Apache Spark, Present and Future

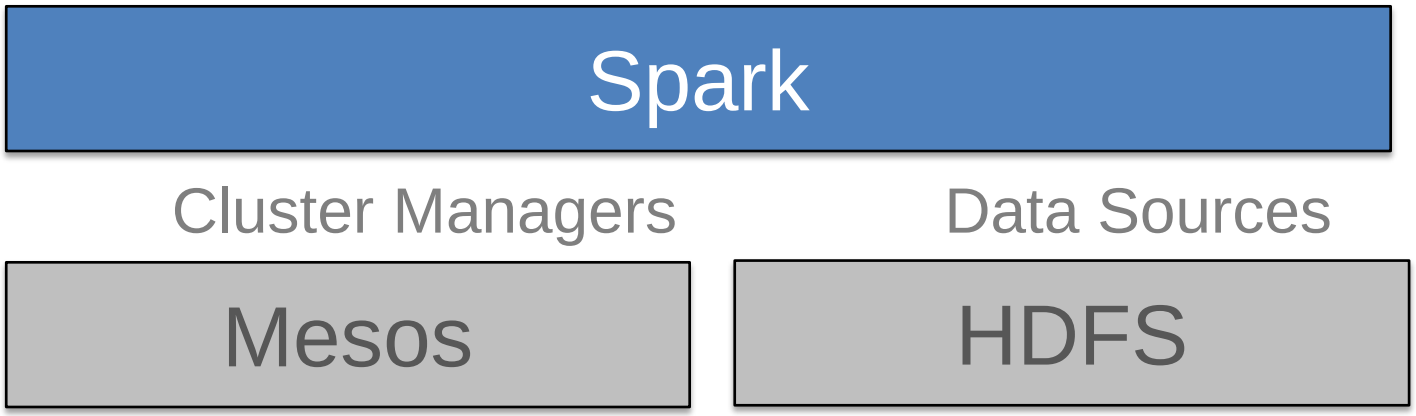
Ion Stoica
UC Berkeley

ISTC Retreat
September 3-5, 2014

Spark (circa 2009)

Original goals

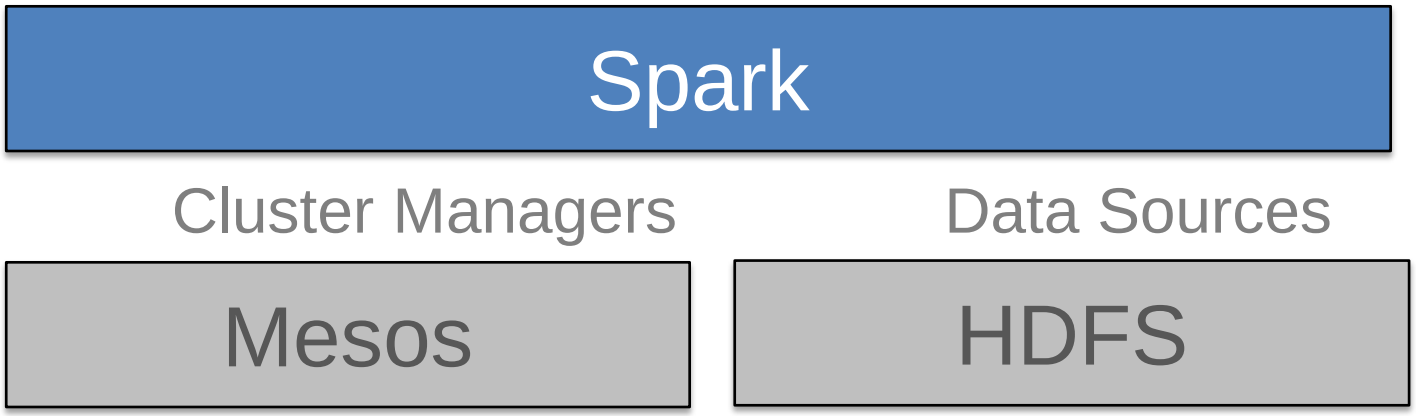
- Illustrate convenience to build a framework on Mesos
- Target workloads not well supported by Hadoop: **iterative** and **interactive** computation



Spark (circa 2009)

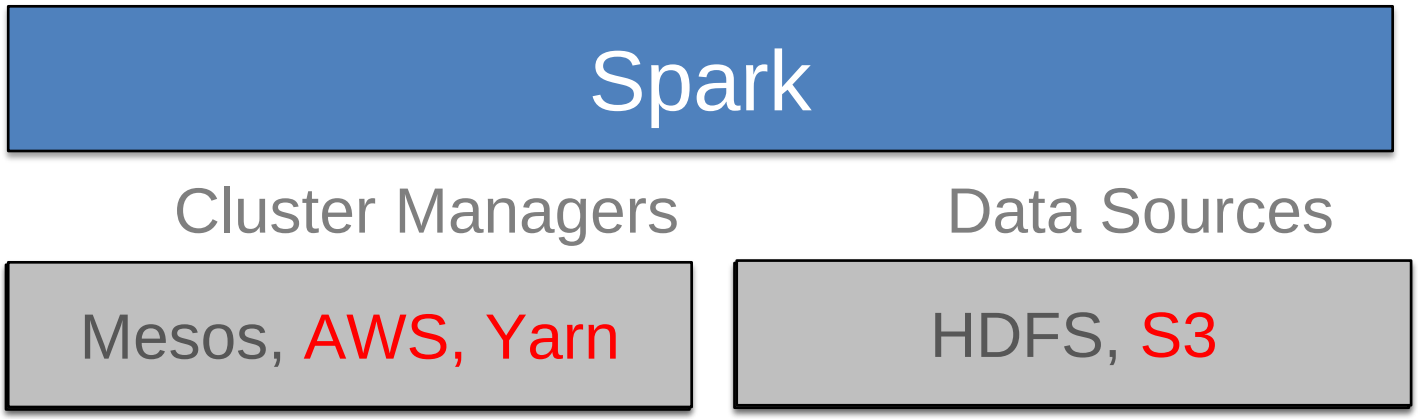
Properties

- Resilient (in-memory) data structures, RDDs
- General computation model, e.g., support DAGs
- Compact code: ~2,000 LoC



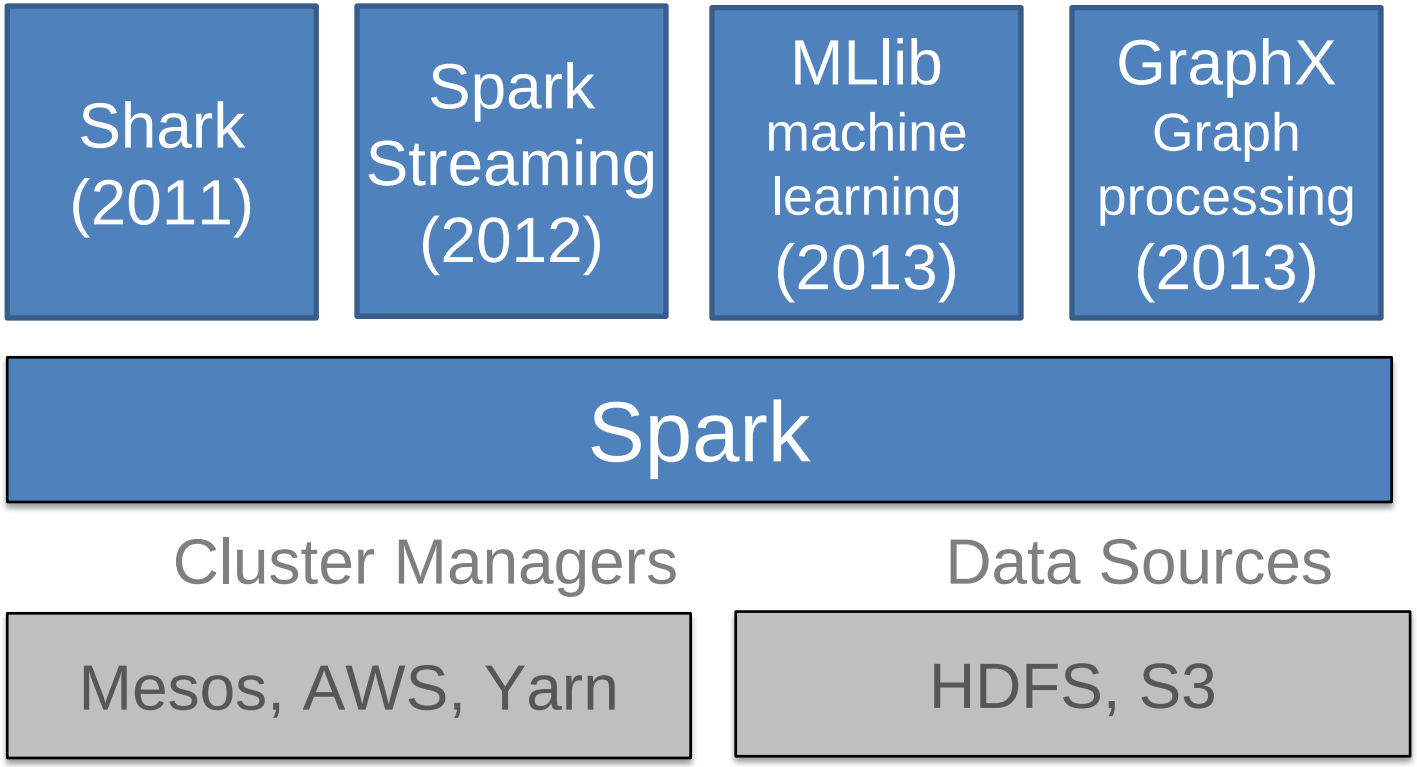
Spark (2009-13)

Shortly, started to enhance data & cluster manager support, and…

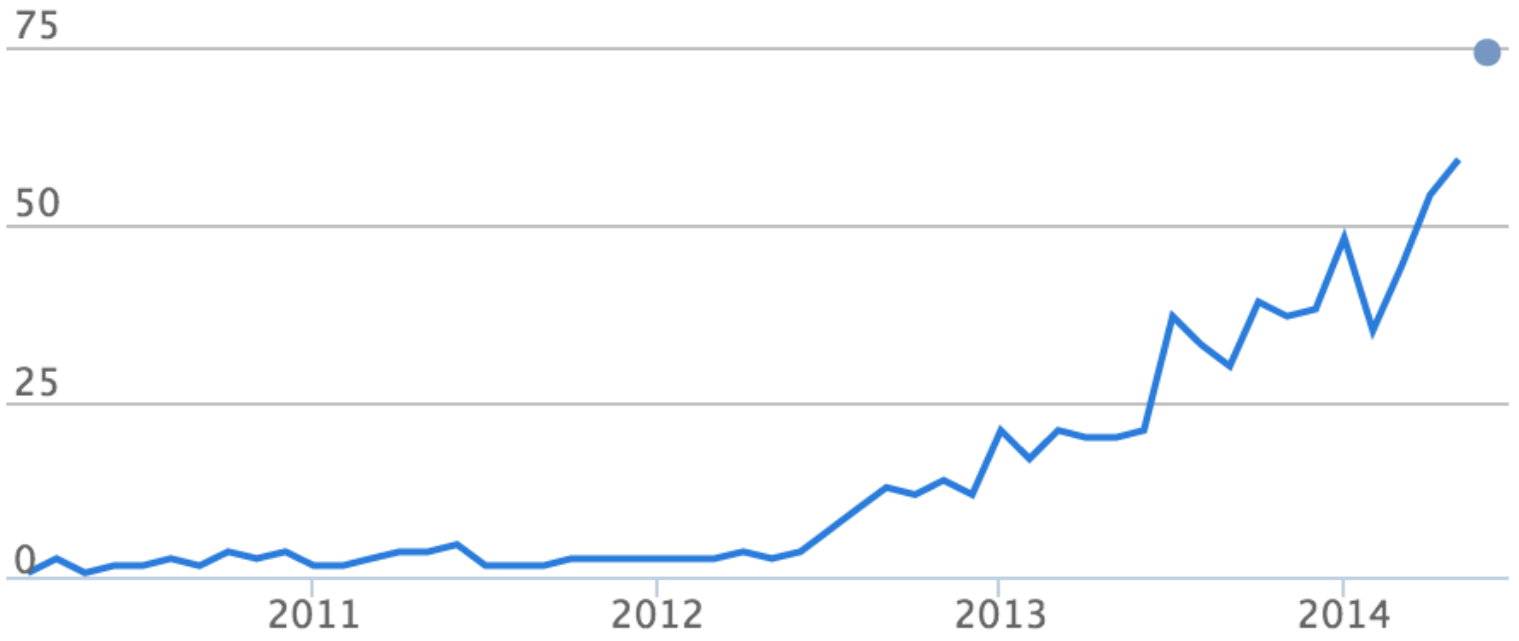


Stack (circa 2013)

… add new components on top



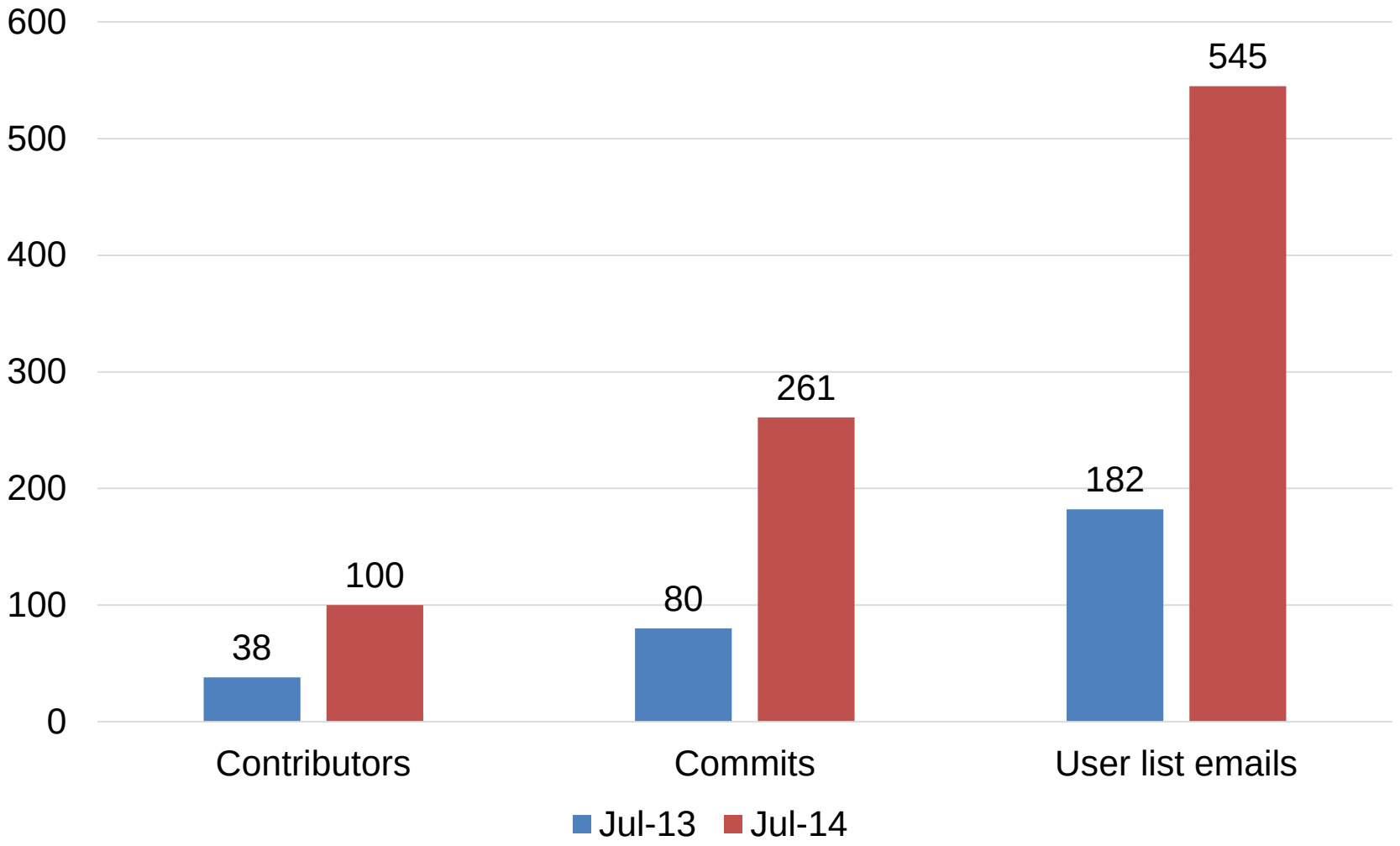
Rapid Growth



Contributors per month to Spark

source: ohloh.net

Rapid Growth



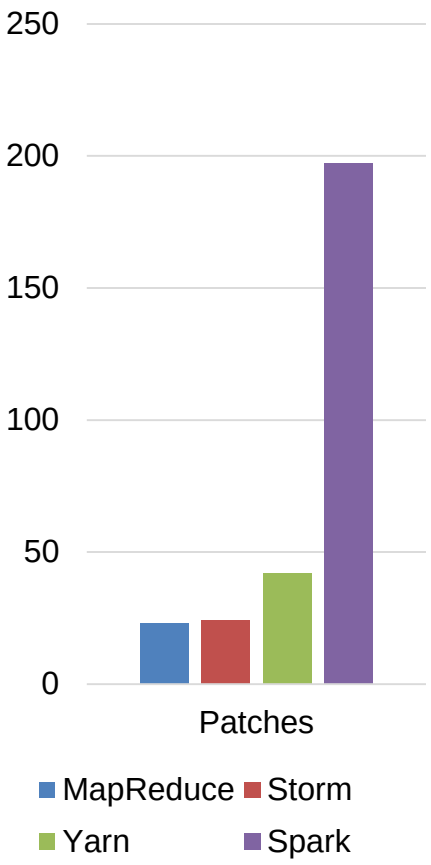
Developer Friendly Release Process

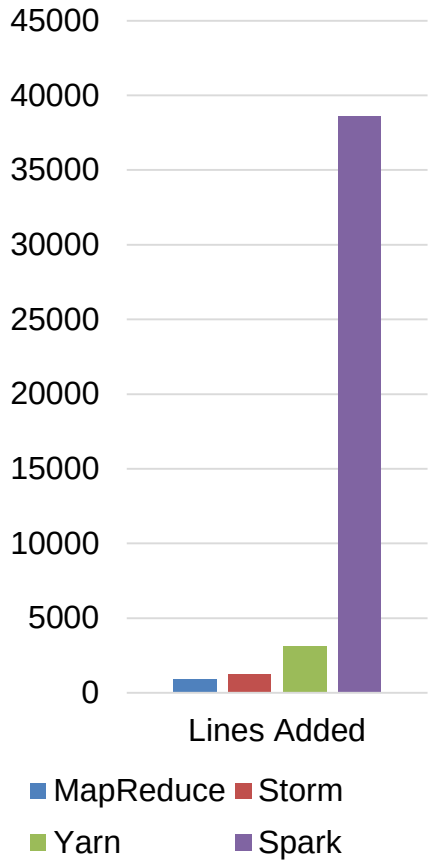
Release cycle: ~3 months, time scooped

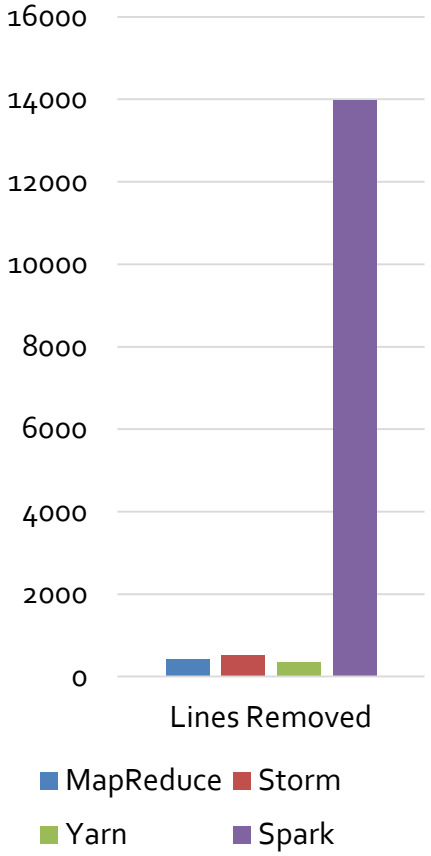
- 2 months feature development
- 1 month QA

Maintain old branches with bug fixes

Rapid Development



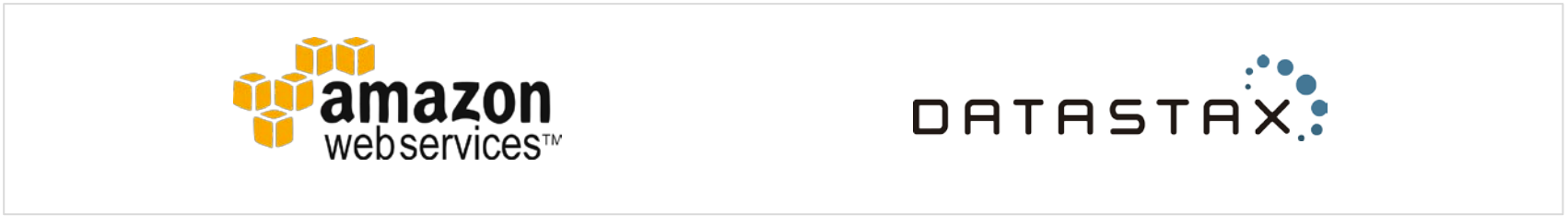




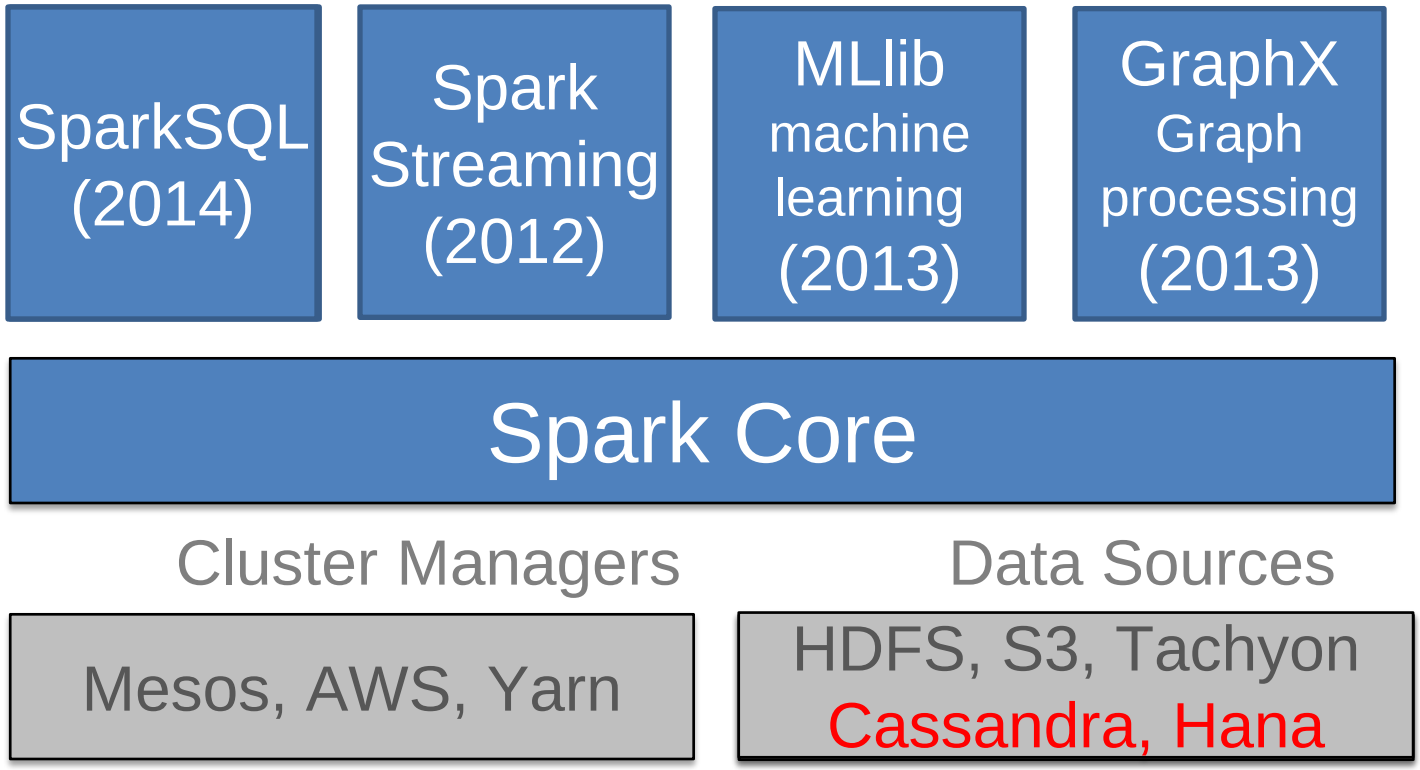
Wide Adoption

All major Hadoop distributions include Spark

Beyond Hadoop



The Spark Stack: Present



Spark 1.0

Well defined public API’s and well defined experimental API’s

Apps written against Spark API will be portable in new versions

Patches that break our API automatically fail build

Spark Vision:

Spark Vision: Become Big Data App Platform

Free Certification Process

Scripts for certifying Spark distributions

- Developed by community
- Open-source

Anyone will be able to certify any Spark distribution

The future of Spark is Libraries

Critical component of any successful app ecosystem

Packaged and distributed with Spark to provide full inter-operability

Lead by experts in respective fields, highly curated and integrated with Spark core API

The Spark Stack

Newer, focused on adding capabilities

More mature, focus on optimization and pluggability

Spark SQL

Support for SQL language and notion of typed schema RDDs

- Deep integration between Spark and other Spark comp.

Growing faster than any other component

Turning an RDD into a Relation

```
// Define the schema using a case class.
case class Person(name: String, age: Int)

// Create an RDD of Person objects, register it as a table.
val people =
  sc.textFile("examples/src/main/resources/people.txt")
      .map(_.split(","))
      .map(p => Person(p(0),  p(1).trim.toInt))

people.registerAsTable("people")
```

Querying using SQL

```
// SQL statements can be run directly on RDD's
val teenagers =
  sql("SELECT name FROM people
   WHERE age >= 13 AND age <= 19")

// The results of SQL queries are SchemaRDDs and support
// normal RDD operations.
val nameList = teenagers.map(t => "Name: " + t(0)).collect()

// Language integrated queries (ala LINQ)
val teenagers =
 people.where('age >= 10).where('age <= 19).select('name)
```

SparkSQL Status

1.0 was the first “preview” release

1.1 provides upgrade path for Shark

Replaced Shark in our benchmarks with 2-3X perf gains

Can perform optimizations with 10-100X less effort than Hive.

Focus Going Forward

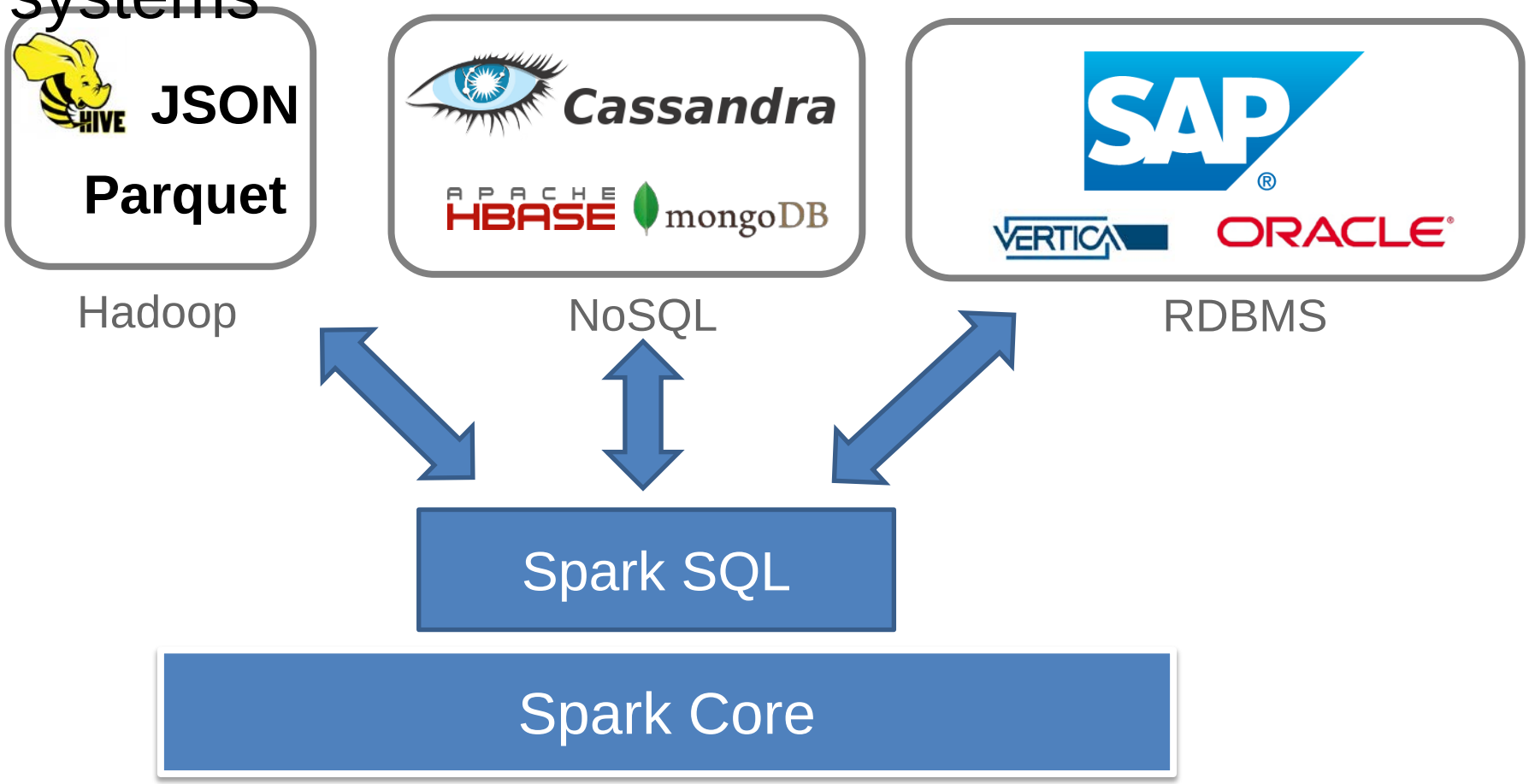
Optimizations, e.g., code gen, faster joins, etc

Language extensions (towards SQL92)

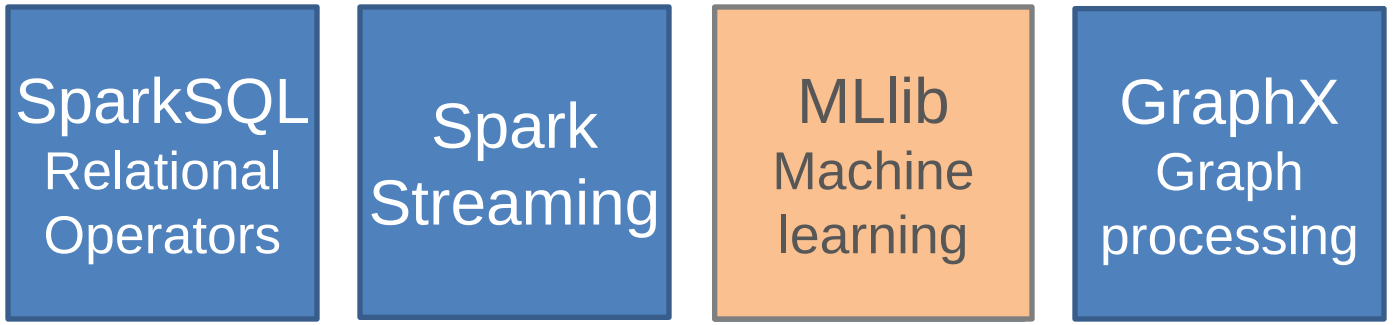
Improved integration (next slide)

Spark SQL and SchemaRDD

Will facilitate deeper integration with other systems



The Spark Stack



Newer, focused on adding capabilities

More mature, focus on optimization and pluggability

MLlib

Second fastest growing component

- Contributors: 40 (1.0) → 68 (1.1)

MLlib 1.0 has15+ algorithms

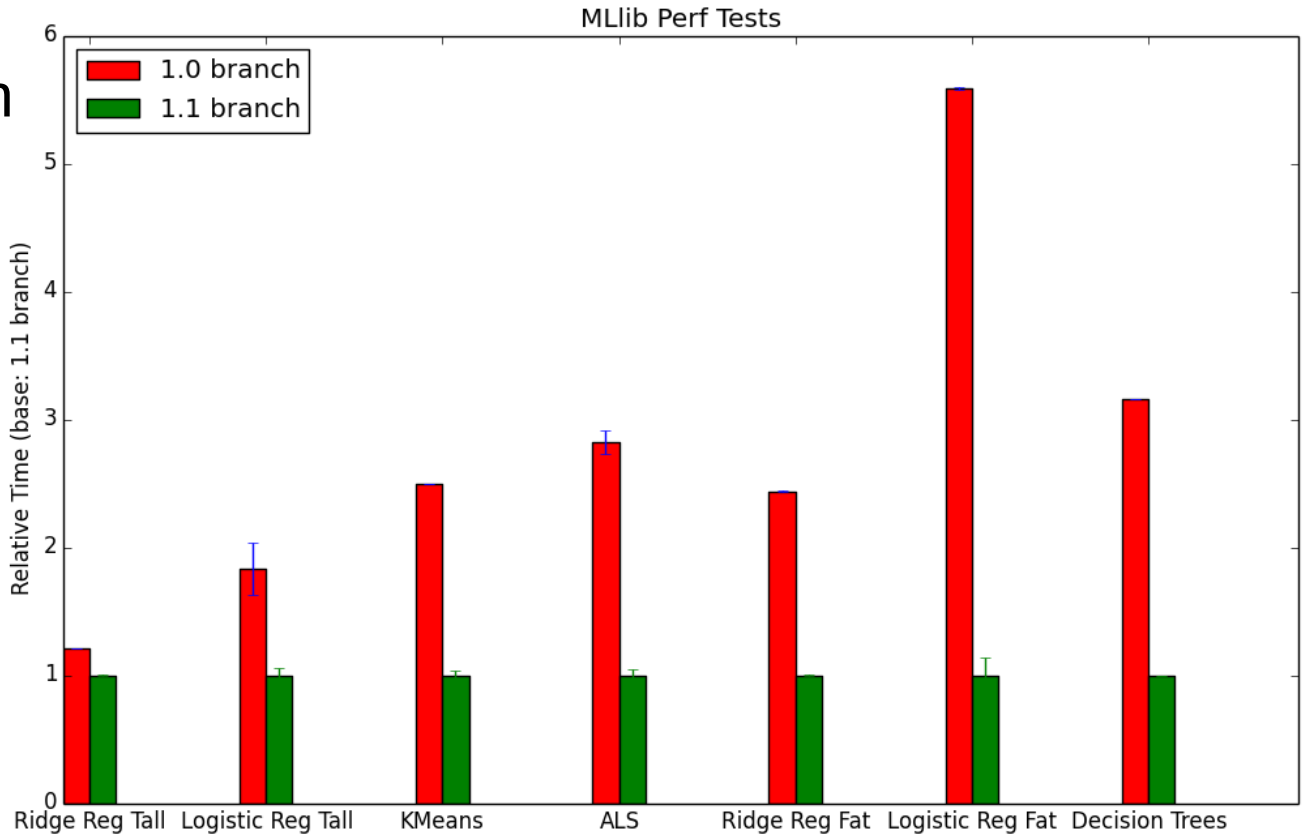
MLlib 1.1 has 25+ algorithms

- Traditional descriptive statistics: sampling, correlation, estimators, tests
- New learning algorithms: NMF, Sparse SVD, LDA…

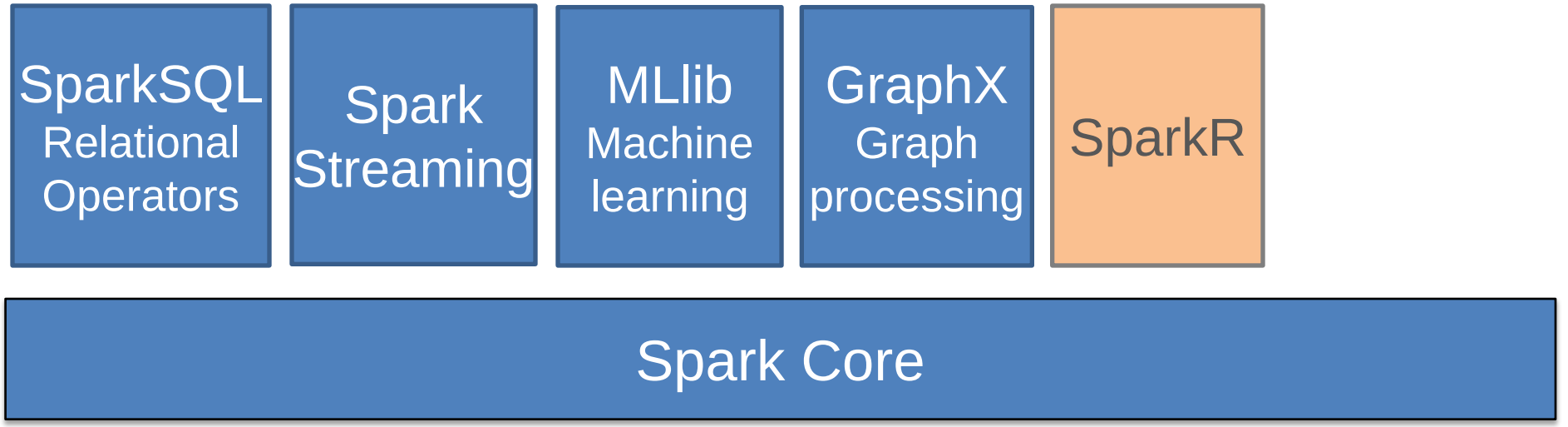
Performance (1.0 vs. 1.1)

2~3x speed-up on linear methods, clustering, and collaborative filtering

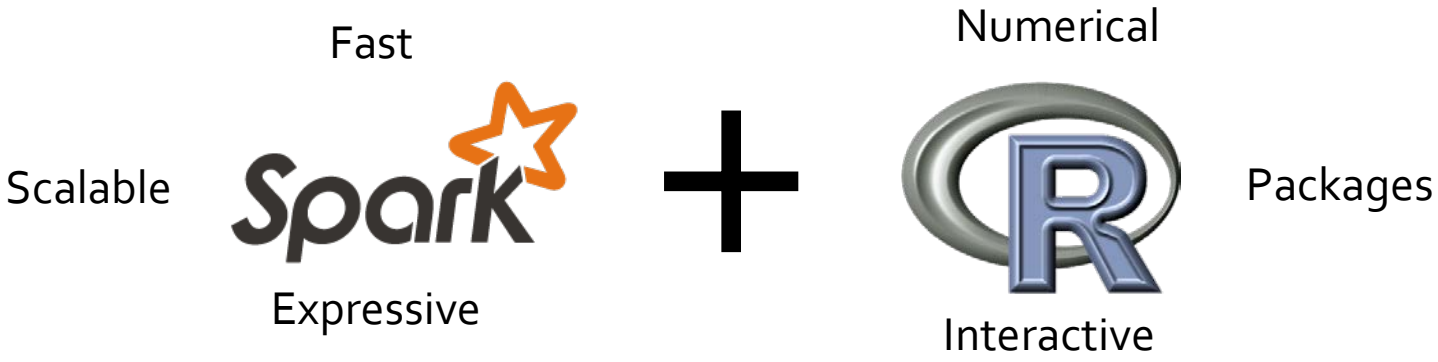
5x speed-up on decision trees



New Components



SparkR

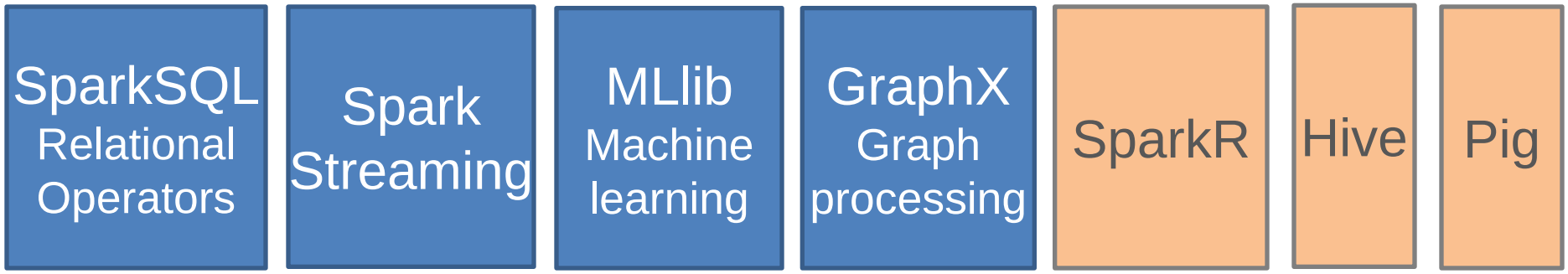


Make SparkR “production ready” (AMPLab, Alteryx and Databricks).

Integration with MLlib

Consolidating the data frame and RDD concepts

New Components

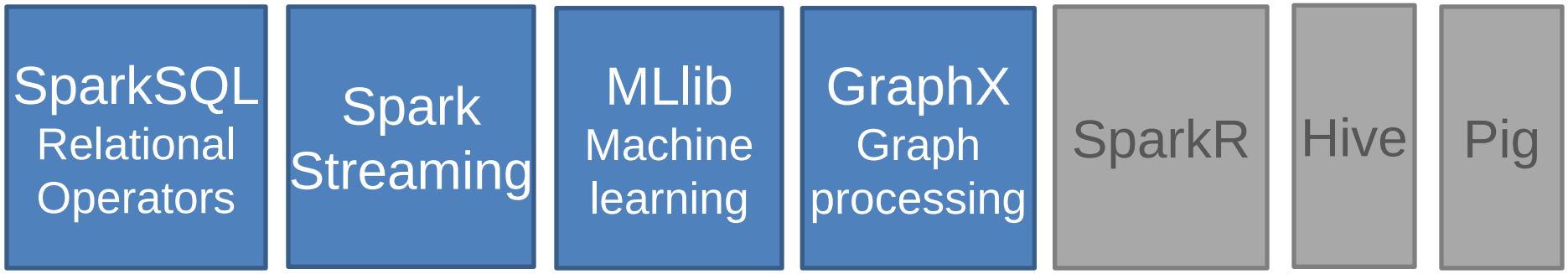


Newer, focused on adding capabilities

Spark Core

More mature, focus on optimization and pluggability

New Components



Newer, focused on adding capabilities

Spark Core

More mature, focus on optimization and pluggability

Trends

Hardware

- Memory prices still continue to fall and capacity to increase
- SSD's becoming widely deployed

Storage

- Tachyon and other memory-optimized storage systems

Spark Core

Define internal APIs to allow extensions/innovations

Internal storage API

- Support for SSD's
- Shared in-memory storage systems such as Tachyon

Spark shuffle API

- Sort based shuffle
- Pipeline shuffle

Longer Term: 1.2+

GraphX: optimizations and API stability

Streaming: new data sources, Python API

SparkR productization

Core: Elastic scaling on YARN, user-defined metrics and counters

Package repository for 3rd party libraries

Long Term

Support for incremental updates

Extend RDD abstraction:

- » Indexed RDDs
- » Queries on compressed data
- » …

BlinkDB (approx query processing) on top of SparkSQL

- » Support for on-line aggregation

…

Summary

Spark now is real: widely deployed

Expect to continue substantial growth

Focus is on libraries and improving core internals for future innovation

- Release process and cadence provides users with stable releases despite fast growth