Apache Storm

tutorialspoint

SIMPLYEASYLEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

Storm was originally created by **Nathan Marz** and team at **BackType**. BackType is a social analytics company. Later, Storm was acquired and open-sourced by **Twitter**. In a short time, Apache Storm became a standard for distributed real-time processing system that allows you to process large amount of data, similar to Hadoop. Apache Storm is written in Java and Clojure. It is continuing to be a leader in real-time analytics.

This tutorial will explore the principles of Apache Storm, distributed messaging, installation, creating Storm topologies and deploy them to a Storm cluster, workflow of Trident, real-time applications and finally concludes with some useful examples.

Audience

This tutorial has been prepared for professionals aspiring to make a career in Big Data Analytics using Apache Storm framework. This tutorial will give you enough understanding on creating and deploying a Storm cluster in a distributed environment.

Prerequisites

Before proceeding with this tutorial, you must have a good understanding of Core Java and any of the Linux flavors.

Copyright & Disclaimer

Table of Contents

About the Tutorial i

Audience i

Prerequisites i

Copyright & Disclaimer i

Table of Contents ii

1. APACHE STORM – INTRODUCTION 1

What is Apache Storm? 1

Apache Storm vs Hadoop 1

Use-Cases of Apache Storm 2

Apache Storm – Benefits 3

2. APACHE STORM – CORE CONCEPTS 4

Topology 5

Tasks 5

Workers 6

Stream Grouping 6

3. STORM – CLUSTER ARCHITECTURE 9

4. APACHE STORM – WORKFLOW 11

5. STORM – DISTRIBUTED MESSAGING SYSTEM 12

What is Distributed Messaging System? 12

Thrift Protocol 13

6. APACHE STORM – INSTALLATION 14

Step 1: Verifying Java Installation 14

Step 2: ZooKeeper Framework Installation 15

Step 3: Apache Storm Framework Installation 17

7. APACHE STORM – WORKING EXAMPLE 19

Scenario – Mobile Call Log Analyzer **19**

Spout Creation **19**

Bolt Creation **23**

Call log Creator Bolt **24**

Call log Counter Bolt **26**

Creating Topology **27**

Local Cluster **28**

Building and Running the Application **29**

Non-JVM languages **30**

8. APACHE STORM – TRIDENT 32

Trident Topology **32**

Trident Tuples **32**

Trident Spout **32**

Trident Operations **33**

State Maintenance **37**

Distributed RPC **37**

When to Use Trident? **37**

Working Example of Trident **37**

Building and Running the Application **41**

9. APACHE STORM IN TWITTER 43

Twitter **43**

Hashtag Reader Bolt **47**

Hashtag Counter Bolt **49**

Submitting a Topology **50**

Building and Running the Application **51**

10. APACHE STORM IN YAHOO! FINANCE 53

Spout Creation **53**

Bolt Creation **55**

Submitting a Topology **57**

Building and Running the Application **58**

11. APACHE STORM – APPLICATIONS 59

Klout **59**

The Weather Channel **59**

Telecom Industry **59**

Apache Storm – Introduction

What is Apache Storm?

Apache Storm is a distributed real-time big data-processing system. Storm is designed to process vast amount of data in a fault-tolerant and horizontal scalable method. It is a streaming data framework that has the capability of highest ingestion rates. Though Storm is stateless, it manages distributed environment and cluster state via Apache ZooKeeper. It is simple and you can execute all kinds of manipulations on real-time data in parallel.

Apache Storm is continuing to be a leader in real-time data analytics. Storm is easy to setup, operate and it guarantees that every message will be processed through the topology at least once.

Apache Storm vs Hadoop

Basically Hadoop and Storm frameworks are used for analyzing big data. Both of them complement each other and differ in some aspects. Apache Storm does all the operations except persistency, while Hadoop is good at everything but lags in real-time computation. The following table compares the attributes of Storm and Hadoop.

Storm	Hadoop
Real-time stream processing	Batch processing
Stateless	Stateful
Master/Slave architecture with ZooKeeper based coordination. The master node is called as **nimbus** and slaves are **supervisors**.	Master-slave architecture with/without ZooKeeper based coordination. Master node is **job tracker** and slave node is **task tracker**.
A Storm streaming process can access tens of thousands messages per second on cluster.	Hadoop Distributed File System (HDFS) uses MapReduce framework to process vast amount of data that takes minutes or hours.
Storm topology runs until shutdown by the user or an unexpected unrecoverable failure.	MapReduce jobs are executed in a sequential order and completed eventually.
Both are distributed and fault-tolerant	

If nimbus / supervisor dies, restarting makes it continue from where it stopped, hence nothing gets affected.	If the JobTracker dies, all the running jobs are lost.

Use-Cases of Apache Storm

Apache Storm is very famous for real-time big data stream processing. For this reason, most of the companies are using Storm as an integral part of their system. Some notable examples are as follows:

Twitter – Twitter is using Apache Storm for its range of "Publisher Analytics products". "Publisher Analytics Products" process each and every tweets and clicks in the Twitter Platform. Apache Storm is deeply integrated with Twitter infrastructure.

NaviSite – NaviSite is using Storm for Event log monitoring/auditing system. Every logs generated in the system will go through the Storm. Storm will check the message against the configured set of regular expression and if there is a match, then that particular message will be saved to the database.

Wego – Wego is a travel metasearch engine located in Singapore. Travel related data comes from many sources all over the world with different timing. Storm helps Wego to search real-time data, resolves concurrency issues and find the best match for the end-user.

Apache Storm – Benefits

Here is a list of the benefits that Apache Storm offers:

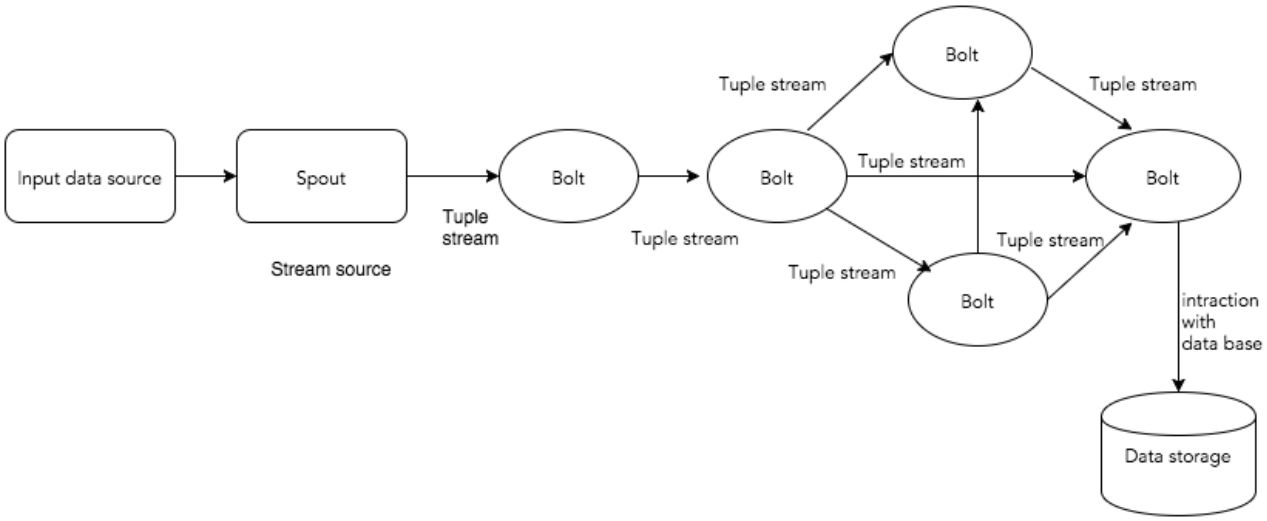
- Storm is open source, robust, and user friendly. It could be utilized in small companies as well as large corporations.
- Storm is fault tolerant, flexible, reliable, and supports any programming language.
- Allows real-time stream processing.
- Storm is unbelievably fast because it has enormous power of processing the data.
- Storm can keep up the performance even under increasing load by adding resources linearly. It is highly scalable.
- Storm performs data refresh and end-to-end delivery response in seconds or minutes depends upon the problem. It has very low latency.
- Storm has operational intelligence.

- Storm provides guaranteed data processing even if any of the connected nodes in the cluster die or messages are lost.

Apache Storm – Core Concepts

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.

The following diagram depicts the core concept of Apache Storm.

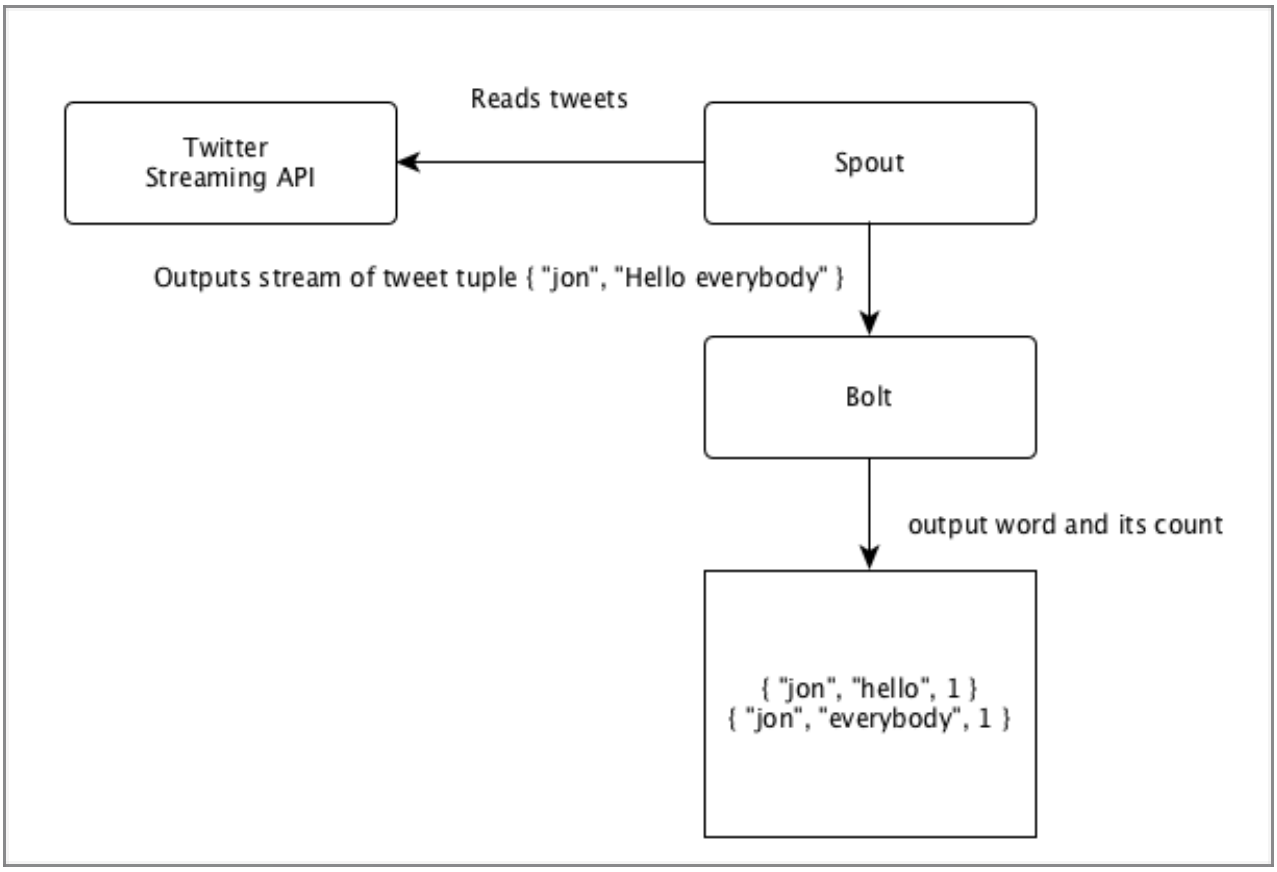


Let us now have a closer look at the components of Apache Storm:

Components	Description
Tuple	Tuple is the main data structure in Storm. It is a list of ordered elements. By default, a Tuple supports all data types. Generally, it is modelled as a set of comma separated values and passed to a Storm cluster.
Stream	Stream is an unordered sequence of tuples.
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 datasources. "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.

Let's take a real-time example of "Twitter Analysis" and see how it can be modelled in Apache Storm. The following diagram depicts the structure.



The input for the "Twitter Analysis" comes from Twitter Streaming API. Spout will read the tweets of the users using Twitter Streaming API and output as a stream of tuples. A single tuple from the spout will have a twitter username and a single tweet as comma separated values. Then, this steam of tuples will be forwarded to the Bolt and the Bolt will split the tweet into individual word, calculate the word count, and persist the information to a configured datasource. Now, we can easily get the result by querying the datasource.

Topology

Spouts and bolts are connected together and they form a topology. Real-time application logic is specified inside Storm topology. In simple words, a topology is a directed graph where vertices are computation and edges are stream of data.

A simple topology starts with spouts. Spout emits the data to one or more bolts. Bolt represents a node in the topology having the smallest processing logic and the output of a bolt can be emitted into another bolt as input.

Storm keeps the topology always running, until you kill the topology. Apache Storm's main job is to run the topology and will run any number of topology at a given time.

Tasks

Now you have a basic idea on spouts and bolts. They are the smallest logical unit of the topology and a topology is built using a single spout and an array of bolts. They should be executed properly in a particular order for the topology to run successfully. The execution of each and every spout and bolt by Storm is called as "Tasks". In simple words, a task is either the execution of a spout or a bolt. At a given time, each spout and bolt can have multiple instances running in multiple separate threads.

Workers

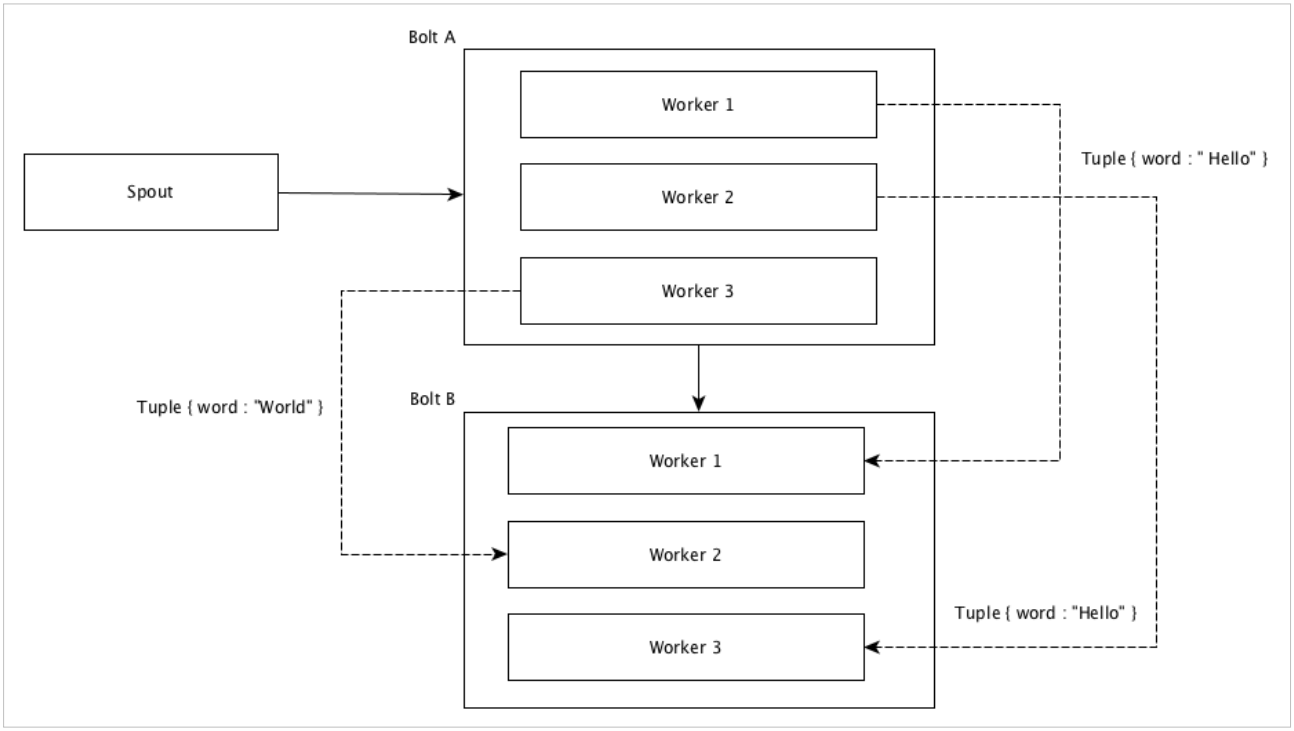
A topology runs in a distributed manner, on multiple worker nodes. Storm spreads the tasks evenly on all the worker nodes. The worker node's role is to listen for jobs and start or stop the processes whenever a new job arrives.

Stream Grouping

Stream of data flows from spouts to bolts or from one bolt to another bolt. Stream grouping controls how the tuples are routed in the topology and helps us to understand the tuples flow in the topology. There are four in-built groupings as explained below.

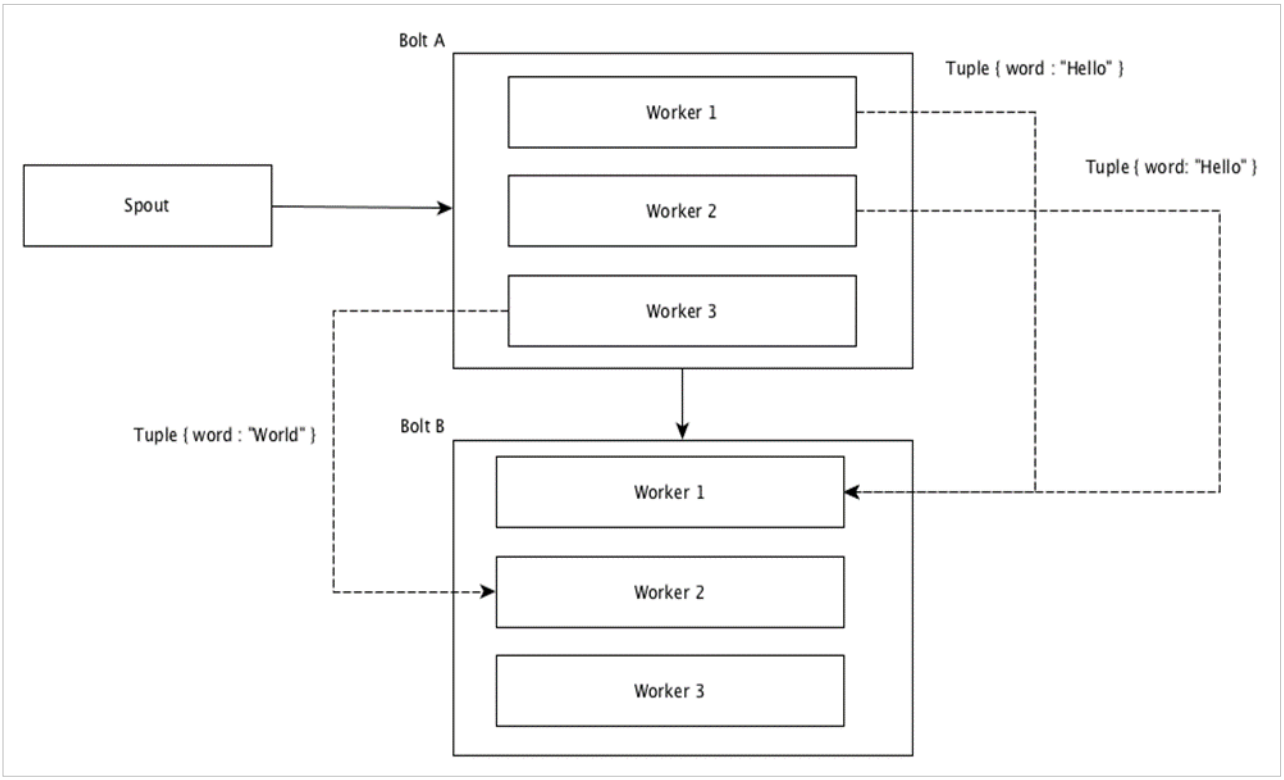
Shuffle Grouping

In shuffle grouping, an equal number of tuples is distributed randomly across all of the workers executing the bolts. The following diagram depicts the structure.



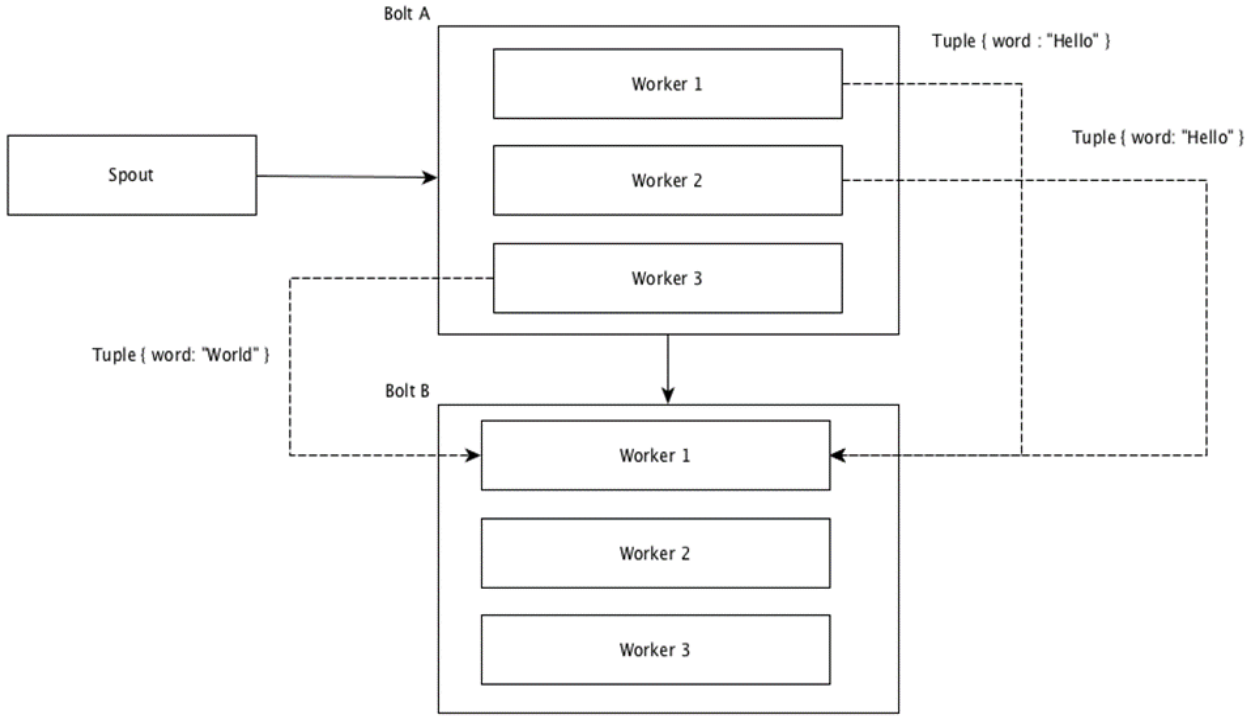
Field Grouping

The fields with same values in tuples are grouped together and the remaining tuples kept outside. Then, the tuples with the same field values are sent forward to the same worker executing the bolts. For example, if the stream is grouped by the field "word", then the tuples with the same string, "Hello" will move to the same worker. The following diagram shows how Field Grouping works.



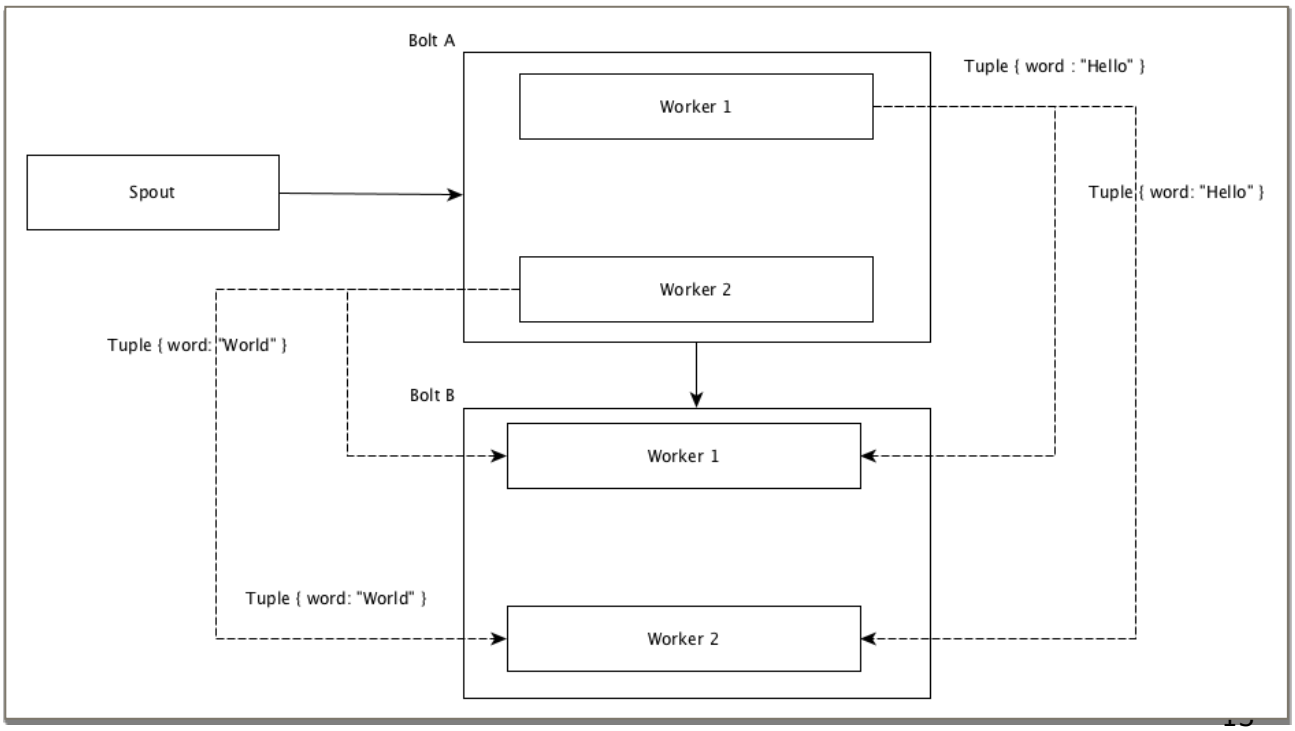
Global Grouping

All the streams can be grouped and forward to one bolt. This grouping sends tuples generated by all instances of the source to a single target instance (specifically, pick the worker with lowest ID).



All Grouping

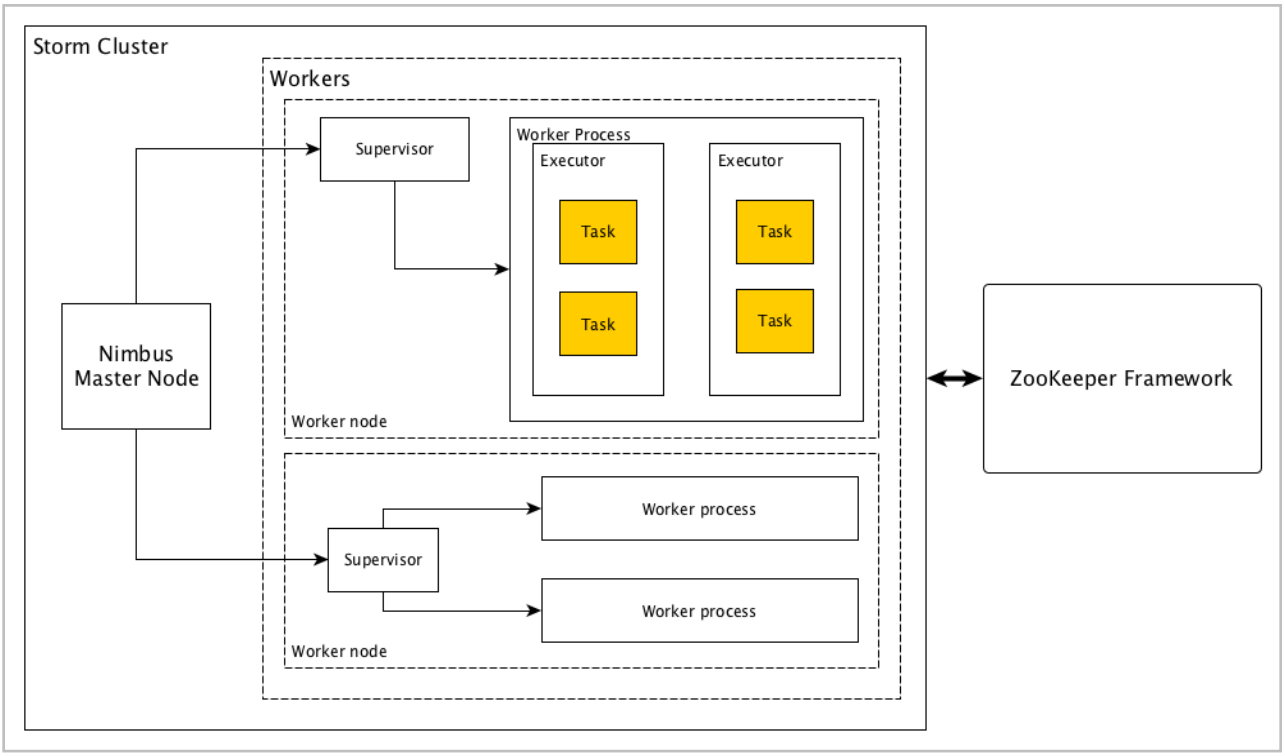
All Grouping sends a single copy of each tuple to all instances of the receiving bolt. This kind of grouping is used to send signals to bolts. All grouping is useful for join operations.



Storm – Cluster Architecture

One of the main highlight of the Apache Storm is that it is a fault-tolerant, fast with no "Single Point of Failure" (SPOF) distributed application. We can install Apache Storm in as many systems as needed to increase the capacity of the application.

Let's have a look at how the Apache Storm cluster is designed and its internal architecture. The following diagram depicts the cluster design.



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.

Components	Description

Nimbus	Nimbus is a master node of Storm cluster. All other nodes in the cluster are called as **worker nodes**. Master node is responsible for distributing data among all the worker nodes, assign tasks to worker nodes and monitoring failures.
Supervisor	The nodes that follow instructions given by the nimbus are called as **Supervisors**. A supervisor has multiple worker processes and it governs worker processes to complete the tasks assigned by the nimbus.
Worker process	A worker process will execute tasks related to a specific topology. A worker process will not run a task by itself, instead it creates **executors** and asks them to perform a particular task. A worker process will have multiple executors.
Executor	An executor is nothing but a single thread spawn by a worker process. An executor runs one or more tasks but only for a specific spout or bolt.
Task	A task performs actual data processing. So, it is either a spout or a bolt.
ZooKeeper framework	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. ZooKeeper helps the supervisor to interact with the nimbus. It is responsible to maintain the state of nimbus and supervisor.

Storm is stateless in nature. Even though stateless nature has its own disadvantages, it actually helps Storm to process real-time data in the best possible and quickest way.

Storm is *not entirely* stateless though. It stores its state in Apache ZooKeeper. Since the state is available in Apache ZooKeeper, a failed nimbus can be restarted and made to work from where it left. Usually, service monitoring tools like **monit** will monitor Nimbus and restart it if there is any failure.

Apache Storm also have an advanced topology called **Trident Topology** with state maintenance and it also provides a high-level API like Pig. We will discuss all these features in the coming chapters.

Apache Storm – Workflow

A working Storm cluster should have one nimbus and one or more supervisors. Another important node is Apache ZooKeeper, which will be used for the coordination between the nimbus and the supervisors.

Let us now take a close look at the workflow of Apache Storm:

- Initially, the nimbus will wait for the "Storm Topology" to be submitted to it. The
- 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.

By default, there are two modes in a Storm cluster:

- **Local mode**: This mode is used for development, testing, and debugging because it is the easiest way to see all the topology components working together. In this mode, we can adjust parameters that enable us to see how our topology runs in different Storm configuration environments. In Local mode, storm topologies run on the local machine in a single JVM.
- **Production mode**: In this mode, we submit our topology to the working storm cluster, which is composed of many processes, usually running on different machines.

As discussed in the workflow of storm, a working cluster will run indefinitely until it is shutdown.

End of ebook preview

If you liked what you saw…

Buy it from our store @ https://store.tutorialspoint.com