Jurnal Mantik, 5 (2) (2021) 994-1005

Published by:Institute of Computer Science (IOCS)

Jurnal Mantik

Journal homepage: www.iocscience.org/ejournal/index.php/mantik/index

Implementation Parallel Computation for Automatic Clustering

Bayu Dwiyan Satria[1], Ali Ridho Barakbah[2], Amang Sudarsono[3]

[123] Informatic and Computer Department, Electronical Engineering Polytechnic Institute of Surabaya

E-mail: bayudwiyansatria@gmail.com, ridho@pens.ac.id, amang@pens.ac.id

A R T I C L E I N F O

Article history:
Received: 10/06/2021
Revised: 20/06/2021
Accepted: 10/07/2021

Keywords:
Parallel Computing, Clustering, Machine Learning, Bigdata

A B S T R A C T

Developments in the industrial world, especially in the field of computer technology, demand solutions to needs ranging from computing resources, storage media, and communication speeds. This was followed by many new studies in this field, including those related to clustering. Clustering is an exploratory data analysis tool that deals with the task of grouping objects that are similar to each other [1], [2]. Clustering requires computers with high resources, usually the price for computers with high resources, while computers with not too high specifications will be less reliable in handling such large data. Clustering that runs on a single-core processor takes a long time to execute tasks, so parallel computing is needed to speed up computing performance especially at Automatic Clustering. This research will produce faster performance in grouping large data by utilizing parallel computing and automatic clustering methods as methods for grouping data. This technology allows data processing to be carried out in parallel and distributed in hundreds or even thousands of computers, so this technology is very appropriate for processing very large amounts of data)

1. Introduction

Clustering is one of the important tasks for machine learning and big data that's groups items in a dataset into meaningful classes. Clustering Can also be defined as the process of defining mapping. In this case, clusters are a good choice for processing large and large amounts of data. Clustering is commonly used in many fields, such as data mining, pattern recognition, image classification, biology, marketing, urban planning, document search, and so on. Clustering divides the population or data points into groups so that data points in the same group are more similar to other data points in the same group than other groups [1], [3]. Many clustering algorithms have been proposed before such KMeans, Hierarchical Clustering, etc.

Automatic Clustering requires computers with high resources, usually, the price for computers with high resources is not cheap, while computers with not too high specifications will be less reliable in handling such large data. Thus, technology on a large scale related to improving system performance is needed.

In general, the software is built using a serial computing paradigm, in which the software is designed to be executed by a machine that has a Central Processing Unit (CPU) [4]. In serial computing, the problem is solved by a series of instructions that are executed one by one by the CPU, where only one instruction can run at a time. This will raise problems for program execution that require large processor and memory computing resources, namely long execution times, even though several instructions or sets of instructions can be executed simultaneously.

Cluster technology enables organizations to increase their processing power using standard technology or commodity hardware and software components that can be obtained at a relatively low cost [4]. Automatic clustering that runs on a single-core processor takes a long time to execute tasks, so parallel computing is needed to speed up computing performance. The issue of the execution time of automatic clustering will be carried out on 5 nodes, including 1 master node and the rest are worker nodes by clustering these nodes on one. The main purpose of using parallel computing is to shorten the execution time of programs that use serial computing.

At this rate, we use Apache Spark and Apache Hadoop as a platform that supports parallel computing [5]. Apache Hadoop enables distributed processing of large datasets across computer clusters using a programming model designed to scale up computation from one machine to thousands of machines, each of

which offers distributed data computation (Map Reduce) and local storage (Hadoop File System) which is the very large and large storage of data sets [6]. Apache spark is also a general distributed computing based on the Hadoop MapReduce algorithm [7]. Taking advantage of the Hadoop Map Reduce, but unlike Map Reduce, the results and output from Spark can be stored in memory, which is called Memory Computing. Memory Computing increases the efficiency of data computation in that the processing that runs in memory is faster than the processing on the disk

2. Literatur Review

2.1 Parallel Computing

Parallel computing is a computation of a computer in which many computations or processes are performed simultaneously [8]. Big problems can often be divided into smaller problems, which can then be solved at the same time.

Parallel computing is related to concurrent computing, often shared, and often combined, although those are different: it is possible to have parallelism without concurrency such as bit-level parallelism and concurrency without parallelism such as multi-tasking by time-sharing on a single-core CPU. There's is also called High-Performance Computing (HPC) [8].

The parallel computing approach has provided an increase in the speed of the execution time to solve the problem [9]. Parallel computing is a technique of running a program for a process using more than one computing unit. Parallel computing can speed up computer performance by dividing the program into several tasks that can be done separately and then executed in parallel.

There are several different forms of parallel computing: bit-rate, instruction-level, data, and task parallelization [9]. Parallelization is used in high-performance computing, but it has physical constraints and prevents frequency scaling due to power consumption by the computer. Parallel computing has become the dominant paradigm in computer architecture, especially in the form of multi-core processors.

The main purpose of using parallel computing is to shorten the execution time of programs that use serial computing. Some other reasons that make a program use parallel computing include:

a. For large problems, sometimes the existing computing resources are not sufficient to support the resolution of the problem.
b. The existence of non-local resources that can be used over the network or the internet.
c. Savings in hardware procurement costs, by using multiple machines that are modest.
d. There is a limited memory capacity on the machine for serial computing.

The potential for acceleration of an algorithm on a parallel computing platform is given by Amdahl's law [9]. Amdahl's law is a formula used to find the maximum increase possible by repairing a particular part of a system. In parallel computing, Amdahl's law is used to predict the maximum speed for program processing using multiple processors [10].

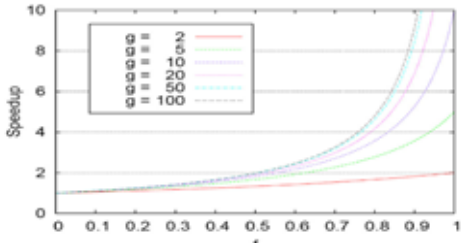


Fig 1. Graphical representation of Amdahl's law

Program acceleration from parallelization is limited by how many programs can be paralleled. Amdahl's law can be formulated in the following ways:

$$S = \frac{T_0}{(1-f)\ x\ T_0 + \left(\frac{f}{g}\right) x\ T_0}$$

Accredited "Rank 4"(Sinta 4), DIKTI, No. 36/E/KPT/2019, December 13th 2019.

$$S = \frac{1}{1 - f + \frac{f}{g}}$$

where T0 = time for unseeded computation; g = highest performance obtained from accelerated computing; f = fraction of the calculation that is not accelerated to be accelerated; S = calculation speed with applied acceleration [7].

Flynn's taxonomy divides parallel computer architectures using the point of view of instructions and data so that there are four types of parallel computer architectures.

Table 1.
Parallel Computer Architecture

SISD (Single Instruction, Single Data)	SIMD (Single Instruction, Multiple Data)
MISD (Multiple Instruction, Single Data)	MIMD (Multiple Instruction, Multiple Data)

a. SISD (Single Instruction, Single Data)
This architecture is an architecture that represents a serial computer, where there is only one processor and one data input stream (memory) so that only one task can be executed at a time. The von-Neumann architecture is of this type [9].

b. SIMD (Single Instruction, Multiple Data)
Execution of instruction will be carried out simultaneously by several processors, where a processor can use data that is different from other processors. Another characteristic of this architecture is the deterministic flow of instruction execution or the state of instructions and data at a time can be easily known. This architecture is suitable for programs that can be divided into tasks that have a high degree of regularity, for example, graphics processing systems [9].

c. MISD (Multiple Instruction, Single Data)
Various instructions will be executed simultaneously by several processors using the same data. This architecture is less popular because there are only a few problems that require solutions using these architectural characteristics. Examples of problems that may require this architecture include multiple frequency filters and encryption programs that use multiple cryptographic algorithms [9].

d. MIMD (Multiple Instruction, Multiple Data)
Instructions can be executed by multiple processors where each processor can use different data. Instruction execution in this architecture can be done synchronously over some time or the number of instructions executed by all processors is the same or asynchronous, deterministic or non-deterministic. In addition, this architecture can do the job according to the characteristics of the three previous architectures [9].

Accredited "Rank 4"(Sinta 4), DIKTI, No. 36/E/KPT/2019, December 13th 2019.

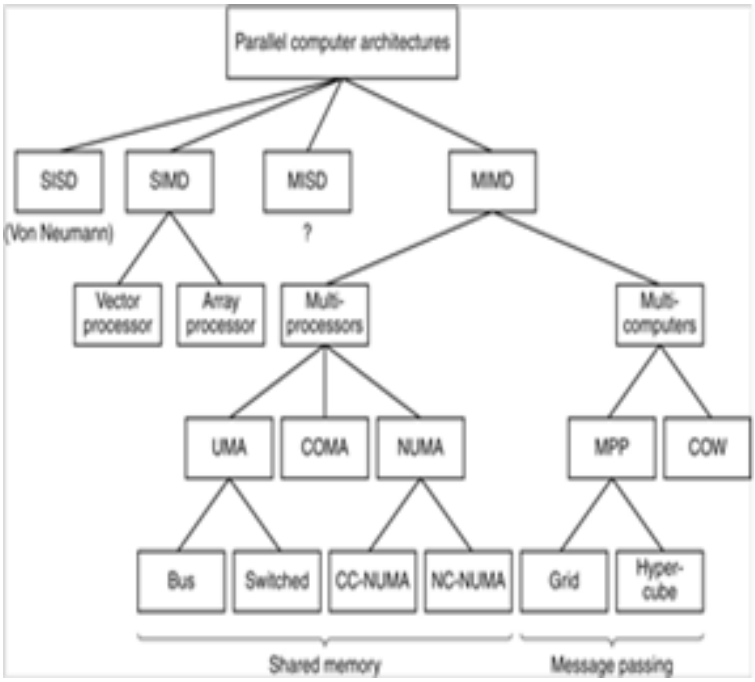


Fig 2. Parallel Computing Architecture

2.2 Clustering

Clustering is the process of grouping so that all members of each partition have similarities based on a specific matrix. A cluster is a group of objects that are joined together because of their similarity or proximity. Clustering is a very useful technique because it will translate an intuitive measure of an equation into a quantitative measure. Clustering is a very useful technique because it will translate an intuitive measure of an equation into a quantitative measure.

2.3 K-means

K-means clustering is one of the popular algorithms for unsupervised data clustering at machine learning. To perform K-means clustering, we must first determine the desired number of K clusters, then the K-means algorithm will assign each observation exactly one of the K clusters.

The K-means algorithm clusters data by trying to separate the samples into n groups with the same variance, minimizing a criterion known as inertia or the number of squares in the cluster. Once the algorithm has been run and the groups are defined, any new data can be easily assigned to the correct group.

KMeans Algorithm describes as follow:

$$J = \sum_{j=1}^{k} \sum_{i}^{n} \left| x_i^{(j)} - c_j \right|^2$$

a. Hierarchical Clustering

Hierarchical Based Clustering is based on data that has more similarities to nearby objects than other data which are further away [1] [2]. Therefore, groups are generated from this algorithm based on the distances between the objects that will be analyzed. The hierarchical clustering model may be divisive in which partitions are constructed from the entire available data set or agglomerate where each partition starts with one point and other points are added into the data set [11]. Euclidean distance is usually used to calculate the distance with formulas:

$$d(x, y) = \sqrt[2]{\sum_{i=1}^{n} | x_i - y_i |^2}$$

2.4 Centroid Based

Centroid Linkage is a clustering process based on the distance between centroids. The centroid-based algorithm creates cluster partitions based on the dissimilarity function, such as those of $c \leq n$ [13]. The main

Accredited "Rank 4"(Sinta 4), DIKTI, No. 36/E/KPT/2019, December 13th 2019.

problem with implementing this algorithm is determining the correct number of clusters for unsupervised data. Therefore, most of the research in clustering analysis has focused on the automation of processes.

Another method is by modifies the k-means algorithm to automatically select the optimal number of clusters by using a G-means algorithm. This method is developed from a hypothesis that the subset of data follows a Gaussian distribution. Thus, the number of clusters increases until each k-means data center is Gaussian. This method requires a standard statistical significance level as a parameter and does not have to set a limit for covariant data.

2.5 High-Performance Computing and Cluster Computing

High-Performance Computing (HPC) is the use of supercomputers and parallel processing techniques to solve complex computing problems. HPC Technology is commonly used for developing parallel processing algorithms and systems by combining parallelization and administrative computing techniques [9].

High-Performance Computing is also used to solve a problem through computer modeling, simulation, and analytics problem. HPC can sustain performance through the use of computing resources, which the parallel computing approach has provided increased execution time speed to solve the problem.

HPC is sometimes used interchangeably. HPC make sure several technologies such as computer architecture, program algorithms, and software/application under a single canopy to solve multiple problems quickly and efficiently. HPC systems are very efficient and require a network with high bandwidth and low latency to connect multiple nodes and clusters.

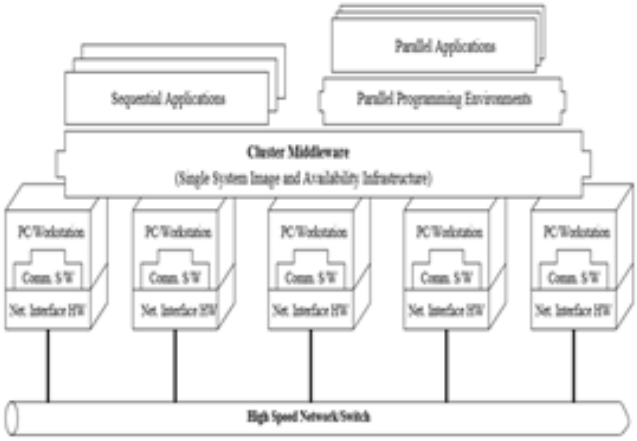


Fig 3. Cluster Computing architecture

Cluster computation that is included in the classification of Distributed Memory Multicomputer or is two or more computers or nodes that are connected into one integrated system capable of solving computational problems simultaneously [8], [14]. Parallel computing requires several nodes to be run simultaneously to execute and process the data to produce faster execution times.

2.6 High Performance Computing Cluster

High-Performance Computing Cluster or Data Analytics Supercomputer is a computationally intensive platform. The HPCC platform incorporates a software architecture implemented on a commodity computing cluster to provide high-performance, parallel processing of data for applications using big data [5].

The HPCC platform includes a system configuration to support parallel batch data processing and high-performance online query applications using indexed data files. The HPCC platform also includes a data center declarative programming language for parallel data processing [8].

Computing Clusters is a distributed processing system, which consists of a collection of interconnected standalone computers that work together as one integrated system computing resource [3]. Compute nodes can be single or multi-processor systems (PC, workstation, or SMP) with memory, operating system, and facilities (I)⁄O. Cluster generally refers to two or more nodes, computers, or machines that are connected. The nodes can be physically separated or at a single cabinet and must connect via a network such as a Local Area Network (LAB) based connection. LAN-based interconnected computer clusters can appear as a single

Accredited "Rank 4"(Sinta 4), DIKTI, No. 36/E/KPT/2019, December 13th 2019.

system for users and applications. Such systems can provide a cost-effective way to get the features and benefits or the fast and reliable service historically only found on more expensive shared memory systems.

2.7 Apache Hadoop and Apache Spark

Apache Hadoop is a framework that enables distributed processing of large datasets across computer clusters using a programming model designed to scale up computation from a single machine to thousands of machines, each offering local computes and storage [9].

Rather than relying on hardware that provides high availability, Apache Hadoop is also designed to detect and handle failures at the application layer. Hadoop runs on commodity hardware, using the Hadoop File System (HDFS) which is very large and large storage of data sets [10]. Hadoop runs MapReduce for distributed data processing and works with both structured and unstructured data and reliably stores files in machine clusters [11]. Hadoop saves each file as a sequence of blocks that have the same size [12]. Hadoop takes a specific part of the computation into several small blocks and distributes it over many computational resources. This is done using MapReduce, which groups and sorts the calculations into smaller chunks and then aggregates the results of the smaller calculations that have been performed.

The Hadoop Distributed File System (HDFS) has a similarity with other distributed file systems but differs in several ways. One of the differences in the write-once-read-many models which loosen concurrency control requirements simplifies data coherence and allows high-throughput access [12]. HDFS consists of a collection of interconnected nodes where files and directories are located. The HDFS cluster consists of a single node that manages the file system namespace and manages client access to files and Data Nodes stores data as blocks in files [9].

Spark is a general distributed computing framework based on the Hadoop MapReduce algorithm. Taking advantage of Hadoop MapReduce, but unlike Hadoop MapReduce, the advanced output of Spark jobs may be stored at the memory, which is called Memory Computing [14]. Memory Computing increases the performance and efficiency of data processing. So, the spark is more suitable for iterative applications, such as Machine Learning and Data Mining.

The Spark API structure provides the same APIs for batch and real-time streaming. The Spark architecture supports integration with many storage solutions in the Hadoop ecosystem such as Hadoop HDFS, MapR XD Distributed File and Object Store, Kafka, HBase, MapR Database JSON, and also Apache Hive.

2.8 Identifying Moving Variance to Make Automatic Clustering for Normal Dataset

This study uses the automatic clustering method with single-linkage hierarchical methods, this method runs by creating a cluster that has the same thing which then continues iteratively. Hierarchical grouping algorithms can be agglomerative (agglomerative) or divisive. The agglomerative method is the process of combining a series of "n" objects together into one group, while the divisive method separates the "n" objects in a row into better groupings. One of the factors of similarity between objects in the hierarchical method is a single link whose similarity is closely related to the smallest distance between objects.

This study conducted experiments with several clustering cases for normal data sets, this method can solve the clustering problem and create separate clusters properly. This method can also avoid the local optimum and find the global optimum.

2.9 Determining Constraints of Moving Variance to Find Global Optimum and Automatic Clustering

This study proposed a new approach to finding the global optimum of clustering by analyzing the moving variance of the cluster at each stage of cluster construction. This research use Valley Tracing and Hill-Climbing to find the global optimum. This study also uses automatic clustering with single-linkage hierarchical methods (SLHM).

Based on this research the global optimum usually resides in the stages that have far different values from its next stage. To find the global optimum this method considers climbing the hill for each cluster construction. It has altitude value to determine the global optimum as described with:

$$V\_(i+1) > \alpha . V\_i$$

Where α is the altitude value.

For the Valley-tracing the possible global optimum can be determined by trace the valley moving variance and describe as:

$$(V\_(i-1) \geq V\_i) \cap (V\_(i+1) > V\_i$$

Optimization of High-Performance Computing Cluster based on Intel MIC

Accredited "Rank 4"(Sinta 4), DIKTI, No. 36/E/KPT/2019, December 13th 2019.

This study focuses on theoretical analysis, computational testing, and optimization of the High-Performance Computing Cluster (HPCC) [6]. HPCC is built on Intel Many Integrated Core (MIC) for High-Performance Linpack (HPL) testing. The platform is configured by 5 nodes with an Intel Xeon Phi ™ Co-processor 31S1P per node to analyze the power consumption and parallel level of the Intel MIC accelerator and compiled with the Passing Interface Message (MPI) Library and Math Kernel Library (MKL). The author modifies the Make file and is debugged by adjusting its parameters. According to some evaluations of InfiniBand nodes, and the Iibhpl library from Intel is used and the NB value is set to 960 for one node with one MIC when debugging. In addition, in this study, the algorithm for the Double Precision General Matrix Multiplication (DGEMM) and PTRANS tests was optimized to obtain efficient and precise working time duration.

3. Reaserch Methods

To process very large data in a distributed manner and run-on clusters consisting of several connected computers, we use the Hadoop and Spark software framework. This technology allows data processing to be carried out in parallel and distributed on hundreds or even thousands of computers, so this technology is very appropriate for processing very large amounts of data, the system to be created will also take advantage of Map Reduce. The use of the Apache Spark platform is for fast data processing that can work faster than Apache Hadoop itself, while the Apache Hadoop is used for data storage by utilizing HDFS features from Apache Hadoop.

The data received at the master node will be instructed to the worker node by parallelizing the tasks into several pieces of many worker nodes to speed up the processing time. Map Reduce is in charge of dividing large data into smaller chunks and organizing them into tuples for parallel processing. Map Reduce which consists of the map stage functions to process input data which is generally in the form of files stored in HDFS and then the shuffle and reduce stages are combined into one stage, namely the reduce stage which processes input data from the map process results, which is then carried out in the shuffle stage and reduce which results in the new data set being stored in HDFS again.

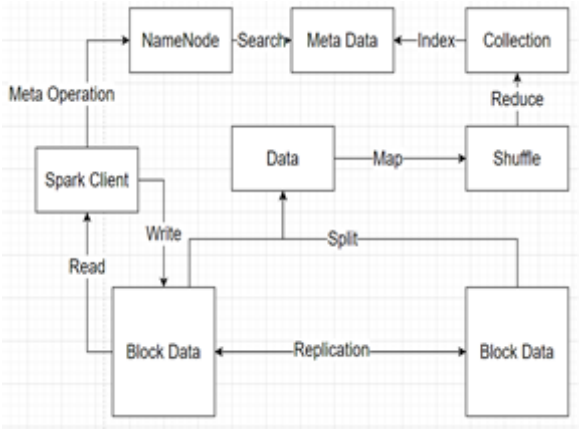


Fig 4. Hadoop and Spark processing

The automatic clustering method performs optimal searches by constructing an explicit probabilistic sampling model to find solutions. Optimization is seen as a series of additional updates to the probabilistic model, starting with a model that provides an acceptable distributed solution and ending with a model that produces only global optima.

At this part we use Euclidian Distance to calculate distance between a point which describe in a formula:

$$d(x,y) = \sqrt[2]{\sum_{i=1}^{n} | x_i - y_i |^2}$$

Accredited "Rank 4"(Sinta 4), DIKTI, No. 36/E/KPT/2019, December 13th 2019.

Algorithms of this type provide different methods of finding groups in data. The density of a cluster can be determined by variance within-cluster (Vw) and variance between cluster (Vb) [1]. The variant of each stage of cluster formation can be calculated using the formula:

$$V_c^2 = \frac{1}{n_c - 1}\sum_{i=1}^{n}(y_i - \bar{y}_c)^2$$

Where n_c is the total number of each cluster, y_i is the point of data and y¯_c is the centroid of the cluster. Furthermore, from the variance value above, we can calculate the variance within-cluster (Vw) with the formula:

$$V_w^2 = \frac{1}{N - c}\sum_{i=1}^{c}(n_i - 1)\,V_c^2$$

Where N = Sum of all data; c = number of clusters; ni = Number of cluster member i; ⟦V_c⟧^2 = Variant in cluster i.
And the variance value between clusters (Vb) with the formula:

$$V_b^2 = \frac{1}{c - 1}\sum_{i=1}^{c} n_i(\bar{y}_i - \bar{y})^2$$

Where (y)¯= average of y¯_i; c = number of clusters; ni = Number of cluster member i, which the ideal cluster has a minimum variance within-cluster (V_w) which represents internal homogeneity and maximum variance between clusters.

4. Research Results and Discussion

In this section, we discuss the experiment and the result of the research the data used is the result generated data with a total data of 10,000 to 1,000,000 data. This data consists of 2 attributes with the file format .csv. We serialization of data since it's important in the performance of any distributed application. Incorrect format causes object serialization to consume too many bytes, which slows down calculations considerably.

At this research we use several components as follows:

Table 2.
Hardware Specification

Component	Master Node	Worker Nodes
Architecture	x86_64	x86_64
Number of Nodes	1	4
Processor	Intel(R) Core (TM) i7-7567U CPU @ 3.50GHz	Intel(R) Core (TM) i7-7567U CPU @ 3.50GHz
CPU Cores	4	16
Base frequently	3.5 GHz	3.5 GHz
Turbo frequently	4.0 GHz	4.0 GHz
Memory	4 GB – 1 x 4 GB DDR4 2,666 MHz RDIMM	16 GB – 4 x 4 GB DDR4 2,666 MHz RDIMM
OS Drive	WD 500GB 7200 Rpm SATA 6Gb/s 32 MB Cache 2.5	WD 500GB 7200 Rpm SATA 6Gb/s 32 MB Cache 2.5

In the hardware specification table above, some components are separated into two in the machine specification that is carried out, which consists of a master node and a worker node. Master nodes and worker nodes use the same types of computers and components from each other. But in this case, it is separated to make it easier to identify the hardware specification.

Otherwise, the configuration of the software and services running on the computer also greatly affects the computational ability, for that in the tuning environment process it is necessary to pay attention to what services are running on the computer user so that it does not interfere with the computation process. In this case, the computer uses a minimal operating system installation so that no services interfere with computing operations. For the specifications of the services and software that are prepared like the following table

Accredited "Rank 4"(Sinta 4), DIKTI, No. 36/E/KPT/2019, December 13th 2019.

Table 3.
Software Specification

Component	Management Node
OS	CentOS Linux Release 7 (Core)
Kernel	3.10.0-1062.4.1.el7.x86_64
Java Development Kit (JDK)	OpenJDK 1.8.0_232 64-bit Server VM
Apache Spark	2.4.4e
Apache Hadoop	Hadoop v3.2.1

Apache Spark is a programming environment designed for parallel computation in groups that will be used in this experiment. This environment allows shared-memory and distributed-memory execution across parallel applications. All systems exhibit different characteristics and have aggregative requirements. To handle parallel jobs running on top of the system.

Environment Spark uses Java serialization by default. Spark serializes objects using the Java Object Output Stream framework which can work with any class created by implementing java.io. Serializable. Spark manages data using partitions that help parallel distributed data processing with minimal network traffic to send data between workers. By default, Spark tries to read data to the RDD from the node closest to it. Because Spark accesses partitioned data, to optimize the transformation operation, spark creates a partition to hold chunks of data.

The process for spark uses a job progress listener and an RDD operation graph listener to compute metrics for the tasks performed in the spark environment. Spark uses the job progress listener stageIdToData registry to be able to access stages according to the given task.

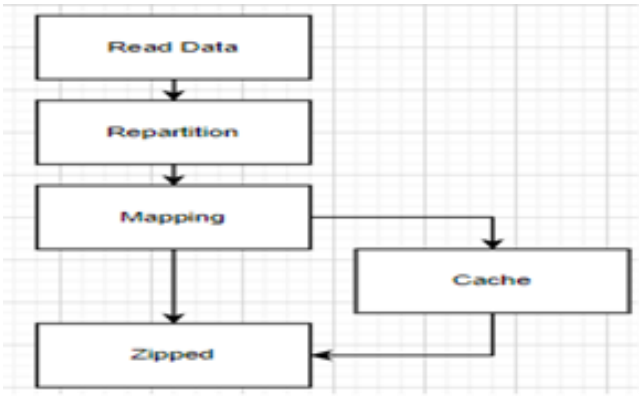


Fig 5. Spark DAG Scheduler

As is known, Spark RDD is a very large collection of various data, so it cannot enter into one node and must first be partitioned into various nodes. It is when the partitioning process takes place that coalesce and repartition, two methods that allow repartitioning data for a certain number of partitions. Coalesce is a method for partitioning data in a data frame, it is used to reduce the number of partitions in a data frame. Unlike repartitioning, coalesce does not shuffle to create partitions. Spark partitions using 2 partitions by default. With some conditions, it will greatly affect data processing time, therefore the partition determination needs to be adjusted optimally.

The experiments were carried out using multiple partitions and durations of computation is shown as follow:

Table 4.
Partition Calculation Result

Total Data	Partition	Duration
1,000,000	1	3.7 min
1,000,000	2 (default)	2.2 min
1,000,000	3	1.9 min
1,000,000	6	1.3 min

Accredited "Rank 4"(Sinta 4), DIKTI, No. 36/E/KPT/2019, December 13th 2019.

In addition, the number of partitions that are created will greatly affect the speed of data processing, for that it needs to be optimized for the data partition. To improve computational capabilities in the experiment, we utilizing the Intel Math Kernel Library (MKL) for the matrix multiplication capabilities that exist in the clustering algorithm. When examining performance and compute capability, several factors greatly influence how much capacity is required for the hardware configuration to support an application. The hardware capacity required to support an application depends on the application specifications and configuration.

The data and computation in this research are divided into several scenarios to be processed which:

Table 5.
Total Data

Total Attribute	Total Data
2	10,000
2	100,000
2	1,000,000

Each data will be processed into multiple computation scenarios:

Table 6.
Unit Computation

Number of Nodes	Cores	Memory (GB)
1	2	4
2	4	8
3	6	12
4	8	16
5	10	20

Spark takes data that has been prepared on HDFS, where Spark will access HDFS under the permissions that have been given to the Hadoop File System. Hadoop then distributes the data set according to the number of replications set in core.xml. Of course, Spark will access HDFS installed on the machine as opposed to accessing a shared file system which requires a large amount of throughput and bandwidth to be able to access data stored in the shared file system.

The cluster ID is used to be able to connect between the nodes in the cluster. If the cluster-ID between nodes is different, then it will not be able to distribute the dataset to other machines. When Name Node is started, Hadoop will be in safe mode, write operations cannot be performed on HDFS. But Hadoop can still read data or register files stored in HDFS.

At this time, we also modifying Transparent Huge Pages (THP) and Enabling Jumbo Frame. Linux has 2 types of huge pages, transparent huge pages, and explicit huge pages. The difference between the two is in the Linux kernel's own transparent huge pages which do what process settings and which page size is used. The CPU has a built-in memory management unit and Memory is managed in blocks known as pages which are made up of 4096 bytes. 1MB memory equals 256 pages, while 1GB memory equals 256,000 pages and so on.

Because THP is not good for data system performance, THP should be deactivated by:

```
$ echo never > \ /sys/kernel/mm/transparent_hugepage/defrag
$ echo never > \
/sys/kernel/mm/transparent_hugepage/enable
$ swapoff -a
    $/sys/devices/system/cpu/cpu*/cpufreq/scalling_governor
```

Enabling the Jumbo Frames, the operating system uses the maximum transmission unit (MTU) to control the maximum size of packets or frames sent over TCP. By default, the MTU is set to 1500 but the

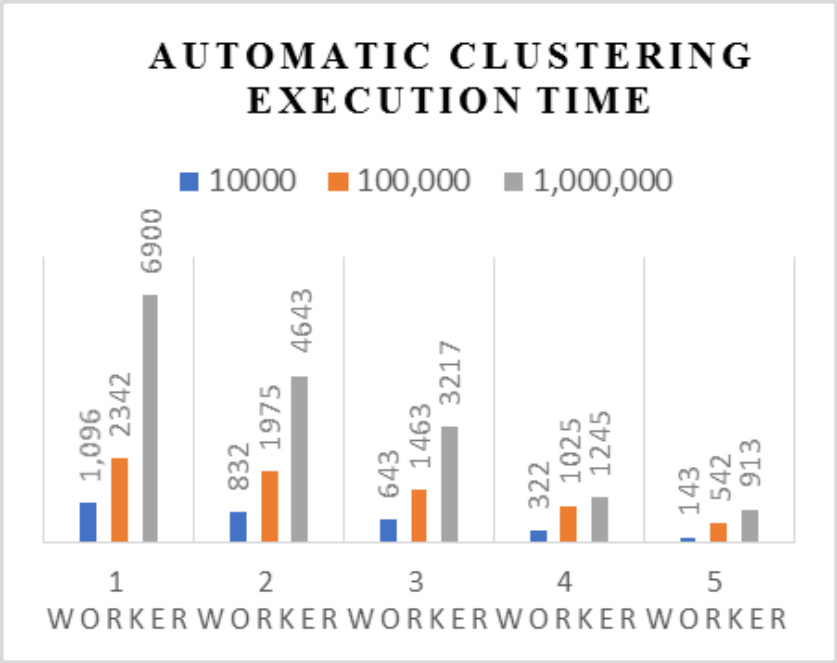
Accredited "Rank 4"(Sinta 4), DIKTI, No. 36/E/KPT/2019, December 13th 2019.

value can be adjusted upwards to a maximum of 9000. When the MTU value is greater than the default value, it is called "Jumbo Frames". To change the MTU value, add MTU = 9000 to:

```
$ / etc / sysconfig / network-scripts / ifcfg-eth0
```

Using the configuration according to the previous optimization, clustering is sent in the spark cluster at these rates we came to the result of the process performed on the spark cluster get the following results:

Table 7.
Cluster Execution time Analysis



Based on the resulting computation time, there is also an increase in the speed of each additional worker in each clustering process, meaning that the computing process also depends on the data processed by the executor which map reduce significantly can reduce the processing time around 45% in the spark environment. Also, perform the jumbo frame give an improvement at the execution since data need to distribute across all nodes in the Spark cluster. However, since automatic clustering compute based on the k that is reduced on each stage of cluster construction it will take more time to perform the cluster analytics.

5. Conclusion

According to the experiment and the results, the automatic clustering that we run on multiple nodes is faster than running by utilizing parallel computing to accelerate performance in processing data. By adding features such as the Intel Math Kernel Library and modifying the configuration of spark, Automatic clustering as a method for grouping data can run faster.

This spark technology allows data processing to be carried out in parallel and distributed across hundreds or even thousands of computers, so this technology is very appropriate for processing very large amounts of data. The modifications to this spark environment by adding the function of automatic clustering which is to calculate cluster variance, the variance within clusters, the variance between clusters, and variance are useful for finding the global optimum.

Accredited "Rank 4"(Sinta 4), DIKTI, No. 36/E/KPT/2019, December 13th 2019.

6. References

[1] A. R. Barakbah and K. Arai, "Identifying moving variance to make automatic clustering for normal data set," in IECI Japan Workshop, 2004, pp. 26–30.

[2] R. Edelani, A. R. Barakbah, T. Harsono, and A. Sudarsono, "Association analysis of earthquake distribution in Indonesia for spatial risk mapping," Proc. - Int. Electron. Symp. Knowl. Creat. Intell. Comput. IES-KCIC 2017, vol. 2017-Janua, pp. 231–238, 2017, doi: 10.1109/KCIC.2017.8228592.

[3] A. R. Barakbah and K. Arai, "Determining Constraints Of Moving Variance to Find Global Optimum and Make Automatic Clustering," in IECI Japan Workshop, 2004, pp. 409–413.

[4] M. Bakery and R. K. Buyya, "Cluster computing at a glance," in High-Performance Cluster Computing: …, 1st ed., Prentice Hall PTR, 1999, pp. 3–47.

[5] K. Sharmila, S. Kamalakkannan, R. Devi, and C. Shanthi, "Big data analysis using apache Hadoop and spark," Int. J. Recent Technol. Eng., vol. 8, no. 2, pp. 167–170, 2019, doi: 10.35940/ijrte.A2128.078219.

[6] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop distributed file system," in 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies, MSST2010, May 2010, pp. 1–10, doi: 10.1109/MSST.2010.5496972.

[7] "Apache SparkTM - Unified Analytics Engine for Big Data." https://spark.apache.org/ (accessed Jan. 02, 2020).

[8] T. Sterling, M. Anderson, and M. Brodowicz, "Introduction," in High Performance Computing, Elsevier, 2018, pp. 1–42.

[9] T. Sterling, M. Anderson, and M. Brodowicz, "HPC Architecture 1," in High Performance Computing, Elsevier, 2018, pp. 43–82.

[10] S. Xu, Z. Wu, H. Yujing, Q. Xue, S. Liao, and B. Liu, "Optimization of High Performance Computing Cluster based on Intel MIC," in 2016 2nd IEEE International Conference on Computer and Communications, ICCC 2016 - Proceedings, Oct. 2017, pp. 1028–1033, doi: 10.1109/CompComm.2016.7924860.

[11] P.Baby and K.Sasirekha, "Agglomerative Hierarchical Clustering Algorithm- A Review," Int. J. Sci. Res. Publ., vol. 3, no. 3, pp. 2–4, 2013.

[12] V. Marinova–Boncheva, "Using the Agglomerative Method of Hierarchical Clustering as a Data Mining Tool in Capital Market," BulDML Inst. Math. Informatics, vol. 15, no. 4, pp. 382–386, 2008.

[13] L. Chen, S. Wang, and X. Yan, "Centroid-based clustering for graph datasets," Proc. - Int. Conf. Pattern Recognit., no. Icpr, pp. 2144–2147, 2012.

[14] A. M. Pfalzgraf and J. A. Driscoll, "A low-cost computer cluster for high-performance computing education," IEEE Int. Conf. Electro Inf. Technol., pp. 362–366, 2014, doi: 10.1109/EIT.2014.6871791.