Advanced Monitoring Capabilities of the CMS Experiment for LHC Run 3 and Beyond

Federica Legger[1], Nikodemas Tuckus[2], and Brij Kishor Jashal[3*] for the CMS Collaboration

[1]Istituto Nazionale di Fisica Nucleare, Torino, Italy
[2]CERN, Switzerland
[3]Rutherford Appleton Laboratory, UK

Abstract. The Compact Muon Solenoid (CMS) experiment computing infrastructure, spread globally over 150 Worldwide LHC Computing Grid (WLCG) sites, forms an intricate ecosystem of computing resources, software, and services. In 2024, the production computing cores breached the half-million mark and storage capacity reached 250 petabytes on disk and 1.20 exabytes on tape. To monitor these resources in real time, CMS, working closely with CERN IT, has developed a multifaceted monitoring system providing real-time insights using more than 100 dashboards in production by CMS teams. In preparation for Run 3, the CMS monitoring infrastructure underwent significant evolution to broaden the scope of monitored applications and services while enhancing sustainability and ease of operation. Leveraging open-source solutions, provided either by the CERN IT department or managed internally, monitoring applications have transitioned from bespoke solutions to standardized data flow and visualization services. Notably, monitoring applications for distributed workload management and data handling have migrated to utilize technologies like OpenSearch, VictoriaMetrics, and HDFS, with access facilitated through programmatic APIs, Apache Spark, or Sqoop, and visualization primarily via Grafana. The majority of CMS monitoring applications are now deployed on Kubernetes clusters based on a micro-services architecture. This contribution unveils the comprehensive stack of CMS monitoring services, showcasing how the integration of common technologies enables versatile monitoring applications and addresses the computation demands of LHC Run 3. Additionally, it explores the incorporation of analytics into the monitoring framework, demonstrating how these insights contribute to the operational efficiency and scientific output of the CMS experiment.

1 Introduction

The Compact Muon Solenoid (CMS) experiment [1] relies on an extensive computing infrastructure spanning over 150 Worldwide LHC Computing Grid (WLCG) [2] sites to process and store data. Monitoring these services is essential for efficient operations, ensuring reliability, scalability, and performance.

The monitoring system has evolved significantly since its inception, incorporating advanced tools and techniques to address the increasing complexity of CMS operations. This

*Corresponding author: brij.kishor.jashal@cern.ch

© The Authors, published by EDP Sciences. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

paper provides an overview of the current monitoring infrastructure, key advancements, and future directions.

To highlight the importance of monitoring, consider the scale of CMS data operations, which involve petabytes of data generated annually, requiring seamless coordination between computing resources across the globe. This scale necessitates real-time data analytics, intelligent alerting, and historical insights to optimize workflows and reduce downtime.

2 Requirements and Infrastructure

2.1 Baseline Requirements

The CMS monitoring system must meet several key requirements to ensure effective operations. Firstly, real-time monitoring and alerting capabilities are paramount, enabling quick detection and resolution of issues. As the dataset grows, scalability becomes critical, allowing the system to handle increased data volumes and service demands without compromising performance.

Another important aspect is long-term data storage and analytics. Historical trends provide invaluable insights, helping identify patterns and optimize system performance. User-friendly dashboards and tools are equally vital, as they empower stakeholders to access and interpret critical metrics without needing technical expertise.

Finally, the integration of diverse monitoring data sources into a cohesive platform ensures streamlined operations and reduces complexity. By consolidating data pipelines and utilizing advanced analytics, CMS achieves a unified view of its computing ecosystem.

2.2 Monitoring Infrastructure

The CMS monitoring infrastructure is built on a robust foundation of modern technologies. High availability is achieved through the use of Prometheus and AlertManager [3], VictoriaMetrics [4], and almost everything is running on Kubernetes [5], ensuring continuous uptime and reliability. These tools work together to collect, store, and analyze vast amounts of metric data.

For efficient data aggregation and transformation, Apache Spark [6] and Sqoop [7] are employed. CMS continues to use Apache sqoop for legacy workflows for now while exploring tools for replacing it since it is no longer supported by Apache. These tools handle the extraction, transformation, and loading (ETL) processes, enabling seamless data integration. Over 150 Grafana dashboards [8] are actively used, each tailored to specific metrics and user needs. These dashboards provide real-time visualizations, trend analyses, and detailed reports.

Moreover, the infrastructure integrates with the CERN-MONIT backend [9], a powerful analytics platform that consolidates monitoring data from various sources. This integration enhances the system's ability to generate actionable insights and predictive analytics. Additionally, CMS leverages the CMS Monitoring framework [10] for experiment-specific analytics.

Figure 1 illustrates the overall architecture of the monitoring system, showcasing the data flow from ingestion to visualization.

3 Monitored Components

The CMS monitoring system provides comprehensive coverage of various components, ensuring the smooth functioning of critical operations.

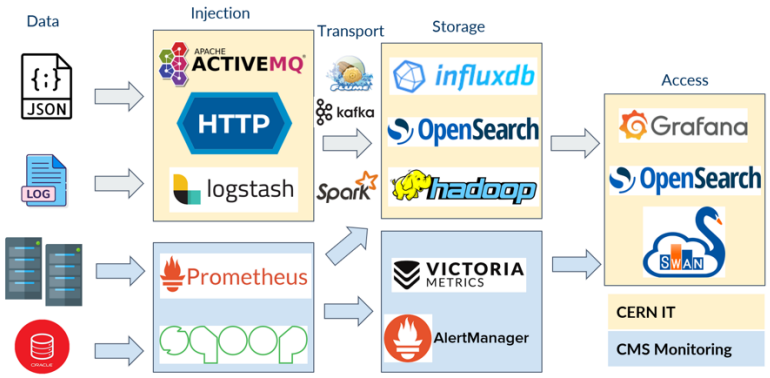

Figure 1. Overview of the CMS monitoring architecture. The flow includes data ingestion, processing, storage, and visualization.

3.1 Data Management Tools

Rucio [11] and the Data Bookkeeping Service (DBS) [12] are essential for managing datasets within the CMS infrastructure. Monitoring these tools ensures efficient tracking of datasets, including their sizes, access patterns, and replication statuses. For example, Rucio dataset monitoring provides a detailed view of dataset metrics, including creation dates, last access times, and storage endpoints.

3.2 Job Submission Frameworks

HTCondor [13] and the CMS Remote Analysis Builder (CRAB) [14] are pivotal in job submission and execution. Monitoring these frameworks provides insights into job efficiency, resource utilization, and failure rates. This information helps identify bottlenecks and optimize job scheduling, improving overall throughput. The HTCondor metric collection and processing chain are shown in Figure 2.

3.3 Storage Systems

Systems like EOS [15] and XRootD [16] are monitored to ensure reliable storage and data retrieval. Key metrics include storage availability, throughput, and error rates. Real-time monitoring of these systems helps detect and resolve issues promptly, minimizing downtime.

3.4 Network Performance

Monitoring data transfer rates and latency between sites is crucial for maintaining efficient workflows. Network performance metrics help identify connectivity issues and optimize data movement across the grid.

These components collectively generate millions of messages per hour, processed by Advanced Message Queuing (AMQ) brokers [17] and stored in OpenSearch [18] and HDFS [19] for advanced analytics. Table 1 summarizes the key metrics monitored.

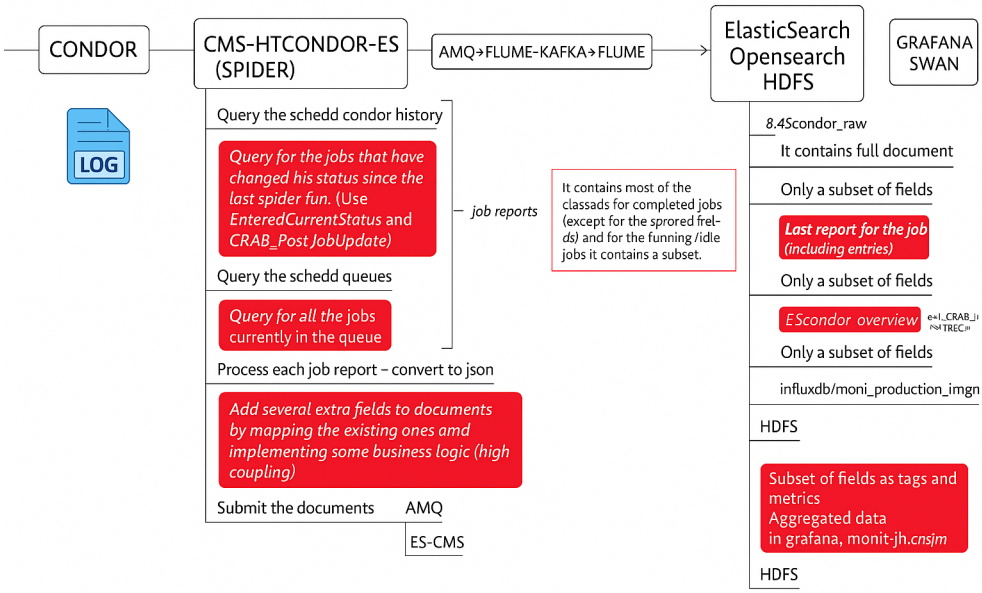

Figure 2. The CMS HTCondor ES (a.k.a cms-htcondor-es or spider) is a set of scripts which consumes CMS HTCondor ClassAds (Job processing docs), converts them to JSON documents and feeds them via CMSMonitoring/ActiveMQ (AMQ) Broker module to CERN MONIT infrastructure.

Table 1. Summary of monitored metrics.

Component	Metrics Monitored
Rucio	Dataset size, access frequency, replication status
HTCondor	Job status, resource usage, failure rates
EOS	Storage availability, throughput, error rates
XrootD	File transfer rates, errors, bandwidth utilization
Network	Transfer latency, bandwidth usage, packet loss

4 Key Advancements

4.1 Infrastructure Enhancements

The CMS monitoring infrastructure has undergone significant upgrades to enhance performance and usability. The migration from MongoDB [20] to OpenSearch, for instance, has dramatically improved query speeds and storage efficiency. This transition enables faster data retrieval, supporting real-time analytics.

Cronjobs, which were previously executed on virtual machines, have been migrated to Kubernetes. One of our bigger applications—the CMS Software (CMSSW) [21] popularity service that is used to receive UDP messages, transform them and send them to the HDFS and OpenSearch—was also moved to Kubernetes from a dedicated virtual machine. This shift ensures better resource utilization, scalability, and fault tolerance. By leveraging Kubernetes' capabilities, CMS achieves a more resilient and flexible infrastructure.

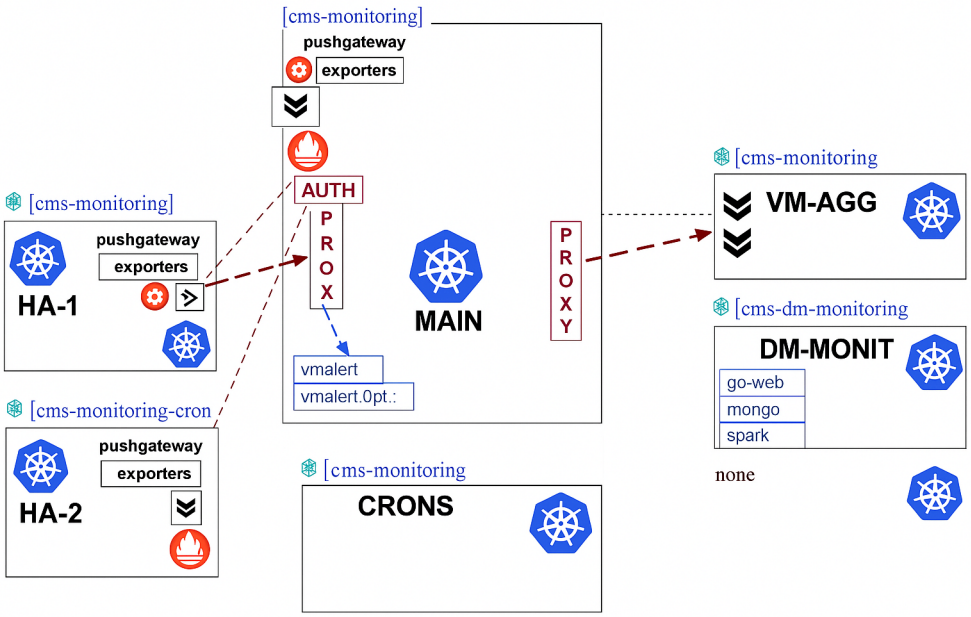

Figure 3. Architecture overview illustrating the integration of Prometheus (in High Availability mode) with VictoriaMetrics, AlertManager, and Exporters (MAIN, HA-1, HA-2). A single Grafana data source connects to thousands of Prometheus and VictoriaMetrics metrics through Promxy. Long-term time-series metrics storage is handled by aggregated VictoriaMetrics instances (VM-AGG). Rucio dataset monitoring is managed by DM-MONIT, and data aggregation/ETL batch jobs (e.g., Spark, Sqoop) are executed via CRONS.

Unified configuration management has also been introduced, consolidating ownership of data pipelines under a single service account. This approach simplifies system administration and reduces the risk of misconfigurations.

4.2 Visualization Improvements

The adoption of Grafana for data visualization has improved the way CMS stakeholders interact with monitoring data. Interactive dashboards provide a real-time overview of system health, while summarized charts offer quick insights into key metrics. Historical data views enable trend analysis, facilitating proactive decision-making.

Customizable widgets allow users to tailor dashboards to their specific needs, enhancing usability and accessibility. Figure 4 showcases a sample Grafana dashboard used in CMS monitoring showing CMSSSW UDP collector service.

5 Impact and Future Plans

The CMS monitoring system has significantly improved operations during Run 3. By enabling faster issue resolution and better resource planning, it has minimized downtime and enhanced system efficiency. Comprehensive reporting has increased transparency, benefiting both technical teams and stakeholders.

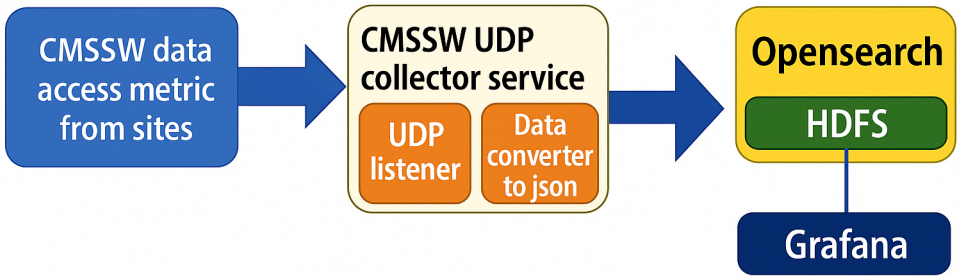

Figure 4. CMSSW popularity service chain of data processing with sample Grafana dashboard.

Looking ahead, CMS aims to expand its monitoring capabilities. Predictive analytics will be a key focus, leveraging machine learning models to anticipate issues and optimize workflows. The integration of anomaly detection algorithms will further enhance system reliability, identifying potential problems before they impact operations.

Support for new workflows and services in Run 4 and beyond is also planned. Unified APIs will facilitate cross-platform integration, ensuring seamless communication between diverse systems. These advancements will prepare CMS for the challenges of future LHC runs.

6 Conclusion

The CMS monitoring infrastructure has evolved into a robust and scalable system, meeting the challenges of LHC operations. Its continuous development ensures that CMS remains well-prepared for future demands, with the guiding principle that "what gets monitored gets improved."

Acknowledgments

The authors thank the CMS collaboration and CERN IT for their support in developing and maintaining the monitoring infrastructure. Special thanks to the teams managing the Grafana and OpenSearch systems for their contributions.

References

[1] The CMS Collaboration et al., "The CMS Experiment at the CERN LHC," JINST 3 (2008) S08004, doi:10.1088/1748-0221/3/08/S08004.
[2] WLCG, https://wlcg-public.web.cern.ch/, accessed May 2025.
[3] Prometheus, https://prometheus.io, accessed May 2025.
[4] VictoriaMetrics, https://victoriametrics.com, accessed May 2025.
[5] Kubernetes, https://kubernetes.io, accessed May 2025.
[6] Apache Spark, https://spark.apache.org, accessed May 2025.
[7] Apache Sqoop, https://sqoop.apache.org, accessed May 2025.
[8] Grafana, https://grafana.com, accessed May 2025.

[9] Aimar Alberto et al., "MONIT: Monitoring the CERN Data Centres and the WLCG Infrastructure," EPJ Web Conf. 214 (2019) 08031, doi:10.1051/epjconf/201921408031.

[10] Ariza-Porras, C., Kuznetsov, V. Legger, F. "The CMS monitoring infrastructure and applications," Comput Softw Big Sci 5, 5 (2021), doi:10.1007/s41781-020-00051-x.

[11] Barisits, M., Beermann, T., Berghaus, F. et al. Rucio: Scientific Data Management. Comput Softw Big Sci 3, 11 (2019). doi:10.1007/s41781-019-0026-3.

[12] M Giffels et al 2014 J. Phys.: Conf. Ser. 513 042022 "CMS Data Bookkeeping Service," doi:10.1088/1742-6596/513/4/042022.

[13] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: the Condor experience," Concurrency and Computation: Practice and Experience, vol. 17, no. 2–4, pp. 323–356, 2005, doi:10.1002/cpe.938.

[14] M Mascheroni et al 2015 J. Phys.: Conf. Ser. 664 062038., "CMS distributed data analysis with CRAB3," DOI 10.1088/1742-6596/664/6/062038.

[15] EOS, https://eos-web.web.cern.ch/eos-web/, accessed May 2025.

[16] Dorigo, A., Elmer, P., Furano, F., Hanushevsky, A. (2005). XROOTD - A highly scalable architecture for data access. WSEAS Transactions on Computers, 4(4), 348-353.

[17] Red Hat AMQ Broker, https://access.redhat.com/documentation/en-us/red_hat_amq/7.0/html/using_amq_broker/index, accessed May 2025.

[18] OpenSearch, https://opensearch.org, accessed May 2025.

[19] HDFS, https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html, accessed May 2025.

[20] MongoDB, https://www.mongodb.com, accessed May 2025.

[21] CMS Software (CMSSW), https://cms-opendata-workshop.github.io/workshop-lesson-cmssw/01-introduction/index.html, accessed May 2025.