The ATLAS Event Service: A New Approach to Event Processing

P Calafiura[1], K De[2], W Guan[3], T Maeno[4], P Nilsson[4], D Oleynik[2], S Panitkin[4], V Tsulaia[1], P Van Gemmeren[5], and T Wenaus[4] on behalf of the ATLAS Collaboration

[1]Lawrence Berkeley National Laboratory, 1 Cyclotron Rd, Berkeley, CA 94720, USA
[2]University of Texas at Arlington, 701 S Nedderman Dr, Arlington, TX 76019, USA
[3]University of Wisconsin – Madison, Madison, WI 53706, USA
[4]Brookhaven National Laboratory, Upton, NY 11973, USA
[5]Argonne National Laboratory, 9700 South Cass Avenue, Lemont, IL 60439, USA

E-mail: `wenaus@gmail.com`

Abstract. The ATLAS Event Service (ES) implements a new fine grained approach to HEP event processing, designed to be agile and efficient in exploiting transient, short-lived resources such as HPC hole-filling, spot market commercial clouds, and volunteer computing. Input and output control and data flows, bookkeeping, monitoring, and data storage are all managed at the event level in an implementation capable of supporting ATLAS-scale distributed processing throughputs (about 4M CPU-hours/day). Input data flows utilize remote data repositories with no data locality or pre-staging requirements, minimizing the use of costly storage in favor of strongly leveraging powerful networks. Object stores provide a highly scalable means of remotely storing the quasi-continuous, fine grained outputs that give ES based applications a very light data footprint on a processing resource, and ensure negligible losses should the resource suddenly vanish. We will describe the motivations for the ES system, its unique features and capabilities, its architecture and the highly scalable tools and technologies employed in its implementation, and its applications in ATLAS processing on HPCs, commercial cloud resources, volunteer computing, and grid resources.

Notice to publisher (can be excised from the published manuscript): This manuscript has been authored by employees of Brookhaven Science Associates, LLC under Contract No. DE-AC02-98CH10886 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.

1. Introduction

The ATLAS experiment [1] has accumulated to date a globally distributed data volume in excess of 160 Petabytes, processed at a scale of about 4M CPU-hours/day at about 140 computing centers around the world. Even with this massive processing scale the experiment is resource limited. The ATLAS physics program can be substantially enriched by applying more computing resources, particularly to Monte Carlo simulation to enable the deeper exploration of rare physics channels through a better understanding of their backgrounds. In the future, computing resources will be even more constraining: the LHC and ATLAS upgrade programs over the next decade will bring an order of magnitude increase in computing requirements. For these reasons it is essential that the Collaboration makes maximal and efficient use of all available processing resources.

Opportunistic processing resources have a large potential for expanding the ATLAS processing pool. High performance supercomputers (HPCs), cost-effective clouds such as the Amazon spot market [2], volunteer computing (ATLAS@Home) [3], and shared grid resources are all promising sources. It is characteristic of such resources that job slot lifetime is unpredictable, variable and may be very short (or very long); exploiting them fully and efficiently requires adapting to this characteristic. HPC workloads should adapt to the scheduling opportunities that these systems afford, namely the gaps between large parallel jobs; these systems are nominally full all the time with such jobs but just as a full jar of rocks has room for a lot of sand, there is room on such systems for fine grained adaptive workloads. On commercial clouds it is the Amazon spot market model that is most cost-effective, and workloads on these dynamic market-driven resources must be robust against instantaneous loss of the node. Similarly on shared grid resources, which often impose preemption, and on volunteer computers which can disappear at any moment. We describe here a new fine grained approach to event processing that is largely motivated by these considerations, the ATLAS Event Service.

2. The ATLAS Event Service

The ATLAS Event Service (ES) is designed to open opportunistic resources to efficient utilization, and to improve overall efficiency in the utilization of processing and storage. The ES implements quasi-continuous event streaming through worker nodes, enabling full and efficient exploitation through their lifetime whether that is 30 minutes, 30 hours, or 30 seconds from now, with no advance notice. The ES achieves this by decoupling processing from the chunkiness of files, streaming events into a worker and streaming the outputs away in a quasi-continuous manner, with a tunable granularity typically set to 15 minutes or so. While the worker persists, it will elastically continue to consume events and stream away outputs with no need to tailor workload execution time to resource lifetime. When the worker terminates for whatever reason, losses are limited to the last few minutes, corresponding to a single event when the task is ATLAS Monte Carlo simulation. The approach offers the efficiency and scheduling flexibility of preemption without the application needing to support or utilize checkpointing.

The Event Service also has benefits for conventional processing on clusters and grids. Processing resources can be reassigned in a quick and agile way, e.g. to transition quickly from single to multi-core, avoiding inefficiencies from resource draining, because the ES can vacate a resource at any time with negligible losses. Predicting job duration in order to tailor jobs to a slot lifetime is a subject of much detailed study, often with imprecise results; the ES makes such predictions unnecessary. The ES also does away with couplings between input file size, output file size, and job duration, allowing them to be independently tailored.

The Event Service is designed to bring efficiencies in storage utilization as well as processing. Storage is the largest cost component in ATLAS computing, making continuous improvement in storage efficiency an essential complement to expanding processing resources. The available disk storage volume is not going to scale with the processing in coming years, it is too expensive. The ES improves storage efficiency by driving down the need for data replicas through the efficient use of wide area network (WAN) data access, thus decoupling processing from data locality requirements. Data retrieval across the WAN is fully asynchronous to the processing in the ES design, avoiding inefficiencies from WAN latencies. Data access in the ES is designed to be mediated by the Event Streaming Service (ESS) capable of providing additional efficiency measures such as utilizing local cache preferentially over WAN access, and marshaling data sent over the WAN to limit data transferred to what is actually needed by the application. The ESS is a work in progress, not yet part of the deployed Event Service.

Crucial to the ES's agility in vacating resources with negligible losses is the fine grained streaming of outputs (and bookkeeping metadata) off the resource in near real time. This results in outputs that are many (one for every 15 or so core-minutes) and small (typically a few MB or less). They must be sent to a (generally remote) store that is fast, highly scalable and efficient in handling small objects. Object stores are an ideal technology, and are the basis of ES output management. The Event Service automatically merges outputs into conventional ATLAS data files when ES jobs complete. Promptly streaming away outputs has further benefits in minimizing local storage needs, making outputs quickly available to users, and avoiding outputs being trapped on storage local to the processing when it becomes temporarily inaccessible, a common problem on the grid.

3. Architecture and Implementation

In its workflow aspects the Event Service builds principally on three ATLAS core software systems: the PanDA distributed workload manager [4], PanDA's new JEDI extension [5] for flexible dynamic workflow management (part of the new ATLAS production system Prodsys2 [6]), and the AthenaMP [7] parallel offline framework. PanDA operates by autonomous pilot jobs [8] harvesting resources by launching in job slots and requesting to a central PanDA service for the dispatch of work. The PanDA server manages a coherent global queue of ATLAS production and analysis work to be performed, and allocates work to requesting pilots based on intelligent brokerage and the characteristics of the resource. The JEDI extension enables PanDA to dynamically partition and manage workflows down to fine event level granularities, managing the associated bookkeeping in PanDA's Oracle back end database. PanDA and JEDI manage the workflows in a fully automatic way, from high level task specifications defined by physicists through partitioning and executing the work on dynamically selected resources, merging results, and chaining subsequent downstream processing. The payloads launched by PanDA for the ES are instances of AthenaMP; its parallel capabilities are used to manage the distribution and processing of events concurrently among parallel workers. The HPC implementation of the ES benefits from the BigPanDA project that has extended PanDA's operation to HPCs [9].

In its dataflow aspects the Event Service relies on the ATLAS event I/O system's support for efficient WAN data access [10], the uniform support for xrootd based remote data access across ATLAS [11], highly scalable object stores for data storage that architecturally match the fine grained data flows, and the excellent high performance networking fabric on which the success of LHC computing has long been built.

The Event Service architecture is shown schematically in Figure 1, and its present realization in ATLAS is shown in Figure 2. Event Service processing begins with the dispatch of an ES job from PanDA to the pilot placing the job request. Receiving the ES job, the pilot goes into event processing mode, entering an event loop in which new event ranges to process are requested every few minutes. In the target architecture the event list is passed to a data fetcher that interacts with an Event Streaming Service (ESS) to acquire the data, from local cache if available, otherwise over the WAN. The fetch is asynchronous with the execution of the payload, AthenaMP in the present instance. However, in the initial implementation, the data fetch is done within AthenaMP itself, via direct read to local or remote files (using a locator token obtained from an event indexing service), and does not use the still-to-be-implemented ESS.

Parallel workers managed by AthenaMP consume events and produce outputs in an area monitored by an output stager service. As new outputs appear every few minutes per worker, the stager uploads them to an object store (presently at BNL [12] or CERN) and informs PanDA of the completion. PanDA uses JEDI's event level bookkeeping to keep track of the processing, marking events as finished or failed. Failed or timed out events are reassigned to

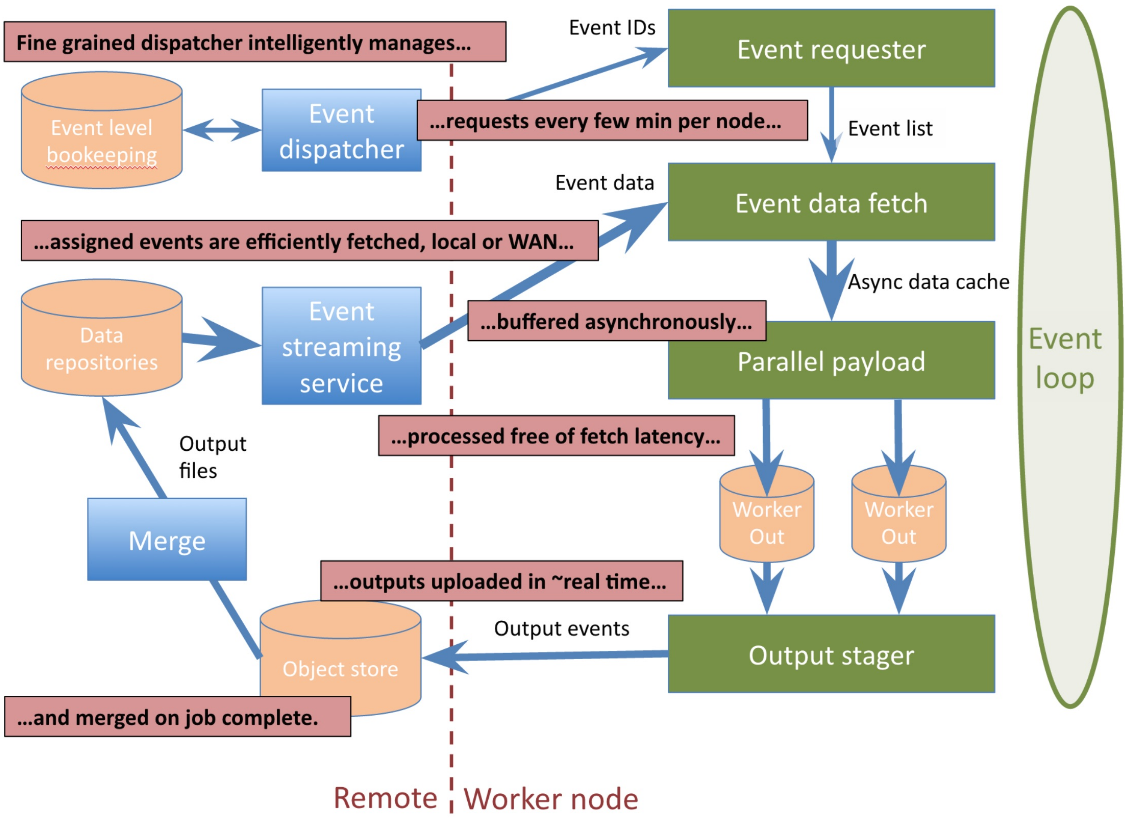

Figure 1. A schematic view of the ATLAS Event Service architecture

other workers. In this way a job advances towards completion, flexibly consuming whatever resources are available as expressed through pilots acquiring slots and requesting work. PanDA detects when all events for a job are successfully processed, and triggers a merge job that merges the small object store resident outputs into conventional ATLAS files, with whatever granularity is desired for the final output files.

A specialization of the Event Service for HPCs, Yoda [13], was developed to accommodate the particular features of HPC architectures, most notably the lack of outbound access from worker nodes. Yoda is a miniaturization of the PanDA/JEDI event workflow management to operate within the HPC itself, with MPI replacing HTTP communication. Yoda's master/client architecture allows tailoring of workloads automatically and dynamically to whatever scheduling opportunities the resource presents – the sand filling the jar of rocks.

4. Using the Event Service

The present and near term use case for the Event Service is in Monte Carlo simulation, the single largest consumer of CPU in ATLAS at about 50% of all processing resources. MC is a relatively simple, CPU-intensive payload that is amenable to operating on HPCs. Event Service based simulation is operating on several opportunistic platforms, and will begin production operation in Spring 2015.

Yoda was successfully demonstrated running ATLAS MC on NERSC's Edison supercomputer [14] in a demo at Supercomputing 2014 (DOE booth) [15]. During ramp-downs to make room for a large scheduled job or a maintenance shutdown, Yoda expanded into the resources freed by the ramp, with negligible losses to the processing when the jobs were terminated by the

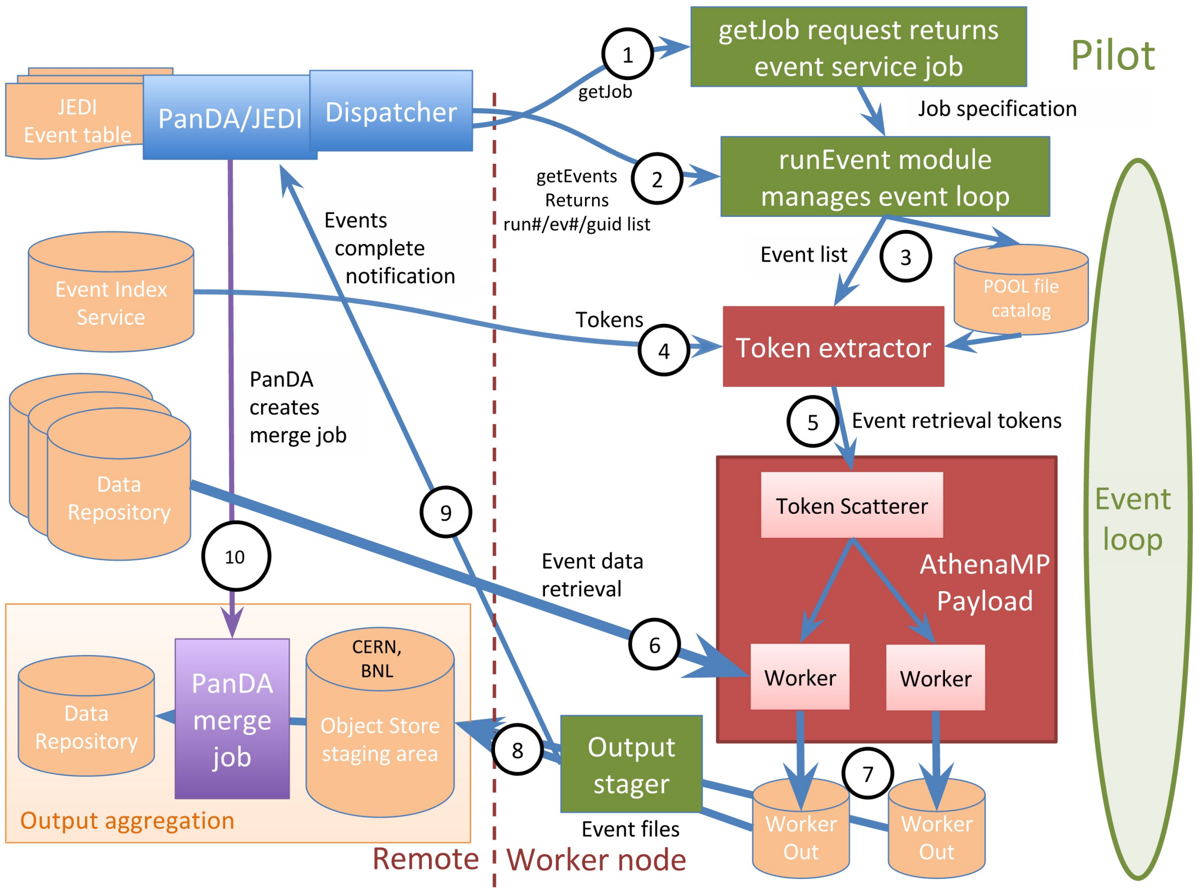

Figure 2. The present implementation of the ATLAS Event Service

scheduler. Yoda has demonstrated good scaling on Edison to 50k concurrent cores. Physics validation is completed and production can begin once allocated time becomes available.

On the Amazon spot market, a collaboration between BNL, ESnet and Amazon is providing the basis for large scale Amazon production, up to about 50k concurrent cores, supported by an Amazon grant. The currently seven-fold cheaper cost of spot market machines relative to on-demand brings them close to economic parity with owned resources, and the economics are ever improving. The BNL ATLAS Tier 1 is able to elastically and transparently expand PanDA resources into cloud resources, enabling peak usage to expand transparently into the cloud if desired. Physics validation and production commissioning are expected to be completed on Amazon in Spring 2015.

ATLAS has recently implemented a BOINC based ATLAS@Home volunteer computing service [3]. The Event Service is well suited to the volunteer computing environment because it is accommodating of machines disappearing suddenly, results are streamed off incrementally and so cannot be trapped locally, and there is no need to shape job duration to unpredictable job slot lifetimes. Mechanisms to secure PanDA based processing in the ATLAS@Home environment have been implemented and the Event Service port is underway, expected to complete in Summer 2015.

Concurrently with these efforts on opportunistic platforms, the ES is also being readied for simulation production on the ATLAS high level trigger farm (a cloud based resource where rapid switch-over between online and offline usage is a requirement) and conventional grid resources. If production experience with the ES is good, it is expected to become the predominant means of running MC production.

5. Current and Future Development

The one element of the Event Service's design architecture that remains to be implemented is the Event Streaming Service as the basis for pre-fetching event data asynchronously from the processing. The ESS will decouple event data retrieval and its associated latency – substantial for WAN access – from the processing. Implementation of the ESS is in progress with a first version expected in late 2015. We envision the ESS evolving to become a sophisticated service on the Content Delivery Network model that intelligently applies knowledge of the data being requested, the locality and caching status of that data, its popularity and its replication patterns to marshal the needed data and deliver it to the client in as efficient a manner as possible. Such a service will be able to cope efficiently with sparsely sampled data and so will open the ES to analysis applications.

Extending the ES beyond Monte Carlo simulation production to reconstruction and analysis also requires work in the Event Service workflow at the pilot and payload levels, and in the event I/O to enable fine grained output partitioning for these processing modes. This work will proceed over the next year or so, with moderate priority because the greatest gains come from using the ES for Monte Carlo simulation.

Work is underway to extend Yoda operation to HPCs beyond NERSC, in particular to ORNL's Titan [16], where PanDA is being applied to harvest backfill resources opportunistically.

6. Conclusions

The ATLAS Event Service takes a new fine grained approach to event processing in order to make the most of the many varieties of opportunistic computing available to compute-limited science. Such resources typically reward agile, flexible workflows that can efficiently occupy potentially short lived job slots. The ES serves this scenario well, building on key enabling ATLAS software systems developed in the last two years. The ES is also designed to leverage WAN data access and high performance networking to economize on storage. The system is close to entering production on HPC, commercial cloud, grid and volunteer computing platforms.

The ES is a system designed for data intensive, network centric, platform agnostic computing, an increasingly important paradigm in scientific computing. The ES approach to fine grained processing can have benefit to any application able to partition its processing and data outputs to an appropriate granularity. The ATLAS ES team welcomes interest from others in prospective collaboration at any level, from sharing of ideas to trying out the implemented system.

References

[1] ATLAS Collaboration 2008, The ATLAS Experiment at the CERN Large Hadron Collider, JINST **3** S08003

[2] The Amazon Elastic Computing Cloud, http://aws.amazon.com/ec2/

[3] Cameron D et al. [ATLAS Collaboration] 2015, ATLAS@Home: Harnessing Volunteer Computing for HEP. Proceedings of the CHEP 2015 conference J. Phys.: Conf. Series. See also http://atlasathome.cern.ch/

[4] Maeno T et al. [ATLAS Collaboration] 2012, Evolution of the ATLAS PanDA production and distributed analysis system, J. Phys.: Conf. Series **396**. See also http://twiki.cern.ch/twiki/bin/view/PanDA/PanDA

[5] Maeno T et al. [ATLAS Collaboration] 2015, The Future of PanDA in ATLAS Distributed Computing. Proceedings of the CHEP 2015 conference J. Phys.: Conf. Series.

[6] De K et al. [ATLAS Collaboration] 2014, Task Management in the New ATLAS Production System, J. Phys.: Conf. Series **513** 032078.

[7] Binet S et al. [ATLAS Collaboration] 2012, Multicore in production: Advantages and limits of the multiprocess approach in the ATLAS experiment, J. Phys.: Conf. Series **368** 012018.

[8] Nilsson P et al. [ATLAS Collaboration] 2014, Next Generation PanDA Pilot for ATLAS and Other Experiments, J. Phys.: Conf. Series **513**.

[9] Panitkin S et al. 2015, Integration of PanDA workload management system with Titan supercomputer at OLCF. Proceedings of the CHEP 2015 conference J. Phys.: Conf. Series.
[10] Maier T et al. [ATLAS Collaboration] 2015, ATLAS I/O Performance Optimization in As-Deployed Environments. Proceedings of the CHEP 2015 conference J. Phys.: Conf. Series.
[11] Gardner R et al. [ATLAS Collaboration] 2014, Data federation strategies for ATLAS using XRootD, J. Phys.: Conf. Series **513** 042049.
[12] Ito H et al. 2015, Current Status of the Ceph Based Storage Systems at the RACF. Proceedings of the CHEP 2015 conference J. Phys.: Conf. Series.
[13] Tsulaia V et al. [ATLAS Collaboration] 2015, Fine grained event processing on HPCs with the ATLAS Yoda system. Proceedings of the CHEP 2015 conference J. Phys.: Conf. Series.
[14] National Energy Research Supercomputing Center (NERSC), http://www.nersc.gov/
[15] DOE Science Data Pilot Project: Granular Data Processing on HPCs Using an Event Service, http://scdoe.info/2014/11/11/torre-wenaus-brookhaven/
[16] Oak Ridge Leadership Computing Facility (OLCF) Titan, http://www.olcf.ornl.gov/