A Comparison of Apache Tez and Apache Spark

Cuneyt G. Akcora
Data Security and Privacy Lab
University of Texas at Dallas
November 2016

Outline

- Apache Tez (Hortonworks)
- Apache Spark (Amp Lab, Databricks, also Cloudera)
- A comparison of frameworks: efficiency, popularity
- Other alternatives

Apache Tez Basics

- Apache Tez is inspired by the Microsoft Dryad data flow framework. Tez comes from the Stinger initiative to speed up Hive
- A dataflow graph where vertices represent the application logic and dataflow edges
- Java API to express a DAG of data processing. The API has three components: DAG, Vertex and Edge

Apache Tez Basics

- The user creates a DAG object for each data processing job
- The user creates a Vertex object for each step in the job
- The user creates an Edge object and connects the producer and consumer vertices using it

Apache Tez Basics

- Output of the vertices can be persisted to a local file system or HDFS with or with out replication based on the reliability requirements

- Pluggable vertex management modules gather runtime information, the execution engine uses this data to optimize and reconfigure execution plan downstream (e.g., increase number of reducers based on data size)

Apache Tez: Bottom line

"MapReduce and Tez use the same logical programming paradigm, but Tez uses dataflow DAGs for resource optimization and data pipeline planning. This means that Tez provides an order of magnitude speed boost over MapReduce, but has the same overly rigid design limitations" [4].

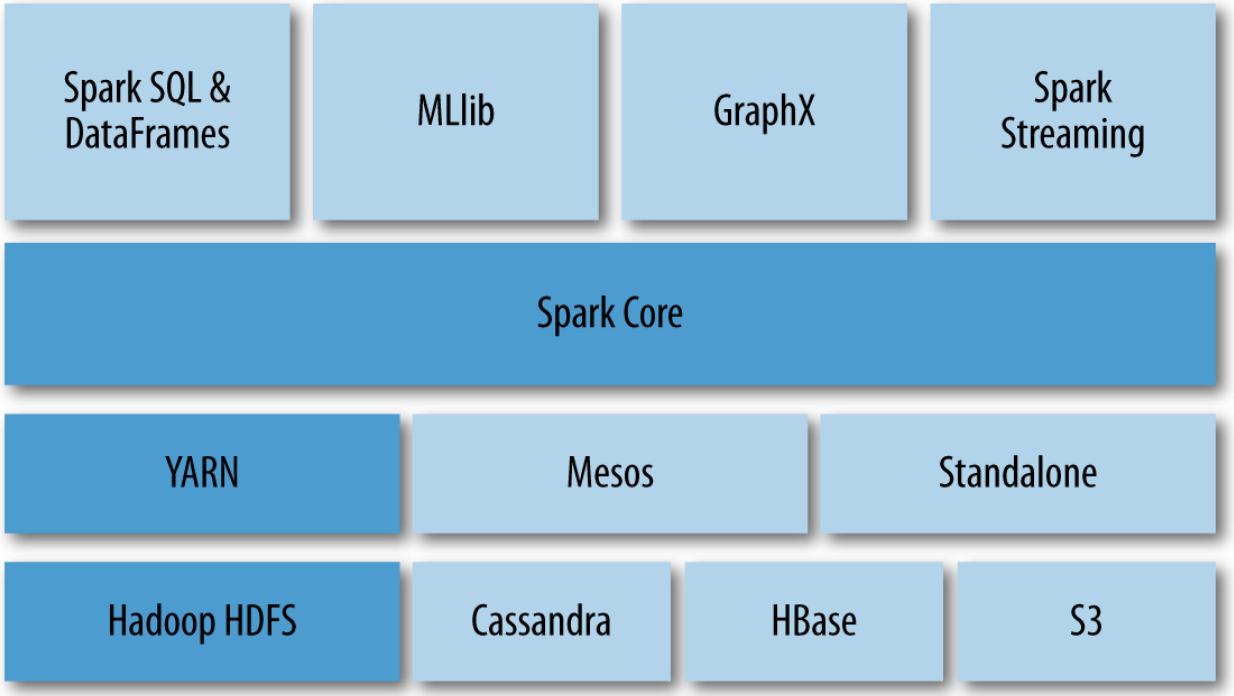
"You are more likely to be using tools (e.g., Hive) powered by Apache Tez (as an execution engine) rather than Tez directly" Thejas Nair, Hortonworks

Apache Spark Basics

- Apache Spark was developed in 2009. Has APIs in Scala, Java, R, and Python
- Interactive, iterative cluster computing platform [3]
- 1) abstraction over distributed memory on clusters 2) abstract transformations (e.g., sample, filter, join, collect) that apply the same operation to many data items

Apache Spark Basics

- Spark converts all transformations (e.g., map()) and terminal actions (e.g, collect(), reduce()) into a DAG and executes it using a DAG execution engine similar to that of Dryad



Apache Spark Basics

- In-memory cluster computation enables Spark to run iterative algorithms, as programs can checkpoint data and refer back to it without reloading it from disk

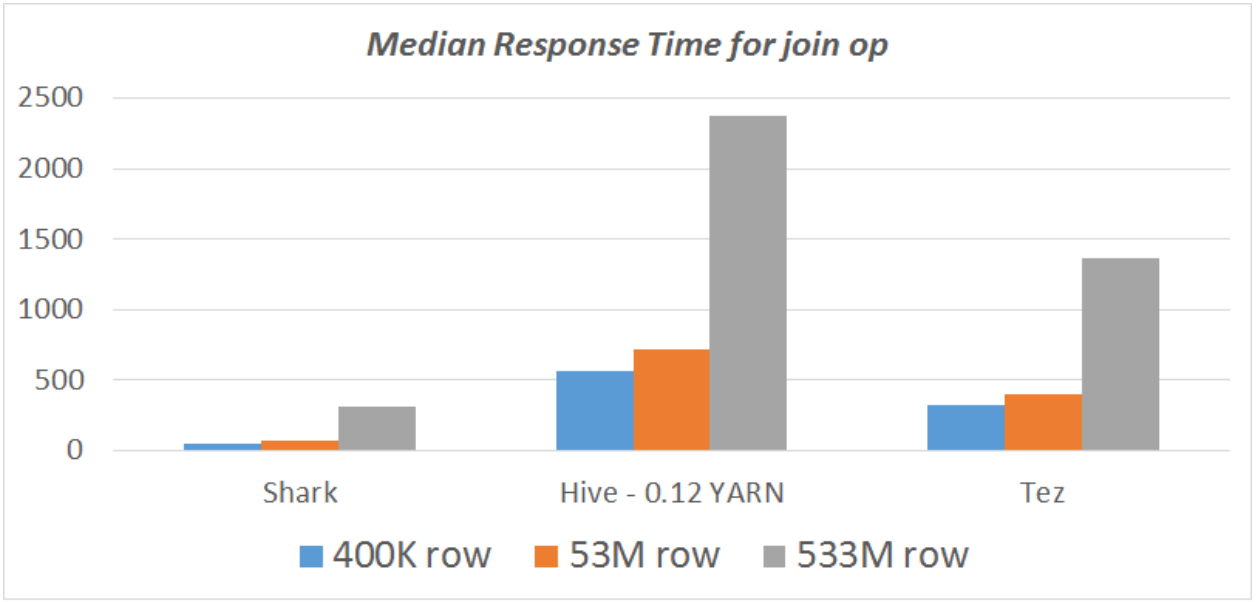
- RDD is a read-only, partitioned collection of records that can be created though deterministic operations on data in a stable storage or other RDD

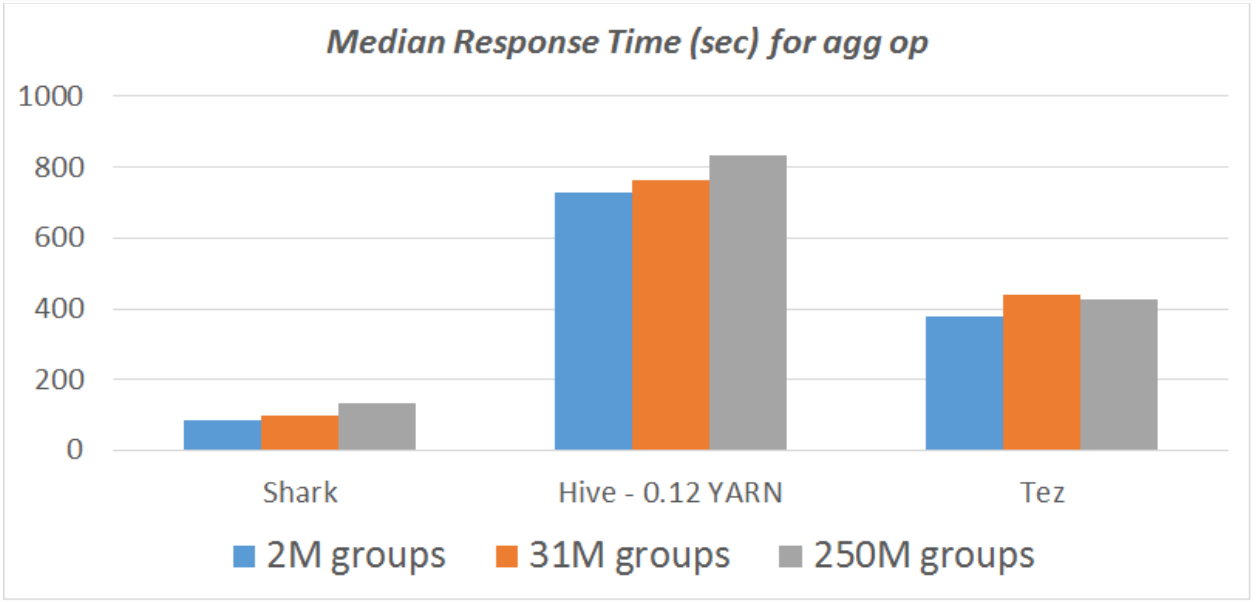
- A series of data flow transformations before performing some action that requires coordination like a reduction or a write to disk

Apache Spark Basics

- Spark focuses purely on computation rather than data storage
- Typically run in a cluster that implements data warehousing and cluster management tools
- When Spark is built with Hadoop, it utilizes YARN to allocate and manage cluster resources like processors and memory via the ResourceManager

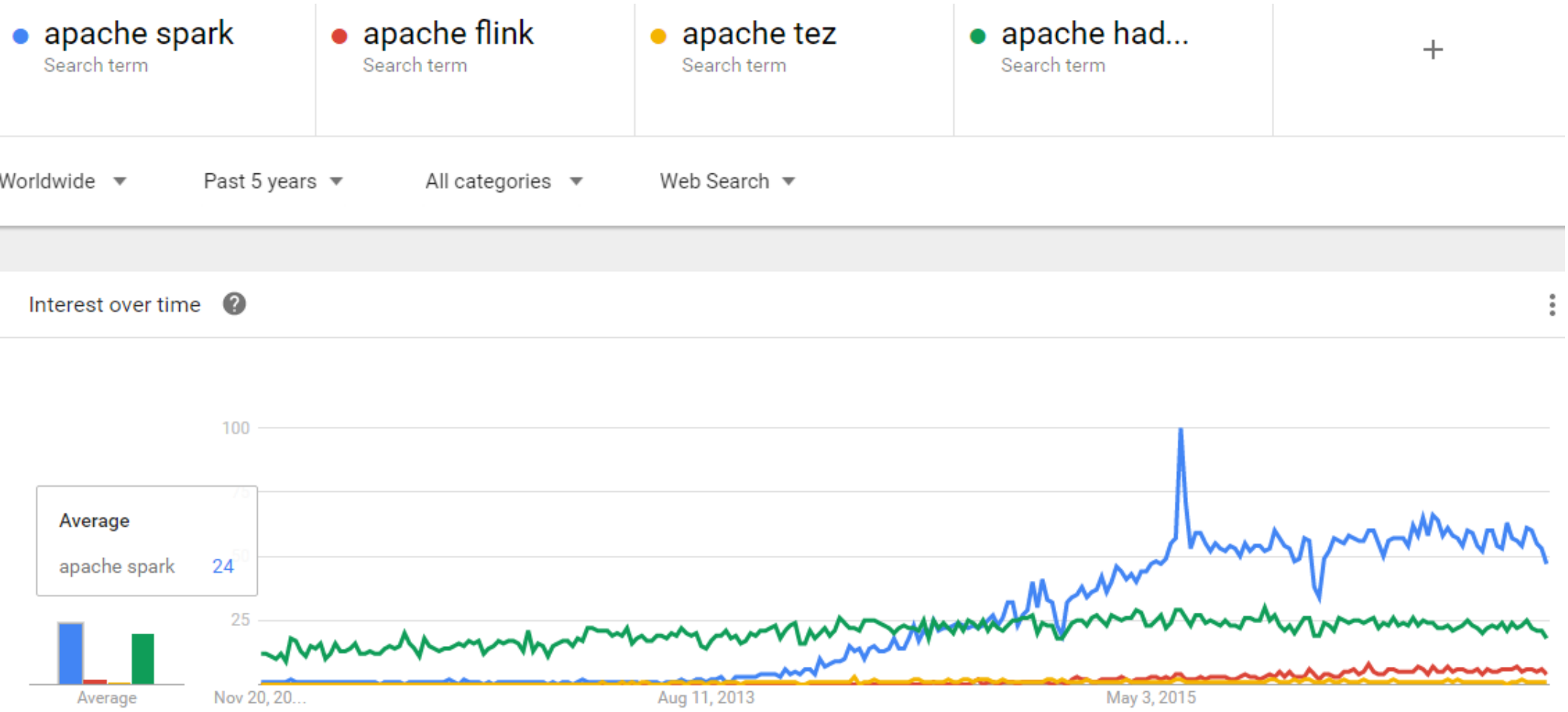
Comparisons - Time





https://amplab.cs.berkeley.edu/benchmark/

Comparisons - Popularity



Comparisons - Code

Word count code examples:

Apache Spark code on the Databricks repo

Apache Tez code on the official repo

Other candidates

Apache Flink [1]:

- Streaming dataflow engine
- Can emulate batch processing, however at its core it is a native streaming engine
- Has advanced streaming capabilities (such as windowing features, etc)

Apache Storm:

- Has a compositional API
- You build up your topology with basic building blocks like sources or operators
- Used for streaming data

Source: Stefan Papp on Quora

Attribute Comparison

	Spark Streaming	Storm	Flink
Current version	1.6.1	1.0.0	1.0.2
Category	ESP	ESP/CEP	ESP/CEP
Event size	micro-batch	single	single
Available since (incubator since)	Feb 2014 (2013)	Sep 2014 (Sep 2013)	Dec 2014 (Mar 2014)
Contributors	838	207	159
Main backers	AMPLab Databricks	Backtype Twitter	dataArtisans
Delivery guarantees	exactly once at least once (with non-fault-tolerant sources)	at least once	exactly once
State management	checkpoints	record acknowledgements	distributed snapshots
Fault tolerance	yes	yes	yes
Out-of-order processing	no	yes	yes
Event prioritization	programmable	programmable	programmable
Windowing	time-based	time-based count-based	time-based count-based
Back-pressure	yes	yes	yes
Primary abstraction	DStream	Tuple	DataStream
Data flow	application	topology	streaming dataflow
Latency	medium	very low	low (configurable)
Resource management	YARN Mesos	YARN Mesos	YARN
Auto-scaling	yes	no	no
In-flight modifications	no	yes (for resources)	no
API	declarative	compositional	declarative
Primarily written in	Scala	Clojure	Java

Source: twitter.com/IanHellstrom

References

[1] Beyond Hadoop MapReduce
Apache Tez and Apache Spark, Prakasam Kannan

[2] Coursera course on Big Data:
https://www.coursera.org/learn/hadoop/lecture/KnDdc/yarn-tez-and-spark

[3] Data Analytics with Hadoop, Safari Books. Online at
https://www.safaribooksonline.com/library/view/data-analytics-with/9781491913734/ch04.html

[4] The Tragedy of Tez, Paige Roberts. Online at
http://bigdatapage.com/tragedy-tez/

References

[5] I Like Tez, DevOps Edition (WIP), Gopal V., Online at https://github.com/t3rmin4t0r/notes/wiki/I-Like-Tez,-DevOps-Edition-%28WIP%29