ANALYSING TWITTER DATA USING HADOOP

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

Piyush Rahi(141314)

Manjeet Choudhary(143201)

Under the supervision of

Mrs. Ruchi Verma (Assistant Professor)

to

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

Candidate's Declaration

We hereby declare that the work presented in this report entitled **" Analysing Twitter Data Using Hadoop"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2016 to December 2016 under the supervision of **(Mrs. Ruchi Verma)** (**Assistant Professor (**CSE).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Piyush Rahi(141314)
Manjeet Choudhary(143201)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Mrs. Ruchi Verma
Assistant Professor
CSE
Dated:

ACKNOWLEDGEMENT

We take this opportunity to express our profound gratitude and deep regards to our guide Mrs. Ruchi Verma(Assistant Professor ,CSE) for her exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by her time to time shall carry us a long way in the journey of life on which we are about to embark.

We are also obliged to staff members of JUIT College, for the valuable information provided by them in their respective fields. We are grateful for their cooperation during the period of our assignment.

Lastly, we thank almighty, our parents and our classmates for their constant encouragement without which this assignment would not have been possible.

ABSTRACT

Big Data has got important preference over the few years because of the amount of data generated every year. The role of big data comes into use when we need to store huge amount of data which normal database are not able to process and store. The challenge in Big Data is not only storing the data, but also accessing and analyzing the required data in specified amount of time. The problems of big data they are basically solved using Hadoop. Hadoop is an open source platform and uses MapReduce programming model for solving problems of Big Data. Hadoop provides the service to users to store and process bulk amount of data which normal databases are not able to provide.

Twitter one of the largest social media network receives huge amount of data. This data is measured in Zettabyte per year. This project provides a way of analyzing big data such as twitter data using Apache Hadoop which will process and analyze tweets on Hadoop clusters. The terms used in this project are- Big Data, Hadoop, MapReduce.

Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction to Project

We live in a society and many people used social site where the textual data on the Internet is growing at a rapid pace and many companies are trying to use this flood of data to extract people's views towards their products. Micro blogging today has become a very prevalent communication tool in to Internet users. Twitter, one of the largest social media site and user tweet millions of tweets every day on deferent of important topic. Authors of those messages write about their life, share opinions on variety of issues and discuss current issues. These posts analysis can be used for decision making in different fields like Business, Elections, Product review, government, etc.

HADOOP

The Apache Hadoop project develops open-source software for scalable, reliable, distributed computing. The Apache Hadoop library is a framework that allows for the distributed processing of large data sets beyond clusters of computers using a thousands of computational independent computers and large amount (terabytes, petabytes) of data. Hadoop was derived from Google File System (GFS) and Google's Map Reduce. Apache Hadoop is good choice for twitter analysis as it works for distributed huge data. Apache Hadoop is an open source framework for distributed storage and large scale distributed processing of data-sets on clusters. Hadoop runs applications using the MapReduce algorithm, where the data is processed in parallel on different clusters nodes. In short, Hadoop framework is able enough to develop applications able of running on clusters of computers and they could perform complete statistical analysis for a huge amounts of data. Hadoop MapReduce is a software framework [2] for easily writing applications which process big amounts of data in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.

APACHE FLUME

Apache Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of streaming data into the Hadoop Distributed File System (HDFS). It can be used for dumping twitter data in Hadoop HDFS.

APACHE PIG

Apache Pig is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. The salient property of Pig programs is that their structure is amenable to substantial parallelization, which in turns enables them to handle very large data sets.

1.2 Problem Statement

Social media is one of the popular media right now to share opinions or variety of topics and twitter is very popular social site to share every thing related to opinions on variety of topics and discussions on current issues. These tweets generates the huge information related to different area like government, election, etc. millions of tweets is generated every day and which is very useful in decision making because every one is share their view and opinions on issues or variety of topics. Twitter sites receives petabytes of data every day and these data is nothing but a collection of tweets so these data is very important in real life to analyse different scenario through which its helps us in decision making. The analysis of twitter data gives real view or different user opinions regarding what they think and to analysis these data provide a better way for making any decision.

1.3 Objective

The main scope of the project is to analyzing and fetching the Twitter IDs of those users whose statuses have been retweeted the most by the user whose tweets are being analyzed. First the system involves collecting the tweets from the social network using the twitter API's. Then second, this consists of standard platform as Hadoop to solve the challenges of big data through MapReduce framework where the complete data is mapped to frequent datasets and reduced to smaller sizable data to ease of handling. And finally includes analysing the collected tweets and fetching the Twitter IDs of those users whose statuses have been retweeted the most by the user whose tweets are being analysed.

Here MapReduce is the heart of Hadoop and each job is performed by this technique only. It is this programming paradigm that allows for massive scalability across hundreds or thousands of servers in a Hadoop cluster. The term MapReduce refers to two tasks in Hadoop i.e. Map and Reduce. In the first step, it takes the data and converts it into another set of data. Here the each word is referred as key and the number of occurrences is treated as value. So MapReduce tuple consists of key value pairs. Then second step consists of reduce operation where it takes the output from a map as input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce job is always performed after the map job.

1.4 Methodology

As we have seen the procedure how to overcome the problem that we are facing in the existing problem that is shown clearly in the proposed system.

So, to achieve this we are going to follow the following methods:

- Creating Twitter Application.
- Getting data using Flume.
- Querying using Hive Query Language (HQL)

1.4.1 Creating Twitter Application

First of all if we want to do sentiment analysis on Twitter data we want to get Twitter data first so to get it we want to create an account in Twitter developer and create an application by clicking on the new application button provided by them. After creating a new application just create the access tokens so that we no need to provide our authentication details there and also after creating application it will be having one consumer keys to access that application for getting Twitter data. The following is the figure that show clearly how the application data looks after creating the application and here it's self we can see the consumer details and also the access token details. We want to take this keys and token details and want to set in the Flume configuration file such that we can get the required data from the Twitter in the form of tweets.

1.4.2 Getting data using Flume

After creating an application in the Twitter developer site we want to use the consumer key and secret along with the access token and secret values. By which we can access the Twitter and we can get the information that what we want exactly here we will get everything in JSON format and this is stored in the HDFS that we have given the location where to save all the data that comes from the Twitter. The following is the configuration file that we want to use to get the Twitter data from the Twitter.

3.3 Querying using Hive Query Language (HQL)

After running the Flume by setting the above configuration then the Twitter data will automatically will save into HDFSwhere we have the set the path storage to save the Twitter data that was taken by using Flume. From these data first we want to create a table where the filtered data want to set into a formatted structured such that by which we can say clearly that we have converted the unstructured data into structured format. For this we want to use some custom serde concepts.

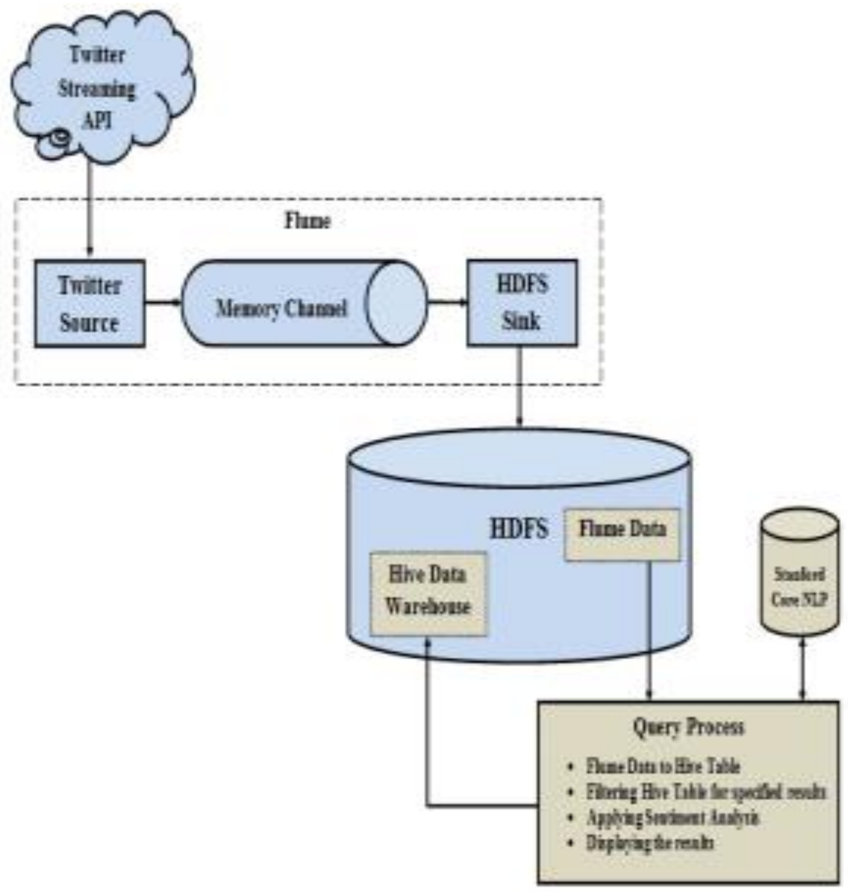


Figure 1. Architecture Diagram For Proposed System

Chapter 2

Literature Survey

Over past ten years, industries and organizations doesn't need to store and perform much operations and analytics on data of the customers. But around from 2005, the need to transform everything into data is much entertained to satisfy the requirements of the people. So Big data came into picture in the real time business analysis of processing data. The term big data refers to the data that is generating around us everyday life. It is generally exceeds the capacity of normal conventional traditional databases. For example by combining a large number of signals from the user's actions and those of their friends, Facebook developed the large network area to the users to share their views, ideas and lot many things.

The value of big data to an organizations falls into two categories: analytical use and enabling new products based on the existing ones. Big data can reveal the issues hidden by data that is too costly to process and perform the analytics such as user's transactions, social and geographical data issues faced by the industry.

The major characteristics and challenges of big data are Volume, Velocity, and Variety. These are called as 3V's of big data which are used to characterize different aspects of big data.

Velocity

The Velocity is defined as the speed at which the data is created, modified, retrieved and processed. This is one of the major challenges of big data because the amount of time for processing and performing different operations are considered a lot when dealing with massive amounts of data.

If we use traditional databases for such types of data, then it is useless and doesn't give full satisfaction to the customers in the organizations and industries.

So processing rate of such data is taken into account considerably when talking about big data. Hence volume is one of the big challenges in dealing with big data.

Volume

The second challenge is the volume i.e amount of data which is processed. There are many business areas where we are uploading and processing the data in very high rate in terms of terabytes and zeta bytes. If we take present statistics, every day we are uploading around 25 terabytes of data into facebook, 12 terabytes of data in twitter and around 10 terabytes of data from various devices like RFID, telecommunications and networking.

Here the storing of these huge amounts of data will require high clusters and large servers with high bandwidth. And here the problem is not only storing the information but also the processing at much higher speed. This became the major issue nowadays in most of the companies.

Variety

In the distributed environment there may be the chances of presenting various types of data. This is known as variety of data. These can be categorized as structured, semi structured and unstructured data. The process of analysis and performing operations are varying from one and another. Social media like Facebook posts or Tweets can give different insights, such as sentiment analysis on your brand, while sensory data will give you information about how a product is used and what the mistakes are. So this is the major issue to process information from different sets of data.

Veracity

Big data Veracity refers to the biases, noise and abnormally in data. It is the data that is being stored, and mined meaningful to the problem being analysed. In other words, Veracity can be treated as the uncertainty of data due to data inconsistency and incompleteness, ambiguities, latency, model approximations in the process of analysing data.

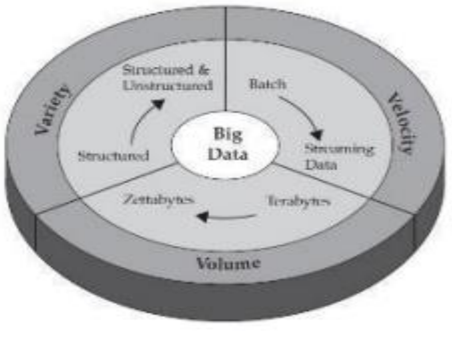


Fig 2. Big Data Challenges

HADOOP- MAPREDUCE

In the distributed environment, the data should be available to users and capable of performing different analysis from databases in a specified amount of time. If we use the normal approaches, it will be difficult to achieve big data challenges. So these types of data required a specialized platform to deal in such cases. Hadoop is the one of the solution to solve big data problems. Hadoop is the open source flexible infrastructure for large scale computation and data processing on a network commodity of hardware systems. Here it deals with both structured and unstructured data and various challenges of big data are solved. The main components of Hadoop are commodity hardware and MapReduce.

Here MapReduce is the heart of Hadoop and each job is performed by this technique only. It is this programming paradigm that allows for massive scalability across hundreds or thousands of servers in a Hadoop cluster. The term MapReduce refers to two tasks in Hadoop i.e. Map and Reduce. In the first step, it takes the data and converts it into another set of data. Here the each word is referred as key and the number of occurrences is treated as value. So MapReduce tuple consists of key value pairs. Then second step consists of reduce operation where it takes the output from a map as input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce job is always performed after the map job.

The major issues involved in big data are the following: The first challenge faced is storing and accessing the information from the large huge amount of data sets from the clusters. We need a standard computing platform to manage large data since the data is growing, and data stores in different data storage locations in a centralized system, which will scale down the huge data into sizable data for computing. The second challenge is retrieving the data from the large social media data sets. In the scenarios where the data is growing daily, it’s somewhat difficult to accessing the data from the large networks if we want to do specific action to be performed. The third challenge concentrates on the algorithm design for handling the problems raised by the huge data volume and the dynamic data characteristics.

Apache Hadoop is good choice for twitter analysis as it works for distributed big data. Apache Hadoop is an open source software framework for distributed storage and large scale distributed processing of data-sets on clusters. Hadoop runs applications using the MapReduce algorithm, where the data is processed in parallel on different CPU nodes. In short, Hadoop framework is capable enough to develop applications capable of running on clusters of computers and they could perform complete statistical analysis for a huge amounts of data. Hadoop MapReduce is a software framework for easily writing applications which process big amounts of data in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. Hadoop framework includes different modules like MapReduce, Flume, Hive, Pig, Sqoop, Oozie, Zookeeper, Hbase for different functionality as shown in below diagram. I will be using FLUME and HIVE for twitter analysis.

Hadoop use HDFS (Hadoop Distributed File System) file system. The Hadoop Distributed File System (HDFS) is based on the Google File System (GFS) and provides a distributed file system that is designed to run on large clusters (thousands of computers) of small computer machines in a reliable, fault-tolerant manner. HDFS uses a master/slave architecture where master consists of a single NameNode that manages the file system metadata and one or more slave DataNodes that store the actual data. Benefit of using Hadoop is distributed storage , Distributed Processing , Security, Reliability, Speed , Efficiency, Availability, Scalability and lots more. This is the reason of using Hadoop for tweet processing.

Chapter 3

System Development

We want to look at which users are responsible for the most retweets, in descending order of most retweeted. However, querying Twitter data in a traditional RDBMS is inconvenient, since the Twitter Streaming API outputs tweets in a JSON format which can be arbitrarily complex. In the Hadoop ecosystem, the Hive project provides a query interface which can be used to query data that resides in HDFS. The query language looks very similar to SQL, but allows us to easily model complex types, so we can easily query the type of data we have. Seems like a good place to start. So how do we get Twitter data into Hive? First, we need to get Twitter data into HDFS, and then we'll be able to tell Hive where the data resides and how to read it.

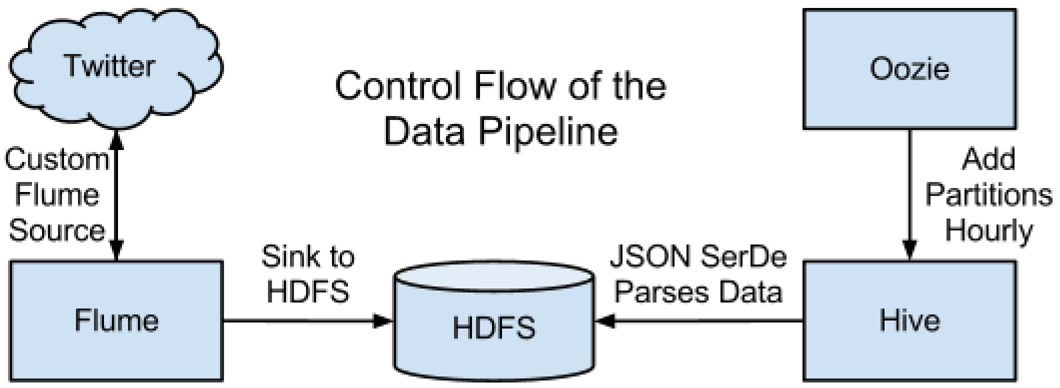
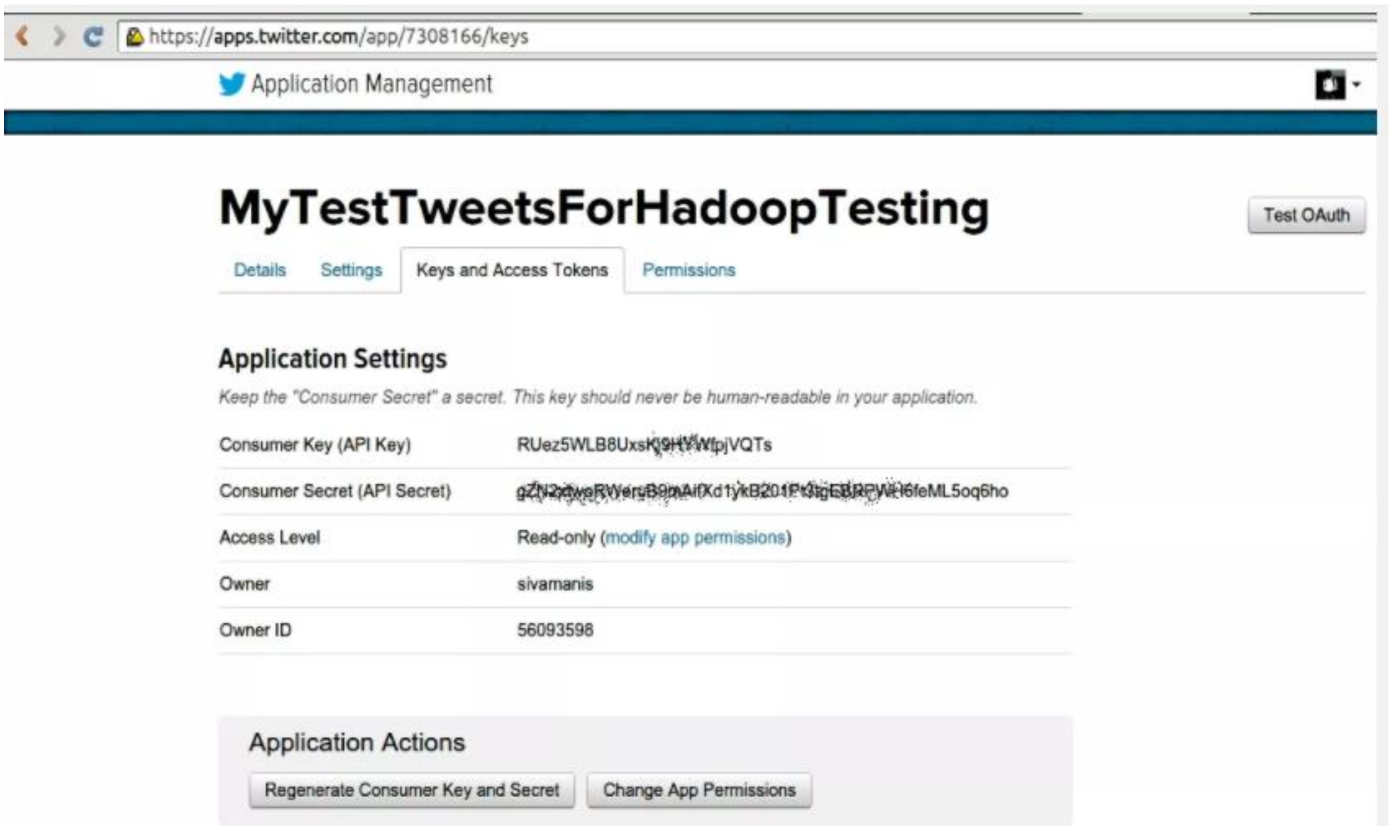


Fig 3.Control Flow Diagram

The diagram above shows a high-level view of how some of the CDH (Cloudera's Distribution Including Apache Hadoop) components can be pieced together to build the data pipeline we need to answer the questions we have. The rest of this post will describe how these components interact and the purposes they each serve.

Creating Twitter Application

First of all if we want to do sentiment analysis on Twitter data we want to get Twitter data first so to get it we want to create an account in Twitter developer and create an application by clicking on the new application button provided by them.[3] After creating a new application just create the access tokens so that we no need to provide our authentication details there and also after creating application it will be having one consumer keys to access that application for getting Twitter data. The following is the figure that show clearly how the application data looks after creating the application and here it's self we can see the consumer details and also the access token details. We want to take this keys and token details and want to set in the Flume configuration file such that we can get the required data from the Twitter in the form of tweets.



MyTestTweetsForHadoopTesting

Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key)	RUez5WLB8UxsK[redacted]pjVQTs
Consumer Secret (API Secret)	[redacted]6feML5oq6ho
Access Level	Read-only (modify app permissions)
Owner	sivamanis
Owner ID	56093598

Application Actions

Fig 5. Creating Twitter Application

Gathering Data with Apache Flume

The Twitter Streaming API will give us a constant stream of tweets coming from the service. One option would be to use a simple utility like curl to access the API and then periodically load the files. However, this would require us to write code to control where the data goes in HDFS, and if we have a secure cluster, we will have to integrate with security mechanisms. It will be much simpler to use components within CDH to automatically move the files from the API to HDFS, without our manual intervention.

Apache Flume is a data ingestion system that is configured by defining endpoints in a data flow called sources and sinks. In Flume, each individual piece of data (tweets, in our case) is called an event; sources produce events, and send the events through a channel, which connects the source to the sink. The sink then writes the events out to a predefined location. Flume supports some standard data sources, such as syslog or netcat. For this use case, we'll need to design a custom source that accesses the Twitter Streaming API, and sends the tweets through a channel to a sink that writes to HDFS files. Additionally, we can use the custom source to filter the tweets on a set of search keywords to help identify relevant tweets, rather than a pure sample of the entire Twitter firehose.

Partition Management with Oozie

Once we have the Twitter data loaded into HDFS, we can stage it for querying by creating an external table in Hive. Using an external table will allow us to query the table without moving the data from the location where it ends up in HDFS. To ensure scalability, as we add more and more data, we'll need to also partition the table. A partitioned table allows us to prune the files that we read when querying, which results in better performance when dealing with large data sets. However, the Twitter API will continue to stream tweets and Flume will perpetually create new files. We can automate the periodic process of adding partitions to our table as the new data comes in.

Apache Oozie is a workflow coordination system that can be used to solve this problem. Oozie is an extremely flexible system for designing job workflows, which can be scheduled to run based on a set of criteria. We can configure the workflow to run an ALTER TABLE command that adds a partition containing the last hour's worth of data into Hive, and we can instruct the workflow to occur every hour. This will ensure that we're always looking at up-to-date data.

Querying Complex Data with Hive

Before we can query the data, we need to ensure that the Hive table can properly interpret the JSON data. By default, Hive expects that input files use a delimited row format, but our Twitter data is in a JSON format, which will not work with the defaults. This is actually one of Hive's biggest strengths. Hive allows us to flexibly define, and redefine, how the data is represented on disk. The schema is only really enforced when we read the data, and we can use the Hive SerDe interface to specify how to interpret what we've loaded.

SerDe stands for Serializer and Deserializer, which are interfaces that tell Hive how it should translate the data into something that Hive can process. In particular, the Deserializer interface is used when we read data off of disk, and converts the data into objects that Hive knows how to manipulate. We can write a custom SerDe that reads the JSON data in and translates the objects for Hive. Once that's put into place, we can start querying.

```
{
  "retweeted_status": {
    "contributors": null,
    "text": "#Crowdsourcing – drivers already generate traffic data for your smartphone to suggest
alternative routes when a road is clogged. #bigdata",
    "geo": null,
    "retweeted": false,
    "in_reply_to_screen_name": null,
    "truncated": false,
    "entities": {
      "urls": [],
      "hashtags": [
        {
          "text": "Crowdsourcing",
          "indices": [
            0,
            14
          ]
        },
        {
          "text": "bigdata",
        "indices": [
          129,
          137
        ]
      }
    ],
    "user_mentions": []
  },
"in_reply_to_status_id_str": null,
"id": 245255511388336128,
"in_reply_to_user_id_str": null,
"source": "SocialOomph",
"favorited": false,
"in_reply_to_status_id": null,
"in_reply_to_user_id": null,
"retweet_count": 0,
"created_at": "Mon Sep 10 20:20:45 +0000 2012",
"id_str": "245255511388336128",
"place": null,
"user": {
  "location": "Oregon, ",
  "default_profile": false,
  "statuses_count": 5289,
  "profile_background_tile": false,
  "lang": "en",
  "profile_link_color": "627E91",
  "id": 347471575,
  "following": null,
  "protected": false,
```

"favourites_count": 17,

"profile_text_color": "D4B020",

"verified": false,

"description": "Dad, Innovator, Sales Professional. Project Management Professional (PMP). Soccer Coach, Little League Coach #Agile #PMOT - views are my own -",

"contributors_enabled": false,

"name": "Scott Ostby",

"profile_sidebar_border_color": "404040",

"profile_background_color": "0F0F0F",

"created_at": "Tue Aug 02 21:10:39 +0000 2011",

"default_profile_image": false,

"followers_count": 19005,

"profile_image_url_https": "https://si0.twimg.com/profile_images/1928022765/scott_normal.jpg",

"geo_enabled": true,

"profile_background_image_url": "http://a0.twimg.com/profile_background_images/327807929/xce5b8c5dfff3dc3bbfbdef5ca2a62b4.jpg",

"profile_background_image_url_https": "https://si0.twimg.com/profile_background_images/327807929/xce5b8c5dfff3dc3bbfbdef5ca2a62b4.jpg",

"follow_request_sent": null,

"url": "http://facebook.com/ostby",

"utc_offset": -28800,

"time_zone": "Pacific Time (US &amp; Canada)",

"notifications": null,

"friends_count": 13172,

"profile_use_background_image": true,

"profile_sidebar_fill_color": "1C1C1C",

"screen_name": "ScottOstby",

"id_str": "347471575",

"profile_image_url": "http://a0.twimg.com/profile_images/1928022765/scott_normal.jpg",

"show_all_inline_media": true,

```
        "is_translator": false,
        "listed_count": 45
      },
      "coordinates": null
    },
    "contributors": null,
    "text": "RT @ScottOstby: #Crowdsourcing – drivers already generate traffic data for your smartphone to suggest alternative routes when a road is  ...",
    "geo": null,
    "retweeted": false,
    "in_reply_to_screen_name": null,
    "truncated": false,
    "entities": {
      "urls": [],
      "hashtags": [
        {
          "text": "Crowdsourcing",
          "indices": [
            16,
            30
          ]
        }
      ],
      "user_mentions": [
        {
          "id": 347471575,
          "name": "Scott Ostby",
          "indices"}
      ]
}
```

We've now managed to put together an end-to-end system, which gathers data from the Twitter Streaming API, sends the tweets to files on HDFS through Flume, and uses Oozie to periodically load the files into Hive, where we can query the raw JSON data, through the use of a Hive SerDe.

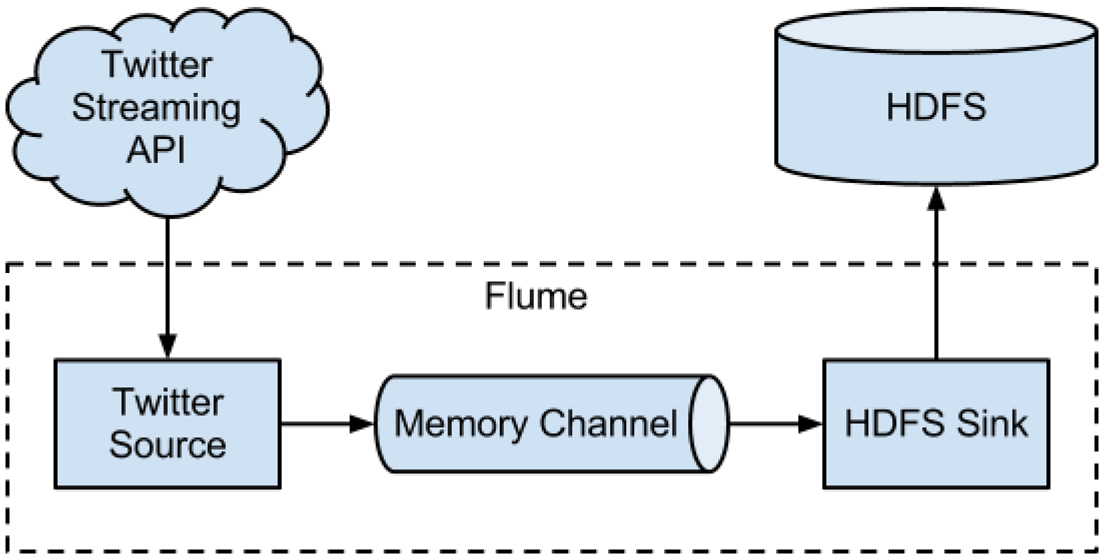


Fig 5. Flume Components

Apache Flume is one way to bring data into HDFS using CDH. The Apache Flume website describes Flume as "a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data." At the most basic level, Flume enables applications to collect data from its origin and send it to a resting location, such as HDFS. At a slightly more detailed level, Flume achieves this goal by defining dataflows consisting of three primary structures: *sources*, *channels* and *sinks*. The pieces of data that flow through Flume are called *events*, and the processes that run the dataflow are called *agents*.

In the Twitter example, we used Flume to collect data from the Twitter Streaming API, and forward it to HDFS.

Sources

A *source* is just what it sounds like: the part of Flume that connects to a source of data, and starts the data along its journey through a Flume dataflow. A source processes events and moves them along by sending them into a channel. Sources operate by gathering discrete pieces of data, translating the data into individual events, and then using the channel to process events one at a time, or as a batch.

Sources come in two flavors: event-driven or pollable. The difference between event-driven and pollable sources is how events are generated and processed. Event-driven sources typically receive events through mechanisms like callbacks or RPC calls. Pollable sources, in contrast, operate by polling for events every so often in a loop. Another good way to frame this differentiation is as a push-versus-pull model, where event-driven sources have events pushed to them, and pollable sources pull events from a generator.

Examining the TwitterSource

In our Twitter analysis example, we built a custom source called TwitterSource. To understand how sources operate more thoroughly, let's look at how the TwitterSource was built.The `start()` method contains the bulk of the source's logic. In the TwitterSource, the twitter4j library is used to get access to the Twitter Streaming API.

The StatusListener implements a set of callbacks that will be called when receiving a new tweet, represented by a Status object. There are other callbacks available but for the purposes of this source, we're only concerned with new tweets. As can be seen in the TwitterSource, the StatusListener is created and registered in the `start()` method.

Channels

Channels act as a pathway between the sources and sinks. Events are added to channels by sources, and later removed from the channels by sinks. Flume dataflows can actually support multiple channels, which enables more complicated dataflows, such as fanning out for replication purposes.

Memory channels use an in-memory queue to store events until they're ready to be written to a sink. Memory channels are useful for dataflows that have a high throughput; however, since events are stored in memory in the channel, they may be lost if the agent experiences a failure. If the risk of data loss is not tolerable, this situation can be remedied using a different type of channel – i.e., with one that provides stronger guarantees of data durability like a FileChannel.

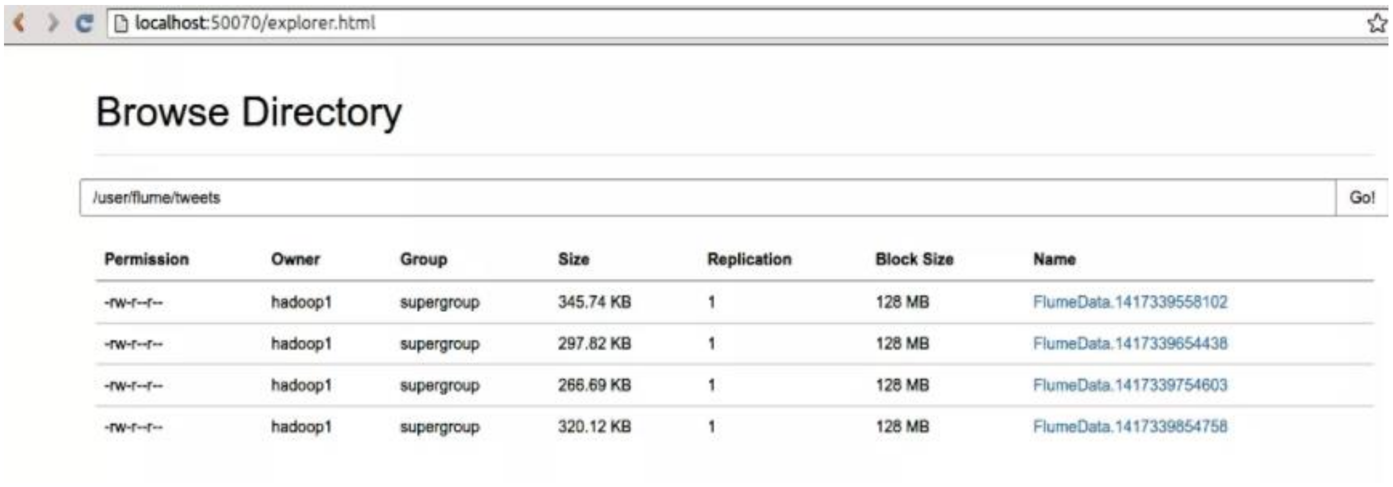
Sinks

The final piece of the Flume dataflow is the *sink*. Sinks take events and send them to a resting location or forward them on to another agent. In the Twitter example, we utilized an HDFS sink, which writes events to a configured location in HDFS.

Starting the Agent

Now that we understand the configuration of our source, channel and sink, we need to start up the agent to get the dataflow running. Before we actually start the agent, we need to set the agent to have the appropriate name as defined in the configuration.

The file /etc/default/flume-ng-agent contains one environment variable defined called FLUME_AGENT_NAME. In a production system, for simplicity, the FLUME_AGENT_NAME will typically be set to the hostname of the machine on which the agent is running. However, in this case, we set it to TwitterAgent, and we're ready to start up the process.



Permission	Owner	Group	Size	Replication	Block Size	Name
-rw-r--r--	hadoop1	supergroup	345.74 KB	1	128 MB	FlumeData.1417339558102
-rw-r--r--	hadoop1	supergroup	297.82 KB	1	128 MB	FlumeData.1417339654438
-rw-r--r--	hadoop1	supergroup	266.69 KB	1	128 MB	FlumeData.1417339754603
-rw-r--r--	hadoop1	supergroup	320.12 KB	1	128 MB	FlumeData.1417339854758

Fig 6. Starting Flume Agent

Characterizing Data

One of the first questions to ask when deciding on the right tool for the job is: "what does my data look like?" If your data has a very strict schema, and it doesn't deviate from that schema, maybe you should just be using a relational database. MySQL is just as free as Hive, and very effective for dealing with well-structured data. However, as you start to try to analyze data with less structure or with extremely high volume, systems like MySQL become less useful, and it may become necessary to move out of the relational world.

Unstructured, semi-structured, and poly-structured are all terms for data that doesn't fit well into the relational model. This is data like JSON, XML, RDF, or other sorts of data with a schema that may vary from record to record. What do we do with this data? Here's where Hive shines. Hive is extremely effective for dealing with data that doesn't quite fit into the relational bucket, because it can process complex, nested types natively. Hive avoids the need for complicated transformations that might be otherwise necessary to handle this sort of data in a traditional relational system. Hive can also gracefully handle records that don't strictly conform to a table's schema. For example, if some columns are missing from a particular record, Hive can deal with the record by treating missing columns as NULLs.

It's fairly easy to see that there is a good bit of complexity to this data structure. Since JSON can contain nested data structures, it becomes very hard to force JSON data into a standard relational schema. Processing JSON data in a relational database would likely require significant transformation, making the job much more cumbersome.

Looking at this particular bit of JSON, there are some very interesting fields: At the very top, there is a retweeted_status object, whose existence indicates that this tweet was retweeted by another user. If the tweet was not a retweet, you would not have a retweeted_status object at all. Tweets also contain an entities element, which is a nested structure. It contains three arrays, the elements of which are all nested structures in their own right, as can be seen in the hashtags array, which has two entries.

Complex Data Structures

Hive has native support for a set of data structures that normally would either not exist in a relational database, or would require definition of custom types. There are all the usual players: integers, strings, floats, and the like, but the interesting ones are the more exotic maps, arrays, and structs. Maps and arrays work in a fairly intuitive way, similar to how they work in many scripting languages:

If the user_mentions array is empty, Hive will just return NULL for that record.

The PARTITIONED BY clause utilizes a feature of Hive called partitioning, which allows tables to be split up in different directories. By building queries that involve the partitioning column, Hive can determine that certain directories cannot possibly contain results for a query. Partitioning allows Hive to skip the processing of entire directories at query time, which can improve query performance dramatically.

The LOCATION clause is a requirement when using EXTERNAL tables. By default, data for tables is stored in a directory located at /user/hive/warehouse/

However, EXTERNAL tables can specify an alternate location where the table data resides, which works nicely if Flume is being used to place data in a predetermined location. EXTERNAL tables also differ from regular Hive tables, in that the table data will not be removed if the EXTERNAL table is dropped.

The ROW FORMAT clause is the most important one for this table. In simple datasets, the format will likely be DELIMITED, and we can specify the characters that terminate fields and records, if the defaults are not appropriate. However, for the tweets table, we've specified a SERDE.

Serializers and Deserializers

In Hive, SerDe is an abbreviation for Serializer and Deserializer, and is an interface used by Hive to determine how to process a record. Serializers and Deserializers operate in opposite ways.
The Deserializer interface takes a string or binary representation of a record, and translates it into a Java object that Hive can manipulate. The Serializer, on the other hand, will take a Java object that Hive has been working with, and turn it into something that Hive can write to HDFS. Commonly, Deserializers are used at query time to execute SELECT statements, and Serializers are used when writing data, such as through an INSERT-SELECT statement. In the Twitter analysis example, we wrote a JSONSerDe, which can be used to transform a JSON record into something that Hive can process.

Putting It All Together

By utilizing the SerDe interface, we can instruct Hive to interpret data according to its inherent structure (or lack thereof). Since a SerDe is just a property of a Hive table, rather than the data, itself, we can also swap out SerDes as our data evolves. That flexibility allows us to choose the right tools for the job at hand, and to interpret data in different ways. It makes Hive a spectacular choice for getting quick access to data of all types

CHAPTER 4

RESULT AND PERFORMANCE ANALYSIS

The following figure shows the system architecture of the proposed system:

In the first step I collected the Twitter data (Tweets) using API streaming of tokens and apache flume. Then I uploaded the tweets into Hadoop Files Systems by hdfs commands. It includes moving of complete tweets of different users to file systems. And Finally I applied MapReduce Technique to find out the Twitter ID's of the most tweeted people. After performing the MapReduce job, it retrieves the resulted data with the ids.

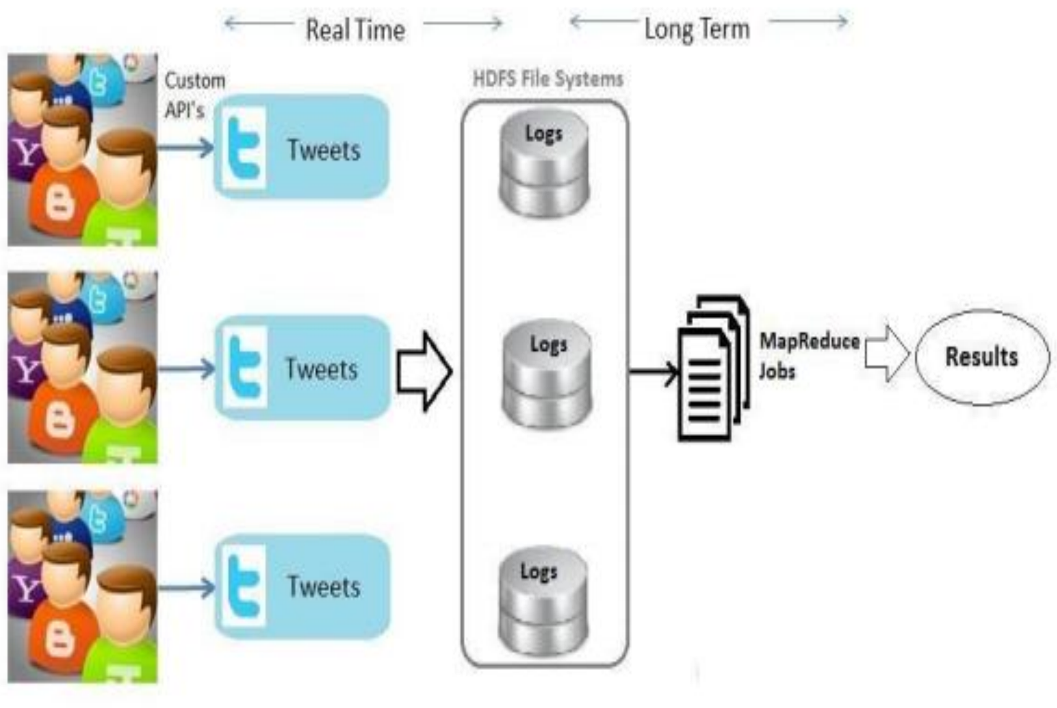


Fig 7. System Architecture

The collected data was about half a GB of JSON data, and here is an example of what a tweet looks like. The data has some structure, but certain fields may or may not exist. The retweeted_status field, for example, will only be present if the tweet was a retweet. Additionally, some of the fields may be arbitrarily complex. The hashtags field is an array of all the hashtags present in the tweets, but most RDBMS's do not support arrays as a column type. This semi-structured quality of the data makes the data very difficult to query in a traditional RDBMS. Hive can handle this data much more gracefully.

The query below will find usernames, and the number of retweets they have generated across all the tweets that we have data for:

```
SELECT
  t.retweeted_screen_name,
  sum(retweets) AS total_retweets,
  count(*) AS tweet_count
FROM (SELECT
        retweeted_status.user.screen_name as retweeted_screen_name,
            retweeted_status.text,
            max(retweet_count) as retweets
      FROM tweets
      GROUP BY retweeted_status.user.screen_name,
               retweeted_status.text) t
GROUP BY t.retweeted_screen_name
ORDER BY total_retweets DESC
LIMIT 10;
```

For the few days of data, I found that these were the most retweeted users for the industry:

```
retweeted_screen_name    total_retweets    tweet_count
  mauricefreedman        493               1
  HarvardBiz             362               6
  TechCrunch             314               7
  googleanalytics        244               10
  BigDataBorat           201               6
  stephen_wolfram        182               1
  CloudExpo              153               28
  TheNextWeb             150               1
  GonzalezCarmen         121               10
  IBMbigdata             100               37
```

From these results, we can see whose tweets are getting heard by the widest audience, and also determine whether these people are communicating on a regular basis or not. We can use this information to more carefully target our messaging in order to get them talking about our products, which, in turn, will get other people talking about our products.

APPLICATIONS AND SCOPE

The major applications of the big data are

- Sentiment Analysis: Sentiment data is unstructured data that represents opinions, emotions, and attitudes contained in sources such as social media posts, blogs, online product reviews, and customer support interactions. Different companies and Organizations use social media analysis to understand how the public feels about something at a particular moment in time, and also to track how those opinions change over time.

- Text Analytics: It is the process of deriving the high quality information from the raw data such as unstructured data and predicting the analysis.

- Volume Trending: Here volume is estimated in terms of amount of data to process a job. Volume trending is a big issue nowadays. Day by day it has been increasing in a much higher rate in the organizations and social media sites etc.

- Predictive Analytics: Predictive analysis gives the predictive scores to the organizations to help in making the smart decisions and improve business solutions. It optimizes marketing campaigns and website behavior to increase customer responses in business, conversions and meetings, and to decrease the burden on the people. Each customer's predictive score informs actions to be taken with that customer.

- Massively Scalable Architectures:

- Social Media Data: With Hadoop, we can mine Twitter, Facebook and other social media conversations for sentiment data about people and used it to make targeted, real time decisions that increase market share.

- Web Clickstream data: Hadoop makes easy to track customers and their activities in different issues like products purchasing and viewing etc. It makes analyzers to know the behavior and interest of the customers and can able to visualize similar type of products to the customers.

A. Scope

The Proposed system can finds the most popular information about the people, organizations and can be used in the field of analytics.

Applications

- Finding out the twitter id’s of those persons whose tweets are retweeted number of times.
- Finds the most number of follows in the social networking sites.
- This system can be useful to track the business analysis of the organizations.
- Allows researchers to retrieve and analyze the data easily from large datasets.

CHAPTER 5
CONCLUSIONS

6.1 Conclusions

Traditional Enterprise Data Warehouses do not have the ability to keep up with rapidly increasing social media data. With this system, one can build a dashboard to monitor the sentiment of Twitter traffic around any given topic in near real-time (that is, with a delay of 1-2 minutes), allowing you or your users to take advantage of near real-time Twitter sentiment for business insights or any other purpose. There are several ways to define and analyse the social media data such as facebook, Twitter etc. Here anyone can perform different operations queries in these type of data. But the problem arises when dealing with bigdata of several types of unstructured data. Here it is solved by using Hadoop and its packages. And we have done some analysis on the tweets and the most number of tweet ids. So it is concluded that processing time and retrieving capabilities are made very easy when compared to other processing and analysing techniques for large amounts of data.

6.2 Future Scope

Nowadays big data has become the buzzword in IT industry organizations. The need of analysing and processing of information has grown a lot. This paper implemented the analysing of big data (tweets) only for text. Further analysis can be done to images and all types of multimedia files based on index support. The result of Text mining and data analysis would help in suggesting related pages based on different types of data. So that industries make the data easily available to people who is using and trying accessing such type of data.

References

[1] (Online Resource) Hive (Available on:http://hive.apache.org/).

[2] (Online Resource)http://jsonlint.com/

[3] (Online Resource)http://nlp.stanford.edu/software/corenlp.shtml.

[4]T. White, "The Hadoop Distributed Filesystem," Hadoop: The Definitive Guide, pp. 41-73, GravensteinHighwaNorth, Sebastopol: O'Reilly Media, Inc., 2010.

[5] (Online Resource) http://flume.apache.org/

[6]S. W. Ambler. Relational databases 101: Looking at the whole picture.www.AgileData.org, 2009.

[7] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. H. Byers, "Big Data: The Next Frontier For Innovation, Competition, And Productivity", May 2011.