Twitter Trends Using Storm

Bharath Rajakumar

Dept. Computer Science

University of Texas at Dallas

Dallas, Texas

bharath.rajakumar@gmail.com

Harshita Vissa

Dept. Computer Science

University of Texas at Dallas

Dallas, Texas

harshitavissa@gmail.com

Sripadmanaban Subramanian

Dept. Computer Science

University of Texas at Dallas

Dallas, Texas

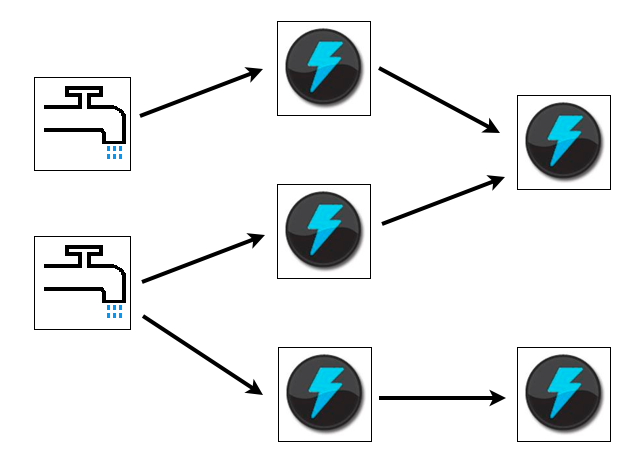
sripadmanabans@gmail.com

***Abstract***

We are living in a world where internet is starting to play a larger role in each of our lives. Social networking is show a remarkable growth rate in the recent years, “Twitter” and “Facebook” in particular. Twitter uses storm to handles the millions of tweets that it receives. This is handled by the use of spout and bolts. This paper describes in detail the way different trends in twitter are discovered and executed.

**Introduction**

Storm is a real time computation system which is free and is open source. Storm makes the handling of unbound real time processing of unbounded data reliable and very easy like how hadoop handles is batch processing. The best part of storm is it can be used with any programming language.



Storm has many use cases: real time analytics, online machine learning, continuous computation, distributed RPC, ETL, and more. Imagine processing millions of tuples per second one each node, that’s the speed in which the storm was benchmaked. It can be made to grow easily by adding nodes thus making it scalable, because it takes the data in streams storm can guarantee that the data will be processed. And on top all of this it’s very easy to setup and operate. Also it integrates with the queuing and databases that we use.

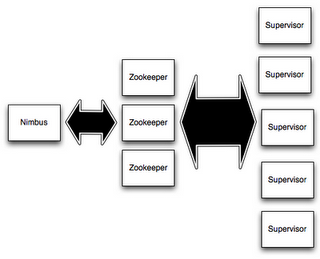
Storm uses topologies to handle the data. A Storm topology consumes streams of data and processes those streams in arbitrarily complex ways, repartitioning the streams between each stage of the computation however needed.

**Storm**

Storm has a few components that basically constitute to how it works. Topologies that was mentioned earlier is comparable to that of a job on the Map Reduce in hadoop. The primary difference would be that of the job and the topologies is that jobs will eventually finish, but the topologies will never end producing messages forever until the user kills it.

Storm uses the same master-slave architecture like hadoop. The master node runs a daemon called the “Nimbus” which is like the “JobTracker”. Nimbus is responsible for distributing code among the different clusters, assigning tasks to machines and monitoring for failures.

The worker nodes or the slaves here run a daemon called the “supervisor”. The supervisor listens to the nimbus with respect to the work assigned to it and the start and stop of the work process as necessary. A topology is said to be running topology if it consist of many worker processes spread across many machines.



All the coordination between the daemons the “nimbus” and the “supervisor” are over seen by the “Zookeeper” cluster. The nimbus and the supervisor are run on fail fast and stateless machines, so the zookeeper is in charge of keeping the states on local disk. What this does is it makes the system more robust by not crumbling even if a large number of nimbus or supervisors fail, it just seems to start like nothing happened.

To perform the real time processing of data in storm we use topologies. Topology is the graph of computation. The nodes in the graph have processing logic; the links between the nodes show how the data should be passed between the nodes.

Stream is a core part of storm. Stream is basically the unbound sequence of tuples. Storm gives us the ability to change a stream of data into a new stream in a distributed and reliable manner.

The stream transformation is done using “spouts” and “bolts”. Spout is the source of the streams. When it comes to message passing, the main parameter that needs to be addressed is that of message reliability. If a message cannot be processed, we need to determine what needs to be done from there on – will the message be passed again or will it just exit without trying again? These are some examples of what needs to be addressed. In storm, the message reliability is the responsibility of the author and the needs of the same may vary from one topology to another. In order for the spouts to collect data efficiently from various sources, there are some methods that can be implemented such as the direct connection. As we see, it is important to define the spout connection based on the problem/project that one is working upon.

Bolts can consume any number of input streams then it processes them and can emit them as a stream. Bolts can do anything from run functions, filter tuples, do streaming aggregations, do streaming joins, talk to databases, and more.

Complex stream transformations, like computing a stream of trending topics from a stream of tweets, require multiple steps and thus multiple bolts. There are two main types of bolts which are reliable and unreliable bolts. The reliable bolts ack or fail – which is the message the bolt sends back to storm, so that the storm knows when a message is acknowledged or fails. If it fails, the storm notifies the spout so that the message can be sent again. This is the most popular way to keep track of the messages that have been sent.

Another technique, anchoring, is also popular. In anchoring, the way messages are kept track of is that the original spout instance includes a reference to the originating spout in the message tuple. A bolt can also emit to multiple streams using the emit() function.

Topology which is the top level abstraction has the spouts and bolts packed into it. This is then submitted to the storm cluster to be executed.

Storm uses tuples as its data model. A tuple is a named list of values, and a field in a tuple can be an object of any type.

**How to run storm topology:**

The storm cluster is analogous to that of the hadoop cluster. Using this analogy, we have “topologies” in storm, correspond to map-reduce jobs in hadoop. The difference is that the map-reduce jobs end whereas the topologies do not, unless they are killed.

In order to run storm, there are two mode by which it can be done. The first is the local mode, in which the storm topology is set up on the local JVM machine. This mode is more widely used as it is easy to see all the components working together. In order to achieve this, the storm development dependencies need to be downloaded onto the local machine. The other method is the remote mode where in, as the name suggests, the storm topology cluster is running on different machines. In our project, we have used local mode and initially, we have to set up the topology. In order to do so, first we need to set up the zookeeper which is used to coordinate the cluster. A few things need to be kept in mind after setting up the zookeeper, one which is that it needs supervision since it will fail quickly and exit the process if it encounters any errors. The next step involved is to set up the nimbus cloud along with all of its dependencies. Next, we download the storm release and the file may be extracted onto the nimbus and its worker (bolts) machines.

**Implementation**:

This project’s purpose is to analyze twitter data using storm cluster. To imitate a spout which emits data frequently, a flat file that contains the twitter data with fields like username, location, timestamp and tweet will be used as a spout. Also the topology is run in a local mode as this project was completed on a pseudo distributed environment (One Nimbus node, one zookeeper node and one slave node). Storm does not have its own storage system to collect the tuples emitted by a bolt. In order to persist the tuples that gets accumulated on a data structure like hash map, we dump it to relational database like MySQL.

The following technologies were used to complete this project:

1. Storm 0.8
2. Zookeeper
3. ZeroMQ
4. MySQL

The topology consists of the following components:

Spout: TwitterSpout – To read tweets.csv file line by line and emit each line as a tuple

Bolts: TwitterCounterLocation – To count tweets based on location. The output will consist of the location and the number of tweets tweeted from that location. TwitterHashTag –This bolt will filter all the tweets that contain a hashtag (#) symbol.

HashTagCount – This bolt will emit each hashtag term followed by the count of each hashtag.

FilterByhour – This bolt will emit the time period (for example 11:00 – 12:00, for all tweets that has its timestamp between 11:00 to 12:00)

TweetCountHour – This bolt will emit each time period followed by the count (which signifies the number of tweets that were tweeted for the given time period)

EmitUser – This bolt will emit all the twitter users.

EmitUserMentions – This bolt will emit all tweets that contain user mention (@) symbol.

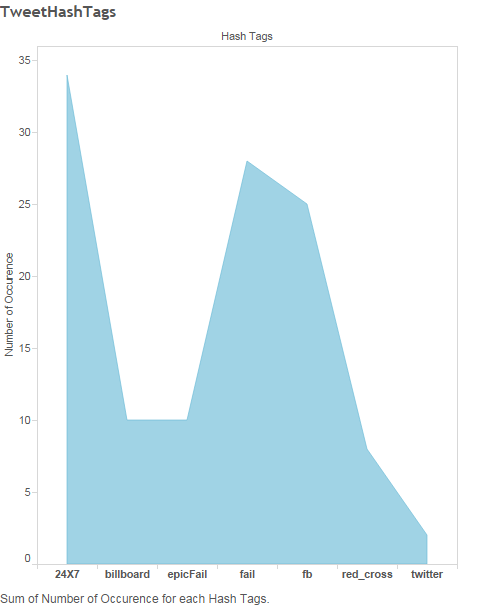
UserMentionsJoin – This bolt will join the data that comes from the above bolts based on the user name.

FilterByHourDate – This bolt will group all the tweets that have been tweeted on same day and that fall between same time periods.

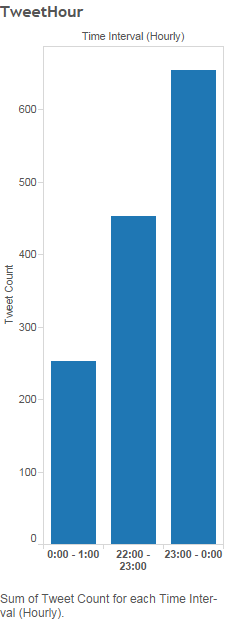
TweetCountHourDate – This bolt will count the tweets that have the same date and time period.

Using the above topology (spout/bolt) setup we calculate the following from the input dataset

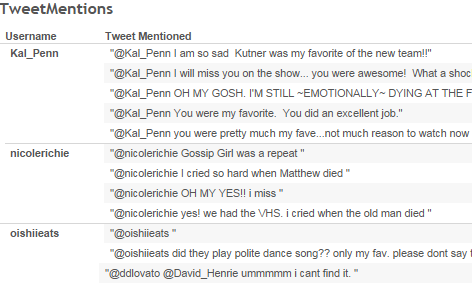
1. To count the occurrence of hashtags



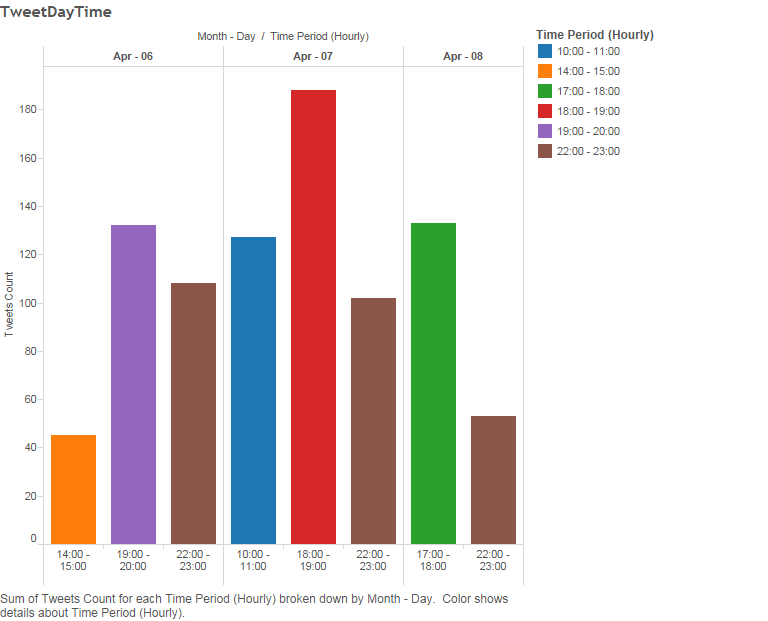
1. To count tweets on an hourly basis



1. To perform join between username and the tweets to find out the tweets that contain the user mentions (self-join)



1. To count most number of tweets tweeted in an hour for the entire date range



1. To count tweets based on location

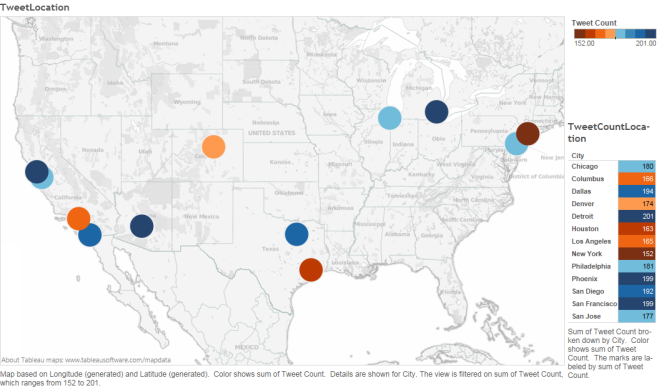


Tableau - A Business Intelligence tool was used to generate the above charts, maps and tables.

**Acknowlegements**

We would like to thank, Dr. Latifur Khan, Professor, Department of Computer Science, The University of Texas at Dallas, for this providing this opportunity.

**References**

1. Jonathan Leibuiusky, Gabriel Eisbruch, Dario Simonassi – “Getting started with Storm”
2. <https://github.com/nathanmarz/storm/>
3. <http://storm-project.net/>
4. <http://jayatiatblogs.blogspot.com/2011/11/storm-installation.html>
5. <http://www.michael-noll.com/tutorials/running-multi-node-storm-cluster/>