Big data solutions for CMS computing monitoring and analytics

Christian Ariza-Porras[1], *Valentin* Kuznetsov[2], and *Federica* Legger[3,*]

[1]Universidad de los Andes, Colombia
[2]Cornell University, Ithaca NY, 14850 USA
[3]Istituto Nazionale di Fisica Nucleare, via Pietro Giuria 1, 10125 Torino, Italy

Abstract. The CMS computing infrastructure is composed of several subsystems that accomplish complex tasks such as workload and data management, transfers, submission of user and centrally managed production requests. Till recently, most subsystems were monitored through custom tools and web applications, and logging information was scattered over several sources and typically accessible only by experts. In the last year, CMS computing fostered the adoption of common big data solutions based on open-source, scalable, and no-SQL tools, such as Hadoop, InfluxDB, and ElasticSearch, available through the CERN IT infrastructure. Such systems allow for the easy deployment of monitoring and accounting applications using visualisation tools such as Kibana and Grafana. Alarms can be raised when anomalous conditions in the monitoring data are met, and the relevant teams are automatically notified. Data sources from different subsystems are used to build complex workflows and predictive analytics (such as data popularity, smart caching, transfer latency), and for performance studies. We describe the full software architecture and data flow, the CMS computing data sources and monitoring applications, and show how the stored data can be used to gain insights into the various subsystems by exploiting scalable solutions based on Spark.

1 Introduction

The CMS experiment [1] at the Large Hadron Collider (LHC) exploits a tiered distributed computing infrastructure to process LHC data and produce Monte Carlo simulated events of relevant physics processes. Data are stored and processed in more than 100 computing centers worldwide, connected through a set of grid services responsible for storage, computing, and connectivity. A recent overview of the computing model of the LHC experiments can be found in [2].

The CMS distributed computing infrastructure includes central services and middleware that handle authentication, workload and data management. The workload management system executes payloads in compute nodes provisioned through GlideinWMS [3] and thus made available as execution slots in a Vanilla Universe HTCondor [4]. HTCondor jobs are submitted via specific workload management tools: WMAgent for central data processing

*e-mail: federica.legger@cern.ch

and Monte Carlo production jobs, and CRAB for user jobs [5]. The data management system is modular. The main components are PhEDEx, the data transfer and location system; the Data Bookkeeping Service (DBS), a metadata catalog; and the Data Aggregation Service (DAS), designed to aggregate views and provide them to users and services [6]. Data from these services are available to CMS collaborators through a web suite of services known as CMSWEB.

Over the last decade, the various computing services were monitored through custom tools and web applications, partly developed within the CMS computing community, and partly by the CERN IT department (monitoring the execution of jobs, data transfers, and site availability). As a result, the monitoring data were scattered over several sources, and, mostly, accessible only to experts. The maintenance and operation of the monitoring applications were therefore becoming increasingly time-consuming and complicated due to the high turnover of experts in the computing community. Nowadays, several solutions are available on the market to gather, store and process large amounts of data such as those produced by monitoring and logging services of computing applications. Many technologies are available under an open-source licence, and are developed and supported by large software companies and communities. The CMS computing community decided, therefore, to gradually abandon in-house solutions, in favour of the adoption of widely-used technologies such as Collectd, Kafka, Spark, Elasticsearch, InfluxDB, Grafana, and others. In the following sections we describe MONIT, the monitoring infrastructure at CERN which is based on the open-source technologies listed above, the organisation of CMS monitoring applications based on MONIT, and future developments.

2 The monitoring infrastructure at CERN

The monitoring infrastructure at CERN [7], known as MONIT, is based on the following architecture (see also Fig. 1):

- Data providers: the computing systems and services providing the monitoring data. In the case of CMS computing, these can be related to the execution of HTCondor jobs, or performance metrics of various subsystems such as CRAB, WMA, Submission Infrastructure, PhEDEx. The data providers are responsible for providing the data in JSON (JavaScript Object Notation) format.
- Data injection: monitoring data are pushed into MONIT through HTTP, for data providers inside the CERN network, or through ActiveMQ [8] messaging. CMS data providers are mostly using ActiveMQ. For the injection of logs, Logstash [9] is used.
- Data transport and processing: monitoring data are routed to an Apache Kafka cluster [10], whereas Apache Spark [11] and Apache Flume [25] are used for data processing. At this stage the data may be enriched with additional information or data from several sources can be aggregated as needed.
- Data sources: data can be stored in MONIT using three different technologies, to be chosen according to the retention policy and on the amount of information to be stored. Raw data needed for short term monitoring are stored in ElasticSearch [12]. The time retention policy varies according to the data source from 30 to 40 days. Raw data needed for a longer time period are stored in Apache HDFS [13]. Aggregated data in the form of time series can be stored for 5 years in InfluxDB [14]. In addition, performance metrics of the CMSWEB cluster are stored in Prometheus [15]. Prometheus is not part of the MONIT infrastructure, and it is managed by CMS.

- Data access: depending on the storage technology, several possibilities are available. Data in ElasticSearch can be visualised using Kibana [16]. Grafana [17] is a visualization tool that supports several data sources (ElasticSearch, InfluxDB, and Prometheus). In addition, Grafana can be used to set up simple threshold-based alerts. For analytics purposes data are accessible through a Grafana proxy. Data on HDFS can be processed using Apache Spark, for example through the SWAN service [18] at CERN.

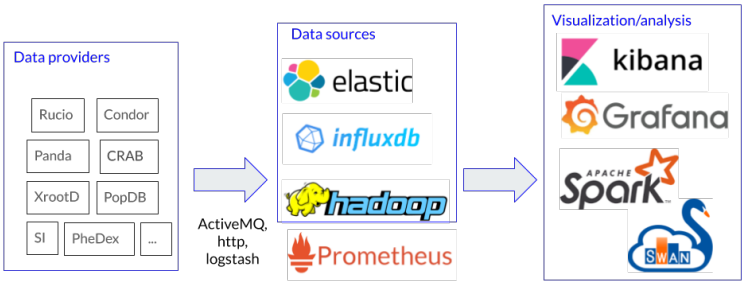

Figure 1. The architectural diagram of the MONIT infrastructure at CERN, as used by the CMS monitoring applications.

3 Organisation of CMS monitoring

The main entry point to the CMS monitoring portal is through Grafana (see Fig. 2). The portal provides access to:

- Monitoring dashboards from various CMS subsystems, organised by topics (for example HTCondor jobs, CRAB, WMA, Submission Infrastructure, CMSWEB). These are further divided according to the read/write access policy. *Production* dashboards are dashboards in production state, and should be modified by group members only if needed. *Development dashboards* are under development by the various groups, and candidates to become production dashboards at the end of the development cycle. *Playground* dashboards are open for everyone to experiment and learn Grafana.
- Data popularity plots produced on a regular basis, to show access patterns of CMS computing applications.
- The status of the various alerts implemented in Grafana.
- Documentation on CMS data sources and the code used to inject the data into MONIT.
- Various tutorials on data injection into MONIT, data visualisation with Grafana or Kibana, example Jupyter notebooks showing how to access data in HDFS.

In addition, the CMS monitoring team provides a common set of tools [20] to ease the transition of CMS monitoring applications to the MONIT infrastructure: a common interface to push data into MONIT using the Stomp protocol for ActiveMQ, a common tool to fetch data from MONIT using a Grafana proxy, and a common set of libraries to execute Spark jobs on the Hadoop clusters at CERN via CMSSpark framework [21].

4 Migration to the MONIT infrastructure

During the last year, several CMS monitoring applications have been ported to use the MONIT infrastructure, most notably the historical monitoring of HTCondor jobs, the task

Figure 2. Screenshot of the CMS monitoring portal based on Grafana.

monitoring dashboards for user jobs, and the production of data popularity plots. The historical monitoring of HTCondor jobs is based on a limited set of job metrics (such as job type, the site the job was executed at, job status, software version), aggregated over different time periods (twelve minutes, one hour, one day, one week). Several visualizations showing the number of jobs in execution, completed, or waiting to be executed are available to allow for real time monitoring of the performances of the CMS distributed computing infrastructure (see Fig. 3). Since InfluxDB performances degrade at high cardinality, effort was put into selecting only the most relevant metrics and limiting the range of their possible values. We currently store about twenty metrics, a few with a cardinality of up to 100 values. In total, the current series cardinality is almost eight million.

The task monitoring dashboard is designed to provide access to information about the user tasks and jobs, their status, and links to job logs. Due to the number of metrics to be displayed and the limited history needed (user tasks may need a few days to complete, but rarely exceed one week of duration), data from ElasticSearch is used. Two views are available to CMS users: an overview of all their tasks, and a detailed view showing detailed metrics for each task.

Data popularity plots are produced on a monthly basis by executing Spark jobs on data stored in HDFS, since access patterns of interest are typically studied over several months or years. Several visualisations are available to show the number of processed events or accesses for various input data and access types.

Currently there are more than 50 Grafana production dashboards, organised in more than 10 different topics, and another 50 dashboards in Kibana. The Grafana dashboards are used to monitor the health of several computing services, notably the CMSWEB services, and the built-in Grafana alert system is being used to notify the system administrators in case of failures. CMS monitoring data amount to 30 TB in HDFS, mostly used to store information about HTCondor jobs and data transfers.

5 Current developments

We are currently migrating the monitoring applications showing the status and availability of the various CMS sites, also known as Site Status Board (SSB), to the MONIT infrastructure.

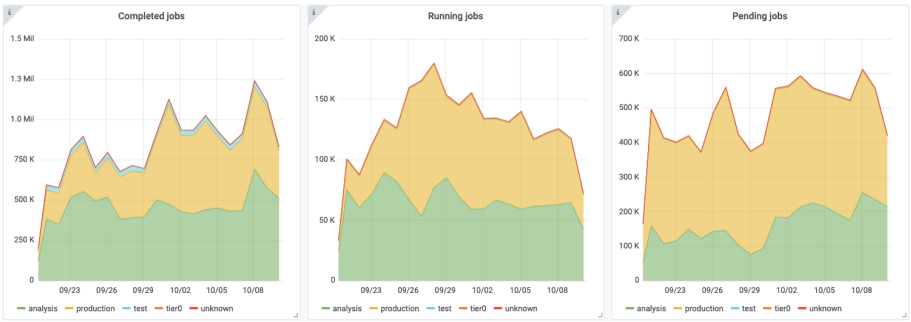

Figure 3. Screenshot of the CMS job monitoring dashboard.

The SSB dashboards are currently under review from CMS site experts and operator teams, and are expected to be in production in early 2020.

In addition to the MONIT infrastructure, we are evaluating additional monitoring technologies to increase the variety of monitoring applications available to the CMS groups. For distributed computing operations it is crucial to spot in real time possible failures in the various subsystems or at the different sites, and promptly notify the experts. Different operation teams are typically interested in different sets of metrics. For example, a site team is interested in the performances of all jobs at certain sites, the central production team is interested in the status of all jobs belonging to a certain production campaign. To tackle this use case, we deployed a message-based monitoring system using the Neural Autonomic Transport System (NATS) [22] for message delivering within system components. The production of the monitoring information is completely decoupled from its distribution and visualization components. The monitoring data is produced by services and delivered through NATS. The NATS messages are then converted into the metrics of interest and injected into VictoriaMetrics [23], an open-source time-series database used as Prometheus back-end storage. These metrics can be visualized in Grafana dashboards, and the Prometheus AlertManager [24] is used to notify the operation teams through various channels, such as Slack or email groups, about potential issues.

6 Conclusion

During the past year, the CMS offline and software computing team ported several of its monitoring applications from custom solutions to open source products such as ElasticSearch, InfluxDB and HDFS. The use of such technologies allows the development of a common data flow for CMS monitoring applications, therefore limiting the needs for custom developments and easing maintenance in the long term. A complementary approach based on NATS is available to target specific use cases requiring additional personalisation and prompt delivery of the monitoring information.

The availability of the monitoring information in widely used technologies such as ElasticSearch or HDFS allows the easy development and integration of further analytic workflows, based for example on Apache Spark and its ecosystem of products and libraries. We envisage for the future that predictive techniques such as anomaly detection will be studied to further improve the current alert systems. In addition, we are exploring the possibility to use the monitoring technologies discussed in this paper for a wider range of applications in CMS (for example, data quality monitoring) than those needed by the distributed computing systems.

7 Acknowledgements

This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement LHCBIGDATA No. 799062.

References

[1] CMS Collaboration, JINST **3** S08004 (2008)
[2] I. Bird et al, CERN-LHCC-2014-014, LCG-TDR-002 (2014)
[3] I. Sfiligoi et al, WRI World Congress on Computer Science and Information Engineering, **2428-432** (2009)
[4] D. Thain, T. Tannenbaum, and M. Livny, **17**, No.2-4, pages 323-356 (2005)
[5] T. Ivanov et al, EPJ Web Conf. **214** 03006 (2019)
[6] M. Giffels, Y. Guo, V. Kuznetsov, N. Magini and T. Wildish, Journal of Physics: Conference Series, **513**, Track 4
[7] A. Aimar et al, EPJ Web Conf., **214** 08031 (2019)
[8] Apache ActiveMQ, *http://activemq.apache.org*
[9] Logstash, *https://www.elastic.co/products/logstash*
[10] Apache Kafka, *http://kafka.apache.org*
[11] Apache Spark, *http://spark.apache.org*
[12] Elasticsearch, *http://elastic.co*
[13] Apache Hadoop, *http://hadoop.apache.org*
[14] InfluxDB, *https://www.influxdata.com/time-series-platform/influxdb/*
[15] Prometheus, *https://prometheus.io/*
[16] Kibana, *https://www.elastic.co/products/kibana*
[17] Grafana, *http://grafana.org*
[18] Piparo D et al, Future Generation Computer Systems, (2016)
[19] Jupyter, *https://jupyter.org*
[20] CMSMonitoring framework, *https://github.com/dmwm/CMSMonitoring*
[21] CMSSpark framework, *https://github.com/dmwm/CMSSpark*
[22] NATS *https://nats.io/*
[23] VictoriaMetrics *https://victoriametrics.com/*
[24] Prometheus AlertManager *https://prometheus.io/docs/alerting/alertmanager/*
[25] Apache Flume *https://flume.apache.org/*