Journal of Computers for Society

Journal homepage: https://ejournal.upi.edu/index.php/JCS

Apache Spark Implementation on Algorithms Boyer-Moore Horspool for Case Studies Internal Transcribed Spacer and Restriction Enzyme

Fidela Zhafirah, Topik Hidayat, Lala Septem Riza*

Faculty of Mathematics and Natural Sciences Education, Universitas Pendidikan Indonesia, Indonesia
*Correspondence: E-mail: fidela.zhafirah@student.upi.edu

ABSTRACT

The huge increase in the amount of data is a problem today. The increase in large amounts of data makes storage very large and processing data becomes very long. Meanwhile, the speed of the process is very necessary to streamline time. This research is dedicated to solving storage and process problems as a big data processing solution by creating a string matching computational model using the Boyer-Moore Horspool algorithm using the Big Data platform, Apache Spark where the Hadoop Distributed File System as data storage on the cluster. In this study, a comparison of string matching process time between stand-alone, the use of Apache Spark single nodes, the use of Apache Spark 3 nodes, 5 nodes, 11 nodes and 16 nodes using Hadoop Distributed File System storage on clusters on Google Cloud Platform. The case study used is bioinformatics by solving two problems in the field of biology, namely the search for motives related to determining the group of flowering plants with other plant groups and the search for motives as detection of begomovirous symptoms as the cause of curly leaf disease. In the results of the study, insignificant time was obtained because the data used could still be processed by classical programs so that the execution time was not much different. The accuracy of the program run on Apache Spark is 83.5%.

ARTICLE INFO

Article History:
Submitted/Received 2 Mar 2024
First Revised 25 Apr 2024
Accepted 06 Jun 2024
First Available Online 07 Jun 2024
Publication Date 15 Jun 2024

Keyword:
Apache Spark,
Big data,
Bioinformatics.

1. INTRODUCTION

The emergence of new technologies, devices and means of communication such as social networking sites, causes the amount of data generated by humanity to grow rapidly every year (Salehan & Negahban, 2013). The amount of data generated from the beginning of time until 2003 was 5 billion gigabytes. If someone stacked data on disks, it might fill an entire football field. The same amount was created every two days in 2011, and every ten minutes in 2013. This rate is still growing enormously. Even though all this generated information is meaningful and can be useful when processed, it is being ignored.

Since the popularity of the Internet is one of the main reasons for the rapid growth of communication and connectivity in the world, we see the emergence of Big Data platforms in the Internet environment. Every day, 2.5 billion bytes of data are created and 90 percent of the data in the world today was produced in the last two years (Devakunchari, 2014). The ability to generate data has never been so powerful and immense since the invention of information technology in the early 19th century. As another example, on October 4, 2012, the first presidential debate between President Barack Obama and Governor Mitt Romney sparked more than 10 million tweets within 2 hours. Among all these tweets, the specific moments that generated the most discussion actually expressed public interest, such as discussions about Medicare and vouchers. Such online discussions provide a new way to sense public interest and generate feedback in real time, and are largely engaging compared to generic media, such as radio or TV broadcasts. Another example is Flickr, a public photo sharing site, which received 1.8 million photos per day, on average, from February to March 2012. Assuming the size of each photo is 2 megabytes (MB), this requires 3.6 terabytes (TB). storage every day. Indeed, as the old adage states: “a picture is worth a thousand words,” the billions of images on Flicker are a treasure tank for us to explore human society, social events, public affairs, disasters, and so on, only if we have the power to utilize a large number data (Wu *et al*., 2013).

Data is an important commodity today, especially for biological researchers. How could it not be, ten years ago it took a researcher three years to find a gene involved in a disease. But now, thanks to the existence of genome information stored in large databases where the public can obtain that data, the same task can possibly be done in just half an hour (Derrien *et al*., 2012).

There are many projects that will be carried out in 2025 and a comparison will be made of the 3 big Big Data generators such as astronomy, YouTube and Twitter. From this comparison, it is estimated that genomics will make many demands on the needs of the analysis domain. Therefore, it is necessary to develop technology that can be used to help deal with this genomic problem. This is what makes ganomics something that science in the field of Big Data pays attention to (Stephens *et al*., 2015).

String matching is the main task used in many applications such as signature-based malware detection, computational biology, web search engines, and several other applications. This is the process of finding all sub-strings S in a text sequence T. The Boyer-Moore algorithm is one of the most popular because it can handle increasingly complex problems when dealing with large search data (Hoong & Ameedeen, 2017).

This research will use a variation of the Boyer-Moore Algorithm, namely the Boyer-Moore Horspool Algorithm. As previously conducted, research in Tang *et al*. (2020) is a reference for research carried out by carrying out several innovations, especially in the use of equivalent string matching algorithms and developments related to new technology which is currently still developing, namely Big Data, especially Apache Spark.

In recent years, the scientific world has developed a lot of mapping of the human genome, even now starting to map genomes for other organisms. The analysis of emerging genomic sequence data and genome projects of humans and other organisms is an important achievement for bioinformatics (Bayat, 2002).

There are two cases in the field of biology that require computational completion of string matching, namely the case of string matching a pattern or motif in plant DNA to distinguish between flowering plant groups and other plant groups such as ferns, mosses, fungi and algae (Hidayat *et al* ., 2016). Then there was the case of melon cultivation which became an obstacle due to the presence of a virus with Begomovirous symptoms which resulted in melon plants being infected with curly leaf disease (Wilisiani *et al*., 2014).

Table 1. Label describing the table in short.

Reinforcement Condition	Loss of Mass (g)
Plain – Gl	1.38
Plain - Adva	1.23
Carbon microfiber, 0.24 vol%-(CMF-0.24) - Gl	0.98
Carbon microfiber, 0.24 vol%-(CMF-0.24) - Adva	0.93
Carbon microfiber, 0.48 vol%-(CMF-0.48) - Gl	0.93
Carbon microfiber, 0.48 vol%-(CMF-0.48) - Adva	0.92
Carbon microfiber, 0.96 vol%-(CMF-0.96) - Gl	0.85

2. METHODS

2.1. Computing Models

In implementing the use of Apache Spark with the Boyer-Moore Horspool Algorithm, the author designed a model to run the Big Data computing concept as presented in **Figure 1** below.

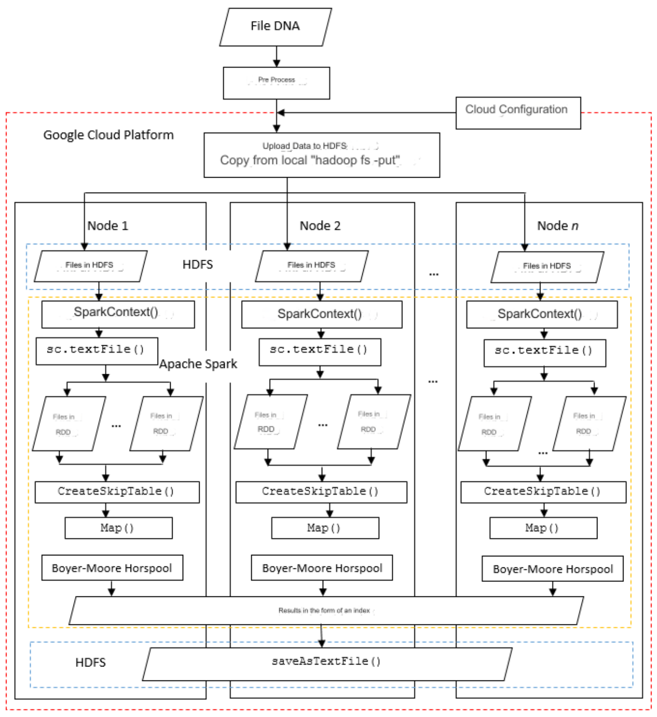


Figure 1. Computational model for implementing the Boyer-Moore Horspool algorithm on Apache Spark.

2.1.1. Pra proccess

Data files that will be used in research will go through a preprocessing stage to produce clean sequence strings. Previously, the contents of the file contained data information, whereas in this research it will only require the sequences contained in it. The sequence in question is data on the base pair letters 'A', 'C', 'G', 'T' and 'N' as a notation for the cryptic DNA in the contents of the file. An example of the content of sequence data that will be preprocessed is as follows.

>gi|1149526168|ref|NW_017972257.1| Asparagus officinalis chromosome 1 genomic scaffold, Aspof.V1 AsparagusV1_01, whole genome shotgun sequence

GAGAGGAGAGGTTTTTAGTTTTTAGGATAGCGAGGTTTGTATTTGAACCTCTCCCTATATATATAT ATAC

ATATATTATATATATACATATATACATATATTATATATATATACATATATTATACATATAAGAAAAT AAA

AAAATACTATTCCGGGCCGGGTCTGGGCCCGGGCCGGGCGCCAACGACCGGACCCGTGTCTGGC CCGGGT

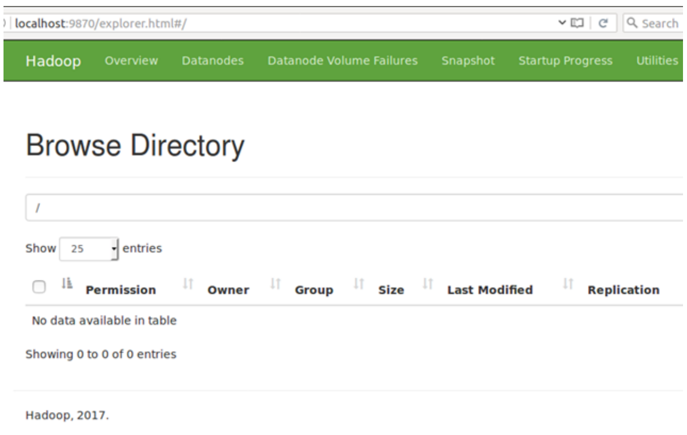
//

After preprocessing is carried out on the DNA data file as previously explained, the preprocessing results of the data above will become a sequence string that is ready to be used in the main process, namely "GAGAGGAGAGGTTTTTAGTTTTT...". Likewise, data with a large number of base pairs will result in strings that can reach hundreds of millions in length. Then the file will be saved in a different format to the original data, namely .txt with sequential naming, namely data1.txt, data2.txt, ..., datan.txt.

2.1.2. Upload Data to HDFS

The computing process that will be carried out is entirely carried out in cloud computing, especially in this research using the Google Cloud Platform. So the cloud is prepared according to the computing required in research, namely installed with Apache Hadoop and Apache Spark as well as the programming language used in the computing process. Because the computing process is carried out on Google Cloud Platform, all data needs to be uploaded to the cloud. Uploads can be done using the command prompt as a link between the user's computer and the cloud.

The first stage in implementing Big Data using Apache Spark is that the DNA data is ready to be uploaded to HDFS which is a feature in Apache Hadoop as a data distributor. When DNA data is uploaded to HDFS, large data will be cut into smaller sizes, namely with a maximum size of 128MB for this research or can be adjusted to suit needs when installing Apache Hadoop. The cuts made by HDFS will not be visible to the naked eye. This is the advantage of a distributed system where the user does not need to think about the correct string cutting method. Data uploaded to HDFS is distributed and replicated as many times as specified in the Apache Hadoop installation settings. In this research, 3 replications will be carried out. To upload files to HDFS, use the hdfs -put command in the terminal. HDFS files can be seen in the UI provided by Apache Hadoop for users to access HDFS data so they can download all files, head files, tail files, delete files, upload files (usually resulting in the file owner becoming dr.who and it is feared that there will be problems regarding reading permissions file) and check the existing directories on HDFS. The port used is: 9870 on localhost and cluster. The appearance of Apache Hadoop for HDFS on port :9870 can be seen in **Figure 2.**



Browse Directory

/

Show 25 entries

	Permission	Owner	Group	Size	Last Modified	Replication
No data available in table						

Showing 0 to 0 of 0 entries

Hadoop, 2017.

Figure 2. HDFS user interface display.

2.1.3. Files in HDFS

After ensuring that the data is in HDFS, the data needs to be retrieved for use by Apache Spark so that it can be processed to the next stage. So you need to know the directory where the files on HDFS are located at the time of the call. HDFS file directory links vary depending on where the Apache Hadoop platform is used. In this research, Google Cloud Platform is used, therefore calling data on HDFS is by using the link format hdfs://[cluster_name]/user/[user_name]/[folder]/[file]. As for the local one, use port: 9000.

2.1.4. SparkContext()

In this research, all configurations to start working with Apache Spark are in program code created using the Python language, namely a special shell created for Apache Spark called pyspark (Tang *et al*., 2020). Once the shell has been selected, use pyspark, which means all processing will be carried out in Apache Spark. The next step after the initial 'gate' in using Apache Spark is to call the python library, namely from pyspark which contains SparkContext().

2.1.5. Sc.textFile()

After the initial gate for using Apache Spark is called, namely SparkContext(), the next step is to call the data in HDFS for processing. In this process, text file data in HDFS is taken and converted into RDD objects. In this process RDD takes over data from HDFS which has been split into small blocks so that it can be processed in Apache Spark.

2.1.6. Files in RDD

RDD will act as an object where there are several functions that can be performed. The two types of functions in RDD are transformation and action (Villar *et al*., 2021). All processes in Spark are combined using transformation functions both in looping or iteration, mapping and other engineering. This is so that the core processes remain within Apache Spark as the main function of Apache Spark is to run processes in parallel. Meanwhile, the action category function is used to change RDD objects into data or other variables.

2.1.7. Map()

After the file to be processed is in the form of an RDD object, basically the object is in one unit. So if it is processed directly, the process will take place sequentially. Therefore, here the transformation map() function from the Apache Spark platform will be used as a data solver or distribution where the Boyer-Moore Horspool algorithm function will be distributed to each block.

2.1.8. CreateSkipTable()

Before the matching or searching process is carried out, the first step in using the Boyer-Moore Horspool algorithm is to calculate a skip table containing values which will later be used in index shifts when matching or searching strings. The formula used to fill in the values in this skip table is the following formula. See **Eq. 1**.

$$\text{value}=\text{length}-\text{index}-1 \tag{1}$$

From **Eq. 1** the value will be used when calling the Boyer-Moore Horspool algorithm function in the parameter. Shifts in the matching or search process depend on the values created in the createSkipTable() function.

2.1.9. Use of the Boyer-Moore Horspool Algorithm

In the Big Data concept, data that has been broken down into smaller blocks is maintained in Apache Spark (Kadkhodaei *et al*.,2021). So the Boyer-Moore Horspool algorithm that has been created is sent to each data block. Therefore, the entire string matching process using the Boyer-Moore Horspool algorithm is carried out in Spark by carrying out transformations in the RDD so that the result will be an index with an object in the form of an RDD. The Boyer-Moore Horspool algorithm is used to look for repetitions up to the end of the sequence. So it is possible for the output index to have more than one.

2.1.10. SaveAsTextFile()

The output results from the Boyer-Moore Horspool algorithm are in the form of indices of patterns that match the sequence in the form of RDD objects (Yahya, 2021). As already explained, RDD objects cannot be displayed directly as output. Therefore, there are several ways that can be done so that this RDD object can be displayed, namely by changing it into ordinary variables such as lists, dataframes, strings and other common variables, saving it in a text file for the operating system, for example in .txt format or you can by storing it on HDFS as the data source is used using the action function provided by the Apache Spark platform. For this research, the output will be saved to HDFS using the saveAsTextFile() function, so that users can download data from HDFS if necessary for viewing.

2.2. Experimental Design

Before carrying out an experiment, an experimental design must first be carried out. There are two types of experiments that will be carried out, namely stand alone and using the Apache Spark platform for various nodes. In each type of experiment, there are several experiment categories based on the type of data. Each type of data will be searched for two cases of patterns to be searched for. The patterns to be searched for are 'GAATTGCAGAATCC' and 'GGATCC'.

If the pattern 'GAATTGCAGAATCC' is found in the DNA sequence, it can be concluded that the plant is a higher plant. This pattern will not be found in all chromosomes, but only in certain chromosomes where the location will be different for various plant species. The 'GGATCC' pattern is a pattern which, if found on a certain index, will make that index susceptible to dry leaf disease.

There are 4 types of data used in this research, taking into account the differences in each file used. This type of data will be referred to as scenario A, scenario B, scenario C and scenario

D. Scenario A is dummy data with a very small size where the string is less than 30 characters in size. This is done to prove that the pattern found is correct and in accordance with the results carried out manually. manual as a form of program validation. Scenario B is original plant DNA data with 3 types of species and 6 files with the size of each file between 13-146 MB. Scenario C is a scenario that is carried out to see the effect of speed on large amounts of data which when added up becomes quite large data where the directory being tested contains files with the DNA of the Oryza sativa japonica plant. Scenario D was carried out to prove a large process where the data used in one directory does not matter what species it is in to test the speed of the program implemented on the Apache Spark cluster platform.

The total number of experiments in this research is 102 experiments which are divided into two computing scenarios (stand alone and Apache Spark cluster) where the results of these two experiments will be used as material for discussion about accuracy and accuracy speed comparison based on computing concepts and the number of nodes used in the cluster Apache Spark on research.

2.2.1. Stand Alone Experiment Scenario

In the experimental scenario with stand alone computing, all experimental scenarios will be carried out with various kinds of test data, namely scenarios A, B, C and D as explained in chapter III Experimental Design. In detail the scenarios that will be tested in the stand alone experiment can be seen in **Table 2**.

Table 2 shows the experiments that will be tested on several data scenarios along with an explanation of what data and how to search for patterns on a stand-alone basis. In the information column in table 1 it is explained that there are two patterns, namely pattern 1 and pattern 2. In this case pattern 1 is a symbol or re-naming of the pattern in the first case study, namely 'GAATTGCAGAATCC' as a determinant of whether the species being tested is a low level plant or high level (flowering). And pattern 2 is a symbol of the pattern in the second case study, namely 'GGATCC' as a detection of the possibility of viral infection in that plant species. Then the dummy pattern used is 'CCCT' while the dummy data is 'CTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCC'.

Table 2. Stand alone experiment scenario.

No	Experiment Name	Experimental Description
1	Scenario A	Data Dummy and Pattern Dummy
2	Scenario B1	Data1.txt and Pattern 1
3	Scenario B1	Data1.txt and Pattern 2
4	Scenario B2	Data10.txt and Pattern 1
5	Scenario B2	Data10.txt and Pattern 2
6	Scenario B3	Data28.txt and Pattern 1
7	Scenario B3	Data28.txt and Pattern 2
8	Scenario B4	Data36.txt and Pattern 1
9	Scenario B4	Data36.txt and Pattern 2
10	Scenario B5	Data49.txt and Pattern 1
11	Scenario B5	Data49.txt and Pattern 2
12	Scenario B6	Data52.txt and Pattern 1
13	Scenario B6	Data52.txt and Pattern 2
14	Scenario C	Directori Oryza Sativa Japonica (Data1.txt – Data27.txt) and Pattern 1
15	Scenario C	Directori Oryza Sativa Japonica (Data1.txt – Data27.txt) and Pattern 2
16	Scenario D	Directori Master (Data1.txt – Data53.txt) and Pattern 1
17	Scenario D	Directori Master (Data1.txt – Data53.txt) and Pattern 2

2.2.2. Apache Spark Cluster Experiment Scenario

In the experimental scenario for Apache Spark cluster computing, the same data scenario as stand alone will be carried out, but in the Apache Spark cluster computing experimental scenario it will be carried out on several types of computing, the difference being the number of nodes that will be used in the experiment, namely single node, 3 nodes, 5 nodes, 11 nodes, 16 nodes. Details regarding the experimental scenarios for Apache Spark cluster computing for the number of nodes 1, 3, 5, 11, and 16 will be shown in **Table 3**.

Table 3. Apache spark cluster experiment scenario.

Node	Pattern	Scenario	Description
1, 3, 5, 11, 16 Node	Pattern dummy	Scenario A	Data DNA Dummy and Pattern Dummy
	Pattern 1, 2	Scenario B 1	Data1.txt
	Pattern 1, 2	Scenario B 2	Data10.txt
	Pattern 1, 2	Scenario B 3	Data28.txt
	Pattern 1, 2	Scenario B 4	Data36.txt
	Pattern 1, 2	Scenario B 5	Data49.txt
	Pattern 1, 2	Scenario B 6	Data52.txt
	Pattern 1, 2	Scenario C	Directori Oryza Sativa Japonica
	Pattern 1, 2	Scenario D	Directori Master

In **Table 3** it can be seen that the number of nodes used in the 'Node' column is then the pattern that will be searched or matched in the file located in the 'Scenario' column with details in the 'Description' column as details of what data and how the search will be carried out or matching.

3. RESULTS AND DISCUSSION

From the experiments that have been carried out, analysis and explanation will be carried out in several discussions. The first explanation is about the comparison of time for each type of computing used, including the comparison of stand alone computing time, single node, 3 nodes, 5 nodes, 11 nodes, and 16 nodes. A comparison of computing time is presented with a comparison in each data scenario used that the data scenarios are divided into Scenarios A, B1, B2, B3, B4, B5, B6, C and D for the two case studies or patterns sought. The second explanation is regarding the accuracy of the pattern results that can be found in each stand alone computation, single node, 3 nodes, 5 nodes, 11 nodes, and 16 nodes. In stand alone computing the pattern results are used as a reference for absolute computing results as they have been tested on dummy data and the process is carried out sequentially (no data is cut) so the results are considered to have 100% accuracy so the results of this stand alone computing pattern will be used as a reference to calculate the accuracy value of computing single nodes, 3 nodes, 5 nodes, 11 nodes and 16 nodes.

3.1. Comparison of the Speed of Each Computation

The experimental results comparing computing time will be differentiated according to the data scenarios used. The following are the details and discussion.

3.1.1. Scenario A

In scenario A, as has been explained, the data in scenario A is DNA fragment data or dummy data which is tested to validate that the program output is as required. The following is a time comparison diagram for each computation from scenario A.

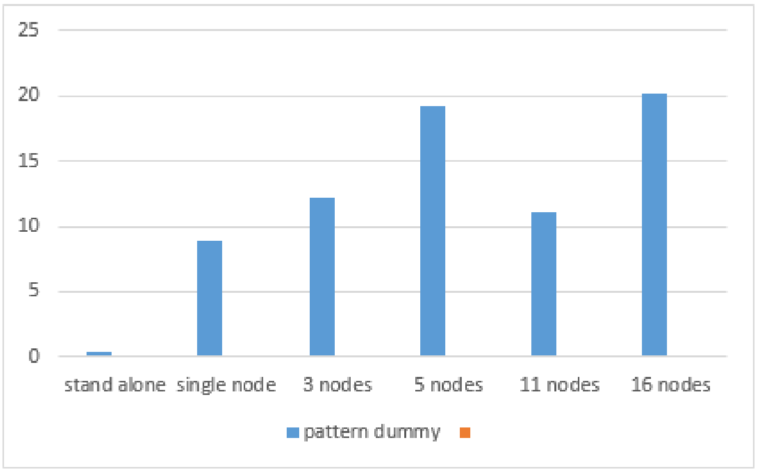


Figure 3. Experiment time speed graph scenario A.

From the experimental results it can be seen in the graph in **Figure 3** that the speed varies for scenario A. In stand-alone computing, it is ranked top for speed because the data used is small, even though Apache Spark is used for big data (Big Data), so in this case the Apache Spark cluster is still inferior compared to stand-alone computing. This could be caused by the 'communication' carried out between nodes in carrying out the process so that small data will make the process more difficult for more nodes.

3.1.2. Scenario B1-B6 Sample

In scenario B1, the data used is 41.8 MB. The data used is data1.txt, namely DNA of Oryza sativa japonica chromosome 1. From the results of the experiments carried out, the results obtained for each computation are shown in the graph in **Figure 4.**

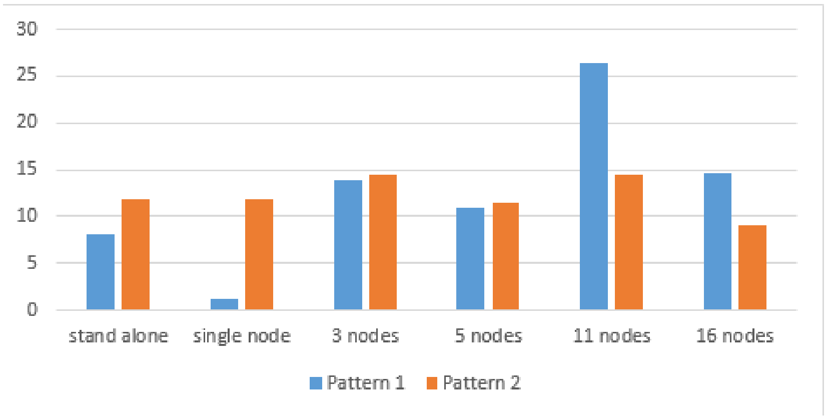


Figure 4. Experiment time speed graph scenario B1.

From the graph in **Figure 4** it can be seen that the experimental results carried out for all computations vary greatly. It was found that the longest process occurred in the Apache Spark cluster computing with the number of nodes 11 for matching pattern 1, and the fastest results occurred in the Apache Spark cluster computing with the number of nodes 1 or single node.

For Scenario B1, it can be said that the data is quite small because in stand-alone computing it can still be processed even in a matter of seconds. So it could be that the data is not considered Big Data for this scenario.

Next are the experimental results in scenario B2. In this scenario the data used is 28.7 MB. The data used is data10.txt, namely DNA of Oryza sativa japonica chromosome 7. From the results of the experiments carried out, the results obtained for each computation are shown in the graph in **Figure 5**.

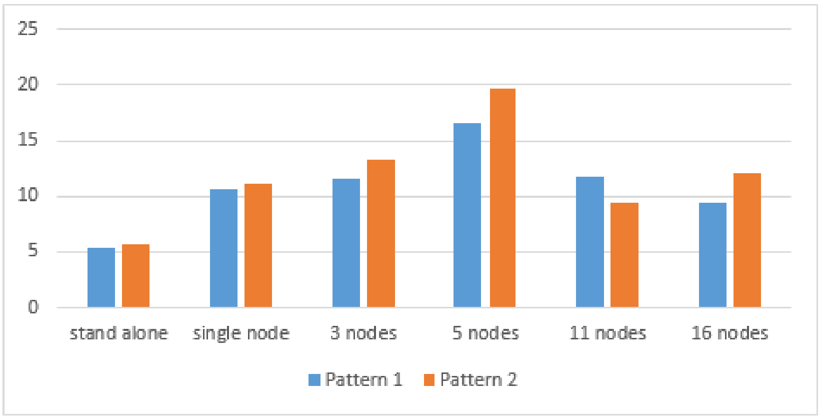


Figure 5. Experiment time speed graph scenario B2.

Not much different from scenario B1, in scenario B2 it can be seen from the graph in Figure 6 that the experimental results carried out for all computations vary greatly. It was found that the longest process occurred in the Apache Spark cluster computing with 5 nodes for matching pattern 2, and the fastest results occurred in stand alone computing for pattern 1. For Scenario B2, this is almost the same as scenario B1, it could be said that the data is quite small because it is in stand-alone computing alone, the process can still be carried out in a matter of seconds. So it could be that the data is not considered Big Data for this scenario.

3.1.3. Scenario C

The next experimental result is scenario C. In this scenario the data used is 625 MB. The data used is data1.txt to data27.txt, namely Oryza sativa japonica DNA throughout its chromosomes (1-12). From the results of the experiments carried out, the results obtained for each computation are shown in the graph in **Figure 6**.

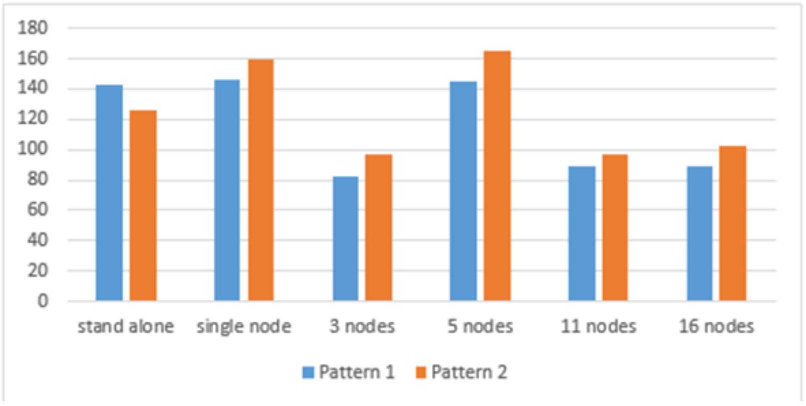


Figure 6. Experiment time speed graph scenario C.

In scenario C, it can be seen from the graph in Figure 7 that the experimental results carried out for all computations vary greatly. It was found that the longest process occurred in the Apache Spark cluster computing with the number of nodes 5 for matching pattern 2, and the fastest results occurred in the Apache Spark cluster computing with the number of nodes 3 for pattern 1. For Scenario C, the difference between stand alone computing and several Apache Spark cluster computing that in scenario C, the Apache Spark cluster is able to beat stand-alone computing for running time. The data used is approaching Big Data because the process takes more than one minute.

3.1.4. Scenario D

And the last one is the experimental results of scenario D. In this scenario, master data is used throughout the research. In this scenario, it is to test quite large data. The data used is 1.86 GB. The data used is data1.txt to data53.txt, namely the DNA of Oryza sativa japonica, Oryza brachyantha and asparagus officinalis throughout their chromosomes. From the results of the experiments carried out, the results obtained for each computation are shown in the graph in **Figure 7** below.

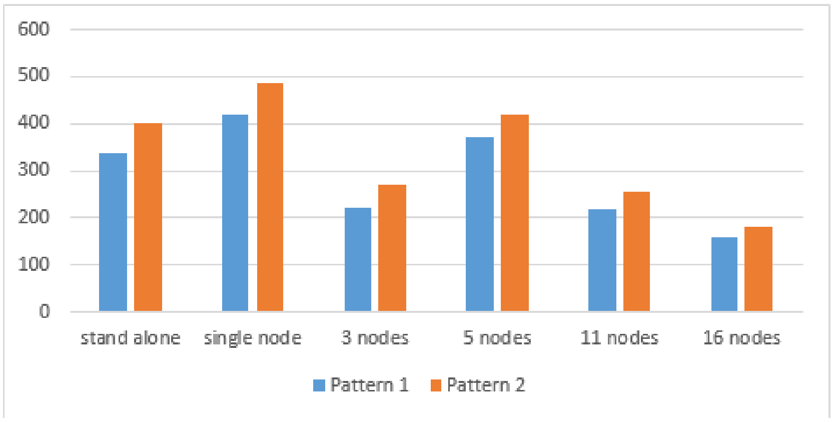


Figure 7. Experiment time speed graph scenario D

In scenario D, it can be seen from the graph in Figure 8 that the experimental results carried out for all computations vary greatly. It was found that the longest process occurred in the Apache Spark cluster computing with a single node in matching pattern 2, and the fastest results occurred in the Apache Spark cluster computing with the number of nodes 16 for pattern 1. For Scenario D, quite a big difference between stand alone computing has begun to be seen. with several Apache Spark cluster computations, especially for the largest number of nodes, in scenario D, the Apache Spark cluster is able to beat stand alone computation for running time. The data used is approaching Big Data because the process takes more than one minute.

From scenarios A to D which have been presented in the graph as well as the explanation, an analysis can be obtained that the use of the Apache Spark platform in all scenarios is still not optimal. In almost every scenario the increase and decrease in time are unstable. Because communication between nodes in data distribution is not comparable to processes and data that are still light or can be handled stand alone. So the benefits of using the Apache Spark platform are not yet maximized.

3.2. Comparison of Accuracy Values

The processes that occur in the Apache Spark cluster run in parallel, which means that long input data will be broken up and searched or matched on each piece by the Boyer-Moore Horspool algorithm (Muthunagai & Anitha, 2022). This of course can lead to the possibility that patterns that should be found are not found because they are truncated during distribution, especially in RDD where the chunks are smaller than HDFS. Therefore, it is necessary to carry out calculations related to accuracy when using the Apache Spark platform. How bad is the accuracy value of the patterns that can be found when using Apache Spark.

In this research, stand alone will be used as a base accuracy value to compare with the Apache Spark cluster. Stand alone results will be considered to have an accuracy value of 100% on the grounds that the processes carried out in stand alone do not experience intersections or are searched sequentially (Ridha *et al*, 2021).

For the first case study, namely regarding searching for patterns to determine whether a plant belongs to the high level or low level plant category, it has an accuracy value of 71.48611%. For the second case study, detecting patterns related to dry plant viruses (begomovirous group), the accuracy value was 93.01301%. In the first case study, it was smaller because there was only 1 pattern in each sequence, at most 2 patterns were found in a DNA sequence. However, not for the second case study. There will be so many patterns that can be found that the lowest accuracy value is 56.3421%. Meanwhile, in the first case study, the lowest accuracy was 0% because the pattern was truncated when distributing the data.

4. CONCLUSION

After conducting research regarding the implementation of Apache Spark on the Boyer-Moore Horspool Algorithm for the Internal Transcrbed Spacer and Restriction Enzyme case studies, the author came to several conclusions.

(i) After analyzing the experimental results, it is concluded that the time cut does not coincide with the number of nodes. It can be concluded that in this research the use of the Big Data platform was not optimal. Because the curve produced in the diagram throughout the experiment does not decrease. It even experiences unstable increases and decreases. This is an indication that the use of Big Data for the case in this research is not on target.

(ii) The Apache Spark Big Data platform is proven to be able to be implemented for string matching, especially the Boyer-Moore Horspool algorithm.

Further research could result in a much better program in terms of speed. Here are some suggestions that the author can suggest.

(i) In doing data input is still done in the program code so it must be done by those who understand programming.

(ii) Regarding the theme of Big Data research, it must first be carefully examined whether the data and case studies are heavy and large enough so that standing alone will take a long time so that the use of the Big Data platform will be maximized.

(iii) Creating some more diverse and dynamic preprocessing functions so that they can be used on universal data, not based on data with NCBI/Ensembl format standards and also naming headers when data is called in one directory at once so that chromosome sequences will not be unified.

(iv) Create a function to overcome the intersection of DNA sequences that occur so that no patterns are cut in the sequence and produce better accuracy values.

(v) The author hopes that this program can be used to support the development of science, especially in the fields of computer science and biology.

(vi) The author hopes that this program can be developed much better.

5. AUTHORS' NOTE

The authors declare that there is no conflict of interest regarding the publication of this article. The authors confirmed that the paper was free of plagiarism.

6. REFERENCES

Bayat, A. (2002). Science, medicine, and the future: Bioinformatics. *BMJ: British Medical Journal, 324*(7344), 1018.

Derrien, T., Estellé, J., Marco Sola, S., Knowles, D. G., Raineri, E., Guigó, R., and Ribeca, P. (2012). Fast computation and applications of genome mappability. *PloS one, 7*(1), e30377.

Devakunchari, R. (2014). Analysis on big data over the years. *International Journal of Scientific and Research Publications, 4*(1), 1-7.

Hidayat, T., Priyandoko, D., Wardiny, P. Y., and Islami, D. K. (2016). Molecular phylogenetic screening of withania somnifera relative from indonesia based on internal transcribed spacer region. *HAYATI Journal of Biosciences, 23*(2), 92-95.

Hoong, C. C., and Ameedeen, M. A. (2017). Boyer-moore horspool algorithm used in content management system of data fast searching. *Advanced Science Letters, 23*(11), 11387-11390.

Kadkhodaei, H., Moghadam, A. M. E., and Dehghan, M. (2021). Big data classification using heterogeneous ensemble classifiers in Apache Spark based on MapReduce paradigm. *Expert Systems with Applications, 183*, 115369.

Muthunagai, S. U., and Anitha, R. (2022). TDOPS: Time series based deduplication and optimal data placement strategy for IIoT in cloud environment. *Journal of Intelligent and Fuzzy Systems, 43*(1), 1583-1597.

Ridha, H. M., Gomes, C., Hizam, H., Ahmadipour, M., Heidari, A. A., and Chen, H. (2021). Multi-objective optimization and multi-criteria decision-making methods for optimal design of standalone photovoltaic system: A comprehensive review. *Renewable and Sustainable Energy Reviews, 135*, 110202.

Salehan, M., and Negahban, A. (2013). Social networking on smartphones: When mobile phones become addictive. *Computers in human behavior, 29*(6), 2632-2639.

Stephens, Z. D., Lee, S. Y., Faghri, F., Campbell, R. H., Zhai, C., Efron, M. J., ... and Robinson, G. E. (2015). Big data: astronomical or genomical?. *PLoS biology, 13*(7), e1002195.

Tang, S., He, B., Yu, C., Li, Y., and Li, K. (2020). A survey on spark ecosystem: Big data processing infrastructure, machine learning, and applications. *IEEE Transactions on Knowledge and Data Engineering, 34*(1), 71-91.

Villar, S., Hogg, D. W., Storey-Fisher, K., Yao, W., and Blum-Smith, B. (2021). Scalars are universal: Equivariant machine learning, structured like classical physics. *Advances in Neural Information Processing Systems, 34*, 28848-28863.

Wilisiani, F., Somowiyarjo, S., and Hartono, S. (2014). Identifikasi molekuler virus penyebab penyakit daun keriting isolat bantul pada melon. *Jurnal Perlindungan Tanaman Indonesia, 18*(1), 47-54.

Wu, X., Zhu, X., Wu, G. Q., and Ding, W. (2013). Data mining with big data. *IEEE transactions on knowledge and data engineering, 26*(1), 97-107.

Yahya, A. A. (2021). Android-Based Horspool Algorithm for Proverb Search. *Instal: Jurnal Komputer, 13*(1), 1-9.