Interactive data analysis with Apache Spark

Mark Stetzer
@stetzer

What is Spark?

- Product of UC Berkeley's AMPLab
- Commercial support/development provided by recently-funded Databricks ($14 million!)
- Currently in incubator status as Apache project

What is Spark?

- Open source cluster computing platform
- Supports placing data in memory
- Originally intended for iterative algorithms (e.g. machine learning) and interactive data mining
- Language bindings for Scala, Java, and Python

Why use Spark?

- Shown to be up to 100x faster than Hadoop Map/Reduce
- Iterative problems like linear regression
- Interactive data analysis problems like applying arbitrary functions, regexes, etc. against data

Spark Design

- Supports both in-memory and disk-based computation
- Main data structure is Resilient Distributed Datasets (RDDs)
- Because of RDD design, transformations can be lazily computed

RDDs

- Read-only, partitioned collection of records (e.g. RDD[String])
- Do not need to be materialized at all times (stores information on HOW it was computed)
- If a partition is lost, it can be recomputed (hence the resilience/fault tolerance claim)

Geek out on RDDs

- Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing
- http://www.cs.berkeley.edu/~matei/papers/2012/nsdi_spark.pdf

(Scala) API Design

- Parallelized collections - form distributed dataset from existing collection
- Hadoop datasets (load data from hdfs://, hbase://, s3n://, etc.)
- Supports shared variables: broadcast and accumulators

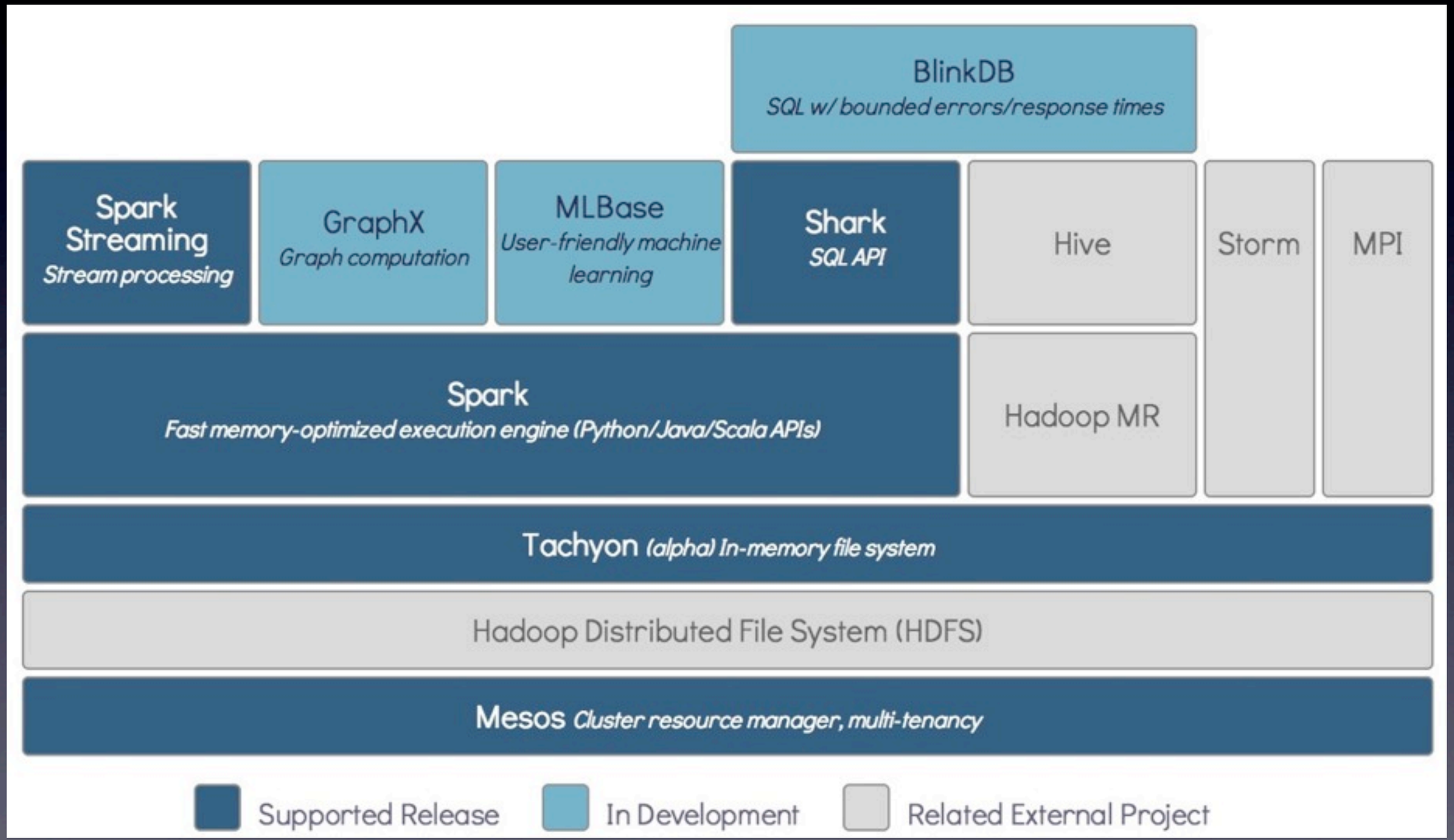
(Scala) API Design

- RDD Operations include: map, reduce, filter, sample, union, distinct, count, saveAsSequenceFile, etc.
- RDDs can be persisted with their cache or persist methods
- http://spark.incubator.apache.org/docs/latest/scala-programming-guide.html#rdd-operations

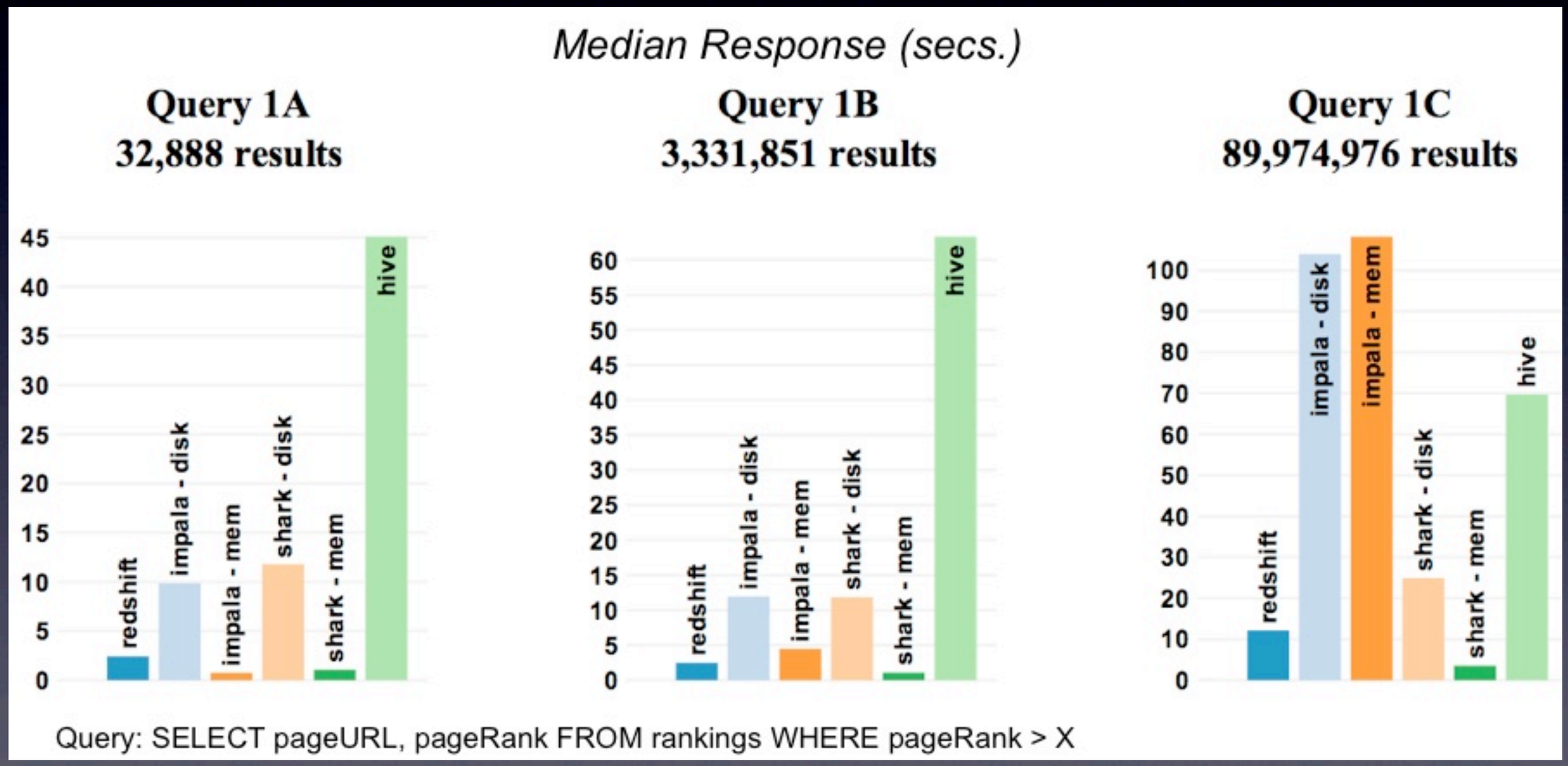
AMPLab BDAS

- Berkeley Data Analytics Stack
- Open source collection of projects built on top of, or ancillary to, Spark and Hadoop/ MapReduce
- Cluster management, stream processing, in-memory filesystem, graph computation, etc.

BDAS Components



How does Spark (Shark) perform?



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

Getting Started

- Download 0.8.0 from spark.incubator.apache.org
- Can download prebuilt versions for Hadoop v1 on Cloudera CDH{3,4}
- Can run against cluster or interactive local shell (which we will be using)

Interactive Shell

- MASTER=local[8] ./spark-shell
- Provides a Scala REPL (version 2.9.3 - do not use > 2.9.x)
- Customized version of REPL to capture variables & broadcast

Wikilinks Corpus

- ~ 5.5 GB of textual references to English Wikipedia pages
- Intended for use in disambiguation work

- https://code.google.com/p/wiki-links/
- http://googleresearch.blogspot.com/2013/03/learning-from-big-data-40-million.html

Wikilinks Corpus

```
URL     ftp://38.107.129.5/Training/Training%20Documentation/Latitude%20V6.2%20Training%20Binder/06%20Latitude%206%202%2
MENTION Microsoft       80679   http://en.wikipedia.org/wiki/Microsoft
MENTION Microsoft       134415  http://en.wikipedia.org/wiki/Microsoft
MENTION Windows Server 2008     80862   http://en.wikipedia.org/wiki/Windows_Server_2008
MENTION Windows Server 2008     134744  http://en.wikipedia.org/wiki/Windows_Server_2008
MENTION Windows 7       81028   http://en.wikipedia.org/wiki/Windows_7
MENTION Windows 7       134910  http://en.wikipedia.org/wiki/Windows_7
MENTION operating systems.      81109   http://en.wikipedia.org/wiki/Operating_system
MENTION Windows Vista   134573  http://en.wikipedia.org/wiki/Windows_Vista
TOKEN   Fresh   54828
TOKEN   evidence        32081
TOKEN   Allow   72597
TOKEN   operator        148693
TOKEN   notice  507684
TOKEN   save    77567
TOKEN   subfolder       154988
TOKEN   PELCO   490470
TOKEN   crashed 301434
TOKEN   audit   296060
```

Obligatory Word Count

Show me the terms with the most mentions

```
1  val lines = sc.textFile("/Users/mstetzer/code-projects/wikilinks/data*")
2  val mentions = lines.filter(l => l.startsWith("MENTION"))
3  val mentionCounts = mentions.map{l => val txt = l.split("\t")(1); (txt, 1)}.reduceByKey((a, b) => a + b, 8)
4  val sortedMentions = mentionCounts.map{case (k, v) => (v, k)}.sortByKey(false, 8)
5  sortedMentions.take(10).foreach(println _)
```

More than Word Count

Let's count the most popular URL paths

```
1  import scala.util.Try
2
3  val paths = mentions.map{l => (Try(new java.net.URL(l.split("\t")(3)).getPath).getOrElse(null), 1)}.filter(p => p != null)
4  val pathCounts = paths.reduceByKey((a, b) => a + b, 8)
5  val sortedPaths = pathCounts.map{case (k, v) => (v, k)}.sortByKey(false, 8)
6  sortedPaths.take(10).foreach(println _)
```

What have I used Spark for?

- Interactive data mining - apply one regex to set of data, slightly different or additional regexes to same data, etc.
- ETL - Fast cluster-wide ETL with interactive shell
- Stream processing w/ Spark Streaming - calculating topN results over a time window / time-skewed joins

Questions?

Flickr / marsmet451