INTRODUCING RDD'S

Frank Kane

RDD

- Resilient
- Distributed
- Dataset

The SparkContext

- Created by your driver program
- Is responsible for making RDD's resilient and distributed!
- Creates RDD's
- The Spark shell creates a "sc" object for you

Creating RDD's

- nums = parallelize([1, 2, 3, 4])
- sc.textFile("file:///c:/users/frank/gobs-o-text.txt")
  - *or s3n:// , hdfs://*
- hiveCtx = HiveContext(sc) rows = hiveCtx.sql("SELECT name, age FROM users")
- Can also create from:
  - *JDBC*
  - *Cassandra*
  - *HBase*
  - *Elastisearch*
  - *JSON, CSV, sequence files, object files, various compressed formats*

Transforming RDD's

- map
- flatmap
- filter
- distinct
- sample
- union, intersection, subtract, cartesian

map example

- rdd = sc.parallelize([1, 2, 3, 4])
- squaredRDD = rdd.map(lambda x: x*x)

- This yields 1, 4, 9, 16

What's that lambda thing?

Many RDD methods accept a *function* as a parameter

```
rdd.map(lambda x: x*x)
```

Is the same thing as

```
def squareIt(x):
    return x*x

rdd.map(squareIt)
```

There, you now understand functional programming.

RDD actions

- collect
- count
- countByValue
- take
- top
- reduce
- … and more ...

Lazy evaluation

- Nothing actually happens in your driver program until an action is called!