Cloudera Streaming Analytics 1.11.1

Storm Flink Migration

Date published: 2019-12-17
Date modified: 2024-04-16

https://docs.cloudera.com/

Legal Notice

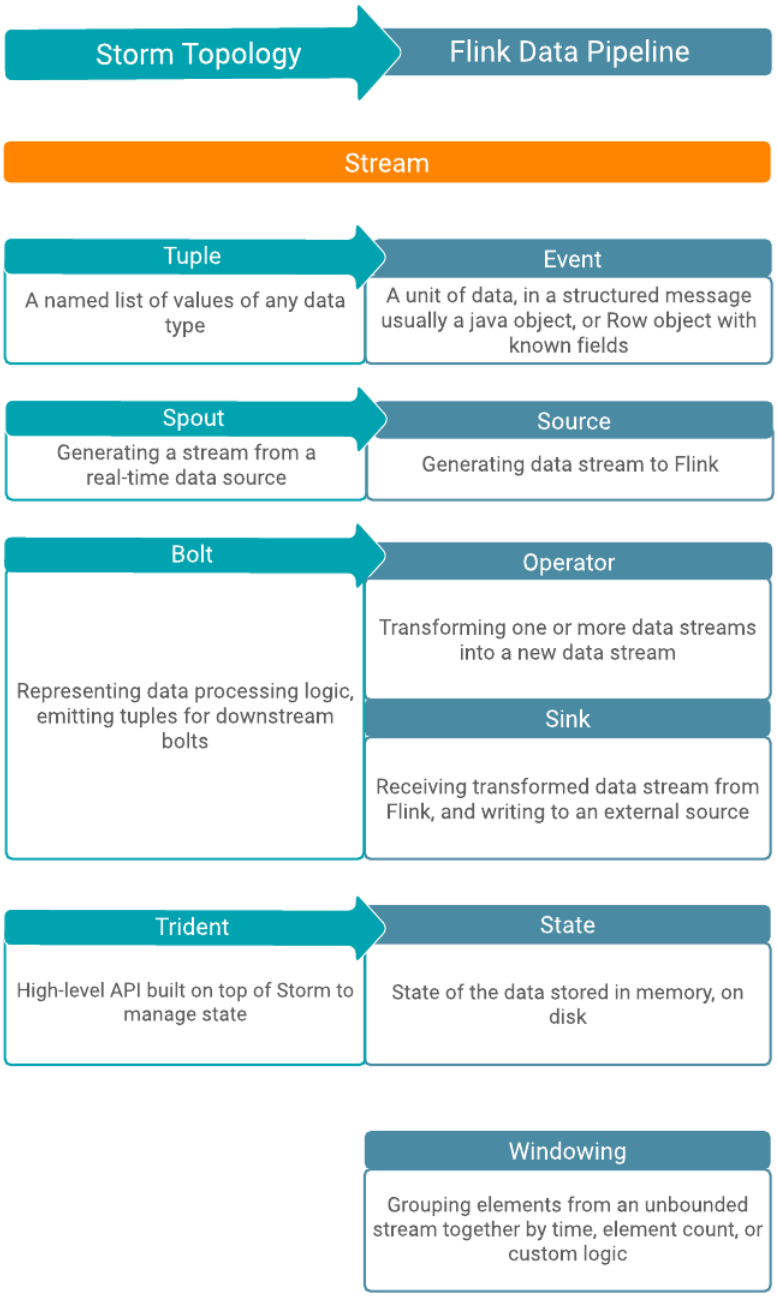
Contents

Comparing Storm and Flink

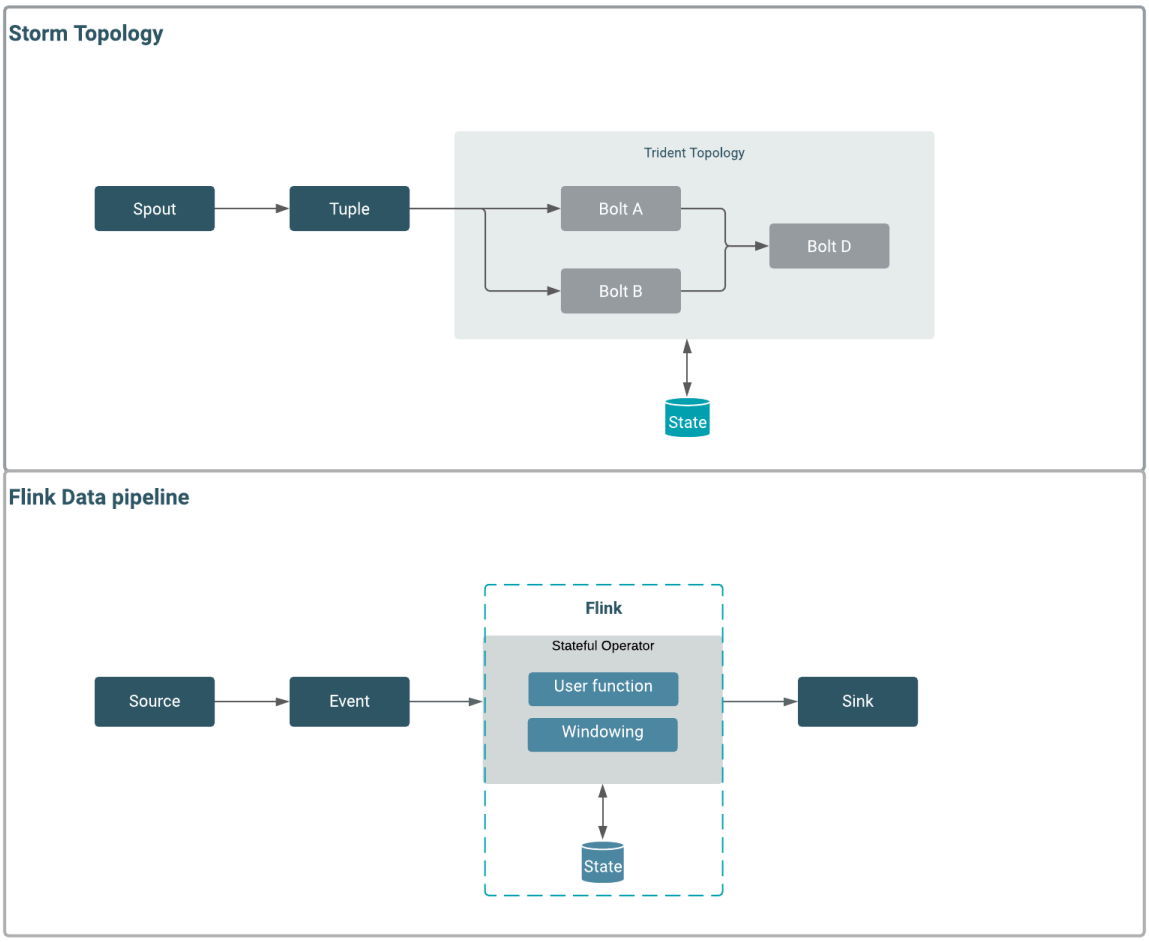
Before you start the migration process from Storm to Flink, you need to understand the differences and similarities between the two frameworks. You can create real-time data processing applications with both systems, but there are differences in concept, architecture, and data distribution. Understanding these differences can make the migration process easier.

Conceptual differences

Storm and Flink can process unbounded data streams in real-time with low latency. Storm uses tuples, spouts, and bolts that construct its stream processing topology. For Flink, you need sources, operators, and sinks to process events within its data pipeline. Other than the terminology, the two systems handle state differently. Furthermore, Flink has an event windowing function to achieve exatly-once processing.



In a Flink program, the incoming data from a source are transformed by a defined operation which results in one or more output streams. The transformation or computation on the data is completed by an operator where you can also add windowing function or join data streams. The main conceptual difference between Storm and Flink is state handling. While you need Trident API to manage state and fault tolerancy in Storm, Flink handles state in-memory and on disk, which makes the process of checkpointing and state management faster in Flink. This also makes the maintenance, troubleshooting, and upgrading processes easier in Flink, because the state of an application and the state of the different operators within the data pipeline are saved. The following illustration details how these concepts in Storm and Flink structure of their dataflow.

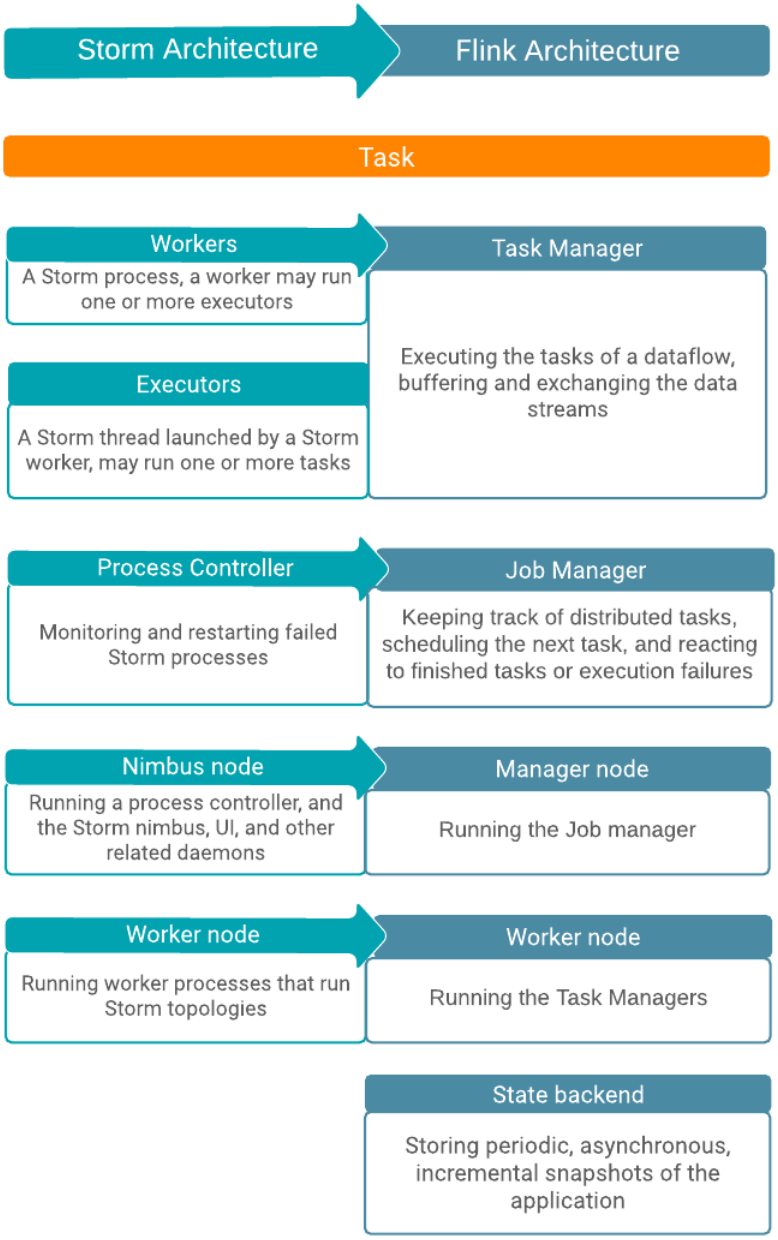


The following table shows the connectors supported for Storm and Flink. When choosing sources and sinks for Flink, you need to first determine the purpose and business logic of your application and then decide on the suitable connectors within the data pipeline.

Storm Connectors		Flink Connectors	
Spout	Kafka	Source	Kafka
	HDFS		HDFS
Bolt	Kafka	Sink	Kafka
	HBase		HBase
	Hive		Kudu
	HDFS		HDFS
			Hive

Differences in architecture

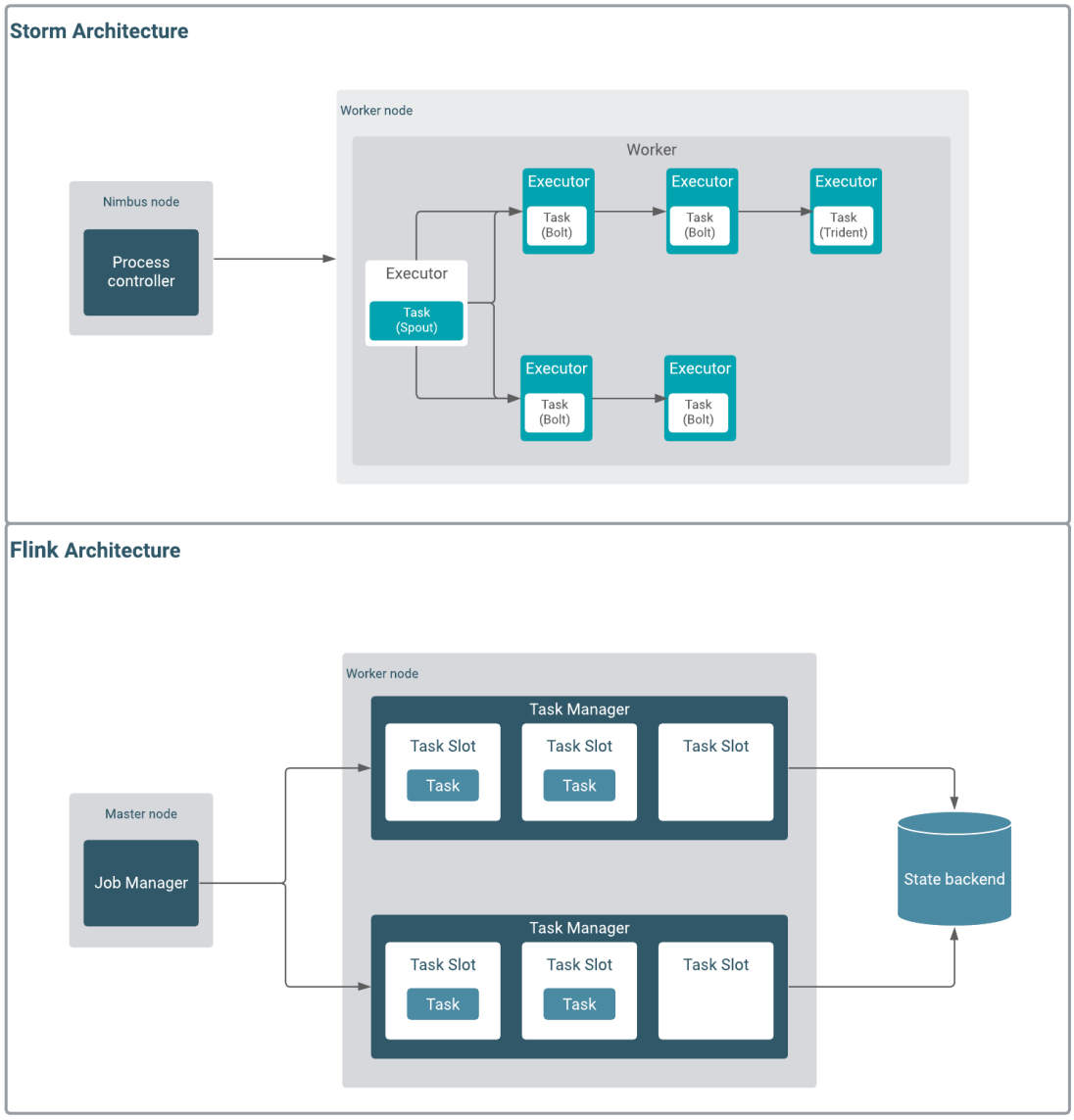
The basic architecture of task execution is similar in Storm and Flink. The main difference between the two systems is that Workers and Executors are responsible for executing the tasks in Storm, while in Flink the execution is done by only the Task Managers. The Task Managers also manage the state backend, which is a durable storage for storing states.



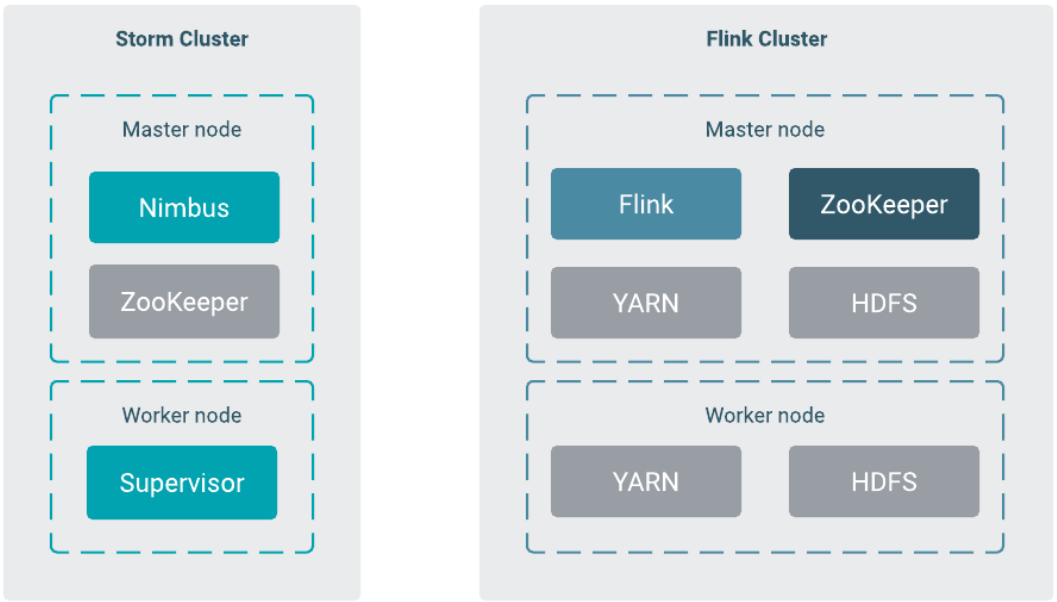
Flink has a simpler architecture compared to Storm as the Task Managers fulfill the jobs of Workers and Executors. The process of task execution is similar: a Process Controller/Job Manager on a master node starts a worker node. On a worker node the Workers and Executors in Storm, Task Managers in Flink are responsible for running the Tasks. As the Executor, the Task Manager can also run more than one tasks at the same time. The resource management for the tasks are completed by the Process Controller in Storm and by the Job manager in Flink.

In a Storm cluster, the Nimbus runs the Storm topology and distributes it to the Supervisor from which the processes are delegated to workers. ZooKeeper is needed to coordinate the communication between the Nimbus and Supervisor node. In a Flink cluster, Flink jobs are executed as YARN applications. HDFS is used to store recovery and log data, while ZooKeeper is used for high availability coordination for jobs. The following illustrations detail the architecture, task execution, and cluster layout in Storm and Flink.

For Architecture

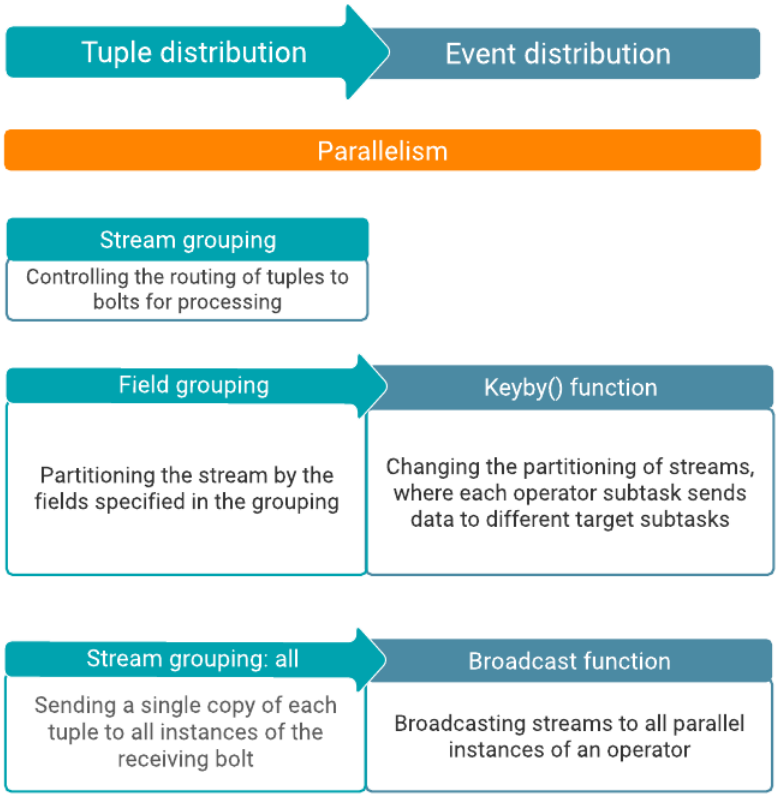


For Cluster layout



Differences in data distribution

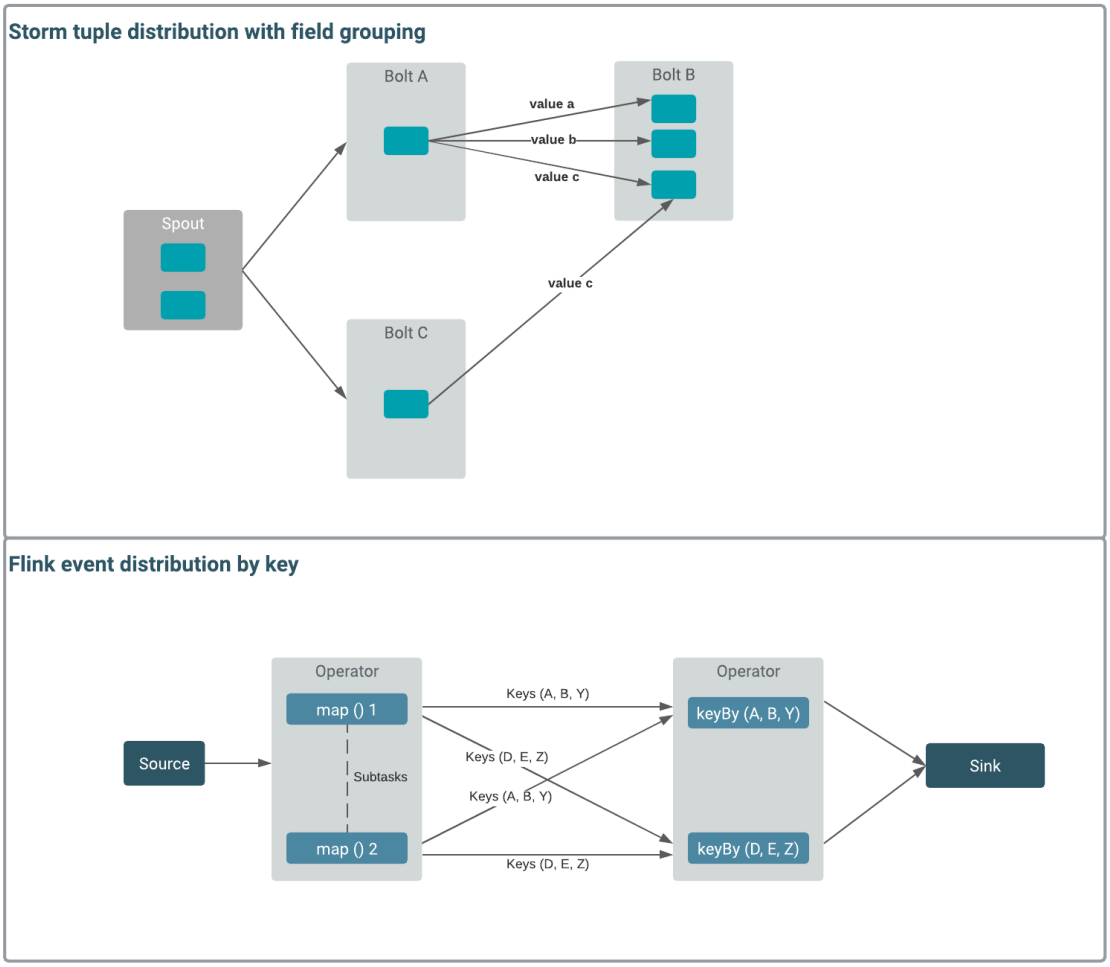
Both Flink and Storm distribute data within their processing elements. Stream grouping in Storm controls the routing of tuples. There is no similar function in Flink, but you can use keys and the broadcast function on your data stream to handle the distribution of events.



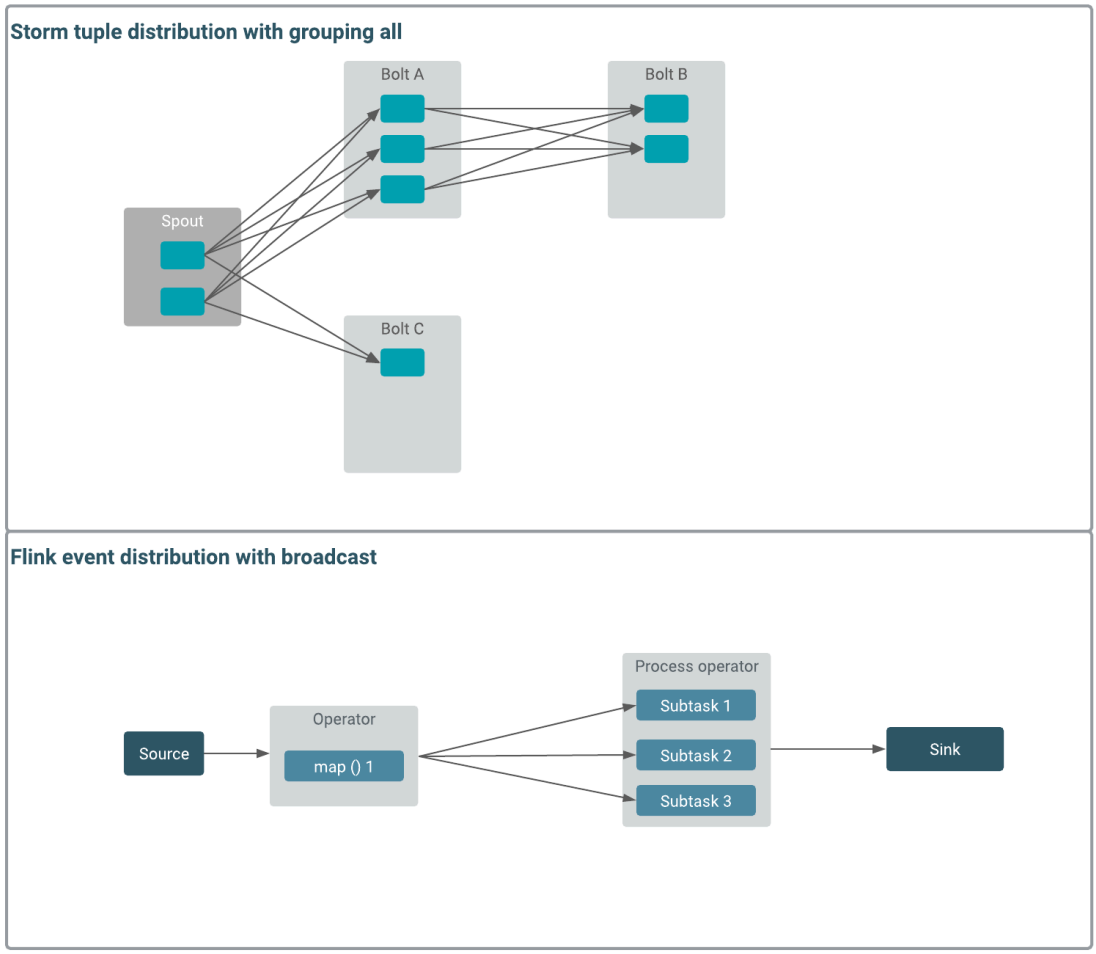
When exchanging data between the elements, Storm supports different methods that include shuffle, field, all, direct, custom, and global. These methods determine if all the data is shared between all bolts, or just certain data with defined fields. In Flink, you can achieve similar result using keys and the broadcast function. The keyBy function is used to partition and group the data together within the incoming stream by given properties or keys. When broadcasting, you share an incoming stream with all parallel instances of an operator. The most common use case for broadcast is sharing a set of rules or raw data within the operators. Like this, the operators process the stream, based on the same configuration, or they work on the same data for analytical purposes.

The following illustrations show the comparison of data distributing methods of Storm and Flink.

For Field grouping and keyby function



For All grouping and broadcast function



Migrating from Storm to Flink

After understanding the differences and similarities of Storm and Flink, you can migrate your Storm topologies to Flink data pipelines. When your Flink application is ready, you need to submit your Flink job to your Cloudera Streaming Analytics (CSA) cluster. Using the Flink Dashboard and Atlas you can monitor your application and job metadata.

The following high-level steps summarize the migration process from Storm to Flink:

1. Identify the Flink application sources and sinks in your Storm topology.
2. Identify the Flink application business logic from your Storm topology.
3. Map your Storm topology to the Flink data pipeline.
    a. If you are using Trident, you need to consider how Flink handles state in itself. For more information, see State handling in Flink and the Stateful Tutorial.
4. Build your Flink application project.

After creating the Flink application, you only need to set up the CSA cluster, and submit your Flink application to your cluster.

Related Concepts

Core features of Flink

Stateful Tutorial

Related Information

CSA Quickstart

Installing CSA Parcel

Adding Flink as a Service

Flink Quickstart Archetype

Flink streaming application structure

Running a Flink Job

Metadata Manegement with Atlas

Monitoring with Flink Dashboard