BIG DATA ANALYTICS

LAB MANUAL

(C87PC7)

List of Experiments

S.No.	Experiment Name
1	Installation of Hadoop Framework, it's components and study the HADOOP ecosystem.
2	Write a program to implement word count program using MapReduce
3	Experiment on Hadoop Map-Reduce / PySpark: -Implementing simple algorithms in Map-Reduce: Matrix multiplication.
4	Install and configure MongoDB/ Cassandra/ HBase/ Hypertable to execute NoSQL Commands.
5	Implementing DGIM algorithm using any Programming Language/ Implement Bloom Filter using any programming language
6	Implement and Perform Streaming Data Analysis using flume for datacapture, PYSpark / HIVE for data analysis of twitter data, chat data, weblog analysis etc.
7	Implement any one Clustering algorithm (K-Means/CURE) using Map-Reduce.
8	Implement Page Rank Algorithm using Map-Reduce.

Experiment No. 1

Aim: Installation of Hadoop Framework, it's components and study the HADOOPecosystem

Hadoop is an open-source framework that allows to store and process big data in a distributed environment across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering localcomputation and storage.

Hadoop Architecture:

The Apache Hadoop framework includes following four modules:

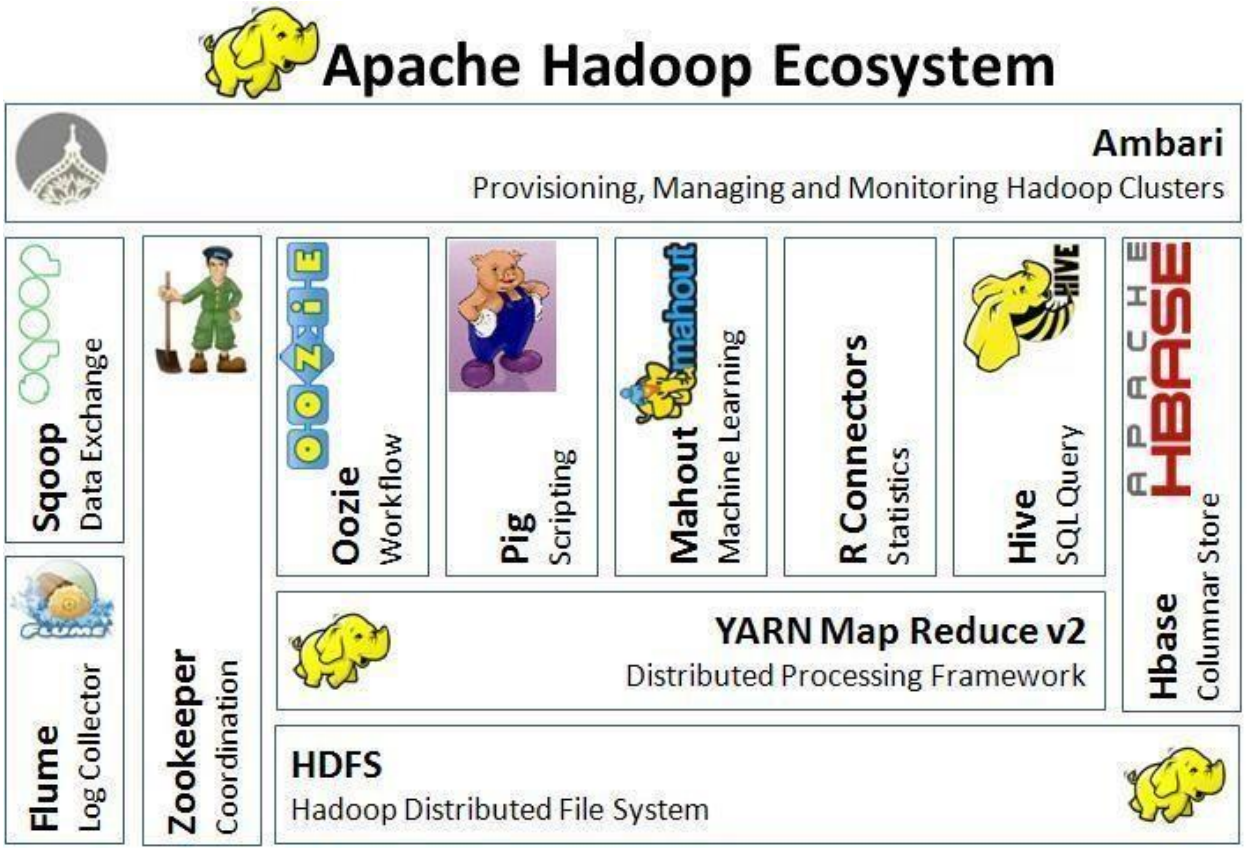
Hadoop Common: Contains Java libraries and utilities needed by other Hadoop modules. These libraries give file system and OS level abstraction and comprise of the essential Java files and scripts that are required to start Hadoop.

Hadoop Distributed File System (HDFS): A distributed file-system that provides high-throughput access to application data on the community machines thus providing very high aggregate bandwidth across the cluster.

Hadoop YARN: A resource-management framework responsible for job scheduling and cluster resource management.

Hadoop MapReduce: This is a YARN- based programming model for parallel processing of large data sets.

Hadoop Ecosystem:



Hadoop has gained its popularity due to its ability of storing, analyzing and accessing large amount of data, quickly and cost effectively through clusters of commodity hardware. It won'tbe wrong if we say that Apache Hadoop is actually a collection of several components and notjust a single product.

With Hadoop Ecosystem there are several commercial along with an open source products which are broadly used to make Hadoop laymen accessible and more usable.

MapReduce

Hadoop MapReduce is a software framework for easily writing applications which process big amounts of data in-parallel on large clusters of commodity hardware in a reliable, fault-tolerantmanner. In terms of programming, there are **two functions** which are most common in MapReduce.

- **The Map Task:** Master computer or node takes input and convert it into divide it into smaller parts and distribute it on other worker nodes. All worker nodes solve their own small problem and give answer to the master node.
- **The Reduce Task:** Master node combines all answers coming from worker node and forms it in some form of output which is answer of our big distributed problem.

Generally both the input and the output are reserved in a file-system. The framework is responsible for scheduling tasks, monitoring them and even re-executes the failed tasks.

Hadoop Distributed File System (HDFS)

HDFS is a distributed file-system that provides high throughput access to data. When data is pushed to HDFS, it automatically splits up into multiple blocks and stores/replicates the data thus ensuring high availability and fault tolerance.

Here are the **main components of HDFS:**

- **Name Node:** It acts as the master of the system. It maintains the name system i.e.,directories and files and manages the blocks which are present on the Data Nodes.
- **Data Nodes:** They are the slaves which are deployed on each machine and provide theactual storage. They are responsible for serving read and write requests for the clients.
- **Secondary Name Node:** It is responsible for performing periodic checkpoints. In theevent of Name Node failure, you can restart the Name Node using the checkpoint.

Hive

Hive is part of the Hadoop ecosystem and provides an SQL like interface to Hadoop. It is a data warehouse system for Hadoop that facilitates easy data summarization, ad-hoc queries, and the analysis of large datasets stored in Hadoop compatible file systems.

HBase (Hadoop DataBase)

HBase is a distributed, column oriented database and uses HDFS for the underlying storage. As said earlier, HDFS works on write once and read many times pattern, but this isn't a case

always. We may require real time read/write random access for huge dataset; this is where HBase comes into the picture. HBase is built on top of HDFS and distributed on column-oriented database.

Installation Steps –

Following are steps to Install Apache Hadoop on Ubuntu 14.04

Step1. Install Java (OpenJDK) - Since hadoop is based on java, make sure you have java jdk installed on the system. Please check the version of java (It should be 1.7 or above it)

```
$ java –version
```

If it returns "The program java can be found in the following packages", If Java isn't been installed yet, so execute the following command:

```
$sudo apt-get install default-jdk
```

Step2: Configure Apache Hadoop

1. Open bashrc in gedit mode

```
$sudo gedit ~/.bashrc
```

2. Set java environment variable

```
export JAVA_HOME=/usr/jdk1.7.0_45/
```

3. Set Hadoop environment variable

```
export HADOOP_HOME=/usr/Hadoop 2.6/
```

4. Apply environment variables

```
$source ~/.bashrc
```

Step3: Install eclipse

Step4: Copy Hadoop plug-ins such as

- **hadoop-eclipse-kepler-plugin-2.2.0.jar**
- **hadoop-eclipse-kepler-plugin-2.4.1.jar**
- **hadoop-eclipse-plugin-2.6.0.jar** from **release** folder of **hadoop2x-eclipse-plugin-master to eclipse plugins**

Step5: In **eclipse,** start new **MapReduce project**

```
File->new->other->MapReduce project
```

Step 6: Copy Hadoop packages such as **commons-io-2.4.jar commons-lang3-3.4.jar** in **src** file of MapReduce project

Step 7: Create Mapper, Reducer, and driver

Inside a project->src->File->new->other->Mapper/Reducer/Driver

Step 8: Copy Log file **log4j.properties** from **src** file of hadoop in src file of MapReduce project

Hadoop is powerful because it is extensible and it is easy to integrate with any component. Itspopularity is due in part to its ability to store, analyze and access large amounts of data, quicklyand cost effectively across clusters of commodity hardware. Apache Hadoop is not actually a single product but instead a collection of several components. When all these componentsare merged, it makes the Hadoop very user friendly.

Experiment No. 2

Aim: Implementing distinct word count problem using Map-Reduce

The function of the mapper is as follows:

- Create a IntWritable variable 'one' with value as 1
- Convert the input line in Text type to a String
- Use a tokenizer to split the line into words
- Iterate through each word and a form key value pairs as
  Assign each work from the tokenizer (of String type) to a Text 'word'
- Form key value pairs for each word as < word,one > and push it to the output collector

The function of Sort and Group:

After this, "aggregation" and "Shuffling and Sorting" done by framework. Then Reducers task these final pair to produce output.

The function of the reducer is as follows

- Initialize a variable 'sum' as 0
- Iterate through all the values with respect to a key and sum up all of them
- Push to the output collector the Key and the obtained sum as value

For Example:

For the given sample input1 data file (input1.txt : Hello World Bye World) mapper emits:
<Hello, 1>
<World, 1>
<Bye, 1>
<World, 1>

The second input2 data file (input2.txt : Hello Hadoop Goodbye Hadoop) mapper emits:
<Hello, 1>
<Hadoop, 1>
<Goodbye, 1>
<Hadoop, 1>

WordCount also specifies a combiner. Hence, the output of each map is passed through the local combiner (which is same as the Reducer as per the job configuration) for local aggregation, after being sorted on the keys.

The output of the first map:

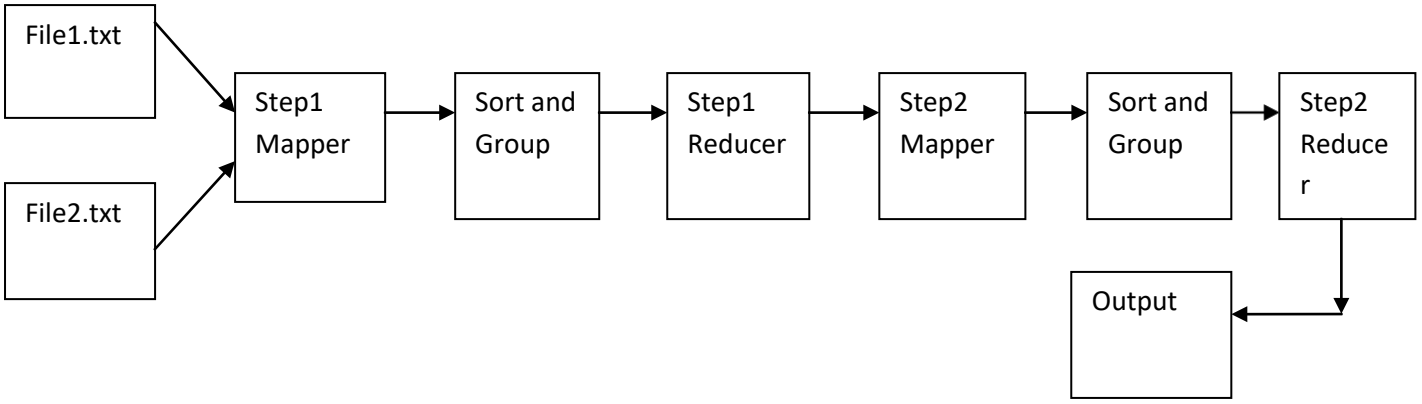
<Bye, 1>
<Hello, 1>
<World, 2>

The output of the second map:
<Goodbye, 1>
<Hadoop, 2>
<Hello, 1>

The Reducer implementation via the reduce method just sums up the values, which are the occurence counts for each key (i.e. words in this example).

Thus the output of the job is:
<Bye, 1>
<Goodbye, 1>
<Hadoop, 2>
<Hello, 2>
<World, 2>

Data Flow Diagram:



Experiment No. 3

Aim: Implementation of Matrix Multiplication using MapReduce.

Definitions:

M is a matrix with element m_{ij} in row i and column j.

N is a matrix with element n_{jk} in row j and column k.

P is a matrix = MN with element p_{ik} in row i and column k, where $p_{ik} = \sum_j m_{ij} n_{jk}$

Mapper function does not have access to the i, j, and k values directly. An extra MapReduce Job has to be run initially in order to retrieve the values.

The Map Function:

For each element m_{ij} of M, emit a key-value pair (i, k), (M, j, m_{ij}) for k = 1, 2, . . .number of columns of N.

For each element n_{jk} of N, emit a key-value pair (i, k), (N, j, n_{jk}) for i = 1, 2, . . . number of rows of M.

The Reduce Function:

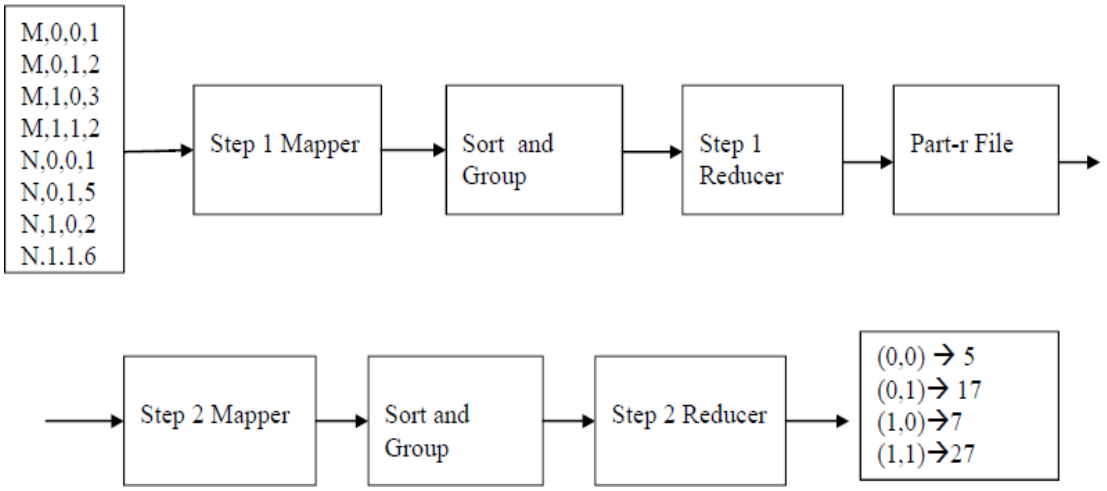
For each key (i, k), emit the key-value pair (i, k), p_{ik} where, $P_{ik} = \sum_j m_{ij} * n_{jk}$

The product MN is almost a natural join followed by grouping and aggregation.That is, the natural join of M(I, J, V) and N(J,K,W), having only attribute J in common, would produce tuples (i, j, k, v,w) from each tuple(i, j, v) in M and tuple (j, k,w) in N.

This five-component tuple represents the pair of matrix elements (mij ,njk).What we want instead is the product of these elements, that is, the four-component tuple (i, j, k, v × w), becausethat represents the product mijnjk. Once we have this relation as the result of one MapReduceoperation, we can perform grouping and aggregation, with me and K as the grouping attributes and the sum of V × W as the aggregation. That is, we can implement matrix multiplication as the cascade of two MapReduce operations, as follows.

Flow of entire matrix multiplication using map-reduce

Flow Diagram



The input file contains two matrices M and N. The entire logic is divided into two parts:

Step1: Find the product.

Step2: Find sum of the products.

1. Algorithm

Algorithm 1: The Map Function

1 **for** *each element* m_{ij} *of* M **do**
2 produce $(key, value)$ pairs as $((i,k),(M,j,m_{ij}))$, for $k = 1,2,3,..$ up to the number of columns of N
3 **for** *each element* n_{jk} *of* N **do**
4 produce $(key, value)$ pairs as $((i,k),(N,j,n_{jk}))$, for $i = 1,2,3,...$ up to the number of rows of M
5 **return** *Set of (key, value) pairs that each key,* (i,k)*, has a list with values* (M,j,m_{ij}) *and* (N,j,n_{jk}) *for all possible values of* j

Algorithm 2: The Reduce Function

1 **for** *each key (i,k)* **do**
2 sort values begin with M by j in $list_M$
3 sort values begin with N by j in $list_N$
4 multiply m_{ij} and n_{jk} for j_{th} value of each list
5 sum up $m_{ij} * n_{jk}$
6 **return** $(i,k), \sum_{j=1} m_{ij} * n_{jk}$

Experiment No. 4

Aim: Install and Configure MongoDB to execute NoSQL Commands.

Hardware / Software Required: MongoDB

A NoSQL (originally referring to "non SQL" or "non-relational") database provides a mechanism for storage and retrieval of data which is modeled in means other than the tabular relations used in relational databases. Relational databases were not designed to cope with the scale and agility challenges that facemodern applications, nor were they built to take advantage of the commodity storage and processing power available today.

The Benefits of NoSQL

- When compared to relational databases, NoSQL databases are more scalable and provide superior performance, and their data model addresses several issues that the relational model is not designed to address:
- Large volumes of rapidly changing structured, semi-structured, and unstructured data.
- Agile sprints, quick schema iteration, and frequent code pushes.
- Object-oriented programming that is easy to use and flexible.
- Geographically distributed scale-out architecture instead of expensive, monolithic architecture.

MongoDB

It is an open-source document database, and leading NoSQL database. MongoDB is written in c++. It is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

MongoDB Installation steps (Ubuntu 18.04):

Step 1 — Update system

sudo apt-get update

Step 2 — Installing and Verifying MongoDB
Now we can install the MongoDB package itself.

sudo apt-get install -y mongodb

This command will install several packages containing latest stable version of MongoDB

along with helpful management tools for the MongoDB server.

After package installation MongoDB will be automatically started. You can check this by running the following command.

sudo service mongod status

If MongoDB is running, you'll see an output like this (with a different process ID).

Output

mongod start/running, process 1611
You can also stop, start, and restart MongoDB using the service command (e.g. service mongod stop, service mongod start).

Commands

MongoDB ***use DATABASE_NAME*** is used to create database. The command will create a new database, if it doesn't exist otherwise it will return the existing database.

Syntax:
Basic syntax of **use DATABASE** statement is as follows:

use DATABASE_NAME

The dropDatabase() Method

MongoDBdb.dropDatabase() command is used to drop a existing database.

Syntax:

Basic syntax of dropDatabase() command is as follows:

db.dropDatabase()

This will delete the selected database. If you have not selected any database, then it will delete default 'test' database

The createCollection() Method

MongoDBdb.createCollection(name, options) is used to create collection.

Syntax:

Basic syntax of createCollection() command is as follows

db.createCollection(name, options)

In the command, name is name of collection to be created. Options is a document and

used to specify configuration of collection

The find() Method

To query data from MongoDB collection, you need to use MongoDB'sfind() method.

Syntax

Basic syntax of find() method is as follows

```
>db.COLLECTION_NAME.find().
```

MongoDB'supdate() and save() methods are used to update document into a collection. The update() method update values in the existing document while the save() method replaces the existing document with the document passed in save() method.

MongoDBUpdate() method

The update() method updates values in the existing document.

Syntax

Basic syntax of update() method is as follows

```
>db.COLLECTION_NAME.update(SELECTIOIN_CRITERIA, UPDATED_DATA)
```

Experiment No. : 5

Aim: Write a program to implement bloom filtering

Bloom Filter is a data structure that can do this job.

For understanding bloom filters, you must know what is hashing. A hash function takes inputand outputs a unique identifier of fixed length which is used for identification of input.

What is Bloom Filter?

A Bloom filter is a space-efficient probabilistic data structure that is used to test whether an element isa member of a set. For example, checking availability of username is set membership problem, where the set is the list of all registered username. The price we pay for efficiency is that it is probabilistic in nature that means, there might be some False Positive results.

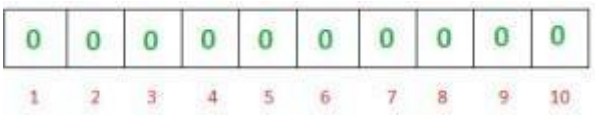
False positive means, it might tell that given username is already taken but actually it's not.

Interesting Properties of Bloom Filters

- Unlike a standard hash table, a Bloom filter of a fixed size can represent a set with an arbitrarily large number of elements.
- Adding an element never fails. However, the false positive rate increases steadily as elements are added until all bits in the filter are set to 1, at which point all queries yield a positive result.
- Bloom filters never generate **false negative** result, i.e., telling you that a username doesn't exist when it actually exists.
- Deleting elements from filter is not possible because, if we delete a single element by clearing bits at indices generated by k hash functions, it might cause deletion of few other elements. Example – if we delete "geeks" (in given example below) by clearingbit at 1, 4 and 7, we might end up deleting "nerd" also Because bit at index 4 becomes 0 and bloom filter claims that "nerd" is not present.

Working of Bloom Filter

A empty bloom filter is a **bit array** of **m** bits, all set to zero, like this –



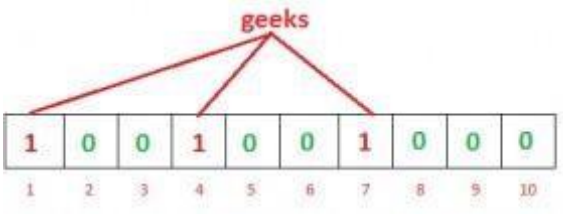
We need **k** number of **hash functions** to calculate the hashes for a given input. When wewant to add an item in the filter, the bits at k indices h1(x), h2(x), … hk(x)are set, where indices are calculated using hash functions.

Example – Suppose we want to enter "geeks" in the filter, we are using 3 hash

functions and a bit array of length 10, all set to 0 initially. First we'll calculate the hashes as following :

```
h1("geeks")   % 10 = 1
h2("geeks")   % 10 = 4
h3("geeks")   % 10 = 7
```

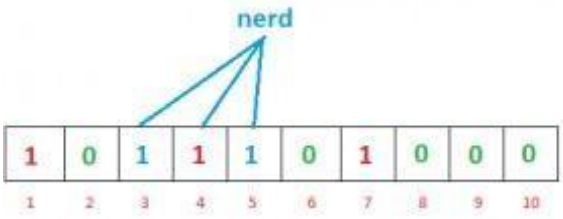
Note: These outputs are random for explanation only.
Now we will set the bits at indices 1, 4 and 7 to 1



Again we want to enter "nerd", similarly we'll calculate hashes

```
h1("nerd")   % 10 = 3
h2("nerd")   % 10 = 5
h3("nerd")   % 10 = 4
```

Set the bits at indices 3, 5 and 4 to 1



Now if we want to check "geeks" is present in filter or not. We'll do the same process but this Time in reverse order. We calculate respective hashes using h1, h2 and h3 and check if all theseindices are set to 1 in the bit array. If all the bits are set thenwe can say that "geeks" is **probably present**. If any of the bit at these indices are 0 then "geeks" is **definitely not present**.

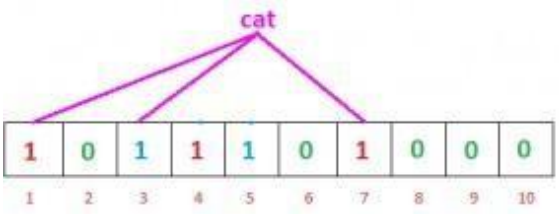
False Positive in Bloom Filters

The question is why we said **"probably present"**, why this uncertainty. Let's understand this with an example. Suppose we want to check whether "cat" is presentor not. We'll calculate hashes using h1, h2 and h3

```
h1("cat")   % 10 = 1
h2("cat")   % 10 = 3
h3("cat")   % 10 = 7
```

If we check the bit array, bits at these indices are set to 1 but we know that "cat" wasnever added to the filter. Bit at index 1 and 7 was set when we added "geeks" and bit3 was set we

added "nerd".



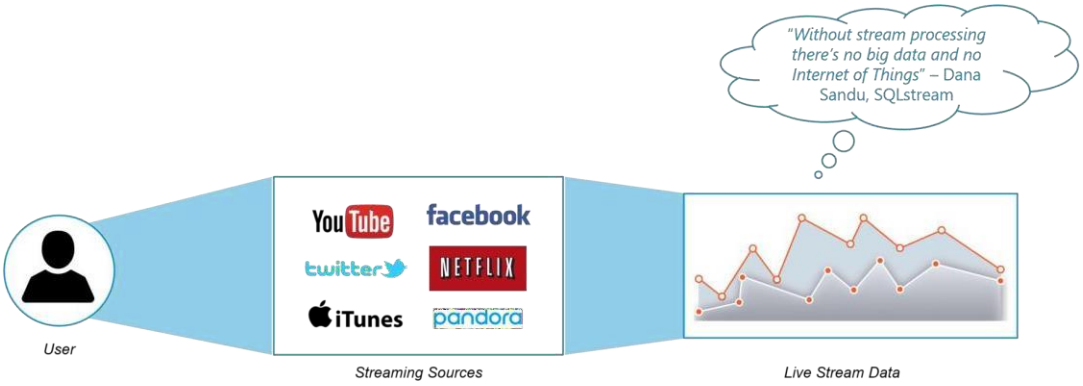
So, because bits at calculated indices are already set by some other item, bloom filter erroneously claim that "cat" is present and generating a false positive result. Depending on theapplication, it could be huge downside or relatively okay.

We can control the probability of getting a false positive by controlling the size of the Bloom filter. More space means fewer false positives. If we want decrease probability of false positive result, we have to use more number of hash functionsand larger bit array. This would add latency in addition of item and checking membership.

Experiment No. 6

Aim: Write a program to implement Twitter Sentiment Analysis System where we populate real-time sentiments for crisis management, service adjusting and target marketing using PYSpark

Data Streaming is a technique for transferring data so that it can be processed as a steady and continuous stream. Streaming technologies are becoming increasingly important with the growth of the Internet.

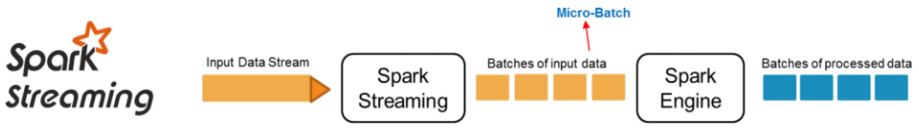


Why Spark Streaming?

We can use Spark Streaming to stream real-time data from various sources like Twitter, Stock Market and Geographical Systems and perform powerful analytics to help businesses.

Spark Streaming Overview

Spark Streaming is used for processing real-time streaming data. It is a useful addition to the core Spark API. Spark Streaming enables high-throughput and fault-tolerant streamprocessing of live data streams.



Spark Streaming Features

1. **Scaling:** Spark Streaming can easily scale to hundreds of nodes.
2. **Speed:** It achieves low latency.
3. **Fault Tolerance:** Spark has the ability to efficiently recover from failures.
4. **Integration:** Spark integrates with batch and real-time processing.
5. **Business Analysis:** Spark Streaming is used to track the behavior of customers which can be used in business analysis.

Spark Streaming Fundamentals

1. Streaming Context
2. DStream
3. Caching
4. Accumulators, Broadcast Variables and Checkpoints

Streaming Context

Streaming Context consumes a stream of data in Spark. It registers an *Input DStream* to produce a *Receiver* object. It is the main entry point for Spark functionality. Spark provides a number of default implementations of sources like Twitter, Akka Actor and ZeroMQ that are accessible from the context.

DStream

Discretized Stream (DStream) is the basic abstraction provided by Spark Streaming. It is a continuous stream of data. It is received from a data source or a processed data stream generated by transforming the input stream.

Caching

DStreams allow developers to cache/ persist the stream's data in memory. This is useful if the data in the DStream will be computed multiple times. This can be done using the *persist()* method on a DStream

Accumulators, Broadcast Variables and Checkpoints

Accumulators: *Accumulators* are variables that are only added through an associative and commutative operation. They are used to implement counters or sums. Tracking

accumulators in the UI can be useful for understanding the progress of running stages. Spark natively supports numeric accumulators. We can create named or unnamed accumulators.

Broadcast Variables: *Broadcast variables* allow the programmer to keep a read-only variable cached on each machine rather than shipping a copy of it with tasks. They can be used to give every node a copy of a large input dataset in an efficient manner. Spark also attempts to distribute broadcast variables using efficient broadcast algorithms to reducecommunication cost.

Checkpoints: *Checkpoints* are similar to checkpoints in gaming. They make it run 24/7 and make it resilient to failures unrelated to the application logic.

Applications of Sentiment Analysis:

- Predict the success of a movie
- Predict political campaign success
- Decide whether to invest in a certain company
- Targeted advertising
- Review products and services

Experiment No. 7

Aim: Write a program to implement K-means Clustering algorithm using MapReduce

Data clustering is the partitioning of a data set or sets of data into similar subsets. During the process of data clustering a method is often required to determine how similar one object or groups of objects is to another. This method is usually encompassed by some kind of distance measure. Data clustering is a common technique used in data analysis and is used in many fields including statistics, data mining, and image analysis. There aremany types of clustering algorithms. Hierarchical algorithms build successive clusters using previously defined clusters. Clustering algorithms can alsobe partitioned meaning they determine all clusters at once.

K-Means Algorithm

K-means clustering is a classical clustering algorithm that uses an expectationmaximization like technique to partition a number of data points into k clusters. K-means clustering is commonly used for a number of classification applications. Because k-means is run on such large data sets, and because of certain characteristics of the algorithm, it is a good candidate for parallelization.

In this problem, we have considered inputs a set of n 1-dimensional points and desired clusters of size 3.

Once the k initial centers are chosen, the distance is calculated (Euclidean distance) from every point in the set to each of the 3 centers & point with the corresponding center is emitted by the mapper. Reducer collect all of the points of a particular centroid and calculate a new centroid and emit.

Termination Condition:
When difference between old and new centroid is less than or equal to 0.1

1. Algorithm

1. Initially randomly centroid is selected based on data. (e,g. 3 centroids)
2. The Input file contains initial centroid and data.
3. Mapper is used to first open the file and read the centroids and store in the data structure(we used ArrayList)
4. Mapper read the data file and emit the nearest centroid with the point to the reducer.
5. Reducer collect all this data and calculate the new corresponding centroids and emit.

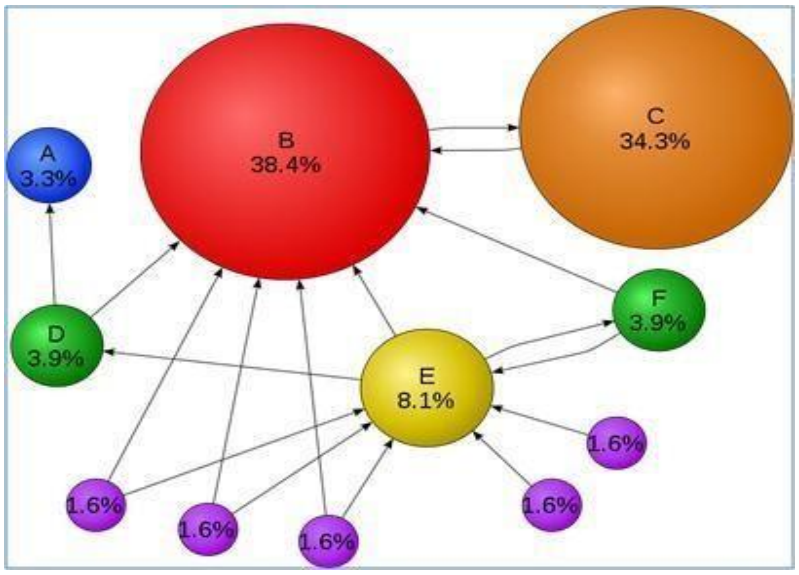
6. In the job configuration, we are reading both files and checking
   if
      difference between old and new centroid is less than 0.1 then
      convergence is reached
   else
         repeat step 2 with new centroids.

Experiment No. 8

Aim: Implement Page Rank algorithm using Map Reduce.

PageRank, is a way of measuring the importance of website pages. PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.

PageRank is a link analysis algorithm and it assigns a numerical weighting to each elementof a hyperlinked set of documents, such as the World Wide Web, with the purposeof measuring its relative importance within the set. The algorithm may be applied to any collection of entities with reciprocal quotations and references. The numerical weight that it assigns to any given element E is referred to as the PageRank of E and denoted by PR(E).Other factors like Author Rank can contribute to the importance of the family. A hyperlinkto a page counts as a vote of support. The PageRank of a page is defined recursively and depends on the number and PageRank metric of all pages that link to it(―incoming links‖).A page that is linked to by many pages with high PageRank receives a high rank itself.



Mathematical PageRank for a

simple network**Formula**

Eqn.1 is the formula to calculate the rank value for each webpage. We will learn this formula by applying it to the case in Fig.1. There are 11 webpages in Fig.1, which include:
{A, B, C, D, E, F, G1, G2, G3, G4, G5}. Assuming the probability distribution for a websurfer accessing all these 11 pages in current iteration is {PR(A), PR(B), PR(C), … PR(G5)}, then the probability for the surfer to access Page B in the next iteration is:

PR(B) = PR(D)/2 + PR(E)/3 + PR(F)/2 + PR(C) + PR(G1)/2 + PR(G2)/2 + PR(G3)/2

In a general case, the PageRank value for any page *u* can be expressed as:

$$PR(u) = \sum_{v \in Set} \frac{PR(v)}{L(v)}$$ Eqn.1

The vertices seen in the right of the formula contain all the webpages that point to target webpage _u'. The L(v) refers to the out degree of each webpage in the vertices set. The initial rank values of each webpage, like PR'(u), can be any double value. After several iteration calculations, the rank values converge to the stationary distribution regardless of what their initial values are.

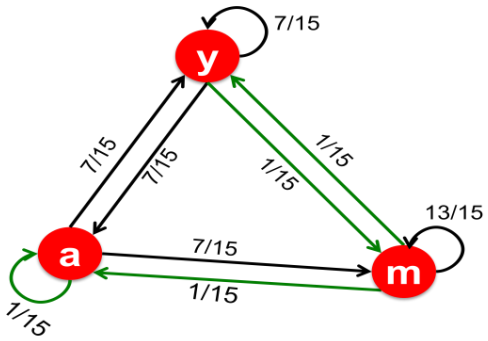
Damping factor

The PageRank theory holds that even an imaginary surfer who is randomly clicking on links will eventually stop clicking. The probability, at any step, that the person will continueis a damping factor **d**. Various studies have tested different damping factors, but it is generally assumed that the damping factor will be around 0.85. The formula considering damping factor is shown in Eqn.2. N refers to the total number of uniqueurls.

$$PR(u) = \frac{1-d}{N} + d * \sum v \in Set \frac{PR(v)}{L(v)}$$

..... Eqn.2

Example:



$$0.8 \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} + 0.2 \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$$

M $\quad$ $[1/N]_{NxN}$

$$\begin{array}{c} \\ y \\ a \\ m \end{array} \begin{bmatrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 13/15 \end{bmatrix}$$

A

$$\begin{array}{c} y \\ a \\ m \end{array} = \begin{array}{ccccc} 1/3 & 0.33 & 0.24 & 0.26 & \\ 1/3 & 0.20 & 0.20 & 0.18 & \ldots \\ 1/3 & 0.46 & 0.52 & 0.56 & \end{array} \begin{array}{c} 7/33 \\ 5/33 \\ 21/33 \end{array}$$