An Information Aggregation and Analytics System for ATLAS Frontier

Andrea Formica[1,3,*], *Nurcan* Ozturk[2,**], *Millissa* Si Amer[3,***], *Julio Lozano* Bahilo[4,****], *Elizabeth J* Gallas[5,†], and *Ilija* Vukotic[6,‡]

[1]Université Paris-Saclay, CEA/Saclay IRFU, 91191 Gif-sur-Yvette, France
[2]University of Texas at Arlington, Department of Physics, Arlington Texas 76019, USA
[3]Ecole Nationale Supérieure d'Informatique, Alger Oued Smar 16309, Algeria
[4]University of Valencia, Instituto de Física Corpuscular, Parque Científico, E-46980 Paterna, Spain
[5]University of Oxford, Denys Wilkinson Bldg, Keble Rd, Oxford OX1 3RH, UK
[6]University of Chicago, Enrico Fermi Institute, 933 East 56th Street, Chicago IL 60637, USA

Abstract. ATLAS event processing requires access to centralized database systems where information about calibrations, detector status and data-taking conditions are stored. This processing is done on more than 150 computing sites on a world-wide computing grid which are able to access the database using the Squid-Frontier system. Some processing workflows have been found which overload the Frontier system due to the Conditions data model currently in use, specifically because some of the Conditions data requests have been found to have a low caching efficiency. The underlying cause is that non-identical requests as far as the caching are actually retrieving a much smaller number of unique payloads. While ATLAS is undertaking an adiabatic transition during the LHC Long Shutdown 2 and Run 3 from the current COOL Conditions data model to a new data model called CREST for Run 4, it is important to identify the problematic Conditions queries with low caching efficiency and work with the detector subsystems to improve the storage of such data within the current data model. For this purpose ATLAS put together an information aggregation and analytics system. The system is based on aggregated data from the Squid-Frontier logs using the Elasticsearch technology. This paper[§] describes the components of this analytics system from the server based on Flask/Celery application to the user interface and how we use Spark SQL functionalities to filter data for making plots, storing the caching efficiency results into a Elasticsearch database and finally deploying the package via a Docker container.

ATL-SOFT-PROC-2020-026
13 March 2020

1 Introduction

In HEP experiments we use the term Conditions data to refer to non-event data representing the detector status (e.g. calibrations and alignments, data taking conditions and similar).

*e-mail: Andrea.Formica@cea.fr
**e-mail: Nurcan.Ozturk@cern.ch
***e-mail: em_si_amer@esi.dz
****e-mail: Julio.Lozano.Bahilo@cern.ch
†e-mail: Elizabeth.Gallas@physics.ox.ac.uk
‡e-mail: Ilija.Vukotic@cern.ch

These data are essential for the processing of physics data, in order to reconstruct events optimally and to exploit the full detector's potential. In the ATLAS experiment [1] at the LHC these data are stored in relational DB (Oracle), using a model based on the LCG Conditions database infrastructure and the COOL API, both developed mainly by CERN IT [2]. During data processing these Conditions data are accessed worldwide by thousands of jobs in parallel, which requires a solid architecture for their distribution and access.

2 Access to Conditions data in ATLAS

ATLAS utilizes the Frontier [3] system, a Java based application serving Conditions data via HTTP access, and allowing an optimal caching of previously requested data via Squid proxy. The caching performance is determined by the capability of the client to reproduce exactly the same SQL request for the same data loaded. We can see in Figure 1 a schema of the distributed architecture which is used by ATLAS. Each Tier 1 site (CERN, CC-IN2P3 in France, RAL in the UK and TRIUMF in Canada) in which the Frontier system is deployed has a Filebeat service running to transmit the Frontier log file lines to the Logstash service running at CERN. The Logstash process extracts part of the logging information and sends these data to the Elasticsearch [4] platform hosted by the University of Chicago, where the information is then stored in well organized indexes for further analysis [5]. In this project we are focusing on the analysis of the caching performance of this system, via the usage of the data collected in the Chicago monitoring infrastructure. Additional metadata are needed to fully understand the requests from the client jobs; these metadata are available directly from the Conditions database (COOL) deployed at CERN, and can be retrieved only via a dedicated API (COOLR [6]).

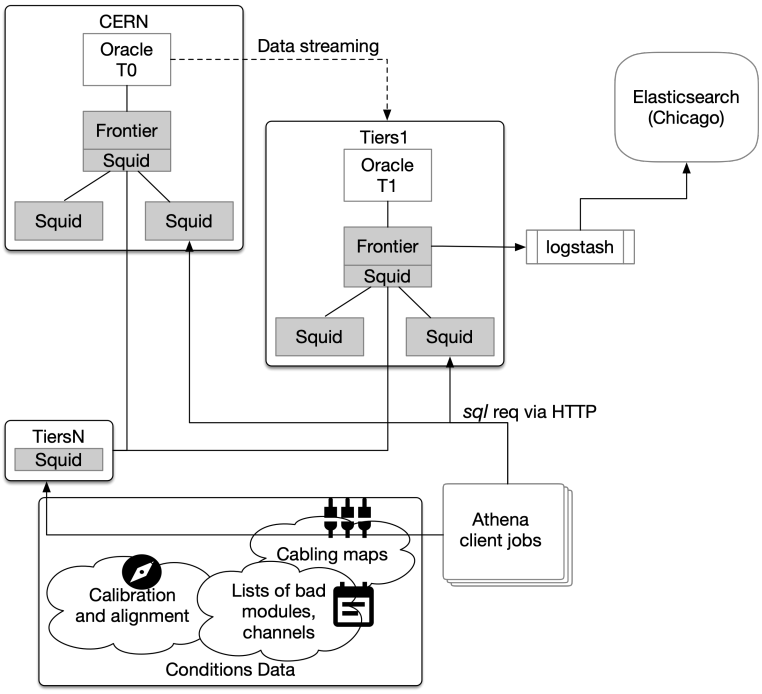

Figure 1. ATLAS distributed architecture for the Conditions data access.

3 Frontier analytics project

The Squid-Frontier system suffered from service degradation leading to failures in particular workflows; the overlay of the underlying event from real data onto the simulated data and other specialised reprocessing. Requests from these workflows were much less likely to be found in the cache (i.e. low caching efficiency). Using Frontier logs we could extract the SQL requests and re-play them on a separate Frontier instance or via COOLR service. The exercise allowed us to determine that several requests were accessing the same payload data but using slightly different SQL requests (essentially the time range selection was different between different jobs). This behaviour indicates that the client is performing an access which may lead to an overload of the caching system because the same data are requested from Oracle (via Frontier) in different ways. The identification of such problematic request patterns is essential to improve the system for Run 3. In order to quickly study problematic workflows a dedicated analytics application was set up and deployed. The application allows extraction of logging information specific to a given *task* (a set of jobs with some defined data input and analysis configuration) and renders a visualization of the queries performed by the Frontier server for each type of Conditions data that was requested by the jobs. Using COOLR services the application allows as well to compare the payload retrieved for different SQL requests, and to compute the caching efficiency on the given payload type.

3.1 Architecture

The application consists of a set of services. The main entry point is a REST API implemented in Python using a *Flask* [7] web server, allowing a given task to be selected and to submit a request for the retrieval of monitoring data from the Elasticsearch index. The request is executed asynchronously by a *Celery* [8] service. Communication between the *Flask* web server and *Celery* is performed via *Redis* [8]. Once the data are loaded an additional step is performed to merge the collected SQL queries with metadata coming directly from the COOL database at CERN. This step is important to better understand which type of data was requested in each SQL query performed by Frontier. A web application developed in JavaScript Bootstrap allows then to visualize summary results on the queries associated to that task. A detailed schema of the application can be seen in Figure 2.

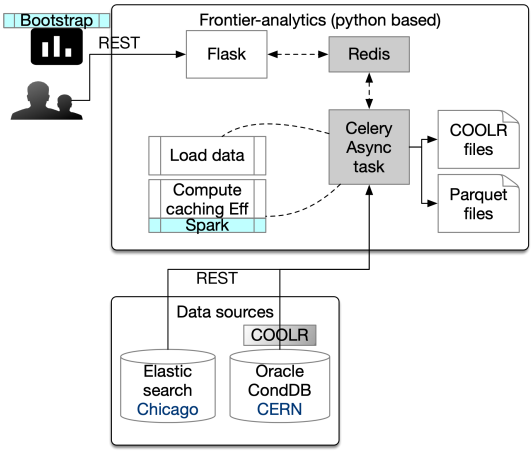

Figure 2. ATLAS Frontier Analytics architecture.

3.2 Deployment

The whole application code is stored in a *gitlab* repository at CERN. A continuous integration job allows the *Flask* and *Celery* based services to be packaged inside two different *Docker* images, which are then used for the deployment. It has been decided that because of the tight interactions with the Elasticsearch service, the best deployment location is the University of Chicago. Here a cloud infrastructure is available, and the *Kubernetes* [9] orchestrator can be used to manage clusters of machines. The *Docker* images have been deployed at the University of Chicago for production usage. The application is running on https://frontier.uc.ssl-hep.org. A similar infrastructure was set up at CERN in a small private *Kubernetes* cluster, for integration purposes. A summary sketch of the deployment cycle is illustrated in Figure 3.

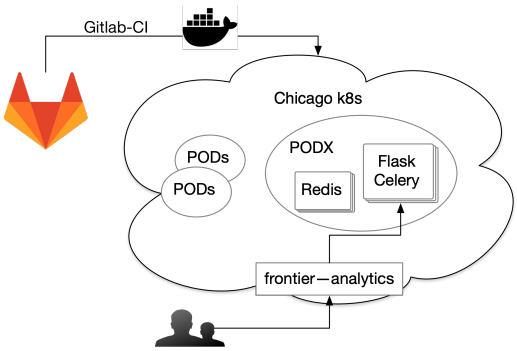

Figure 3. ATLAS Frontier Analytics deployment.

3.3 Results

The user interface of the service allows the input data to be filtered and the Parquet files [10] to be prepared based on the task IDs that are selected from those problematic workflows. The input data is stored in the Elasticsearch database by extracting the relevant information from the Frontier server log files via the Filebeat and Logstash services [5]. The Parquet is a columnar storage file format on which we run Spark [11] SQL functionalities to filter data for making plots. Then we visualize the data in the form of plots in four categories:

1. The count of cached and not-cached queries per database instances, the type of the not-cached queries (first time queries vs. disconnected queries), the count of queries per COOL data schema, the percentage of cached vs. not-cached queries per COOL data schema and the percentage of the queries per node for a given COOL data schema.

2. The time distribution of the queries (wall time and query time).

3. The response size (payload data) of the queries, the percentage of the response size per COOL data schema and the percentage of the response size per node for a given COOL data schema.

4. The count of queries with high processing time (e.g. more than 1 s.) per COOL data schema and per node for a given schema.

As an example two such plots can be seen in Figures 4 and 5 for a proton-lead overlay task. The plot in Figure 4 shows the percentage of the queries (cached vs. not-cached) per COOL

data schema and the plot in Figure 5 shows the count of queries with high processing time per node for the ONL_SCT schema which has a high rate of the not-cached queries as seen from the plot in Figure 4. The ONL_SCT is a schema from the online system filled in by the Semiconductor Tracker (SCT) which is a part of the ATLAS inner detector. The node /SCT/DAQ/Config/Chip has the most queries with the high processing time among all other nodes.

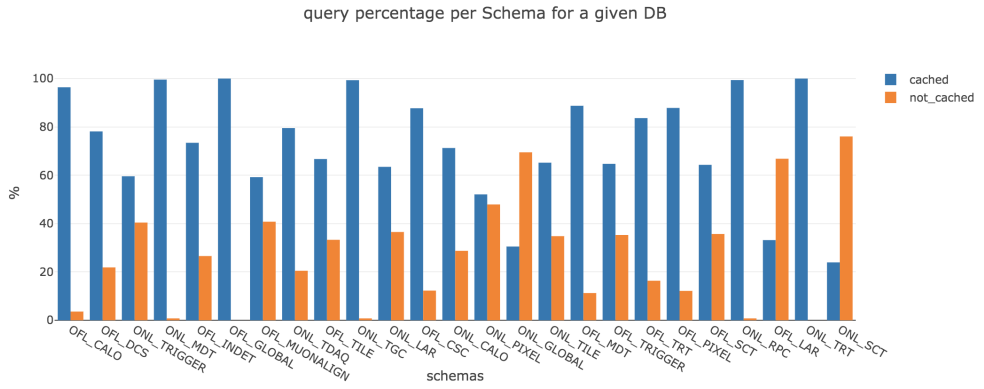

Figure 4. Visualization of data on https://frontier.uc.ssl-hep.org. The percentage of the queries (cached vs. not-cached) per COOL data schema.

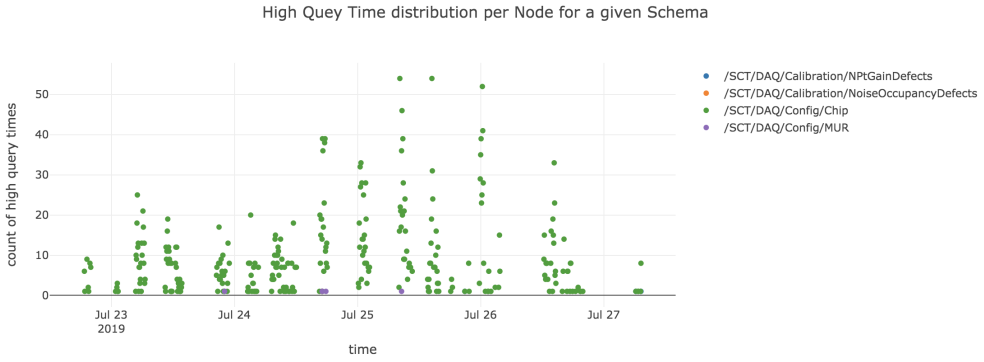

Figure 5. Visualization of data on https://frontier.uc.ssl-hep.org. The count of queries with high processing time per node for the ONL_SCT schema.

The user interface also allows the calculation of the caching efficiencies for the nodes with queries with the high processing time. As an example Table 1 shows the results of the caching efficiencies as calculated for the selected ONL_SCT and ONL_TDAQ (Trigger and Data Acquisition) nodes. The number of the total queries asking for the payload from the list of the folders are given in the second column. The number of the different queries are the queries with a different range in time e.g they correspond to different SQL/URLs to the Squid-Frontier system. The number of the payloads show the number of the different (unique) Conditions data retrieved by those queries. The payload size is the size of the Conditions data for one payload. These results show that we do not have a good caching efficiency for the

Table 1. Summary table for the caching efficiency results for some folders in the ONL_SCT and ONL_TDAQ schemas. The Payloads column represents the number of unique payloads retrieved by the queries.

Folder	# Queries	# Diff Queries	# Payloads	Size (KB)
/SCT/DAQ/Config/Chips	3437	2885	2	24736
/SCT/DAQ/Config/MUR	5635	4067	1	1027
/SCT/DAQ/Config/Module	2392	2063	1	907
/SCT/DAQ/Config/ROD	834	804	1	35
/TDAQ/OLC/CALIBRATIONS	168779	19805	7	22

folders in the ONL_SCT and ONL_TDAQ schemas as we make redundant queries to retrieve the same payload.

4 Conclusions

An application has been developed to analyse the problematic Conditions data access patterns in several workflows in ATLAS. This application benefits from the existing monitoring infrastructure at the sites where the Squid-Frontier system is set up, thanks to all the experts at these sites; CERN, CC_IN2P3 Lyon in France, TRIUMF in Canada, RAL in the UK, and the University of Chicago in the USA. The application will be used by the experts in order to improve the Conditions access stability for the current operations through the LHC Long Shutdown 2 as well as to design a more cache-friendly system for Run 3. In particular the identification of COOL folders for which the impact of the caching efficiency is most significantly affecting the Squid-Frontier system can be addressed by improving the client access with better parameters for the default interval length used in the queries. The CREST [12] system under development for future Conditions access in the horizon of Run 4 provides an optimal caching of the payloads by design.

References

[1] ATLAS Collaboration, JINST **3**, S08003 (2008)

[2] A. Valassi, R. Basset, M. Clemencic, G. Pucciani, S.A. Schmidt, M. Wache, *COOL, LCG conditions database for the LHC experiments: Development and deployment status*, in *IEEE Nuclear Science Symposium Conference Record, 2008. NSS '08* (2008), pp. 3021–3028

[3] D. Dykstra, J. Phys.: Conf. Ser. **331**, 042008 (2011)

[4] *Elasticsearch*, `https://www.elastic.co/`

[5] M. Svatos, A. De Salvo, A. Dewhurst, E. Vamvakopoulos, J. Lozano Bahilo, N. Ozturk, J. Sanchez, D. Dykstra, EPJ Web Conf. **214**, 03020 (2019)

[6] A. Formica, E. Gallas, Journal of Physics: Conference Series **664**, 042016 (2015)

[7] *Flask*, `http://flask.palletsprojects.com/`

[8] *Celery*, `http://www.celeryproject.org/`

[9] *Kubernetes*, `https://kubernetes.io/`

[10] *Parquet*, `http://parquet.apache.org/`

[11] *Spark*, `https://spark.apache.org/`

[12] R. Sipos, A. Formica, G. Franzoni, G. Govi, A. Pfeiffer, Journal of Physics: Conference Series **898**, 042047 (2017)