PROCEEDINGS OF SCIENCE

The ATLAS EventIndex: an event catalogue for experiments collecting large amounts of data

Dario Barberis and Andrea Favareto [1]
Università di Genova and INFN, Genova, Italy
E-mail: dario.barberis@cern.ch, andrea.favareto@ge.infn.it
Jack M. Cranshaw and David Malon
Argonne National Laboratory, Argonne, IL, United States
E-mail: cranshaw@anl.gov, malon@anl.gov
Álvaro Fernández Casaní, Santiago González de la Hoz, José Salt Cairols and Javier Sánchez
Instituto de Fisica Corpuscular (IFIC), University of Valencia and CSIC, Valencia, Spain
E-mail: alvaro.fernandez@ific.uv.es, sgonzale@ific.uv.es, salt@ific.uv.es, sanchezj@ific.uv.es
Elizabeth J. Gallas
Oxford University, Oxford, United Kingdom
E-mail: e.gallas1@physics.ox.ac.uk
Julius Hřivnáč and Ruijun Yuan
LAL, Université Paris-Sud and CNRS/IN2P3, Orsay, France
E-mail: julius.hrivnac@cern.ch, ruijun.yuan@cern.ch
Marcin Nowak
Brookhaven National Laboratory, Upton, NY, United States
E-mail: marcin.nowak@cern.ch
Fedor Prokoshin
Universidad Técnica Federico Santa María, Valparaíso, Chile
E-mail: Fedor.Prokoshin@cern.ch
Rainer Töebbicke
CERN, Geneva, Switzerland
E-mail: Rainer.Toebbicke@cern.ch

Modern scientific experiments collect vast amounts of data that must be catalogued to meet multiple use cases and search criteria. In particular, high-energy physics experiments currently in operation produce several billion events per year. A database with the references to the files including each event in every stage of processing is necessary in order to retrieve the selected events from data storage systems. The ATLAS EventIndex project is developing a way to store the necessary information using modern data storage technologies that allow saving in memory key-value pairs and select the best tools to support this application from the point of view of performance, robustness and ease of use.

The International Symposium on Grids and Clouds (ISGC) 2014
March 23-28, 2013
Academia Sinica, Taipei, Taiwan

[1] Speaker and Corresponding Author

PoS(ISGC2014)023

1. Introduction

The ATLAS experiment has a catalogue of all real and simulated events in a database (TAGDB). The TAGDB is implemented in Oracle, as that was the only proven technology that could hold the impressive amount of expected data when this project started long before the start of LHC operations [1] [2][3][4]. The main purposes of this database were the selection of events on the basis of physical variables, the identification of files that contain a list of events already selected by other means, and consistency checks on the completeness of event processing.

In TAGDB each event is recorded several times, once for each reconstruction cycle, and contains three main data blocks: information identifying the event (event and run number and trigger decisions), information allowing the localisation of the event data in files that contain it, and physical variables of the event, such as the number of reconstructed particles, their identification and properties, which might be useful to select events for specific analyses.

The TAGDB, with all related services and user interfaces, works for the retrieval of event information for limited numbers of events and for the technical checks described above; the event selection use case by physics variables turned out not to be generally useful because sometimes those quantities changed after the initial reconstruction and there was no mechanism in the system to update them. Nevertheless the TAGDB is slow, intrinsically complex and at times unreliable. In addition, the enormous amount of data (1 kB per record, which turns into several tens of TB in Oracle for all data collected by ATLAS so far) makes the implementation in Oracle particularly labour-intensive and expensive [5]. Oracle storage capacity was deployed to store the ~8 billion real and simulated events recorded during LHC Run1; future data taking rates will be higher, exceeding the limits of Oracle scalability required by the existing workflow [6][7].

As it was necessary to redesign the overall system, we tried to avoid the intrinsic schema limitations of relational database systems that hindered development of the TAGDB in the past and considered the data storage technologies that were developed in the academic and commercial computing world in recent years [8]. The technologies of "NoSQL" databases are used by large commercial information companies and are well adapted to this type of applications. At the same time, one can separate the parts of the project that can be of interest to other generic scientific applications from the parts closely related to ATLAS, and create a system that can be potentially useful for several other scientific and technical applications.

.

2. The EventIndex project

This project consists of the development and deployment of a catalogue of events for experiments with large amounts of data, such as those currently taking data with the LHC accelerator at CERN. The ultimate goal is to create an infrastructure that allows fast and efficient selection of events of interest, based on various criteria, from the billions of events recorded, using an indexing system that points to those events in millions of files scattered in a world-wide distributed computing system.

The project starts from the knowledge accumulated over the years with the development and operation of the ATLAS TAGDB [1]-[7]. The work has been divided into several stages:

- Studies of the patterns of data storage in terms of functionality, scalability and sustainability of the various types of NoSQL databases currently available; local tests of functionality and distributed test of scalability; system architecture design of both the database and the auxiliary services to enter the data and facilitate the data mining and extraction.
- Implementation of a catalogue in the selected technology; population of the database with all LHC Run1 ATLAS events; complete functional tests of the scalability of databases containing billions of records; comparison of performance with the ATLAS TAGDB. Adaptation and consolidation of services, including the automatic loading of data from each phase of reconstruction and event selection.
- Definition of public interfaces to load, search and retrieve the data; packaging and release of the produced software so that it can be used by other scientific communities.

The most innovative part of this project is the adaptation and application of the NoSQL technology for the cataloguing of data of a large experiment. ATLAS alone accumulated in 2011-2012 $2 \cdot 10^9$ real and $6 \cdot 10^9$ simulated events per year. If the production of events were uniform in time, this would correspond to a rate of creation of the records in TAGDB of about 100 Hz. Since each event is processed multiple times, the number of records and the record creation rate in TAGDB is considerably larger.

Each reconstruction campaign produces new versions of every event, in different formats. For ATLAS, they are ESD (Event Summary Data, full reconstruction output), AOD (Analysis Object Data, summary of reconstruction to be used for physics analysis) and several versions of ROOT n-tuples [9]-[10]. For each reconstruction campaign and for each format one can generate a key-value pair (where the value is always the navigational information, i.e. the identifier of the file that contains the event in question and the internal structured pointer) and add it to the original record. The result is that an event will always correspond to one and only one logical record in the database that contains the entire processing history of the event.

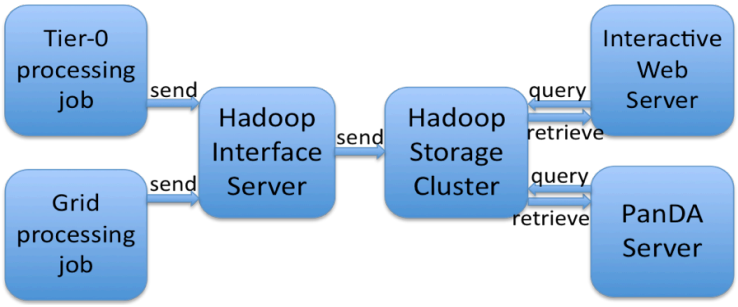

Figure 1: Block diagram of the EventIndex data flow. Information is produced by jobs running at CERN or on the Grid, inserted into the database, and retrieved by interactive and batch processes.

Two main applications must be developed simultaneously with the back-end database: the infrastructure to populate the database and the tools to search for the records and extract the

identifiers of the files that contain the events of interest. Because of the large amount of data being continuously processed, care must be taken in the development and optimization of these applications. The whole transmission chain of this information must be optimized in order to sustain rates of the order of 10 to 100 kHz. The system must also include cross-checks to assure catalogue completeness (no missing processing stages for any subset of events). These aspects of performance and scalability are extremely important to have a product that will be useful to the scientific community.

The major use cases of the EventIndex project are:

- Event picking: give me the reference (pointer) to "this" event in "that" format for a given processing cycle.
- Production consistency checks: technical checks that processing cycles are complete (event counts match).
- Event service: give me the references (pointers) for “this” list of events, or for the events satisfying given selection criteria, to be distributed individually to processes running on (for example) HPC or cloud clusters.

3. Core architecture

NoSQL technologies offer several advantages over relational databases for applications like the EventIndex: they scale linearly with the amount of data, there are many external tools to optimize data storage and data searches, they use clusters of low-cost Linux servers, including disks, and finally they belong to the world of open source software, so there is no license for the installation and use of these products.

Using NoSQL technologies, it is possible to store information for each event in a single logical record. The record is created upon recording of the event from the online system, with initially only limited information, such as event number, run number, time stamp, luminosity block number, which triggers selected the event, and the identifier of the file that contains the event in RAW format. ATLAS uses the GUID (Global Unique IDentifier) as identifier of each file but any other system can be used, provided it is capable of uniquely associating identifiers and files. This information can be stored as key-value pairs in NoSQL systems, together with the internal file navigation information.

As CERN-IT is providing a Hadoop [11] service, and Hadoop and the many associated tools seem to be a good solution for the EventIndex, we started testing the performance of solutions based on plain HDFS (the Hadoop file system) and/or HBase [12] (the Hadoop database system), using different data formats. For these initial tests we imported 1 TB of data from the TAGDB, which correspond to the first processing of the full real data collected in 2011.

The data are stored in Hadoop map files[2]. The map files may be later upgraded into some of their sub-formats in case further optimisation is needed. The event attributes are stored in several groups according to their nature and access pattern (constants, variables, references,

[2] The map file is a special Hadoop storage format - keys are stored in an "index" file and kept in memory for fast access, values are stored in standard sequential files on disk. Keys are ordered so that "get" operations are immediate and "search" operations are very fast. Note: It is not necessary to use the MapReduce framework to get the data.

physics). The data can be indexed in various ways (for example using inverted indices for the trigger information). Some index files will be created at upload, others will be added later. Index files will just have a key plus references to data files.

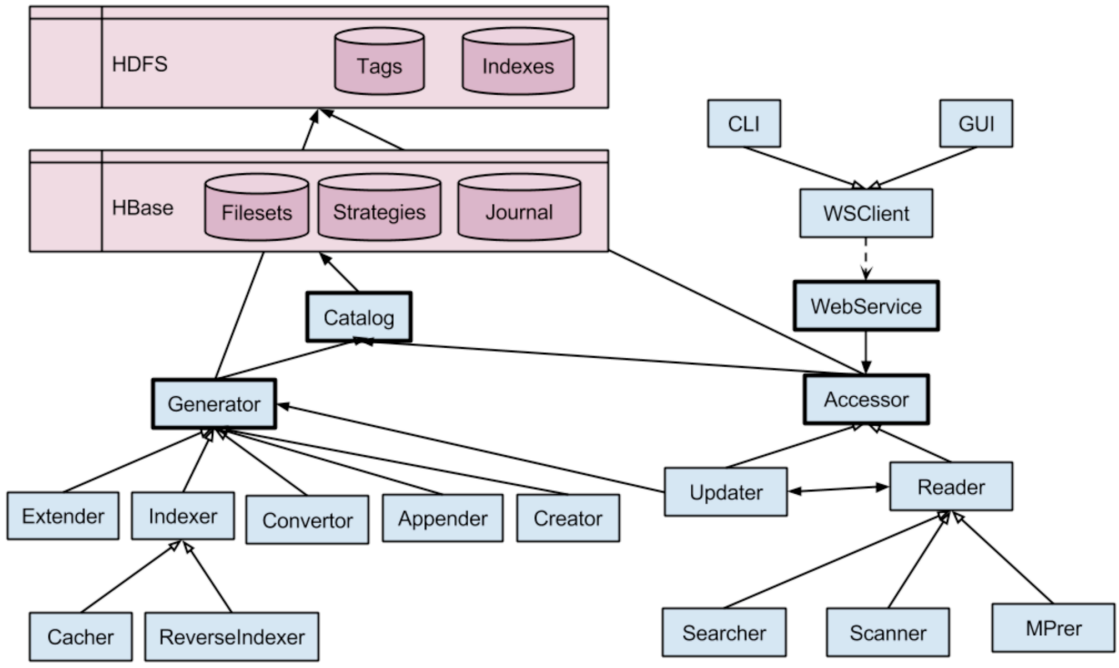

Figure 2: Core architecture of the EventIndex.

The searches can be then performed in two steps:

1. get reference from the index file,
2. using that reference, get the data,

Searches can be performed using keys, with immediate results (keys are in memory), or with full searches on file-based data.

Access to the data is achieved by a single and simple interface for:

- upload: copy file into HDFS and create basic structure
- read: get & search
- update: adding new (vertical) data (in general by a MapReduce job)
- index: create new indices

These interfaces are being implemented in Hadoop by Java classes with remote access and used through a Python client.

Figure 2 shows the internal organization of the core system. The Generator module creates new filesets and registers them into the Catalog. The Creator creates new filesets. The Extender adds information (new columns). The Appender adds new data. The Indexer creates new indices. The Convertor converts filesets to different formats, including the MapFiles. The Accessor allows remote access, through an HTTP server on a dedicated machine; a publicly visible WebService can be accessed through the CLI or the GUI. The Reader performs the search using one of the available tools (Searcher, Scanner or MapReducer), depending on the query.

Figure 3 shows the file organization and relationships in the Hadoop file system (HDFS). The data are stored in collections of "filesets", where each collection represents an event collection (grouping all events that have been processed by the same software version). The TagFiles can be files or directories of files. A collection of TagFiles makes a TagSet, each of which has: a master record, pointers to before/after filesets (vertical partitions), slave TagFiles (horizontal partitions) and index TagFiles. The catalogue checks the presence and consistency of all components. Each TagFile is represented by one entry in the HBase table.

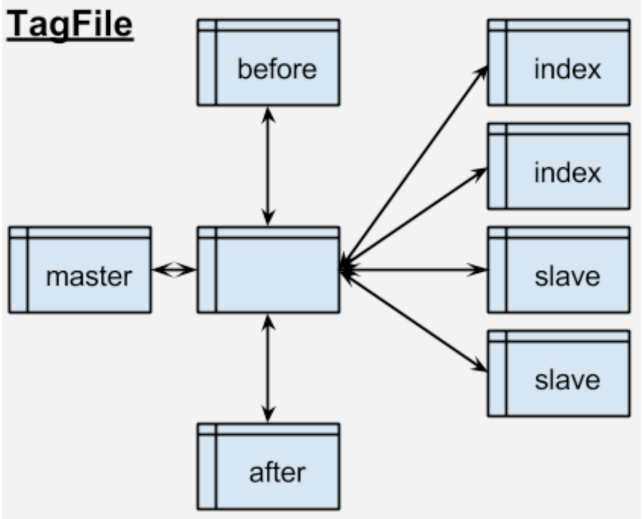

Figure 3: Data file organization in HDFS.

4. Data collection and upload

Data to be stored in the EventIndex are produced by all production jobs that run at CERN or on the Grid. For every permanent output file, a snippet of information, containing the file unique identifier and for each event the relevant attributes, has to be sent to the central server at CERN. The estimated rate (in May 2013, during the LHC shutdown) would be ~20 Hz of file records containing ~7 kHz of event records. During data-taking periods these numbers should be doubled; as both the data-taking rate and the Grid processing power are expected to increase by perhaps a factor two by 2015, the system should be ready to accept over 80 Hz of file records and insert into the back-end over 30 kHz of event records (plus contingency).

The architecture is based on producers of the EventIndex information on the worker nodes where event data are processed, a messaging system to transfer this information and asynchronous consumers that effectively insert the data in the Hadoop database. The producers run within the “pilot” process on each worker node and transmit the data using a messaging system to a queue in one of the endpoints. ActiveMQ [13] has been chosen as the messaging service, and STOMP [14] as the message protocol, as they are supported at CERN. An ssl endpoint at the broker can be used to secure the communications, and we can also have replication of the message queues in other sites, to permit duplication and high availability. The consumers are asynchronous processes, continually reading data, checking the data’s validity with the production system and passing the data into Hadoop storage and related cataloguing

tasks. The final step is to notify the production system of the successful insertion, and remove the data from the queue. Figure 4 shows a schematic view of the data collection system.

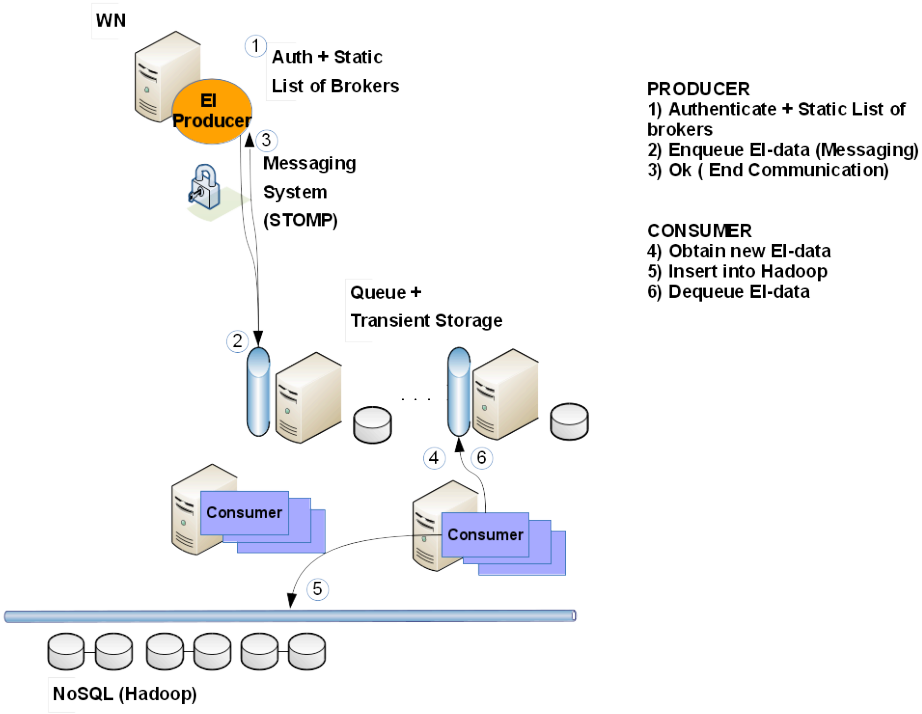

Figure 4: the data collection system.

The data producer process is split in two parts: the first part reads the data file to create an intermediate EventIndex file; the second one reads information from the EventIndex file, builds the messages and sends them to the broker. The EventIndex file is used as intermediate storage to decouple the processing of the input file from sending the messages. In the current prototype, the first step consists of a Python script that reads the event file and extracts the relevant information to be transmitted (event identification, trigger masks, references to the files containing the event). The EventIndex file is a SQLite3 [15] file of Python serialized objects, containing key-value pairs in one table "shelf". The second step is a Python script that reads the EventIndex file, builds a sequence of messages (about 10 kB each) and sends them to the broker. The messages are sent in a compact format encoded using the JSON [16] standard. Tests show that it is possible with a single broker to achieve the required data transfer rates, with peaks in the load of 200 kHz of event records, and about 60 kB/s. If an adequate number of consumer processes is active, the payload is retrieved immediately and no backlog is created, but even in case of temporary consumer unavailability, the backlog is only limited by the disk size on the broker server and can be absorbed as soon as the consumers restart. In practice, for robustness reasons, the broker system has to consist of at least two servers, located in different network domains.

The consumers receive the messages, decode the information in JSON format, order the events by run number in memory queues, and when a given threshold is reached, build CSV

records and append them to files in HDFS (the Hadoop file system). In order to guarantee scalability, several consumer processes must be run in parallel. Figure 5 shows the processes running in the data collection system.

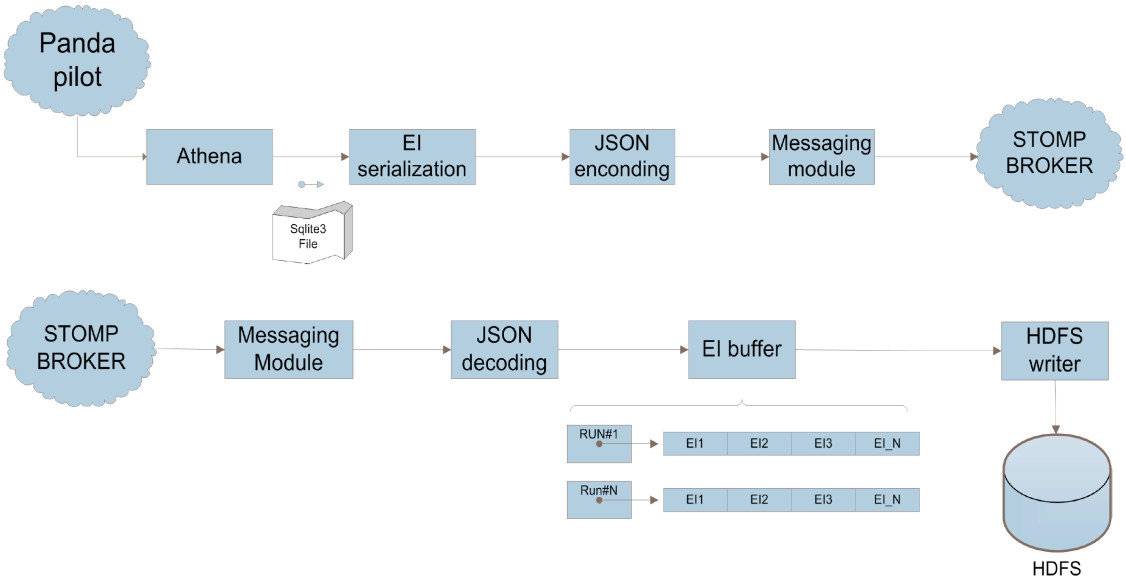

Figure 5: the two parts of the data collection system: (above) data are retrieved from running production jobs and transmitted to the broker using the messaging system; (below) data are read from the broker, decoded and stored into Hadoop.

5. Data queries and retrieval

The performance of data query interfaces is crucial for the usability of the EventIndex. For the initial tests, we imported a full year worth of ATLAS data (2011, first processing version) from TAGDB into Hadoop and catalogued them. This corresponds to just over 1 TB of data in CSV format, before internal replication. Simple data scan operations on this dataset using the available Hadoop cluster of 20 nodes take about 15 minutes; we consider this as the worst case, as normally no query would need to read in all data from disk for a full year of data taking. Queries that give the run number and event number (the event picking use case) are very fast; even using the so-far unoptimised "accessor" method these queries return their results in 3.7 seconds (just "fast"). For comparison, the same queries using a full Map-Reduce job return in 90 seconds. Again, these should be considered as worst cases as they are the results of the first prototype implementation.

An important use case is the query (or count) of the events that satisfied a given trigger condition, or a combination of trigger conditions. In ATLAS, every "fired" trigger condition is marked by a bit set to "1" in the relevant trigger word (Level-1, Level-2 and Event Filter). The correspondence between bits in the trigger words and the actual triggers depends on the run and luminosity block, as the list of active triggers and their possible prescale factors is adjusted according to the variations of the LHC luminosity, in order to maximize the collection of useful data for physics analyses. Queries to the EventIndex involving trigger chains need to be first decoded by knowing the trigger menu via the run number and finding in the COMA database

[17] the corresponding trigger name for each bit in the trigger words. The COMA database is stored in Oracle and accessed using ODBC drivers; the retrieved information is then used by MapReduce jobs in the Hadoop cluster. We are also considering replicating and caching the relevant data to Hadoop in order to improve the performance by using internal Hadoop calls instead of polling an external database.

6. Outlook

The EventIndex project is on track to provide ATLAS with a global event catalogue in advance of the resumption of LHC operations in 2015. The global architecture is defined and prototyping work is in progress; preliminary results are encouraging. Next steps consist of the full implementation of the final prototype, loaded with Run1 data, during the first half of 2014, and the completion of deployment and operation infrastructure by the end of 2014.

References

[1] Cranshaw J et al. 2008 Building a scalable event-level metadata service for ATLAS J. Phys. Conf. Ser. 119:072012, http://iopscience.iop.org/1742-6596/119/7/072012

[2] Cranshaw J et al. 2008 Integration of the ATLAS tag database with data management and analysis components J. Phys. Conf. Ser. 119:042008, http://iopscience.iop.org/1742-6596/119/4/042008

[3] Cranshaw J et al. 2008 A data skimming service for locally resident analysis data J. Phys. Conf. Ser. 119:072011, http://iopscience.iop.org/1742-6596/119/7/072011

[4] Mambelli M et al. 2010 Job optimization in ATLAS TAG-based distributed analysis J. Phys. Conf. Ser. 219:072042, http://iopscience.iop.org/1742-6596/219/7/072042

[5] Viegas F et al. 2010 The ATLAS TAGS database distribution and management: operational challenges of a multi-terabyte distributed database J. Phys. Conf. Ser. 219:072058, http://iopscience.iop.org/1742-6596/219/7/072058

[6] Cranshaw J et al. 2010 Event selection services in ATLAS J. Phys. Conf. Ser. 219:042007, http://iopscience.iop.org/1742-6596/219/4/042007

[7] The ATLAS Collaboration (Ehrenfeld W et al.) 2011 Using TAGs to speed up the ATLAS analysis process J. Phys. Conf. Ser. 331:032007, http://iopscience.iop.org/1742-6596/331/3/032007

[8] Barberis D et al. 2014 The future of event-level information repositories, indexing, and selection in ATLAS Proceedings of CHEP 2013, https://cds.cern.ch/record/1625848/files/ATL-SOFT-PROC-2013-046.pdf

[9] Brun R and Rademakers F 1997 ROOT – An Object Oriented Data Analysis Framework Nucl. Inst. & Meth. in Phys. Res. A 389 81-86, http://www.sciencedirect.com/science/article/pii/S016890029700048X

[10] Antcheva I et al. 2009 ROOT – A C++ framework for petabyte data storage, statistical analysis and visualization Comp. Phys. Comm. 180-12 2499–2512, http://www.sciencedirect.com/science/article/pii/S0010465509002550

[11] Hadoop: http://hadoop.apache.org

[12] HBase: http://hbase.apache.org

[13] ActiveMQ: http://activemq.apache.org

[14] STOMP: http://stomp.github.io

[15] SQLite3: http://www.sqlite.org/sqlite.html

[16] JSON: http://json.org

[17] Gallas EJ et al. 2012 Conditions and configuration metadata for the ATLAS experiment J. Phys. Conf. Ser. 396: 052033, http://iopscience.iop.org/1742-6596/396/5/052033