The State of Spark
And Where We’re Going Next

Matei Zaharia
Community Growth
Project History

Spark started as research project in 2009

Open sourced in 2010
  » 1st version was 1600 LOC, could run Wikipedia demo

Growing community since

Entered Apache Incubator in June 2013
Development Community

With over 100 developers and 25 companies, one of the most active communities in big data

Comparison: Storm (48), Giraph (52), Drill (18), Tez (12)

Past 6 months: more active devs than Hadoop MapReduce!
Development Community

Healthy across the whole ecosystem

amplab / shark

Hive on Spark http://shark.cs.berkeley.edu/

1,029 commits
10 branches
5 releases
28 contributors
Release Growth

Spark 0.6:
- Java API, Maven, standalone mode
- 17 contributors

Spark 0.7:
- Python API, Spark Streaming
- 31 contributors

Spark 0.8:
- YARN, MLlib, monitoring UI
- 67 contributors
Some Community Contributions

YARN support (Yahoo!)
Columnar compression in Shark (Yahoo!)
Fair scheduling (Intel)
Metrics reporting (Intel, Quantifind)
New RDD operators (Bizo, ClearStory)
Scala 2.10 support (Imaginea)
Conferences

<table>
<thead>
<tr>
<th>Event</th>
<th>Attendees</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMP Camp 1 (Aug 2012)</td>
<td>150</td>
</tr>
<tr>
<td>AMP Camp 2 (Aug 2013)</td>
<td>250</td>
</tr>
<tr>
<td>Spark Summit (Nov 2013)</td>
<td>450</td>
</tr>
</tbody>
</table>
Projects Built on Spark

- BlinkDB
- Shark (SQL)
- Spark Streaming (real-time)
- GraphX (graph)
- MLbase (machine learning)
What’s Next?
Our View

While big data tools have advanced a lot, they are still far too difficult to tune and use.

**Goal:** design big data systems that are as powerful & seamless as those for small data.
Current Priorities

Standard libraries

Deployment

Out-of-the-box usability

Enterprise use .databricks™ + cloudera
Standard Libraries

While writing K-means in 30 lines is great, it’s even better to call it from a library!

Spark’s MLlib and GraphX will be standard libraries supported by core developers
  » MLlib in Spark 0.8 with 7 algorithms
  » GraphX coming soon
  » Both operate directly on RDDs
val rdd: RDD[Array[Double]] = ...
val model = KMeans.train(rdd, k = 10)

val graph = Graph(vertexRDD, edgeRDD)
val ranks = PageRank.run(graph, iters = 10)
Standard Libraries

Beyond these libraries, Databricks is investing heavily in higher-level projects

Spark Streaming:
easier 24/7 operation and optimizations coming in 0.9

Shark:
calling Spark libs (e.g. MLlib), optimizer, Hive 0.11 & 0.12

Goal: a complete and interoperable stack
Deployment

Want Spark to easily run anywhere

Spark 0.8: much improved YARN, EC2 support

Spark 0.8.1: support for YARN 2.2

SIMR: launch Spark in MapReduce clusters as a Hadoop job (no installation needed!)
  » For experimenting; see talk by Ahir
Ease of Use

Monitoring and metrics (0.8)
Better support for large # of tasks (0.8.1)
High availability for standalone mode (0.8.1)
External hashing & sorting (0.9)

Long-term: remove need to tune beyond defaults
Next Releases

Spark 0.8.1 (this month)
  » YARN 2.2, standalone mode HA, optimized shuffle, broadcast & result fetching

Spark 0.9 (Jan 2014)
  » Scala 2.10 support, configuration system, Spark Streaming improvements
What Makes Spark Unique?
Big Data Systems Today

General batch processing

Specialized systems (iterative, interactive and streaming apps)
Spark’s Approach

Instead of specializing, generalize MapReduce to support new apps in same engine

Two changes (general task DAG & data sharing) are enough to express previous models!

Unification has big benefits
  » For the engine
  » For users
Code Size

- Hadoop MapReduce
- Storm (Streaming)
- Impala (SQL)
- Giraph (Graph)
- Spark

non-test, non-example source lines
Code Size

- Hadoop MapReduce
- Storm (Streaming)
- Impala (SQL)
- Giraph (Graph)
- Spark

non-test, non-example source lines
Code Size

- Hadoop MapReduce
- Storm (Streaming)
- Impala (SQL)
- Giraph (Graph)
- Spark

non-test, non-example source lines

* also calls into Hive
Code Size

- Hadoop MapReduce
- Storm (Streaming)
- Impala (SQL)
- Giraph (Graph)
- Spark

- GraphX
- Shark*
- Streaming

* also calls into Hive

Non-test, non-example source lines
Performance

![Throughput (MB/s/node)](image)

- Streaming
- Storm
- Spark

![Response Time (s)](image)

- SQL
- Hive
- Impala (disk)
- Impala (mem)
- Shark (disk)
- Shark (mem)

![Response Time (min)](image)

- Graph
- Hadoop
- Giraph
- GraphX

Throughput (MB/s/node)

Response Time (s)

Response Time (min)
What it Means for Users

Separate frameworks:

- HDFS read
- ETL
- HDFS write

- HDFS read
- train
- HDFS write

- HDFS read
- query
- HDFS write

Spark:

- Interactive analysis

HDFS

ETL

train

query

python

Scala
Combining Processing Types

From Scala:

``` scala
val points = sc.runSql[Double, Double](
  "select latitude, longitude from historic_tweets")

val model = KMeans.train(points, 10)

sc.twitterStream(...)
  .map(t => (model.closestCenter(t.location), 1))
  .reduceByWindow("5s", _ + _)
```
Combining Processing Types

From SQL (in Shark 0.8.1):

```
GENERATE KMeans(tweet_locations) AS TABLE tweet_clusters
```

// Scala table generating function (TGF):
```
object KMeans {
  @Schema(spec = "x double, y double, cluster int")
  def apply(points: RDD[(Double, Double)]) = {
    ...
  }
}
```
Conclusion

Next challenge in big data will be complex and low-latency applications.

Spark offers a *unified* engine to tackle and combine these apps.

Best strength is the community: enjoy Spark Summit!