Streaming Data Processing - Storm Vs Spark.

Apache Spark is an in-memory distributed data analysis platform– primarily targeted at speeding up batch analysis jobs, iterative machine learning jobs, interactive query and graph processing.

One of Spark's primary distinctions is its use of RDDs or Resilient Distributed Datasets. RDDs are great for pipelining parallel operators for computation and are, by definition, immutable, which allows Spark a unique form of fault tolerance based on lineage information. If you are interested in, for example, executing a Hadoop MapReduce job much faster, Spark is a great option (although memory requirements must be considered).

Apache Storm is focused on stream processing or what some call complex event processing. Storm implements a fault tolerant method for performing a computation or pipelining multiple computations on an event as it flows into a system. One might use Storm to transform unstructured data as it flows into a system into a desired format.

Storm and Spark are focused on fairly different use cases. The more "apples-to-apples" comparison would be between Storm and Spark Streaming. Since Spark's RDDs are inherently immutable, Spark Streaming implements a method for "batching" incoming updates in user-defined time intervals that get transformed into their own RDDs. Spark's parallel operators can then perform computations on these RDDs. This is different from Storm which deals with each event individually.

Xinh's Tech Blog.

> **Overview**
>
> Both Storm and Spark Streaming are open-source frameworks for distributed stream processing. But, there are important differences as you will see in the following side-by-side comparison.
>
> **Processing Model, Latency**
>
> Although both frameworks provide scalability and fault tolerance, they differ fundamentally in their processing model. Whereas Storm processes incoming events one at a time, Spark Streaming batches up events that arrive within a short time window before processing them. Thus, Storm can achieve sub-second latency of processing an event, while Spark Streaming has a latency of several seconds.
>
> **Fault Tolerance, Data Guarantees**
>
> However, the tradeoff is in the fault tolerance data guarantees. Spark Streaming provides better support for stateful computation that is fault tolerant. In Storm, each individual record has to be tracked as it moves through the system, so Storm only guarantees that each record will be processed at least once, but allows duplicates to appear during recovery from a fault. That means mutable state may be incorrectly updated twice.
>
> **Spark Streaming**, on the other hand, need only track processing at the batch level, so it can efficiently guarantee that each mini-batch will be processed exactly once, even if a fault such as a node failure occurs. [Actually, Storm's Trident library also provides exactly once processing. But, it relies on transactions to update state, which is slower and often has to be implemented by the user.]

	Storm	Spark Streaming
Origin	BackType, Twitter	UC Berkeley
Implemented in	Clojure	Scala
API language	Java (and others)	Scala, Java
Batch framework integration	N/A	Spark

Figure 1: StormVsSparkStreaming

From Stackoverflow.

> **Apache Spark** is an in-memory distributed data analysis platform– primarily targeted at speeding up batch analysis jobs, iterative machine learning jobs, interactive query and graph processing.
>
> One of Spark's primary distinctions is its use of RDDs or Resilient Distributed Datasets. RDDs are great for pipelining parallel operators for computation and are, by definition, immutable, which allows Spark a unique form of fault tolerance based on lineage information. If you are interested in, for example, executing a Hadoop MapReduce job much faster, Spark is a great option (although memory requirements must be considered).
>
> **Apache Storm** is focused on stream processing or what some call complex event processing. Storm implements a fault tolerant method for performing a computation or pipelining multiple computations on an event as it flows into a system. One might use Storm to transform unstructured data as it flows into a system into a desired format.
>
> Storm and Spark are focused on fairly different use cases. The more "apples-to-apples" comparison would be between Storm and Spark Streaming. Since Spark's RDDs are inherently immutable, Spark Streaming implements a method for "batching" incoming updates in user-defined time intervals that get transformed into their own RDDs. Spark's parallel operators can then perform computations on these RDDs. This is different from Storm which deals with each event individually.
>
> One key difference between these two technologies is that Spark performs Data-Parallel computations while Storm performs Task-Parallel computations. Either design makes tradeoffs that are worth knowing. I would suggest checking out these links.