Identification of Influential Nodes in Social Network: Big Data - Hadoop

Rajnish Kumar [a,1,*], Laxmi Ahuja [a,2], Suman Mann [b,3]

[a] Amity Institute of Information Technology, Amity University, Uttar Pradesh-201313, India
[b] Maharaja Surajmal Institute of Technology, New Delhi, India
[1] rajnishghose@gmail.com, [2] lahuja@amity.edu, [3] sumanmann@msit.com
* corresponding author

ARTICLE INFO

Article history
Received October 15, 2023
Revised December 7, 2023
Accepted February 17, 2024

Keywords
DevOps
cloud pipelines
continuous integration
continuous development
influential nodes
agile development

ABSTRACT

Software development and associated data is the most critical factor these days. Currently, people are living in an internet world where data and related artifacts are major sets of information these days. The data is correlated with real-world data. The analysis of large datasets was done as part of the experimental analysis. The dataset for online social media like Facebook and Twitter was taken for the identification of influential nodes. The analysis of the dataset provides an overview and observation of the dataset for Facebook or Twitter. Here, in the current activity, an overview of cloud computing and big data technologies are discussed along with effective methods and approaches to resolve the problem statement. Particularly, big data technologies such as Hadoop provided by Apache for processing and analysis of Gigabyte(GB) or petabyte(PB) scale datasets are discussed for processing data in distributed and parallel data fashion. Here, the processing of large datasets is done by big data technology by implementing Apache Hadoop in online social media.

1. Introduction

The current world is moving from paperwork to digital work where everything is getting digitized in our day-to-day life starting from social media networks to the online classroom, sending messages on Twitter to uploading social media photos on social media networks like Facebook. Processing and getting the exact extract with such a large dataset is never an easy task. Also, the online Facebook dataset does not contain not only text data but images and special characters as well. This data type cannot be processed in any traditional tool. Hence, to overcome these challenges, there was a demand for any special type of tool and methodology which was capable of solving these types of data. Hadoop Eco-system and Map-Reduce were invented to solve these issues. As per information available on Statista [1] , globally, the total amount of data inflow and outflow in the year 2020 increased to 64.2 Zettabytes, for the year 2021 the same was 79 Zettabytes and the same data inflow and outflow is expected to be around 180 Zettabytes by the year 2025. This change of digitization has taken place due to a variety of usage of the internet these days. On the other hand, Forbes suggests that there will be the requirement of 150 Zettabytes [2] of real-time data processing will be required by the year 2025. These days social media platform has become the most convenient way to share thoughts and ideas these days because of the high popularity of social networking sites. The data type available online is of multiple types, sources, and formats [3]. Since these shared social networking sites are available to all, finally results in the generation of a high volume of data. Since this information is available to all, it provides an option for further debugging and extracting relevant information from these data for which Hadoop Eco-system [4] [5] is considered to be the best.

Since the people on the social media network are connected to each other directly or indirectly, the whole dataset creates a group network. The social media network consists of interconnection with each other with the primary or main nodes and the leaf nodes. The nodes in the network are connected with each other as a linked community on the social network where every day the people are connected with each other and contribute or share their thoughts and ideas. The additional feature of social networks is that people are connected with each other directly or indirectly. As the people on the network keep sharing their thoughts and ideas, the social network is considered as dynamic in nature. On any particular website like Facebook or Twitter, thousands of people create novel links or destroy them. Thus the links on these websites among people are temporary. Since the coverage is very much diversified in terms of location or people with a different mindset, it is considered a perfect platform for extracting the facts and trends about the connection and their relationship.

Data mining: It is considered the proven tool and technology to process the available data set with the intent of getting useful information that is not obvious but hidden in nature. This technique is intended to unearth the unidentified information available within the dataset. Since the majority of the data information available is unstructured in nature, the memory requirement is increasing extensively [6]. Also, the concept of multi-threading increases the requirement of memory usage while doing distributed data processing [7]. A lot of available data processing techniques like centrality measure is available to perform this task. This information can be local or global in nature. The available information can be small (small sample from the full dataset) or large in nature. As per requirement, a data processing technique is applied. For the large dataset, a technology which is known as Big Data is applied. It is a Data Intensive technology popularly being used in Information technology, data science, and related business. Big data is not only concentrated on the Hadoop-related problem, rather it is a proven mechanism where data is stored, processed, and the derived outcome for either direct usage or as feed data for further analysis. Open source tools and concepts like Map-Reduce [8] and spark [9] are the optimal solutions to solve the big data-related problem statement where as there are multiple developers who work on traditional Application programming interface (API) [10] based solutions to solve the big data related solution. Big data is considered a base for data processing, and sourcing for the target application along with a required conclusion derivative technique. Map-Reduce in Hadoop Eco-System creates a cluster to enable parallel data processing [11]. In the traditional database, the data is stored in rows and columns whereas in HDFS (Hadoop Distributed File System) the data is stored in a distributed fashion. Generally, traditional databases are secure in nature. In addition to having the capability of processing large datasets, the Hadoop Ecosystem is quite fault tolerant [12]. Hence, the transaction from Traditional databases to distributed databases following issues are likely to occur and have a resolution of the following.

- Maintain the meaning and integrity of the dataset
- Ensure to have data security for unauthorized access
- Data ownership along with privacy
- Maintain the relationship in the dataset

Measuring the centrality of nodes is a major activity in data science-related activity which requires a lot of computations possible in small graphs. Managing and extracting information from small graphs is easy and possible whereas for large graphs the centrality measurement of a node is too expensive [13]. Computation of the shortest path between vertices pair is the requirement of betweenness centrality and it is possible only in the case of small graphs. With the extension of data and graphs, it becomes time-consuming. The high volume of data comes with lots of computation complexity and it’s really tough to decrease the computational complexity [14]. This is applicable to all types of algorithms. Traditional methods which derive the outcome have millions of nodes existing in the graph. To analyze the multiple nodes in a single graph becomes practically impossible to be analyzed. The problem statement becomes worse when the work is done for the real-world network problem like a social network where everything is dynamic in nature and the direction keeps on changing very frequently [15]. New nodes keep on adding to the network whereas old nodes get detached frequently.

In the present work, we have suggested an approach to solving the identification of influential node issues for the dataset which is very big in nature by the implementation of Hadoop Eco-system, MySQL, Python, and Power BI. For the smaller set of data, Python is a very effective tool to use but for large datasets, python cannot be effective because of the noise and variety of data in the dataset. To make the data in a usable format, the data must be processed in Hadoop first and then data analysis to be done in Python, and finally the plotting to be done in PowerBI.

2. Historical Evidence

A lot of historical work is done in the same context. Oktey and Balkir [16] have worked on distance measurement in big data networks. Distance calculation was considered a key factor in the social network mining application and implementation. For example, centrality and clustering using the MapReduce parallel processing framework to powerfully and precisely evaluate the distance for the large networks by the suggestion of a network structure index on the map-reduce framework supposing that networks are undirected and are unweighted. Kang and Spiros have worked on the centrality algorithm. The centrality for the node was calculated for a very huge graph consisting of millions of nodes in the network with the implementation of centrality measures. Behnam worked on the analysis of the network with the target of getting the interconnection among the different nodes. This is also known as behavioral analysis. The hypothesis defined in his project was the ranked-based measurement for behavior. The influential rank identification was a major factor in the analysis and hypothesis establishment [17]. In his experiment, he has established that the influential node rank is higher than other neighboring nodes in the network. Both streaming and non streaming data are processed in Map-Reduce and Spark [18].

Considering the work done by the previous research scholars, their work majorly focuses on the identification of influential nodes using a new method with the implementation of a map-reduced framework. It was lean towards proposing a new method for the identification centrality measure or the node ranking. Additionally, it was directed toward working on a small network. To short out the problem statement and overcome the existing limitation, in our approach, the work has been carried out to implement the usage of traditional centrality algorithm on large social media networks using Apache Hadoop and Map-reduce framework [19] with known limitations. The map-reduce has the option of scaling up high-volume data processing in a very less time frame on the distributed data platform.

3. Materials and Methods

Following is the list of algorithms taken into consideration,

3.1 Degree Centrality

The degree centrality is termed as the immediate risk on the node for getting impacted by the inflow within the same network. Here is the discussion about the impact of overall network flow on any particular node. For example, on any social media network, overall data flow on Twitter is influenced by some specific nodes or person's presence on the network. A similar case can be seen for the virus infection on the human body where because of the presence of a certain virus, the overall body gets impacted. The degree centrality for any vertex v, for the graph G=(V, E) where |V| is termed as the number of vertices and |E| is the number of edges in the network is defined as:

$$CD(v) = degree(v) \quad (1)[20]$$

3.2 Betweenness Centrality

The betweenness centrality defines as the effective methodology for detecting the quantum of influence given by any particular node on the overall network by the flow of information on any graph. Generally, used for the identification of nodes that act as a bridge or connector from one part to another part of the graph ie calculating the shortest path between different pairs of nodes within the graph for any network.

$$c_b(v) = \Sigma s \neq v \neq t \epsilon v \frac{\sigma st(v)}{\sigma s_t} \quad (2)[20]$$

Where

σst: The total number of shortest paths from node s to t

$\sigma st(v)$: Number of those paths which pass through v

3.3 Naive Forecasting Model

The distance of a node is termed as the sum of the distances of any particular node from every other node in the network where as the closeness is termed as the sum of the shortest distance of any particular node from all other available nodes in the network. Hence, closeness can be considered as the reciprocal of fairness. The node which is more in the center of the network, the less the distance of the same node from other available nodes. Hence it can be said that the shortest path distance is a base tool for the identification of closeness centrality. But still, it is proven that this concept is not correct for all types of circumstances.

$$c(x) = \frac{1}{\Sigma_y d\,(x,y)} \quad (3)[20]$$

4. MapReduce and Hadoop

MapReduce is a framework for writing real-time applications in which the processing of large amounts of data in GBs or TBs is done on commodity hardware . The data processing is done in parallel in the clustered environment in a fault-tolerant manner. A MapReduce framework divides the whole work into jobs and these jobs the whole dataset into a small set of data and which is processed by the map tasks in a parallel manner[Fig:1]. Both Map and Reduce work in a sequential manner [21]. The MapReduce frame first breaks the dataset into a small segment [22] which is arranged by the maps, and is further processed by the Reduced task [Fig: 2]. There are two trackers existing known as Master Job Tracker and Task Tracker. This tracker exists in the MapReduce framework per cluster node. The MapReduce framework work in a cluster mode which consists of master and slave node. The master node is responsible for tracking and scheduling the task for slave nodes, monitoring the overall execution, and the task which remains unexecuted is re-executed [23]. Slave nodes execute the task as per direction from the master node. When the dataset is less in size it becomes easy to mine the dataset and conclude the result but with the increase in data size, the size of the network increases eventually making it difficult to mine the complete dataset and network and finally determine the final required result. Hence the traditional data mining approach does not work in the case of large datasets rather pattern matching is the key approach for data mining. The finding is required for big data research and practices and data mining. There are multiple data collection tools like Sebek [24], Honeywall [25], Nepenthes [26], Kojoney [27], HFlow [28] and Capture-HPC [29] are tools for collecting the real time data.In addition to this, other tools like Apache Kafka and Apache Flume [30] are widely used tools available for data collection. The Apache kafka and Flume are very effective and useful tool for batch data processing.

Therefore, the distributed frameworks technique for graph and data mining is becoming too popular these days. To handle large and big data, historical researchers propose the usage of a high-level programming model known as MapReduce which is used for processing high-volume data by the usage of parallel processing and distributed computing on large nodes or machine clusters which are designed on the master-slave architecture on commodity hardware. As part of development, there are two approaches either understand the model & then use it or directly use the framework. The main advantage of using the model which is popularly known as the MapReduce model where users focus on the maps and reduce functions without bothering with the implementation for data distribution, the same data is processed on multiple nodes in the secondary slot. The primary node assigns the job for the secondary node which is further processed on commodity hardware. Graph mining algorithm was developed by Kang for the Hadoop framework which included different approaches like finding the width of the network, and getting the interconnection between different connected components in a very large network graph. This activity was done with the generation of a matrix vector in which graph mining was done on very large graphs.

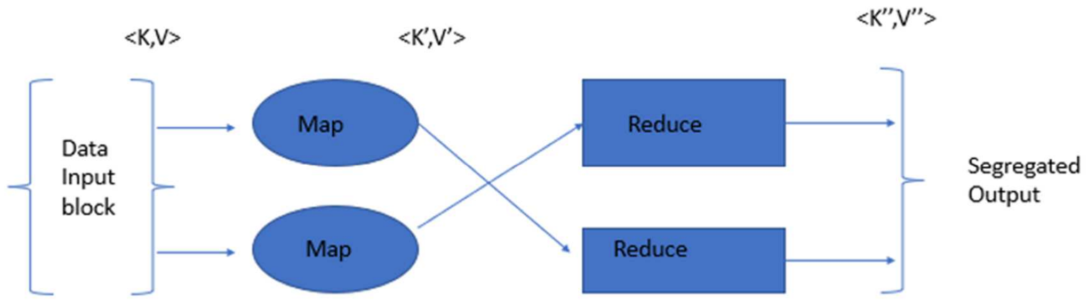


Figure 1. Computation within MapReduce

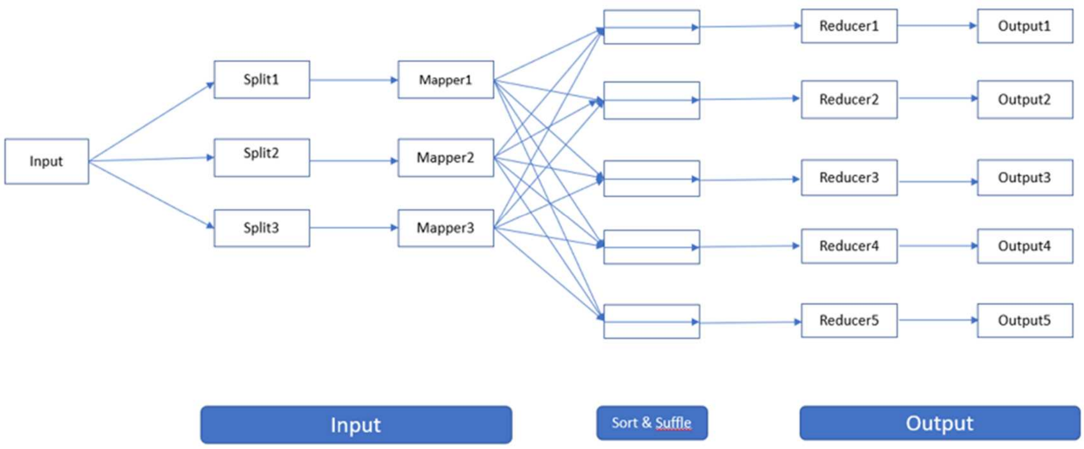


Figure 2. Mapper and Reducer Data splitting

Current Approach

As part of the experimental analysis, the Hadoop ecosystem setup was done on commodity hardware having the operating system Ubuntu 20 on a multi-node cluster. Each node was consisting of a 4GB Ram 2-core processor and 100 GB storage on each of the boxes. The whole setup was done on-prem infrastructure. Though there was the possibility of using cloud-based infrastructure on the Google Cloud Platform (GCP) or Amazon workspace (AWS) because of certain constraints finally activity was done on on-prem infrastructure.

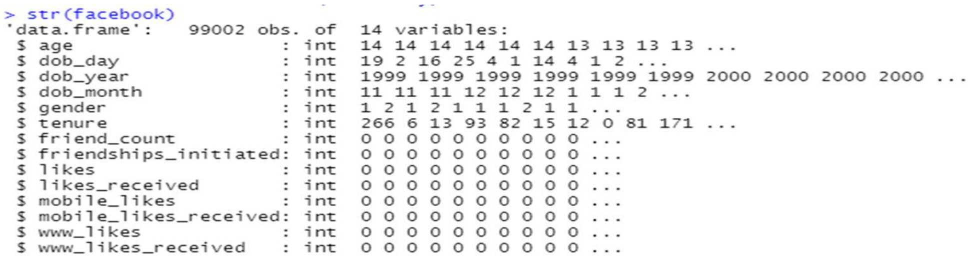
Data Processing: For experimental analysis, the test data was taken from Kaggle.

https://www.kaggle.com/datasets/sheenabatra/Facebook-data

The dataset consists of 99903 rows and 14 columns.

Dataset :

The data set is taken from the Kaggle website consisting of 99903 rows and 14 column network[Fig:3]. The network is designed from the available dataset. The whole network was designed with a combination of edges and a set of circles in the network. The circle was designed with nodes and edges that are undirected. The dataset was consisting of some noisy entries that are redundant in nature. These redundant entries were deleted from the available base dataset. The small set of networks collectively designs large networks Post removing the noisy data from the dataset, the normalization of data was done so as it is used by the algorithm of MapReduce. The dataset taken as CSV (Comma separated values) in nature so that it can be tokenized by MapReduce Framework.



```
> str(facebook)
'data.frame':   99002 obs. of  14 variables:
 $ age                  : int  14 14 14 14 14 14 13 13 13 13 ...
 $ dob_day              : int  19 2 16 25 4 1 14 4 1 2 ...
 $ dob_year             : int  1999 1999 1999 1999 1999 1999 2000 2000 2000 2000 ...
 $ dob_month            : int  11 11 11 12 12 12 1 1 1 2 ...
 $ gender               : int  1 2 1 2 1 1 1 2 1 1 ...
 $ tenure               : int  266 6 13 93 82 15 12 0 81 171 ...
 $ friend_count         : int  0 0 0 0 0 0 0 0 0 0 ...
 $ friendships_initiated: int  0 0 0 0 0 0 0 0 0 0 ...
 $ likes                : int  0 0 0 0 0 0 0 0 0 0 ...
 $ likes_received       : int  0 0 0 0 0 0 0 0 0 0 ...
 $ mobile_likes         : int  0 0 0 0 0 0 0 0 0 0 ...
 $ mobile_likes_received: int  0 0 0 0 0 0 0 0 0 0 ...
 $ www_likes            : int  0 0 0 0 0 0 0 0 0 0 ...
 $ www_likes_received   : int  0 0 0 0 0 0 0 0 0 0 ...
```

Figure 3. Facebook Data type

Centrality Algorithm Implementation

The <Key, Value> pair is the base for the implementation of the MapReduce framework ie interpretation of input data is done by the framework job as a set of <key,value > pair and the output is provided as <key,value> pair in the different form. For the analysis purpose, each of the entry keys is marked for Node ID, and distance measure is taken for value.

Algorithm 1: Degree Centrality algorithm

Degree Centrality algorithm.txt

Fig 4 below depicts the Degree centrality algorithm on the MapReduce framework. Here the data is taken in the input parameter in the <key,value> pair which is processed, and then further it is reduced by the reducer, and data is moved to the next stage.

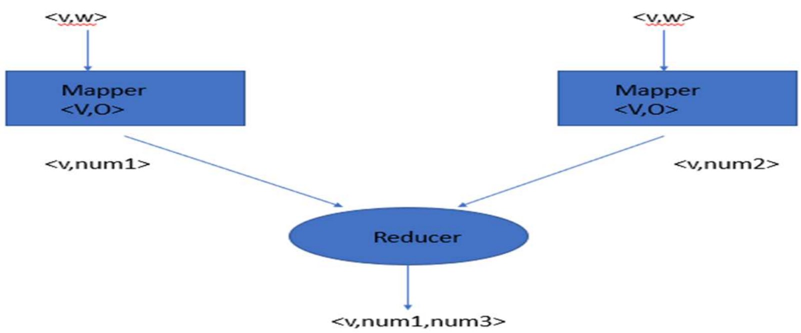


Figure 4. Degree centrality algorithm on MapReduce framework

Algorithm 2. Betweenness centrality Algorithm

Betweenness centrality is dependent on the entire graph and interdependency is lost with the division of data. The interdependencies are actually important to measure which is considered one of the most important limitations within the Hadoop ecosystem.

In the current analysis, the intention was to overcome the dependencies of interdependencies within the dataset by a minimum spanning tree. The minimum spanning tree help during the reducing phase by joining the common nodes and thus resolving the interdependent shortest paths.

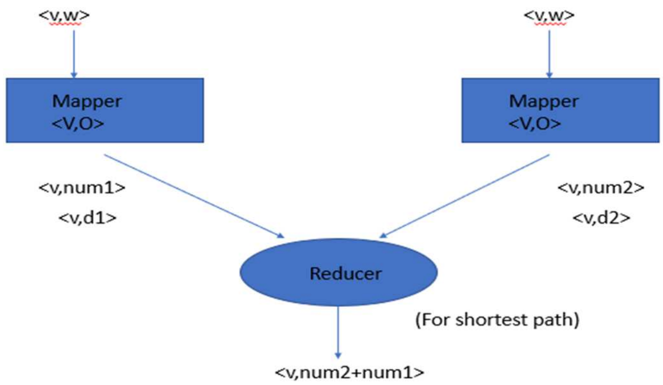


Figure 5. Execution of betweenness centrality algorithm on MapReduce framework

Algorithm 3: Closeness centrality Algorithm

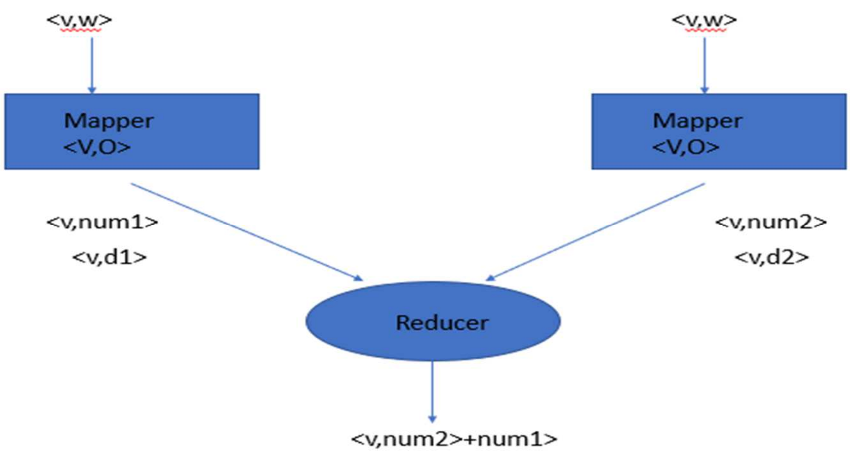


Figure 6. Execution of closeness centrality algorithm on MapReduce framework

The number of paths, distance and degree cannot be zero because the nodes inside the graph are interconnected to each other. Hence the best approach for the closeness centrality identification [Fig :6] is to find the reciprocal of distance.

5. Implementation

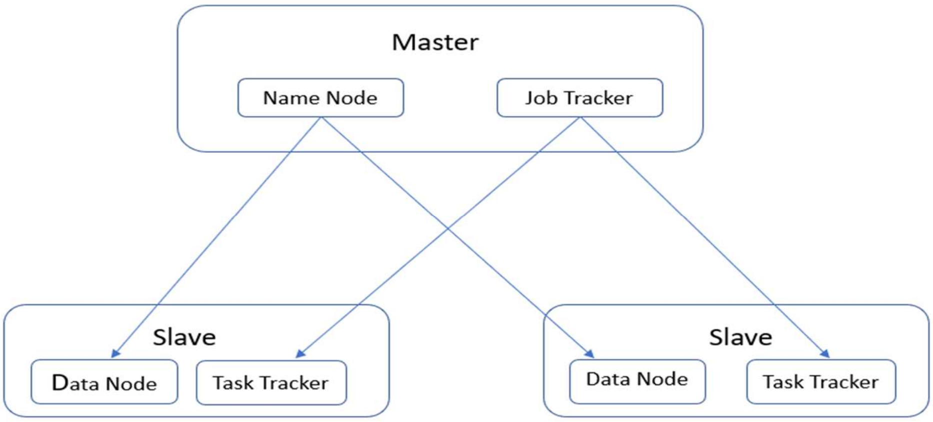


Figure 7. Hadoop Implementation Architecture

Hadoop has a core concept of Master and Slave. In the current implementation, One master node and two data nodes were created [Fig : 7]. The master node consists of Name Node and Job tracker where as the slave consists of Data Node and Task Tracker. The task of the Job tracker is to get the details of the task being created within the Hadoop Ecosystem.

Data Processing in Hadoop Eco-System :

In order to process the large dataset obtained from Kaggle about Facebook, the Multi-Node cluster setup for Hadoop 2.7.3 was done on GCP (Google Cloud Platform). In addition to Hadoop Eco-System, for data analysis Python and PyCharm were used. For the Presentation of the data, Microsoft Power BI was used. The whole setup was done on commodity hardware including open-source Java 11. For processing the data inside the Hadoop Eco-System, Apache Hive was used along with MySQL with JDBC data source. For the data movement before and after processing, Apache Sqoop was used which is able to do two-way data processing between Hadoop Eco-System and MySQL. All these setups were done with Ubuntu 20 with 4 core processors and 16GB RAM [31] with storage of 100 GB in Master as well as in the two slave machines.

Post Installation the setup was confirmed with the JPS command in Fig[8] stating all daemons are up and running without any issues.

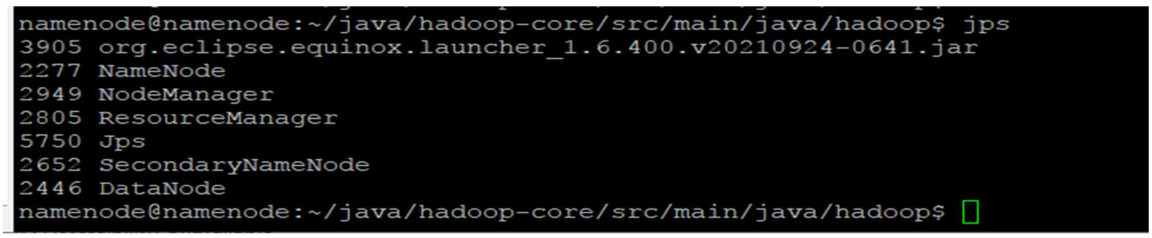
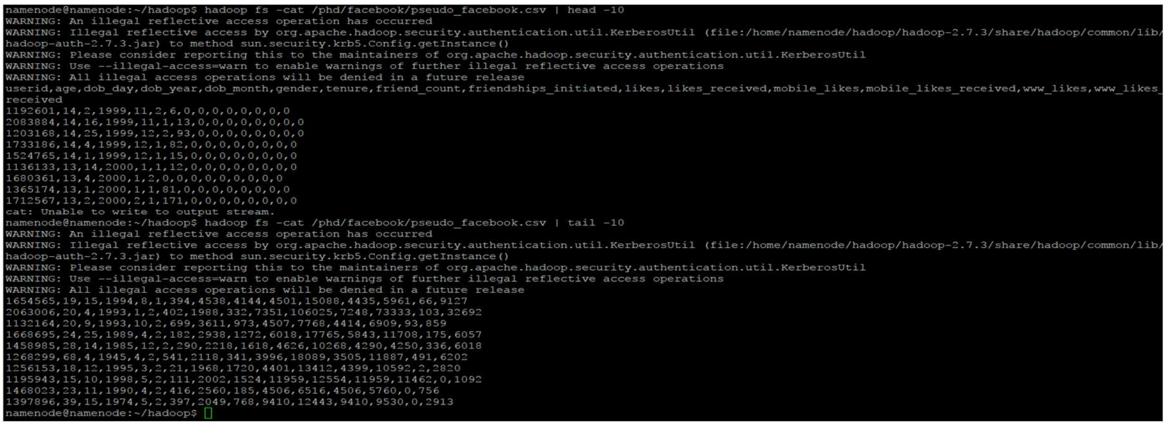


Figure 8. JPS command output

The file stored in the local file system was copied into the HDFS directory which was confirmed with the cat command inside the HDFS in Figure[9].



```
namenode@namenode:~/hadoop$ hadoop fs -cat /phd/facebook/pseudo_facebook.csv | head -10
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.authentication.util.KerberosUtil (file:/home/namenode/hadoop/hadoop-2.7.3/share/hadoop/common/lib/
hadoop-auth-2.7.3.jar) to method sun.security.krb5.Config.getInstance()
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.security.authentication.util.KerberosUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
userid,age,dob_day,dob_year,dob_month,gender,tenure,friend_count,friendships_initiated,likes,likes_received,mobile_likes,mobile_likes_received,www_likes,www_likes_
received
1192601,14,2,1999,11,2,6,0,0,0,0,0,0,0,0
2083884,14,16,1999,11,1,13,0,0,0,0,0,0,0,0
1203168,14,25,1999,12,2,93,0,0,0,0,0,0,0,0
1733186,14,4,1999,12,1,82,0,0,0,0,0,0,0,0
1524765,14,1,1999,12,1,15,0,0,0,0,0,0,0,0
1136133,13,14,2000,1,1,12,0,0,0,0,0,0,0,0
1680361,13,4,2000,1,2,0,0,0,0,0,0,0,0,0
1365174,13,1,2000,1,1,81,0,0,0,0,0,0,0,0
1712567,13,2,2000,2,1,171,0,0,0,0,0,0,0,0
cat: Unable to write to output stream.
namenode@namenode:~/hadoop$ hadoop fs -cat /phd/facebook/pseudo_facebook.csv | tail -10
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.authentication.util.KerberosUtil (file:/home/namenode/hadoop/hadoop-2.7.3/share/hadoop/common/lib/
hadoop-auth-2.7.3.jar) to method sun.security.krb5.Config.getInstance()
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.security.authentication.util.KerberosUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
1654565,19,15,1994,8,1,394,4538,4144,4501,15088,4435,5961,66,9127
2063006,20,4,1993,1,2,402,1988,332,7351,106025,7248,73333,103,32692
1132164,20,9,1993,10,2,699,3611,973,4507,7768,4414,6909,93,859
1668695,24,25,1989,4,2,182,2938,1272,6018,17765,5843,11708,175,6057
1458985,28,14,1985,12,2,290,2218,1618,4626,10268,4290,4250,336,6018
1268299,68,4,1945,4,2,541,2118,341,3996,18089,3505,11887,491,6202
1256153,18,12,1995,3,2,21,1968,1720,4401,13412,4399,10592,2,2820
1195943,15,10,1998,5,2,111,2002,1524,11959,12554,11959,11462,0,1092
1468023,23,11,1990,4,2,416,2560,185,4506,6516,4506,5760,0,756
1397896,39,15,1974,5,2,397,2049,768,9410,12443,9410,9530,0,2913
namenode@namenode:~/hadoop$
```

Figure 9. Head and Tail output after Data import in HDFS

Further, the hive table was created with the create table statement as below.

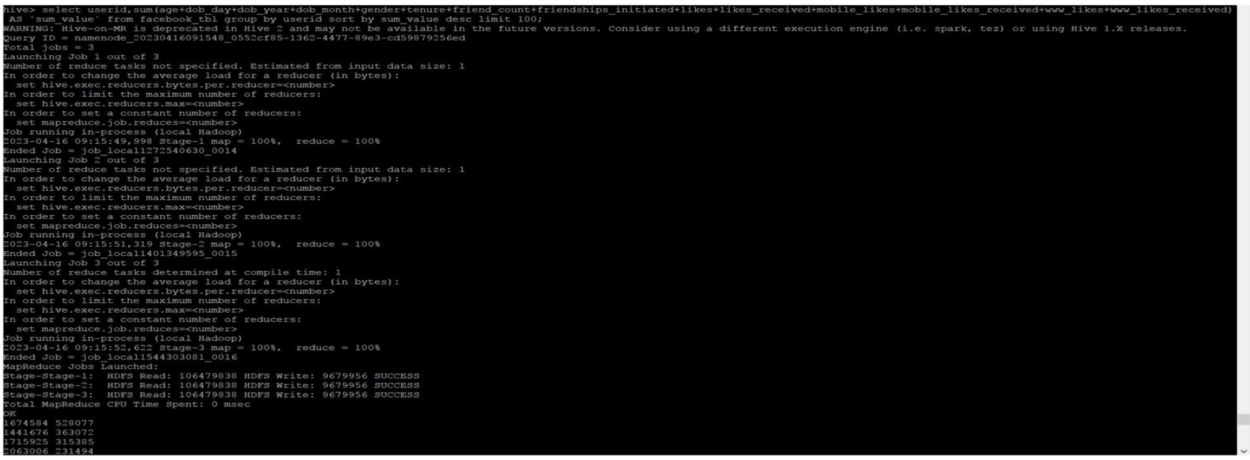
create external table facebook_tbl

(
userid int,age int,dob_day int,dob_year int,dob_month int,gender int,tenure int,friend_count int,friendships_initiated int,likes int,likes_received int,mobile_likes int,mobile_likes_received int,www_likes int,www_likes_received int
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;

Post creation of the table, the data was inserted in the hive table with the "Load Data" HiveQL command which was confirmed by the select statement in the hive table.

Further, the data was processed inside the hive with the select statement with sum function for 100 hundred nodes out of 99003 nodes in the whole data set with the below command in Figure[10].

hive> select userid,sum(age+dob_day+dob_year+dob_month+gender+tenure+friend_count+friendships_initiated+likes+likes_received+mobile_likes+mobile_likes_received+www_likes+www_likes_received) AS `sum_value` from facebook_tbl group by userid sort by sum_value desc limit 100;



```
hive> select userid,sum(age+dob_day+dob_year+dob_month+gender+tenure+friend_count+friendships_initiated+likes+likes_received+mobile_likes+mobile_likes_received+www_likes+www_likes_received)
 AS `sum_value` from facebook_tbl group by userid sort by sum_value desc limit 100;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = namenode_20230416091548_0552cf85-1362-4477-89e3-cd59879256ed
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-04-16 09:15:49,998 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1272540630_0014
Launching Job 2 out of 3
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-04-16 09:15:51,319 Stage-2 map = 100%,  reduce = 100%
Ended Job = job_local1401349595_0015
Launching Job 3 out of 3
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-04-16 09:15:52,622 Stage-3 map = 100%,  reduce = 100%
Ended Job = job_local1544303081_0016
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 106479838 HDFS Write: 9679956 SUCCESS
Stage-Stage-2:  HDFS Read: 106479838 HDFS Write: 9679956 SUCCESS
Stage-Stage-3:  HDFS Read: 106479838 HDFS Write: 9679956 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
1674584 528077
1441676 363072
1715925 315385
2063006 231494
```

Figure 10. Head and Tail output after Data import in HDFS

Once the sum is done as a sum_value variable, the data is inserted in another hive table facebook_processed table with the insert statement [32].

Hive> insert into facebook_processed(select userid,sum(age+dob_day+dob_year+dob_month+gender+tenure+friend_count+friendships_initiated+likes+likes_received+mobile_likes+mobile_likes_received+www_likes+www_likes_received) AS `sum_value` from facebook_tbl group by userid sort by sum_value desc limit 100);

```
hive> create external table facebook_processed
    > (
    > userid int,sum_value int
    > )
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
    > LINES TERMINATED BY '\n'
    > STORED AS TEXTFILE;
OK
Time taken: 0.08 seconds
```

Figure 11. External Table in hive post-processing

Post-processing the data in the hive table, the processed data was moved into the table facebook_processed for further analysis in MySQL.

```
hive> insert into facebook_processed (select userid,sum(age+dob_day+dob_year+dob_month+gender+tenure+friend_count+friendships_initiated+likes+likes_received+mobile_likes+mobile_likes_receiv
ed+www_likes+www_likes_received) AS `sum_value` from facebook_tbl group by userid sort by sum_value desc limit 100);
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = namenode_20230416092404_d756863e-acf2-4949-90ea-b82e7a3db8be
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-04-16 09:24:05,779 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1176628972_0017
Launching Job 2 out of 3
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-04-16 09:24:07,071 Stage-2 map = 100%,  reduce = 100%
Ended Job = job_local1700051765_0018
Launching Job 3 out of 3
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-04-16 09:24:08,388 Stage-3 map = 100%,  reduce = 100%
Ended Job = job_local431867362_0019
Loading data to table phd.facebook_processed
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 116159794 HDFS Write: 9679956 SUCCESS
Stage-Stage-2:  HDFS Read: 116159794 HDFS Write: 9679956 SUCCESS
Stage-Stage-3:  HDFS Read: 116159794 HDFS Write: 9681447 SUCCESS
```

Figure 12. Map-Reduce to insert data in the external table

After processing the data inside the hive table, the data is moved further in the MySQL table for further processing using the Apache Sqoop tool as shown in Figure[13]. Below Sqoop export statement was used for data movement from Hadoop to MySQL table which was confirmed by the select statement. For each of the process, inside Hadoop ecosystem, MapReduce function [33] is executed for mapping and reducing activity.

```
sqoop export --connect "jdbc:mysql://localhost:3306/phd" \
  --username root \
  --password ******** \
  --table facebook_processed \
  --export-dir hdfs://localhost:9000/user/hive/warehouse/phd.db/facebook_processed \
  --input-fields-terminated-by '|' \
  --input-lines-terminated-by '\n' \
  --num-mappers 2
```

```
namenode@namenode:/home/kishu010378/hadoop/hadoop-2.7.3/sbin$ sqoop export --connect "jdbc:mysql://localhost:3306/phd" \
>     --username root \
>     --password Rajan@123 \
>     --table facebook_processed \
>     --export-dir hdfs://localhost:9000/user/hive/warehouse/phd.db/facebook_processed \
>     --input-fields-terminated-by '|' \
>     --input-lines-terminated-by '\n' \
>     --num-mappers 2
Warning: /home/kishu010378/hadoop/sqoop-1.4.7/../hbase does not exist! HBase imports will fail.
Please set $HBASE_HOME to the root of your HBase installation.
Warning: /home/kishu010378/hadoop/sqoop-1.4.7/../hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /home/kishu010378/hadoop/sqoop-1.4.7/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /home/kishu010378/hadoop/sqoop-1.4.7/../zookeeper does not exist! Accumulo imports will fail.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
23/04/17 12:28:00 INFO sqoop.Sqoop: Running Sqoop version: 1.4.7
23/04/17 12:28:00 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
23/04/17 12:28:00 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
23/04/17 12:28:00 INFO tool.CodeGenTool: Beginning code generation
Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new driver class is `com.mysql.cj.jdbc.Driver'. The driver is automatically registered via the SPI and manual loading of the d
river class is generally unnecessary.
23/04/17 12:28:00 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `facebook_processed` AS t LIMIT 1
23/04/17 12:28:00 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `facebook_processed` AS t LIMIT 1
23/04/17 12:28:00 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /home/kishu010378/hadoop/hadoop-2.7.3
Note: /tmp/sqoop-namenode/compile/f71e95c351a244c1648e513cabd67851/facebook_processed.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
23/04/17 12:28:02 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-namenode/compile/f71e95c351a244c1648e513cabd67851/facebook_processed.jar
23/04/17 12:28:02 INFO mapreduce.ExportJobBase: Beginning export of facebook_processed
23/04/17 12:28:02 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
23/04/17 12:28:02 INFO Configuration.deprecation: mapred.jar is deprecated. Instead, use mapreduce.job.jar
23/04/17 12:28:03 INFO Configuration.deprecation: mapred.reduce.tasks.speculative.execution is deprecated. Instead, use mapreduce.reduce.speculative
23/04/17 12:28:03 INFO Configuration.deprecation: mapred.map.tasks.speculative.execution is deprecated. Instead, use mapreduce.map.speculative
23/04/17 12:28:03 INFO Configuration.deprecation: mapred.map.tasks is deprecated. Instead, use mapreduce.job.maps
23/04/17 12:28:03 INFO client.RMProxy: Connecting to ResourceManager at /127.0.0.1:8032
23/04/17 12:28:06 INFO input.FileInputFormat: Total input paths to process : 1
23/04/17 12:28:06 INFO input.FileInputFormat: Total input paths to process : 1
23/04/17 12:28:06 INFO mapreduce.JobSubmitter: number of splits:2
23/04/17 12:28:06 INFO Configuration.deprecation: mapred.map.tasks.speculative.execution is deprecated. Instead, use mapreduce.map.speculative
23/04/17 12:28:06 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1681734444738_0001
23/04/17 12:28:07 INFO impl.YarnClientImpl: Submitted application application_1681734444738_0001
23/04/17 12:28:07 INFO mapreduce.Job: The url to track the job: http://namenode:8088/proxy/application_1681734444738_0001/
23/04/17 12:28:07 INFO mapreduce.Job: Running job: job_1681734444738_0001
```

Figure 13. Sqoop Export activity

6. Actual Result Extraction and Analysis

Once the data is available inside MySQL, further processing is done in Python. As part of the activity, the Co-variance matrix, Eigen Value & Eigen Vector along with the following centrality measures were calculated for the top 100 nodes in the available dataset.

6.1 Eigen Value and Eigen Vector

Eigenvalue and eigenvector were computed for the top 100 Nodes available in the Facebook dataset using Python script [Fig:14].

```
D:\personal_data\Projects\Python\venv\Scripts\python.exe D:/personal_data/Projects/Python/venv/eigenvalue.py
Printing the final co-verriance matrix:
              0            1
0  1.102035e+11  5.644537e+08
1  5.644537e+08  4.590535e+09
Printing the Eigen values of the given square array:
 [1.10206483e+11 4.58751833e+09]
Printing Right eigenvectors of the given square array:
 [[ 0.99998572 -0.00534432]
 [ 0.00534432  0.99998572]]

Process finished with exit code 0
```

Figure 14. Eigen Value and Eigen Vector

Table 1. Betweenness Centrality for top 100 Nodes

Betweenness Centrality Value	node id	Betweenness Centrality Value	node id	Betweenness Centrality Value	node id	Betweenness Centrality Value	node id
0.004476327	0	0.005760743	25	0.008091899	50	0.006182517	75
0.004312992	1	0.006716126	26	0.004149246	51	0.006721459	76
0.004802972	2	0.00509878	27	0.006300464	52	0.004517673	77
0.004381601	3	0.006720242	28	0.006648596	53	0.005001688	78
0.004481542	4	0.003542665	29	0.003898949	54	0.005121525	79
0.005332263	5	0.003866528	30	0.003608417	55	0.005390086	80
0.004989913	6	0.003970053	31	0.006817155	56	0.00481694	81
0.005816029	7	0.007099378	32	0.006607189	57	0.005405561	82
0.004910355	8	0.004455137	33	0.006473682	58	0.004582856	83

0.00528855	9	0.003950528	34	0.004897233	59	0.004373483	84
0.004925876	10	0.005347055	35	0.004084775	60	0.00422479	85
0.005303227	11	0.003796582	36	0.00637585	61	0.004921447	86
0.004593215	12	0.00523771	37	0.003419892	62	0.004099452	87
0.003994046	13	0.004793117	38	0.005247611	63	0.004614786	88
0.005634864	14	0.005914435	39	0.003823754	64	0.00497586	89
0.006729404	15	0.004869796	40	0.004166861	65	0.004398654	90
0.005816907	16	0.005717724	41	0.003282443	66	0.003879844	91
0.005755496	17	0.005506186	42	0.003814427	67	0.004040443	92
0.003137765	18	0.004763371	43	0.004000215	68	0.003828644	93
0.003560787	19	0.005003289	44	0.004006745	69	0.005293799	94
0.003549137	20	0.00807948	45	0.00572517	70	0.005476698	95
0.003795295	21	0.003625728	46	0.003813489	71	0.00607612	96
0.005197688	22	0.003765082	47	0.004668499	72	0.004942767	97
0.005065982	23	0.006575232	48	0.004692801	73	0.005016539	98
0.005099556	24	0.007418954	49	0.006203377	74	0.007308706	99

6.2 Betweenness Centrality

For the Facebook dataset, betweenness centrality was calculated. It is considered as the significant mechanism for the detection of the influence of any particular node over the flow of information within the graph. The nodes which act as a bridge between one part of the graph to the other part of the graph and the betweenness centrality serve as the method of such node identification. The betweenness centrality algorithm help in calculating the shortest path between the pair of nodes within the graph. The higher value of betweenness centrality has more control over the network as the maximum information flow from that particular node.

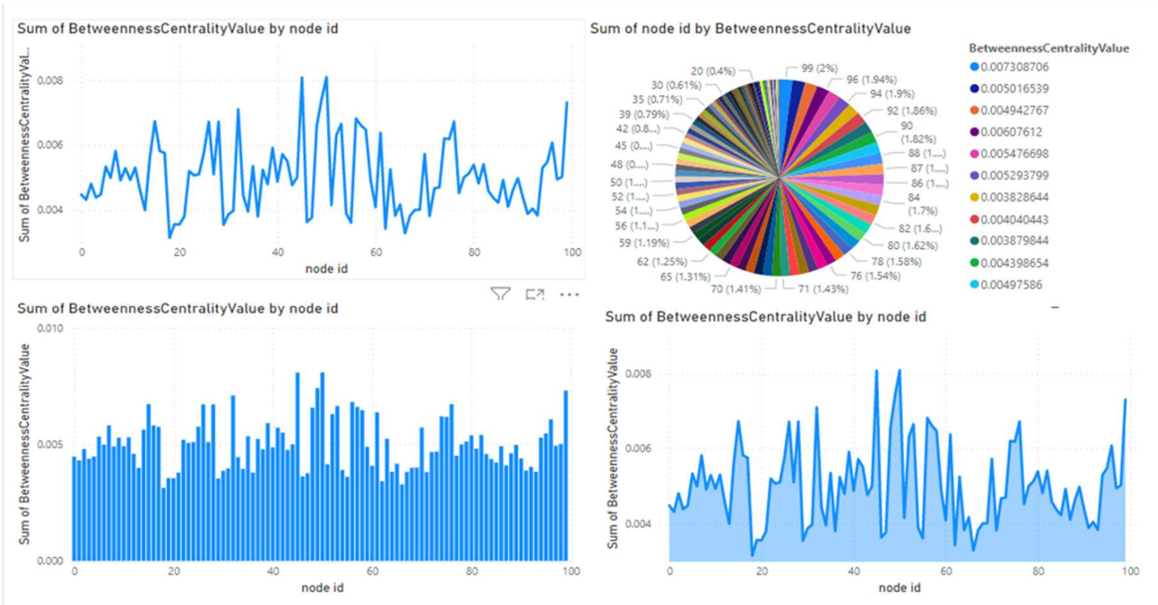


Figure 15. Betweenness Centrality BI Plot

Table :1 and Figure 15 show the betweenness centrality for the Facebook dataset where it is quite evident that there are few nodes for which the betweenness centrality value is higher as compared with the other nodes available in the data set. The higher centrality value of a node depicts the higher influence of that particular node in the dataset network.

6.3 Degree Centrality

The number of edges associated with any node is the degree of centrality of the node ie the higher no of nodes associated with any node, the more central that node will be. Additionally, the lesser number of nodes associated with any node the less central that node will be. This is considered as the most effective measure for the most influential nodes within the network.

Table 2. Degree Centrality for top 100 nodes

Degree Centrality Value	Node id	Degree Centrality value	Node id	Degree Centrality value	Node id	Degree Centrality value	Node id
0.424242424	0	0.545454545	25	0.606060606	50	0.494949495	75
0.535353535	1	0.505050505	26	0.404040404	51	0.434343434	76
0.545454545	2	0.535353535	27	0.535353535	52	0.414141414	77
0.484848485	3	0.464646465	28	0.585858586	53	0.484848485	78
0.505050505	4	0.434343434	29	0.404040404	54	0.505050505	79
0.444444444	5	0.484848485	30	0.515151515	55	0.626262626	80
0.474747475	6	0.535353535	31	0.515151515	56	0.484848485	81
0.484848485	7	0.484848485	32	0.555555556	57	0.474747475	82
0.393939394	8	0.565656566	33	0.474747475	58	0.525252525	83
0.464646465	9	0.525252525	34	0.575757576	59	0.545454545	84
0.545454545	10	0.494949495	35	0.505050505	60	0.505050505	85
0.444444444	11	0.484848485	36	0.505050505	61	0.474747475	86
0.505050505	12	0.555555556	37	0.535353535	62	0.494949495	87
0.464646465	13	0.414141414	38	0.444444444	63	0.575757576	88
0.464646465	14	0.565656566	39	0.474747475	64	0.555555556	89
0.464646465	15	0.474747475	40	0.545454545	65	0.525252525	90
0.474747475	16	0.555555556	41	0.545454545	66	0.595959596	91
0.535353535	17	0.494949495	42	0.585858586	67	0.444444444	92
0.585858586	18	0.585858586	43	0.494949495	68	0.585858586	93
0.484848485	19	0.505050505	44	0.454545455	69	0.505050505	94
0.434343434	20	0.606060606	45	0.474747475	70	0.585858586	95
0.525252525	21	0.464646465	46	0.565656566	71	0.484848485	96
0.484848485	22	0.565656566	47	0.575757576	72	0.474747475	97
0.515151515	23	0.464646465	48	0.525252525	73	0.525252525	98
0.565656566	24	0.505050505	49	0.484848485	74	0.505050505	99

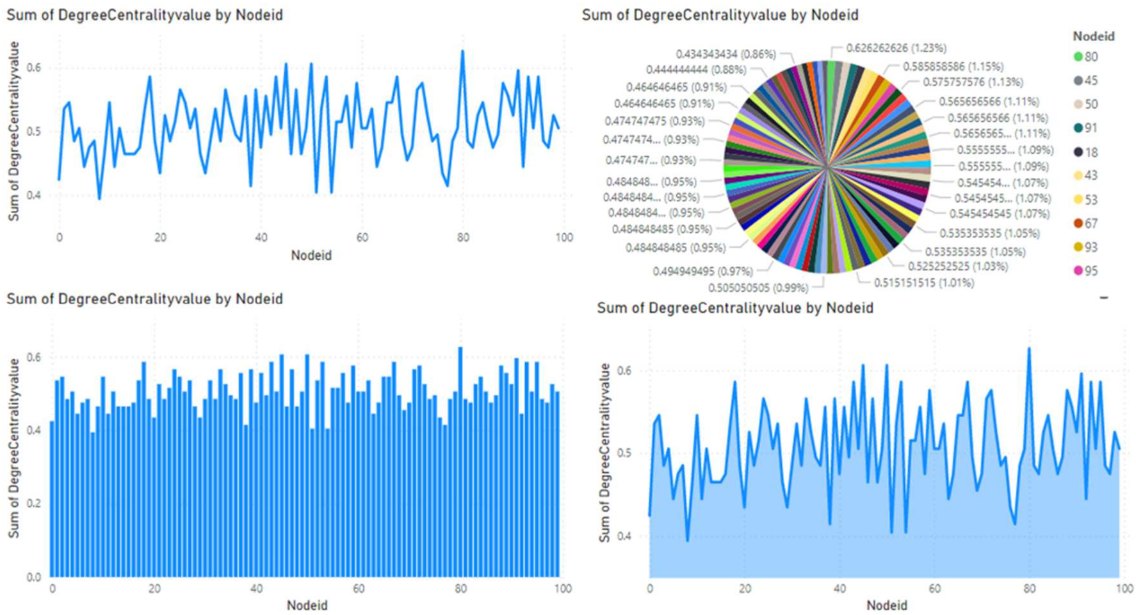


Figure 16. Degree centrality BI Plot

The table:2 and Figure 16 show the degree centrality for the Facebook dataset where it is quite evident that there are few nodes for which the degree centrality value is higher as compared with the other nodes available in the data set. The higher centrality value of a node depicts the higher influence of that particular node in the dataset network.

6.4 Closeness Centrality

The closeness centrality is a measurement of the shortest distance average in the number for each of the vertex from each other vertex in the network. The closeness centrality is the inverse of any particular vertex in comparison with all other vertexes in the network in the context of the average shortest distance.

Table 3. Closeness centrality for top 100 Nodes

Closeness centrality value	node id	Closeness centrality value	node id	Closeness centrality value	node id	Closeness centrality value	node id
0.744360902	0	0.70212766	25	0.761538462	50	0.712230216	75
0.707142857	1	0.733333333	26	0.712230216	51	0.697183099	76
0.727941176	2	0.727941176	27	0.678082192	52	0.75	77
0.673469388	3	0.75	28	0.727941176	53	0.717391304	78
0.761538462	4	0.744360902	29	0.733333333	54	0.707142857	79
0.697183099	5	0.744360902	30	0.722627737	55	0.707142857	80
0.755725191	6	0.717391304	31	0.692307692	56	0.673469388	81
0.75	7	0.682758621	32	0.682758621	57	0.744360902	82
0.727941176	8	0.697183099	33	0.722627737	58	0.727941176	83
0.6875	9	0.733333333	34	0.733333333	59	0.722627737	84
0.727941176	10	0.707142857	35	0.682758621	60	0.6875	85
0.722627737	11	0.755725191	36	0.707142857	61	0.733333333	86
0.722627737	12	0.707142857	37	0.761538462	62	0.697183099	87
0.717391304	13	0.755725191	38	0.697183099	63	0.733333333	88
0.722627737	14	0.707142857	39	0.717391304	64	0.70212766	89

Closeness centrality value	node id	Closeness centrality value	node id	Closeness centrality value	node id	Closeness centrality value	node id
0.733333333	15	0.712230216	40	0.6875	65	0.733333333	90
0.717391304	16	0.655629139	41	0.73880597	66	0.707142857	91
0.707142857	17	0.73880597	42	0.673469388	67	0.733333333	92
0.712230216	18	0.722627737	43	0.73880597	68	0.727941176	93
0.673469388	19	0.678082192	44	0.744360902	69	0.712230216	94
0.744360902	20	0.6875	45	0.73880597	70	0.70212766	95
0.682758621	21	0.733333333	46	0.712230216	71	0.707142857	96
0.792	22	0.6875	47	0.692307692	72	0.712230216	97
0.70212766	23	0.73880597	48	0.744360902	73	0.707142857	98
0.722627737	24	0.70212766	49	0.75	74	0.697183099	99

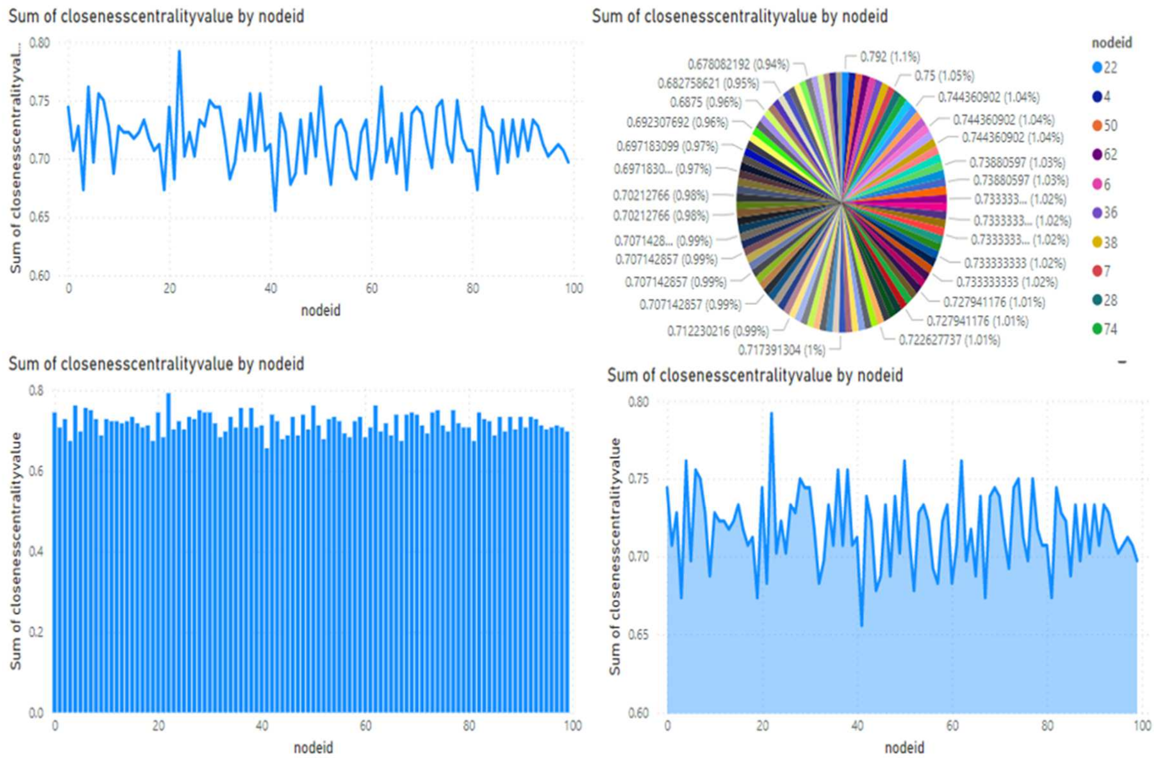


Figure 17. Closeness Centrality BI Plot

The table:3 and Figure 17 show the degree centrality for the Facebook dataset where it is quite evident that there are few nodes for which the closeness centrality value is higher as compared with the other nodes available in the data set. The higher centrality value of a node depicts the higher influence of that particular node in the dataset network.

Vol. 5, No. 1, June 2024, pp. 1-18

6.5 Individual and Collective Analysis of Facebook Dataset

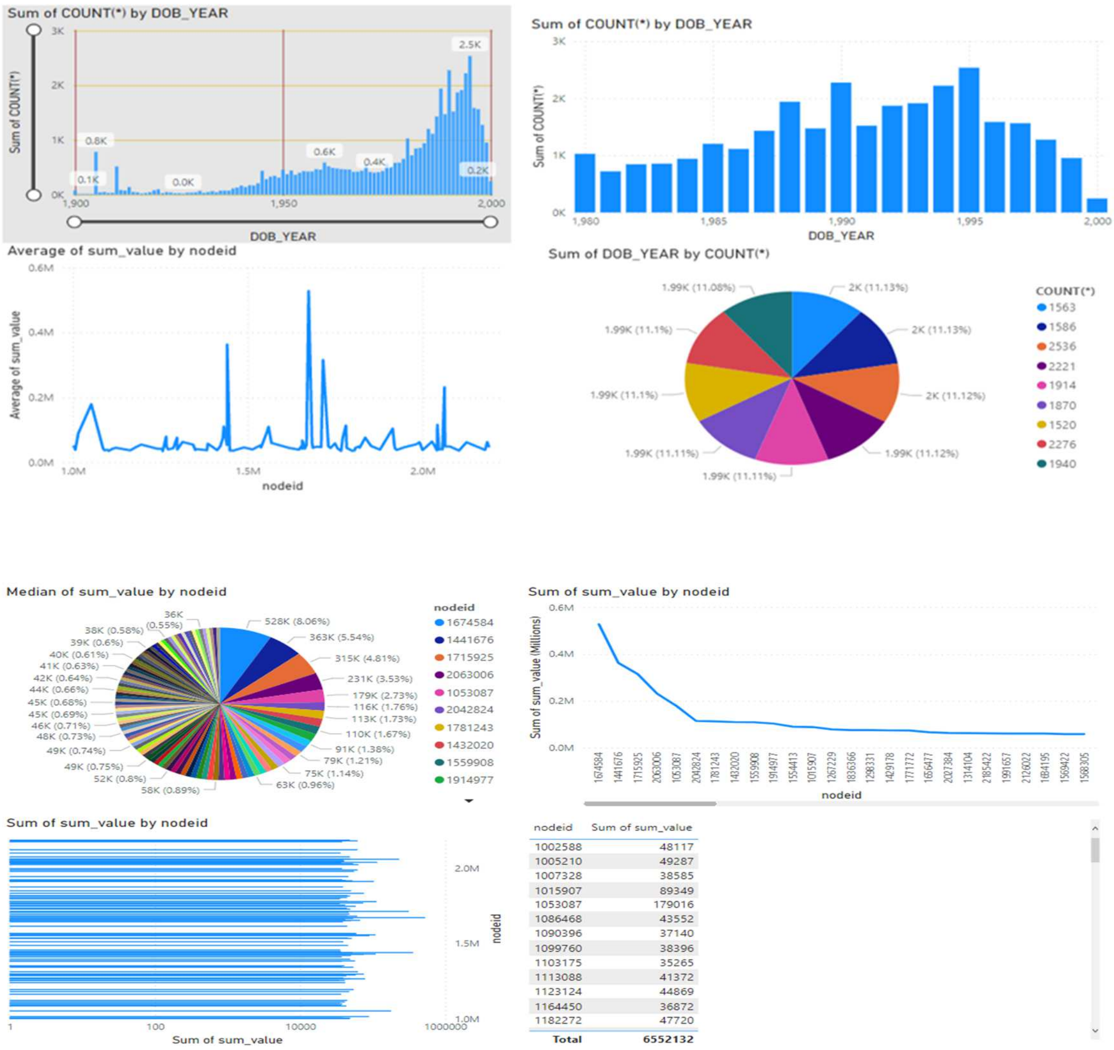


Figure 18. Closeness Centrality BI Plot

Using the Hadoop Eco-system and data processing inside the Apache Hive, the initial analysis of DOB year was done and further, the plot was done in Power BI clearly indicating that the maximum footprint for the activity was for the age group starting 1995 till 2000 whereas the footprint for the age group less than that is minimum. Also, some occurrences of data is seen that are earlier than that which is considered as the noisy data in the available dataset and has no significance in actual data. Hence these data were removed before the actual data analysis.

Additionally, in the last part of the analysis, all three observations, betweenness centrality, degree centrality, and closeness centrality were merged against the node id which shows a similar trend. But since in the current analysis, the focus was on the usage of the Hadoop eco-system in the data analysis which is majorly been used for processing the large dataset by using the RDBMS capability of MySQL.

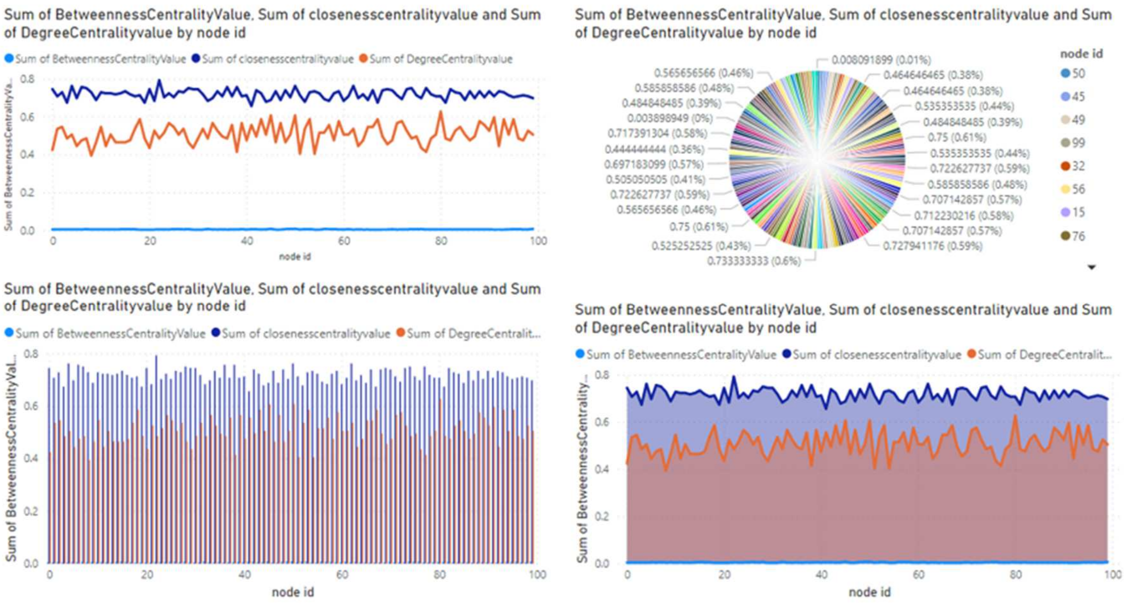


Figure 19. Closeness Centrality BI Plot

7. Conclusion and Future Scope of Activity

From the experimental analysis, historical evidence, and different BI plots, it is quite evident that for the dataset readiness and processing of a very large dataset, the Hadoop ecosystem is the best. Similar processing can be done in both traditional databases as well as in other tools like MS-SQL/Oracle but these come with some limitations like licenses and limitations of data processing. The Hadoop ecosystem can easily be integrated with RDBMS like MySQL which is again an open source for data storage. The Whole setup of the Hadoop Eco-System can be done on both On-Prem architecture as well as on a cloud-based system that can be accessed across the globe. The processing in the Hadoop system because of its architecture is very fast. Also, the plots and analysis using Power BI are very impressive and conclusive. Also, the data set extracted as part of different analyses can be merged to do a hand-to-hand comparison.

As part of the future scope of activity, similar analysis can be done on a more micro level and a different set of data using other open-source tools. Also, the experimental analysis can be leveraged not only to social media but other expects like health care, telecom, and other domain in addition to real-time online data like tracking of flight movement in open space using the Hadoop eco-system.

References

[1] Statista. Total Data Volume Worldwide 2010–2025. Available online: https://www.statista.com/statistics/871513/worldwide/data-created/.

[2] Forbes. Big Data Goes Big. Available online: https://www.forbes.com/sites/rkulkarni/2019/02/07/big-data-goes-big/?sh=5b9 85d0920d7.

[3] Bhosale, H.S.; Gadekar, D.P. A review paper on big data and Hadoop. IJSR 2014, 4, 1–7.

[4] SangeethaLakshmi, M.G.; Jayashree, M.M. Comparative Analysis of Various Tools for Data Mining and Big Data Mining. IRJET 2019, 6, 704–708.

[5] Apache Hadoop Home Page. Available online: https://hadoop.apache.org/.

[6] Wu, Y.; Wu, C.; Li, B.; Zhang, L.; Li, Z.; Lau, F.C. Scaling social media applications into geo-distributed clouds. IEEE ACM Trans. Netw. 2014, 23, 689–702.

[7] Zaharia, M.; Chowdhurry, M.; Das, T.; Dave, A.; Ma, J.; McCauley, M.; Franklin, J.M.; Shenker, S.; Stoica, I. Fast and interactive analytics over Hadoop data with Spark. Usenix Login 2012, 37, 45–51.

[8] Apache Hadoop. MapReduce Tutorial. Available online: https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/ hadoop-mapreduce-client-core/MapReduceTutorial.html.

[9] Apache SparkTM. Unified Engine for Large-Scale Data Analytics. Available online: https://spark.apache.org/.

[10] Ahmed, N.; Barczak, A.L.; Susnjak, T.; Rashid, M.A. A comprehensive performance analysis of Apache Hadoop and Apache Spark for large scale data sets using HiBench. J. Big Data 2020, 7, 110.

[11] Ahmadvand, H.; Goudarzi, M.; Foroutan, F. Gapprox: Using gallup approach for approximation in big data processing. J. Big Data 2019, 6, 20.

[12] Samadi, Y.; Zbakh, M.; Tadonki, C. Performance comparison between Hadoop and Spark frameworks using HiBench benchmarks. Concurr. Comput. Pract. Exp. 2018, 30, e4367.

[13] Isah, H.; Abughofa, T.; Mahfuz, S.; Ajerla, D.; Zulkernine, F.; Khan, S. A survey of distributed data stream processing frameworks. IEEE Access 2019, 7, 154300–154316.

[14] Khezr, S.N.; Navimipour, N.J. MapReduce and its applications, challenges, and architecture: A comprehensive review and directions for future research. J. Grid Comput. 2017, 15, 295–321

[15] Aziz, K.; Zaidouni, D.; Bellafkih, M. Big data processing using machine learning algorithms: Mllib and mahout use case. In proceedings of the 12th International Conference on Intelligent Systems: Theories and Applications, Rabat, Morocco, 24–25 October 2018, 1st ed.; Association for Computing Machinery: New York, NY, USA, 2018; pp. 1–6.

[16] H. Oktay, A. S. Balkir, I. Foster, and D. Jensen "Distance estimation with MapReduce for large networks", in Proceedings of the Workshop on Information Networks, WIN, pp. 1-6, 2011

[17] Mavrogiorgou, A.; Kiourtis, A.; Kyriazis, D. Plug 'n'play IoT devices: An approach for dynamic data acquisition from unknown heterogeneous devices. In Proceedings of the Conference on Complex, Intelligent, and Software Intensive Systems, Turin, Italy, 10–13 July 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 885–895.

[18] Goudarzi, M. Heterogeneous architectures for big data batch processing in mapreduce paradigm. IEEE Trans. Big Data 2017, 5, 18–33.

[19] Koo, J.; Kang, G.; Kim, Y.G. Security and privacy in big data life cycle: A survey and open challenges. Sustainability 2020, 12, 10571.

[20] Prerna, Agarwal.; Rafeeq, Ahmed.;Tanvir, Ahmad.; Identification and ranking of key persons in a Social Networking Website using Hadoop & Big Data Analytics AICTC '16: Proceedings of the International Conference on Advances in Information Communication Technology & ComputingAugust 2016Article No.: 65Pages 1–6https://doi.org/10.1145/2979779.2979844

[21] Perakis, K.; Miltiadou, D.; De Nigro, A.; Torelli, F.; Montandon, L.; Magdalinou, A.; Mavrogiorgou, A.; Kyriazis, D. Data Sources and Gateways: Design and Open Specifification. Acta Inform. Med. 2019, 27, 341.

[22] Mavrogiorgou, A.; Kiourtis, A.; Kyriazis, D. A pluggable IoT middleware for integrating data of wearable medical devices. Smart Health 2022, 26, 100326.

[23] Anderson, J.W.; Kennedy, K.E.; Ngo, L.B.; Luckow, A.; Apon, A.W. Synthetic data generation for the internet of things. In Proceedings of the 2014 IEEE International Conference on Big Data (Big Data), Washington, DC, USA, 27–30 October 2014; pp. 171–176.

[24] Sebek Homepage. Available online: https://honeynet.onofri.org/tools/sebek/.

[25] Honeynet. Hflflow2. Available online: https://www.honeynet.org/projects/old/hflflow2/.

[26] Viecco, C. Improving honeynet data analysis. In Proceedings of the 2007 IEEE SMC Information Assurance and Security Workshop, West Point, NY, USA, 20–22 June 2007; pp. 99–106.

[27] Honeynet. Nepenthes Pharm. Available online: https://www.honeynet.org/2009/11/29/nepenthes-pharm/.

[28] Kojoney—A Honeypot for the SSH Service. Available online: http://kojoney.sourceforge.net/.Information 2023, 14, 93 32 of 34

[29] Honeynet. Capture-HPC. Available online: https://www.honeynet.org/projects/old/capture-hpc/.

[30] Apache Kafka Home Page. Available online: https://kafka.apache.org/.

[31] Padgavankar, M.H.; Gupta, S.R. Big data storage and challenges. Int. J. Comput. Sci. Inf. Technol. 2014, 5, 2218–2223.

[32] Hypertable.org Home Page. Available online: https://hypertable.org/.

[33] Khezr, S.N.; Navimipour, N.J. MapReduce and its applications, challenges, and architecture: A comprehensive review and directions for future research. J. Grid Comput. 2017, 15, 295–321