TECHNIUM

Technium Vol. 5, pp.13-20 (2023)
ISSN: 2668-778X
www.techniumscience.com

GamaBox-One: A Proposed Architecture for Cloud-based Big Data Management Platform for Multipurpose Computation using Hybrid Architecture

Mardhani Riasetiawan*[1,2], Ahmad Ashari[1,2], Tri Kuntoro Priyambodo[1,2], Yohannes Suyanto[1,2], Bambang Nurcahyo Prastowo[1,2], Abdul Rouf[1,2], Idham Ananta Timur[1,2], Triyogatama Wahyu Widodo[1,2], I Gede Mujiyatna[1,2], Bagaskoro Saputro[1]

[1]Department of Computer Science and Electronics, Faculty of Mathematics and Natural Sciences
[2]Center of Excellence for Big Data Research, Computer System and Network Research Labs
Universitas Gadjah Mada, Yogyakarta, Indonesia
mardhani@ugm.ac.id*(corresponding author)

Abstract. The architecture of the data center is the main key for producing a highly functional demand big data management platform for multipurpose uses. Nowadays various technologies have come to provide construction of that purpose, providing several use cases for big data analytics and processing. In this paper, we want to explore possibilities of architecture that had to be built in answer to the multipurpose data center, such as analytical research, scientific simulation, machine learning, deep data learning, and data orchestration. We discover how Hadoop and its element supporter can be used alongside cloud orchestrators such as Terraform or Occopus and container orchestrators such as Kubernetes or Docker Swarm. We also provide possible supporting components that can handle the different jobs in High-Performance Computing and how the system can be secured. Our proposed approach in this research has developed the architecture for cloud-based big data management for multipurpose computation.

Keywords. Architecture, data center, big data, cloud, scientific computation.

1. Introduction

Data is future gold, while a data center is the refinery infrastructure for producing the best value gold that humans can invest in. In the process of making gold from raw materials into high-quality precious metal, a system and architectural design of each infrastructure hold the main key. Fine gold cannot be a byproduct of the entire process if each process is not efficient, effective, and precisely managed. A similar analogy can be applied to data infrastructure to produce high-quality insights. In big data environments, great quality of system and architecture design for the data center is somewhat necessary

From previous studies, we find that technologies supporting big data platforms have grown enormously and getting more complex. Apache Hadoop [1] is the most popular framework in this field, built under multiple layers of the latest technologies supporting big data engine architecture such as Spark, Hive, Mahout, Cassandra, ZooKeeper. This Hadoop itself triggers the renaissance of big data processing and orchestration alongside with advancement of hardware technology in computing.

On the other hand, containerization is now a state of the art in building applications on the cloud, especially with the help of the microservices paradigm, today's web services development become far more flexible than ever. The usage of VM-only machines on the server becomes obsolete. In this paper, we developed the concept of the data center for big data processing usage in a cloud environment and how it is structured for multipurpose computation. The paper will define into architecture requirement section, follow by next section discuss about cloud-based big data management, and conclusion.

2. Big Data Architecture and Requirement

The first paragraph after a heading is not indented (Bodytext style). Gamabox cluster supercomputer has already been around since 2013 while in 2019 it had a few boosts on its technology for implementing the Gamabox OS [2] onto the Kubernetes container orchestration platform [3]. The updates on the system architecture seem to be quite promising, although later challenges need to be conquered to answer more real usages scenario in industries or the academic world. The need to build a big data management platform over Gamabox environment is essential since the aggression of Industrial 4.0 starting to reach multi-dimensional peers of data sources. In that case, strategic planning on building an architecture toward a data center to give the best services to those peers is necessary. Before jumping to the answer of what architecture we use for the system, we need to analyze what the objectives of the missions are first. The objectives of building a data center are described as follows.

As a data center, the first capability that is needed is storing data, although it is not as simple as the usual relational database management system (RDMS) [24], it supposes to be a big data storage management system (BDSMS) that can store and access not only structured databases but also a semi-structured database like JSON files and an unstructured database like media files. Other than that, as a big data management platform, it also needs to store up to Petabytes of storage, which differs from usual RDMS that can only store up to hundreds of Terra bytes, while also receiving data from various sources since big data needs to connect several domains of data to get bigger picture insights needed by users.

Another difference between RDMS and BDSMS is the processing after storage. In RDMS, data stored in the server is usually directly used as it is to various applications requested, meanwhile, in BDSMS that scenario will never happen, since the high dimensional structure of the database needs to be processed before it can be used by the users. For that reason, big data processing is a must for BDSMS. Besides the "extract, transform load" (ETL) process before storing those data on the system, it also needs to be simplified so that messiness of those data would not disturb the process end users need in the end.

The second requirement for the data center is the capability of building models for machine learning (ML) and for deep learning (DL) purposes [25]. On an individual server itself, the DL method for building an AI model needs some specific software and hardware requirements. So, does in a big data center machine. The third requirement is the capability to do high-performance computation stuff for purpose of scientific research. It can be for building simulations for physic, chemistry, biology, sociology, or other high-demand scientific research [26]. The software and hardware requirements are varied for each job. Moreover, sometimes specific task also needs a particular operating system (OS) for that purpose.

3. Cloud-based Big Data Management Platform

If we talk about big data management data centers, we could not ignore the existence of multiple data sources and multiple machine processing [27]. The bigger the data needs more computational resources to process. That means the use of the multi-node server is necessary, moreover, to increase flexibility on build-up applications to process those data, the use of a Virtual Machine is a common answer for that challenge. Virtual Machines can assist developers to host as many systems as the users need in limitation of the hardware availability. Data center for big data processing surely needs this kind of flexibility [28].

However, sometimes those applications need a whole different level of flexibility, in particular for the reason of continuous deployment and continuous integration. In that scenario, the applications need technology that can handle flexibility purposes and also manage the cluster processes as one whole

system. The container technology can be the answer to that big question, but what specific container can manage the whole system would be answered in this next chapter.

In the course of processing big data, capable hardware needs to exist in the system, or it will fail the process. What hardware is needed by the system is enough storage and memory to store all of the data being processed, a capable CPU to run all of the application systems, and maybe needs accelerator for multipurpose jobs in different scenarios. The key is that the hardware needs to be robust enough to process the whole activity in a consistent condition in a long run.

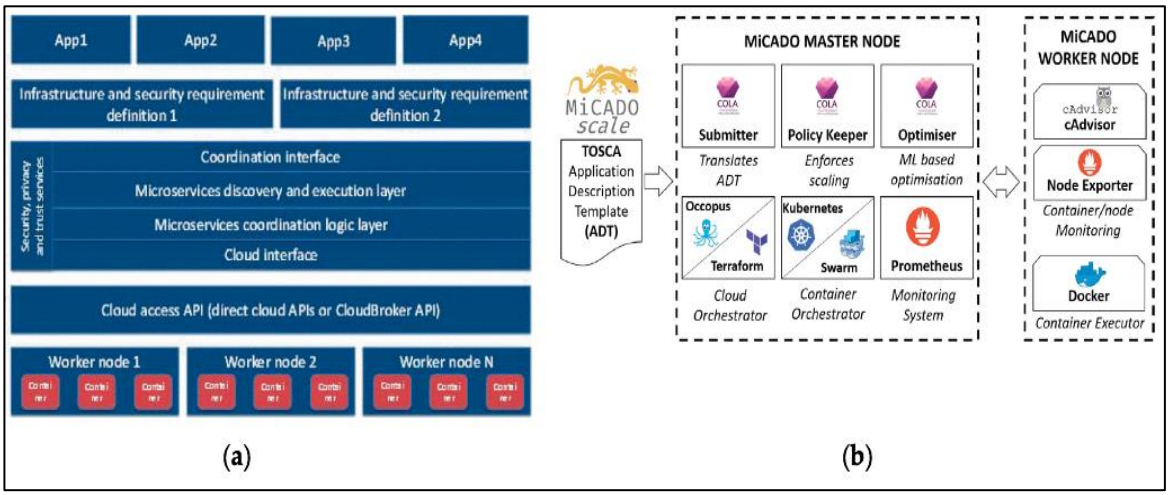

Fig. 1. MiCADO architecture: (a) How original MiCADO was constructed as cloud architecture; (b) Integration and improvement of MiCADO for cloud big data engine using several supporting technologies.

Depending on where the server is constructed, a data center can be built on a cloud service or self-managed in a private network. Each scenario has its pros and cons, depending on what advantage developers need in building the system. It would be better if the system has flexibility in running both public cloud servers as well as on-premise private machines. Whether the choice is fall for either cloud services or an on-promise server, the capability to be controlled remotely is a prime requirement since maintaining the machine can be easier in that way. The server maintainer doesn't have to be always on site if something happened in a sudden condition or if there are needs for deployment or software installation to the machine. There is a challenge when the server is built on-premise, but the local network doesn't allow everything to be publicly exposed. There is a need for a system that can safely access the machine without exposing the network publicly. While the remote access system will require a multi-layer of security to protect the communication, the whole connection also need to be guarded, especially when the data acquisition process is happened to store from all source available to the storage. Besides that, if there is a process that requires communicating data to an outside network, it will also need much more watching so that no possibility of data leaks inside the system happen.

Technium Vol. 5, pp.13-20 (2023)
ISSN: 2668-778X
www.techniumscience.com

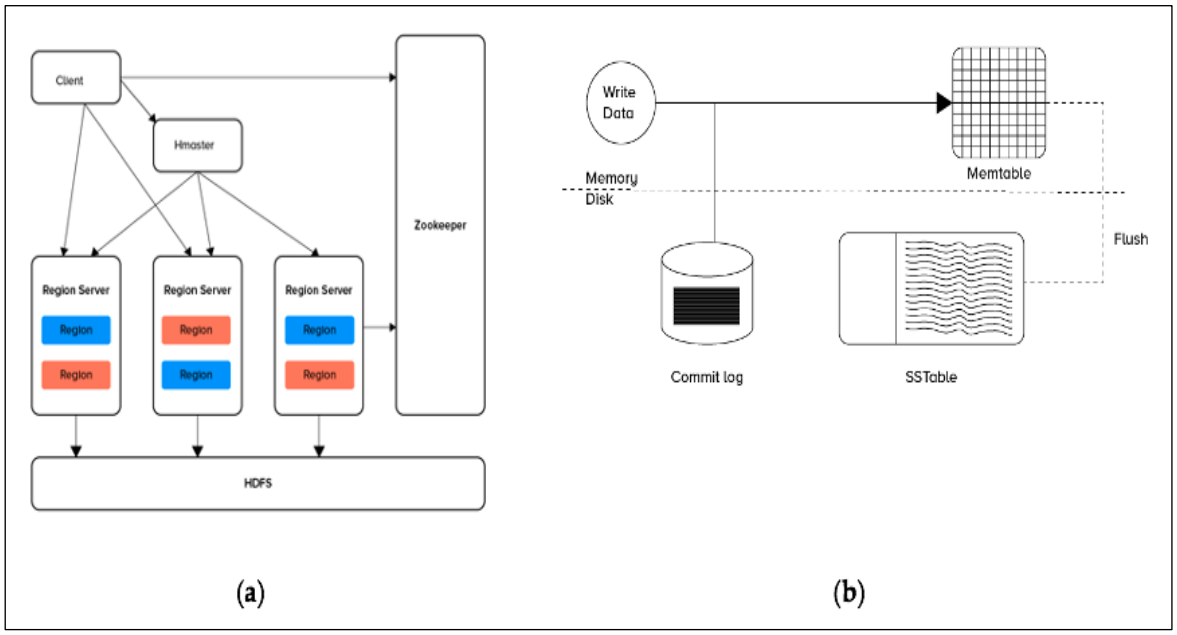

Fig. 2. MiCADO No-SQL facility from Apache: (**a**) HBase architecture; (**b**) Apache Cassandra architecture for No SQL purposes

3.1. Hybrid Architecture

To answer the entire requirements listed above, we need to break down possible architecture postulated by the latest study. Mosa et all [1] offer a platform they call themselves MiCADO (Microservices-based Cloud Application-level Dynamic Orchestrator). MiCADO is an open-source Hadoop-based big data platform with main features have a highly customizable feature to running the multi-cloud orchestration and auto-scaling framework in Docker containers environment, orchestrated by Kubernetes. This platform manages the deployment and an automatic horizontal scaling (in and out) of the underlying cloud infrastructure. Figure 1 shows how MiCADO is built on multi-source technology.

Initially, we need to answer the first requirement which is storage and data access. The current technology that meets this requirement is Hadoop architecture with its satellite application. Hadoop itself is structured by four essential elements as listed in Table 1.

Tabel 1. Hadoop Architecture

Components	**Description**	
	Function	***Components***
Hadoop Common	Common utilities	Modules
Hadoop Distributed File system [8]	Access to data	Distributed file system
Hadoop YARN [9]	Job scheduling and cluster	Resource management
Hadoop MapReduce [10]	Large dataset handling	Parallel processing

3.2. Storage and Big Data Access

These essential elements can be used separately, as a whole, or supported by Hadoop's satellite elements, which are listed in Table 2. Hadoop has HDFS as a tool for managing distributed data sources and

storage used in the system. It does ETL every data source to be orchestrated into the usable-and-linked database. HDFS not only store data in redundant scenarios but also in multiple chunk formats of 128Mb each, making data fault tolerant. While YARN can manage job scheduling and cluster resources in HDFS, it actually can be levelled up by using Spark which has the ability of simpler language (Python, SQL, Scala, Java, or R) on programming distribution storage for ETL, ML, stream processing, and graph computation. Spark can ease developers on accessing SQL DB since YARN builds on top of Java programming language that is more difficult in accessing SQL. No-SQL (Not only Structured Query Language) is one kind of database (DB) type that supports a non-relational database format. This DB type is necessary for a big data platform to store multiple format data like JSON, image, multimedia file, document, and other data that RDMS cannot handle. Hadoop itself has a product for No-SQL needs, the HBase and Apache Cassandra. No-SQL provides simpler and more flexible data management in the storage, increasing the efficiency and performance of the system. Hbase can be used if the data center needs consistency in the large-scale batch processing and MapReduce for it has a direct relation with HDFS. Meanwhile, Cassandra can be used to support. The high availability of large-scale process with advantages on a very minimum setup with less administration overhead, also greater flexibility in CAP theorem trade-offs.

Tabel 2. Storage and Access

Components	**Description**	
	Function	***Components***
Ambari [11]	Provisioning, managing, and monitoring	Monitoring and management
Avro [12]	Data serialization	Data serialization system
Cassandra [13]	Multi-master database	database
Chukwa [14]	Data collection	Large distributed system management
HBase [15]	Data storage with large tables	Structured large tables data storage
Hive [16]	Data summarization and ad-hoc query	Data warehouse
Mahout [17]	Machine learning and data mining	Data library
Ozone [18]	Distributed object store	Object store management
Pig [19]	Data-flow language and execution	Framework
Spark [20]	Expressive programming model	Computing engine
Submarine [21]	Machine learning dan deep learning workload	AI Platform
Tez [22]	Data-flow programming	Framework
ZooKeeper [23]	Distributed application coordination	Coordination services

3.3. Multipurpose Computation

High-performance computation needs both hardware and software support. Hardware configuration would be discussed in the next chapter. While for software support, it can be configured as follows. For scientific simulation in physic, chemistry, or geological research, a software accelerator is essential. It can be openMP, MPI, or MPC for parallelization on the CPU. Meanwhile, some simulations can be accelerated by GPU or equivalent accelerator. For GPU, it can be AMD products or Nvidia's. Each of

them has different software supporting acceleration as if AMD has a ROCm library or Nvidia has CUDA programming to parallelize multi processes.

As for Machine Learning and Deep Learning, for now, the Nvidia ecosystem is still the prime option to run modeling. Performance and compatibility of CUDA programming in a combination of various neural engines like Tensorflow, PyTorch, Keras, or else others are still prime choices for a developer to run their computation. Meanwhile, the AMD ROCm library can handle parallelization in ML but the community support and performance level still have not on par with Nvidia products. The package for implementing ML from Hadoop already exists, which is Mahout, ML and data mining library, or Submarine, a package of AI, ML, and DL in distributed platforms. Some library or software, like geology or meteorology, needs a specific operating system (OS) to be run, such as Windows. Either way, a program such as openMP and MPI only support Linux OS to run seamlessly. In this case, the use of a Virtual Machine or might be Container is fundamental.

For Orchestrating all data that is stored in the system and managing it for further usage, we need a system that is based on dynamically programmable scalability policies. In building BDSMS, two orchestration types exist, they are cloud orchestration and container orchestration. Cloud have capabilities to manage interconnections and between nodes, Cloud have capacities to allocate workloads on public and private cloud infrastructure trough programming techniques [5, 6]. The cloud has several main components such as Control Plane Components (including kube-apiserver, etcd, kube-scheduler, kube-controller-manager, and cloud-controller-manager), Node Components (kubelet, kube-proxy, and container runtime), also several add-ons such as NS, Web UI (dashboard), Container Resource Monitoring, and Cluster-level Logging. Alternative to Kubernetes, we can also use Docker Swarm for container orchestration. The difference between Kubernetes and Docker Swarm is demonstrated in Figure 4. From the above perspective, Virtual machines (VM) and containers are essential. The usage of VM can be implemented for hosting multi-container machines for purpose of building microservices-based application. Docker is a prime example of containerization besides Linux containers (LXD, LXC, LXCS).

High-grade hardware is an absolute answer for big data processing. For storage only, it needs massive storage to be used in this system, although it can be set up as one single machine storage, separate cluster storage, or also can be as distributed storage machine in different places. High-density server-grade processor is ideal, like AMD EPYC 7713P (64c 128t) or 7543 (32c 64t) or Intel Xeon Platinum 8351N

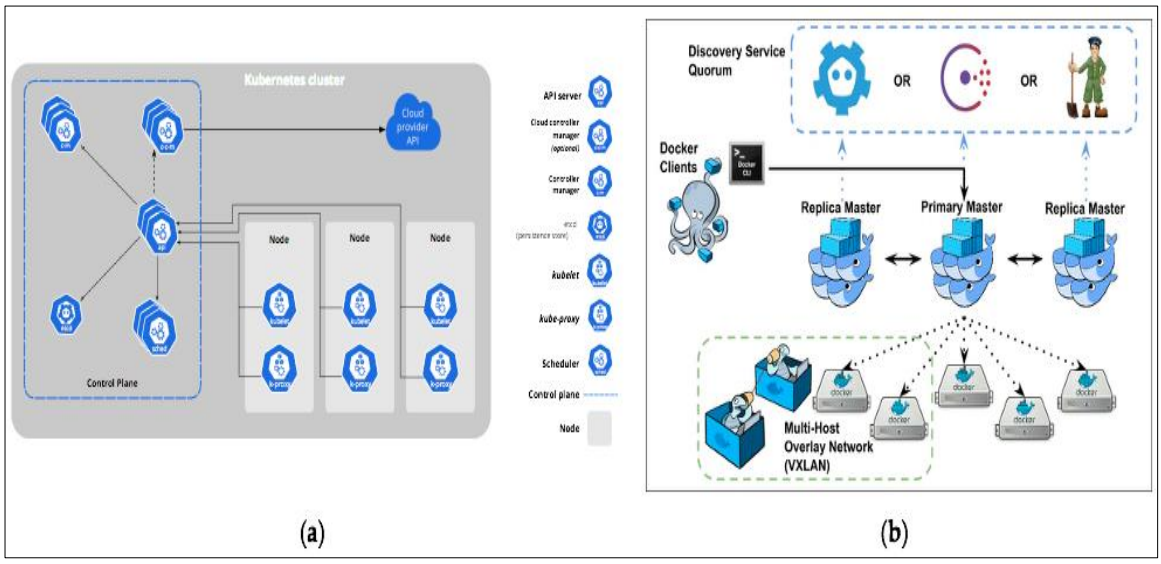

Fig. 3. The Container Orchestrator Architecture in general: (**a**) Kubernetes architecture in building container cluster; (**b**) Docker Swarm as container cluster management architecture.

(36c 72t) or 8380 (40c 80t). Not only that but scalable RAM is also needed for accessibility and capacity to provide multiple VM-container scenarios. Ideally, memory with ECC spec is essential for data corrupt tolerant purposes, though non-ECC still can be implemented but in a minor fraction of the whole system. On cloud services, if the choice is fall to it, remote access is already provided by the provider with several types of security strategies, such as the usage of pem key and Kerberos method. Another way, if the server is on-premise and using a private network, it can use VPN tunneling with Next-Generation Firewall (NGFW) [7] to support both remote access and security. In another scenario when the network is an open network, various methods can be implemented, such as two-factor identification, intrusion prevention, and detection system. Hadoop itself has already adopted Kerberos method to secure its system, so that when HDFS is implemented on the system, Kerberos can be set up to secure the entire communication.

A server accelerator is also needed for multipurpose usage, as mentioned in the chapter before, to run the multi-process program for purpose of scientific computation such as simulation and machine learning. The GPU hardware can be implemented in the system, such as for AMD product AMD Instinct MI200 to support HPC and AI workloads or for Nvidia product Nvidia Tesla A100 for multipurpose high-performance workloads. Combination usage for cloud services and on-premise data centers is easier to deploy with Kubernetes. This configuration can be used for redundancy scenarios and expandability.

4. Conclusion

In this paper, we have discussed how data centers for big data can be built. The usage of Hadoop as central architecture in the system is critical, but deployment with Kubernetes or Docker Swarm also has significant importance when planning for data center architecture. Other than that, the role of Terraform or Occopus also makes the architecture stronger when deployed in a cloud services scenario. Nevertheless, the usage of Kerberos in Hadoop itself as a security layer is also essential in case of avoiding unauthorized attempts into the system, while VPN with NGFW technology can be implemented on the private network server

Acknowledgement

The research is deployed in The Center of Excellence for Big Data Research – Computer System and Network Research Labs, and supported by Department of Computer Science and Electronics, Faculty of Mathematics and Natural Sciences, Universitas Gadjah Mada. The supporting infrastructure has supported by GamaBox and GamaCloud (cloud.wg.ugm.ac.id).

References

[1] A. Mosa, T. Kiss, G. Pierantoni, J. DesLauriers, D. Kagialis, and G. Terstyanszky, Towards a Cloud Native Big Data Platform using MiCADO, in *2020 19th International Symposium on Parallel and Distributed Computing (ISPDC)*, Jul. 2020, pp. 118–125. doi: 10.1109/ISPDC51135.2020.00025.

[2] C. G. N. TAMA, Sistem Operasi untuk Pemrosesan Big Data dengan berbasis Centos 7, Universitas Gadjah Mada, 2017.

[3] M. M. Abror, Implementasi Container dan Kubernetes Orchestration dalam Peluncuran, Pengawasan dan Pengaturan Ekosistem Big Data, Universitas Gadjah Mada, 2019.

[4] T. Kiss, Scalable multi-cloud platform to support industry and scientific applications, in *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, May 2018, pp. 0150–0154. doi: 10.23919/MIPRO.2018.8400029.

[5] What is cloud orchestration (cloud orchestrator)? - Definition from WhatIs.co’,

SearchITOperations. https://searchitoperations.techtarget.com/definition/cloud-orchestrator (accessed Dec. 17, 2021).
[6] What is container orchestration? https://www.redhat.com/en/topics/containers/what-is-container-orchestration (accessed Dec. 17, 2021).
[7] B. Soewito and C. E. Andhika, Next Generation Firewall for Improving Security in Company and IoT Network, in *2019 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, Aug. 2019, pp. 205–209. doi: 10.1109/ISITIA.2019.8937145
[8] Apache Hadoop, https://hadoop.apache.org/ (accessed November 10, 2022).
[9] Yarn, https://yarnpkg.com/ (accessed November 10, 2022).
[10] Map Reduce Tutorial, https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html (accessed November 10, 2022).
[11] Apache Ambari, https://ambari.apache.org/ (accessed November 10, 2022).
[12] Avro, https://binx.io/2018/12/09/apache-avro/ (accessed November 10, 2022).
[13] Apache Cassandra, https://cassandra.apache.org/_/index.html (accessed November 10. 2022).
[14] Apache Chukwa, https://chukwa.apache.org/ (accessed November 2022).
[15] Apache HBase, https://hbase.apache.org/ (accessed November 2022).
[16] Apache Hive, https://hive.apache.org/ (accessed November 2022).
[17] Apache Mahout, https://mahout.apache.org// (accessed November 2022).
[18] Apache Hadoop Ozone, https://hadoop.apache.org/ozone/ (accessed November 2022).
[19] Apache Pig, https://pig.apache.org/ (accessed November 11, 2022).
[20] Apache Spark, https://spark.apache.org/ (accessed November 11, 2022).
[21] Apache Submarine, https://submarine.apache.org/ (accessed November 11, 2022).
[22] Apache Tez, https://tez.apache.org/ (accessed November 11, 2022).
[23] Apache Zookeeper, https://zookeeper.apache.org/ (accessed November 11, 2022).
[24] Belov, V., Nikulchev, E. Analysis of Big Data Storage Tools for Data Lakes based on Apache Hadoop Platform. International Journal of Advanced Computer Science and Applications(IJACSA), Volume 12 Issue 8, 2021, DOI: 10.14569/IJACSA.2021.0120864
[25] Voit, A., Stankus, A., Megomedov, S, Ivanoca, I., Big Data Processing for Full-Text Search and Visualization with Elasticsearch. International Journal of Advanced Computer Science and Applications(IJACSA), Volume 8 Issue 12, 2017. DOI: 10.14569/IJACSA.2017.081211
[26] Acharjya, D.P., Kauser, A.P. A Survey on Big Data Analytics: Challenges, Open Research Issues and Tools. International Journal of Advanced Computer Science and Applications(IJACSA), Volume 7 Issue 2, 2016. DOI: 10.14569/IJACSA.2016.070267
[27] Shah, H., Din, A.u., Abizar, Khan, A., Din, S.u. Enhancing the Quality of Service of Cloud Computing in Big Data Using Virtual Private Network and Firewall in Dense Mode. International Journal of Advanced Computer Science and Applications(IJACSA), Volume 11 Issue 3, 2020. DOI : 10.14569/IJACSA.2020.0110351
[28] Shabbir, A., Abu Bakar, K., Radzi, R.Z.R.M., Siraj, M. Resource Management in Cloud Data Centers. International Journal of Advanced Computer Science and Applications(IJACSA), Volume 9 Issue 10, 2018. DOI: 10.14569/IJACSA.2018.091051