Apache Storm and Spark Streaming Compared

P. Taylor Goetz, Hortonworks
@ptgoetz

Honestly...

- I know a lot more about Apache Storm than I do Apache Spark Streaming.
- I've been involved with Apache Storm, in one way or another, since it was open-sourced.
- I'm admittedly biased.

But...

- A number of articles/papers comparing Apache Storm and Spark Streaming are inaccurate in terms of Storm’s features and performance characteristics.
- Code and configuration for those studies is not available, so independent verification is impossible.
- Claims don't match real-world observations.

But...

- There is an inherent “Home Team Advantage” in any benchmark comparison.
- Without open source code, any benchmark claims are essentially marketing fluff, and should be taken with a grain or two of NaCl.
- Any benchmark claim should be independently verifiable.

Spark Streaming Paper

- Compares Spark Streaming (Micro-Batch) to Core Storm (One-at-a-Time)
- A more appropriate comparison would have been with Storm's Trident (Micro-Batch) API
- Trident mentioned only in passing (on pages 3 and 12)

http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-259.pdf

Spark Streaming Paper

- Benchmark code/configuration not publicly available
- Performance claims not independently verifiable

http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-259.pdf

Spark Streaming Paper

- Granted, the Spark Streaming paper is almost 2 years old and written at a time when Trident was relatively new.
- However, that paper is often cited when comparing Apache Storm and Spark Streaming, particularly in terms of performance.
- A lot can change in 2 years.

http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-259.pdf

Streaming and batch processing are fundamentally different.

Batch vs. Streaming

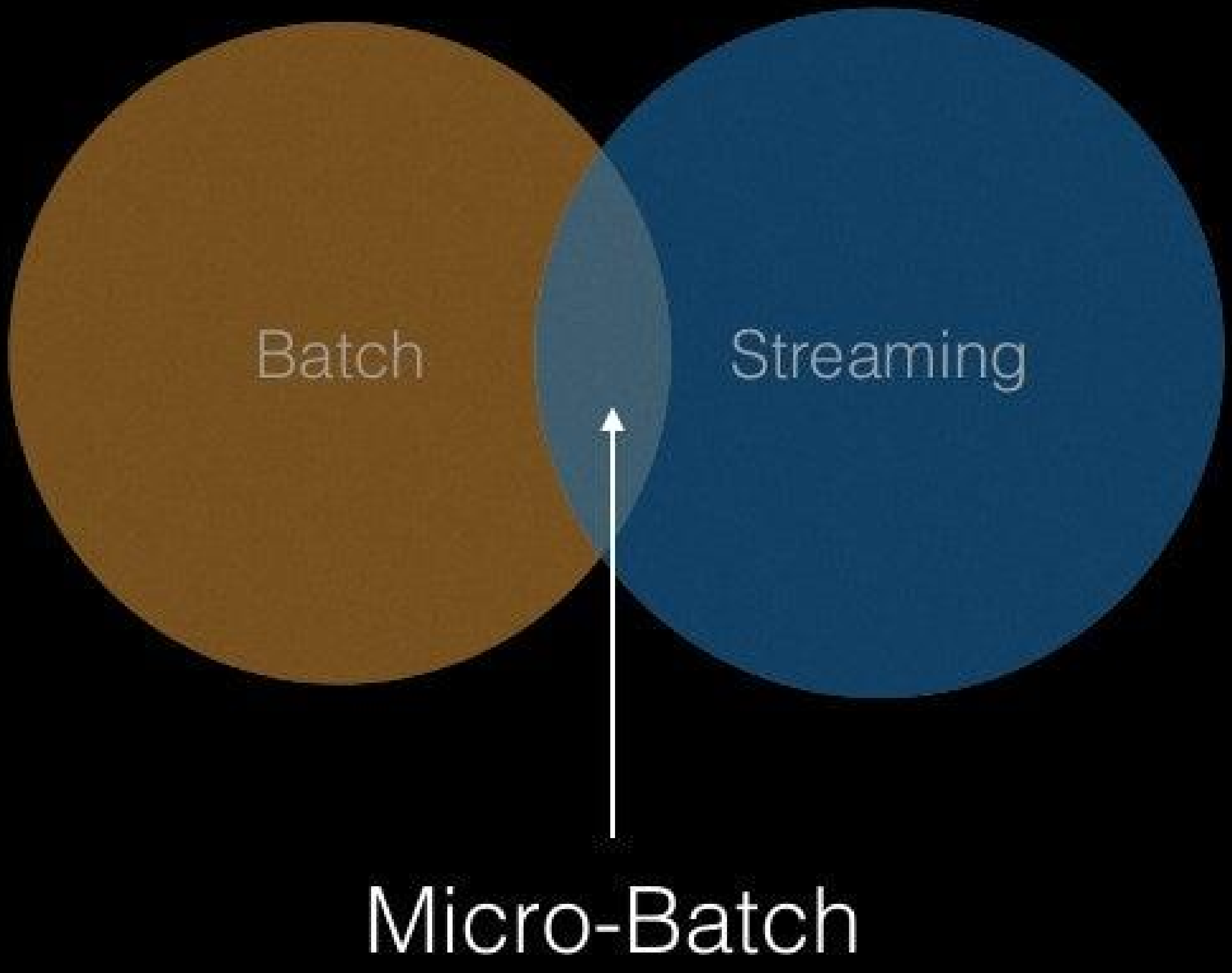
- ***Storm*** is a stream processing framework that also does micro-batching (Trident).

- ***Spark*** is a batch processing framework that also does micro-batching (Spark Streaming).

Batch vs. Streaming

Batch
Streaming

Batch vs. Streaming



Apache Storm: Two Streaming APIs

Core Storm (Spouts and Bolts)

- One at a Time
- Lower Latency
- Operates on Tuple Streams

Trident (Streams and Operations)

- Micro-Batch
- Higher Throughput
- Operates on Streams of Tuple Batches and Partitions

Language Options

Core Storm	Storm Trident	Spark Streaming
• Java • Clojure • Scala • Python • Ruby • others*	• Java • Clojure • Scala	• Java • Scala • Python

*Storm's Multi-Lang feature allows the use of virtually any programming language.

Reliability Models

	Core Storm	Storm Trident	Spark Streaming
At Most Once	Yes	Yes	No
At Least Once	Yes	Yes	No*
Exactly Once	No	Yes	Yes*

*In some node failure scenarios, Spark Streaming falls back to at-least-once processing or data loss.

Programing Model

	Core Storm	Storm Trident	Spark Streaming
Stream Primitive	Tuple	Tuple, Tuple Batch, Partition	DStream
Stream Source	Spouts	Spouts, Trident Spouts	HDFS, Network
Computation/ Transformation	Bolts	Filters, Functions, Aggregations, Joins	Transformation, Window Operations
Stateful Operations	No (roll your own)	Yes	Yes
Output/ Persistence	Bolts	State, MapState	foreachRDD

Production Deployments

Apache Storm	Spark Streaming
• Too many to list http://storm.incubator.apache.org/documentation/Powered-By.html	• Sharethrough http://engineering.sharethrough.com/blog/2014/06/27/sharethrough-at-spark-summit-2014-spark-streaming-for-realtime-auctions/

Support

	Apache Storm	Spark	Spark Streaming
Hadoop Distro	Hortonworks, MapR	Cloudera, MapR, Hortonworks (preview)	Hortonworks, Cloudera, MapR
Resource Management	YARN, Mesos	YARN, Mesos	YARN*, Mesos
Provisioning/ Monitoring	Apache Ambari	Cloudera Manager	?

*With issues: http://spark-summit.org/wp-content/uploads/2014/07/Productionizing-a-247-Spark-Streaming-Service-on-YARN-Ooyala.pdf

Failure Scenarios

Worker Failure: Spark Streaming

"So if a worker node fails, then the system can recompute the lost from the the left over copy of the input data. However, if the worker node where a network receiver was running fails, ***then a tiny bit of data may be lost***, that is, the data received by the system but not yet replicated to other node(s)."

Only HDFS-backed data sources are fully fault tolerant.

https://spark.apache.org/docs/latest/streaming-programming-guide.html#fault-tolerance-properties

Worker Failure: Spark Streaming

Solution?: Write Ahead Logs (SPARK-3129)

- Enabling WAL requires DFS (HDFS, S3) — no such requirement with Storm
- Incurs a performance penalty that adds to overall latency
- ***Full fault tolerance still requires a data source that can replay data (e.g. Kafka)***
- Architectural band aid?

https://databricks.com/blog/2015/01/15/improved-driver-fault-tolerance-and-zero-data-loss-in-spark-streaming.html