A Survey on Storm @Twitter

CS5543 Real-Time Big Data Analytics- Pooja Shekhar(23)

Abstract

Storm – A real-time computation system is used at Twitter. What Hadoop did for batch processing, Storm does for real time. The paper illustrates the architecture of Storm at Twitter. It goes further explaining how Storm achieves the power it has using the key component topology.

Keywords

Topology, Bolt, Spout, Nimbus, Zookeeper," At least once"," At most once".

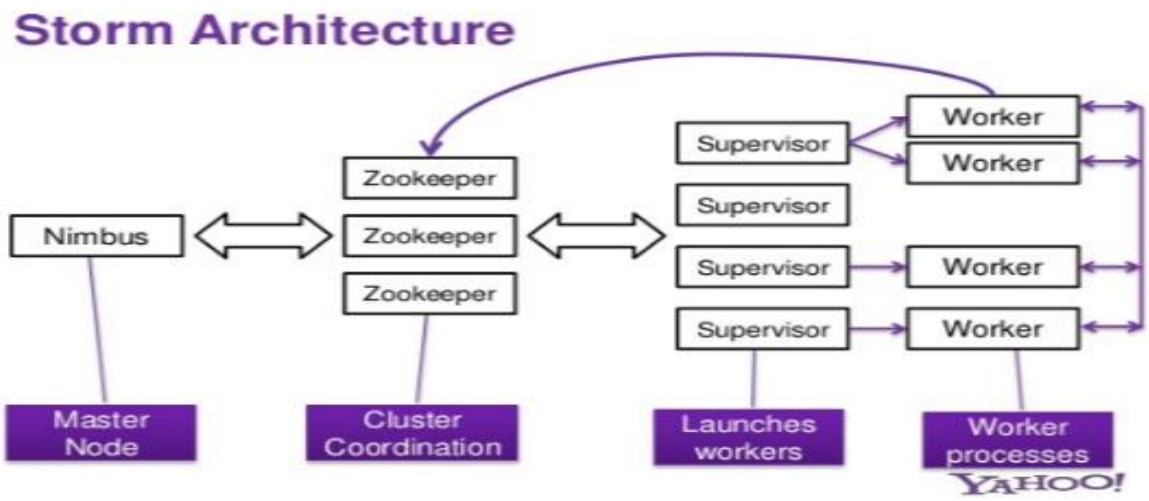
Introduction

Storm works on processing a real time streaming data. It was developed at BackType by Nathan Marz and later taken up by Twitter. This was his tweet when the idea of developing Storm hit him.

Storm topologies helps it to become scalable & fault tolerant system. This was one of the earliest of its kind. Many new systems have developed later like Samza, Spark Streaming etc. This was developed to help enterprises for real time data processing which is the need of this hour. The paper goes further to discuss the architecture and topology used at Twitter. In my opinion, even after the advent of new real time systems in place, Storm still outstands them in performance.

Storm Overview & Architecture

Client submits topologies to a master node that is Nimbus. Actual task execution is done by Worker nodes which are slave nodes. The coordination between them is done by Zookeeper. Each worker has several worker processes which runs a JVM which runs one or more executors.



Nimbus and Supervisors are stateless which means they do not record the history of tasks that they are currently executing. Zookeeper comes to their rescue in this. This is why the even when the Nimbus fails while running a topology, the tasks at executors continues. Until then, Nimbus is started and it recovers its state from Zookeeper from its failure point. Therefore, the process continues to progress.

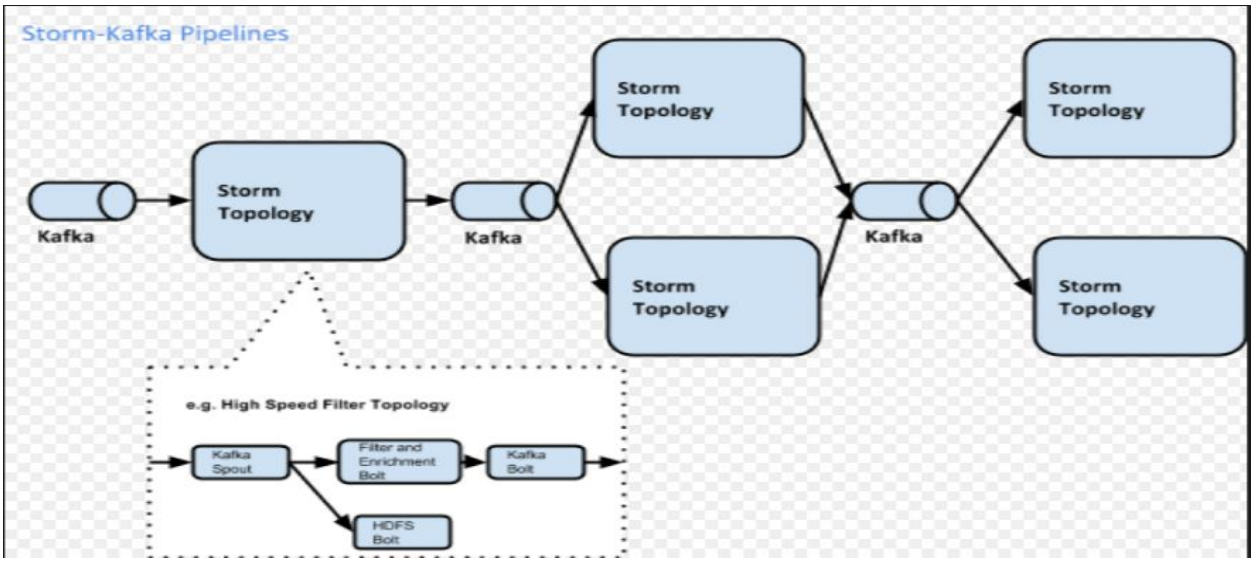


Figure from http://hortonworks.com/blog/storm-kafka-together-real-time-data-refinery/

Storm ensures “At Least Once” & “At most once” delivery semantics. “At least once” semantics is achieved using “Acker Bolt”.

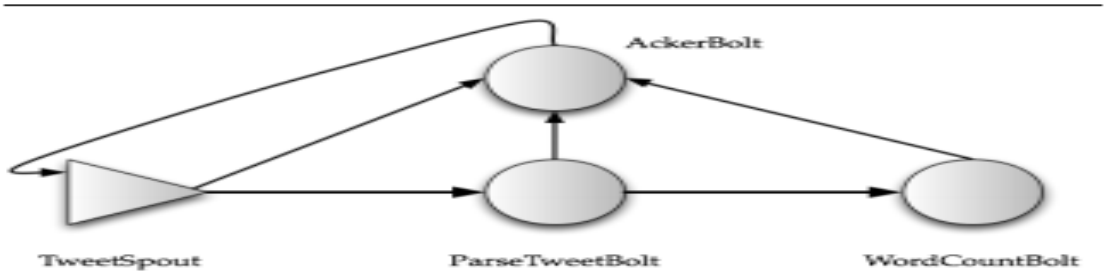


Figure 6. Augmented word count topology

As soon as a tuple leaves a spout it registers itself by sending a signal to Acker Bolt. The moment the processing of tuple is done, it sends an 'ack' signal for that particular tuple, then Acker Bolt communicates to the Spout to flush out that tuple. This is achieved by assigning a message id to each tuple." At most once "is easily achievable by removing the above system of Acker Bolt.

Operational Challenges

The first issue faced was the zookeeper's very frequent writes on local disk which slowed down the entire system. Second challenge faced was tuning of "max spout pending parameter" – because a very low value lowers the number of topologies submitted whereas a higher value overloads. They have an algorithm in place to best tune this parameter using a metric called "progress".

Conclusion/Future Work

Storm has a really bright chances of extending its features like automating the optimization of the topology. They also want to include "Exact Once Semantics" to it.

Continued…

A Survey on Spark: Cluster Computing with Working Sets

CS5543 Real-Time Big Data Analytics – Pooja Shekhar(23)

Abstract

Spark is a framework which suits a class of applications which reutilizes the results of map-reduce jobs. Spark ensures this by doing in memory computation. This has been achieved by a read only data structure called resilient distributed datasets.

Keywords

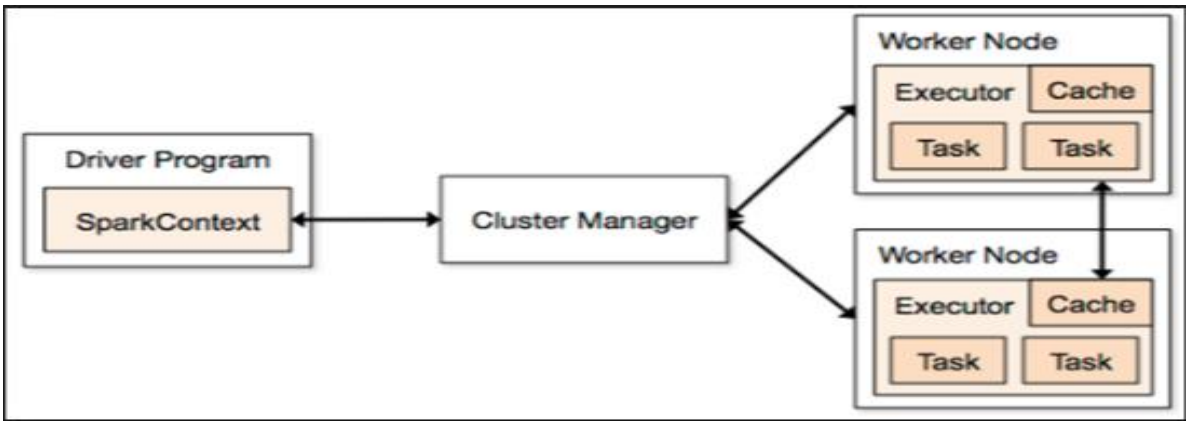
Resilient Distributed datasets(RDDs), Action, Transformation, map, flatmap ,reduce

Introduction

MapReduce aims at batch processing hence it's not interactive. Similar systems then in market were Dryad and Merge. But these did not permit parallel use of working set of data. This was achieved by Spark. It also helps in making this computation interactive.

Spark Architecture

RDDs can be constructed by 4 methods – "parallelizing" a Scala collection, Processing an existing RDD etc. Spark has lazy evaluation system which was implemented using cache function. There are several parallel operations that can be used to transform RDD – reduce, collect etc. Concept of shared variables helps us to work in a distributed mode – Broadcast variables & Accumulators. These concepts though has been used from earlier concepts but its use in implementation of developing a distributed framework was unique.



The machine library of Spark is an addition to all the features. Therefore, it can be used for statistical learning methods too. The algorithm used in this research is ALS(Alternating Least Squares).This algorithm is intended to be used in collaborative filtering, predictions etc. Spark is built on the top of “Mesos” a cluster operating system.

Internally RDDS implement same set of interfaces – getPartition, getIterator,getPreferredLocation which return partition ids. Difference in types of RDDs is due to the way they implement these interfaces.

Continued ….

A Survey on Kafka: A Distributed Messaging System for Log Processing

CS5543 Real-Time Big Data Analytics- Pooja Shekhar (23)

Abstract

Kafka is a distributed *pull model* based messaging system used and developed at LinkedIn. This system has the potential to huge logs.It works both real time and batch processing. The performance of Kafka is much better than the existing log aggregators**.**

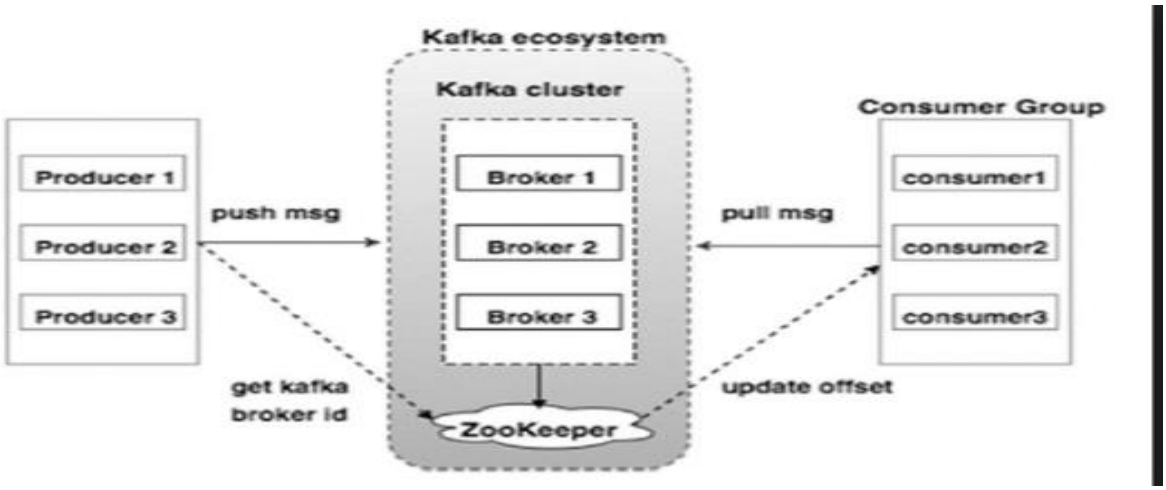
Keywords

Topics, Kafka Producer, Kafka Consumer, Broker, Delivery Semantics

Introduction

Kafka is a log aggregator and publish-subscribe messaging system. It combines the benefits of two worlds – traditional log aggregators and messaging systems. There are several systems like Kafka in the market like IBM Websphere MQ, Facebook's Scribe, Clodera's Flume. One major difference between them and Kafka is, Kafka is pull model.

Overview Kafka Architecture



https://www.tutorialspoint.com/apache_kafka/apache_kafka_cluster_architecture.htm

Topics is a stream of messages belonging to a particular class. Producers publish the messages to the broker. Broker stores the indices to these messages and stored the messages using partitions. Moreover, Kafka ensures that a message is pulled by the consumer only once by addressing each using offset id. The offset of new message is calculated using length of new message plus the offset.

Sample producer code:

```
producer = new Producer(…);
message = new Message("test message str".getBytes());
set = new MessageSet(message);
producer.send("topic1", set);
```

Sample consumer code:

```
streams[] = Consumer.createMessageStreams("topic1", 1)
for (message : streams[0]) {
  bytes = message.payload();
  // do something with the bytes
}
```

Even with a stateless broker Kafka manages to deliver the messages using the time-based SLA for retention. This removes the overhead of maintaining the state of each message.

Kafka guarantees "at least once "delivery.

There are few design decisions which makes Kafka more robust.

Partitioning the topics induces parallelism. Not having a central master node decentralizes the system.

Continued …..

A Survey on Twitter Heron: Stream Processing at Scale

CS5543 Real-Time Big Data Analytics- Pooja Shekhar (23)

Abstract

After Storm serving Twitter since long time, Heron takes its place. With the increasing data load on the servers, they needed a scalable system and Heron was the solution.

Keywords

Topology, Bolts, Spouts, etc

Introduction

The limitations of Storm caused the development of Heron. Challenges in debugging the logs in addition to inefficient usage of resources led to the better design principles of Heron.

There were issues in the design of Nimbus & Zookeeper which were bottleneck in the performance in times of increased loads in Storm. Nimbus is overloaded with several activities like scheduling, monitoring for several topologies. Zookeeper controls the number of topologies run at a time. Large number of topologies at any instant leads to huge performance issues. Storm has no mechanism where if bolts is unable to process any tuple, then the tuple is dropped.

Overview Heron Architecture

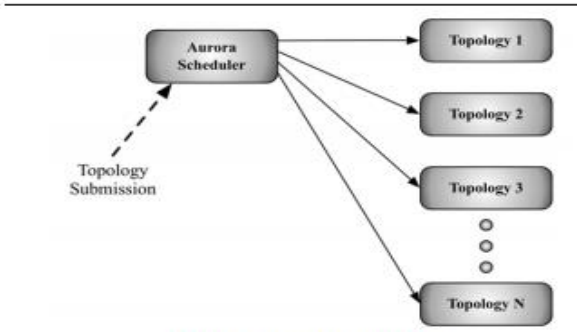


Figure 3: Heron Architecture

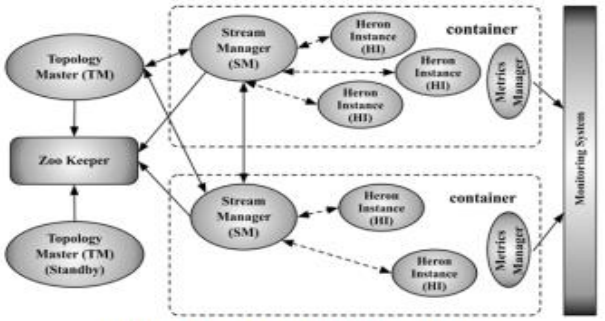


Figure 4: Heron Topology Architecture

Heron runs topologies like Storm but it has no Nimbus in place. Instead we have Aurora Scheduler in place. Moreover it is also compatible with other schedulers like Yarn, Mesos etc.Each topology is run as an Aurora Job. Aurora Job runs in containers which triggers two processes – Topology Master & Stream Manager. As discussed earlier the challenges of Storm that it had no reverse control at the receiver's end. Heron has a mechanism to deal with this.

Important features in the design of Heron are efficient use of resources , since a heron instance runs at a time so its easy to debug.

The delivery semantics ensured by Heron is same as Storm – “At least once” & “ At most once”

It has Metrics Manager that monitors all the components of the system and reports them. There are few more components added in Heron – Heron UI ,Heron Tracker , Heron Viz.

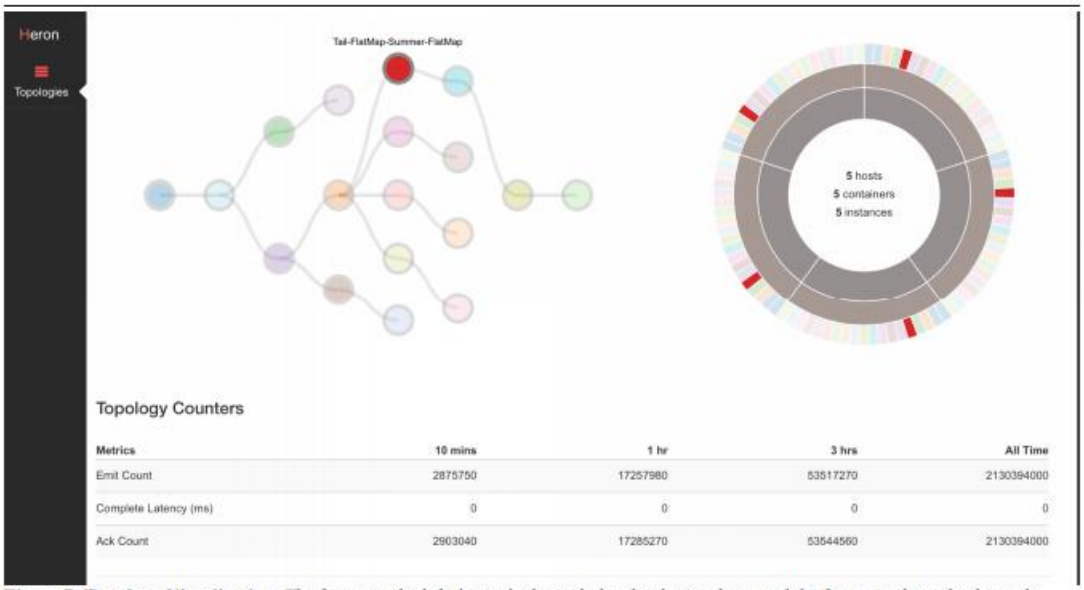


Figure 7: Topology Visualization. *The figure on the left shows the logical plan for the topology, and the figure on the right shows the current location of components of the physical plan for the topology. Clicking on a logical plan component (e.g. the red bolt in the figure on the left) highlights the location of the containers on the map on the right. The table below the two figures shows key metrics.*

Conclusion

Heron in short deals with challenges faced while using Storm at Twitter. With the ever increasing, in my opinion, maybe we have scalability issues with Heron too.