Journal of Engineering Science and Technology
Special Issue on ACSAT, December (2021) 1 - 11

BIG DATA PROCESSING USING HADOOP HDFS AND MAP-REDUCE FOR PUBLIC OPEN DATA (POD)

NAJHAN M. IBRAHIM[1,*], NORBIK B. IDRIS[2], MODH K. A. HASSAN[2], CIARA BREATHNACH[3], AMIR A. A. HUSSIN[3]

[1]Department of Information System, International Islamic University Malaysia, Jalan Gombak, 53100, Selangor, Malaysia
[2]Department of Computer Sciences, International Islamic University Malaysia, Jalan Gombak, 53100, Selangor, Malaysia
[3]Department of History, Health Research Institute, University of Limerick, Limerick, V94 T9PX, Ireland
*Corresponding Author: najhan_ibrahim@iium.edu.my

Abstract

Information acquired and processed by government agencies and made available to the public via a web portal is known as Public Open Data. Many governments have created Public Open Data (POD) to promote government transparency by making data available to the public. Malaysia is no exception, having opened public-access data since 2014. Processing large amounts of data may become increasingly complex as the availability of data sources, data types, data volume, speed, and complexity grow. As a result, it's important to consolidate and find a new method for handling and optimizing massive amounts of POD. With the development of Big Data, computer system overflow has raised some problems. As a result, industry, researchers, and government have entered into a number of comprehensive research and development cooperative agreements. While the large volume of data and variety of data formats can make Big Data processing challenges, it also has the potential to produce innovative solutions with a wider range of applications. There's also a lot of debate over whether or not Big Data can supplant traditional data records. This study aims to investigate the potentials of big data processing utilizing Apache Hadoop and Java map/reduces in the future to discover valuable patterns, correlations, trend preferences, and other important information. With improved the future prediction and prevention mechanisms in place by authorities, the overall cost of government for public service would be reduced.

Keywords: Apache Hadoop, Big data processing, HDFS, Map-reduce, Public open data.

1. Introduction

Public Open Data (POD) is information gathered and processed by government agencies and made available to the public via a web portal. Many countries around the world have developed POD to increase government transparency by sharing data with the public. Malaysia is no different, since data.gov.my was established in 2014 as a data centre open to the public. Nevertheless, as the variety of datasets, volumes, speeds, and complexity will be increased. As a result, finding a common ground for the integration of various data is critical. [1, 2]. POD has gained a lot of attention in recent years, and it's now part of the everyday terminology of transparent activists, non-governmental organization (NGO), and public officials. In order to explain disparities in POD provisions among various government institutions, a growing body of academic study is focusing on POD efforts and policymaking. Without a question, POD and its implications have gotten a lot of scholarly attention.

Volume, verities, and velocity were the three main keys linked with BD at first. Structured, semi-structured, and unstructured data were all included in big data [1]. refers to data collections that are so large or complicated that traditional data processing systems cannot handle them. Research, data acquisition, data duration, search, sharing, storage, conversion, visualisation, querying, and information privacy are all challenges. The key notion of big data with multiple sorts of data will be focused on in this study paper. Big Data (DB) is defined by Graham, Milligan, and Weingart (2017) as "data that you can't handle with traditional technology or, more broadly, data that requires special computational intervention to operate" [3, 4]. Civil registration (CR), which satisfies the definition provided by Graham et al., will be the primary source of data. In recent years, CR data has been recorded in a variety of formats, including Microsoft Excel, Microsoft Word, and traditional database systems such as Microsoft Access and MySql, across a wide geographic area.

BD processing can assess and analyse a large number of datasets from many sources, identifying noteworthy patterns, untapped potential, data trends, and other useful data. The Public Open Data (POD) is not only large in size, but also in format, and the content is difficult to comprehend. POD can also categorise information by its specific attribute, such as “unstructured, incomplete, multi-dimension, and heterogeneous” [5], which may necessitate the development of a new methodology to find a meaningful trend. As a result, rapidly evolving technologies enable us to collect various forms of data in order to uncover a meaningful pattern of data; nevertheless, creating more data also means generating more ambiguous or unexpected data. Thus, it is important to find an analytical method to identify relevant particulars of POD for the specific purpose of the research study [6].

2. Related Works

Fast growing of data collection technique encoring big data processing to develop in recent years, a variety of methods for evaluating and benchmarking them have been developed in order to validate the process and make additional improvements. This section presents a comprehensive review of the typical state-of-the-art data processing techniques.

Several performance evaluation models for HDFS are presented in [5]. Modelling HDFS performance is difficult because of the data's intricacy organisation and DFS foundational pillars, especially when we consider the many elements that can have

an impact on it. Dong et al. [7] In order to construct HDFS performance models in a methodical manner, evaluate the link between HDFS read/write processing time and source file size. The authors describe measurements for the HDFS operations' dynamic features. Their method not only aids in predicting HDFS performance under specific situations, but also in identifying the primary factors that influence it, as well as the correlations between them. From an architectural standpoint, Shafer et al. [8] investigate obstacles in distributed file system processing. The purpose is to evaluate the efficiency and accessibility execution in HDFS.

Because all files are controlled by a single server, HDFS system is particularly efficient for processing large files, but it quickly degrades when processing a huge number of little files. A map task is created while storing a file. Because there are so many little files, a huge number of map jobs must be created, which slows down the file system. Multiple research look at the problems with storing tiny files in HDFS [9]. Several approaches to tackling these have lately been offered in the literature. Rathidevi and Srinivasan [9] compare and contrast these approaches. Bok et al. [10] offer a distributed caching architecture for HDFS that allows for quick access to tiny files. By consolidating numerous small datasets into an unique block of capacity, It reduces the amount of content that the name node has to handle.

In [11], Krishna et al. evaluate the processing efficiency of HDFS in terms of file sizes, which included small and large datasets. To evaluate HDFS processing, they employ 5 nodes Hadoop cluster. According to their observations, HDFS works well for files with block sizes greater than the default, but not so well or efficiently for files with block sizes smaller than the default. They conclude that based on a small number of cluster nodes and do not account for the replication factor. A study that is comparable to this one may be found in [12] Han et al. examine the most up-to-date methodologies for comparing large data storage and processing systems and evaluate their performance using prominent open-source benchmarks. They provide a thorough grasp of the three major components of process improvement: load creation, input data generation, and system evaluation indicators.

3. Research Method

The paper's research methodology included a prime example of Malaysian public open data. This study examined Malaysian public open data since the accessible data is extensive and has been publicly available for some years with a variety of data sources. The research methodology consists of two main phases. First, there's the literature review, which entails a critical assessment of current and previous research based on journals, conferences, research reports, and other academic materials. This inquiry necessitates an awareness of contemporary big data processing difficulties. The goals of this analysis are to determine research gaps, requirements, and the best methodologies for big data processing.

Next, the development of big data processing, which is based on the Apache Hadoop Ecosystem, is the second priority. Apache Hadoop is a set of open-source software tools for solving Big Data challenges with a distributed network of computers. Using the Hadoop Distributed File System (HDFS) and the MapReduce programming model, it provides a software framework for distributed storage and processing of huge amounts of data [8]. Apache Hadoop required a minimum computing power to generate where it is able to install on personal laptop with common specs. Apache Hadoop was installed on Linux Ubuntu. Furthermore, given

to the vast amount of data available and the variety of data types and sources, BD processing has become a prominent topic of research. The lack of understanding in Malaysia Public Open Data about big data processing prompts this study to focus on the development of BD processing.

4. Apache Hadoop

BD analysis is the process of collecting, maintaining, and analysing huge datasets from a number of sources in order to identify hidden patterns, unknown linkages, trend preferences, and other relevant information. The Blu-ray discs are massive in size as well as content complexity. The content's distinguishing properties (multi-dimension, heterogeneous, unstructured, incomplete, and erroneous) can also be detected by BD, which may necessitate the employment of a specialised approach or strategy for common data types. Although technology developments appear to make it more data could be collected in order to identify several relevant patterns, this may result in the generation of more ambiguous or anomalous data. As a result, finding an analytical strategy to discover essential details of datasets for the particular objective of the research work is critical. We use Apache Hadoop for Big Data analysis in this paper [9].

In January 2008, Apache Technologies announced their major project Hadoop. Hadoop has been utilised in a variety of industries since January 2008, including financial services, government services, telecommunications, advertising, and several search engines like as Google. It's an open-source software platform for distributed computing that's dependable, scalable, and secure. connect a single server to tens of thousands of devices, each of which provides local processing and storage. HDFS and MapReduce are the two most important Hadoop components. Apache Hadoop is a set of open-source software and utility applications that encourages the use of distributed computing. network computers to process Big Data problem. It can also be considered of as an application platform for large-scale distributed data processing employing Hadoop Distributed File System (HDFS) and the Map-Reduce programming methodology [10].

4.1. Hadoop ecosystems

Hadoop's ecosystem follows a top-down strategy. There were only two units in Hadoop 1.x that actually worked, MapReduce and HDFS, in earlier releases. The data is processed by MapReduce, and the results are automatically stored in HDFS. However, in a later version of Hadoop 2.× (see Fig. 1), a new component, namely Hadoop 2.x, is added to its ecosystem, i.e., YARN (Yet Another Resource Negotiation). YARN is similar to Map-Reduce; however, the processing method is different. The data in the container is processed by YARN. The logic units are the containers, which are made up of the resource and the task itself (here resource is DataNode). Deadlock situations, which were common in 1.x, are reduced in 2.× because to the presence of YARN [11].

The ecosystem also extends a number of software utilities such as Hive, Pig, Oozie, Sqoop, Flume, Spark, HBase, Zookeeper that enable businesses to accumulation, development, and evaluate huge amounts of datasets, provide real-time customer service, perform various types of enhancements method such as, Extract Transform Load (ETL), machine learning, ETL. Pig is a software utility that allows you to analyze massive amounts of data. On top of the MapReduce programming

approach, it is an alternative abstraction. It makes use of Pig Latin, a unique data flow scripting language. Expressions are utilized in the same way that they are in SQL. The Pig Latin scripts are converted into MapReduce jobs via this tool. Because creating MapReduce code can be tough at times and can take so much of operation time. As a result, it devotes a significant amount of effort to optimizing Java Map-Reduce code and producing Pig Latin scripts. Pig was designed to process large data sets in batches and is not suitable for all data processing or analytic tasks [12].

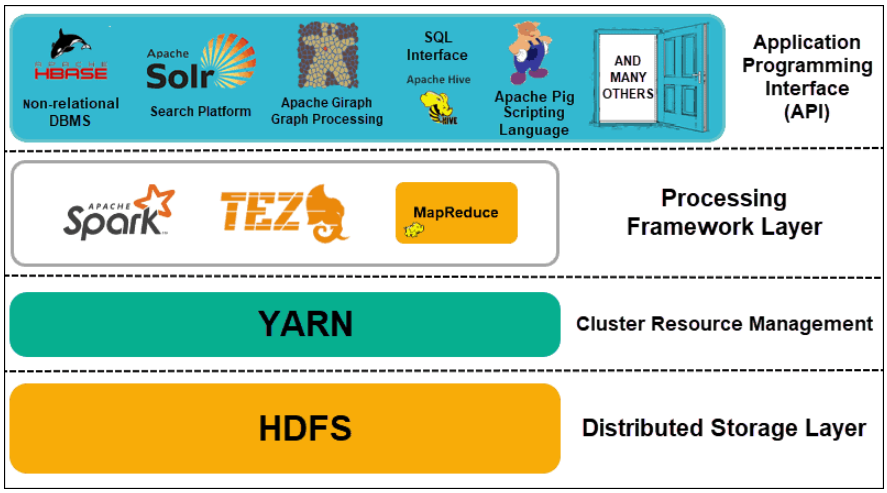

Fig. 1. Hadoop 1.0 and Hadoop 2.0 Ecosystem [11].

4.2. HDFS (Hadoop Distributed File System)

HDFS can be considered as distributed data processing unit that may be used to tackle issues regarding large amounts of datasets with multiple types of data types. Data is collected in a decentralized environment across numerous nodes. The fundamental concept is to create a parallel processing that processes information in DataNodes. The basic principle of HDFS is "read many times but write once." On the HDFS file system, a file can be read several times, but it can only be written once. HDFS supports high-throughput application data access and is well-suited to applications with huge data collections. It distributes data over numerous nodes in a cluster (a collection of machines connected by a LAN or MAN connection).

Hadoop has two components: NameNode and DataNode. NameNode is a Hadoop node that holds metadata (such as name, path, and so on) and assists the client in interacting with the Hadoop architecture. DataNode holds the actual data and processes it with the help of the processing unit. SSH and the TCP/IP protocol are used to communicate between the machines in the cluster. It is based on architectural principles of master-slave relationships as presented in Fig. 2. It also works with a hierarchical file management system. A user has the ability to create directories and store files within them. The NameNode is in responsible of the system files namespace. Any changes to the system files namespace or its attributes are maintained by the NameNode. An application can determine the amount of replicas of a file that should be preserved by HDFS. The copy number of a file is known as the replication factor.The NameNode keeps track of this data [11].

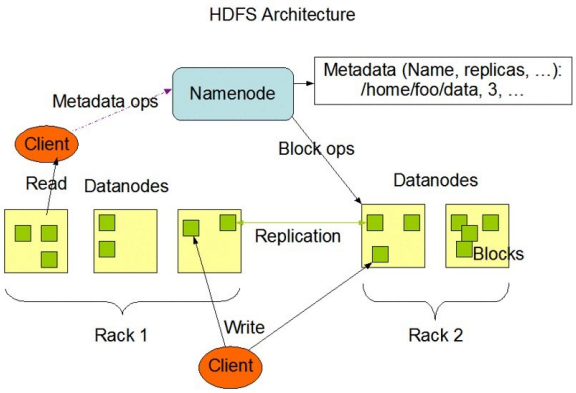

Fig. 2. HDFS Architecture [12].

The Hadoop architecture aims to provide a dependable, scalable, and shared storage system for huge data collections, as well as distributed processing. As a result, it relies on two primary components, which the for distrusted processing, the Map-Reduce programming style is used, and for distrusted storage, the Hadoop Distributed File System (HDFS) is used. Both architectures are capable of handling large data sets. Hadoop takes care of data replication and node failure on its own. As a result, it is less expensive than other traditional data storage systems. Hadoop is suitable for processing enormous amounts of data. It's made up of layers, after all. This architecture performs poorly with low-dimensional data. As a result, for large-scale data, Hadoop should be used. Alternatively, it increases the processing time and complicates data analysis. Hadoop can be installed in three different ways. Install locally, pseudo-distributed, and fully distributed [12].

4.3. Map-reduce

Map-Reduce is a programming module to process a large amount and distributed of datasets. Map and Reduce are two of the most important functions in the Map-Reduce programme module. Map executes a collection of data and transforms it into another set, with each component broken down into value pairs. Second, it simplifies the process by using the map's output as an input and merging the datasets into a smaller collection of value pairs. After the map operation, the reduction process is always run.

A Map-Reduce module consists of a map module that manages and filters data, as well as a reduce function that summarises the results of an operation. The Map-Reduce Module can also be thought of as architecture or framework for marshalling distributed datasets and performing several activities in parallel, such as managing communications and transferring data from diverse system sources.

4.4. Map-reduce paradigms

The Map-Reduce programming structure can improve the complex nature by executing parallel data preparation capacities across multiple computing hubs in a cluster, as flexible, it allows software engineers to distribute programmes among centres as they are conducted in parallel. As a result, Map-Reduce collects data from

multiple hubs and produces a single result or set. Significantly, these stages allow software programmers to respond to internal failure in a straightforward manner.

Map-Reduce is a simple but powerful data processing system for massive datasets. It distributes and parallelizes the massive amount of data it processes. The MapReduce cluster is made up of thousands of nodes connected by a network. It's based on a master/slave system. The Job Tracker is the master node, and the Task Tracker is a slave to the master node. Programs written in this format are automatically parallelized and performed on a huge, large number of distributed devices. The Runtime system is in charge of splitting the input data, scheduling the program's execution over a group of machines, managing machine interruptions and arranging the necessary inter-machine communication This makes it simple for programmers with no prior knowledge with distributed systems to take advantage of the resources of a large distributed system.

The scheduling algorithm determines which tasks are executed at any given moment and on which slave node. Hadoop can improve its efficiency in areas like data locality rate and task completion time using a variety of scheduling strategies. Each of the traditional algorithms improves performance in a different way. Data locality in Hadoop is the practise of placing processing close to where the real data is on the node, rather than transmitting large amounts of data to computation. This reduces network congestion and boosts the system's overall throughput [11- 13].

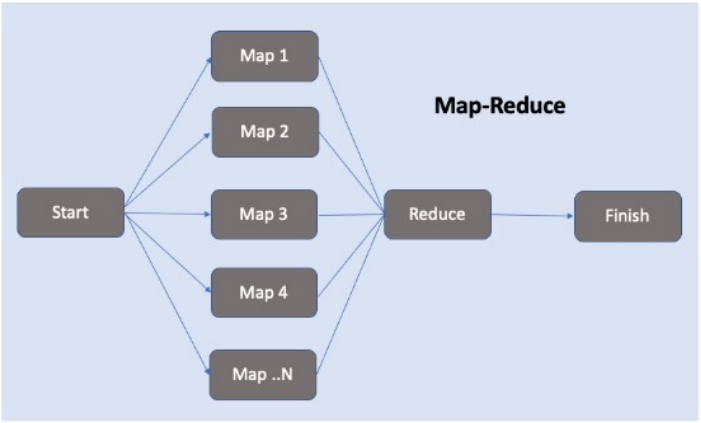

Fig. 3. Map-reduce architecture.

Generally, Map-Reduce paradigm is based on three different processes, which are mapping process, shuffle process, and reduce process as presented in Fig. 3.

1) **Map process:** Each node applies the map function to local data and saves the output to a temporary storage location. The main node will handle only one copy of the redundant input data. This is the first step, which involves converting input data into a collection of data, with individual sets being divided into tuples (key/value pairs).

2) **Shuffle process:** Based on the output of the map function procedure, the nodes redistribute data.

3) **Reduce process:** The nodes are now processing each group of output data in parallel, per key. This process took the result of a map job and combines the data tuples into a smaller set. The decrease task is always done after the map task.

5. The Finding

One of the most intelligent ways to store and compute massive data is through distributed and parallel processing. Map-Reduce is a software module that allows you to process large amounts of data. Programs that use Map-Reduce are essentially concurrent. There is a master and several slaves in a MapReduce system. The framework's management, which includes user interface, job queue organisation, and task scheduling, is handled by the master. Each slave has a certain number of map and reduction slots to complete chores. The master's job scheduler assigns work based on how many available tasks slots each slave reports via a heartbeat signal. A job is characterised by a large number of map and reduce jobs before it is completed. The runtime is in responsible of completing all duties assigned to each job. The jobs are indeed carried out on any of the MapReduce cluster's slave nodes.

The sample of datasets were more than a million rows of records, which required high computing power to execute using traditional method. For example, just to view the content without any analysing by using spreadsheet we were required to slice the data into several peace. This is because the traditional spreadsheet is able to view approximately a million row per sheet and consume several minutes to scroll in down. However, Apache Hadoop capable analyse all of 5 million records of data less than a minute using Map-Reduce method as presented in Figs. 3 and 4. The MapReduce process was executed by the Hadoop API, which provides command to give multiple input path and execution of process as presented in Fig. 3. The information were read, analyzed, and processed before being saved to Hadoop HDFS as an output file. All of the files can be browsed in the default URL: http://localhost:50070/ as presented in Fig. 4.

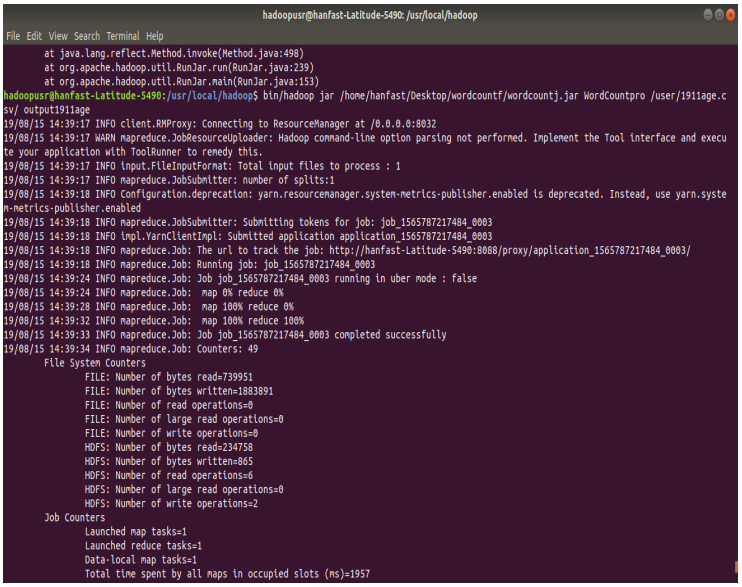

Fig. 4. Time taken for processing the data sample

HDFS is a distributed file system (DFS) as shown in Fig. 5 that manages files across a cluster's capacity. It's in charge of data replication and fragmentation. Large files are split into many blocks, each of which is duplicated on different servers, based on the file replication factor. This improves information visibility even if one or more servers go down. The distributed file platform is constructed on a master-slave design and adheres to the "Write Once, Read Many" principle." It's a fault-tolerant, distributed "key-value" data storage system that's built to manage enormous files.

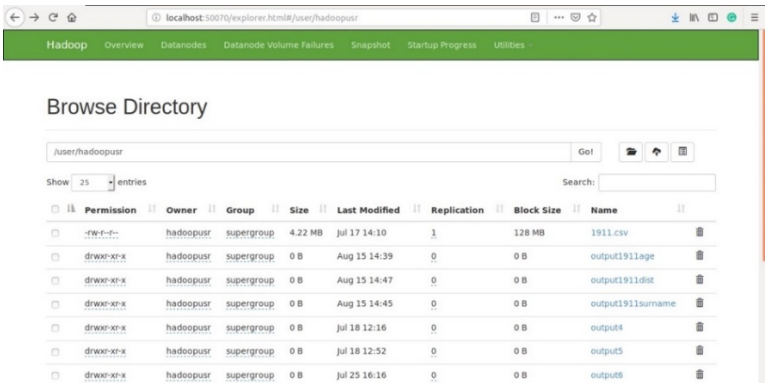

Fig. 5. Browse Directory HDFS File.

Two files in the output directory (directory given in the command on console) are received as part of the Java implementation of the MapReduce programme module. The first file is the _SUCCESS file, which is an indicator/flag of the program's successful execution. The part-r-00000 file is the second file, and it contains the actual output data. If the processor uses more than one reducer, distinct files are generated as output that are designated appropriately (e.g., Part-r-00001 and so on), as seen in Fig. 6. Hadoop produces consistent results when mapping and analysing data. This Data can be employed for further representation.

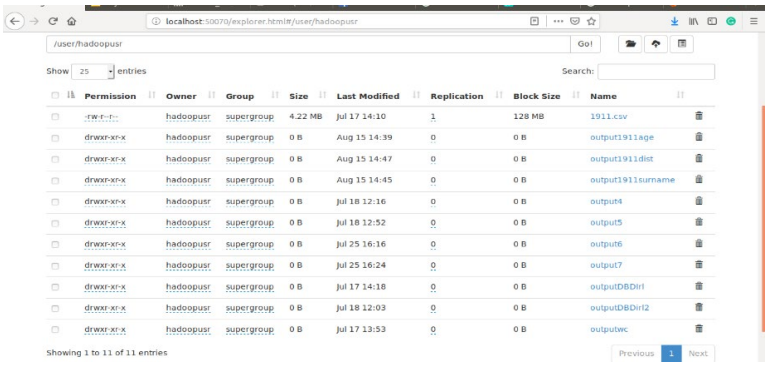

Fig. 6. Output on HDFS File.

In addition, there are growing number of Hadoop usage from different organization in past few years. It is included some of well-known companies like Yahoo, Google, Amazon, Facebook, IBM. They prefer Hadoop compare to other technologies because of processing of unstructured data being easier in Hadoop than

others. However, some of companies are not preferred to use due to the inadaptability of the big data techniques. As of now more than 50% of the fortune 50 companies are using Hadoop as seen in Fig. 7 [14, 15].

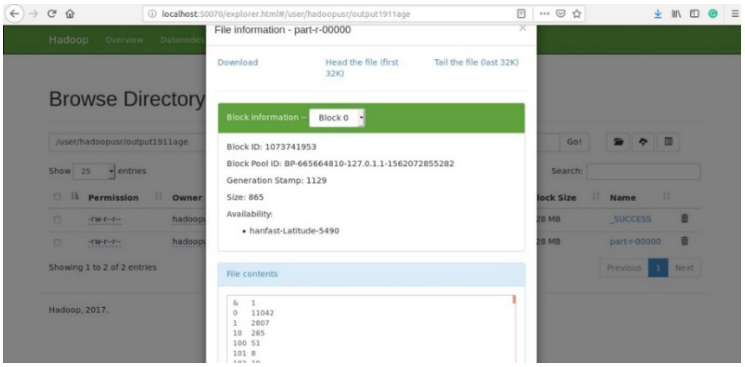

Fig. 7. Output file on web-browser.

6. Conclusions

BD assessment could examine and investigate a large number of datasets from many sources, revealing significant patterns, untapped potential, data trends, and other useful information. The Public Open Data (POD) is not only large in bulk, but also in format and difficult to comprehend. POD also could organize information according to its specific attribute, such as incomplete, unstructured, multiple dimensions, and heterogeneous, which may necessitate the development of a new methodology to find a useful data trend. Consequently, rapidly evolving technologies enable us to collect a variety of data kinds in order to uncover a meaningful pattern of data. Nevertheless, creating extra data also means generating more ambiguous or atypical datasets. As a result, it's critical to consolidate and develop a large data processing method for investigating and generating useful data trend.

Big data processing considers the intricate relationships between storing and analysing data from a variety of sources, as well as the changing patterns over time and other variables. High-performance computer platforms are necessary to facilitate big data mining. From the recent decade, there has been a significant increase in the use of Apache Hadoop. Finally, when compared to traditional data processing methodologies, Hadoop technology allows you to process a huge number of datasets in less time and with fewer resources. Successful implementation of Hadoop framework to process huge amount of data (BD) has been implemented on Linux OS and the sample datasets have been processed according to the needs, using HDFS and Java Map-Reduce programming method of Apache Hadoop. However, Apache Hadoop is an architecture based for BD, which needed serval application (Hadoop ecosystem) to provide more comprehensive data analysis.

Acknowledgments

This research is funded by KICT Initiative research grant by International Islamic University Malaysia (IIUM) Award 2021. The project number is KICT-RG20-006-0006. We are grateful for the full cooperation of the RMC for facilitating this research grant.

References

1. Trandabat,; D. and Gifu, D. (2017). Social media and the web of linked data. *ACM/IEEE Joint Conference on Digital Libraries* (*JCDL*), Toronto, Canada, 1-2.
2. Hammad, S.; Telfah, A.; Ezzeldien, M.; and Morsi, H. (2016). Current developments in biomedical research, *International*. *Journal of Advanced Biomedicine*, 1(1), 1-3.
3. Graham, S.; Milligan, Ian.; and Scott, W. (2016). Big *Digital history: exploring big data through a historian's macroscope*. London: Imperial College Press.
4. Rob, K. (2017). Big data, new epistemologies and paradigm shifts'. *Big Data and Society*, 1(1), 1-12.
5. Najhan, M.I.; Aatieff, A.H.A.; Azmi, M.K.; and Breathnach, C. (2021). *Big data interoperability framework for Malaysian public open data*. Springer.
6. Saeed F., Mohammed F., Al-Nahari A. (2020). Data engineering and communications technologies. *Innovative Systems for Intelligent Health Informatics*. Retrieved July 15, 2021, from https://doi.org/10.1007/978-3-030-70713-2_39.
7. Chang, W.L.; and Nancy, G. (2018). Big data interoperability framework: *National Institute of Standards and Technology* (*NIST*). Retrieved July 15, 2021, from https://doi.org/10.6028/NIST.SP.1500-1.
8. Dong, B.; Zheng, Q.; Tian, F.; Chao, K. M.; and Godwin, X.H. (2016). Performance models and dynamic characteristics analysis for HDFS write and read operations: A systematic view. *Journal of Systems and Software*, 93, 132-151.
9. Shafer, J.; Rixner, S.; and Cox, A.L. (2016). The Hadoop distributed filesystem: Balancing portability and performance. 2015 *IEEE International Symposium on Performance Analysis of Systems and Software* (*ISPASS*), 122-133.
10. Rathidevi, R.; and Srinivasan, S. (2018). Small files problem in Hadoop - a survey. *International Journal of Pure and Applied Mathematics*, 119(15), 2833- 2841.
11. Bok, K.; Oh, H.; Lim, J.; Pae, Y.C.; and Yoo, J. (2017). An efficient distributed caching for accessing small files in HDFS. *Cluster Computing*, 20(4), 3579-3592.
12. Krishna, T.R.; Ragunathan, T., and Battula, S.K. (2016). Performance evaluation of read and write operations in Hadoop distributed file system. 2014 *Sixth International Symposium on Parallel Architectures*, *Algorithms and Programming* 110-113).
13. Han, R.; John, L. K.; and Zhan, J. (2018). Benchmarking Big Data Systems: A Review. IEEE Computer Architecture Letters, 11(03), 580-597
14. Singh, R.K. (2018) Taxonomy of big data analytics: methodology, algorithms and tools. *International Journal on Future Revolution in Computer Science and Communication Engineering*, 4(12), 2454-4248.
15. Gyamfi N.K.; Appiah P.; Sarpong, K.A.; Gah, S.K.; Katsriku, F.; and Abdulai, J. (2017) Big data analytics: survey paper. *Conference Proceeding: Dialogue on Sustainability and Environmental Management*, Accra, Ghana,15-16.