Письма в ЭЧАЯ. 2016. Т. 13, № 5(203). С. 974–979

USE OF THE HADOOP STRUCTURED STORAGE TOOLS FOR THE ATLAS EventIndex EVENT CATALOGUE

A. Favareto[1] *on behalf of the ATLAS Collaboration*

Università di Genova and INFN, Genova, Italy

The ATLAS experiment at the LHC collects billions of events each data-taking year, and processes them to make them available for physics analysis in several different formats. An even larger amount of events is in addition simulated according to physics and detector models and then reconstructed and analysed to be compared to real events. The EventIndex is a catalogue of all events in each production stage; it includes for each event a few identification parameters, some basic non-mutable information coming from the online system, and the references to the files that contain the event in each format (plus the internal pointers to the event within each file for quick retrieval). Each EventIndex record is logically simple but the system has to hold many tens of billions of records, all equally important. The Hadoop technology was selected at the start of the EventIndex project development in 2012 and proved to be robust and flexible to accommodate this kind of information; both the insertion and query response times are acceptable for the continuous and automatic operation that started in Spring 2015.

This paper describes the EventIndex data input and organisation in Hadoop and explains the operational challenges that were overcome in order to achieve the expected performance.

PACS: 29.50.+v; 29.85.Ca; 29.85.Fj

INTRODUCTION

During LHC Run 1 (2009–2013), the ATLAS experiment [1] used a catalogue of all real and simulated events in the TAGDB database. 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. The TAGDB fulfilled its primary objective to satisfy the "event picking" use case and was also useful for checking the completeness and consistency of data processing cycles, thus identifying production problems that could lead to missing or duplicated events. One limitation of the TAGDB was the lack of flexibility of the database schema, which was set prior to the start of LHC data-taking and was hard to modify afterwards to follow the needs of the collaboration. Another limitation was that the system could not easily add information about processing stages downstream of AOD production (which is the data format from which analysis datasets are derived), in fact each event is recorded several times, once for each reconstruction cycle.

In late 2012, ATLAS launched the EventIndex project to design a new system meant to be a complete catalogue of all real and simulated data, in all processing stages [2]. The new

[1]E-mail: andrea.favareto@ge.infn.it

system has to scale with the requirements of Run 2 (2015–2018). The event recording rate is expected to increase to 1 kHz, more than twice that for Run 1. The number of events expected between 2015 and 2018 is about 10^{10}. So this system has to be flexible in its schemas to accommodate a variety of quantities to be stored that could change in the future, use established and possibly open-source technologies and be "easy" to develop, deploy and operate.

THE EventIndex SYSTEM ARCHITECTURE

The main use cases that were identified and analysed in the project design phase were:

• Event picking: given a list events (uniquely identified by run number, event number, trigger stream, event format and processing version), find the events and return pointers to them to the user that issued the query, who can then use the data management tools to retrieve them.

• 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.

• 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 duplicated events).

The studies performed in 2012–2013 resulted in the high-level design of the new ATLAS EventIndex catalogue [3].

In order to bring the new system into production operation as quickly as possible, it was decided to store for each event only the information that is actually needed to satisfy the above use cases. The project was divided into work packages that could be developed almost independently from each other (after having defined the interfaces), and choosing simple and robust technological solutions for each component. The minimal information to be stored for each event consists of:

• Event identification: run number, event number, trigger stream, event format and processing version. For real data, the "luminosity block"[1] number is also stored as it can be used to link to detector or trigger condition information. For simulated data, the "simulation process number" is added to be able to uniquely identify each event.

• Trigger information: the list of trigger chains passed by the given event.

• References of the event: the GUIDs (Global Unique IDentifiers) of the logical files that contain the given event, plus the internal pointers within these files. Users can use the GUID to find the physical files containing the event by querying the ATLAS distributed data management system Rucio [4], and the internal pointers to get the actual event and process it in whichever way they need.

In the next paragraphs, the work packages that provide the functionality needed to operate each stage of the EventIndex data flow are described.

[1] A luminosity block is a short unit of time, during Run 1 normally one minute, during which all data-taking conditions and calibrations are assumed to be constant.

DATA COLLECTION

Data to be stored in the EventIndex are produced by all production jobs that run at Tier-0 or on the Grid. The EventIndex information for each permanent output file is transmitted to a central server at CERN where it is validated, reformatted and stored in the EventIndex storage system. 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 [5].

The EventIndex Producers can run in the Tier-0 cluster or on any ATLAS Grid site, as stand-alone jobs or as part of more complex workflows. Stand-alone jobs read and extract information from existing files that were previously produced, for example, all existing Run 1 data; for the data files that are currently produced, it is more convenient to run the Producer as part of the same job that creates the event files, as the last job step. In both cases, at the end of the job, EventIndex files are sent to the messaging server at CERN using the ActiveMQ [6] messaging service with the STOMP [7] protocol. ActiveMQ was chosen as the data transport system as it is supported by the CERN IT department and its performance and robustness has been measured to be sufficient for the expected data flow.

The EventIndex Consumer processes run in dedicated servers at CERN. They read the messages from the ActiveMQ broker, recombine them to reconstruct the information relative to each data file and save this information in a temporary storage space. A validation process checks that all messages have been received correctly; when all EventIndex files for all jobs of a given task have been received and validated, and the task has been completed successfully, the EventIndex information is formatted for permanent storage.

HADOOP 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 described before. This information can be stored as key-value pairs in NoSQL systems, together with the internal file navigation information. Out of these, the Hadoop eco-system provides a number of data storage options and auxiliary tools that can be used to optimize the data format in order to make the most common query types very efficient. As CERN decided at the same time to support Hadoop at the system level and provide a cluster for large applications like the EventIndex, Hadoop was naturally selected as the baseline back-end storage system.

A few data formats were tested early on, from simple CSV files in Hadoop to the extreme opposite of dumping all data into an HBase [8] table. The selected format has the data in Hadoop “Mapfile” format (“TagFiles”). The Mapfile format is a special Hadoop storage format where the keys are stored in an index file and kept in memory for fast access, and values are stored in standard sequential data files on disk. The data can be indexed in various

ways. 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.

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.

All TagFiles are also registered in the Catalog, which is implemented as an HBase table with three families:

- Descriptions: contains TagFiles characteristics, including definition of keys and fields (schema);
- Relations: contains relations to other TagFiles;
- Attributes: contains any other information that can be manipulated.

The catalogue checks the presence and consistency of all components. This is done nightly, with a creation of a back-up of the catalogue.

Files are organized in directories whose names are derived from the dataset name of the indexed file. In this way, the dataset information (or part of it) can be used to reduce the search range to the minimum, making the search more efficient. On the other hand, Hadoop is much more efficient at storing and reading large files rather than many small files; it is therefore convenient to have additional representations of the data, namely, a single TagFile for all events in a dataset, and a single TagFile for all events in a "campaign"[1]. In this way, queries can be directed by the front-end server to the data representation that is most suitable according to the query characteristics. These additional representations need of course more storage space but as long as it remains in the limits of tens of TB the trade-off between additional disk space and performance is largely positive.

QUERY SERVICES

The Query Server is a web service that acts as a front-end to the storage system. It provides a command line interface and a web interface that can be used to find and retrieve the stored information.

The searches can be then performed in two steps: first get the reference from the index file, and then get the data, using that reference. Searches can be performed using keys, with immediate results (the primary keys consist of RunNumber-EventNumber pairs), or with full searches on file-based data with MapReduce (these queries require 1–2 min for typical event collection).

As expected, the response time of different queries differs substantially for queries that are satisfied by the internal catalogue and those that need to launch a MapReduce task, and the measured times are approximately proportional to the amount of data to be searched (read in from disk) and the amount of data to be retrieved (written to the output file). Taking as reference a dataset with 123.5 million events, simple RunNumber-EventNumber searches

[1]A "campaign" in ATLAS data processing is the union of all datasets with the same format that contain events recorded with the same data conditions and processed using the same software version; for example, all proton–proton collisions taken in 2012 at a centre-of-mass energy of 8 TeV and processed in real time at the Tier-0.

need 30 s, counting all events almost 5 min, and retrieving all information more than one hour (but this is not a "normal" user query).

Access to the data is achieved by a single and simple interface for upload (copy file into HDFS and create basic structure), read (get and search), update (adding new vertical data), index (create new indices). These interfaces are implemented in Hadoop by Java classes with remote access and used through a Python client.

Users can request an accumulation of statistics (for example, the number of entries satisfying the search conditions) instead of the delivery of the full result. All results are stored within the system and can be retrieved later. Users can also specify the required output format and contents.

MONITORING SYSTEM

The monitoring system provides continuous information on the health and load of all the servers involved, as well as on the data traffic and query response times. The EventIndex monitoring system monitors the status and activities of all servers in the chain. This monitoring includes the Hadoop and ActiveMQ servers managed by CERN-IT and the virtual machines that host EventIndex services. It monitors as well the operations performed by EventIndex processes, including the input data flow, occupancy of the buffers, query rates and response times, total disk space used in Hadoop, Trigger Database statistics and consistency information.

Low-level monitoring tools were initially implemented in the CERN Service Level Status (SLS) [9] framework and subsequently ported to the new CERN framework based on Kibana [10].

DEVELOPMENT STATUS, DEPLOYMENT AND OPERATION

After almost two years of development and intensive testing, the deployment phase started in Autumn 2014. The test systems were progressively made more robust and turned into a production service. Currently (Autumn 2015) all major components exist and work satisfactorily. Run 2 data now flowing automatically and continuously from Tier-0/Grid and are available almost in real time.

- Data collection: producer transformation runs at Tier-0 and on the Grid. Consumer reads data from the ActiveMQ servers, validates them and stores to HDFS.
- Storage system: data organization in Hadoop and indexing in catalogue; trigger decoding interface.
- Query system: CLI and web interfaces. Also EventLookup service for event picking.
- Monitoring: system level monitoring in the new CERN Kibana environment.

Filling the EventIndex started in January 2015, with Run 1 data. It was decided to index first all Tier-0 productions (which give also the references to RAW data) and then the last version of reprocessed data, discarding all intermediate reconstruction versions, as they are no longer of interest. This operation needed the recall of large amounts of data from tape at CERN and all Tier-1 centers, which determined the filling rate and at the same time prevented stressing any of the hardware or software components of the EventIndex. The Tier-0 data

collection was completed in April 2015, whereas the data collection from reprocessed data at Tier-1s started on the Grid but, having lower priority than other ATLAS computing activities, is still in progress (September 2015), but almost finished. Run 1 data consist of about 6 billion events, which correspond to approximately 2 TB of EventIndex data (single version, before replication). The size after the internal replication in Hadoop is about 6 TB.

CONCLUSIONS AND OUTLOOK

The first ideas leading to the EventIndex project were discussed in ATLAS in Autumn 2012. Two and a half years later, the EventIndex exists and the deployment phase, including back-filling it with all Run 1 data, is almost completed. All fundamental building blocks perform as expected. Work is in progress to turn deployment into completely automated operations, including more automatic data validation, increased robustness against network problems and hardware failures, additional internal monitoring, and performance (timing) improvements for common queries. This work is expected to be completed during 2015; after that time the monitoring tools will be robust enough to be used by general computing shifters and the active work by experts will be reduced to occasional advice and interventions in case of problems.

REFERENCES

1. *ATLAS Collab.* The ATLAS Experiment at the CERN Large Hadron Collider // JINST. 2008. V. 3. P. S08003; doi:10.1088/1748-0221/3/08/S08003.
2. *Barberis D. et al. on behalf of the ATLAS Collab.* The Future of Event-Level Information Repositories, Indexing, and Selection in ATLAS // J. Phys. Conf. Ser. 2014. V. 513. P. 042009; doi:10.1088/1742-6596/513/4/042009.
3. *Barberis D. et al. on behalf of the ATLAS Collab.* The ATLAS EventIndex: An Event Catalogue for Experiments Collecting Large Amounts of Data // J. Phys. Conf. Ser. 2014. V. 513. P. 042002; doi:10.1088/1742-6596/513/4/042002.
4. *Garonne V. et al. on behalf of the ATLAS Collab.* Rucio — The Next Generation of Large Scale Distributed System for ATLAS Data Management // J. Phys. Conf. Ser. 2014. V. 513. P. 042021; doi:10.1088/1742-6596/513/4/042021.
5. Hadoop. http://hadoop.apache.org.
6. ActiveMQ. http://activemq.apache.org.
7. STOMP. http://stomp.github.io.
8. HBase. http://hbase.apache.org.
9. *Lopienski S.* Service Level Status — A New Real-Time Status Display for IT Services // J. Phys. Conf. Ser. 2008. V. 119. P. 052025; doi:10.1088/1742-6596/119/5/052025.
10. Kibana. https://www.elastic.co/products/kibana.