ATLAS EventIndex general dataflow and monitoring infrastructure

To cite this article: Á Fernández Casani et al 2017 J. Phys.: Conf. Ser. 898 062010

View the article online for updates and enhancements.
ATLAS EventIndex general dataflow and monitoring infrastructure

Á Fernandez Casaní¹, D Barberis², A Favareto², C García Montoro¹, S González de la Hoz¹, J Hřívňáč³, F Prokoshin¹, J Salt¹, J Sánchez¹, R Többicke⁴ and R Yuan⁴ on behalf of the ATLAS Collaboration

¹Instituto de Física Corpuscular, Universitat de València and CSIC, Valencia, Spain
²Università Genova and INFN, Genova, Italy
³LAL, Université Paris-Sud and CNRS/IN2P3, Orsay, France
⁴Universidad Técnica Federico Santa María, Valparaíso, Chile
⁵CERN, Geneva, Switzerland

E-mail: Alvaro.Fernandez@ific.uv.es

Abstract. The ATLAS EventIndex has been running in production since mid-2015, reliably collecting information worldwide about all produced events and storing them in a central Hadoop infrastructure at CERN. A subset of this information is copied to an Oracle relational database for fast dataset discovery, event-picking, crosschecks with other ATLAS systems and checks for event duplication. The system design and its optimization is serving event picking from requests of a few events up to scales of tens of thousand of events, and in addition, data consistency checks are performed for large production campaigns. Detecting duplicate events with a scope of physics collections has recently arisen as an important use case. This paper describes the general architecture of the project and the data flow and operation issues, which are addressed by recent developments to improve the throughput of the overall system. In this direction, the data collection system is reducing the usage of the messaging infrastructure to overcome the performance shortcomings detected during production peaks; an object storage approach is instead used to convey the event index information, and messages to signal their location and status. Recent changes in the Producer/Consumer architecture are also presented in detail, as well as the monitoring infrastructure.

1. Introduction
The EventIndex project [1, 2] is designed to index and catalog all produced events, or particle collisions, of the ATLAS experiment [3]. Information per event, which includes identification, trigger patterns and pointers to the files where it can be found, is reliably collected worldwide by the Distributed Data Collection task of the EventIndex project. All the EventIndex data is stored at the Hadoop backend [4, 5] infrastructure at CERN for later retrieval and analysis, and a subset of this information is copied to an Oracle relational database for a selected range of use cases [6]. Just last year 2015, 12 billion real events were produced in 1 million files, summing up all processing versions, and 5 billion events were simulated storing them in 8 million files.

The use cases that we are currently tackling are the following ones:

- Event picking: users are able to select single events depending on constraints like a list of run numbers and event numbers, trigger stream, event format and processing version.
The system finds the events and returns pointers to them to the user that issued the query, who can then use the data management tools to retrieve them. The service is able to handle hundreds of concurrent users, with requests ranging from 1 event (most common case) to thousands of events.

- Production consistency checks: each production cycle should be checked for completeness (the number of produced events is the same as the number of input events) and consistency (no duplicate events). There are also overlap detection checkings on demand, constructing the matrix identifying common events across the different files produced in the derivation framework [7].

- Trigger checks and event skimming: the population of events that passed given triggers and of events that passed multiple triggers can be retrieved from the event catalogue. Similarly a trigger-based event selection can be done, retrieving the references to the selected events and then the events themselves. In addition a trigger overlap detection is possible, counting the number of events in a real data run/stream satisfying trigger X which also satisfies trigger Y.

In this paper we describe the general architecture of the project with emphasis on the description of the data flow within the system, and some statistics obtained during the operation for this Run 2 phase (2015-2018) with our monitoring systems. We also describe some performance and operation issues which are being addressed by recent developments to improve the throughput of the overall system.

2. EventIndex architecture

In Figure 1 we can observe the general data flow of the EventIndex data and the different parts of the system. The Distributed Data Collection [8] follows a Producer/Consumer architecture, where Producers run at distributed sites, and Consumers centrally at CERN. The Producer entity collects event information from the data files, from CERN and hundreds of Grid sites worldwide, and sends the payload using a messaging infrastructure and ActiveMQ [9] brokers at CERN. The Consumer entities retrieve messages from the brokers, and write the information in the backend HDFS filesystem [10].

The backend storage and processing elements are in charge of efficiently storing and indexing all the data. The main storage is using Hadoop technologies [4], organizing data into dataset oriented structures, physically stored in Mapfiles and in Hbase. One subset of this information is retrieved from the mapfiles in HDFS by the Oracle Importer, which stores it in a Oracle Database [6]. Both subsystems have their own query services for final users. The monitoring system is horizontal to all the rest of the elements of the architecture, and is described in section
In Figure 2 it is depicted in detail the data flow with the messaging system, where payload from the consumers is divided into smaller messages of 10KB each, to keep the sending procedure and broker queues agile. Messages from the same Producer are logically grouped, so will be eventually consumed by exactly one Consumer, and are also sent within a transaction so if there is any sending error, no partial processing occurs. In Section 4 we will describe part of the architecture that explores an alternative transport method using an Object Store as transient storage for the EventIndex payload.

3. Monitoring and statistics
Monitoring plays an important role in the EventIndex project, to understand if the system is behaving correctly or to identify possible pitfalls in a particular moment. Due to the distributed nature of the elements of the architecture, the monitoring is also distributed with agents across different entities in the Grid, the Tier-0 farm at CERN, and the central infrastructure elements of the Hadoop and Oracle backends. The information is gathered from all these entities and submitted to a central Kibana [11] server, which is a data visualization software on top of the ElasticSearch [12] engine, that provides a number of tools to display information, like dashboards, queries, filters and time ranges. A general view of the data flow of the monitoring information is depicted in Figure 3.

The general working mode of the monitoring system was described in [13], and is basically using different approaches for data collecting and processing. Monitoring information is collected by jobs running on each one of the EventIndex servers automatically at specified time managed by a periodical command scheduler. The gathered information is organized into xml files and then pushed to the ATLAS computing services monitoring server running Kibana. The production of EventIndex data at CERN is managed by the Tier-0 team, that indexes the desired datasets with batch jobs running the producer processes. There is a framework system...
Figure 3. Dataflow of the monitoring information.

Table 1. Producer status information per processed file.

| Element      | Description                                      |
|--------------|--------------------------------------------------|
| PRODID       | Producer identifier                              |
| GUID         | Global file identifier                           |
| JOBID        | Job identifier                                   |
| TASKID       | Task identifier with suffix G(grid) or T(Tier0)  |
| SEQID        | Sequence identified of the message inside a group or sequence |
| STARTTIME    | Starting time sending data (unix time in msecs)  |
| ENDTIME      | End time sending data                            |
| FILESTARTPROCTIME | Starting time creating EI file                  |
| FILEENDPROCTIME | End time creating EI file                    |
| FILENO       | Number of order of GUID                          |
| EVTS         | Number of events                                 |
| UNIQEVTS     | Number of unique events (run-evt)                |
| NMSG         | Number of messages sent                          |
| BYTES        | Number of bytes of the payload (without headers) |

for the Tier-0 jobs themselves, that is also consulted by the Supervisor entity of Data Collection task of EventIndex to follow the production. The monitoring information itself is submitted by the Producers with status messages through a specific monitoring queue of broker. This information is sent with a message per processed filed identified by its GUID, or globally unique identifier (see table 1), and a message when the Producer finishes with a summary of all the processed files.

The production of EventIndex grid jobs is managed by the so called Open-Ended production. We are using ProdSys2 [14, 15], a second generation ATLAS production system, to index datasets as soon as they appear in the AMI metadata catalogue [16] and are marked as valid and completed. Information on data produced/modified daily on the Grid is gathered by automatic scripts by querying the AMI catalog for VALID datasets from specific datasets for real and Monte Carlo simulated events produced. Then a catalogue in the ATLAS Distributed Data Management system [17] is queried for the identifiers of the datasets to feed the EventIndex processing jobs. Collected datasets are attached to the special technical containers by a cron script, to be picked up by the EventIndex production tasks. There is one technical container for each type of data to be indexed and trigger information processing setting. A new set of
technical containers is being created when a new version of the EventIndex producer appears. The Open-Ended production can be monitored with the Kibana graphics and reports the number of datasets found and attached at every moment, as can be seen in Figure 4.

For each dataset that appears in the technical container, ProdSys2 creates a Grid task that runs the Producer in a pilot job. The consumers that receive the EventIndex payload also submit monitoring statistics, that include the information by processed GUID similar as it was described at table 1 and since Consumers are long-running processes, a status message every 60 seconds.

The monitoring machines at CERN receive a copy of these status messages sent by the Producers and Consumers, and produce high level graphics of current status of processing of GUIDS on the grid and Tier-0. In addition there are other monitoring frameworks provided by CERN IT messaging group based on Graphite [18] that are being used to retrieve the status of the messaging brokers. Figure 5 shows the number of messages produced per second (blue), and the number of messages consumed per second (green) in the first 6 months of year 2016. We are currently transmitting messages using 5 ActiveMQ brokers physically located at CERN Meyrin and Wigner sites. The blue line represents the number of messages that are sent to the brokers by the Producers, which are a varying number up to 400 simultaneous processes. The usual rate is about 100 messages/s produced, with peaks of 2500 messages/s seen in the graph during this period. The consumed messages represented in the green line are due to the Consumer tasks. We commonly have 3 long-lived attached Consumers per broker, distributed among different physical machines at CERN, and as can be seen in the figure some times all the produced messages are not consumed instantaneously.

As can be seen in Figure 6, these peaks with a maximum seen of 4 K msg/s produced are not consumed instantaneously, in this case with a rate of about 1K msg/s, eventually producing
a backlog of messages in the brokers waiting to be handled to the Consumers. This is currently not a problem because these peaks are infrequent, and in the case they are produced the backlog is absorbed in some hours, but in the future might impose a bottleneck for the system with the expected rises in the production rates.

For this reason we study in the following sections an alternative approach to deal with this bottlenecks. The machines hosting the Consumers are also monitored, along with other machines involved including the Hadoop cluster, allowing the monitoring of cpu and memory consumption, and other details that are stored in an ElasticSearch cluster and can be visualized as interactive plots such as pie charts, histograms, text, maps and other. One interesting information that is available is the amount of data stored in Hadoop HDFS filesystem [10], and that is summarized in Figure 7.

We can see the quantity of real data reaches 50 TB of information, from which 1/3 is from 2016, 1/3 from 2015, and the last 1/3 from previous years. The other 31 TB of information comes from Monte Carlo 2015 simulated data, summing up a total of 81 TB of events data stored in the Hadoop Cluster.

4. Object store data flow
As we have seen in previous sections, the performance of the data collection is satisfactory, but in the future there might be peaks of production that we would like to digest faster. In this direction we are exploring alternatives for the data flow, and the first option is conveying the

**Figure 6.** Example of produced (blue) and Consumed (green) messages during high peak production.

**Figure 7.** Real (green) and Monte Carlo simulated (yellow) data stored in Hadoop HDFS.
Figure 8. Data Collection architecture using an Object Store as transient storage for the EventIndex data.

EventIndex data within a transient Object store. An Object Store [19] is a data storage solution with a flat namespace, in opposition to a traditional file system with multiple directories and a hierarchical namespace. This characteristic makes it specially suitable for applications with big necessities in space. There is no restriction in object size, and the metadata associated with the object is stored within itself, making access fast. These characteristics make object stores more scalable than the traditional block storage solutions, so adding more space to an existing store is seamlessly achieved adding more servers to the object store. Other software elements of the ATLAS distributed computing ecosystem are using an object store as part of their processes, so it is currently supported by the community. For our data collection task, the object store would transport the EventIndex data, with the idea of substituting the messaging system. There are some changes needed in the data collection architecture, that are depicted in Figure 8. At step 1, the producer reads the information from AOD (Analysis Object Data) files, extracting the required EventIndex information. At step 2, submits the information to the object store, obtaining a reference that is used to identify this object. This object id is sent with a control message to a new entity, the EventIndex Supervisor at step 3. We are proposing still using the messaging system to convey this control messages, as these are small messages compared with the real payload, and it is convenient at this development stage. The EventIndex supervisor is in charge of orchestrating the validation of the data procedure. It receives the control messages with status information about what EventIndex data has been produced. It consults external sources to know about jobs running at CERN Tier-0 and worldwide [14, 15], and about the files that belong to a particular dataset [17]. As a dataset comprises several produced files, when a dataset is complete and can be validated, submits this validation information to Consumer at step 4. At this point, Consumer received the validation message from the Supervisor, and as at step 5 retrieves the EventIndex information of the files of a validated dataset from 1 or more objects of the Object Storage. This effectively changes the operation mode compared with the pure previous messaging approach [8] depicted in Figure 2, from a push to a pull model. So only validated information is retrieved from the ObjectStore, instead of receiving all the information with a messaging system and validate later. This potentially reduces the information that goes through Consumers, streamed directly from the Object Store and decompressed on the fly, and stores it at step 6. In addition this pull model saves temporary staging space at the backend HDFS filesystem, as no duplicate and only validated data is stored in final format in HDFS.
filesystem. Later on, at step 7 Consumer will report that the datasets have been processed, so the information of all the stages and processes can be followed with the Supervisor and for the managers via a web interface.

5. Conclusions and future work
In this paper we have presented the current data flow of the EventIndex project based on a messaging system, that is running in production since 2015, successfully collecting millions of messages and thousands of datasets. We have also presented the monitoring infrastructure that we are using, and its key role to detect the correct working mode of all the components. It also helps to identify possible bottlenecks, and allow the operational monitoring and corrective actions done by developers and experts. We have outlined the next generation prototype based on a transient object store that is in development, envisaged to cope with future productions rates and to improve and automate operations. In this direction the next steps will be to check the performance gained with the new approach, and measure it in a real production scenario.

References
[1] Barberis D, Cranshaw J, Favareto A, Fernández Casaní A, Gallas E, González de la Hoz S, Hřívnač J, Malon D, Nowak M, Prokoshin F, Salt J, Sánchez Martínez J, Tőbbicke R and Yuan R 2016 Nuclear and Particle Physics Proceedings 273-275 913–918
[2] Barberis D, Cárdenas Zárate S, Crazhaw J, Favareto A, Fernández Casaní A, Gallas E, Glasman C, González De la Hoz S, Hřívnač J, Malon D, Prokoshin F, Salt Cairols J, Sánchez J, Tőbbicke R and Yuan R 2015 Journal of Physics: Conference Series 664 042003
[3] ATLAS Collaboration 2008 Journal of Instrumentation 3 S08003
[4] Apache Hadoop http://hadoop.apache.org/. Retrieved on 2016-09-11
[5] White T 2012 Hadoop: The Definitive Guide (O’Reilly Media, Inc.) ISBN 1449311520, 9781449311520
[6] Gallas E, Dimitrov G, Petrova P T, Baranowski Z, Canali L, Formica A and Dumitru A (ATLAS Collaboration) 2016 Journal of Physics: Conference Series to appear
[7] Catmore J, Cranshaw J, Gillam T, Granstad E, Laycock P, Ozturk N and Stewart G A 2015 Journal of Physics: Conference Series 664 072007
[8] Sánchez J, Casaní A F and de la Hoz S G 2015 Journal of Physics: Conference Series 664 042046
[9] ActiveMQ http://activemq.apache.org/. Retrieved on 2016-09-11
[10] Shvachko K, Kuang H, Radia S and Chansler R 2010 The Hadoop distributed file system 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies, MSST2010
[11] Kibana https://www.elastic.co/products/kibana. Retrieved on 2016-12-01.
[12] Elasticsearch https://www.elastic.co/products/elasticsearch. Retrieved on 2016-12-01.
[13] Barberis D, Zrate S C, Favareto A, Casaní A F, Gallas E, Montoro C G, de la Hoz S G, Hrivnac J, Malon D, Prokoshin F, Salt J, Sanchez J, Toebbicke R, Yuan R and Collaboration A 2016 Journal of Physics: Conference Series 762 012004
[14] Klimentov A, Buncic P, De K, Jha S, Maeno T, Mount R, Nilsson P, Oleynik D, Panitkin S, Petrosyan A, Porter R J, Read K F, Vaniachine A, Wells J C and Wenaus T 2015 Journal of Physics: Conference Series 608 012040
[15] De K, Klimentov A, Maeno T, Nilsson P, Oleynik D, Panitkin S, Petrosyan A, Schovancova J, Vaniachine A and Wenaus T 2015 Journal of Physics: Conference Series 664 062035
[16] Albrand S, Fulachier J and Lambert F 2010 Journal of Physics: Conference Series 219 042030
[17] Garonne V, Vigne R, Stewart G, Barisits M, eermann T B, Lassnig M, Serfon C, Goossens L, Nairz A and the Atlas Collaboration 2014 Journal of Physics: Conference Series 513 042021
[18] Graphite https://launchpad.net/graphite. Retrieved on 2016-12-01.
[19] Mesnier M, Ganger G R and Riedel E 2003 IEEE Communications Magazine 41 84–90 ISSN 0163-6804