goto; conference

Click ‘Rate Session’
to rate session
and ask questions.

goto; conference

Spark Beyond Shuffling
(Why there isn't magic)

Holden Karau
@holdenkarau

goto; conference

Please

Remember to rate this session

Thank you!

follow us @gotochgo

Who am I?

My name is Holden Karau

Prefered pronouns are she/her

I'm a Principal Software Engineer at IBM's Spark Technology Center

Apache Spark committer

previously Alpine, Databricks, Google, Foursquare & Amazon

co-author of High Performance Spark & Learning Spark (+ more)

Twitter: @holdenkarau

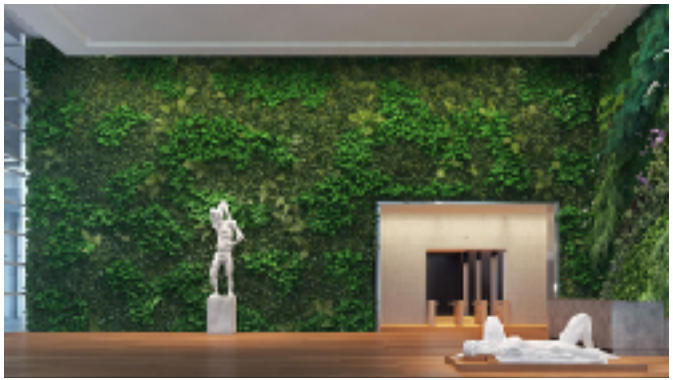
Slideshare http://www.slideshare.net/hkarau

Linkedin https://www.linkedin.com/in/holdenkarau

Github https://github.com/holdenk

Related Spark Videos http://bit.ly/holdenSparkVideos

IBM Spark Technology Center



Founded in 2015.

Location:

Physical: 505 Howard St., San Francisco CA

Web: http://spark.tc Twitter: @apachespark_tc

Mission:

Contribute intellectual and technical capital to the Apache Spark community.

Make the core technology enterprise- and cloud-ready.

Build data science skills to drive intelligence into business applications — http://bigdatauniversity.com

Key statistics:

About 50 developers, co-located with 25 IBM designers.

Major contributions to Apache Spark http://jiras.spark.tc

Apache SystemML is now an Apache Incubator project.

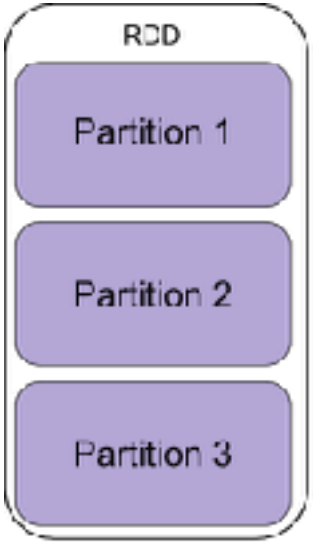
Founding member of UC Berkeley AMPLab and RISE Lab

Member of R Consortium and Scala Center

Who do I think you all are?

Nice people*
Possibly some knowledge of Apache Spark?
Interested in understanding a bit about how Spark works?
Want to make your spark jobs more efficient
Familiar-ish with Scala or Java or Python

What is Spark?



General purpose distributed system

With a really nice API including Python :)

Apache project (one of the most active)

Must faster than Hadoop Map/Reduce

Good when too big for a single machine

Built on top of two abstractions for distributed data: RDDs & Datasets

Why people come to Spark:

Well this MapReduce job is going to take 16 hours - how long could it take to learn Spark?

Why people come to Spark:

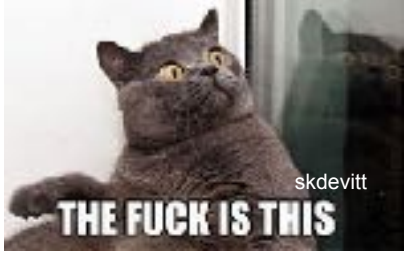
My DataFrame won't fit in memory on my cluster anymore, let alone my MacBook Pro :(Maybe this Spark business will solve that...

Plus a little magic :)

What is the “magic” of Spark?

DAG / “query plan” is the root of much of it
    Think the person behind the curtain
Optimizer to combine steps
Resiliency: recover from failures rather than protecting from failures.
“In-memory” + “spill-to-disk”
Functional programming to build the DAG for “free”
Select operations without deserialization

Spark specific terms in this talk



RDD

Resilient Distributed Dataset - Like a distributed collection. Supports many of the same operations as Seq's in Scala but automatically distributed and fault tolerant. Lazily evaluated, and handles faults by recompute. Any* Java or Kyro serializable object.

DataFrame

Spark DataFrame - not a Pandas or R DataFrame. Distributed, supports a limited set of operations. Columnar structured, runtime schema information only. Limited* data types.

Dataset

Compile time typed version of DataFrame (templated)

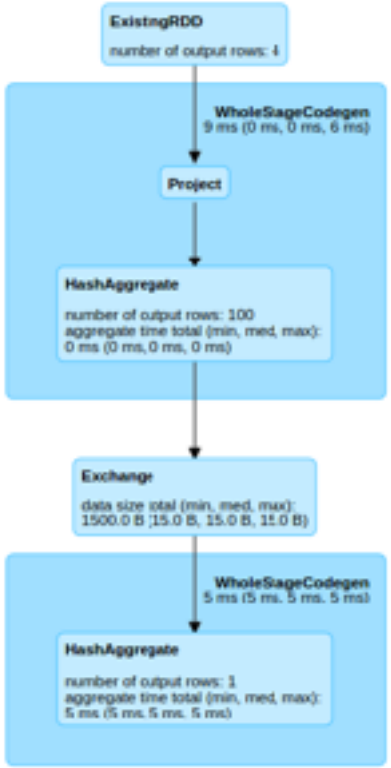
Magic part #1: the DAG


Photo by Dan G

In Spark most of our work is done by transformations
    Things like map
Transformations return new RDDs or DataFrames representing this data
The RDD or DataFrame however doesn't really "exist"
RDD & DataFrames are really just "plans" of how to make the data show up if we force Spark's hand
tl;dr - the data doesn't exist until it "has" to

The DAG

The query plan



Word count (in python)


```
lines = sc.textFile(src)
words = lines.flatMap(lambda x: x.split(" "))
word_count =
  (words.map(lambda x: (x, 1))
        .reduceByKey(lambda x, y: x+y))
word_count.saveAsTextFile(“output”)
```

Word count (in python)


```
lines = sc.textFile(src)
words = lines.flatMap(lambda x: x.split(" "))
word_count =
  (words.map(lambda x: (x, 1))
        .reduceByKey(lambda x, y: x+y))
```

No data is read or processed until after this line

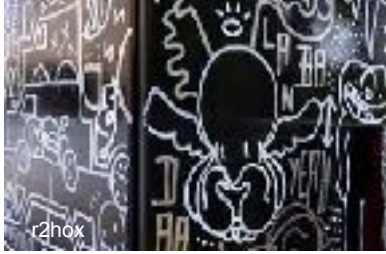
```
word_count.saveAsTextFile("output")
```

This is an “action” which forces spark to evaluate the RDD

How the DAG magic is awesome:

Pipelining (can put maps, filter, flatMap together)
Can do interesting optimizations by delaying work
We use the DAG to recompute on failure
(writing data out to 3 disks on different machines is so last season)
Or the DAG puts the R is Resilient RDD, except DAG doesn't have an R :(

And where it reaches its limits:



- It doesn’t have a whole program view
  - Can only see up to the action, can’t see into the next one
  - So we have to help Spark out and cache
- Combining the transformations together makes it hard to know what failed
- It can only see the pieces it understands
  - can see two maps but can’t tell what each map is doing

Your data is magically distributed

At some point the RDD or DataFrame is forced to exist
Then Spark splits up the data on a bunch of different machines
The default is based on a combination of
If the data needs to be joined (or similar) Spark does a “shuffle” so it knows which keys are where
Partioners in Spark are deterministic on key input (e.g. for any given key they must always send to the same partition)

Key-skew to the anti-rescue… :(

Keys aren’t evenly distributed

- Sales by zip code, or records by city, etc.

groupByKey will explode (but it's pretty easy to break)

We can have really unbalanced partitions

- If we have enough key skew sortByKey could even fail
- Stragglers (uneven sharding can make some tasks take much longer)

Can just the shuffle cause problems?

Sorting by key can put all of the records in the same partition

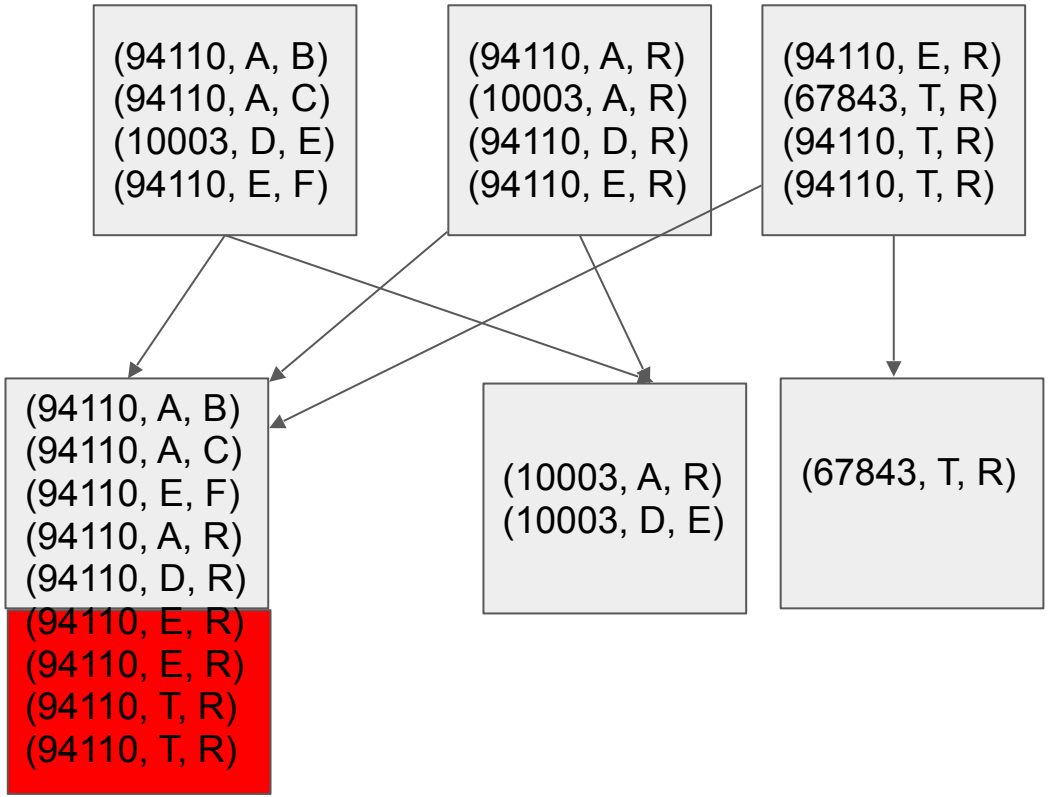
We can run into partition size limits (around 2GB)

Or just get bad performance

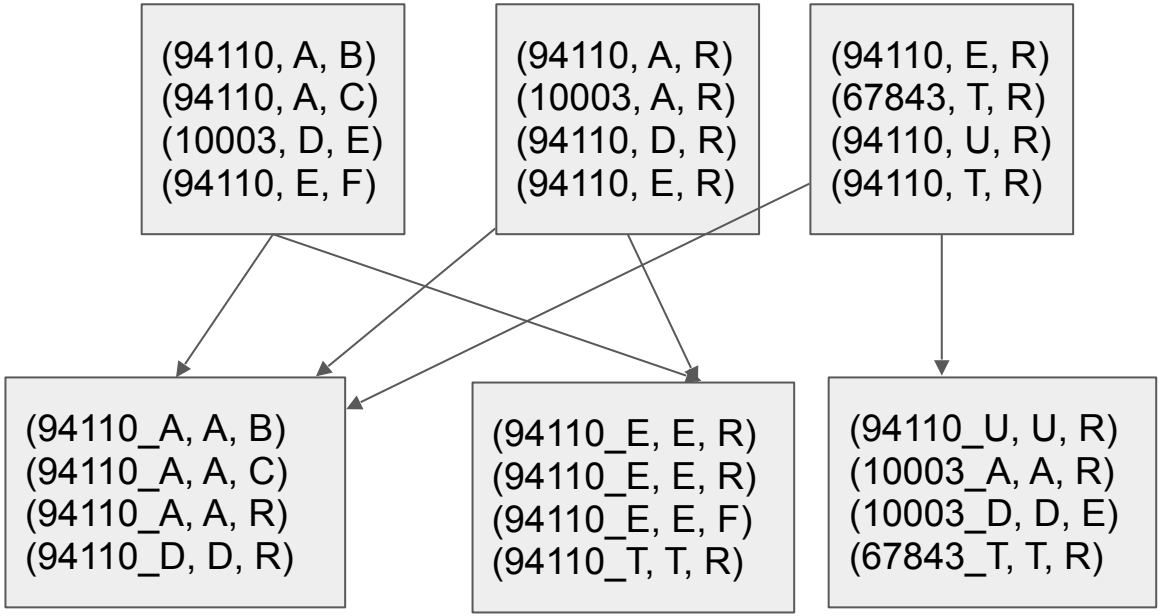
(94110, A, B) (94110, A, C) (10003, D, E) (94110, E, F)	(94110, A, R) (10003, A, R) (94110, D, R) (94110, E, R)	(94110, E, R) (67843, T, R) (94110, T, R) (94110, T, R)

So we can handle data like the above we can add some “junk” to our key

Shuffle explosions :(



Happy Shuffle (100% less explosions)



key-skew + black boxes == more sadness

There is a worse way to do WordCount

We can use the seemingly safe thing called groupByKey

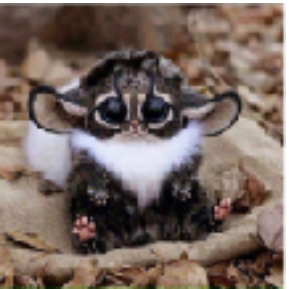
Then compute the sum

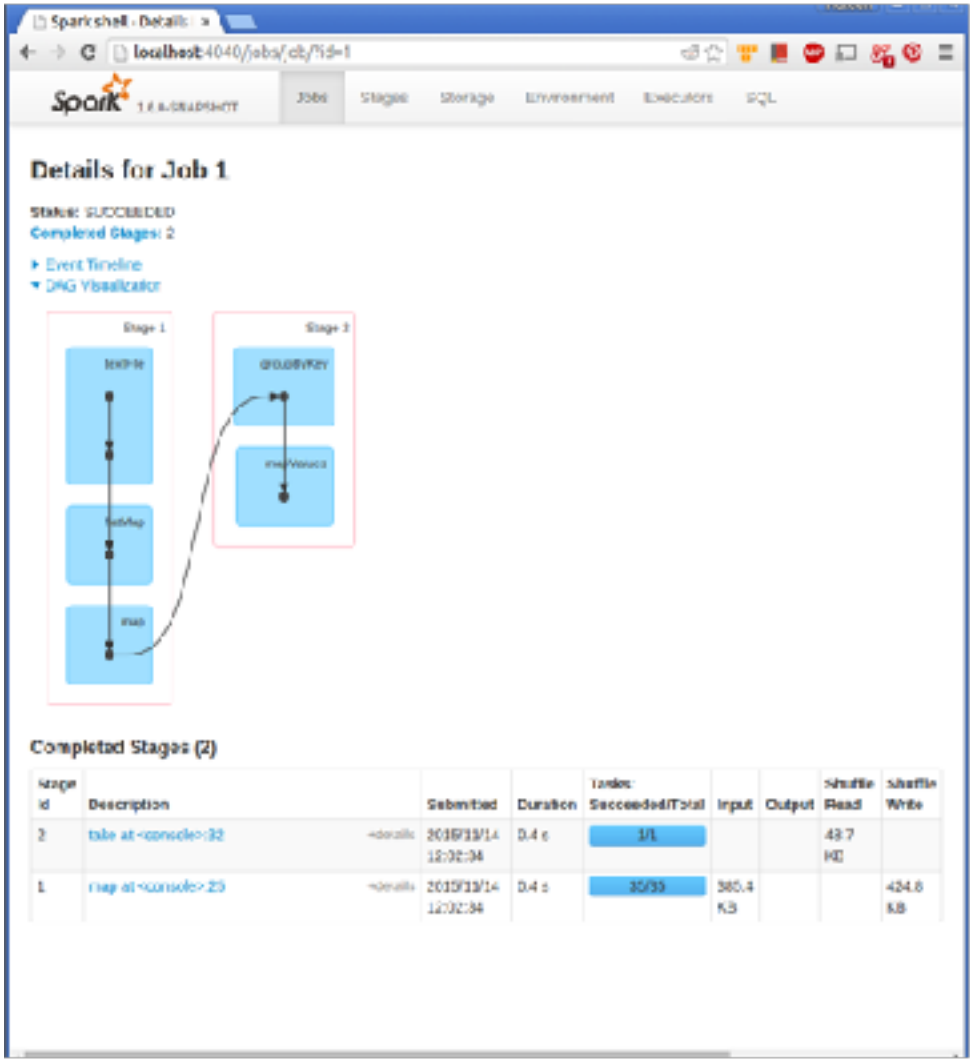
But since it’s on a slide of “more sadness” we know where this is going...

Bad word count :(


```
words = rdd.flatMap(lambda x: x.split(" "))
wordPairs = words.map(lambda w: (w, 1))
grouped = wordPairs.groupByKey()
counted_words = grouped.mapValues(lambda counts: sum(counts))
counted_words.saveAsTextFile("boop")
```

GroupByKey




Spark shell - Details
localhost:4040/jobs/job/?id=1
Spark
Jobs
Stages
Storage
Environment
Executors
SQL
Details for Job 1
Status: SUCCEEDED
Completed Stages: 2
Event Timeline
DAG Visualization
Stage 1
Stage 2
textFile
groupByKey
mapValues
map
Completed Stages (2)
Stage Id
Description
Submitted
Duration
Tasks: Succeeded/Total
Input
Output
Shuffle Read
Shuffle Write
2
take at <console>:32
2015/11/14 12:02:34
0.4 s
1/1
43.7 KB
1
map at <console>:23
2015/11/14 12:02:34
0.4 s
35/35
380.4 KB
424.8 KB

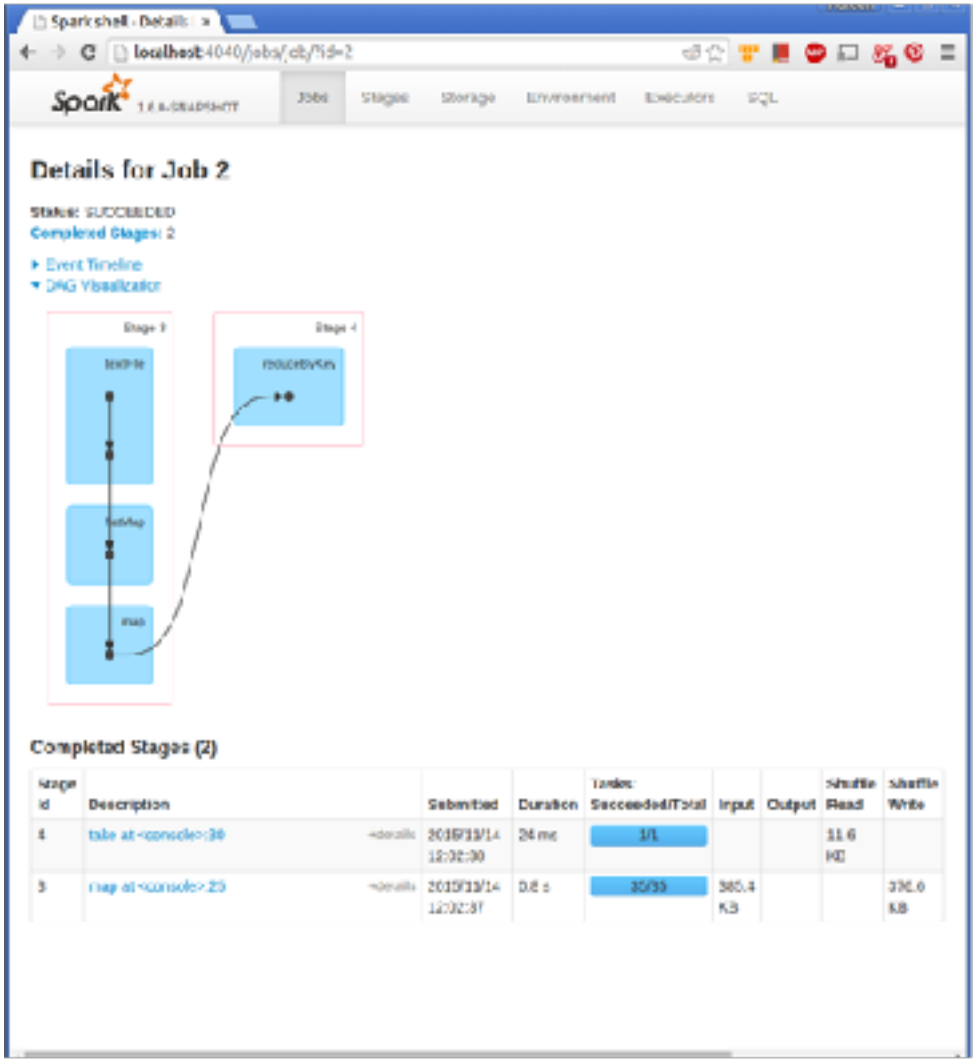
So what did we do instead?

- reduceByKey
  - Works when the types are the same (e.g. in our summing version)
- aggregateByKey
  - Doesn’t require the types to be the same (e.g. computing stats model or similar)

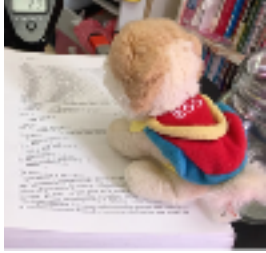
Allows Spark to pipeline the reduction & skip making the list

We also got a map-side reduction (note the difference in shuffled read)

reduceByKey



Mini “fix”: Datasets (aka DataFrames)



Still super powerful

- Still allow arbitrary lambdas
- But give you more options to “help” the optimized
- groupBy returns a GroupedDataStructure and offers special aggregates
- Selects can push filters down for us*
- Etc.

Using Datasets to mix functional & relational style

```
val ds: Dataset[RawPanda] = ...
val happiness = ds.filter($"happy" === true).
  select($"attributes"(0).as[Double]).
  reduce((x, y) => x + y)
```

So what was that?


```
ds.filter($"happy" === true).
  select($"attributes"(0).as[Double]).
  reduce((x, y) => x + y)
```

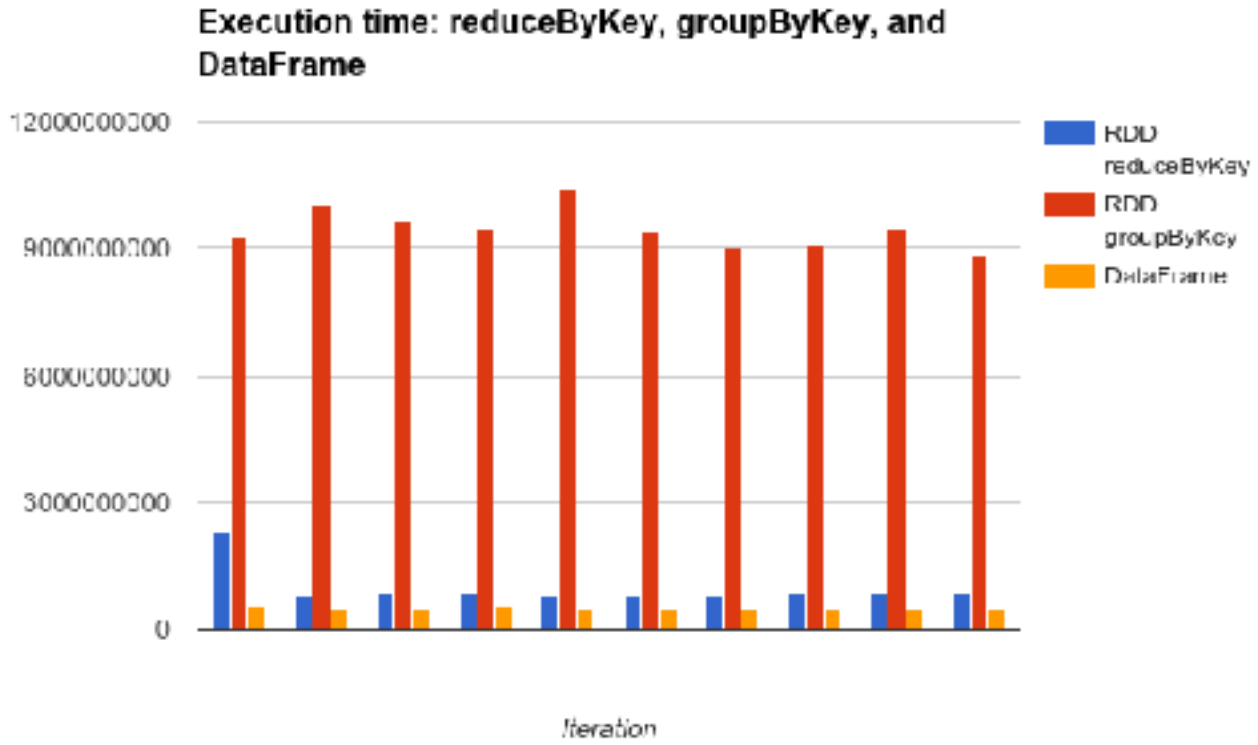
Traditional functional reduction:
arbitrary scala code :)

A typed query (specifies the return type). Without the as[] will return a DataFrame (Dataset[Row])

And functional style maps:

```
/**
 * Functional map + Dataset, sums the positive attributes for the
pandas
 */
def funMap(ds: Dataset[RawPanda]): Dataset[Double] = {
   ds.map{rp => rp.attributes.filter(_ > 0).sum}
}
```

How much faster can it be?



Our final bit of magic today (Python & co):

Spark is written in Scala (runs on the JVM)

Users want to work in their favourite language

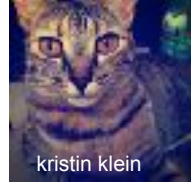
Python, R, C#, etc. all need a way to talk to the JVM

How expensive could IPC be anyways? :P

A quick detour into PySpark's internals

Photo by Bill Ward

Spark in Scala, how does PySpark work?



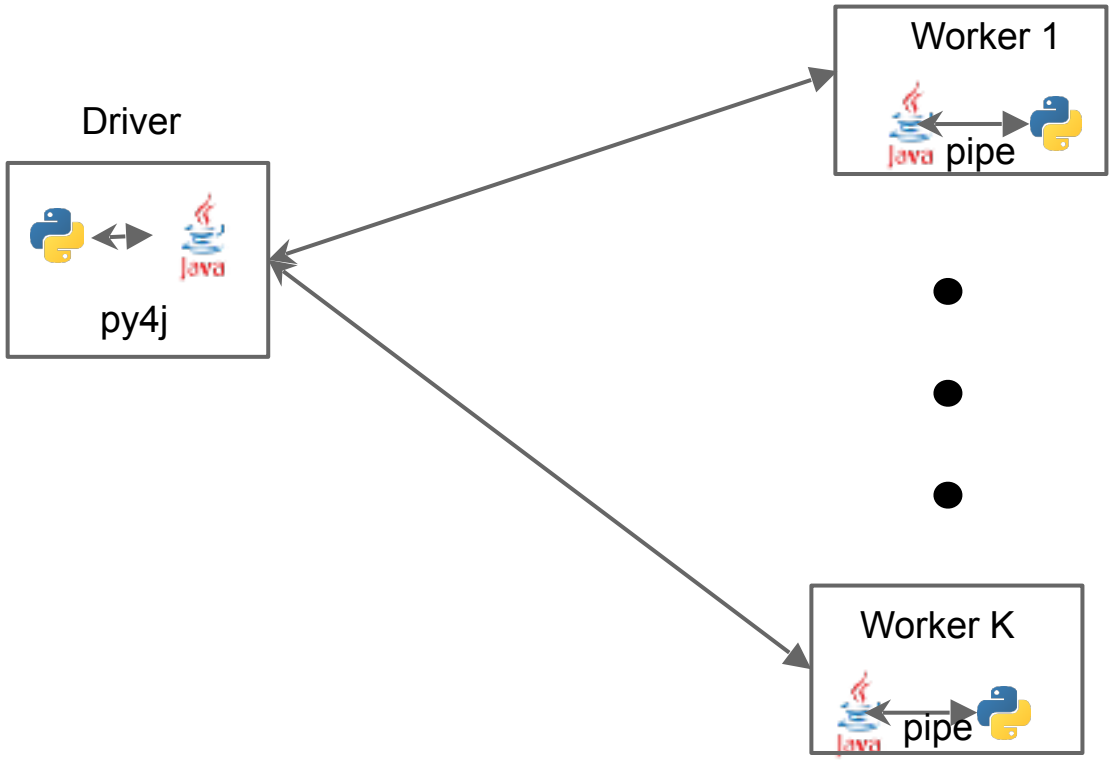
Py4J + pickling + magic

- This can be kind of slow sometimes

RDDs are generally RDDs of pickled objects

Spark SQL (and DataFrames) avoid some of this

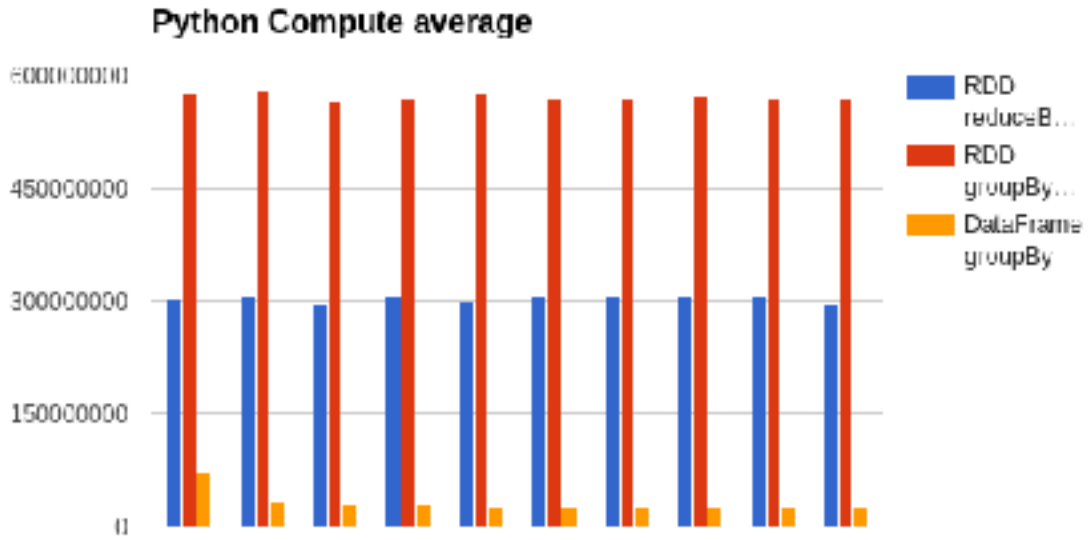
So what does that look like?



So how does this break?

- Data from Spark worker serialized and piped to Python worker
  - Multiple iterator-to-iterator transformations are still pipelined :)
- Double serialization cost makes everything more expensive
- Python worker startup takes a bit of extra time
- Python memory isn't controlled by the JVM - easy to go over container limits if deploying on YARN or similar
- etc.

And back to magic with Dataframes:



*Note: do not compare absolute #s with previous graph - different dataset sizes because I forgot to write it down when I made the first one.

Spark Videos

Apache Spark Youtube Channel

My Spark videos on YouTube -
http://bit.ly/holdenSparkVideos

Spark Summit 2014 training

Paco's Introduction to Apache Spark

PLZ test (Spark Testing Resources)

Libraries

Scala: spark-testing-base (scalacheck & unit) sscheck (scalacheck)
example-spark (unit)

Java: spark-testing-base (unit)

Python: spark-testing-base (unittest2), pyspark.test (pytest)

Strata San Jose Talk (up on YouTube)

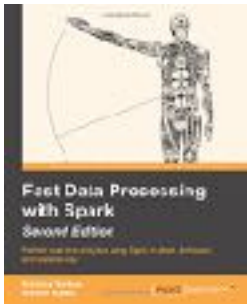
Blog posts

Unit Testing Spark with Java by Jesse Anderson

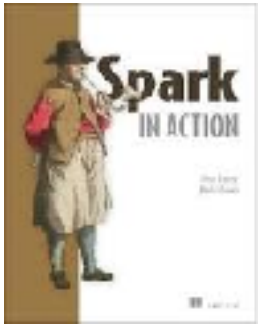
Making Apache Spark Testing Easy with Spark Testing Base

Unit testing Apache Spark with py.test

Learning Spark

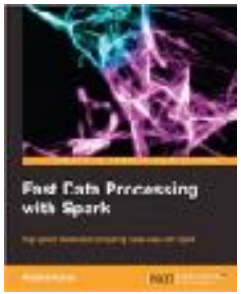


Fast Data Processing with Spark (2nd edition)

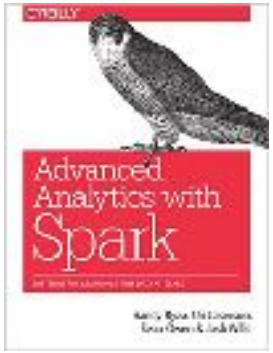


Spark in Action

Learning PySpark



Fast Data Processing with Spark (Out of Date)



Advanced Analytics with Spark

Coming soon:
High Performance Spark

High Performance Spark (soon!)

Available in “Early Release”* (All Chapters):
  Buy from O’Reilly - http://bit.ly/highPerfSpark
Get notified when updated & finished:
  http://www.highperformancespark.com
  https://twitter.com/highperfspark
Currently in QC2 edits
  eg: are you sure you meant to link to this?

* Early Release means extra mistakes, but also a chance to help us make a more awesome book.

And some upcoming talks:

April

- Doing office hours tomorrow

May

- LX Scala (Lisbon, Portugal)
- PyData BCN
- Strata London
- 3rd Data Science Summit Europe in Israel

June

- Scala Days CPH
- Spark Summit West (SF)
- Scala Swarm (Porto, Portugal)

goto; conference

Please

Remember to rate this session

Thank you!

follow us @gotochgo