EPJ Web of Conferences **245**, 01020 (2020) https://doi.org/10.1051/epjconf/202024501020
CHEP 2019

ATLAS Operational Monitoring Data Archival and Visualization

Igor Soloviev[1,*], *Giuseppe* Avolio[2], *Andrei* Kazymov[3], and *Matei* Vasile[4]

[1]University of California, Irvine, CA 92697-4575, US
[2]European Laboratory for Particle Physics, CERN, Geneva 23, CH-1211, Switzerland
[3]Joint Institute for Nuclear Research, JINR, Dubna, Russian Federation
[4]Horia Hulubei National Institute of Physics and Nuclear Engineering, Bucharest, Romania

Abstract. The Information Service (IS) is an integral part of the Trigger and Data Acquisition (TDAQ) system of the ATLAS experiment at the Large Hadron Collider (LHC) at CERN. The IS allows online publication of operational monitoring data, and it is used by all sub-systems and sub-detectors of the experiment to constantly monitor their hardware and software components including more than 25000 applications running on more than 3000 computers. The Persistent Back-End for the ATLAS Information System (PBEAST) service stores all raw operational monitoring data for the lifetime of the experiment and provides programming and graphical interfaces to access them including Grafana dashboards and notebooks based on the CERN SWAN platform. During the ATLAS data taking sessions (for the full LHC Run 2 period) PBEAST acquired data at an average information update rate of 200 kHz and stored 20 TB of highly compacted and compressed data per year. This paper reports how over six years PBEAST became an essential piece of the experiment operations including details of the challenging requirements, the failures and successes of the various attempted implementations, the new types of monitoring data and the results of the time-series database technology evaluations for the improvements towards LHC Run 3.

1 Introduction

The ATLAS experiment [1] at the Large Hadron Collider (LHC) at CERN uses more than a hundred million electronic channels to record the data produced by the collisions. During the data taking, they are configured, controlled and monitored by more than 25000 online applications running on more than 3000 computers. Every application produces some operational monitoring data. The data are shared between online applications themselves for operation needs and are important for experts wanting to know the current state of the system. On average, about 200000 monitoring values are updated per second during data taking runs.

The ATLAS operational monitoring uses the Information Service (IS) [2] introduced more than 20 years ago by the Trigger and Data Acquisition system (TDAQ) [3]. Internally the service is based on CORBA [4] client-server architecture and the object-oriented approach. Thanks to the fast, reliable, efficient and scalable implementation of the service and the rich

*e-mail: igor.soloviev@cern.ch

object data model, it is used by all ATLAS detectors and systems for their online applications. The IS itself is not persistent, so the monitoring data were lost after the end of each data taking session. Over several years, a few tools for archival of small subsets of their specific data using different database technologies were developed, but there was not a suitable common solution across the experiment. At the beginning of LHC Run 1, it was suggested by the TDAQ Controls and Configuration group to create a common IS persistence named PBEAST (Persistent Back-End for the ATLAS information System of TDAQ). The PBEAST has to archive, aggregate and visualise raw ATLAS operational data in a generic way.

2 Cassandra Implementation

In 2011, the Controls and Configuration group evaluated the available database technologies and chose the most promising Apache Cassandra database for prototyping [5] and the CERN EOS [6] for long-term archival storage. Cassandra is a freeware distributed scalable hash table database. Three powerful servers (dual 6-core 2.67GHz Intel Xeon CPU, 4x1 TB RAID disks, 32 GB RAM) were used for the monitoring data archiving.

For one and a half years, several Cassandra deployment models and database schemes were tested using monitoring data from the, at the time, ongoing LHC Run 1. Some problems related to the chosen technology emerged quite soon. For example, the lack of vertical scaling enforced database schema redesigns and complicated data querying – the original design assigned a single row in the data column family per monitoring value. Newly acquired time-series data were added to this row. It was discovered that the Cassandra performance significantly degrades when the size of the row exceeds 80 MB independently of available hardware resources. This issue resulted in the database schema design modification and the creation of a new row after certain conditions (a time interval) as shown in Figure 1.

Figure 1. Use of time buckets in row ID to address Cassandra performance issues.

Despite the use of powerful hardware and the selection of a reduced data sample combined with data smoothing, the Cassandra implementation was never fast enough to keep up with the rate of incoming data. Furthermore, the disk space was used inefficiently, with only 20% of the volume dedicated to actual operational data and the rest reserved for Cassandra database maintenance and information replication. As a consequence, the Cassandra database was used only as a temporary store for about one month of information. Older data were moved to EOS as compressed JSON files. However, an effective way of querying them was not implemented. The implementation did not support data arrays and nested data structures nor redefinition of the data classes. Finally, there were incompatible API changes between major versions of Cassandra. The usage of the Cassandra database for PBEAST implementation was abandoned at the end of LHC Run 1.

3 Splunk Prototype

Splunk, a commercial solution by Splunk Inc [7], was another candidate for PBEAST implementation [8]. It allows non-relational structured time-series textual information to be inserted into a database, with search, analysis and Web-based reporting facilities. The queries

to check aggregation and correlation functionalities using a subset of operational monitoring data were evaluated. Splunk was considered as a possible all-in-one solution framework for archiving and visualizing monitoring data, providing easy integration, fast web applications development and low maintenance costs.

In reality, Splunk has even larger disk space overhead compared to the Cassandra database since it keeps raw text data, so it may only be used as a temporary store of information. The per-day quota licence was not flexible and was too small for real use. The visualisation facilities for scientific data were not sufficient, and their integration with third-party front-end libraries was difficult. It was decided not to use Splunk for PBEAST implementation.

4 Current Implementation

During the long shutdown between LHC Runs 1 and 2, and to minimize the risk of yet another failure by the start of next run, it was decided to implement a simple, robust solution to store operational monitoring data into files, archive them on EOS and implement an interface for data retrieval. After a short technology evaluation, the open-source Google protocol buffers library [9] was selected for efficient binary data serialization providing interoperability, compaction and compression. On top of this library, a private file format was introduced to store time-series data with random access to data metrics. The format supports all IS types, including data arrays and nested data structures and allows schema evolution. The PBEAST monitoring data receiver stores data into files organized by IS data schema and interval buckets [10].

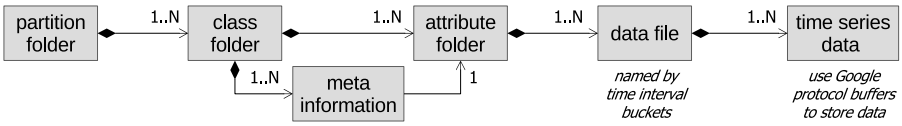

Figure 2. The PBEAST data organization.

The PBEAST service is running on a few dedicated nodes inside the ATLAS experiment area, as shown in Figure 3.

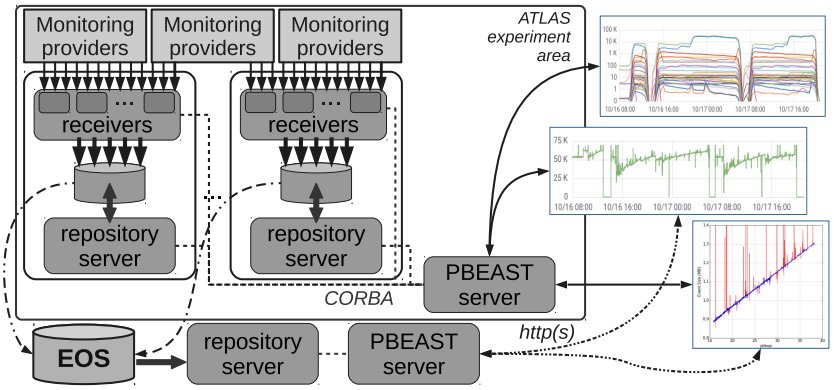

Figure 3. The PBEAST service architecture.

Every PBEAST node has its own file repository accessed by the local server and several monitoring providers. The monitoring providers get data from the IS and via their REST interface. The user requests are processed by the PBEAST server. The server redirects the user request to available nodes, collects their results and returns the merged result. The internal communication of the service is implemented over CORBA. The user communication is implemented over secure HTTP. The API supports C++, Python and Java programming languages as well as the REST functions. The use of secure HTTP allows running clients inside and outside the ATLAS experiment area. Once per week, the data from the PBEAST nodes are replicated on the EOS. As a backup solution, the PBEAST service running on the TDAQ test bed outside the ATLAS experiment area provides read-only access to the EOS repository.

For PBEAST data visualisation the Grafana [11] web toolkit is used. It is an open-source analytics platform to query databases and visualise metrics creating and exploring dashboards. To integrate the PBEAST service with Grafana, the PBEAST REST interface exposing meta information and data to Grafana client was created. From the Grafana side, a new plugin knowing details of the PBEAST data model was implemented. Gradually the PBEAST service was extended and integrated with Grafana adding many useful features and functions such as data averaging, aggregations, array functions, aliases, metrics and data filters, annotations, support for various data plots including scatter plots for correlations and several plots for string data representations. The LDAP service is used for user authentication and permissions on folders and dashboards. In total there are more than 100 PBEAST dashboards used by all ATLAS systems and detectors. A fragment of a typical PBEAST Grafana dashboard is shown in Figure 4.

Figure 4. A fragment of a typical PBEAST Grafana dashboard.

Another suggested graphical visualisation toolkit for PBEAST data is SWAN [12], a CERN service for Web-based analysis in the cloud. The Beauty library extends PBEAST for SWAN. The available Python and C++ language APIs allow any level of complexity of algorithms on top of PBEAST data. The ROOT [13] or interactive matplotlib [14] can be used for data visualisation. One can use Beauty if Grafana and PBEAST functionalities are not enough.

The PBEAST service is deployed on two server nodes installed in the last quarter of 2015. Every node has dual 12-core Intel Xeon 2.50GHz CPU, 256 GB of RAM and 8x4 TB RAID disks. It was able to sustain the data insertion rates of all ATLAS IS and network monitoring providers. The average data insertion rates from ATLAS data taking session during a typical data-taking run in 2018 was 180 kHz. The periodic data insertion spikes above 600 kHz were caused by synchronous publications of monitoring data by the high level trigger applications as shown in Figure 5 [15].

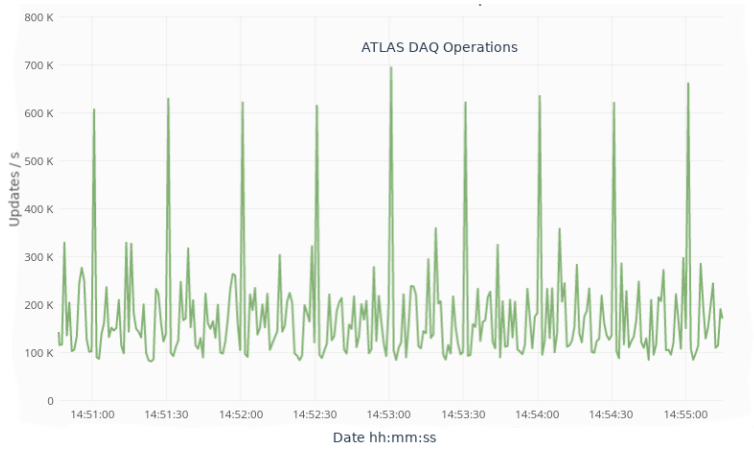

Figure 5. Rate of monitoring data updates during few minutes of a typical ATLAS data taking period [15].

The raw monitoring data are compacted and stored by PBEAST into many files, which are merged and compressed on a weekly basis. This results in an average 1.6 TB storage/month during 2018 data taking period. In Figure 6 [15], one can see the amount of compacted data stored per day in September-October 2018. If the data need to be downsampled on request, this is performed on the server and stored into the cache. All raw monitoring data will be stored for a lifetime of ATLAS. To achieve this, the PBEAST hardware will be upgraded before the start of LHC Run 3.

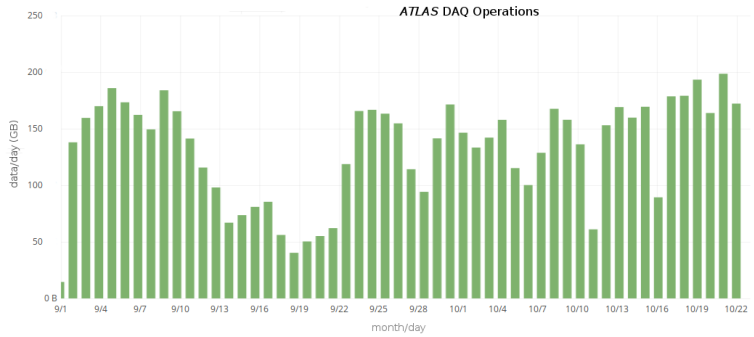

Figure 6. Rate of monitoring data updates during few minutes of typical ATLAS data taking period [15].

PBEAST is used for ATLAS operations and post-mortem analysis by experts. During the last year of LHC Run 2, there were 30 to 60 REST read requests per minute to PBEAST service mainly coming from Grafana dashboards using their auto-refresh. In some short periods, the request rate exceeded 50 requests per second.

5 Technology evaluation

The PBEAST file format is rather simple and may not be efficient for certain types of queries. The fault tolerance is only based on redundant hardware storage. Many time-series oriented

databases appeared since 2014, so it was decided to perform technology evaluation to look for possible candidates for the PBEAST service implementation or ideas on how to improve the implementation in the future.

InfluxDB [16] is an open-source time series database. It is optimized for retrieval of time-series data and easy integration with Grafana. However, the cluster solution is only available in the commercial enterprise database edition, and there is no native support for array data types. ClickHouse [17] is another candidate technology. It is developed by Yandex as an open-source column-oriented database management system for fast queries and hardware efficiency. It supports all required IS data types, including arrays, provides linear scalability, distributed reads, and free cluster solutions. However, it is not a time series database, so some time-oriented functions can be less efficient.

For both technologies, the data models allowing schema evolution and arrays support (missing in InfluxDB) have been implemented. A subset of monitoring data has been imported into both databases, and their insertion rates and storage efficiency have been compared [18] as shown in Figure 7. ClickHouse is faster for data insertion and works better with arrays. InfluxDB is optimal for simple numeric values. Both are less efficient than current PBEAST implementation, that is highly optimized for the IS data model and achieving 1900 kHz insertion rate. The evaluation is ongoing including read tests and cluster deployment.

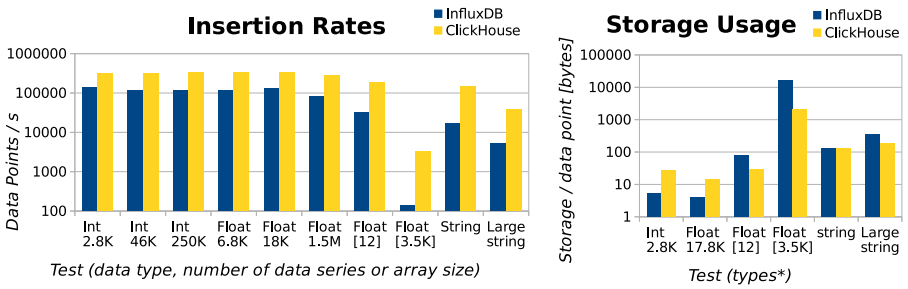

Figure 7. The comparison of InfluxDB and ClickHouse data insertion rates and storage utilisation.

6 Summary

PBEAST was successfully used during LHC Run 2 and is planned to be used in Run 3 without major software changes. It provided a stable service during the development of many required features and improvements during Run 2. While it started as a complementary monitoring tool, thanks to attractive and easy-to-use Grafana and Beauty Web tools, it became essential for ATLAS online monitoring, offline analysis of operations data, various operations reports and publications. The common monitoring technology across the whole experiment is very important for integration with systems and detectors (monitoring data are archived transparently, just configuring a dashboard is needed).

Its implementation on top of the Google protocol buffers was at least one order of magnitude more efficient in terms of CPU and disk utilisation than the best technologies available before the start of the LHC Run 2. In 2014 a single server node accepted the same monitoring data rate as a cluster of Cassandra nodes. Even the modern time-series database technologies have poorer results comparing with current PBEAST implementation. The “price” of such PBEAST service implementation was moderate from human and hardware resources points of view, and, for example, is even less expensive than the effort on Cassandra prototypes and implementations that took about three years of FTEs.

References

[1] The ATLAS Collaboration, JINST **3**, S08003 (2008)
[2] M. Barczyk et al., eConf **C0303241**, THGT003 (2003), hep-ex/0305096
[3] The ATLAS TDAQ Collaboration, JINST **11**, P06008 (2016)
[4] *Common object request broker architecture*, https://www.corba.org/
[5] A.D. Sicoe, G.L. Miotto, L. Magnoni, S. Kolos, I. Soloviev, Journal of Physics: Conference Series **368**, 012002 (2012)
[6] *EOS - open storage documentation*, http://eos-docs.web.cern.ch/
[7] *SPLUNK the data-to-everything platform*, http://splunk.com/
[8] A. Kazarov, G. Avolio, A. Chitan, M. Mineev, J. Phys. Conf. Ser. **1085**, 032052 (2018)
[9] *Google Protocol Buffers*, https://developers.google.com/protocol-buffers
[10] G. Avolio, M. D'Ascanio, G. Lehmann-Miotto, I. Soloviev, Journal of Physics: Conference Series **898**, 032010 (2017)
[11] *Grafana the open observability platform*, http://grafana.com/
[12] *Interactive data analysis in the cloud*, https://swan.web.cern.ch/
[13] R. Brun et al., Computer Physics Communications **180**, 2499 (2009)
[14] *matplotlib python plotting library*, https://matplotlib.org/
[15] ATLAS Collaboration, *Data Acquisition and High Level Trigger system Public Results (2020)*, https://twiki.cern.ch/twiki/bin/view/AtlasPublic/ApprovedPlotsDAQ
[16] *Influx open source time series database*, https://www.influxdata.com/
[17] *ClickHouse open source column-oriented DBMS*, https://clickhouse.yandex/
[18] M.E. Vasile, G. Avolio, I. Soloviev, Tech. Rep. ATL-DAQ-PROC-2019-008, CERN, Geneva (2019), https://cds.cern.ch/record/2674879