ISSN (Online) : 2348 - 2001
International Refereed Journal of Reviews and Research
Volume 6 Issue 1 January 2018
International Manuscript ID : 23482001V6I1012018-07
(Approved and Registered with Govt. of India)

Machine Learning Integrated Big Data and High Performance Computing using Apache Storm

Ankita
Research Scholar
Department of Computer Science
OPJS University, Rajasthan, India

Dr. Om Parkash
Associate Professor
Department of Computer Science
OPJS University, Rajasthan, India

Abstract

Big Data Analytics is one of the key areas of research with assorted approaches in data science and predictive analysis. A number of scenarios exist where enormous data is logged every day and needs deep evaluation for research and development. In Medical Science, there are enormous examples where processing, analysis and predictions from huge amount of data is required regularly. As per the reports from *First Post*, the data of more than 50 Peta Bytes are generated from each hospital of 500 beds in USA. In another research, it is found that one gram of DNA is equivalent to 215 petabytes in digital form. In another scenario of digital communication, the number of smart wearable gadgets increased from 26 millions in year 2014 to more than 100 millions in year 2016.

Keywords : Apache Storm, Big Data, Machine Learning

ISSN (Online) : 2348 - 2001
International Refereed Journal of Reviews and Research
Volume 6 Issue 1 January 2018
International Manuscript ID : 23482001V6I1012018-07
(Approved and Registered with Govt. of India)

Introduction

A prominent neuroscientist *Ann-Shyn Chian* from Taiwan presented in a research that more than 1 GB per cell of brain will be required even for a very small creature on this earth. For imaging of more than 80 billion neurons in human brain, it will take around 17 million years. Now, the volume, velocity, variety of medical data can be imagined with these data figures.

Creature	**No. of Neurons in Brain / Nervous System**
Fly	1,35,000
Cockroach	1,000,000
Ant	2,50,000
Honey Bee	9,60,000
Cat	760,000,000
Monkey	3,246,000,000
Macaque	6,376,000,000
Human	86,000,000,000

Here, the key question comes on the evaluation of huge amount of data with enormously growing speed. To preprocess, analyze, evaluate and predict on such big data based applications, there is need to use high performance computing frameworks and libraries so that processing power of computers can be utilized with maximum throughput and performance [1].

Free and Open Source Big Data Processing Tools

- Apache Storm
- Apache HADOOP
- Lumify

- HPCC Systems
- Apache Samoa
- ElasticSearch
- RapidMiner
- R-Programming
- Scribe
- NoSQL Databases

Map Reduce Technology

Apache Storm is one of the powerful and performance aware realtime Distributed Computation System under Free and Open Source (FOSS) paradigm. The unbounded and free flowing data from multiple channels can be effectively logged and evaluated using Apache Storm [2] with real time processing as compared to batch processing in Hadoop. In addition, Storm is effectually adopted by number for corporate applications with the integration of any programming language without any issues of compatibility. The state of clusters and distributed environment is managed via Apache Zookeeper in the implementation of Apache Storm. The research based algorithms and predictive analytics can be executed in parallel using Apache Storm.

MapReduce refers to a fault tolerant distributed high performance computational framework which is used to process and evaluate the huge amount of data. MapReduce like functions can be effectively implemented in Apache Storm using Bolts as the key logical operations are performed at the level of bolts. In many cases, the performance of Bolts in Apache Storm can be outperform MapReduce [3].

ISSN (Online) : 2348 - 2001
International Refereed Journal of Reviews and Research
Volume 6 Issue 1 January 2018
International Manuscript ID : 23482001V6I1012018-07
(Approved and Registered with Govt. of India)

Key Advantages and Features of Apache Storm

- Free and Open Source
- User Friendly
- Fit for any type of implementation from small to large scale implementations
- Fault Tolerant
- Reliabilty
- Fault Tolerant
- Real Time Processing
- Extremely fast
- Operational Intelligence
- Dynamic Load Balancing and Optimization
- Scalability

Installing Apache Storm and Zookeeper on MS Windows Environment

First of all, Apache Zookeeper is downloaded and installed from https://zookeeper.apache.org/.

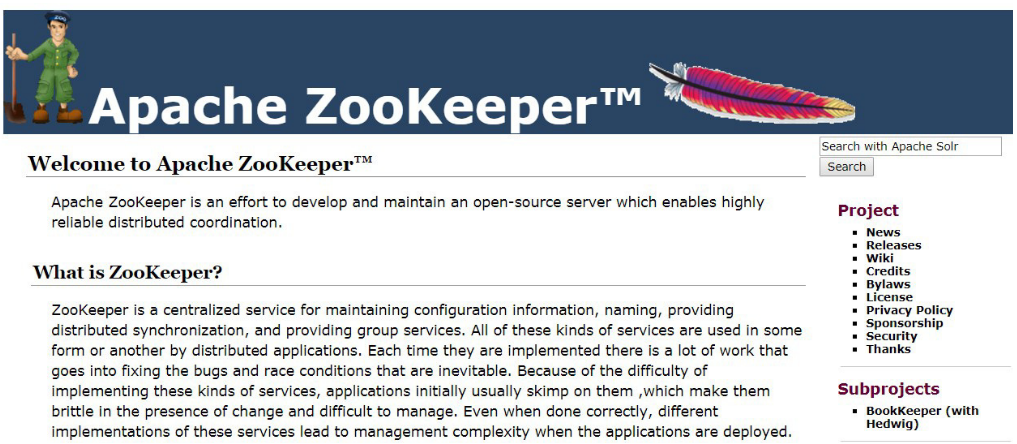

Figure 1: Official Portal of Apache Zookeeper

ISSN (Online) : 2348 - 2001

International Refereed Journal of Reviews and Research

Volume 6 Issue 1 January 2018

International Manuscript ID : 23482001V6I1012018-07

(Approved and Registered with Govt. of India)

Configure and run Zookeeper with the following commands:

MSWindowsDrive:\> cd zookeeper-Version

MSWindowsDrive:\ zookeeper-Version> copy conf\zoo_sample.cfg conf\zoo.cfg

MSWindowsDrive:\ zookeeper-Version> .\bin\zkServer.cmd

Following records are updated in zoo.cfg:

tickTime=2000

initLimit=10

syncLimit=5

dataDir= *MSWindowsDrive*:/zookeeper-3.4.8/data

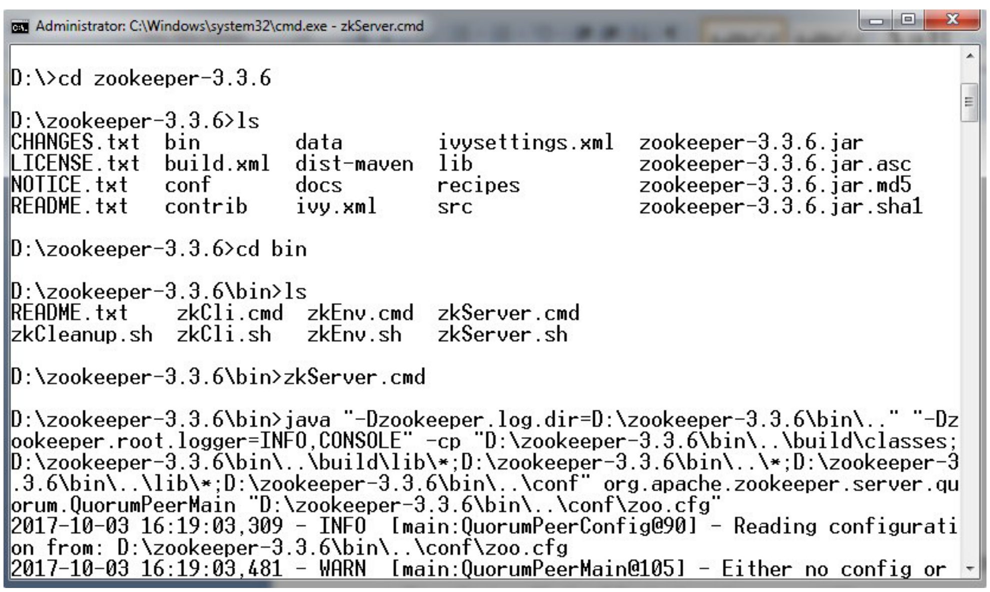

Figure 2: Execution of Commands for Initialization of Apache Zookeeper

Download and Install Apache Storm from http://storm.apache.org/ and set STORM_HOME to *MSWindowsDrive*:\apache-storm-Version following in environment variables.

ISSN (Online) : 2348 - 2001

International Refereed Journal of Reviews and Research

Volume 6 Issue 1 January 2018

International Manuscript ID : 23482001V6I1012018-07

(Approved and Registered with Govt. of India)

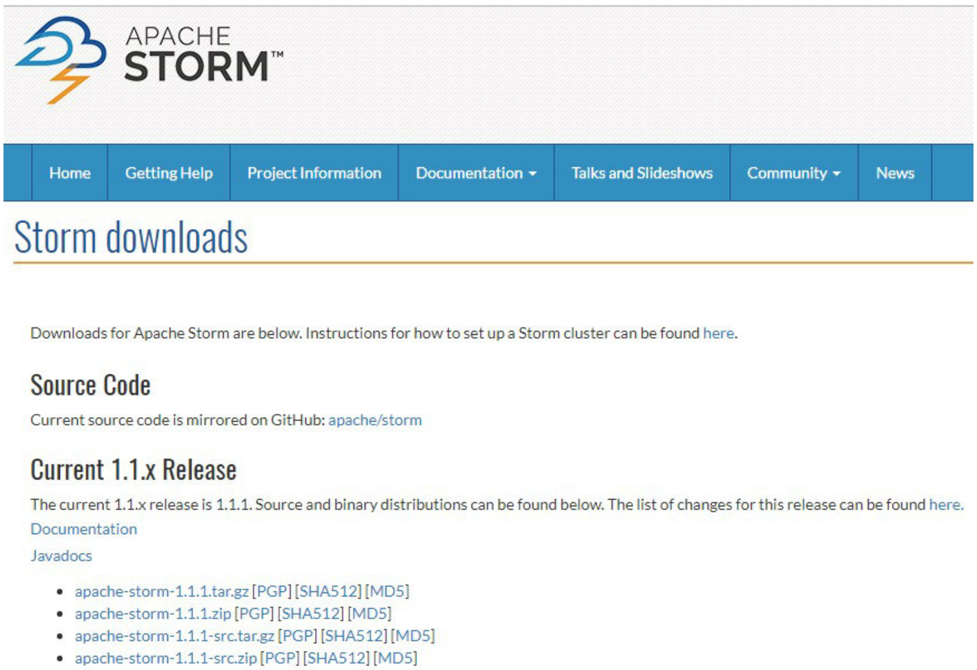

Figure 3: Download Page of Apache Storm for Multiple Platforms

Perform the modifications in storm.yaml as follows:

storm.zookeeper.servers:

– "127.0.0.1"

nimbus.host: "127.0.0.1"

storm.local.dir: "D:/storm/datadir/storm"

supervisor.slots.ports:

– 6700

– 6701

– 6702

– 6703

ISSN (Online) : 2348 - 2001

International Refereed Journal of Reviews and Research

Volume 6 Issue 1 January 2018

International Manuscript ID : 23482001V6I1012018-07

(Approved and Registered with Govt. of India)

In MS Windows Command Prompt, Go To the path of STORM_HOME and execute the following commands:

1. storm nimbus
2. storm supervisor
3. storm ui

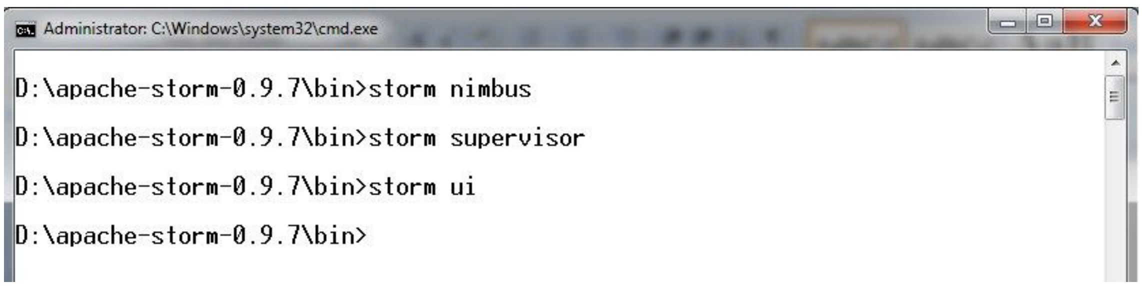

Figure 4: Execution of Commands for Initialization of Apache Storm

In any Web Browser, Execute the URL http://localhost:8080 to confirm the working of Apache Storm

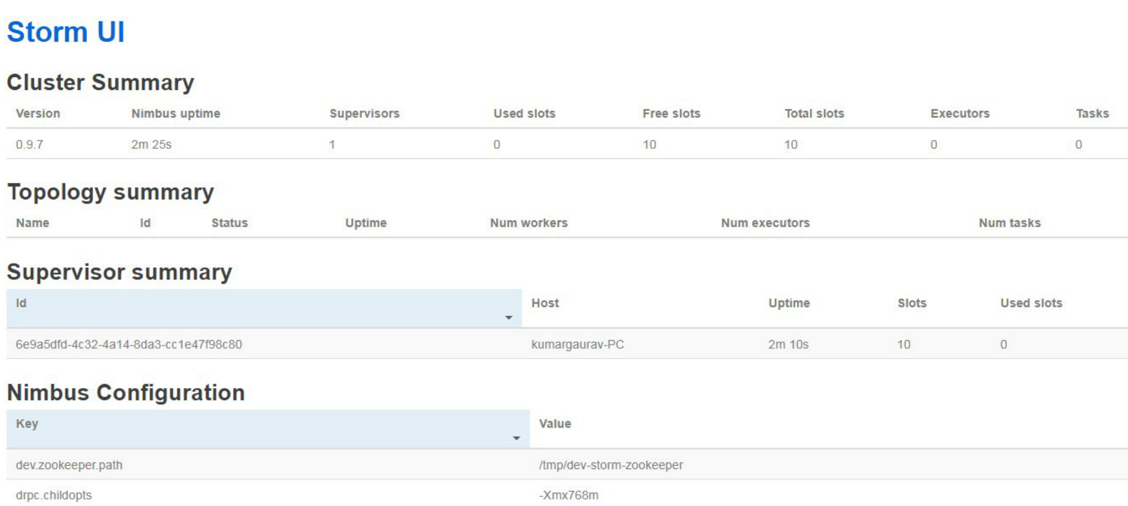

Figure 5: Apache Storm UI with the Base Configurations, Nodes and Cluster Information

ISSN (Online) : 2348 - 2001
International Refereed Journal of Reviews and Research
Volume 6 Issue 1 January 2018
International Manuscript ID : 23482001V6I1012018-07
(Approved and Registered with Govt. of India)

Apache Storm is having association with number of key components and modules which work together to perform the high performance computing [4]. These components include Nimbus Node, Supervisor Node, Worker Process, Executor, Task and many others. Following is the brief description of key components which are used in the implementation of Apache Storm.

Components	**Description**
Nimbus	Master Node in the Cluster of Apache Storm. The other nodes are referred as Worker Nodes. Master Node is the key node used for the distribution of data to the worker nodes and overall monitoring
Supervisor	The nodes which accept the instructions sent by Nimbus node. Supervisor Node is having multiple processes mapped with the workers
Worker process	Execution of tasks associated with particular topology. It creates the executors to execute the tasks
Executor	Thread created to execute the process
Task	The specific task to be performed
Zookeeper Framework	A Service that is used by a group of nodes or simply clusters. Zookeeper assist the supervisor to communicate with Nimbus and maintain the states of communication

ISSN (Online) : 2348 - 2001
International Refereed Journal of Reviews and Research
Volume 6 Issue 1 January 2018
International Manuscript ID : 23482001V6I1012018-07
(Approved and Registered with Govt. of India)

Tuple	Key data structure in Apache Storm supporting all formats and data types
Stream	The sequence of tuples in unordered perspectives
Spout	Used to accept the input data from different channels like Twitter Streaming, Bioinformatics, Medical, Satellite Data and many others
Bolt	These are Logical Processing Units in Apache Storm

Extraction and Analytics of Twitter Social Media using Apache Storm

To extract the live data from Twitter Social Media, the APIs of Twitter4j are used which provides the programming interface to connect with Twitter Servers. In Eclipse IDE, the code of Java can be programmed for predictive analysis and evaluation of the tweets fetched from real time streaming channels. As social media mining is one of the key segment of research for extraction and prediction of popularity, the following code snippets are used to extract the real time streaming and evaluation of user sentiments.

MyTwitterSpout.java

```
public class MyTwitterSpout extends BaseRichSpout {
  SpoutOutputCollector _collector;
  LinkedBlockingQueue<Status> queue = null;
  TwitterStream _twitterStream;
  String myconsumerKey;
  String myconsumerSecret;
  String myaccessToken;
```

ISSN (Online) : 2348 - 2001

International Refereed Journal of Reviews and Research

Volume 6 Issue 1 January 2018

International Manuscript ID : 23482001V6I1012018-07

(Approved and Registered with Govt. of India)

```
String myaccessTokenSecret;
String[] mykeyWords;
public MyTwitterSpout(String myconsumerKey, String myconsumerSecret,
  String myaccessToken, String myaccessTokenSecret, String[] mykeyWords) {
    this.myconsumerKey = myconsumerKey;
    this.myconsumerSecret = myconsumerSecret;
    this.myaccessToken = myaccessToken;
    this.myaccessTokenSecret = myaccessTokenSecret;
    this.mykeyWords = mykeyWords;
}
public MyTwitterSpout() {   }
@Override
public void open(Map conf, TopologyContext context,
  SpoutOutputCollector collector) {
    queue = new LinkedBlockingQueue<Status>(1000);
    _collector = collector;
    MyStatusListener listener = new MyStatusListener() {
      @Override
      public void onStatus(Status MyStatus) {
        queue.offer(status);
      }
      @Override
      public void onDeletionNotice(StatusDeletionNotice sdn) {}
      @Override
      public void onTrackLimitationNotice(int i) {}
      @Override
      public void onScrubGeo(long l, long l1) {}
```

```
      @Override
      public void onException(Exception ex) {}
      @Override
      public void onStallWarning(StallWarning arg0) {
      }         };
    ConfigurationBuilder MyCB = new ConfigurationBuilder();
    MyCB.setDebugEnabled(true)
      .setOAuthMyconsumerKey(myconsumerKey)
      .setOAuthMyconsumerSecret(myconsumerSecret)
      .setOAuthMyaccessToken(myaccessToken)
      .setOAuthMyaccessTokenSecret(myaccessTokenSecret);
    _twitterStream = new TwitterStreamFactory(cb.build()).getInstance();
    _twitterStream.addListener(listener);
    if (mykeyWords.length == 0) {
      _twitterStream.sample();
    }else {
      FilterQuery MyQuery = new FilterQuery().track(mykeyWords);
      _twitterStream.filter(query);
    }   }
@Override
public void nextTuple() {
  MyStatus ret = queue.poll();

  if (ret == null) {
    Utils.sleep(50);
  } else {
    _collector.emit(new Values(ret));
```

```
    }   }
  @Override
  public void close() {
    _twitterStream.shutdown();
  }
  @Override
  public Map<String, Object> getComponentConfiguration() {
    Config ret = new Config();
    ret.setMaxTaskParallelism(1);
    return ret;
  }
  @Override
  public void ack(Object id) {}
  @Override
  public void fail(Object id) {}
  @Override
  public void declareOutputFields(OutputFieldsDeclarer declarer) {
    declarer.declare(new Fields("tweet"));
  } }
```

MyHashtagBolt.java

```
public class MyHashtagBolt implements IRichBolt {
  private OutputCollector collector;
  @Override
  public void prepare(Map conf, TopologyContext context, OutputCollector collector) {
    this.collector = collector;
  }
```

ISSN (Online) : 2348 - 2001
International Refereed Journal of Reviews and Research
Volume 6 Issue 1 January 2018
International Manuscript ID : 23482001V6I1012018-07
(Approved and Registered with Govt. of India)

```
  @Override
  public void execute(Tuple tuple) {
    Status tweet = (Status) tuple.getValueByField("tweet");
    for(HashtagEntity hashtage : tweet.getHashtagEntities()) {
      System.out.println("Hashtag: " + hashtage.getText());
      this.collector.emit(new Values(hashtage.getText()));
    }   }
  @Override
  public void cleanup() {}
  @Override
  public void declareOutputFields(OutputFieldsDeclarer declarer) {
    declarer.declare(new Fields("hashtag"));
  }
  @Override
  public Map<String, Object> getComponentConfiguration() {
    return null;
  } }
```

MyBolt.java

```
public class MyBolt implements IRichBolt {
  Map<String, Integer> counterMap;
  private OutputCollector collector;
  @Override
  public void prepare(Map conf, TopologyContext context, OutputCollector collector) {
    this.counterMap = new HashMap<String, Integer>();
    this.collector = collector;
  }
```

```
@Override
public void execute(Tuple tuple) {
  String key = tuple.getString(0);
  if(!counterMap.containsKey(key)){
    counterMap.put(key, 1);
  }else{
    Integer c = counterMap.get(key) + 1;
    counterMap.put(key, c);
  }
  collector.ack(tuple);
}
@Override
public void cleanup() {
  for(Map.Entry<String, Integer> entry:counterMap.entrySet()){
    System.out.println("Result: " + entry.getKey()+" : " + entry.getValue());
  }   }
@Override
public void declareOutputFields(OutputFieldsDeclarer declarer) {
  declarer.declare(new Fields("hashtag"));
}
@Override
public Map<String, Object> getComponentConfiguration() {
  return null;
} }
```

MyApacheStorm.java

```
public class MyApacheStorm {
```

```
public static void main(String[] args) throws Exception{
  String myconsumerKey = args[0];
  String myconsumerSecret = args[1];
  String myaccessToken = args[2];
  String myaccessTokenSecret = args[3];
  String[] MyArguments = args.clone();
  String[] mykeyWords = Arrays.copyOfRange(arguments, 4, MyArguments.length);
  MyConfig MyConfig = new MyConfig();
  MyConfig.setDebug(true);
  TopologyBuilder builder = new TopologyBuilder();
  builder.setSpout("twitter-spout", new MyTwitterSpout(myconsumerKey,
    myconsumerSecret, myaccessToken, myaccessTokenSecret, mykeyWords));
  builder.setBolt("twitter-hashtag-reader-bolt", new MyHashtagBolt())
    .shuffleGrouping("twitter-spout");
  builder.setBolt("twitter-hashtag-counter-bolt", new MyBolt())
    .fieldsGrouping("twitter-hashtag-reader-bolt", new Fields("hashtag"));
  LocalCluster MyCluster = new LocalCluster();
  MyCluster.submitTopology("MyApacheStorm", MyConfig,
    builder.createTopology());
  Thread.sleep(10000);
  MyCluster.shutdown();
}}
```

Conclusion

The extraction of datasets from live satellite channels and cloud delivery points can be implemented with the integrated approach of Apache Storm to have effectual predictions on specific paradigms. As an example, the live streaming data of longitude and latitude from smart gadget with the deep learning based approach can be implemented to predict the upcoming position of specific person. In bioinformatics and medical sciences, the probability of a specific disease in a person can be predicted with the neural network based learning of historical medical records and health parameters using Apache Storm. Besides, these there are enormous domains and areas where big data analytics including AADHAAR, Banking Datasets, Rainfall Predictions and many others can be done with the effectual results having contribution to the social cause.

References

[1] Kumar, T. V., JNU, D., Rana, P. S., Sinha, M. S., Tagra, H., Misra, M. B., ... & DU, D. (2017). Big Data Analytics.

[2] Iqbal, M. H., & Soomro, T. R. (2015). Big data analysis: Apache storm perspective. International journal of computer trends and technology, 19(1), 9-14.

[3] Das Sarma, A., He, Y., & Chaudhuri, S. (2014). Clusterjoin: A similarity joins framework using map-reduce. Proceedings of the VLDB Endowment, 7(12), 1059-1070.

[4] Mylavarapu, G., Thomas, J., & TK, A. K. (2015, August). Real-time Hybrid Intrusion Detection System using Apache Storm. In High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), 2015 IEEE 12th International Conferen on Embedded Software and Systems (ICESS), 2015 IEEE 17th International Conference on (pp. 1436-1441). IEEE.