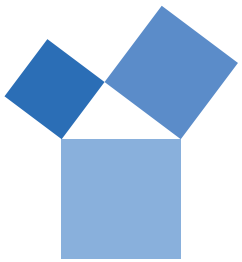
Spark at Euclid

Dave Strauss, PhD

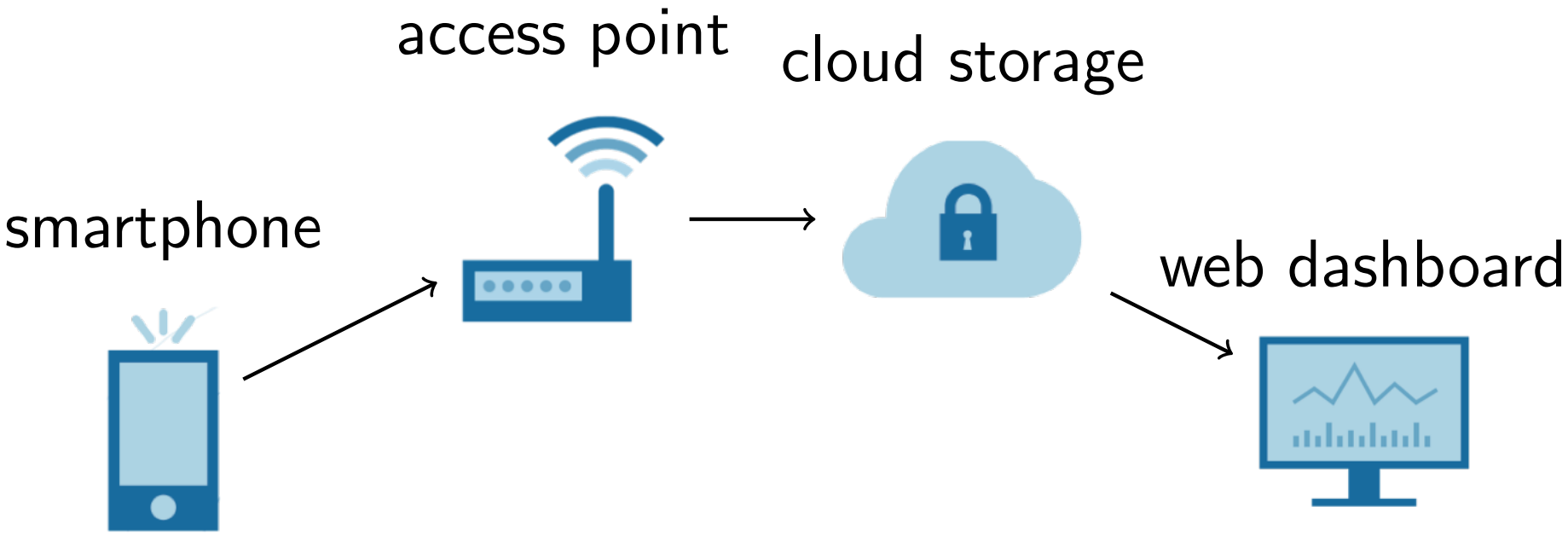
July 1, 2014



euclid

who is Euclid Analytics?

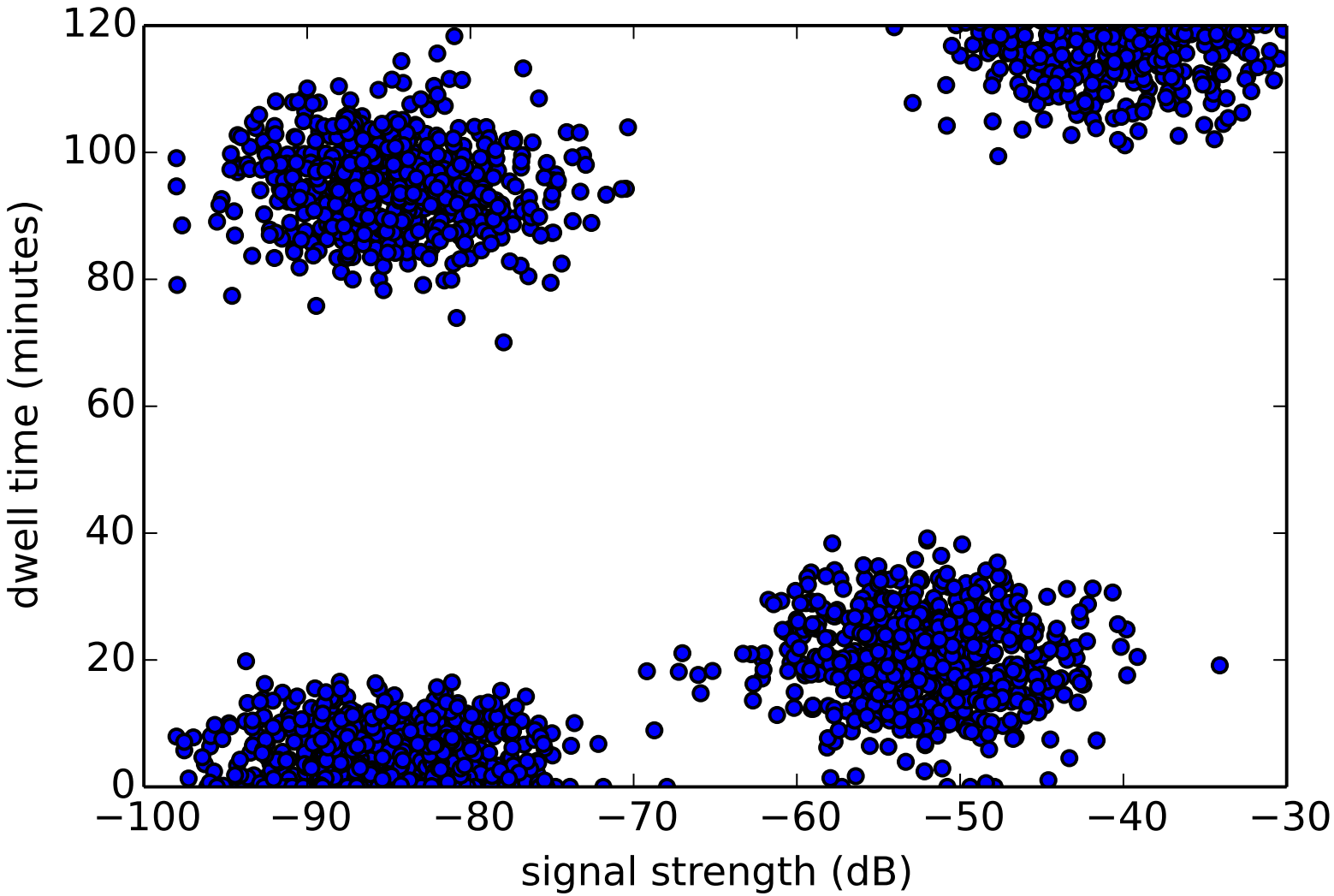
- quantify and measure retail customer behavior
- assign unique random id to all wifi enabled devices
- predict shopper duration, repeat visitors, etc.



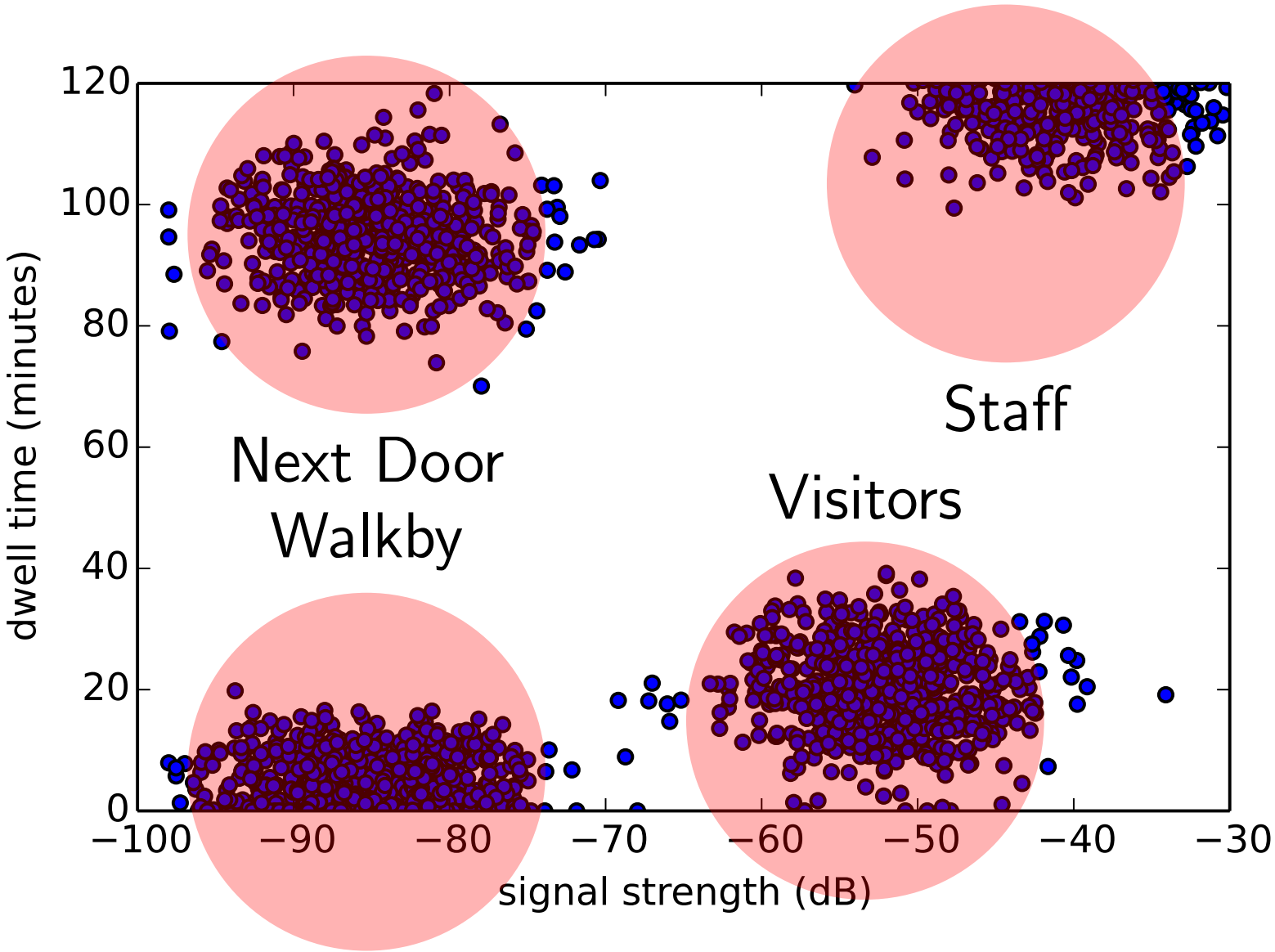
data processing

- process logfiles from wifi access points
- obtain a small amount of information about devices
  - unique id
  - time
  - signal strength
- search for patterns that correspond to user behavior
- monitor those trends

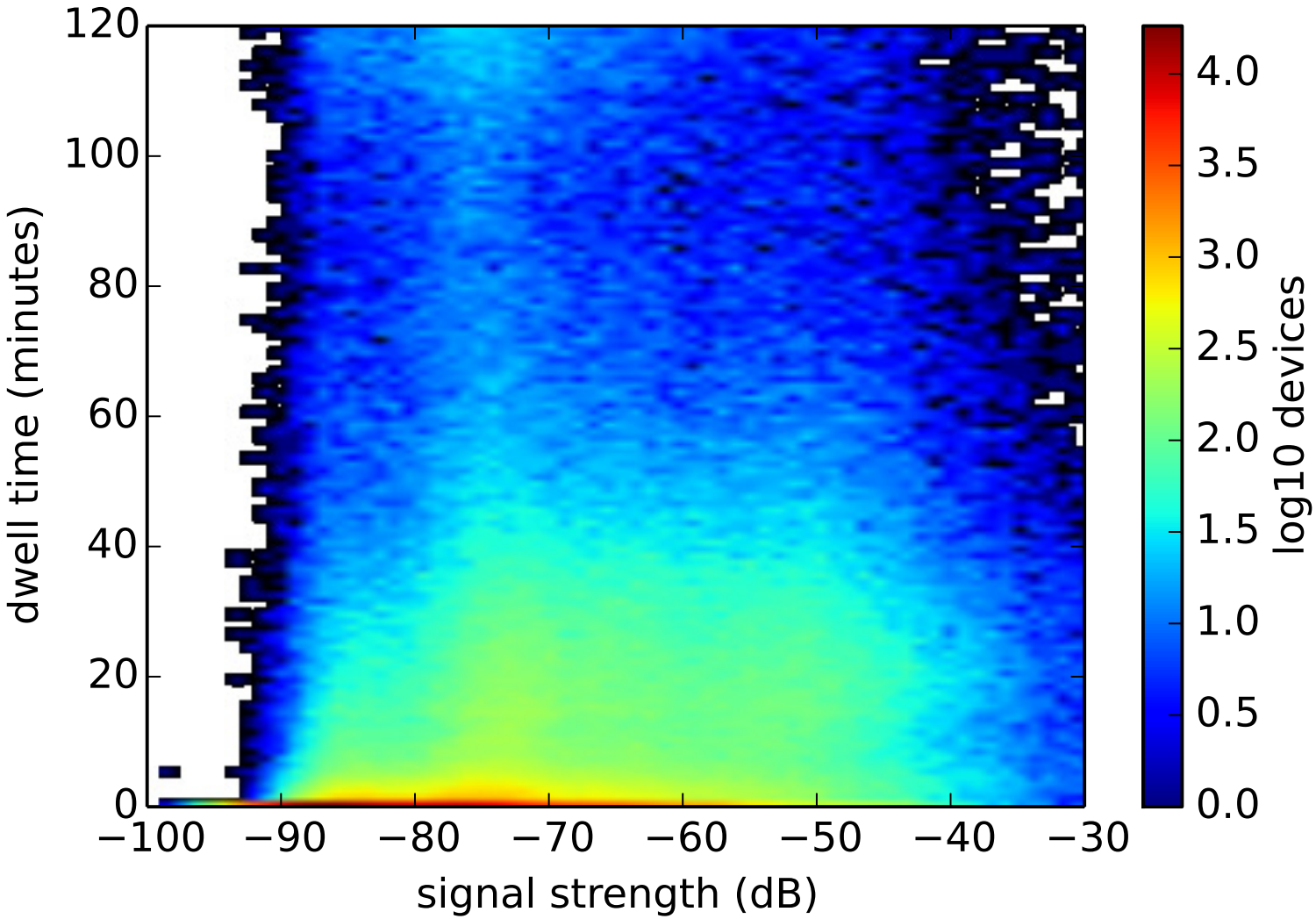
in a perfect world



in a perfect world



what our data really look like



why we use Spark

- code integration
- functional programming in scala
- scalable machine learning
- extensible
- iterative algorithms, complex data flows

challenges

- launching clusters regularly and reliably
- updating and applying models
- distributed optimization problems
- adoption and migration

python context

- created SparkCluster class for managing AWS clusters
- use python context management to manage launch and shutdown
- provide method to execute remote jobs

python context

- created SparkCluster class for managing AWS clusters
- use python context management to manage launch and shutdown
- provide method to execute remote jobs

```
with sparkClusterManager.cluster("production") as spc:
    spc.execute("com.euclidanalytics.foo",arguments)
    uploadDataToDatabases()
```

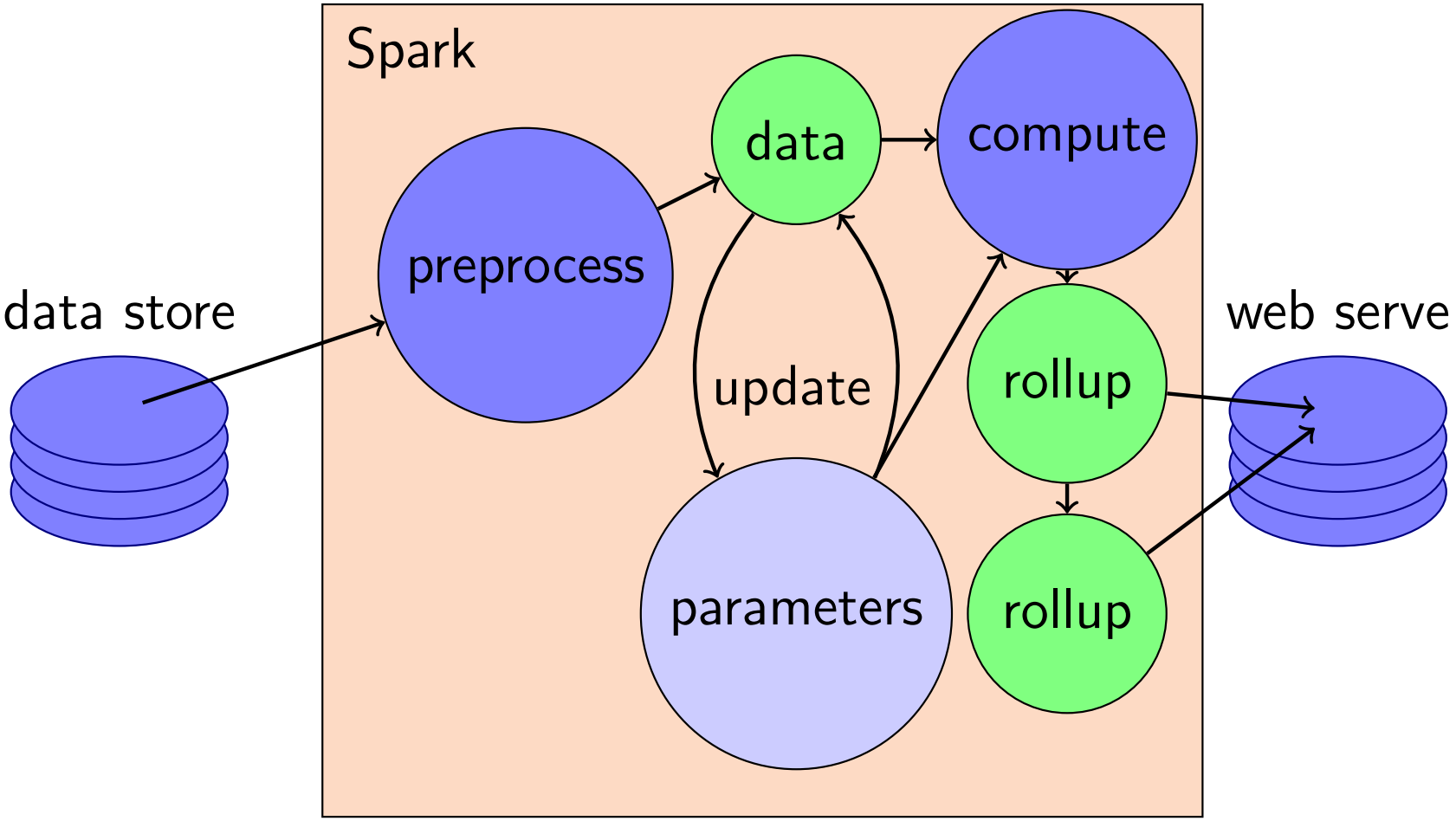
python context

- created SparkCluster class for managing AWS clusters
- use python context management to manage launch and shutdown
- provide method to execute remote jobs

```
with sparkClusterManager.cluster("production") as spc:
    spc.execute("com.euclidanalytics.foo",arguments)
    uploadDataToDatabases()
```

- leaves authentication for config files
- should anything go wrong, the context will automatically terminate the cluster, saving resources
- provide `KEEP_ALIVE` flag

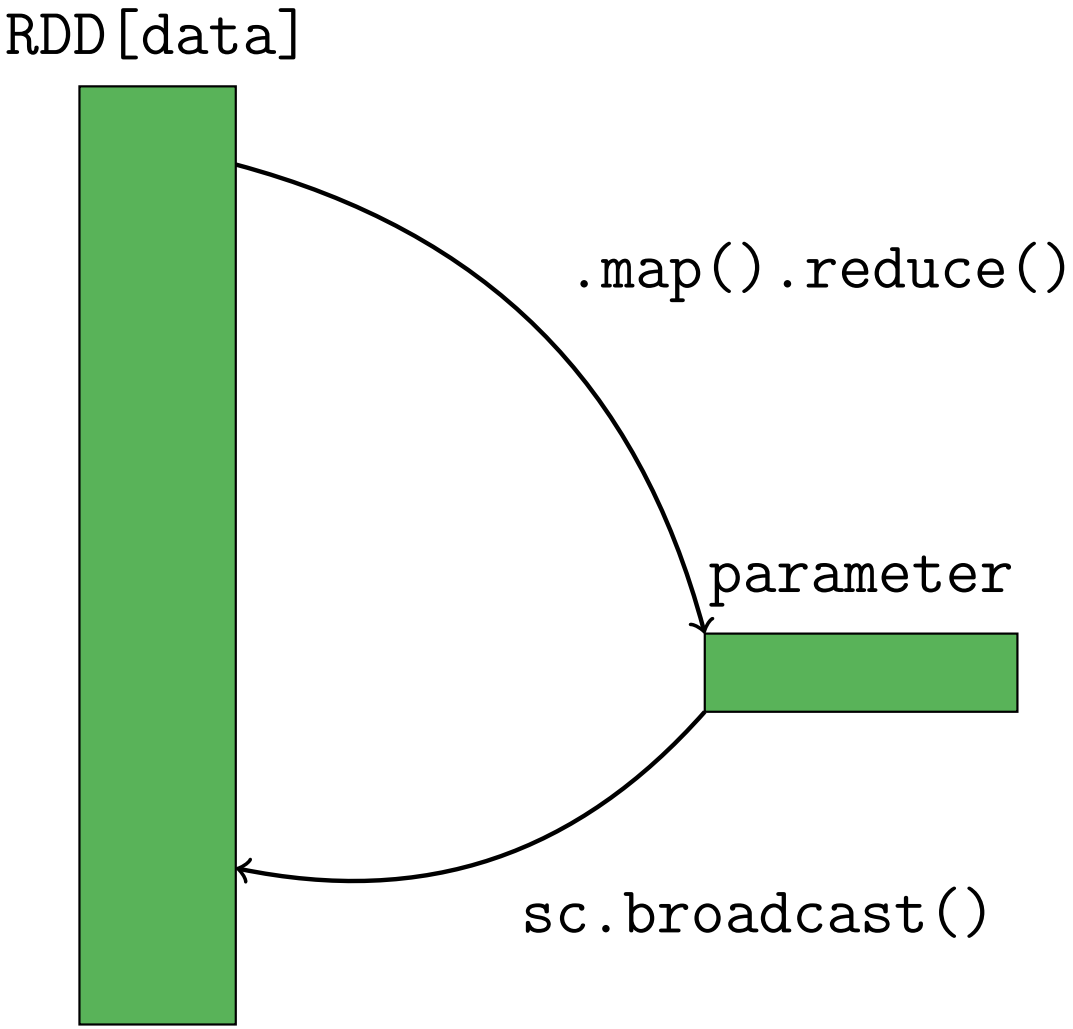
update and apply



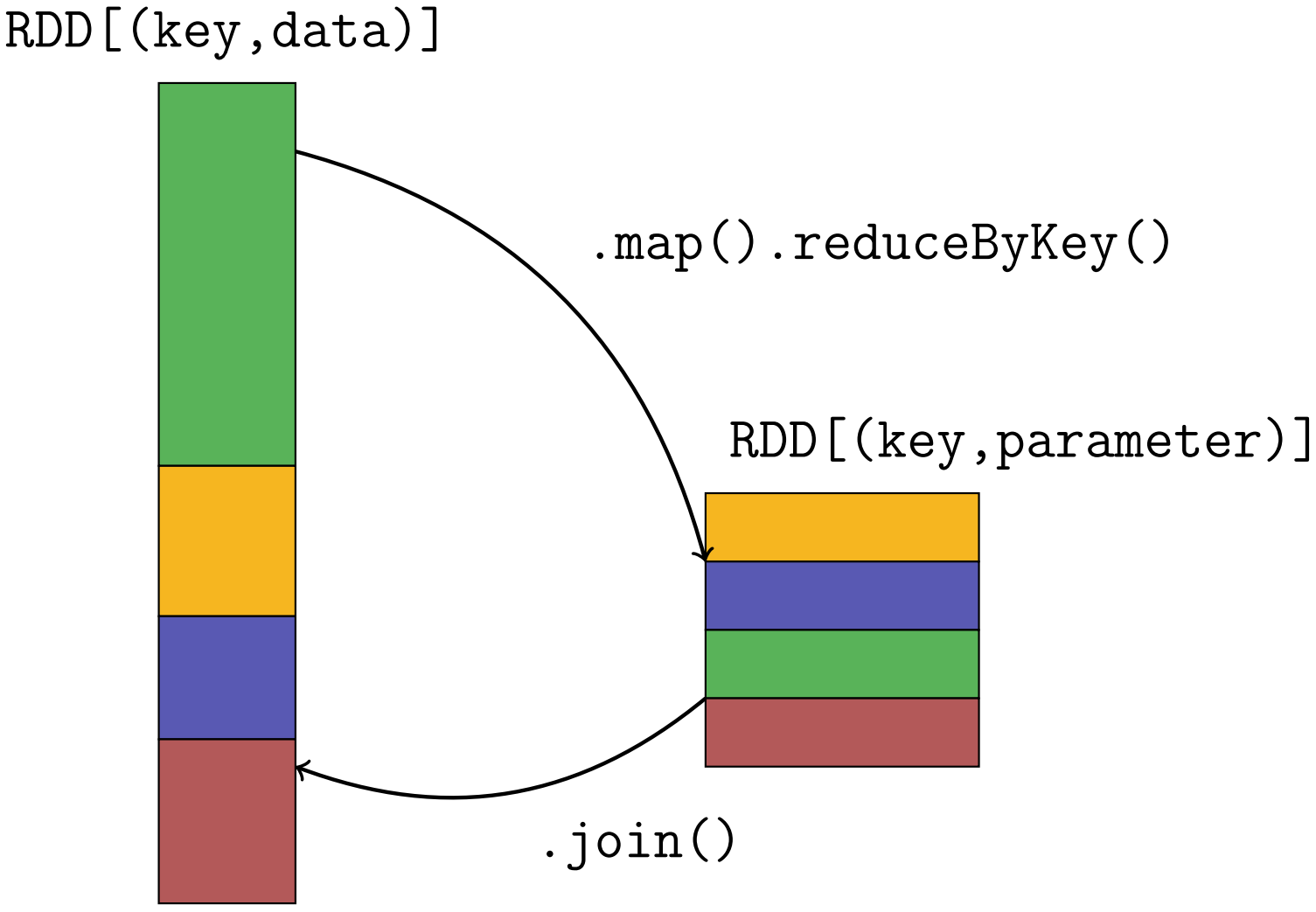
binary parameter search

- search to find optimal parameter in tree model
- test for maximum changes in conditional entropy
- do parameter search for each sensor
- touch all of our data across thousands of sensors
- distributed binary search
- more work necessary to optimize through shuffle steps

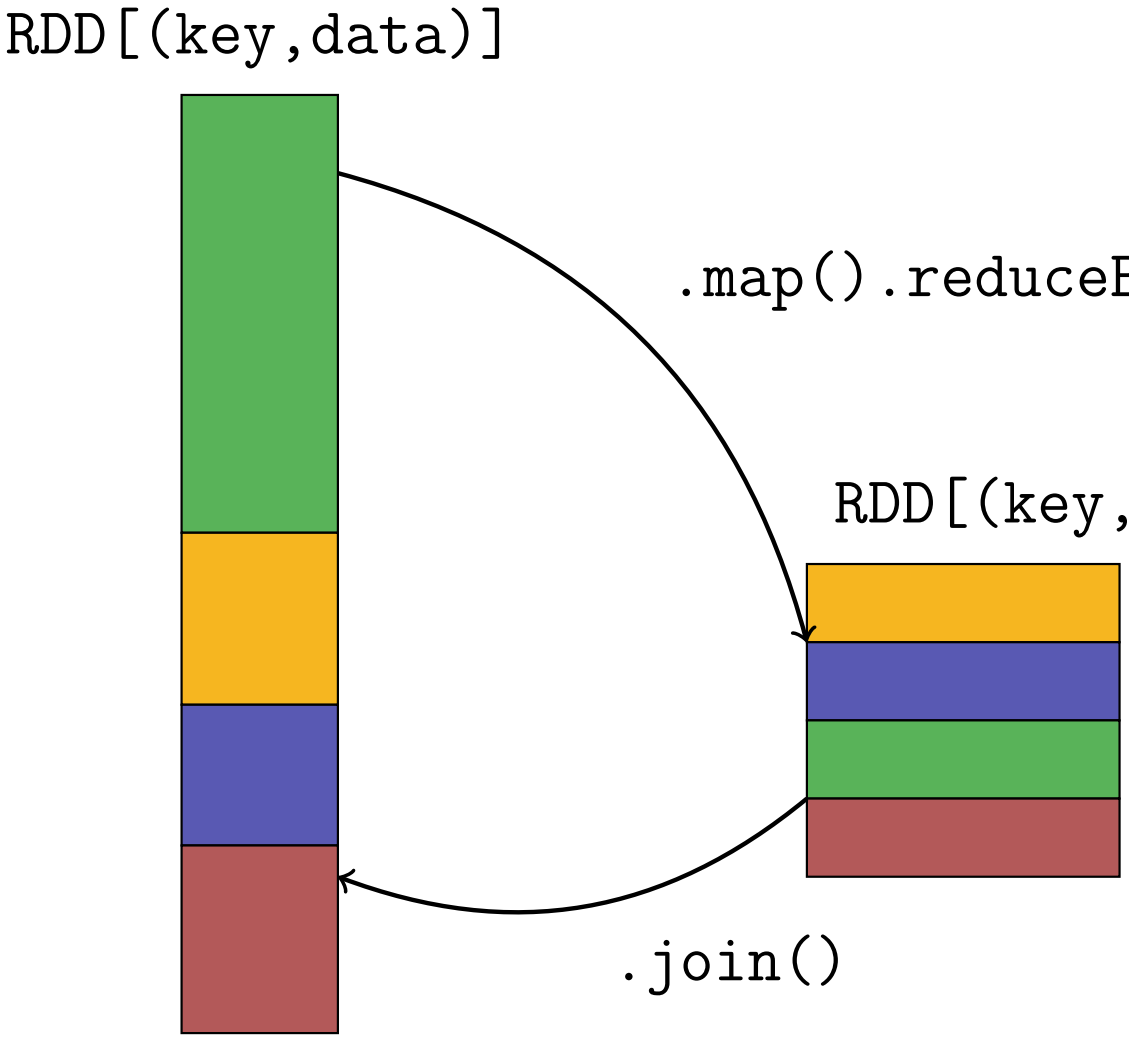
large learning



distributed learning



distributed learning



a `for` loop does weird things

recursive search

```
case class modelParam(low : Double, high: Double)
val data : RDD[record] = preprocess(rawData)
def binarySearch(params: RDD[modelParam], level: Int)
 : RDD[modelParam] = {
  level match {
    case x if (x==0) => params
    case _ => {
      val updated = model.compute(data, params)
       .reduceByKey((a,b)=>aggregate(a,b))
       .map(makeDecision)
      updated.checkpoint()
      binarySearch(updated, level-1)
    }
  }
}
val result = binarySearch(initialParams, 10)
```

evolution of Spark at Euclid

- pig scripts
- introduced AWS redshift for simple models
- migrate to Spark, Scala
- run nightly job and a whole host of ETL using Spark
- looking forward to streaming

questions

- Contact dstrauss@euclidanalytics.com for more