded up since most of the user functions such as resource lookup and job optimization have been moved to the server side.

2.2. Integration of network awareness

In the old ATLAS computing model, analysis jobs were constrained to always read input data from local storage. Job distribution was tightly coupled to data distribution, and consequently computing resources were not always used very efficiently. The job brokerage has been improved to consider costs for data access over wide area networks, which allows ATLAS to slightly relax the data locality constraint so that a fraction of analysis jobs are sent to sites which have free CPUs and good network connection to remote data.

Each regional federation is composed of one Tier 1 site and multiple Tier 2 sites. Files have traditionally been transferred between the Tier 1 and associated Tier 2 sites within the federation. Now by utilizing network awareness, site groupings are dynamically defined based on those with good network access to the data, while previously associations were based solely on regional association, e.g. all UK Tier 2 sites were associated with the UK Tier 1 site. This means that files are transferred via network connections that are actively measured to be good, and avoiding the need for multi-hops through Tier 1s.

2.3. Event Service

The Event Service has been developed to perform processing at fine granularity, down to the event level. It allows workloads to be tailored dynamically to resources currently available. Figure 2 shows how the Event Service works. JEDI manages workloads at the event level, managing the assignment and bookkeeping of the processing, retrying events if necessary, and merging outputs once all events are processed. Multiple workers run on opportunistic resources to process events serially per process but with many processes in parallel. If slots are abruptly revoked, as is common on opportunistic resources, losses are typically limited to the last event being processed, and retry is automatic.

The Event Service has been validated on NERSC's Edison supercomputer [8] and is being ported to other HPC platforms and ATLAS@Home [9]. Validation on the Amazon EC2 spot market is underway. An overview and software architecture of the Event Service is provided in Ref.[6] and Ref. [7], respectively.

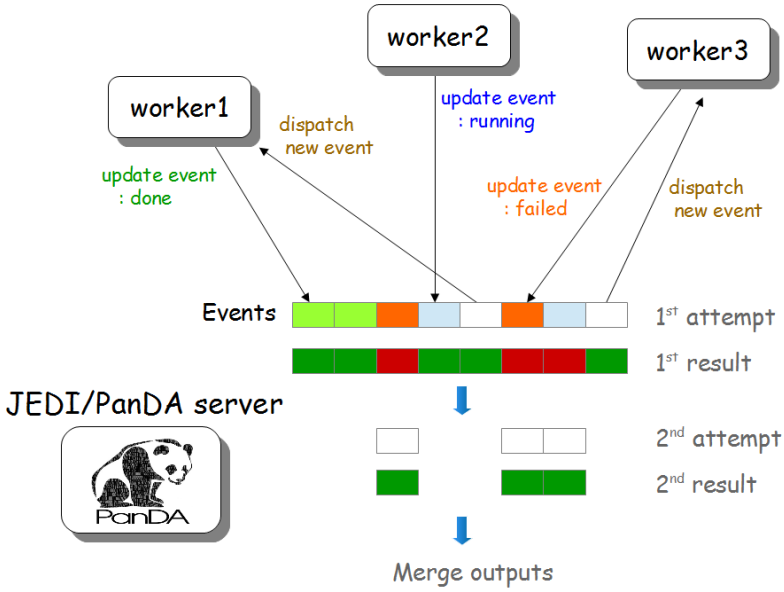

Figure 2. Event processing with Event Service

2.4. Evolving PanDA pilot

The PanDA pilot [10] is one of the major components in the PanDA system and has been actively evolving throughout PanDA's 10 year history. The pilot has lately been re-factored to separate core modules and experiment-specific plugins, to meet requirements to support multiple experiments. The plugin structure has also been adopted to support new workflows for the Event Service and HPCs. HPC plugins have been developed for Titan [11], NERSC Edison, Mira [12], and Anselm [13]. The Yoda software suite [7] acts as an intermediate layer between the PanDA server and the pilot which doesn't have access to outside connections on HPCs. Yoda has been successfully validated on NERSC Edison and is currently being extended to Titan. The pilot has been improved to use object stores as temporary storage, which is particularly useful for Event Service jobs since they produce many small files that are merged after the Event Service job has finished. Traditional storage cannot cope as well with fast access to many small objects. The pilot has also been improved to enable gLExec integration, the ability to dynamically switch identity from the pilot owner to the user and payload owner when executing the payload. Details of the gLExec integration are described in Ref. [14].

2.5. PanDA on Titan

Work on integration of Titan machines with PanDA is ongoing at the Oak Ridge Leadership Computing Facility. Figure 3 shows how Titan is integrated with PanDA. The PanDA pilot has been modified to run on Titan's front-end nodes in backfill mode. The pilot collects information about free resources in quasi real time and submits jobs to Titan's scheduler to fill the available resources. Steady operation for continuous PanDA job submission in backfill mode has been successfully demonstrated. The details are described in Ref.[15].

2.6. New monitoring

High quality user interfaces are essential for supporting effective use and comprehensive optimization and diagnostics of the system. A new user interface and monitoring system have been developed based on the Django framework. Figure 4 shows the front page of the new monitoring. There is clear

separation between data access and visualization in the architecture of the system. In addition to the monitor's built-in system views and drill-down details for diagnostics, REST APIs are available to access entities in the system programmatically so that users can develop their own applications to derive interesting information. Task-oriented views are provided in addition to job-oriented views, consistent with the task-oriented direction of PanDA system evolution.

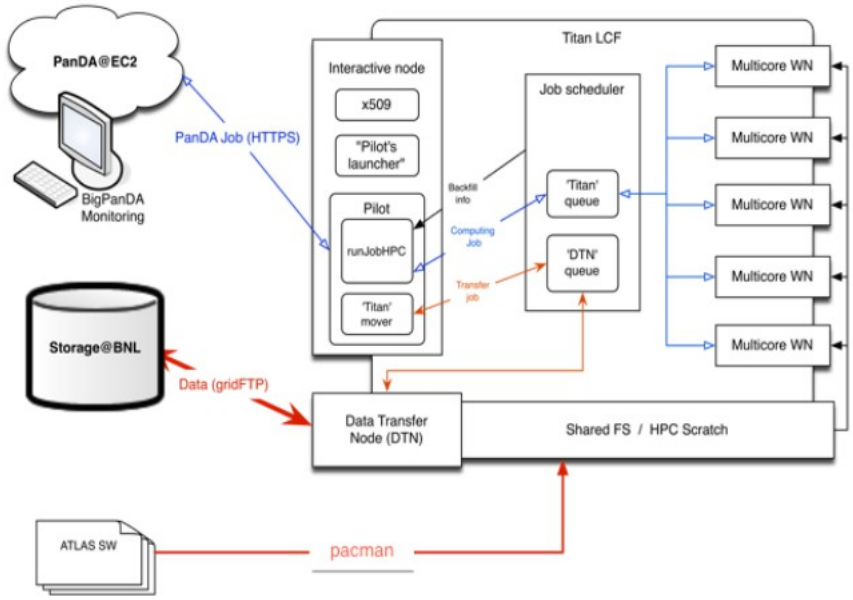

Figure 3. Integration of Titan with PanDA

3. Future plans

Much work remains to be done. More intelligence needs to be added to workload partitioning and brokerage. PanDA should proactively take control of the network to optimize workflows and dataflows. New computing resources will need to be brought in production efficiently and economically. Lightweight tools should be developed for users who are not fully integrated with the grid, to leverage PanDA for utilization of their local resources.

4. Conclusions

The PanDA system performed very well through the LHC Run 1 data-taking period, producing high volume Monte Carlo samples and making large-scale heterogeneous computing resources available for individual analysis. Based on Run 1 experience and Run 2 requirements the system has been actively evolved to meet the rapidly changing production and analysis use cases. New components and features have been delivered to ATLAS in advance of LHC Run 2. Nonetheless many developments and challenges still lie ahead while the system delivers the distributed production and analysis capability required by ATLAS during Run 2.

Figure 4. The front page of new monitoring shows history of the number of CPU cores used by concurrently running jobs per regional federation (top row) and per activity (bottom row) for 1, 7 and 30 days, respectively. There are also hyper-links to various monitor pages, such as task-oriented views and job-oriented views.

Notice:
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 non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

References

[1] Maeno T et al. 2012 Evolution for the ATLAS PanDA production and distributed analysis system *J. Phys. Conf. Ser.* **396** 032071
[2] ATLAS Collaboration 2008 The ATLAS Experiment at the CERN Large Hadron Collider, *J. Inst.* 3 S08003
[3] Evans L and Bryant P (editors) 2008 LHC Machine, *J. Inst.* 3 S08001

[4] Borodin M et al. 2015 Scaling up ATLAS production system for the LHC Run 2 and beyond: project ProdSys2 *Proc. Int. Conf. on Computing in High Energy and Nuclear Physics 2015 J. Phys. Conf. Ser.*
[5] The Amazon Elastic Computing Cloud http://aws.amazon.com/ec2/
[6] Wenaus T et al. 2015 The ATLAS Event Service: A new approach to event processing *Proc. Int. Conf. on Computing in High Energy and Nuclear Physics 2015 J. Phys. Conf. Ser.*
[7] Tsulaia V et al. 2015 Fine grained event processing on HPCs with the ATLAS Yoda system *Proc. Int. Conf. on Computing in High Energy and Nuclear Physics 2015 J. Phys. Conf. Ser.*
[8] NERSC Edison https://www.nersc.gov/
[9] Cameron D et al. 2015 ATLAS@Home: Harnessing Volunteer Computing for HEP *Proc. Int. Conf. on Computing in High Energy and Nuclear Physics 2015 J. Phys. Conf. Ser.*
[10] Nilsson P 2014 Next Generation PanDA Pilot for ATLAS and Other Experiments *J. Phys. Conf. Ser.* **513** 032071
[11] OLCF Titan https://www.olcf.ornl.gov/
[12] ALCF Mira https://www.alcf.anl.gov/mira
[13] Anselm http://www.it4i.cz/
[14] Karavakis E et al. 2015 gLExec Integration with the ATLAS PanDA Workload Management System *Proc. Int. Conf. on Computing in High Energy and Nuclear Physics 2015 J. Phys. Conf. Ser.*
[15] Panitkin S et al. 2015 Integration of PanDA workload management system with Titan supercomputer at OLCF *Proc. Int. Conf. on Computing in High Energy and Nuclear Physics 2015 J. Phys. Conf. Ser.*