Research on Visualization and Indexing of PV Data Based on the ELK Stack

Li Yukun, Cao Jianshe, Ye Qiang, Du Yaoyao

Institute of High Energy Physics, Chinese Academy of Sciences

Introduction

In response to the escalating demands of large-scale scientific facilities, this paper outlines an innovative real-time data acquisition and analysis solution utilizing Kafka and the ELK stack, focused on Beam Position Monitors (BPM) in particle accelerators. With traditional data processing methods buckling under rapidly increasing data volumes, this study introduces a high-throughput, low-latency, and reliable system designed to manage high-velocity data streams. By integrating Kafka for data collection and leveraging the ELK stack for processing and visualization, the proposed pipeline offers standardized data handling and real-time analytical capabilities. This streamlined approach not only enhances data processing efficiency and reliability but also significantly improves operational responsiveness, providing critical insights for advancements in experimental physics.

Aim

The core of this research is to develop a system capable of handling high-speed data streams and providing real-time feedback. The goal is to create an efficient, reliable, and real-time data processing system to optimize beam position monitoring in particle accelerators. Through the automation and standardization of data collection, as well as the enhancement of real-time analysis and visualization capabilities, the system can instantly monitor and respond to critical changes during experiments, thus improving the precision and efficiency of the experimental processes.

Method

To construct an efficient and reliable real-time beam position data processing system, this study employs advanced designs and methods in data acquisition, data processing, data visualization, system architecture, and system reliability and scalability.

- **Data Collection:** PV variables from BPM electronic devices are automatically transmitted to the Kafka message queue, standardizing data collection.
- **Data Processing:** The data is processed by Logstash, which enriches the incoming data streams and forwards them in JSON format to Elasticsearch for storage and indexing.
- **Data visualization：** By transforming PV data into indexed form, it is serialized and stored in Elasticsearch. Kibana searches for data indices from Elasticsearch and presents the data in a visual format. Kibana also supports various data sources.
- **System Architecture:** The system utilizes a microservices architecture, deployed on a Kubernetes cluster using Docker containers to enhance scalability, portability, and ease of deployment across different environments.
- **Operational Reliability and Scalability:** The integration of Kafka with Zookeeper ensures high reliability and consistency in data handling within the Kafka cluster, while the ELK stack facilitates efficient data search, analysis, and visualization.

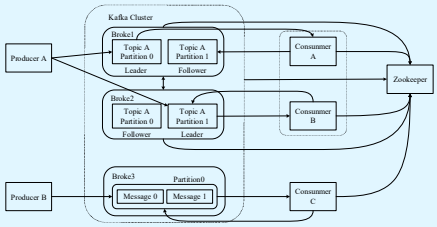

Fig.1 *Kafka architecture diagram*

Kafka is a distributed streaming platform used primarily for handling high-speed and high-volume data streams. Here are the key components depicted in the diagram and their functions:

- **Producers:**Producers are responsible for publishing messages to Kafka. In the diagram, Producer A and Producer B are both message publishers.Each producer can send messages to one or more Kafka topics.
- **Topics and Partitions:**Kafka organizes data into topics. Each topic can be divided into multiple partitions, where messages are stored.In the diagram, Topic A is split into two partitions: Partition 0 and Partition 1. This design enhances data throughput and scalability.Messages are distributed to different partitions, typically based on the hash value of a key.
- **Consumers:**Consumers read data from Kafka topics. They can be standalone applications or services tasked with processing data and possibly performing further operations.Consumers can form a group to consume a topic together, improving processing efficiency.In the diagram, Consumer A, Consumer B, and Consumer C are consumers who can read data from different partitions of the same topic.
- **Zookeeper:**Zookeeper manages Kafka's metadata and maintains the cluster's state, such as topic configuration information and partition assignment.It also coordinates the work of consumers, such as determining which consumers are responsible for which partitions.

Results

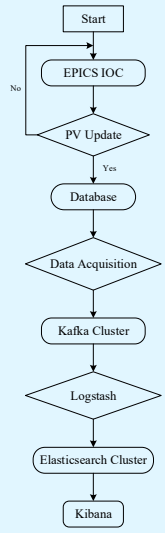

Fig.2 *Data Transmission Flowchart*

When the PV data in the IOC is updated, the data is written to the database. The data acquisition program retrieves the updated data from the database, converts it into JSON format, and writes it to a data cache. At this point, the data in the cache is in array form; it is converted into a list format, and the cached data is then written into a specified Kafka message topic. Once the data is written to Kafka, it is sent to the Elasticsearch cluster via the Logstash pipeline and a data index is created, ultimately realizing data visualization on the Kibana interface.

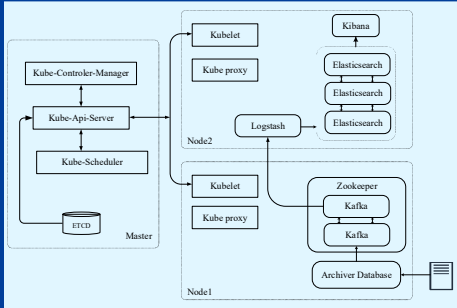

Fig.3 *System architecture diagram*

To enhance the system's compatibility across various environments, as well as to improve its portability and scalability, the entire system described in this document employs a microservices architecture. It uses the Docker container engine to deploy Kafka, Zookeeper, Logstash, Elasticsearch, and Kibana within a three-node Kubernetes high-availability cluster. The data acquisition program is implemented using Python.

Configuration for services such as Kafka, Zookeeper, and the ELK stack is defined through the writing of YAML files. These configuration files enable the definition of services within a Kubernetes cluster, including Service, Deployment, and Statefulset. Deployments are typically used to manage stateless applications, and services like Kafka, Zookeeper, Logstash, and Kibana are categorized as Deployment types, which usually do not involve data storage issues. Kafka, serving as a message queue, typically does not persist data continuously. Statefulsets are designed to run applications that require persistent states. When configuring the Elasticsearch service, 3 replicas were created, and a persistent volume (PersistentVolume) was also created for each of the three pods, ensuring each pod has its own persistent storage. Even if a pod is rescheduled to another node, its storage will be remounted to the corresponding pod.

Conclusion

This research proposes a method for real-time analysis and visualization of PV data within BPM using the Kafka message queue and ELK stack. Given the extensive data querying and processing requirements, this method has significant practical value. The research approach combines Docker container technology and a highly available Kubernetes cluster with a microservices architecture, enhancing system stability and scalability for handling large-scale data in major scientific installations. The system has undergone prolonged testing in a production environment, demonstrating good operational performance and validating the technical feasibility of this approach. Future research will integrate different system data collection, querying, and processing needs to further enhance system performance and reliability.

Acknowledgements

Special thanks to Mr Qiang Ye and Mr Yaoyao Du for the assistance in data collection. Thank to Prof. Jianshe Cao for his supporting to my research.