INDEXING OF ATLAS DATA MANAGEMENT AND ANALYSIS SYSTEM METADATA

M.A. Grigoryeva[1,2,a], M.V. Golosova[1], A.A. Klimentov[3], M.S. Borodin[4], A.A. Alekseev[2], I.A. Tkachenko[1]

[1] *National Research Center "Kurchatov Institute", 1, Akademika Kurchatova pl., Moscow, 123098, Russia*

[2] *Tomsk Polytechnic University, Office 420, 4a Usov Street, Building 19*

[3] *Brookhaven National Laboratory, Brookhaven National Laboratory, P.O. Box 5000, Upton, NY 11973-5000*

[4] *The University of Iowa, Iowa City, IA 52242, USA*

E-mail: [a] maria.grigorieva@cern.ch

This manuscript is dedicated to the development of the system to manage metainformation of ATLAS experiment. The main purpose of the system is to provide scientists with transparent access to the actual and historical metadata related to data analysis, processing and modeling. The system design addresses the following goals: providing a flexible and fast search for metadata on various combinations of keywords, generating aggregated reports, categorized according to selected parameters, such as the studied physical process, scientific topic, physical group, etc. The article presents the architecture of the developed indexing and search system, as well as the results of performance tests. The comparison of the query execution speed within the developed system and in case of querying the original relational databases showed that the developed system provides results faster. Also the new system allows much more complex search requests than the original storages.

Keywords: ElasticSearch, indexing system, HENP, megascience

1. Introduction

Experiments in the field of High Energy and Particle Physics (HENP) and scientific collaborations working on modern accelerators produce hundreds of petabytes of scientific data and terabytes of related metadata. Storage systems contain "raw" (unprocessed) data obtained directly from the experimental setup, and processed data, used for physical analysis. The stages of data processing and analysis are often implemented in a processing system as a set of tasks, each consisting of a set of jobs. The scale can be described by the following numbers: $O(10^3)$ scientists perform $O(10^6)$ tasks per year and $O(10^6)$ jobs per day. The processing stages and the course of their execution are recorded within the data management and processing systems. During the execution of various data processing chains, the following questions inevitably arise: How to start the task? Where will it be executed? How to monitor its progress? How to control task's execution? How to recover failed tasks? How to associate task results with results produced earlier? To solve these problems, specialized systems are being developed, which will be discussed below.

2. HENP Data Management and Workflow Management Systems.

In the ATLAS experiment [1], for recording all data (raw and reduced), and processing and physical analysis tasks, the workload management systems – Production System 2 (PS2) [2] (using PanDA [3] as the main engine and backend), ATLAS Metadata Interface [4], Rucio Distributed Data Management system [5] were developed. BigPanDA Monitoring [6] allows to monitor tasks and jobs execution process. All systems listed above are using Oracle database [7] to store metainformation.

In this paper metainformation related to the global processing of the above-mentioned experimental data is considered. It is mainly stored by subsystems of PS2. Its architecture has three main levels of abstraction:

- DEFT (Database Engine For Tasks) is a top-level subsystem. DEFT accepts and processes requests for the execution of tasks and is responsible for the formation of processing steps, tasks, input data and parameters;
- JEDI (Job Execution and Definition Interface) is an intermediate-level subsystem that uses job descriptions prepared by DEFT. JEDI dynamically determines the number of jobs for each task and is responsible for running and executing individual jobs;
- PanDA – the main "engine" of the system, the subsystem storing metainformation of production and analysis tasks. PanDA determines which resources and at what time each task will utilize, receives information from pilot tasks and information system, manages the flow of tasks.

Routinely PS2 processes up to 2M tasks per year (peaking at 350K/month, figure 1). All of them are registered in DEFT's database.

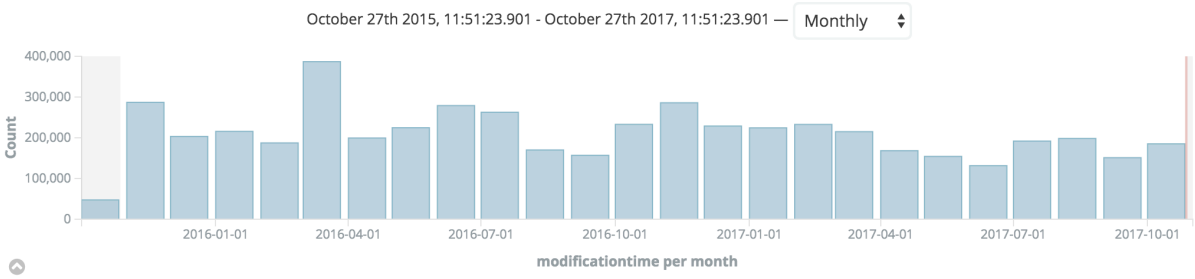

Figure 1 - Monthly distribution of the number of tasks

To date, the total quantity of registered tasks is about 8 million. 1.2 M of them are the tasks that were launched by physical groups on certain scientific topics – production tasks. The rest are the tasks of individual users – user tasks.

All meta-information stored in DEFT can be divided into the following categories:

1. task metadata: task ID, title, status; run, start, status and completion timings; execution priority; name of the user started the task;
2. information on input and output datasets (a dataset is a set of files with data taken under same conditions);
3. metainformation about requests for tasks that come from users, from user groups or from a virtual organizations (experiments);
4. information about the state of the detectors, versions of calibration data, versions of physics analysis software;
5. the number of events necessary for simulation and / or processing, as well as the number of already simulated / processed events;
6. metainformation describing the request for processing / modeling of physical processes: the name of the physics group that launched the task, the names of the Monte Carlo simulation campaigns or the campaign for obtaining real data from the detector, the names of the projects (on-purpose data processing and analysis) for modeling and data processing;
7. keywords (or hashtags) that define the physical process(es) studied within the tasks;
8. predicted completion time of tasks obtained via machine learning methods.

The listed metadata are used by the BigPanDA monitoring system to track data processing and analysis tasks, executed by PS2. Monitoring and management are carried out using various tools, which use tables, graphs and diagrams [6, 11]. This system is used by the participants of the experiment to provide support and control of data processing and analysis processes, as well as engineers responsible for the operation of computing capacities in more than 100 computing sites around the world. In addition to engineering personnel, among users of ProdSys2 there are also research physicists who need to receive not only the reports on the task completion or error reports, but the information on the progress of the task chains execution and about data ready for scientific analysis, and do this for tasks launched within the framework of physical process under investigation, or the problem being investigated. For example, "to get all the tasks launched in 2016 in the campaign MC16a in the search for the Higgs boson at a center of mass energy of 13 TeV and for the period of the LHC operation in 2016, and software version 12.1.4." The execution time of such a query to the existing database in PS2 is rather large, due to the fact that it requires a large number of table JOINs. And the more physical parameters are requested, the more complicated the query becomes, because the database is arranged in such a way that most physical attributes are "hidden" in unstructured JSON strings stored in CLOBs. Query completion time significantly exceeds the interval that is considered to be comfortable for the interactive/real time monitoring/metadata lookup purposes.

This requires a revision of the concept of data access and storage. Relational databases have a concept of materialized views, which may include compounds and / or composite values (aggregates). They allow to reduce the execution time of queries by precalculating the expensive JOINs and aggregating operations. However, their use can significantly increase the volumes of the source database. The Oracle database, which is used as PS2 backend, has a paid license, with limitations on the volumes of data stored. Therefore, it was decided to use free-of-charge data storage and indexing systems to solve the task. In this work, a full-text search engine ElasticSearch, was chosen for this purpose, and on its base the system for indexing metadata from the PS2 repository was implemented.

ElasticSearch (ES) is the fulltext search and powerful analytics engine built on top of the widely-known Lucene library [8]. The main advantages of this system are flexibility, ease of use, speed, scalability (the ability of the distribution to multiple servers), indexing in real time and the ability to transfer the system to the cloud. Adding information to the index and index search are done using simple HTTP-based REST API requests. ES performs basic computations with indexable numerical data, such as summation, counting, calculating the median, arithmetic mean, standard deviation, but essentially is a search engine.

3. Indexing System Architecture for HENP

The architecture of the metadata indexing system for ProdSys2 consists of the following components:

1. initial metadata sources:
    a. DEFT

   b. JEDI
2. ETL (Extract-Transform-Load) subsystem, which provides automation of the processes of export, transformation and loading of data:
   a. data collection and preparation modules, which include retrieving metadata from a relational database, converting metadata to JSON (oracle2JSON.py), and loading data into the index store.
   b. daily synchronization mechanism, providing loading of the new and overwriting of the existing records in the ElasticSearch storage.
3. subsystem for data storage and indexing (ElasticSearch).

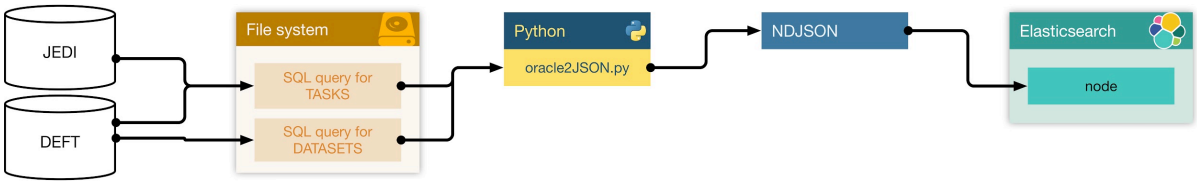

Figure 2 – Architecture of Indexing System

The SQL query for sampling metadata from initial sources takes about 350 lines, and takes on average for 78 seconds to execute all production tasks in one day. The workflow is launched for execution in the following mode: metadata retrieval from Oracle (chunked by days), starting from the initial point to the present, query results conversion to the NDJSON format (that is understood by ES), and population of the ES index store. The process of ingesting all existing production tasks data into ES took about 40 hours on a single HP SL390s G7 (2x Intel Xeon E5620, 48 GB RAM, HP/Seagate SATA 7200 RPM HDD) node also running the ES itself.
Tasks and datasets have hierarchical relations, but in the ElasticSearch storage they are currently stored in the form of flat documents, without arrays, nested objects and parent-child relationships.

As the result, we had developed a system that allows data extraction from relational databases in background mode, convert it and load into the index storage. ES stores the actual data synchronized with the initial source databases. And these data are stored in a form suitable for searching and building aggregations for various combinations of physical parameters. Thus, the developed system can be used by research physicists to flexibly search for information on scientific topics.

4. Performed benchmarks and their results

Benchmark layout was the following: for Oracle, CERN ADCR Active Data Guard database was used, queries were performed from CERN AI node with network RTT to ADCR/ADG [9] database of 1 ms. For ElasticSearch, queries were performed via HTTP from the ES node itself. Requests were done with Python code (v 2.6.6, RHEL 6 builds) code, cx_Oracle/py-cURL as access libraries and datetime.now() as the time source. Timed parts were query executions and result fetching for both Oracle and ES, initial connection to Oracle was not timed, as this was meant to emulate persistent sessions within Web frameworks. ADCR/ADG database is also used for various other tasks, so we did several measurements during different day/week periods (100 takes in a row for each). Thus, Oracle results are of an indicative type, their statistics is non-gaussian (even for a single series), since we can't account for the other activity. Therefore, we had also included minimal query timings for Oracle as the lower bound of query time one can achieve in this real-world setup. On the other hand, ES instance was dedicated for these queries, so we got the proper, Gaussian-type statistics with standard deviation of < 4.5% of the measured value.
The indexing system was tested on two query types:

- event summary report (ESR): it belongs to the meta-data categorization tasks and shows counts of requested and processed events during Monte-Carlo production activity. The particular ESR query was constructed for MC16a sub-campaign of MC16 campaign.
- keyword search (KS): it belongs to the mining-type queries, when one seeks for a set of tasks employing certain combinations of physical parameters, software versions, etc. Two queries were constructed: "MC16a_CP, Atlas-21.0.1, MCGN, mc16_13TeV, ATLAS-R2-2016-01-00-01_VALIDATION, OFLCOND-MC16-SDR-14" ("KS-1", the heavier one, requires 3 JOINs with the current Oracle DB schema) and "MC16a, MCGN, mc16_13TeV" ("KS-2", simpler, no JOINs for Oracle).

Additionally, three ESR query variants for Oracle (referenced as "ESR-ORIG", "ESR" and "ESR-LIKE-ONLY" below) were tested: progressively ranging from straightforward ("ESR-ORIG") query, which could

have been constructed by automated machinery of certain Web frameworks (e.g., Django), to a rather optimized query, which takes into account the specifics of Oracle functions and internal machinery. ESR query was chosen for such tests, since it is an inherently complex one for the given DB schema, so it partly answers the question "Can we use the existing Oracle schema, but optimize queries and keep the things simple?". The results of the performed benchmarks are presented in Tables 1 and 2. In these tables only 3 query measurements are provided as the most averaged.

Table 1 - Results for the latter queries (3 types, Oracle-only):

Attempt	ESR-ORIG, mean/min	ESR, mean/min	ESR-LIKE-ONLY, mean/min
№1	22/7 s	21/9 s	12/4 s
№2	25/8 s	15/6 s	7/4 s
№3	25/7 s	27/7 s	24/5 s

Table 2 - Oracle/ES query comparisons (take numbers designate the same attempts across both tables, ESR timings are the best ones, from "ESR-LIKE-ONLY" query, color codes same query types):

Attempt	ESR, Oracle, mean/min	ESR, ES	KS-1, Oracle	KS-1, ES	KS-2, Oracle	KS-2, ES
№1	12/4 s	0.002 s	2.5 s	0.10 s	0.08 s	0.10 s
№2	7/4 s	0.002 s	2.6 s	0.10 s	0.08 s	0.10 s
№3	24/5 s	0.002 s	2.5 s	0.10 s	0.07 s	0.10 s

5. Discussion of obtained metrics

As we can see, the idea of Oracle ESR query optimization yields at least some benefits: we can cut half of the time. Still, the obtained 4 s (best case) and ~12 s (median) are not lying within the current "area of seamlesness" (~1 s, [10]) for interactive analytics-like Web interfaces. And results for ElasticSearch-based system bring consistent "lightning fast" response time for ESR, so the designed system does better than the current schema in Oracle.

Keyword search for ElasticSearch gives stable 0.10 s result delivery; Oracle gives even less (0.07 – 0.08 s) for the no-JOIN query, but increases to 2.5 s for complex (schema-wise) queries.

6. Conclusion and future work

ElasticSearch- and ETL-based system was implemented to match the specific task of indexing and querying the stored metadata. Two popular query types were implemented and benchmarked on metadata from ~1.2 M of ATLAS production jobs: the results show consistent query timings, which is lower than the ones for the existing RDBMS schema.

The system is now evaluated by the ATLAS collaboration, mainly to understand if its design and abilities match the expectations of the end users. In parallel, we are conducting the scalability tests, working on the Web interface for the system and trying to further explore the limits of the chosen approach, architecture and technologies by extending the functionality and integrating other parts of the metadata and implementing more queries.

5. Acknowledgements

The work was supported by the Russian Ministry of Science and Education under contract No.14.Z50.31.0024 and by the Russian Science Foundation under contract No.16-11-10280.

This work has been carried out using computing resources of the federal collective usage center Complex for Simulation and Data Processing for Mega-science Facilities at NRC "Kurchatov Institute", http://ckp.nrcki.ru/.

References

[1] G Aad et al. The ATLAS Experiment at the CERN Large Hadron Collider // 2008 JINST 3 S08003

[2] M Borodin et al 2015 J. Phys.: Conf. Ser. 664 062005

[3] T Maeno et al 2011 J. Phys.: Conf. Ser. 331 072024

[4] Solveig Albrand et al 2010 J. Phys.: Conf. Ser. 219 042030

[5] V Garonne et al 2014 J. Phys.: Conf. Ser. 513 042021

[6] J Schovancová, K De, A Klimentov, P Love, M Potekhin, T Wenaus on behalf of the ATLAS Collaboration. The next generation of ATLAS PanDA Monitoring // PoS ISGC2014 (2014) 035

[7] E Grancher 2010 J. Phys.: Conf. Ser. 219 052004

[8] Ultra-fast Search Library and Server // http://lucene.apache.org/

[9] G Dimitrov et al 2014 J. Phys.: Conf. Ser. 513 042012

[10] Nielsen Norman Group, Website Response Times, June 2010, https://www.nngroup.com/articles/website-response-times/

[11] J Andreeva et al, Experiment Dashboard - a generic, scalable solution for monitoring of the LHC computing activities, distributed sites and services // 2012, J. Phys.: Conf. Ser., 396, 032093