International Journal of Internet, Broadcasting and Communication Vol.15 No.4 270-278 (2023)
http://dx.doi.org/10.7236/IJIBC.2023.15.4.270

IJIBC 23-4-32

A Container Orchestration System for Process Workloads

Jong-Sub Lee*, Seok-Jae Moon**

**Professor, College of General Education, SeMyung University, Jecheon, Korea*
***Professor, Department of Artificial Intelligence Institute of Information Technology, KwangWoon University, Korea*
e-mail : *99jslee@semyung.ac.kr*, msj8086@kw.ac.kr

Abstract

We propose a container orchestration system for process workloads that combines the potential of big data and machine learning technologies to integrate enterprise process-centric workloads. This proposed system analyzes big data generated from industrial automation to identify hidden patterns and build a machine learning prediction model. For each machine learning case, training data is loaded into a data store and preprocessed for model training. In the next step, you can use the training data to select and apply an appropriate model. Then evaluate the model using the following test data: This step is called model construction and can be performed in a deployment framework. Additionally, a visual hierarchy is constructed to display prediction results and facilitate big data analysis. In order to implement parallel computing of PCA in the proposed system, several virtual systems were implemented to build the cluster required for the big data cluster. The implementation for evaluation and analysis built the necessary clusters by creating multiple virtual machines in a big data cluster to implement parallel computation of PCA. The proposed system is modeled as layers of individual components that can be connected together. The advantage of a system is that components can be added, replaced, or reused without affecting the rest of the system.

Keywords: *Process Workload, Cloud Computing, Container Orchestration, Machine Learning, Metadata Registry*

1. INTRODUCTION

In the field of industrial automation based on distributed cloud computing, information exchange is essential, and techniques used to improve business process models and services through it are necessary [1]. To do this, it is necessary to manage, process, and analyze the data collected during manufacturing production. Accordingly, the service of big data and machine learning technology is needed in the field of industrial automation [2]. In this case, the most important step is to establish a standard and resilient architecture that integrates metadata-based big data and machine learning technologies for efficient industrial data analysis [3]. We propose a container orchestration system for process workloads for enterprise process-oriented workload integrated execution by integrating the potential of big data and machine learning technology. The proposed system consists of 4 layers: Process Workload, Functional Layer, Information Layer, and Asset Integration.

Manuscript Received: october. 4, 2023 / Revised: october. 21, 2023 / Accepted: october. 27, 2023
Corresponding Author: msj8086@kw.ac.kr
Tel:*** - **** - **** Fax: +82-10-916-4751
Author's affiliation: Professor, Department of Artificial Intelligence Institute of Information Technology, KwangWoon University, Korea

Copyright© 2023 by The Institute of Internet, Broadcasting and Communication. This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.org/licenses/by-nc/4.0)

Since the data generated in the process creation stage are heterogeneous in form and use, the proposed system can function to integrate them into a common information model. And it applies a standard metadata protocol to ensure heterogeneous data exchange. This is because data is described through semantics and becomes information, so the information model was constructed by standardizing the metadata registry. Various types of data storage are used to store process data, metadata and analytical models. In addition, this proposed system analyzes big data generated from industrial automation to identify hidden patterns and builds a machine learning (ML) prediction model. For each machine learning use case, the training data is loaded into a data store and configured to be pre-processed for model training. The next step is to use the training data to select and apply an appropriate model. The model is then evaluated using the following test data. This step is called model building, and can be performed in a batch processing framework. In addition, a service that configures a visual layer to display prediction results and facilitate big data analysis is also configured. The structure of this thesis is as follows. Chapter 2 describes related research, and Chapter 3 describes the components and operating scenarios of the proposed system. In Chapter 4, application cases and comparative analysis are described, and finally, in Chapter 5, conclusions are made.

2. RELATED WORK

Existing machine learning algorithms struggle to process the massive amounts of data generated by smart production systems. This is because it is designed under the assumption that the data set and model parameters must be completely loaded into memory [5]. Scalable ML algorithms are a common way to solve this problem, as they are well-suited to handling large datasets and/or models with many parameters. In particular, distributed ML algorithms represent most state-of-the-art scalable ML methods [6]. It can be split into two groups of algorithms that use different methods of parallelism: data parallelism and model parallelism. In the first group, data sets are divided into smaller pieces that are stored on nodes of a computer cluster. All parameters of the model are partially updated simultaneously at each node and combined afterwards. In the second group, the model parameters are divided into subsets and updated simultaneously at each node using the full data set. There are also several hybrid methods in which the dataset as well as the model parameters are partitioned and distributed to clusters [7, 8]. In recent years, several tools have been developed that allow the use of distributed ML algorithms on big data. Mahout, Spark MLlib and H2O are most used in industry and academia [9]. Each of these can be combined with different distributed data processing engines. For example, Mahout can work with MapReduce, Spark, and H2O. These frameworks represent a distributed batch (offline) learning paradigm, where models are trained on a training dataset consisting of historical data before being used to process new data [9]. In contrast, the stream (online) paradigm is required when an algorithm learns from data arriving as a stream. A relatively young framework called Samoa provides these ML algorithms in its distributed stream processing engines Storm, S4 and Samza [9]. A very detailed comparison of the mentioned four distributed ML frameworks in terms of algorithm availability, scalability and speed is discussed in [10].

3. PROPOSED SYSTEM: CONTAINER ORCHESTRATION SYSTEM FOR PROCESS WORKLOADS TITLE AND AUTHOR INFORMATION

3.1 System Component

In this chapter, we propose a container orchestration system for process workloads for integrated execution of process-oriented workloads by integrating big data and machine learning technologies. It is common for most process workload services to use batch data to build models that will later be deployed for online

prediction on stream data in industrial use cases. The system proposed in this paper consists of different layers as shown in Figure 1.

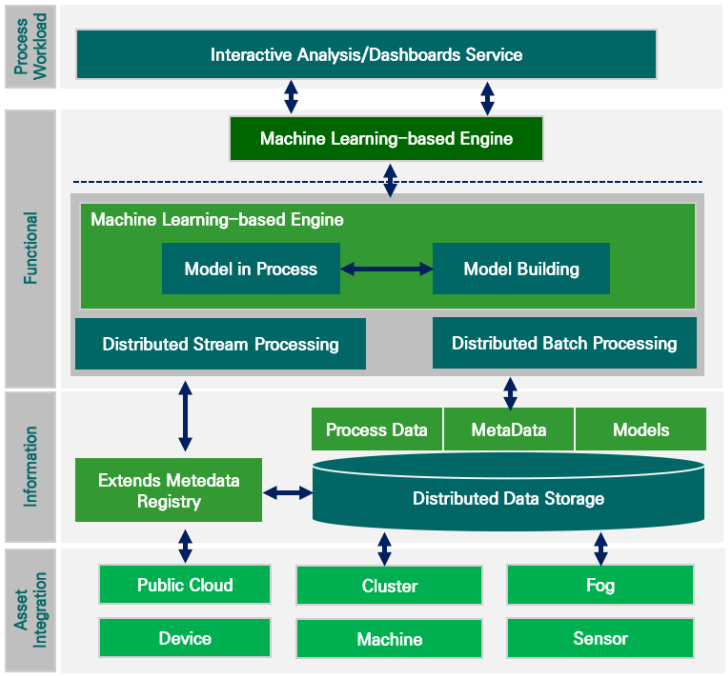

Figure 1. The Conceptual Architecture

- **Process Workload.** In this layer, business requirements, use case descriptions and the workload to be solved are defined. The results of data analysis for the process help stakeholders make appropriate decisions and optimize. In particular, the connectivity of the system for data sharing must be expanded and visually visible. Visual interfaces and dashboards are important for grasping difficult concepts or identifying hidden patterns within process data. The visualization layer displays prediction results and allows data scientists to add expertise in the form of semantic annotations to facilitate analysis tasks. So, this tier includes interactive analytics software, dashboards and client applications.

- **Functional Layer.** This layer is about analyzing data to uncover hidden patterns and building ML predictive models. Training data for each machine learning use case is loaded from a data store and pre-processed for model training. The next step is to select and apply an appropriate model using the training data. The model is then evaluated using the training data. These steps are called model construction and can be performed in distributed batch processing. Once a trained model is determined to be suitable for solving a problem on a business process workload, it can be deployed in distributed stream processing for online prediction on stream data.

- **Information Layer.** In this layer, shared data in the process is provided using semantic techniques. Semantics are applied in a metadata registry or a standardized information model for a particular branch. Shared data is maintained at this layer for later access and analysis. Therefore, in this paper, EMRA, an extended version of metadata, is applied. EMRA uses various types of data storage systems to store process data, metadata analysis and analysis models. These storage systems must be simultaneously scalable and highly coupling-tolerant.

- **Asset Integration Layer.** In this layer, components such as machines, people, products, and engineering systems are included. These components represent the company's primary data sources. The data generated by this layer is heterogeneous in form and purpose and needs to be unified through a common information model. Also, the transition from the physical environment to the virtual environment occurs at this layer. This includes the infrastructure and resources to capture digital/analog signals and make them available on the network as form data.

3.2 Sequence Diagram

Figure 2 show the flow of the proposed system in a sequence diagram.

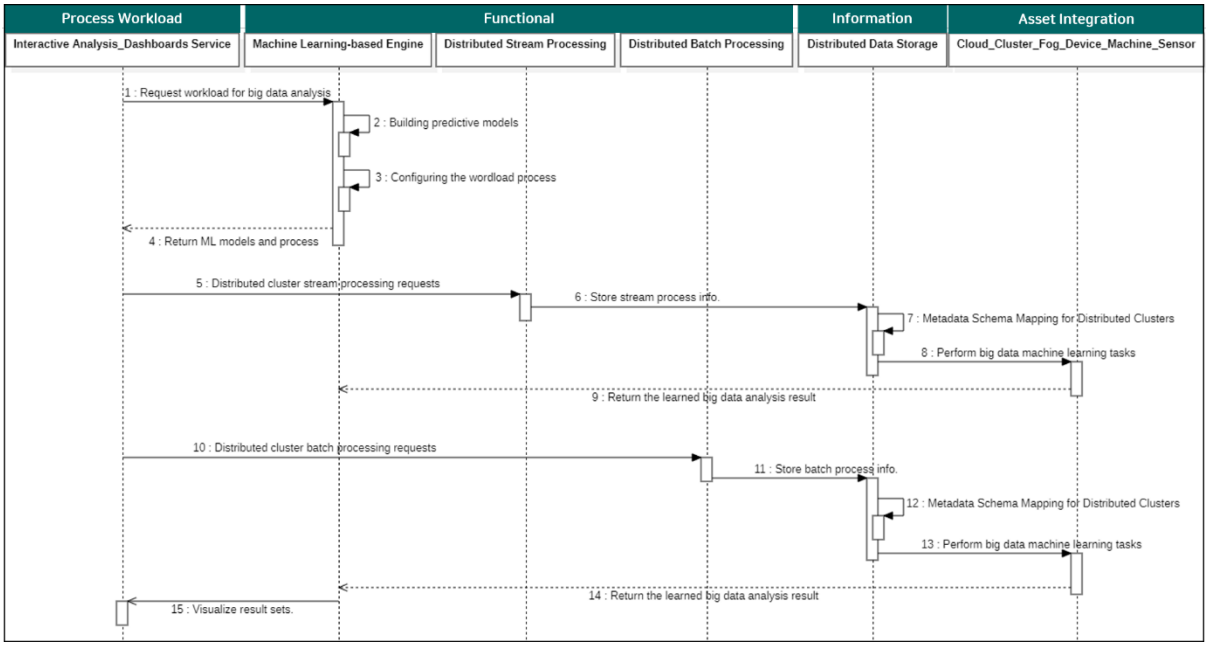

Figure 2. Sequence Diagram of the Proposed System

3.3 Principal Component Analysis

This section demonstrates the benefits of the proposed architecture for specific application cases related to data analysis in enterprise environments. Data analysis often requires learning a system model from historical data and using the learned model for evaluation or processing of current process data. An example of such a model is the Principal Component Analysis (PCA) matrix, which has versatile enterprise applications such as dimension reduction and condition monitoring approaches [32–34]. The PCA matrix can be obtained by k = 1… It is computed from the vector x_k of historical measurements recorded at time instances of n.

$$\sum x = \frac{1}{n-1}\sum_{k=1}^{n}(x_k - \mu_x)(x_k - \mu_x)T \quad (1)$$

$$\mu_x = \frac{1}{n}\sum_{k=1}^{n} x_k \quad (2)$$

Calculating the covariance matrix Σx requires O($n \times n2c$) operations, where nc denotes the number of

components in the past measurement vector. The automated computation for singular value decomposition is of the order of O($n3c$) [35]. In big data applications, a large number of past measurement vectors are evaluated so that the nc << n condition holds. In these applications, the computational effort for SVD is negligible compared to that for computing the covariance matrix. Parallel computation of the covariance matrix Σx is achieved using the relationship between Equations (3), (4), (5), (6) and Equation (7). Computing the sum of S1 and S2 on multiple workers is straightforward, but the entire covariance matrix Σx is quickly computed on a single machine in conjunction with equation (6).

$$\sum x = \frac{1}{n-1}\sum_{k=1}^{n}(x_k - \mu_x)(x_k - \mu_x)T \quad (3)$$

$$= \frac{1}{n-1}\sum_{k=1}^{n}(x_k x_k^T + x_k\mu_x^T - \mu_x x_k^T - \mu_x\mu_x^T) \quad (4)$$

$$= \frac{1}{n-1}\sum_{k=1}^{n}(x_k x_k^T) - \frac{1}{n(n-1)}(\sum_{k=1}^{n} x_k)(\sum_{k=1}^{n} x_k) \quad (5)$$

$$= \frac{1}{n-1}S_2 - \frac{1}{n(n-1)}s_1 s_1^T \quad (6)$$

$$S_1 = \sum_{k=1}^{n} x_k \quad and \quad S_2 = \sum_{k=1}^{n}(x_k x_k^T) \quad (7)$$

Computing S_1 and S_2 on multiple workers is straightforward, but the entire covariance matrix Σx is quickly computed on a single machine by linking Equation (6).

4. APPLICATION CASES AND COMPERATIVE ANALYSIS

In this chapter, in order to implement parallel computing of PCA in the proposed system, several virtual systems were implemented to build clusters necessary for big data clusters.

-Implementation of PCA: To implement the parallel computation of PCA, several virtual machines were created in the big data cluster to build the necessary cluster. VM resource specifications in Table 2 are applied. Since Figure 1 does not show many details of the platform presented, information on the technology, infrastructure and configuration used is shown in Table 1.

Table 1. VM Resource Specification

Item	Details
Processor	Intel(R) Xeon(R) CPU E5-2697 v3 @ 2.60GHz - 8 Virtual CPUs (4 sockets with 2 cores per socket)
Memory	32 GB
Storage	512 GB HDD
Network	1 Gbit/s network card
OS	Ubuntu 16.04 xenial

Table 2. Cluster Description

Category	Framework	Description
Data Ingestion	Kafka Cluster	3 brokers, 20 partitions for the input and output topics
Data Storage	Hadoop Cluster	1 name node and 3 data nodes

	MongoDB	single node
	Influx DB	single node
Batch Processing	Spark Cluster	1 master node and 3 workers
Stream Processing	Kubernetes Cluster	1 master and 3 nodes

The PCA algorithm was implemented on a big data platform with a data set of approximately 36 million historical measurement vectors (each with 41 components) consisting of enterprise process sensor data generated using the TESIM simulation model [36]. The implementation was carried out in two modes.

A) Batch Processing Mode (PCA Model Building): In this mode, the training data set is uploaded as a distributed Spark data frame in the HDFS system, and the number of different Spark workers is calculated. The PySpark PCA class is used to train a model to project vectors into a low-dimensional space. The result of this calculation process is a 41*20 PCA matrix, which will later be used for PCA stream processing.

B) Stream Processing Mode (PCA Model Application): Since Kafka is already used as a stream platform in big data platforms, Kafka Streams is a good and simple option for developing stream (real-time) applications to take advantage of the Kafka platform. Stream data (a vector of 41 components) is collected from the TESim OPC UA server simulator and continuously pushed to a Kafka topic. A Kafka Streams application consumes data from a Kafka input topic, calculates the low-dimensional space for each input (vector) based on the PCA matrix, and then sends the result to another Kafka output topic. This application is developed in Java and uses the Kafka Stream library. However, for simplicity and portability, Java applications are packaged as Docker images that can be deployed in a variety of environments. Dockerized Kafka Streams applications can also run anywhere using multiple Instances (parallel containers), but require a distributed platform for orchestration and automatic scaling. A Kubernetes cluster was used for this.

C) Evaluation: To evaluate the batch processing part of the proposed big data platform, the PCA matrix was computed in parallel on multiple workers in the Apache Spark Framework, and each worker's unique number of workers and cores were evaluated. For each experimental setting, i.e. each combination of operator and core, $n \geq 36$ million past measurement vectors with $nc = 41$ components were used to compute the PCA matrix. Batch processing throughputs of 11859 and 75203 processed records per second were achieved using a single core and eight cores, respectively, in a single worker. In Figure 3, using a parallel implementation with 3 workers (8 cores each) the throughput is improved to 220238 processed records per second. That is, 2.92 times faster.

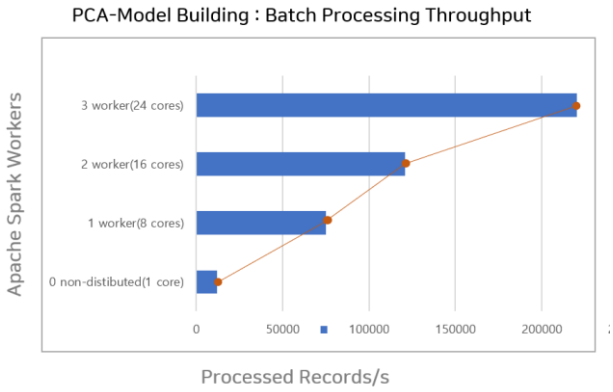

Figure 3. PCA Model Building: Batch Processing Throughput

The introduced architecture is not limited to learning process models, but is tailored to the application of learned models. After learning a PCA model, we can deploy and apply the model on individual processing nodes to achieve dimensionality reduction on local nodes. For this purpose, the Kafka system creates a subject and distributes it into many partitions among intermediaries, one partition for each plant or asset, and enables parallel processing on these partitions. Stream processing is evaluated on continuous data of measurement vectors with nc = 41 components at several instances (1, 2, 3, 10, 20 instances). According to the results, using 20 stream application instances instead of a single application instance can increase the number of records processed per second by 18.6 times from 2736 to 50973 Fi

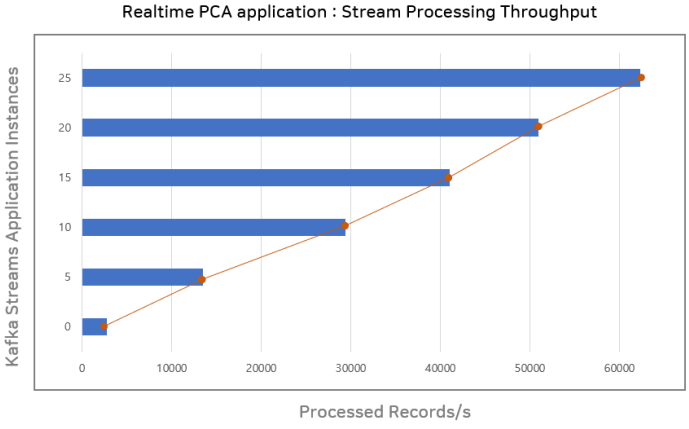

Figure 4. Realtime PCA Application: Stream Processing Throughput

One thing to note here is that in some cases, for example, the performance factor may outweigh the instance count. When the number of instances is a multiple of the number of Kafka brokers (3 in the setup), the interpretation of this is that the computation will be split evenly between the application instances. Finally, it should be taken into account that the parallelization factor can be adjusted based on available resources. For example, the performance of a stream application starts to decrease by 40 (3 brokers) because the number of Kafka brokers is constant. So, you need to scale your Kafka cluster to meet your requirements.

5. CONCLUSION

In this paper, we proposed a container orchestration system for process workloads for enterprise-wide

process-oriented workload integration by combining the potential of big data and machine learning technologies. The proposed system satisfies the following requirements. First, it uses an integrated OPC UA information model with semantic information to meet data integration requirements. Depending on the process data description, the machine's technology, components, and method of executing a given task, multiple machines can share the same information model and easily communicate and use mutual services. Additionally, a schema registry in the message broker system and/or data store facilitates data integration between higher-level systems. Second, the system allows data to be collected not only from legacy systems and tools, but also from other types of machines using a variety of protocols. Supports data collection and processing in both batch and stream modes. Third, all frameworks and systems used for implementation, including Kafka, HDFS, MongoDB, InfluxDB, Spark, and Kubernetes, are distributed, scalable, and redundant across various layers. Fourth, the proposed system is modeled as layers of individual components that can be connected to each other. Future research tasks should also consider security aspects from the asset layer (supported by OPC UA) to the business layer by activating data security technologies such as encryption, authentication, and authorization functions in the framework.

ACKNOWLEDGEMENT

This paper was supported by the SeMyung University Research Grant of 2023.

REFERENCES

[1] F. Li and G. Fang, “Process-Aware Accounting Information System Based on Business Process Management,” Wireless Communications and Mobile Computing, vol. 2022. Hindawi Limited, pp. 1–15, 09-May 2022. DOI: https://doi.org/10.1155/2022/7266164

[2] T. Czvetkó, A. Kummer, T. Ruppert, and J. Abonyi, “Data-driven business process management-based development of Industry 4.0 solutions,” CIRP Journal of Manufacturing Science and Technology, vol. 36. Elsevier BV, pp. 117–132, Jan 2022. DOI: https://doi.org/10.1016/j.cirpj.2021.12.002

[3] R. Pedral Sampaio, A. Aguiar Costa, and I. Flores-Colen, “A Systematic Review of Artificial Intelligence Applied to Facility Management in the Building Information Modeling Context and Future Research Directions,” Buildings, vol. 12, no. 11. MDPI AG, p. 1939, 10 Nov 2022. DOI: https://doi.org/10.3390/buildings12111939

[4] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, “A survey of machine learning for big data processing,” EURASIP Journal on Advances in Signal Processing, vol. 2016, no. 1. Springer Science and Business Media LLC, 28 May 2016. DOI: https://doi.org/10.1186/s13634-016-0355-x

[5] L. Zhou, S. Pan, J. Wang, and A. V. Vasilakos, “Machine learning on big data: Opportunities and challenges,” Neurocomputing, vol. 237. Elsevier BV, pp. 350–361, May 2017. DOI: https://doi.org/10.1016/j.neucom.2017.01.026

[6] M. Nasser Al-Andoli, S. Chiang Tan, and W. Ping Cheah, “Distributed parallel deep learning with a hybrid backpropagation-particle swarm optimization for community detection in large complex networks,” Information Sciences, vol. 600. Elsevier BV, pp. 94–117, Jul 2022. DOI: https://doi.org/10.1016/j.ins.2022.03.053

[7] E. Ezugwu et al., “A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects,” Engineering Applications of Artificial Intelligence, vol. 110. Elsevier BV, p. 104743, Apr 2022. DOI: https://doi.org/10.1016/j.engappai.2022.104743

[8] N. Richter, T. M. Khoshgoftaar, S. Landset, and T. Hasanin, “A Multi-dimensional Comparison of Toolkits for Machine Learning with Big Data,” 2015 IEEE International Conference on Information Reuse and Integration. IEEE, Aug 2015.
DOI: https://doi.org/10.1109/IRI.2015.12

[9] G. D. F. Morales and A. Bifet, “Samoa: scalable advanced massive online analysis.” Journal of Machine Learning Research, 2015.
DOI: https://doi.org/10.5555/2789272

[10] A.Shahraki, M. Abbasi, A. Taherkordi, and A. D. Jurcut, “A comparative study on online machine learning techniques for network traffic streams analysis,” Computer Networks, vol. 207. Elsevier BV, p. 108836, Apr 2022.
DOI: https://doi.org/10.1016/j.comnet.2022.108836