**Final Project Report**

**Real time Stream processing using Kafka and Storm**

**Deeptha kannan Rangarajan**

Hadoop is a powerful framework to do big-data analysis. It takes care of the complex work of utilizing hardware resources efficiently in a cluster, breaking huge files into chunks, running our map-reduce programs, monitoring, etc

But there is a problem with Hadoop which we all might have thought of at one point of time or another. **IT REQUIRES A FILE.** But most of the times, we have streaming data coming in to be processed. What if we have fast-big-data? Does hadoop support real time data processing, answer is NO.

Next big questions is, do we have a framework equivalent of hadoop that supports big-fast-real time data? Fortunately, yes. In fact, it is combination of two beautiful open sourced projects, Kafka and Storm.

Let’s look at these two technologies in detail

**Storm**

Storm is a framework developed at BackType by Nathan Mars and team. It was written in Clojure and Java. This project was opensourced in 2011.

**What is Storm?**

Storm is a distributed, reliable, fault-tolerant system for processing streams of data.

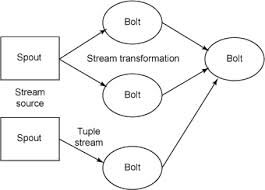
Hadoop relies on batch processing, Storm is a real-time, distributed, fault-tolerant, computation system. Like Hadoop, it can process huge amounts of data but it does in real time

**Introduction to Storm components**

The input stream of a Storm cluster is handled by a component called a **spout**.

The spout passes the data to a component called a **bolt**, which transforms it in some way. A bolt either persists the data in some sort of storage,

or passes it to some other bolt.



A Storm cluster as a chain of bolt components that each make some kind of transformation on the data exposed by the spout.

The arrangement of all the components (spouts and bolts) and their connections is called a **topology**

**Use cases**

stream processing

real-time analytics

distributed RPC

Replaces “Network of Queues”

**Features of storm**

• Allocation of work across nodes

• Queuing / Messaging

• Guaranteed message processing

• Fault Tolerance

• Scalability

• Performance

• Develop in multiple languages

**Guaranteed message processing**

In Storm, it is the author’s responsibility to guarantee message reliability according to

the needs of each topology. This comes with a price. A reliable topology must manage

lost messages, which requires more resources.

**Storm project**

In my demo, I modified the example file that comes with the Storm package to make it work for my problem.

This show how to create a topolology.

We use topology builder to create topologies and submit it to the local cluster to Storm submitter based on the mode that we want storm to run on.

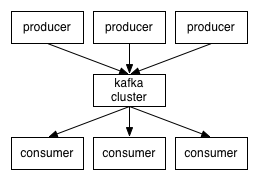


**Kafka**

Originally developed by Linkedin to monitor user actions in their website then open sources in 2011.

**What is Kafka?**

Kafka is a distributed, partitioned, replicated commit log service. It provides the functionality of a messaging system, but with a unique design.

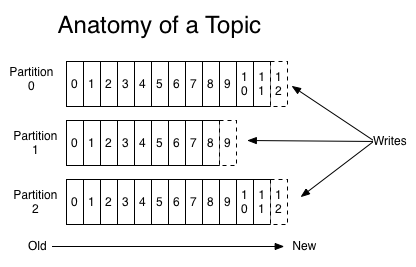


As you can see in the picture above, Kafka has producer-consumer model. Producers write to Kafka cluster and consumers consume from the cluster.

Kafka clusters are highly scalable, durable and provide high write and read speed.

**Topics and Partitions**

A message belongs to a topic and partitions are a way to divide the messages for a topic in a cluster.



**Consumer API**

Kafka provide two consumer APIs.

**HighLevel consumer:**

Easy to implement but has limited functionality. Cannot read old messages

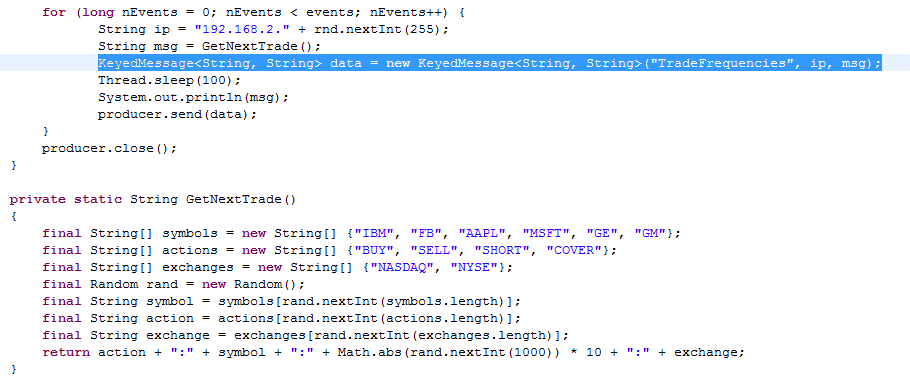
**Simple consumer:**

Takes time to understand implementation details of Kafka. You need it to use this efficiently. You have lot of control when using this but I find this a bit tedious to understand and code.

**Kafka Project**

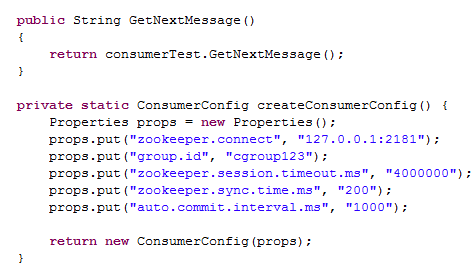
Kafka code is spread across two projects.

Producer code simulates Trade information



Chooses from a set of predefined securities and actions and publishes to Kafka brokers

Kafka consumer spout that helps Storm integrate with Kafka messages



**Conclusion:**

I really enjoyed working with these two frameworks. I had some issues setting up and start coding as I'm new to github and linux style environments. But its worth the pain as this exposed me to a new world of programming.