Eötvös Loránd University (ELTE)
Faculty of Informatics (IK)
Pázmány Péter sétány 1/c
1117 Budapest, Hungary

STREAMING SYSTEMS: SPARK STREAMING, STORM & FLINK

Open-source Technologies for Real-Time Data Analytics

Imre Lendák, PhD, Associate Professor

2020
Budapest, Hungary

Introduction

- **Spark Streaming** → a microbatch processing platform with strong consistency guarantees
- Apache **Storm** is a real-time, distributed stream processing platform
- Apache **Flink** a unified stream and batch processing framework

SPARK

Introduction & history

Definitions

- **DEF:** Apache Spark is an open-source, distributed, general-purpose **cluster-computing framework**
- The authors aimed to perform **in-memory** calculations in computing clusters without ‘touching the disk’ before reaching the final data processing stage (i.e. output)
- Written in Scala
- Additionally optimized for interactive queries and iterative computing jobs

History

- Originally developed by the AMPLab at UC Berkeley around 2009
- As soon as 2009 it was outperforming MapReduce 10-20x in certain types of problems
- Open sourced in 2010 (BSD license)
- Donated to the Apache Software Foundation in 2013
- Top-level Apache project since 2014

https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/zaharia

Whois AMPLab?

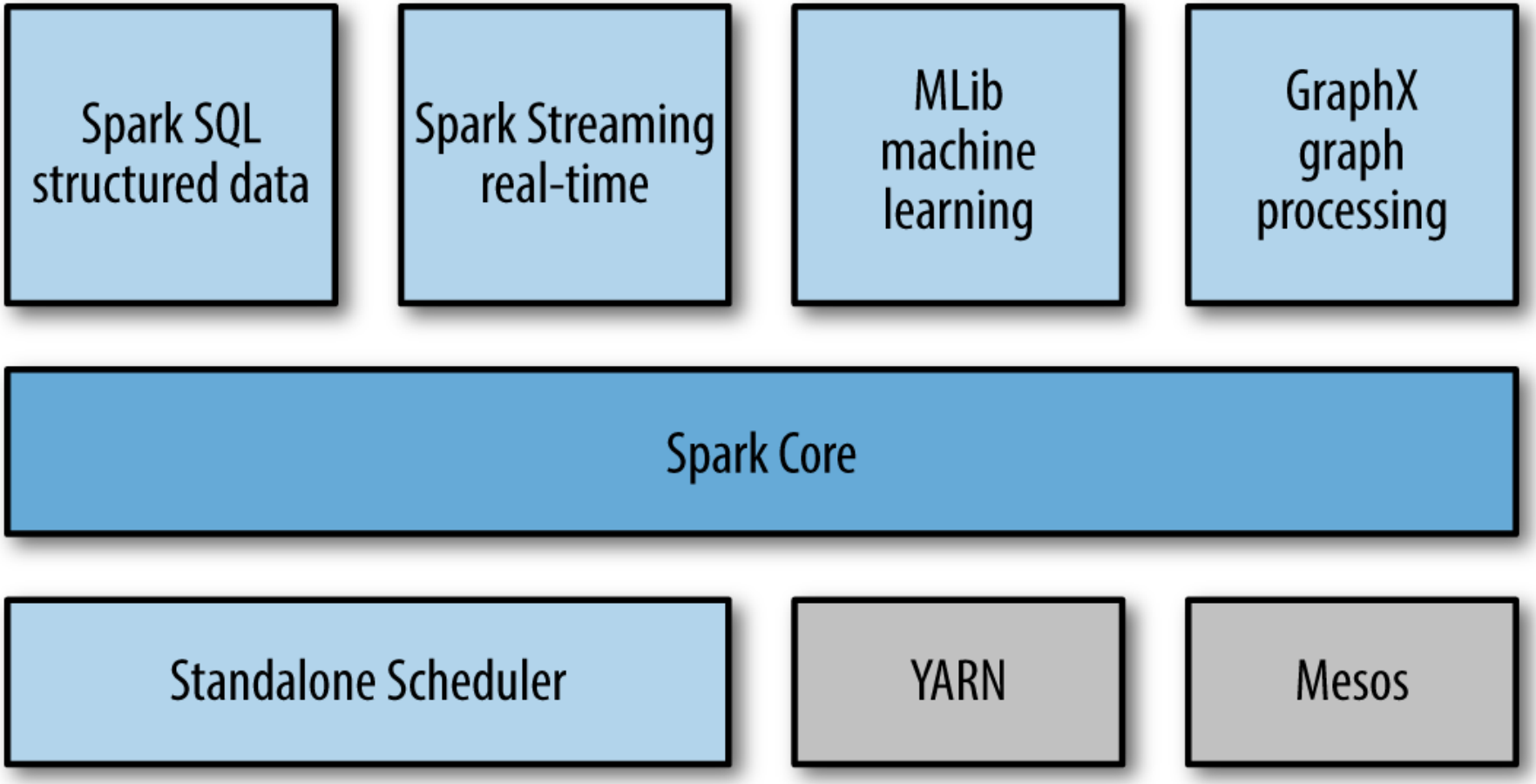
AMPLab

- AMP = Algorithms, Machines and People Lab
- Doing research and publishing scientific publications since 2008
- AMPLab officially launched in 2011
- Worked on different ‘big data’ projects under the Berkeley Data Analytics Stack (BDAS)
- UC Berkeley launched RISELab as the successor to AMPLab in 2017

Best known projects

- **Apache Spark** – distributed, general-purpose computing platform
- **Apache Mesos** – cluster management platform
- **Alluxio** – virtual distributed file system (VFDS) – Alluxio ‘sits’ between computation & storage in large-scale data processing environments. Used by Cray, IBM, Lenovo, Intel, etc.

Sparkitechture



Holden Karau, Andy Konwinski, Patrick Wendell, Matei Zaharia, “Learning Spark: Lightning-Fast Big Data Analysis”, O’Reilly, 2015.

Spark Streaming

- **Spark Core** tasks:
  - Memory management
  - Fault recovery
  - Implements the RDD (v1.0) and Dataset (v2+) Application Programming Interfaces (RDD API vs Dataset API)
  - One of the first platforms to guarantee consistent data analysis
- **Spark Streaming** is a consistent micro-batch processing environment for live streams of data
  - Note: it is important to notice that it is micro-batch oriented → its application is limited to use cases in which processing time windowing is acceptable
  - It was the first stream processing platform to provide strong consistency guarantees

Use cases

- The extension of Spark with Spark Streaming marked a turning point → the Lambda architecture became outdated (for many use cases) as soon as a single platform could handle both batch and stream processing consistently
  - Note: when Spark Streaming appeared, it additionally heated up the micro-batch vs streaming analysis debate(s)

- Common Spark Streaming use cases
  - In-order data
  - Event-time agnostic computations

Storm

Storm Intro

Definitions

- Apache **Storm** is a real-time, distributed stream processing platform
  - Designed as a directed acyclic graph (DAG) data transformation pipeline
  - Note: real-time is a key characteristic
  - Note: Storm was the 1st real streaming system (!)
- Comparable solutions: Flink

Origins & History

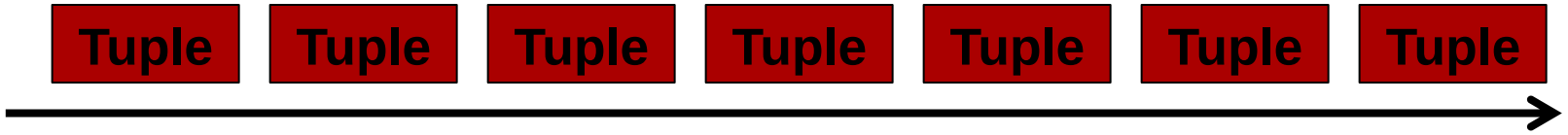
- Original author(s): Nathan Marz (@ BackType)
- Twitter acquired BackType → Storm in Twitter
- Written in: Clojure & Java
- Apache project: 2013
- License: Apache 2.0
- Initial release: September 2011
- Stable release: June 2020 (v2.2.0)

http://nathanmarz.com/blog/history-of-apache-storm-and-lessons-learned.html

Motivation

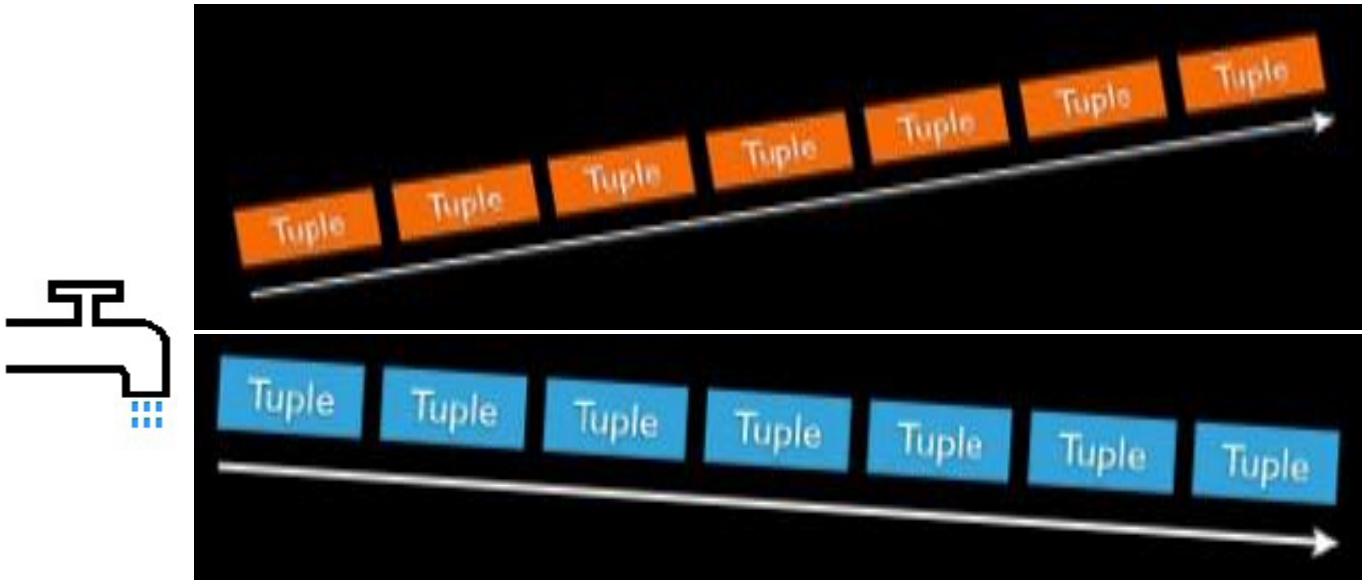
- How to handle the data flow in Twitter?
- How to obtain sufficiently low latency?
- Storm **design decisions**:
  - Combine at-most once and at-least once semantics with per-record processing
  - No internal notion of persistent state (i.e. non-consistent)
- These choices allowed Storm to provide lower latency than any other product on the marketplace in 2011
  - Eventual consistency (in the Twitter use case) was obtained by running a strongly consistent Hadoop cluster in parallel → Lambda architecture → years of dual-pipeline 'darkness'
  - Note: Marz was the author of the Lambda architecture (!)

Streams



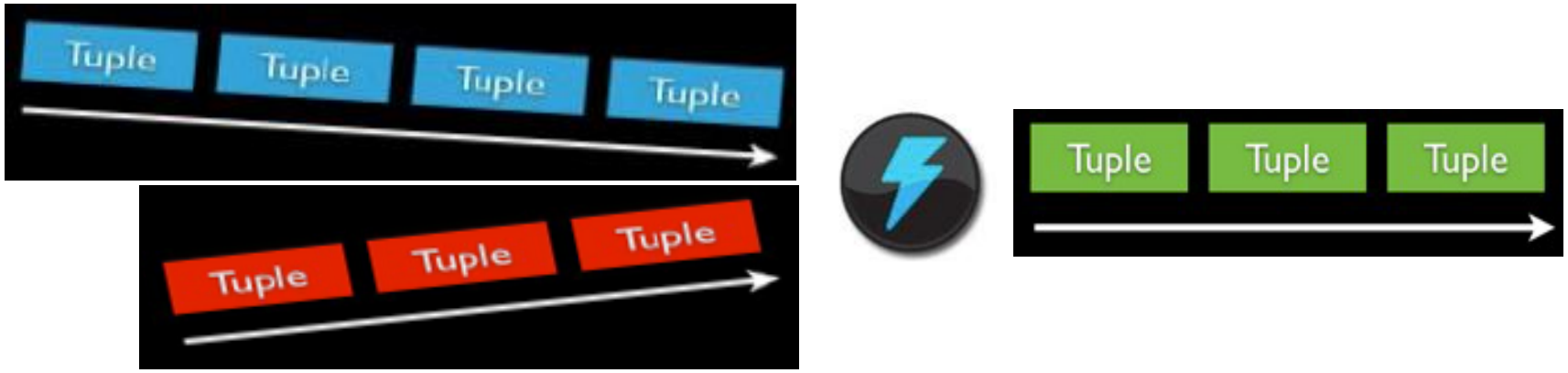
Unbounded sequence of tuples

Spouts



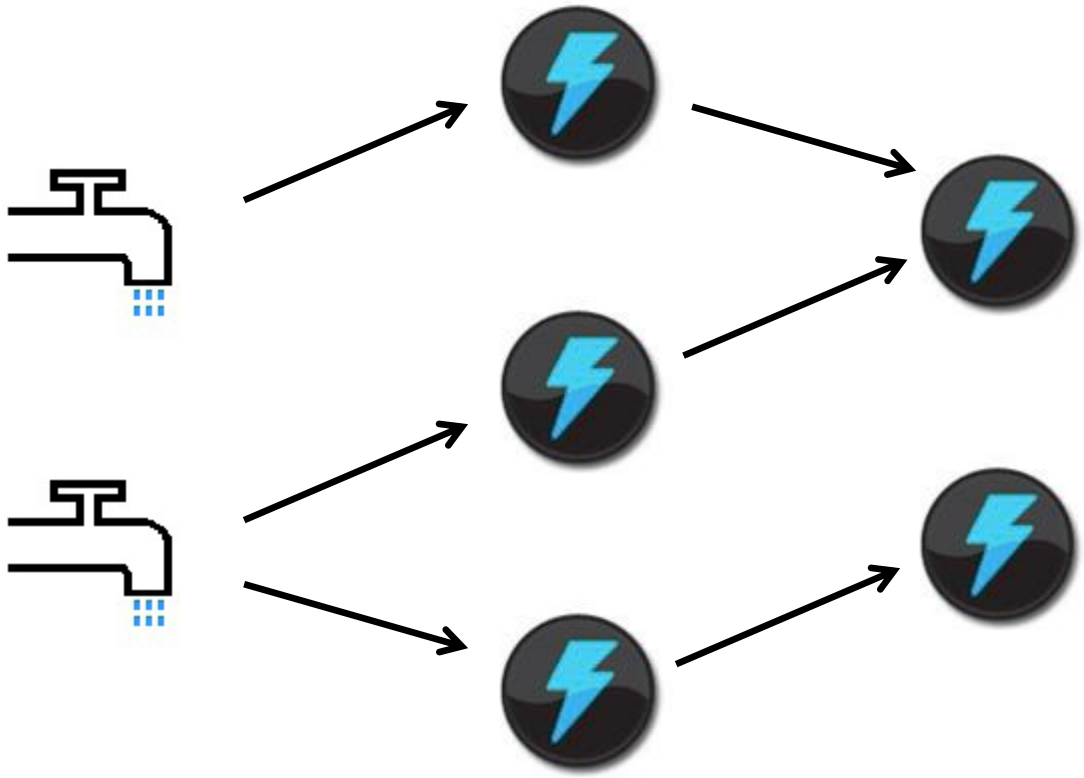
Source of streams

Bolts



Processes input streams and produces new streams:
Can implement functions such as filters, aggregation, join, etc

Topology



Network of spouts and bolts

Who uses the Storm?

Apache Storm vs Kafka Streams
Which companies use these tools?

Apache Storm

Spotify

Twitter

Yelp

Keen IO

Kafka Streams

TransferWise

Red Bull Med...

Doodle

Bottega52

stackshare

Life of Storm

- Twitter announced in 2015 that it was abandoning Storm as its in-house streaming system
- Twitter switched to Heron
- Heron maintained API-level compatibility with Storm
- Heron is an Apache incubator project https://github.com/apache/incubator-heron
- Heron stable release: 0.20.1 (August 2019)
- Storm stable release: 2.2.0 (June 2020)

Additional references

- Twitter Storm Vs Heron
  https://medium.com/@vagrantdev/twitter-storm-vs-heron-12774056f45b
- Apache Flink vs Twitter Heron?
  https://stackoverflow.com/questions/37635736/apache-flink-vs-twitter-heron
- Leaving the Nest: Heron donated to Apache Software Foundation
  https://blog.twitter.com/engineering/en_us/topics/open-source/2018/heron-donated-to-apache-software-foundation.html

FLINK

Flink quick facts

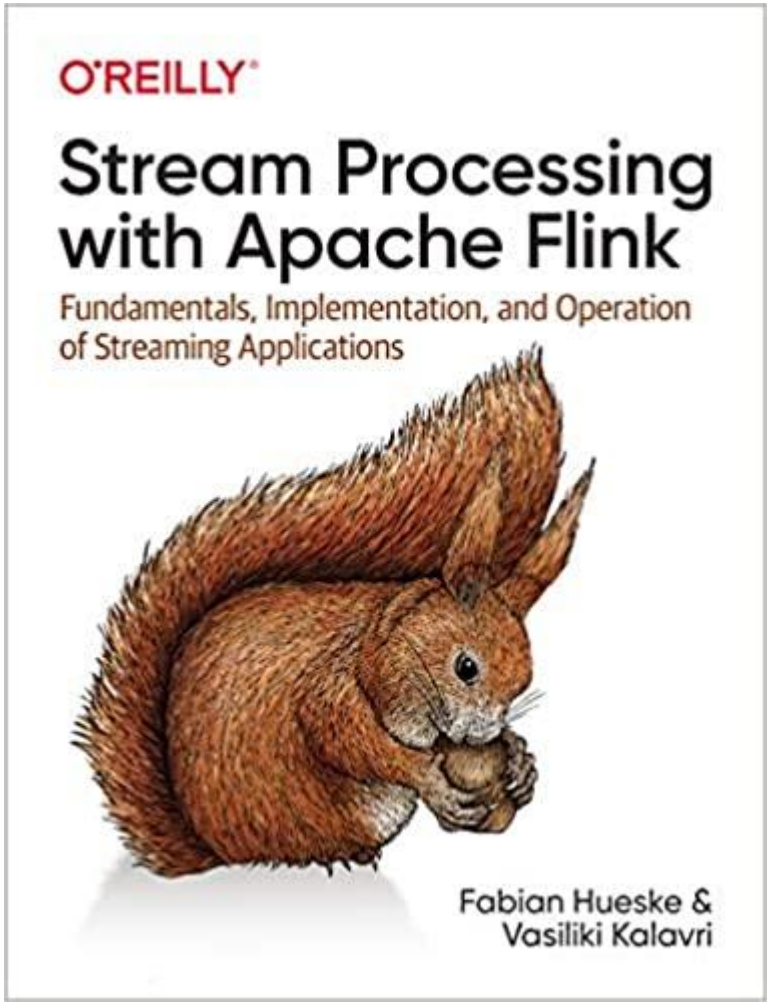
- **DEF:** Apache **Flink** is a unified stream and batch processing framework
- Flink programs consist of streams and transformations
- The Flink runtime supports iterative algorithms → ML
- Fault tolerance via distributed checkpoints
- Two APIs:
  - DataStream API for bounded or unbounded streams of data
  - DataSet API for bounded data sets
- Data source & sink connections: Kafka, HDFS, Cassandra, Elasticsearch

Flink background

- Original code based on the distributed **Stratosphere** project's distributed runtime
  - Stratosphere was financed by the German Research Foundation (DFG)
  - Stratosphere participants: Technical University Berlin, Humboldt-Universität zu Berlin, and Hasso-Plattner-Institut Potsdam
- **Initial release:** May 2011
- **Stable release:** 1.11.0 (July 2020)
- **Written in:** Java & Scala
- **License:** Apache License 2.0
  - Top level since Dec 2014
- Ververica (formerly Data Artisans) employs most of the key contributors

Flink references

- Hueske, F., & Kalavri, V. (2019). Stream Processing with Apache Flink: Fundamentals, Implementation, and Operation of Streaming Applications. O'Reilly Media.



Additional Flink references

- Savepoints: Turning Back Time https://www.ververica.com/blog/turning-back-time-savepoints
- Apache Flink® — Stateful Computations over Data Streams https://flink.apache.org
- Carbone, P., Ewen, S., Fóra, G., Haridi, S., Richter, S., & Tzoumas, K. (2017). State management in Apache Flink®: consistent stateful distributed stream processing. Proceedings of the VLDB Endowment, 10(12), 1718-1729.

Summary

- Spark **Streaming** → a microbatch processing platform with strong consistency guarantees
- Apache **Storm** is a real-time, distributed stream processing platform
- Apache **Flink** a unified stream and batch processing framework

Thank you for your attention!