EPJ Web of Conferences **214**, 04034 (2019) https://doi.org/10.1051/epjconf/201921404034
CHEP 2018

Towards an Event Streaming Service for ATLAS data processing

Alex Brino[1], *Alessandro* Di Girolamo[2], *Wen* Guan[3], *Mario* Lassnig[2], *Tadashi* Maeno[4], *Nicolò* Magini[5,6,*], *Paul* Nilsson[4], *Vakhtang* Tsulaia[7], *Rodney* Walker[8], and *Torre* Wenaus[4], on behalf of the ATLAS Collaboration

[1]University of Udine, Udine, Italy
[2]CERN, Meyrin, Switzerland
[3]University of Wisconsin, Madison, USA
[4]Brookhaven National Laboratory, Upton, USA
[5]INFN Genova, Genova, Italy
[6]Iowa State University, Ames, USA
[7]Lawrence Berkeley National Laboratory, Berkeley, USA
[8]Ludwig Maximilians Universität, München, Germany

Abstract. The ATLAS experiment at the LHC is gradually transitioning from the traditional file-based processing model to dynamic workflow management at the event level with the ATLAS Event Service (AES). The AES assigns fine-grained processing jobs to workers and streams out the data in quasi-real time, ensuring fully efficient utilization of all resources, including the most volatile. The next major step in this evolution is the possibility to intelligently stream the input data itself to workers. The Event Streaming Service (ESS) is now in development to asynchronously deliver only the input data required for processing when it is needed, protecting the application payload from WAN latency without creating expensive long-term replicas. In the current prototype implementation, ESS processes run on compute nodes in parallel to the payload, reading the input event ranges remotely over the network, and replicating them in small input files that are passed to the application. In this contribution, we present the performance of the ESS prototype for different types of workflows in comparison to tasks accessing remote data directly. Based on the experience gained with the current prototype, we are now moving to the development of a server-side component of the ESS. The service can evolve progressively into a powerful Content Delivery Network-like capability for data streaming, ultimately enabling the delivery of 'virtual data' generated on demand.

1 Introduction

The ATLAS experiment [1] at the LHC [2] has accumulated more than 400 Petabytes of data processed on a globally distributed network of computing centers capable of providing about 6M CPU-hours/day. Despite the availability of a processing facility of this scale, the experiment's computing is still resource constrained and its physics programme can be enhanced by exploiting any additional computing resource. Furthermore, the computing needs of ATLAS

*e-mail: Nicolo.Magini@cern.ch

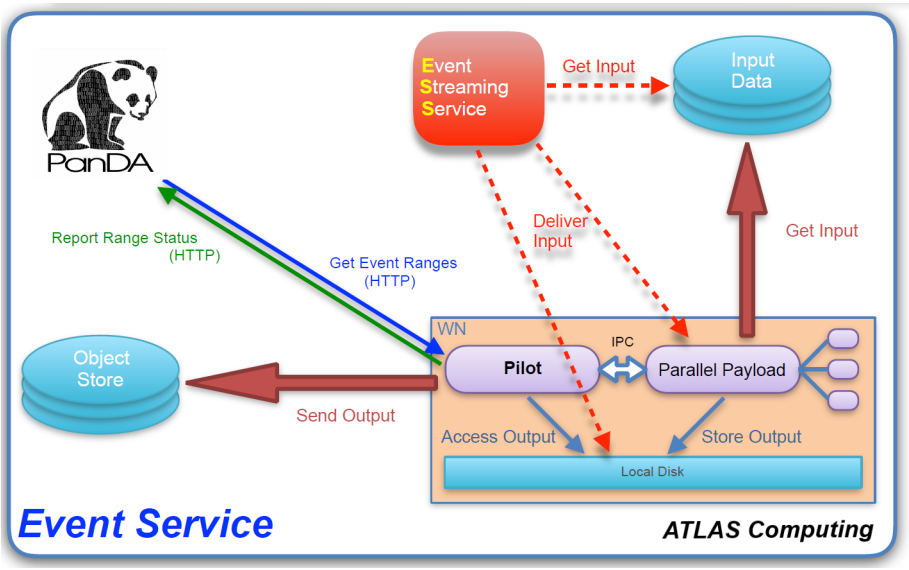

Figure 1. A schematic view of the ATLAS Event Service.

are expected to grow by more than one order of magnitude with the increase in data size and complexity foreseen after the High Luminosity upgrade of the LHC around 2026, making the resource constraints much more limiting. Opportunistic resources such as High Performance Computing centers, commercial clouds, volunteer computing or shared grid resources are a prime target for expanding the computing pool available to ATLAS. A common feature of these diverse resources is the volatility and unpredictability of job slot lifetimes. In recent years, ATLAS has then developed a new approach to make maximal usage of such resources, called the ATLAS Event Service (AES) [3]. The AES takes advantage of the flexible workflow management of PanDA and its JEDI component [4] to implement a processing model in which workflows can be dynamically managed at high granularity, assigning individual events to worker processes, and streaming out the output data almost continuously (every 10-30 minutes) to a remote object store. In this way, the AES can fill resources efficiently without having to fine tune job duration to resource lifetime, and if the worker is terminated prematurely (for example, in case of preemption), the amount of work lost is minimized. A schematic description of AES is given in Figure 1. The Event Service is currently in production for Monte Carlo simulation on HPC, cloud, and grid computing platforms, and is undergoing commissioning on a wider set of resources [5], [6].

2 Event Streaming Service

In the current Event Service implementation, processes running on the worker nodes require access to the full input files, which are usually pre-staged on a local storage element. This will become a major limitation in the future, especially when the AES will expand beyond Monte Carlo simulation to run other, more I/O-intensive, types of workflows requiring larger input datasets. To avoid creating additional expensive replicas of the input files, the application will then need to access the input data directly from a remote storage over the Wide Area Network; this process will need to be optimized to minimize the impact on efficiency of the increased WAN latency.

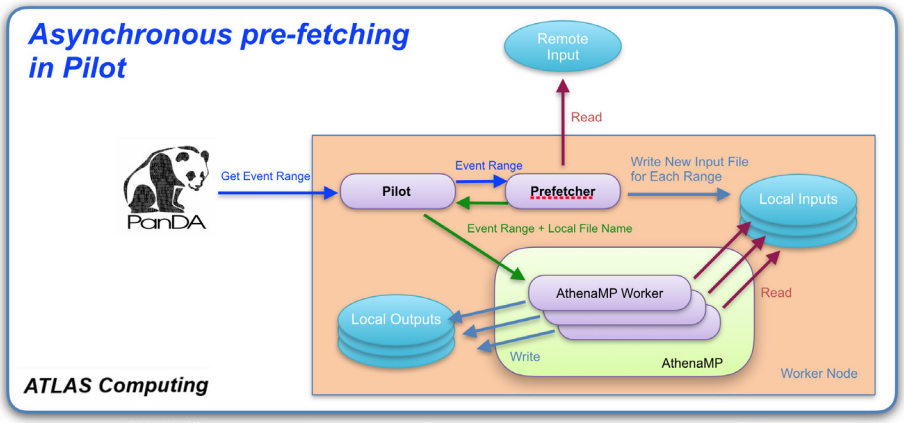

Figure 2. A schematic view of the Prefetcher implementation of the Event Streaming Service.

To decouple event data retrieval from the processing, we have then started the development of a new component in the system, the Event Streaming Service (ESS). The ESS is designed to intelligently stream the input data at the high granularity typical of AES, asynchronously pre-fetching from storage only the data that is actually needed (down to individual event ranges and below), and marshalling the events to the workers just in time for processing. The ESS may also introduce additional optimizations such as using local transient caches. In this way we envision to reduce replica counts and the related storage costs, and to make full use of our high bandwidth networking.

The Event Streaming Service is currently in development in steps at increasing levels of sophistication, described in the following.

3 The Prefetcher prototype

The first prototype of the Event Streaming Service (ESSv1) was implemented as a Prefetcher process started on the worker node by the supervisor (the pilot job) to perform asynchronous data retrieval in parallel to AthenaMP [7], the ATLAS event processing application. The implementation is described in Figure 2.

After setting up the Prefetcher and payload application, the pilot starts to retrieve from PanDA/JEDI the event ranges assigned by AES for processing; however, unlike conventional ATLAS Event Service jobs, the pilot passes the event ranges to the Prefetcher first. The Prefetcher, which is simply implemented as an additional AthenaMP process configured to perform only event I/O, opens the input file over the WAN using a remote access protocol (usually xrootd [8]), reads the given event range, and duplicates it to a small file on the local scratch disk of the worker node. The Prefetcher then reports back to the pilot the path to the local file as physical file name of the event range, and becomes ready for the next event range. The pilot in turn delivers the updated event range to AthenaMP, which will then consume it reading the input data from the local file.

4 Performance of remote access through the Prefetcher

The Prefetcher was implemented to study the feasibility of streaming input events in Event Service processing. In order to validate the ESS approach, “standard” AES simulation tasks

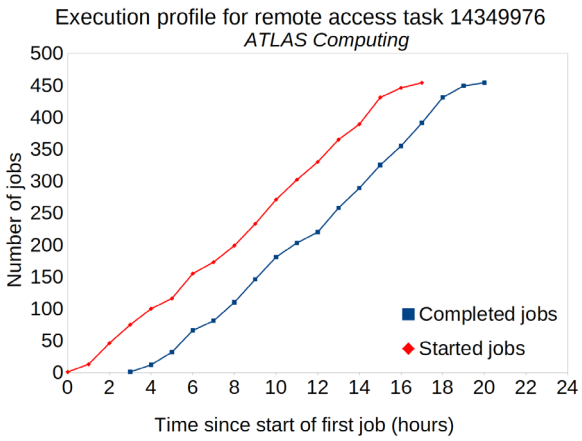

Figure 3. Number of started/completed jobs as a function of time for a test task running with direct remote access.

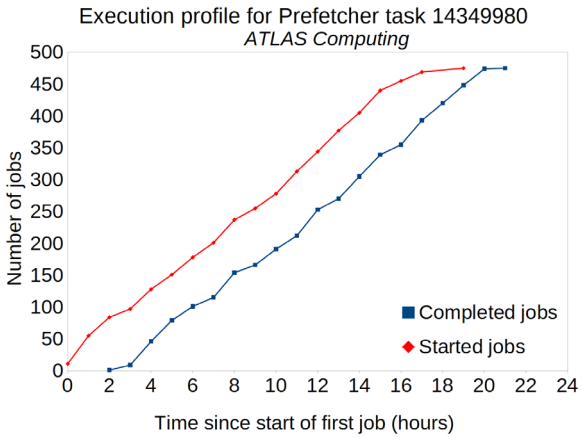

Figure 4. Number of started/completed jobs as a function of time for a test task running with remote access through Prefetcher.

were reconfigured to use the Prefetcher for data access: while these non I/O-bound tasks are not the main use case for ESS in the longer term, they provided a useful reference for comparison with a tried-and-tested AES application.

As a first step, the ATLAS production system was instrumented to allow remote access of input files, instead of pre-staging them by default. Validation tasks processing 100000 events were then defined and submitted to selected resources, running for example on the worker nodes of the ATLAS Great Lakes Tier-2 computing center and reading the input data from the storage of the Brookhaven Tier-1 computing center.

In the first testing phase, issues affecting remote access were identified and addressed. The most common problems observed were stuck or slow data connections, and corrupted

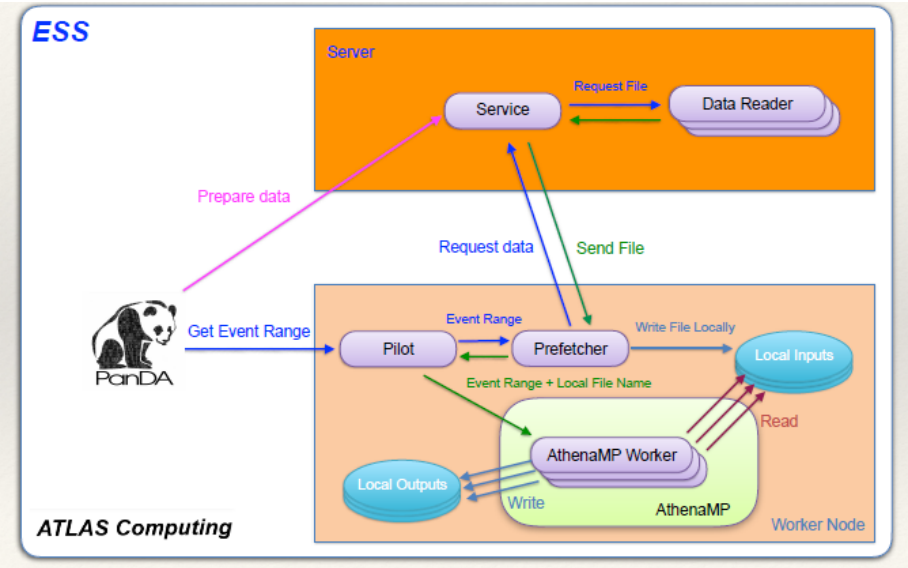

Figure 5. A schematic view of the architecture of Event Streaming Service v2.

or missing source replicas. A reconnection mechanism was implemented to work around the first issue; to improve robustness against the second problem, an automated redirection to a different replica through the Rucio [9] data management system may be used.

In the second phase of validation, identical tasks were then submitted, configured to perform remote access directly or through prefetching, and task metrics such as completion time or success rate were compared between the two configurations. No significant difference was observed, as shown for example in Figures 3 and 4 for the overall running time of one test task, demonstrating that the additional prefetching step does not introduce a significant overhead in the workflow. On the other hand, as expected for these simulation workflows, no large improvement was measured from prefetching either. Work is currently underway to extend the system to run I/O-heavier workflows which should benefit more from ESS.

5 Design of Event Streaming Service v2

The next step foreseen in the development of the Event Streaming Service (ESSv2) is the decoupling of the preparation of input event ranges from the prefetching. The ESS adds a server component close to the data that receives the data streaming requests in advance from the workflow management system, and runs the AthenaMP processes to extract the requested events from the input files. The ESSv2 server can apply its knowledge about data availability and popularity to optimize on demand the preparation of the event ranges to be delivered to the workers, possibly filtering the events and even the objects within events.

Instead of accessing the remote input files directly, the Prefetcher process in the pilot becomes a client asking from the server for the event ranges it was assigned. The server can then marshal and stream to the client only the needed data.

A diagram of the planned ESSv2 architecture is presented in Figure 5.

Since the preparation of input data will be performed independently from processing, the clients will need to be able to discover which event ranges are available and where. Book-

keeping of these transient data requires a highly scalable catalog, potentially capable of tracking hundreds of millions of event ranges depending on the granularity of ESS products and their lifetime. We are investigating the possibility to track these event ranges in the Event WhiteBoard (EWB), a new ATLAS service to register arbitrary metadata for event collections, currently in early development. A prototype database and APIs have been deployed to register in EWB the events ready for streaming, and the service is currently under testing.

6 Outlook and conclusions

We have developed a working ESS prototype based on client side prefetching, which has been useful to improve the support for remote data access in the ATLAS production system. Performance measurements on simulation tasks have proven that event streaming is feasible, and we are now planning additional optimizations, such as tuning the number of input events to prefetch in the background, and caching the prefetched events. The next step will be to extend ESS support also to I/O-heavy workflows, for example the production of derived samples from reconstructed data, which have large, potentially distributed inputs.

Building on the experience gained with the first ESS version, we have started the design of ESSv2, based on a server component and on a scalable Event WhiteBoard catalog. The new ESS can become a service which generates data delivery plans based on central intelligence, like a Content Delivery Network for event data, opening the way for the creation of "virtual datasets" on demand and reducing the amount of persistent storage needed by the experiment.

References

[1] ATLAS Collaboration, JINST **3** S08003 (2008)
[2] L. Evans and P. Bryant (editors), JINST **3** S08001 (2008)
[3] P Calafiura et al., J. Phys.: Conf. Ser. **664** 062065 (2015)
[4] K. De et al, J. Phys.: Conf. Ser. **664** 062035 (2015)
[5] D. Benjamin et al., J. Phys.: Conf. Ser. **762** 012027 (2016)
[6] E. Fullana Torregrosa et al., Grid production with the ATLAS Event Service, Proceedings of the CHEP 2018 conference EPJ Web Conf. (2018)
[7] P. Calafiura et al., J. Phys.: Conf. Ser. **664** 072050 (2015)
[8] A. Dorigo et al., WSEAS Transactions on Computers **4(4)**, 348–353 (2005)
[9] M. Barisits et al., The ATLAS Data Management System Rucio: Supporting LHC Run-2 and beyond, Proceedings of the ACAT 2017 Conference J. Phys.: Conf. Ser. (2017)