Real Time Data Streaming Using Apache Storm and Apache Flink

Team 11

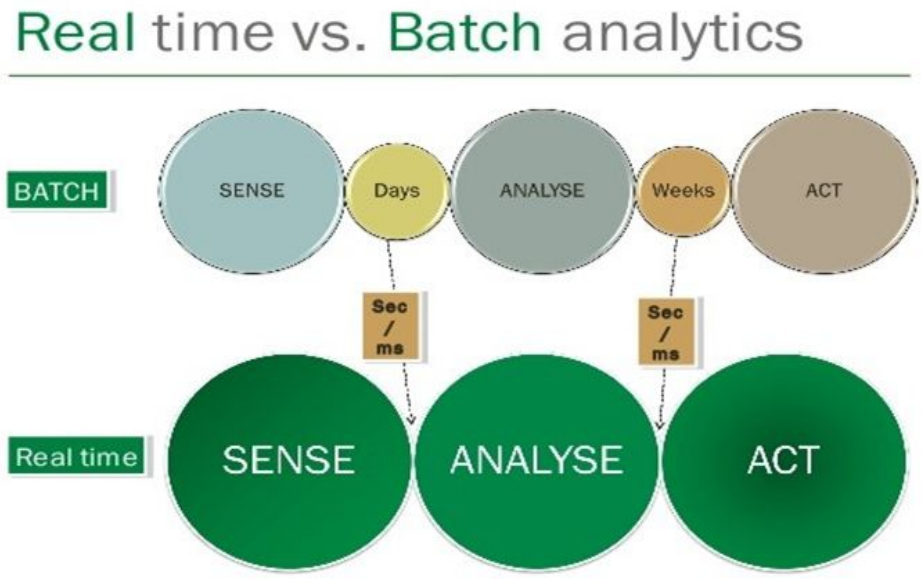
Krutika Dedhia

Kinjal Gada

Ankur Vora

- Prof. Srikanth Krishnamurthy

Batch Processing v/s Real Time Processing



Since data is being entered and processed immediately in real-time processing, the data can be accessed and corrected immediately by the user. Data that is processed in a batch must follow a structured protocol for correcting errors, which often takes more time. Real-time processing produces data that is more up-to-date than data processed in batches. It is also likely to produce more accurate data, since the input tools are readily available to users.

Batch processing can be more cost-effective, using fewer peripheral devices than real-time processing, though the cost savings is reduced as the price of peripheral devices decreases over time. Batch processing also allows a business to schedule when the computer is to be used, allowing for more efficient use of computer hardware and personnel time. Batches can be programmed to be processed at night, and are ready and waiting for workers the next morning.

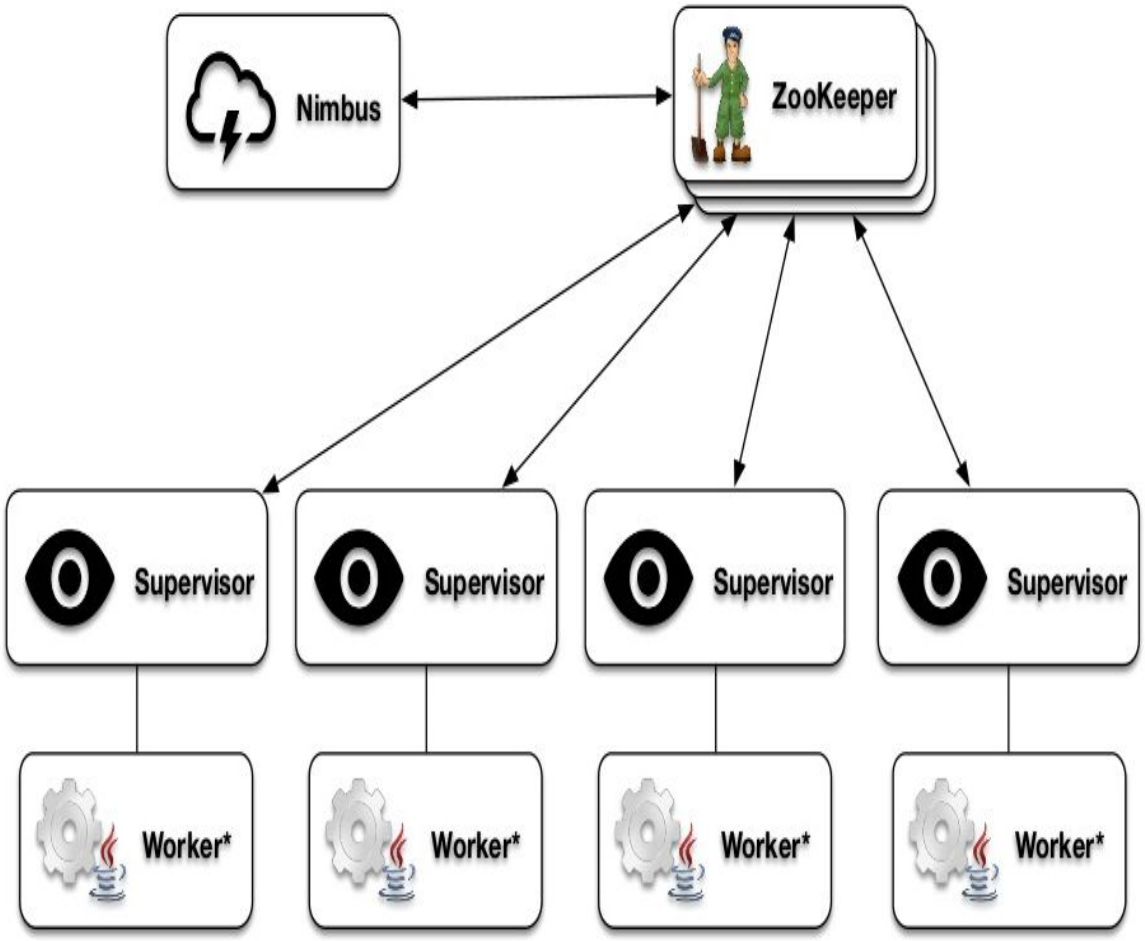
Real - Time Data Processing Applications:

- Risk and Fraud detection
- E-commerce

Apache Storm

- Free open source software.
- Distributed real- time data processing platform.
- Able to process large volumes of data per second.
- Originated by Twitter and is currently being incubated at the Apache Software Foundation

Storm Architecture



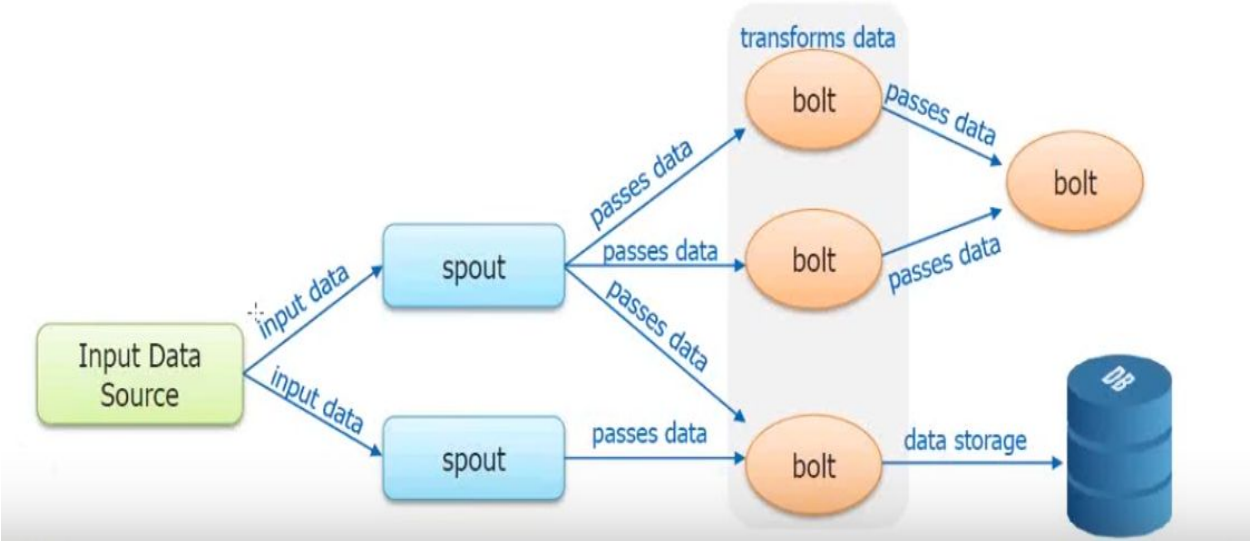
Apache Storm has two type of nodes, Nimbus (master node) and Supervisor (worker node). Nimbus is the central component of Apache Storm. The main job of Nimbus is to run the Storm

topology. Nimbus analyzes the topology and gathers the task to be executed. Then, it will distributes the task to an available supervisor.
A supervisor will have one or more worker process. Supervisor will delegate the tasks to worker processes. Worker process will spawn as many executors as needed and run the task. Apache Storm uses an internal distributed messaging system for the communication between nimbus and supervisors.

Apache ZooKeeper is a service used by a cluster (group of nodes) to coordinate between themselves and maintaining shared data with robust synchronization techniques. Nimbus is stateless, so it depends on ZooKeeper to monitor the working node status.

Apache Storm also have an advanced topology called Trident Topology with state maintenance and it also provides a high-level API like Pig.

Storm Topology



Apache Storm reads raw stream of real-time data from one end and passes it through a sequence of small processing units and output the processed / useful information at the other end.

Let us now have a closer look at the components of Apache Storm:

- Spouts: Source of stream. Generally, Storm accepts input data from raw data sources like Twitter Streaming API, Apache Kafka queue, Kestrel queue, etc. Otherwise you can write spouts to read data from data sources. “ISpout" is the core interface for implementing spouts. Some of the specific interfaces are IRichSpout, BaseRichSpout, KafkaSpout, etc.

- Bolts: Bolts are logical processing units. Spouts pass data to bolts and bolts process and produce a new output stream. Bolts can perform the operations of filtering, aggregation, joining, interacting with data sources and databases. Bolt receives data and emits to one or more bolts. “IBolt” is the core interface for implementing bolts. Some of the common interfaces are IRichBolt, IBasicBolt, etc.

Workflow of Apache Storm

- Initially, the nimbus will wait for the “Storm Topology” to be submitted to it.
- Once a topology is submitted, it will process the topology and gather all the tasks that are to be carried out and the order in which the task is to be executed.
- Then, the nimbus will evenly distribute the tasks to all the available supervisors.
- At a particular time interval, all supervisors will send heartbeats to the nimbus to inform that they are still alive.
- When a supervisor dies and doesn’t send a heartbeat to the nimbus, then the nimbus assigns the tasks to another supervisor.
- When the nimbus itself dies, supervisors will work on the already assigned task without any issue.
- Once all the tasks are completed, the supervisor will wait for a new task to come in.
- In the meantime, the dead nimbus will be restarted automatically by service monitoring tools.
- The restarted nimbus will continue from where it stopped. Similarly, the dead supervisor can also be restarted automatically. Since both the nimbus and the supervisor can be restarted automatically and both will continue as before, Storm is guaranteed to process all the task at least once.
- Once all the topologies are processed, the nimbus waits for a new topology to arrive and similarly the supervisor waits for new tasks.

Implementation on stand-alone (local mode)

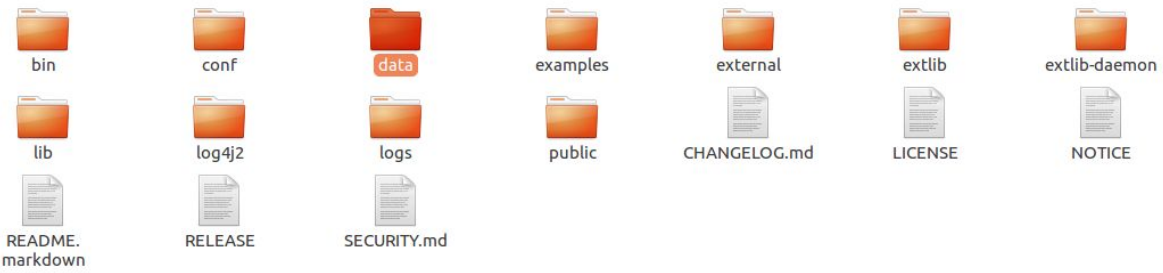
Steps:

1. Download Apache Storm from link.
2. Update file storm.yaml in config

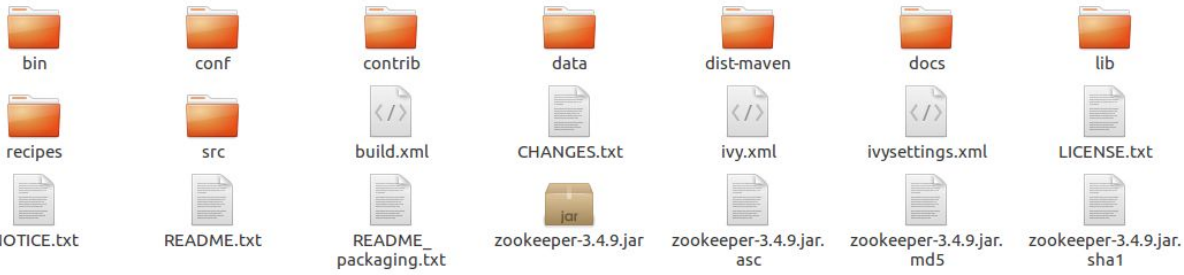
3. Change as follows

```
########### These MUST be filled in for a storm configuration
 storm.zookeeper.servers:
 - "localhost"
#     - "192.168.0.24"

#     - "server2"
#
 storm.zookeeper.port: 2181
 nimbus.host : "localhost"
 nimbus.thrift.port: 6627

 ui.port: 8080

 storm.local.dir: "/home/kinjal/Downloads/apache-storm-1.0.2/data"
# java.library.path
 supervisor.slots.ports:
 - 6700
 - 6701
 - 6702
 - 6703
```

4. Create a folder under storm/ as data



5. Download zookeeper from [link]
6. Create data directory under zookeeper/ as data



7. Create zoo.cfg same as zoo_sample.cfg inside zookeeper/config

8. Update file as follows

zoo.cfg × storm commands to run

```
# The number of milliseconds of each tick
tickTime=2000
# The number of ticks that the initial
# synchronization phase can take
initLimit=10
# The number of ticks that can pass between
# sending a request and getting an acknowledgement
syncLimit=5
# the directory where the snapshot is stored.
# do not use /tmp for storage, /tmp here is just
# example sakes.
dataDir=/home/kinjal/Downloads/zookeeper-3.4.9/data
# the port at which the clients will connect
clientPort=2181
# the maximum number of client connections.
# increase this if you need to handle more clients
#maxClientCnxns=60
#
# Be sure to read the maintenance section of the
# administrator guide before turning on autopurge.
#
# http://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc_maintenance
#
# The number of snapshots to retain in dataDir
#autopurge.snapRetainCount=3
# Purge task interval in hours
# Set to "0" to disable auto purge feature
#autopurge.purgeInterval=1
```

9. Run cmd on terminal and go to sudo gedit ~/.bashrc and add following to the end of the file based on your java, zookeeper and apache storm location

```
kinjal@dell:~/Downloads$ sudo gedit ~/.bashrc.sh
```

```
#Set HIVE_HOME
#export HIVE_HOME="/usr/local/bin/hive-2.1.0"
#PATH=$PATH:$HIVE_HOME/bin
#export PATH

#Set Zookeeper Path
export JAVA_HOME=/usr/local/lib/jdk1.7.0_79
export ZOOKEEPER_HOME=/home/kinjal/Downloads/zookeeper-3.4.9
export PATH=$PATH:$JAVA_HOME/bin:$ZOOKEEPER_HOME/bin
#Set Storm Path
export STORM_HOME=/home/kinjal/Downloads/apache-storm-1.0.2
export PATH=$PATH:$JAVA_HOME/bin:$STORM_HOME/bin:$ZOOKEEPER_HOME/bin
```

10. Now Configuration is done to start apache storm first start zookeeper
11. Go to zookeerper and run cmd as below

```
kinjal@dell:~/Downloads/zookeeper-3.4.9/bin$ zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /home/kinjal/Downloads/zookeeper-3.4.9/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
```

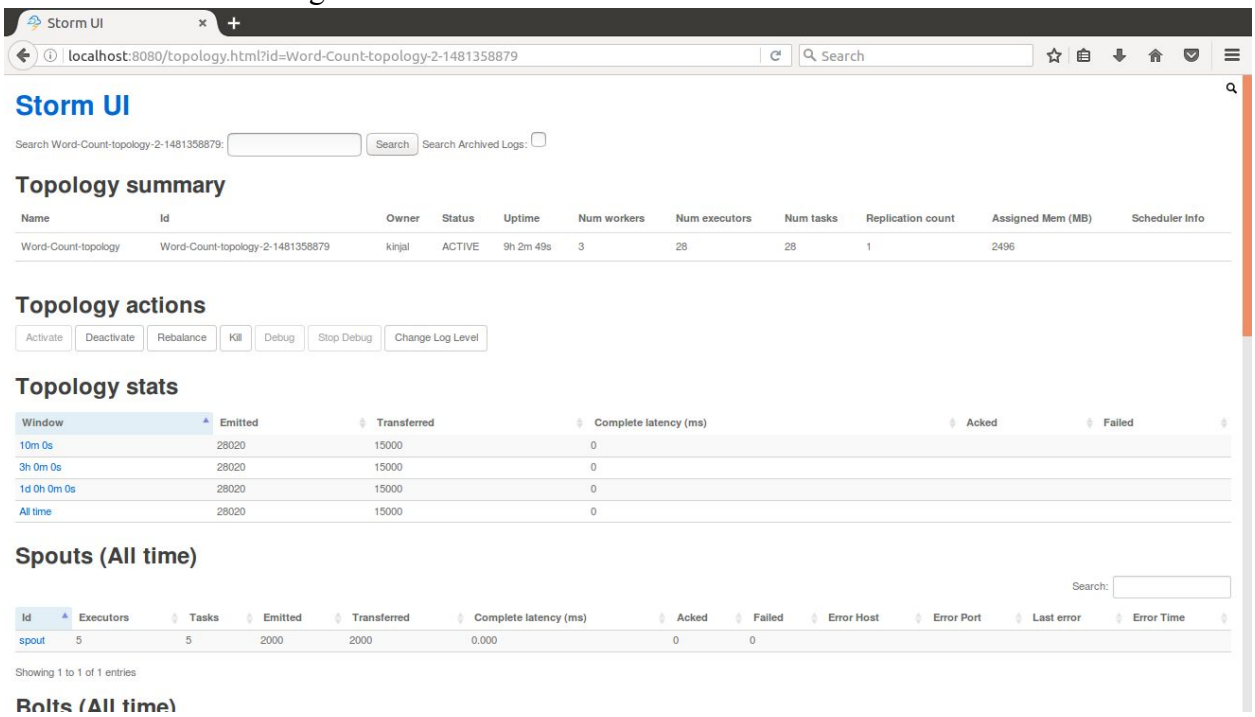
12. Then in another 3 terminals start apache storm by going inside bin directory of storm folder and run storm nimbus, storm supervisor and storm ui as below

```
^Ckinjal@dell:~/Downloads/apache-storm-1.0.2/bin$ storm nimbus
Running: /usr/local/lib/jdk1.7.0_79/bin/java -server -Ddaemon.name=nimbus -Dstorm.options= -Dstorm.home=/home/kinjal/Downloads/apache-storm-1.0.2
 -Dstorm.log.dir=/home/kinjal/Downloads/apache-storm-1.0.2/logs -Djava.library.path=/usr/local/lib:/opt/local/lib:/usr/lib -Dstorm.conf.file= -cp
 /home/kinjal/Downloads/apache-storm-1.0.2/lib/log4j-slf4j-impl-2.1.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/minlog-1.3.0.jar:/home/kinj
al/Downloads/apache-storm-1.0.2/lib/slf4j-api-1.7.7.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/kryo-3.0.3.jar:/home/kinjal/Downloads/apach
e-storm-1.0.2/lib/reflectasm-1.10.1.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/storm-rename-hack-1.0.2.jar:/home/kinjal/Downloads/apache-s
torm-1.0.2/lib/disruptor-3.3.2.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/asm-5.0.3.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/log4
j-over-slf4j-1.6.6.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/servlet-api-2.5.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/log4j-core
-2.1.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/log4j-api-2.1.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/storm-core-1.0.2.jar:/home
/kinjal/Downloads/apache-storm-1.0.2/lib/objenesis-2.1.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/clojure-1.7.0.jar:/home/kinjal/Downloads
/apache-storm-1.0.2/conf -Xmx1024m -Dlogfile.name=nimbus.log -DLog4jContextSelector=org.apache.logging.log4j.core.async.AsyncLoggerContextSelecto
r -Dlog4j.configurationFile=/home/kinjal/Downloads/apache-storm-1.0.2/log4j2/cluster.xml org.apache.storm.daemon.nimbus
```

```
kinjal@dell:~/Downloads/apache-storm-1.0.2/bin$ storm supervisor
Running: /usr/local/lib/jdk1.7.0_79/bin/java -server -Ddaemon.name=supervisor -Dstorm.options= -Dstorm.home=/home/kinjal/Downloads/apache-storm-1
.0.2 -Dstorm.log.dir=/home/kinjal/Downloads/apache-storm-1.0.2/logs -Djava.library.path=/usr/local/lib:/opt/local/lib:/usr/lib -Dstorm.conf.file=
 -cp /home/kinjal/Downloads/apache-storm-1.0.2/lib/log4j-slf4j-impl-2.1.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/minlog-1.3.0.jar:/home/
kinjal/Downloads/apache-storm-1.0.2/lib/slf4j-api-1.7.7.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/kryo-3.0.3.jar:/home/kinjal/Downloads/a
pache-storm-1.0.2/lib/reflectasm-1.10.1.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/storm-rename-hack-1.0.2.jar:/home/kinjal/Downloads/apac
he-storm-1.0.2/lib/disruptor-3.3.2.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/asm-5.0.3.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/
log4j-over-slf4j-1.6.6.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/servlet-api-2.5.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/log4j-
core-2.1.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/log4j-api-2.1.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/storm-core-1.0.2.jar:/
home/kinjal/Downloads/apache-storm-1.0.2/lib/objenesis-2.1.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/clojure-1.7.0.jar:/home/kinjal/Downl
oads/apache-storm-1.0.2/conf -Xmx256m -Dlogfile.name=supervisor.log -DLog4jContextSelector=org.apache.logging.log4j.core.async.AsyncLoggerContext
Selector -Dlog4j.configurationFile=/home/kinjal/Downloads/apache-storm-1.0.2/log4j2/cluster.xml org.apache.storm.daemon.supervisor
```

```
kinjal@dell:~/Downloads/apache-storm-1.0.2/bin$ storm ui
Running: /usr/local/lib/jdk1.7.0_79/bin/java -server -Ddaemon.name=ui -Dstorm.options= -Dstorm.home=/home/kinjal/Downloads/apache-storm-1.0.2 -Dstorm.log.dir=/home/kinjal/Downloads/apache-storm-1.0.2/logs -Djava.library.path=/usr/local/lib:/opt/local/lib:/usr/lib -Dstorm.conf.file= -cp /home/kinjal/Downloads/apache-storm-1.0.2/lib/log4j-slf4j-impl-2.1.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/minlog-1.3.0.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/slf4j-api-1.7.7.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/kryo-3.0.3.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/reflectasm-1.10.1.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/storm-rename-hack-1.0.2.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/disruptor-3.3.2.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/asm-5.0.3.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/log4j-over-slf4j-1.6.6.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/servlet-api-2.5.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/log4j-core-2.1.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/log4j-api-2.1.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/storm-core-1.0.2.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/objenesis-2.1.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/clojure-1.7.0.jar:/home/kinjal/Downloads/apache-storm-1.0.2:/home/kinjal/Downloads/apache-storm-1.0.2/conf -Xmx768m -Dlogfile.name=ui.log -DLog4jContextSelector=org.apache.logging.log4j.core.async.AsyncLoggerContextSelector -Dlog4j.configurationFile=/home/kinjal/Downloads/apache-storm-1.0.2/log4j2/cluster.xml org.apache.storm.ui.core
```

13. Go to localhost:8080 to get ui as below



14. Now to run submit word count topology open a new terminal then, first grab the latest version of maven to compile pom.xml by running $ sudo apt-get install maven

```
kinjal@dell:~/Downloads$ sudo apt-get install maven
```

15. Check if maven got installed by running cmd mvn -version

```
kinjal@dell:~/Downloads$ mvn -version
Apache Maven 3.3.9
Maven home: /usr/share/maven
Java version: 1.7.0_79, vendor: Oracle Corporation
Java home: /usr/local/lib/jdk1.7.0_79/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "4.4.0-53-generic", arch: "amd64", family: "unix"
```

16. Go to examples dir as cd apache-storm/examples/storm-starter
17. Run mvn package to compile pom.xml

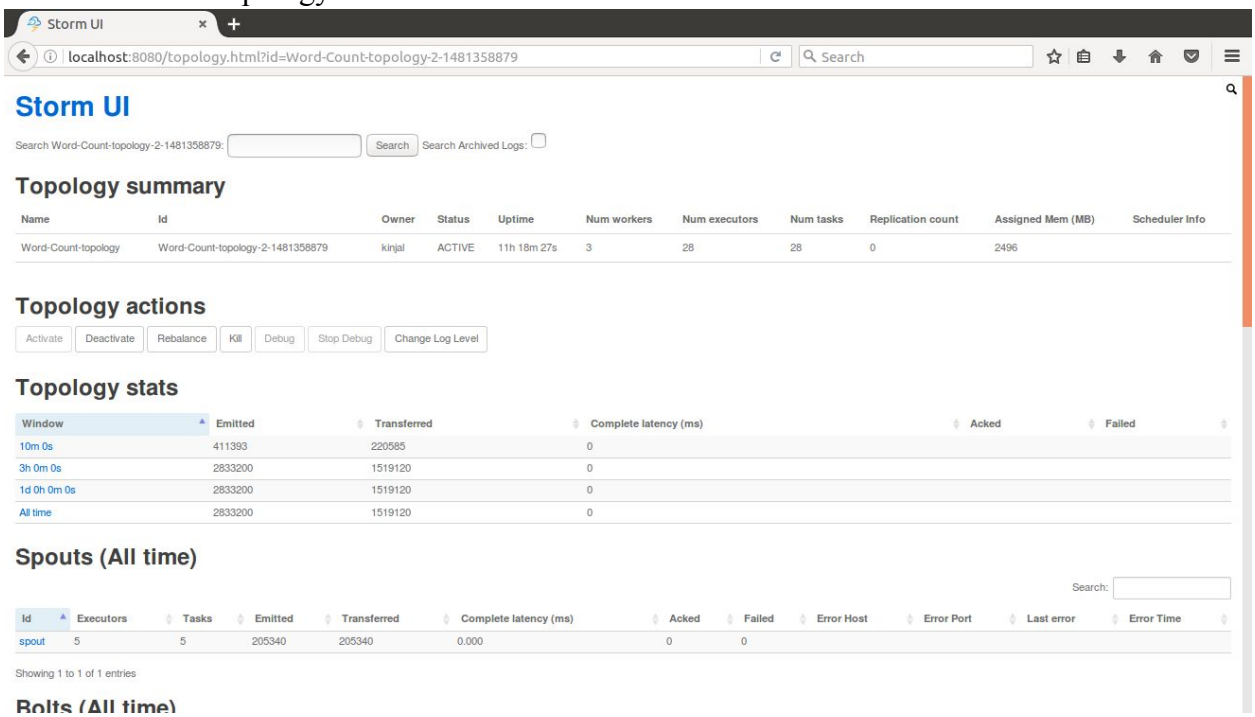
```
kinjal@dell:~/Downloads/apache-storm-1.0.2/examples/storm-starter$ mvn package
[INFO] Scanning for projects...
```

18. Set mvn clean install -DskipTests=true

19. Run
    " storm jar /home/kinjal/Downloads/apache-storm-1.0.2/examples/storm-starter/storm-starter-topologies-1.0.2.jar org.apache.storm.starter.WordCountTopology Word-Count-Topology remote "
    as a a single cmd

```
kinjal@dell:~/Downloads/apache-storm-1.0.2/bin$ storm jar /home/kinjal/Downloads/apache-storm-1.0.2/examples/storm-starter/storm-starter-topologies-1.0.2.jar org.apache.storm.starter.WordCountTopology Word-Count-Topology remote
Running: /usr/local/lib/jdk1.7.0_79/bin/java -client -Ddaemon.name= -Dstorm.options= -Dstorm.home=/home/kinjal/Downloads/apache-storm-1.0.2 -Dstorm.log.dir=/home/kinjal/Downloads/apache-storm-1.0.2/logs -Djava.library.path=/usr/local/lib:/opt/local/lib:/usr/lib -Dstorm.conf.file= -cp /home/kinjal/Downloads/apache-storm-1.0.2/lib/log4j-slf4j-impl-2.1.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/minlog-1.3.0.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/slf4j-api-1.7.7.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/kryo-3.0.3.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/reflectasm-1.10.1.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/storm-rename-hack-1.0.2.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/disruptor-3.3.2.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/asm-5.0.3.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/log4j-slf4j-1.6.6.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/servlet-api-2.5.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/log4j-core-2.1.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/log4j-api-2.1.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/storm-core-1.0.2.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/objenesis-2.1.jar:/home/kinjal/Downloads/apache-storm-1.0.2/lib/clojure-1.7.0.jar:/home/kinjal/Downloads/apache-storm-1.0.2/examples/storm-starter/storm-starter-topologies-1.0.2.jar:/home/kinjal/Downloads/apache-storm-1.0.2/conf:/home/kinjal/Downloads/apache-storm-1.0.2/bin -Dstorm.jar=/home/kinjal/Downloads/apache-storm-1.0.2/examples/storm-starter/storm-starter-topologies-1.0.2.jar org.apache.storm.starter.WordCountTopology Word-Count-Topology remote
1815 [main] INFO  o.a.s.StormSubmitter - Generated ZooKeeper secret payload for MD5-digest: -8405758899647758427:-9065864600373678962
1990 [main] INFO  o.a.s.s.a.AuthUtils - Got AutoCreds []
2006 [main] WARN  o.a.s.u.NimbusClient - Using deprecated config nimbus.host for backward compatibility. Please update your storm.yaml so it only has config nimbus.seeds
2387 [main] WARN  o.a.s.u.NimbusClient - Using deprecated config nimbus.host for backward compatibility. Please update your storm.yaml so it only has config nimbus.seeds
2442 [main] INFO  o.a.s.StormSubmitter - Uploading topology jar /home/kinjal/Downloads/apache-storm-1.0.2/examples/storm-starter/storm-starter-topologies-1.0.2.jar to assigned location: /home/kinjal/Downloads/apache-storm-1.0.2/data/nimbus/inbox/stormjar-152dd57f-02db-4bd3-889f-3cb304919295.jar
Start uploading file '/home/kinjal/Downloads/apache-storm-1.0.2/examples/storm-starter/storm-starter-topologies-1.0.2.jar' to '/home/kinjal/Downloads/apache-storm-1.0.2/data/nimbus/inbox/stormjar-152dd57f-02db-4bd3-889f-3cb304919295.jar' (73405095 bytes)
[==================================================] 73405095 / 73405095
File '/home/kinjal/Downloads/apache-storm-1.0.2/examples/storm-starter/storm-starter-topologies-1.0.2.jar' uploaded to '/home/kinjal/Downloads/apache-storm-1.0.2/data/nimbus/inbox/stormjar-152dd57f-02db-4bd3-889f-3cb304919295.jar' (73405095 bytes)
5001 [main] INFO  o.a.s.StormSubmitter - Successfully uploaded topology jar to assigned location: /home/kinjal/Downloads/apache-storm-1.0.2/data/nimbus/inbox/stormjar-152dd57f-02db-4bd3-889f-3cb304919295.jar
5001 [main] INFO  o.a.s.StormSubmitter - Submitting topology Word-Count-Topology in distributed mode with conf {"storm.zookeeper.topology.auth.scheme":"digest","storm.zookeeper.topology.auth.payload":"-8405758899647758427:-9065864600373678962","topology.workers":3,"topology.debug":true}
5002 [main] WARN  o.a.s.u.NimbusClient - Using deprecated config nimbus.host for backward compatibility. Please update your storm.yaml so it only has config nimbus.seeds
7005 [main] INFO  o.a.s.StormSubmitter - Finished submitting topology: Word-Count-Topology
```

20. Output is seen on console as above
21. Can see detailed topology on UI as below



22. Finally to stop storm simple type ctrl+c on each terminal

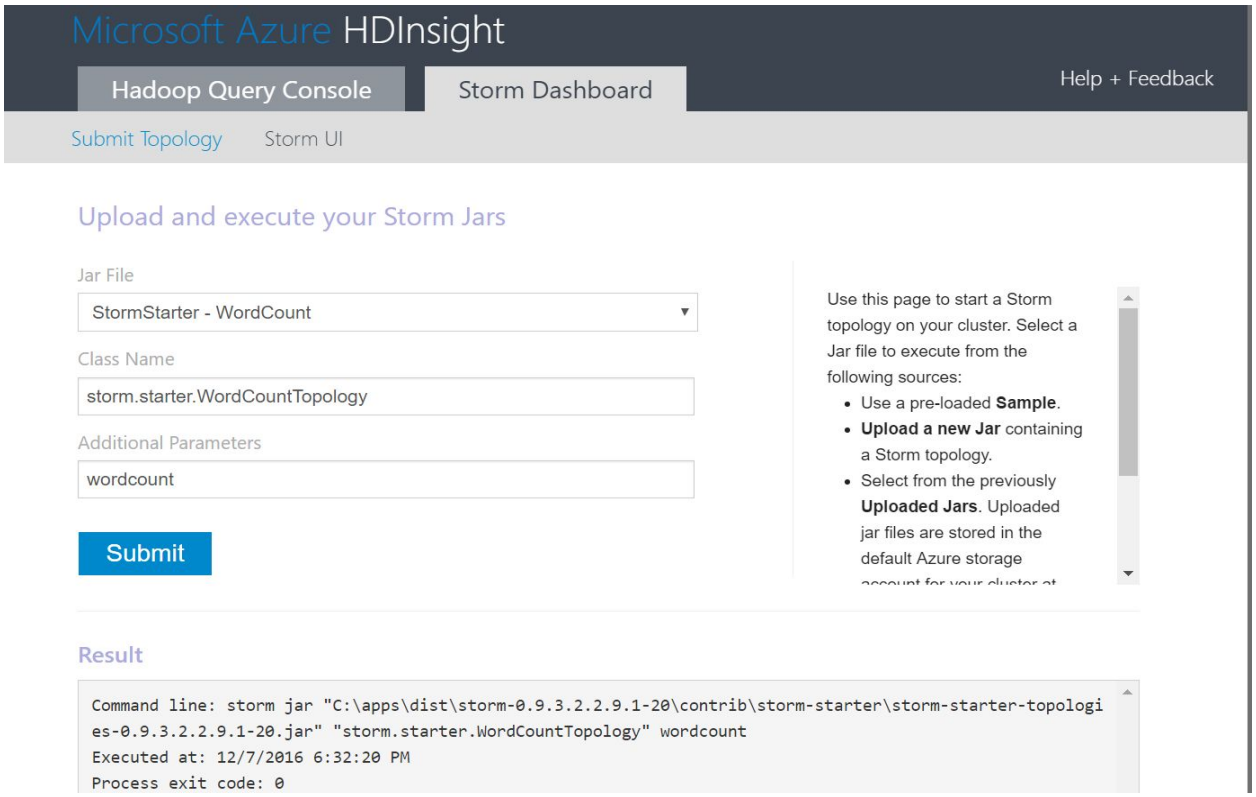
23. And stop apache zookeeper as below

```
kinjal@dell:~/Downloads/zookeeper-3.4.9/bin$ zkServer.sh stop
ZooKeeper JMX enabled by default
Using config: /home/kinjal/Downloads/zookeeper-3.4.9/bin/../conf/zoo.cfg
Stopping zookeeper ... STOPPED
```

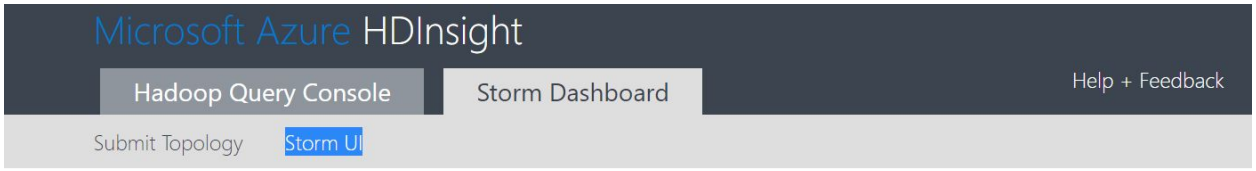
Implementation on Azure

Link for steps to follow for creating storm cluster on azure - https://docs.microsoft.com/en-us/azure/hdinsight/hdinsight-apache-storm-tutorial-get-started

Storm UI



Storm UI Results



Storm UI

Cluster Summary

Version	Nimbus uptime	Supervisors	Used slots	Free slots	Total slots	Executors	Tasks
0.9.3.2.2.9.1-20	14m 26s	1	3	1	4	29	29

Topology summary

Name	Id	Owner	Status	Uptime	Num workers	Num executors	Num tasks	Scheduler Info
wordcount	wordcount-1-1481153546		ACTIVE	1m 59s	3	29	29	

Supervisor summary

Id	Host	Uptime	Slots	Used slots
25c4c34c-967f-46c8-b3d2-52e387eb88b7	workernode0.StormClusterTeam11.b7.internal.cloudapp.net	15m 31s	4	3

Nimbus Configuration

Key	Value
dev.zookeeper.path	/tmp/dev-storm-zookeeper
drpc.authorizer.acl.filename	drpc-auth-acl.yaml
drpc.authorizer.acl.strict	false
drpc.childopts	-Xmx768m
drpc.http.creds.plugin	backtype.storm.security.auth.DefaultHttpCredentialsPlugin
drpc.http.port	3774
drpc.https.keystore.password	
drpc.https.keystore.type	JKS
drpc.https.port	-1
drpc.invocations.port	3773

Deactivate Topology

Topology summary

Name	Id	Owner	Status	Uptime	Num workers	Num executors	Num tasks	Scheduler Info
wordcount	wordcount-1-1481153546		INACTIVE	10m 1s	3	29		29

Topology actions

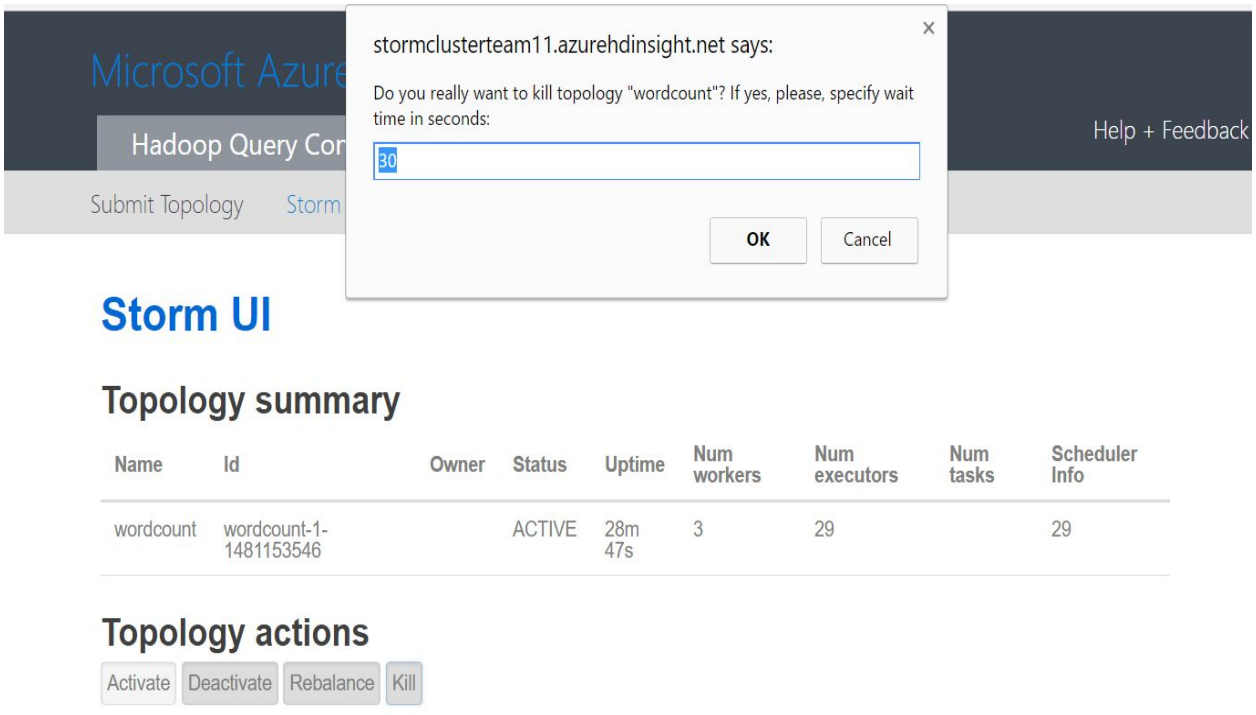
Activate | Deactivate | Rebalance | Kill

Topology stats

Window	Emitted	Transferred	Complete latency (ms)	Acked	Failed
10m 0s	405540	217440	0.000	217700	0
3h 0m 0s	405540	217440	0.000	217700	0
1d 0h 0m 0s	405540	217440	0.000	217700	0
All time	405540	217440	0.000	217700	0

Status changes from Active to Inactive on clicking deactivate

Kill topology



Submit Topology Storm UI

Storm UI

Topology summary

Name	Id	Owner	Status	Uptime	Num workers	Num executors	Num tasks	Scheduler Info
wordcount	wordcount-1-1481153546		KILLED	29m 27s	3	29		29

Topology actions

Activate Deactivate Rebalance Kill

Status changes to killed

Spout and bolt Stat info

Spouts (All time)

Id	Executors	Tasks	Emitted	Transferred	Complete latency (ms)	Acked	Failed	Error Host	Error Port	Last error
spout	5	5	51320	51320	0.000	0	0			

Bolts (All time)

Id	Executors	Tasks	Emitted	Transferred	Capacity (last 10m)	Execute latency (ms)	Executed	Process latency (ms)	Acked	Failed	E H
count	12	12	329340	0	0.009	0.199	329280	0.107	329320	0	
split	8	8	329220	329220	0.000	0.005	51440	2.703	51400	0	

Topology Visualization

Show Visualization

Topology Configuration

Key	Value
dev.zookeeper.path	/tmp/dev-storm-zookeeper
drpc.authorizer.acl.filename	drpc-auth-acl.yaml
drpc.authorizer.acl.strict	false

Input stats (All time)

Component	Stream	Execute latency (ms)	Executed	Process latency (ms)	Acked	Failed
split	default	0.199	351240	0.199	351200	0

Output stats (All time)

Stream	Emitted	Transferred
default	351220	0

Executors

Id	Uptime	Host	Port	Emitted	Transferred	Capacity (last 10m)	Exe late (ms
[5-5]}	24m 9s	workernode0.StormClusterTeam11.b7.internal.cloudapp.net	6702	54940	0	0.007	0.1
[6-6]}	24m 9s	workernode0.StormClusterTeam11.b7.internal.cloudapp.net	6701	33260	0	0.004	0.1
[7-7]}	24m 10s	workernode0.StormClusterTeam11.b7.internal.cloudapp.net	6703	22040	0	0.004	0.2
[8-	24m 9s	workernode0.StormClusterTeam11.b7.internal.cloudapp.net	6702	11040	0	0.001	0.1

Gives the count of every word in the sentence till the topology keeps running

Download Full Log

```
lt, id: {}, [cow]
2016-12-07 23:57:47 b.s.d.task [INFO] Emitting: count default [jumped, 11481]
2016-12-07 23:57:47 b.s.d.executor [INFO] Execute done TUPLE source: split:22, stream: default, id: {}, [cow]
2016-12-07 23:57:47 b.s.d.executor [INFO] BOLT ack TASK: 5 TIME:  TUPLE: source: split:22, stream: default, i
2016-12-07 23:57:47 b.s.d.executor [INFO] Execute done TUPLE source: split:22, stream: default, id: {}, [jump
2016-12-07 23:57:47 b.s.d.task [INFO] Emitting: spout default [the cow jumped over the moon]
2016-12-07 23:57:47 b.s.d.executor [INFO] TRANSFERING tuple TASK: 18 TUPLE: source: spout:29, stream: default
2016-12-07 23:57:47 b.s.d.executor [INFO] Processing received message FOR 11 TUPLE: source: split:18, stream:
2016-12-07 23:57:47 b.s.d.task [INFO] Emitting: count default [cow, 11482]
2016-12-07 23:57:47 b.s.d.executor [INFO] BOLT ack TASK: 11 TIME:  TUPLE: source: split:18, stream: default,
2016-12-07 23:57:47 b.s.d.executor [INFO] Processing received message FOR 5 TUPLE: source: split:18, stream:
2016-12-07 23:57:47 b.s.d.executor [INFO] Execute done TUPLE source: split:18, stream: default, id: {}, [cow]
2016-12-07 23:57:47 b.s.d.task [INFO] Emitting: count default [jumped, 11482]
2016-12-07 23:57:47 b.s.d.executor [INFO] BOLT ack TASK: 5 TIME:  TUPLE: source: split:18, stream: default, i
2016-12-07 23:57:47 b.s.d.executor [INFO] Execute done TUPLE source: split:18, stream: default, id: {}, [jump
2016-12-07 23:57:47 b.s.d.executor [INFO] Processing received message FOR 20 TUPLE: source: spout:27, stream:
2016-12-07 23:57:47 b.s.d.executor [INFO] Execute done TUPLE source: spout:27, stream: default, id: {}, [an a
2016-12-07 23:57:47 b.s.d.task [INFO] Emitting: split default ["an"]
2016-12-07 23:57:47 b.s.d.executor [INFO] TRANSFERING tuple TASK: 9 TUPLE: source: split:20, stream: default,
2016-12-07 23:57:47 b.s.d.task [INFO] Emitting: split default ["apple"]
2016-12-07 23:57:47 b.s.d.executor [INFO] TRANSFERING tuple TASK: 14 TUPLE: source: split:20, stream: default
2016-12-07 23:57:47 b.s.d.executor [INFO] Processing received message FOR 14 TUPLE: source: split:20, stream:
```

Storm Commands

- Storm jar
- Storm kill
- Storm activate
- Strom deactivate
- Storm nimbus
- Storm supervisor
- Storm ui
- Storm help
- Storm logviewer

Features of Storm

- Fast
- Support for multiple programming language - R, Python, Java and Scala
- Scalable, Reliable and Fault tolerant
- Easy to setup and operate
- Guaranteed Data Processing - At least once
- Low Latency

Application of Storm

Groupon: At Groupon we use Storm to build real-time data integration systems. Storm helps us analyze, clean, normalize, and resolve large amounts of non-unique data points with low latency and high throughput.

Twitter: Storm powers a wide variety of Twitter systems, ranging in applications from discovery, realtime analytics, personalization, search, revenue optimization, and many more. Storm integrates with the rest of Twitter's infrastructure, including database systems (Cassandra, Memcached, etc), the messaging infrastructure, Mesos, and the monitoring/alerting systems. Storm's isolation scheduler makes it easy to use the same cluster both for production applications and in-development applications, and it provides a sane way to do capacity planning.

Yahoo!: Yahoo! is developing a next generation platform that enables the convergence of big-data and low-latency processing. While Hadoop is our primary technology for batch processing, Storm empowers stream/micro-batch processing of user events, content feeds, and application logs.

Useful Resources

- http://storm.apache.org/
- https://www.tutorialspoint.com/apache_storm/index.htm
- http://storm.apache.org/releases/current/Setting-up-a-Storm-cluster.html
- https://github.com/apache/storm/tree/master/examples

Apache Flink

- Open source Big Data platform.
- Distributed data flow engine.
- Handles both batch and stream processing jobs.
- A framework that is fully compatible to Hadoop.
- It can be easily integrated with other open source systems for data input/output as well as deployment.

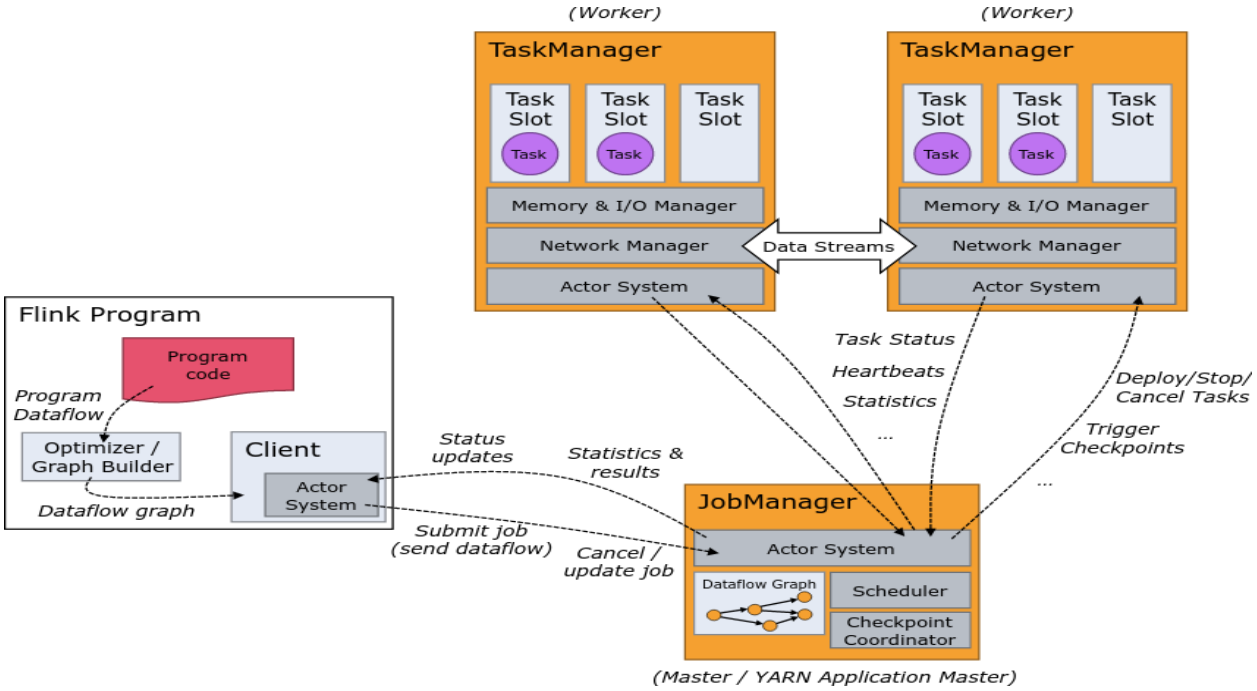
Basic Concepts

The basic building blocks of Flink programs are streams and transformations (note that a DataSet is internally also a stream). A stream is an intermediate result, and a transformation is an operation that takes one or more streams as input, and computes one or more result streams from them.

When executed, Flink programs are mapped to streaming dataflows, consisting of streams and transformation operators. Each dataflow starts with one or more sources and ends in one or more sinks. The dataflows may resemble arbitrary directed acyclic graphs (DAGs). (Special forms of cycles are permitted via iteration constructs, we omit this here for simplicity).

In most cases, there is a one-to-one correspondence between the transformations in the programs and the operators in the dataflow. Sometimes, however, one transformation may consist of multiple transformation operators.

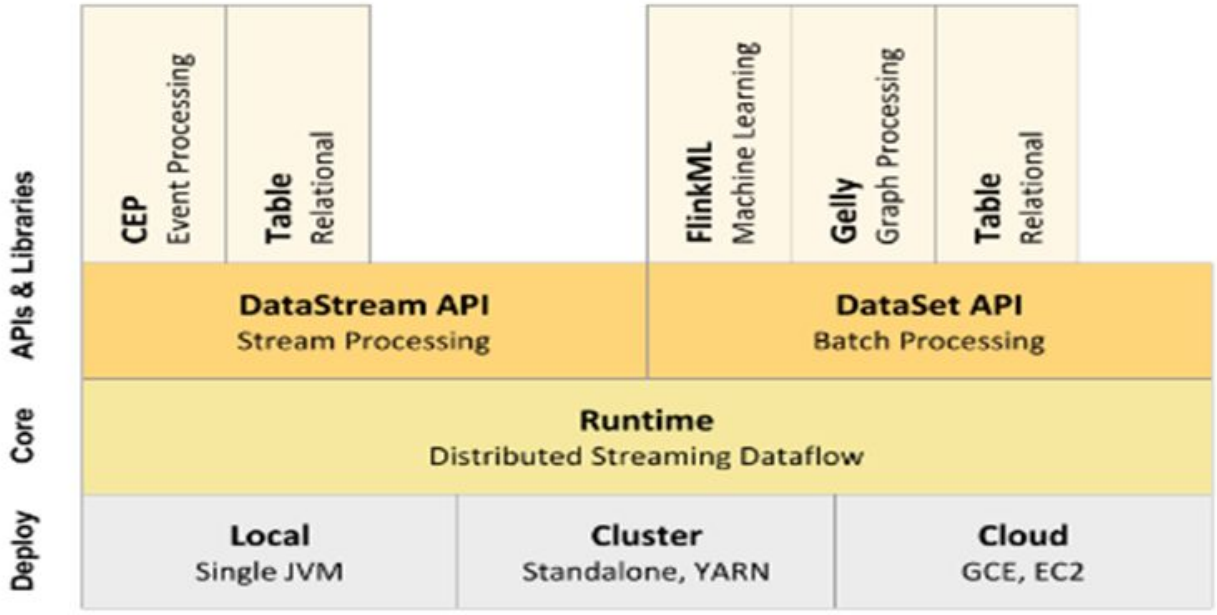
Architecture



When the Flink system is started, it bring up the JobManager and one or more TaskManagers. The JobManager is the coordinator of the Flink system, while the TaskManagers are the worksers that execute parts of the parallel programs. When starting the system in local mode, a single JobManager and TaskManager are brought up within the same JVM.

When a program is submitted, a client is created that performs the pre-processing and turns the program into the parallel data flow form that is executed by the JobManager and TaskManagers.

Flink Library Ecosystem



Steps to install and run Apache Flink

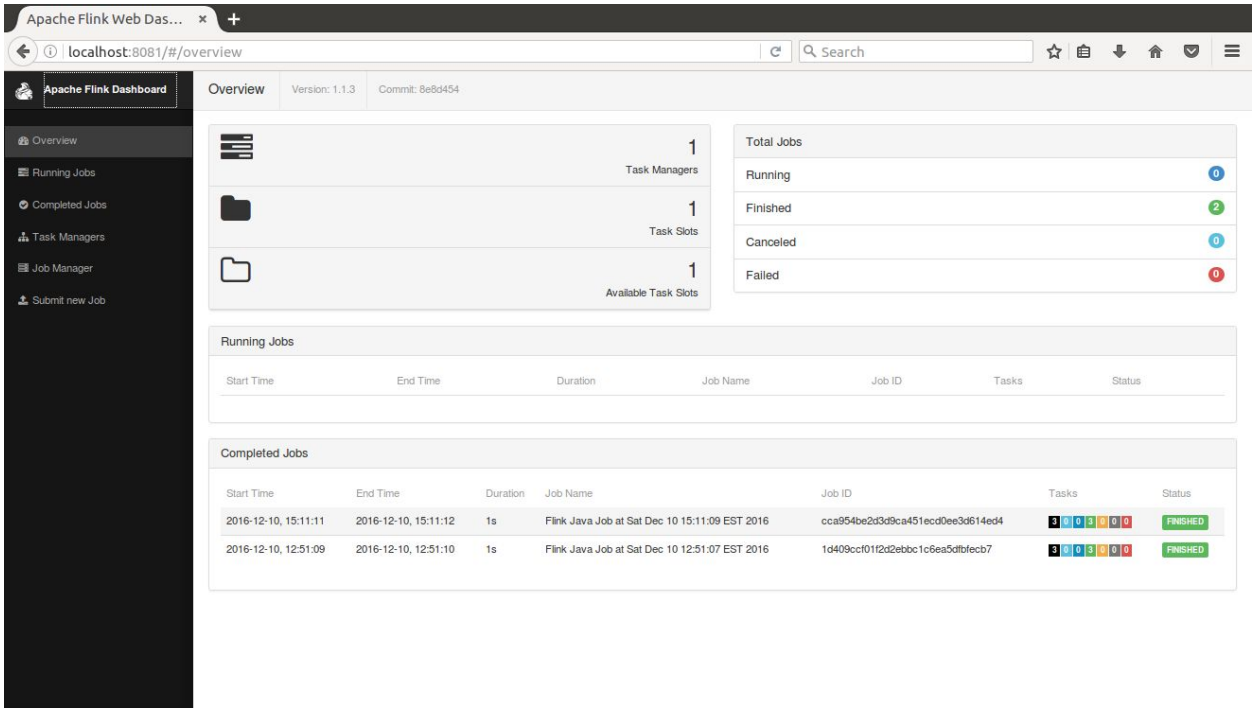
1. Download and extract apache flink from link. This version supports hadoop 1 and sacla 2.10
2. It already has the setup done, no need for addition set up.

```
kinjal@dell:~/Downloads/flink-1.1.3$ bin/start-local.sh
```

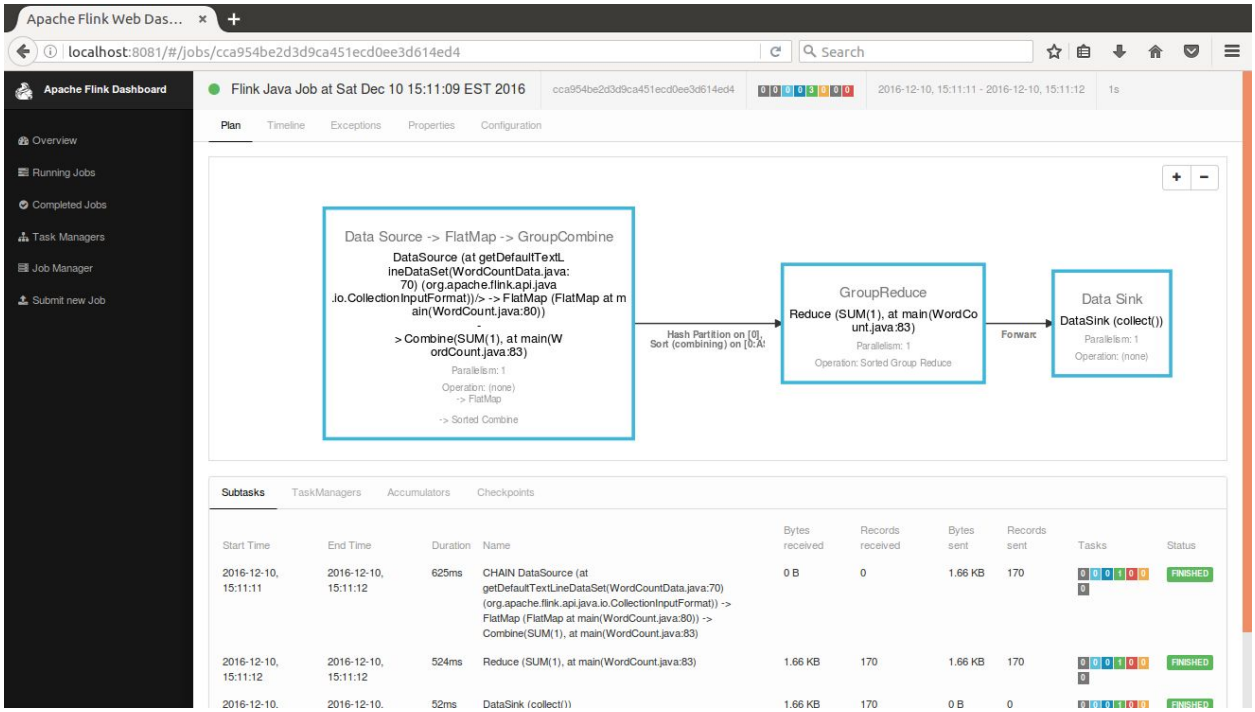
3. To run a word count job on flink do as below

```
kinjal@dell:~/Downloads/flink-1.1.3$ ./bin/flink run ./examples/batch/WordCount.jar
Cluster configuration: Standalone cluster with JobManager at /127.0.0.1:6123
Using address 127.0.0.1:6123 to connect to JobManager.
JobManager web interface address http://127.0.0.1:8081
Starting execution of program
Executing WordCount example with default input data set.
Use --input to specify file input.
Printing result to stdout. Use --output to specify output path.
Submitting job with JobID: cca954be2d3d9ca451ecd0ee3d614ed4. Waiting for job completion.
Connected to JobManager at Actor[akka.tcp://flink@127.0.0.1:6123/user/jobmanager#-2062452274]
12/10/2016 15:11:11     Job execution switched to status RUNNING.
12/10/2016 15:11:11     CHAIN DataSource (at getDefaultTextLineDataSet(WordCountData.java:70) (org.apache.flink.api.java.io.CollectionInputFormat
)) -> FlatMap (FlatMap at main(WordCount.java:80)) -> Combine(SUM(1), at main(WordCount.java:83)(1/1) switched to SCHEDULED
12/10/2016 15:11:11     CHAIN DataSource (at getDefaultTextLineDataSet(WordCountData.java:70) (org.apache.flink.api.java.io.CollectionInputFormat
)) -> FlatMap (FlatMap at main(WordCount.java:80)) -> Combine(SUM(1), at main(WordCount.java:83)(1/1) switched to DEPLOYING
12/10/2016 15:11:11     CHAIN DataSource (at getDefaultTextLineDataSet(WordCountData.java:70) (org.apache.flink.api.java.io.CollectionInputFormat
)) -> FlatMap (FlatMap at main(WordCount.java:80)) -> Combine(SUM(1), at main(WordCount.java:83)(1/1) switched to RUNNING
12/10/2016 15:11:12     Reduce (SUM(1), at main(WordCount.java:83)(1/1) switched to SCHEDULED
12/10/2016 15:11:12     Reduce (SUM(1), at main(WordCount.java:83)(1/1) switched to DEPLOYING
12/10/2016 15:11:12     CHAIN DataSource (at getDefaultTextLineDataSet(WordCountData.java:70) (org.apache.flink.api.java.io.CollectionInputFormat
)) -> FlatMap (FlatMap at main(WordCount.java:80)) -> Combine(SUM(1), at main(WordCount.java:83)(1/1) switched to FINISHED
12/10/2016 15:11:12     Reduce (SUM(1), at main(WordCount.java:83)(1/1) switched to RUNNING
12/10/2016 15:11:12     DataSink (collect())(1/1) switched to SCHEDULED
12/10/2016 15:11:12     DataSink (collect())(1/1) switched to DEPLOYING
12/10/2016 15:11:12     Reduce (SUM(1), at main(WordCount.java:83)(1/1) switched to FINISHED
12/10/2016 15:11:12     DataSink (collect())(1/1) switched to RUNNING
12/10/2016 15:11:12     DataSink (collect())(1/1) switched to FINISHED
(a,5)
(action,1)
(after,1)
(against,1)
(all,2)
(and,12)
(arms,1)
(arrows,1)
(awry,1)
(ay,1)
(bare,1)
(be,4)
(bear,3)
(bodkin,1)
(bourn,1)
(but,1)
```

4. You can view the results or ongoing jobs at localhost:8080 as below



And on clicking the job you can view in depth detail as below



Quick Setup

Requirements

Flink runs on all UNIX-like environments: Linux, Mac OS X, Cygwin. The only requirement is to have a working Java 6.x(or higher) installation.

Download

Download the ready to run binary package. Choose the Flink distribution that matches your Hadoop version. If you are unsure which version to choose or you just want to run locally, pick the package for Hadoop 1.2.

Start

1. Go to the download directory.
2. Unpack the downloaded archive.
3. Start Flink.

```
$ cd ~/Downloads        # Go to download directory
$ tar xzf flink-*.tgz   # Unpack the downloaded archive
$ cd flink-0.8.1
$ bin/start-local.sh    # Start Flink
```

Check the JobManager's web frontend at http://localhost:8081 and make sure everything is up and running.

Run Example

Run the Word Count example to see Flink at work.
Download test data:

```
$ wget -O hamlet.txt http://www.gutenberg.org/cache/epub/1787/pg1787.txt
```

You now have a text file called hamlet.txt in your working directory.
Start the example program:

```
$ bin/flink run ./examples/flink-java-examples-0.8.1-WordCount.jar file://`pwd`/hamlet.txt file://`pwd`/w
ordcount-result.txt
```

You will find a file called wordcount-result.txt in your current directory.

Cluster Setup

Running Flink on a cluster is as easy as running it locally. Having passwordless SSH and the same directory structure on all your cluster nodes lets you use our scripts to control everything.

1. Copy the unpacked flink directory from the downloaded archive to the same file system path on each node of your setup.
2. Choose a master node (JobManager) and set the jobmanager.rpc.address key in conf/flink-conf.yaml to its IP or hostname. Make sure that all nodes in your cluster have the same jobmanager.rpc.address configured.
3. Add the IPs or hostnames (one per line) of all worker nodes (TaskManager) to the slaves files in conf/slaves.

You can now start the cluster at your master node with bin/start-cluster.sh
For more detailed instructions, check out the programming Guides and examples.

Features of Flink

- High Performance & Low Latency
- Fault-tolerance via Lightweight Distributed Snapshots

- One Runtime for Streaming and Batch Processing
- Memory Management
- Program Optimizer
- Flexible Streaming Windows

Application of Flink

Alibaba.com : The world's largest retailer, built a Flink-based system (Blink) to optimize search rankings in real time.

Bouygues Telecom: Uses Flink for real-time event processing over billions of Kafka messages per day.

Capital One: A Fortune 500 financial services company, uses Flink to monitor real-time customer activity.

Ericsson: Used Flink to build a real-time anomaly detection over large infrastructures using machine learning.

Useful Resources

- http://flink.apache.org/
- https://github.com/apache/flink/tree/master/flink-examples
- https://cwiki.apache.org/confluence/display/FLINK/Apache+Flink+Home
- https://twitter.com/apacheflink

Storm v/s Flink

Apache Storm	Apache Flink
Origins from Twitter	Origins from German Research Foundation
Supports only real time stream processing	Supports real time and batch processing
Supports any language	Supports only Java and Scala
Message delivery guarantee ‘At least once’	Message delivery guarantee ‘Exactly once’