



the INTERNET of EVERYWHERE

how The Weather Company scales
Robbie Strickland



the INTERNET of EVERYWHERE

how The Weather Company scales
Robbie Strickland



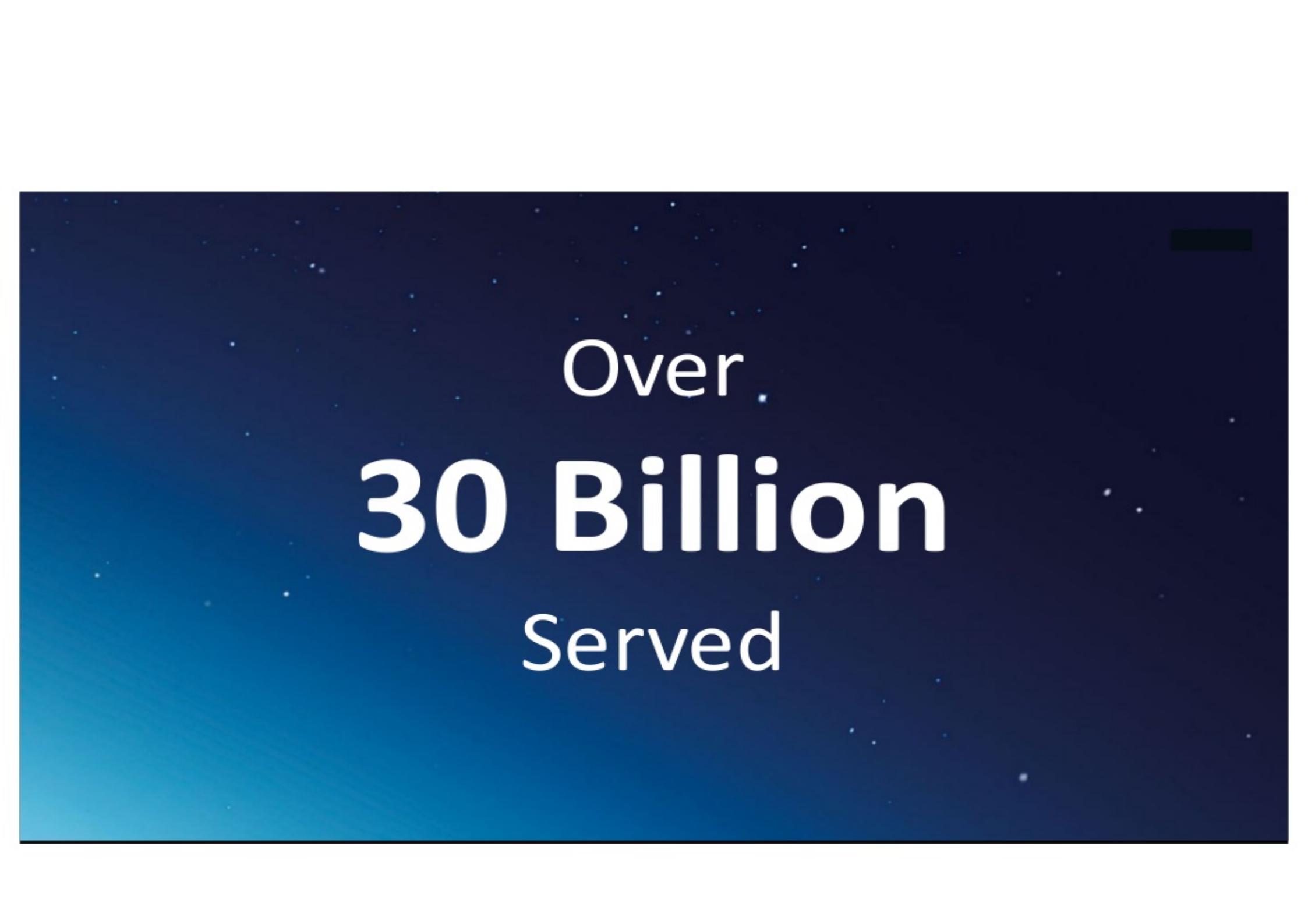
Everywhere Defined

- 26B forecasts /day or *250,000/second*
 - vs *3.5B Google queries daily*
- 2.2 billion unique locations
- 200k personal weather stations
- 200M active mobile users
- Petabytes of data generated daily

Our Brands

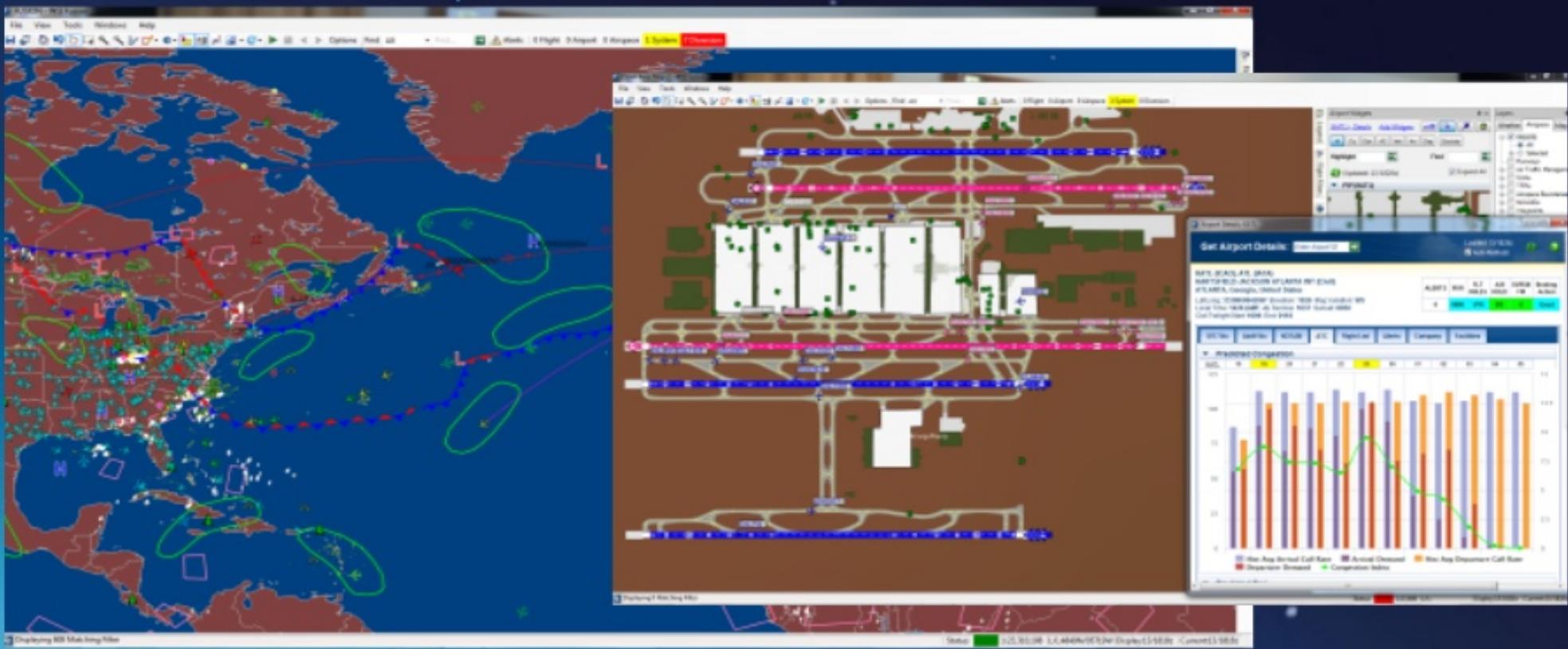
The
Weather
Channel



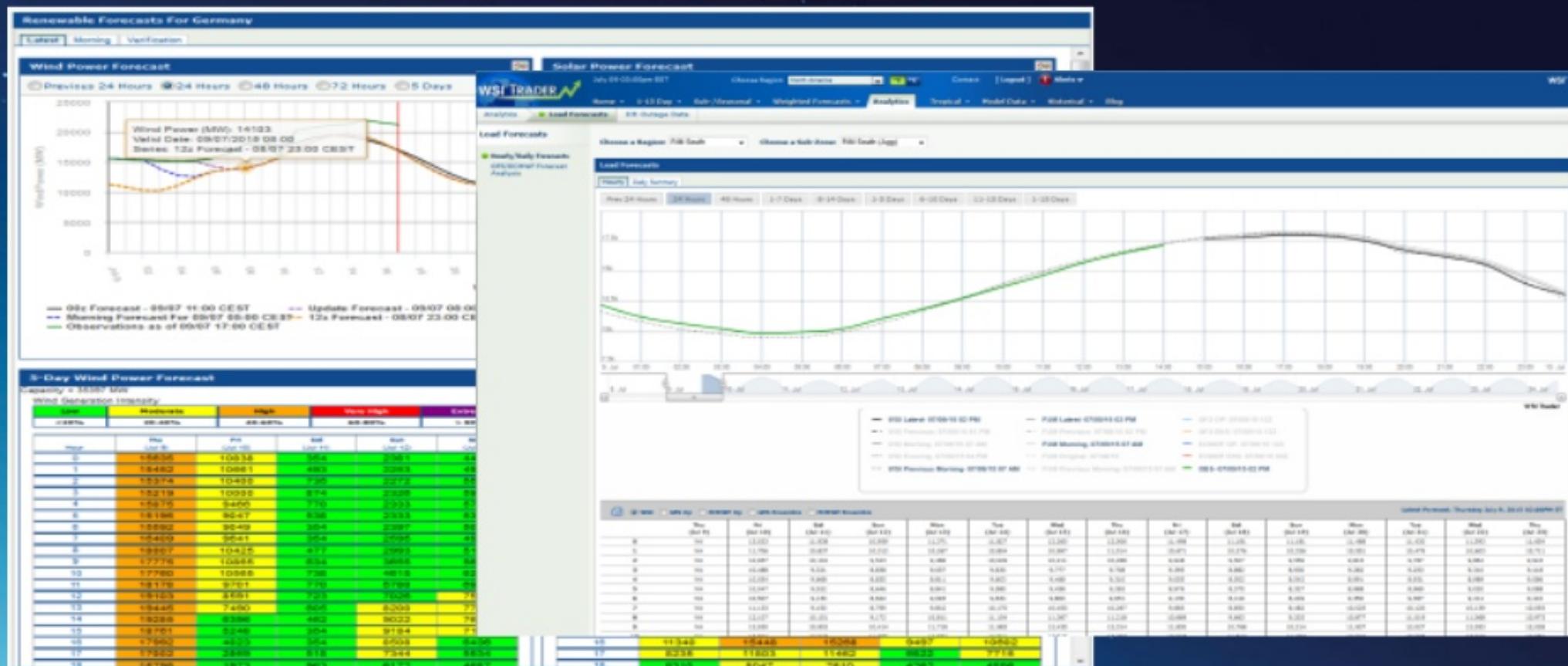


Over
30 Billion
Served

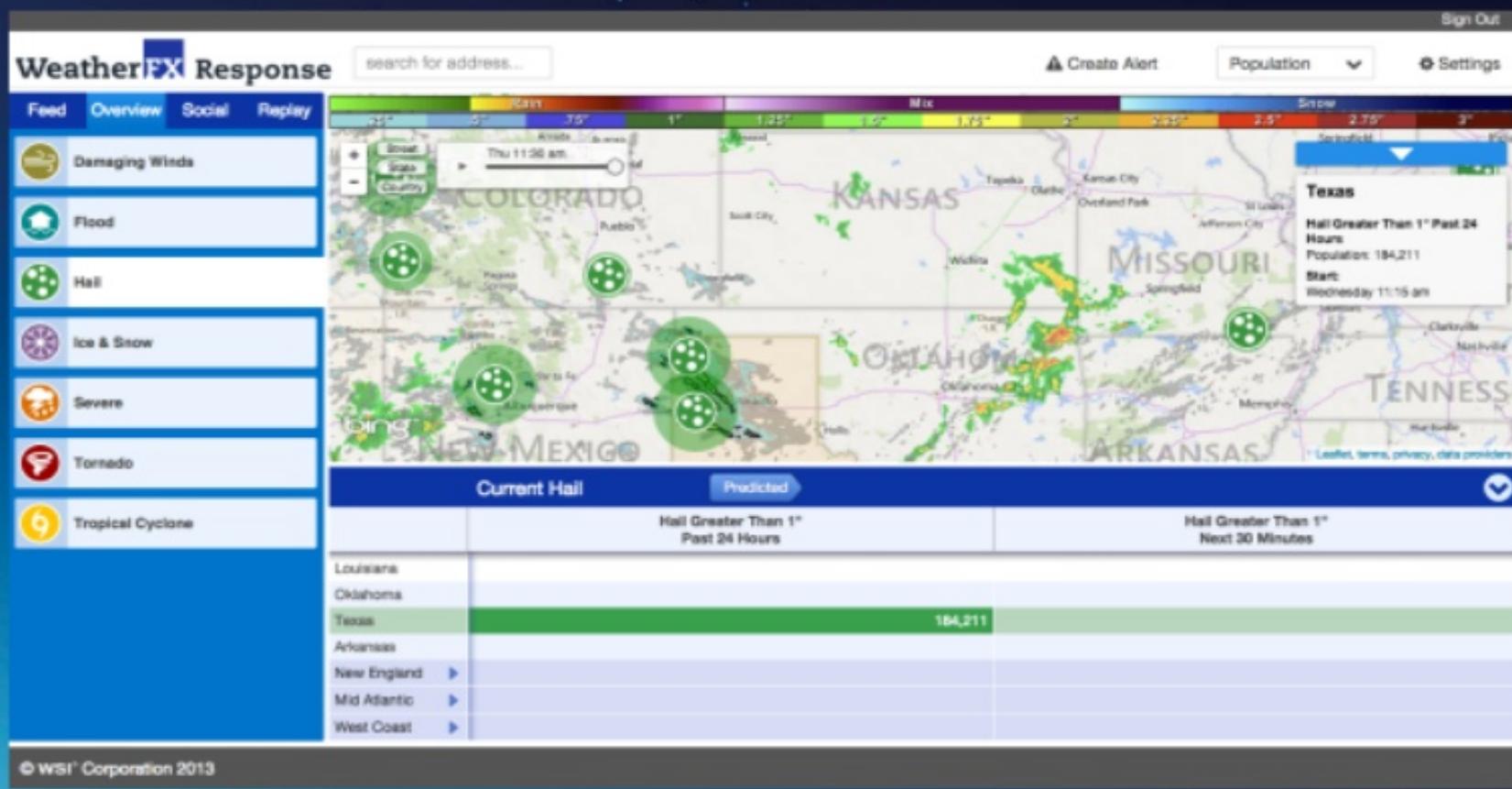
Flight Routing



Energy Trading



Insurance



Weather Alerting

4:32

Monday, April 27



Real-Time Rain Alert

4:32 PM

Rain will begin around 4:49pm, continuing off and on over next half hour. The rain will be light.

11:25

Monday, April 27



The Weather now

Lightning struck at 11:25pm, 10 miles NE of your current location. Make your move to safety now.

slide to view

Decisions at Scale

101001110100101



101001110100101
010100101011001
101010101011100
000011010110010



Who Are You?



RDBMS

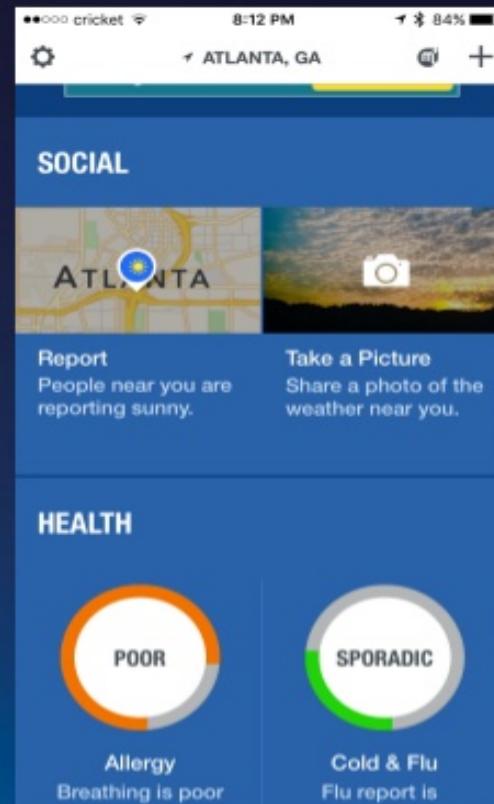
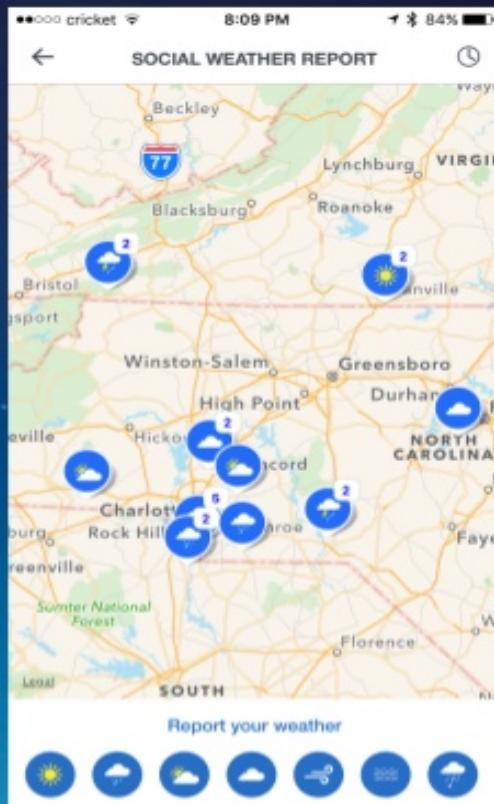
→ *Spark* ?

Who Are You?



→ *Spark*
?

Social Weather



Social Weather



RDBMS

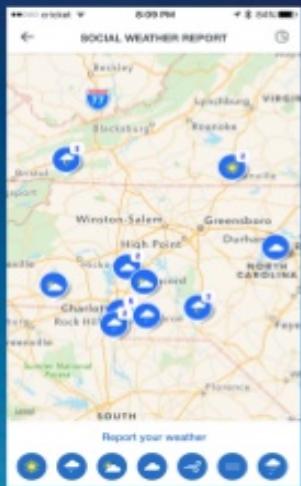
Social Weather



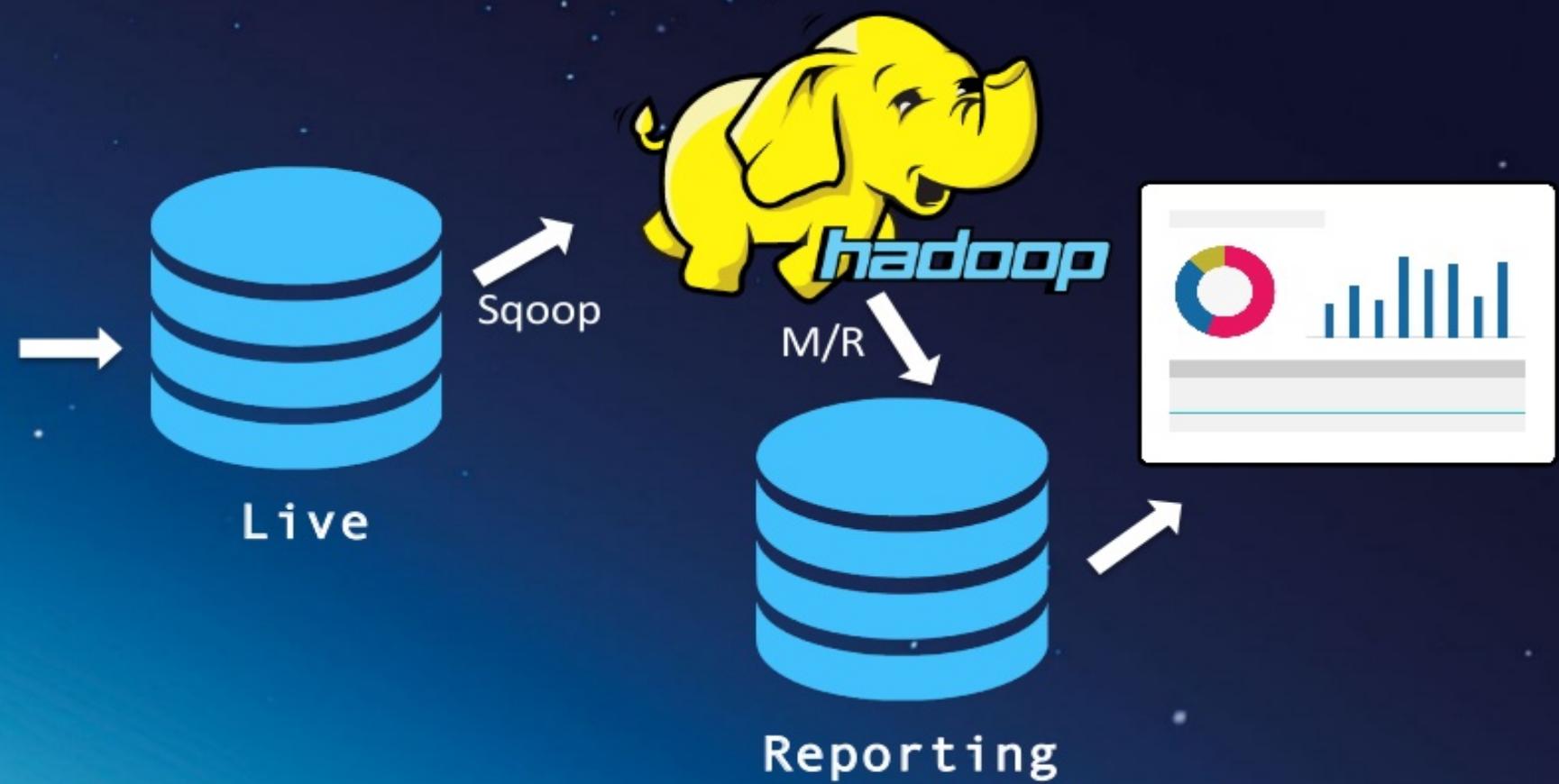
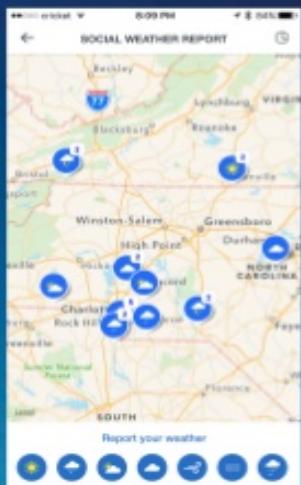
RDBMS

```
SELECT count(*) FROM wx_reports  
GROUP BY time / 300000 * 300000
```

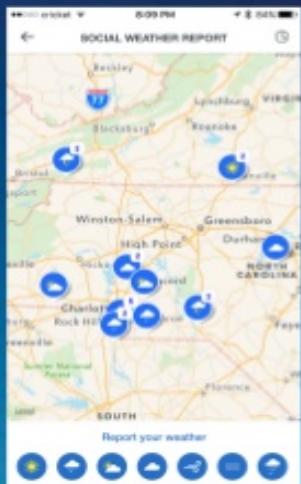
Social Weather



Social Weather



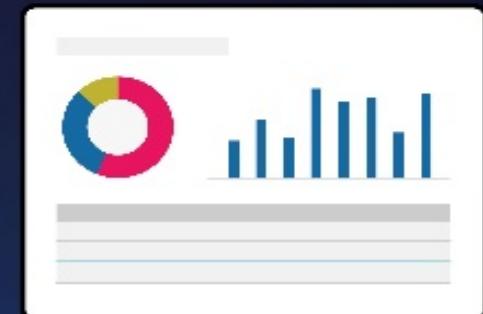
Scaling with Spark



Live



Reporting



Easing the Transition

101001110100101



• 101001110100101
010100101011001
101010101011100
000011010110010



Easing the Transition

101001110100101



101001110100101
010100101011001
101010101011100
000011010110010

Easing the Transition

101001110100101



- 101001110100101
010100101011001
101010101011100
000011010110010

101001110100101
010100101011001
101010101011100
000011010110010



- 10100,11101,00101
01010,01010,11001
10101,01010,11100
00001,10101,...

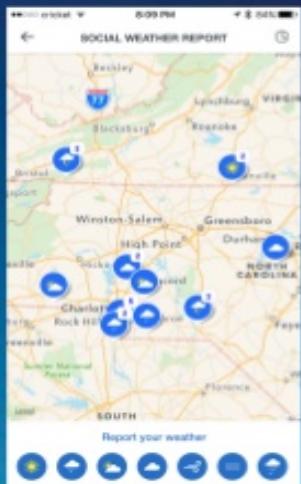
Easing the Transition

10100111010010
10100111010010
010100101011001
101010101011100
000011010110010

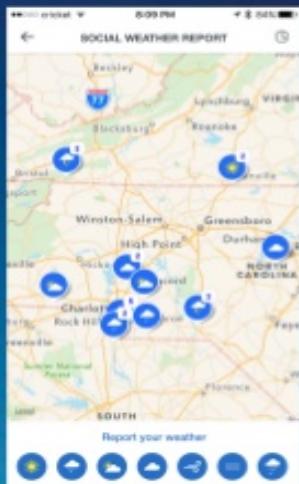


101001110100101
010100101011001
101010101011100
00011010110010
0100,11101,00101
01010,01010,11001
10101,01010,11100
00001,10101,...

Scaling with Spark



Scaling with Spark



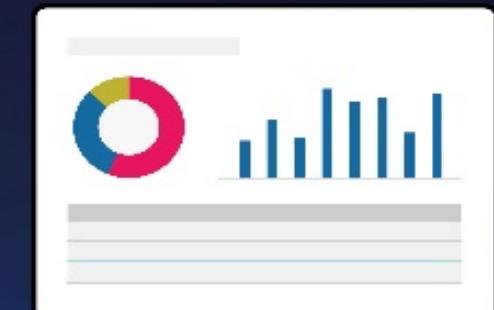
→ **Spark[★]**



Live



Reporting



Batch Aggregation

```
val wx_reports = // load data from database

val sql = new org.apache.spark.sql.SQLContext(sc)
import sql.implicits._

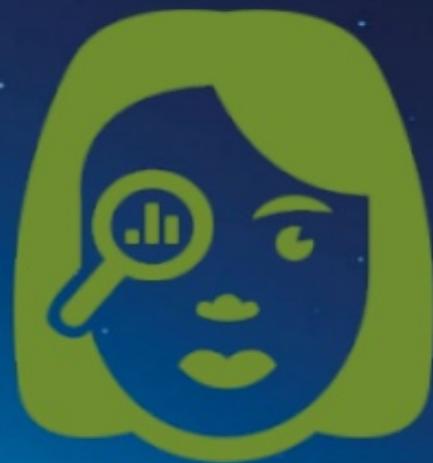
wx_reports.toDF.registerTempTable("wx_reports")

val counts = sql("select count(*) from wx_reports group by
timestamp / 300000 * 300000")
```

Streaming Aggregation

```
val wx_reports = // load from streaming source  
  
wx_reports.foreachRDD { rdd =>  
    val sql = SQLContext.getOrCreate(rdd.sparkContext)  
    import sql.implicits._  
    rdd.toDF.registerTempTable("wx_reports")  
    val count = sql("select count(*) from wx_reports")  
}
```

Data Science Roles



Data Scientist



Data Engineer

Data Science Roles



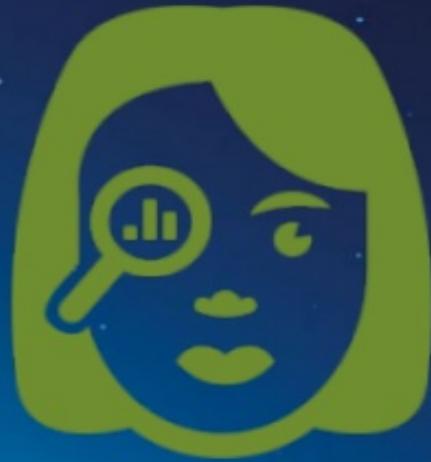
Data Scientist

Machine learning expert



Data Engineer

Data Science Roles



Data Scientist

Machine learning expert



Data Engineer

Scalable algorithms expert

Data Science Roles



Data Scientist

Builds pipelines that
work on her laptop



Data Engineer

Data Science Roles



Data Scientist



Data Engineer

Rewrites her pipelines
to scale better

Collaborative Data Science

Jupyter Spark - Scala Test Last Checkpoint: 12/07/2016 (autosaved)

File Edit View Insert Cell Kernel Help

Control Panel Logout

Spark 1.4.1 (Scala 2.10.4)

In [1]:

```
import com.databricks.spark.connector_
import org.apache.spark._

case class LUE(userid: String, timestamp: _
```

In [1]:

```
sqlContext = kernel.sqlContext
import sqlContext.implicits._

sc.cassandraTable[LUE]("prod_analytics_e
```

In [1]:

```
sqlContext.sql("select * from lues limit
```

In [1]:

```
@@sparkc
sqlContext <- sparkSQL.init(sc)
df <- sql(sqlContext, "select * from lue")
showDF(df)
```

In [1]:

```
@@sql select * from lues limit 10
```

In [1]:

```
@lemagic
```

In [1]:

```
@showtypes
```

In [1]:

```
@@html
<h1>Hello</h1>
<p>world</p>
```

In [1]:

```
@adddeps com.databricks spark-csv_2.10_1
```

In [1]:

```
import org.apache.spark_
val sqlContext = kernel.sqlContext
import sqlContext.implicits.
```

Zeppelin Notebook Interpreter Connected

Verifying Akamai Aggregation

```
val fn = "s3n://luc/prod/akamai/lat-analysis/cleansed.akamai.parquet/file=csx_prod/latDate=20151227"
sqlContext.read.parquet(fn).registerTempTable("csxakamai")
fn: String = s3n://luc/prod/akamai/lat-analysis/cleansed.akamai.parquet/file=csx_prod/latDate=20151227
Task 00 seconds.
```

FINISHED > 23 0 0

```
Msq
select logDate, count(*) as bounces
from dsxakamai
where httpVerb = 'POST'
and requestURI like '/dsx.weather.com/us/ocncc8'
group by logDate
order by logDate
```

FINISHED > 33 0 0

201,197.00
180,000.00
160,000.00
140,000.00
120,000.00
100,000.00
80,000.00
60,000.00
40,000.00
20,000.00
0.00

20151201 20151204 20151206 20151208 20151210 20151212 20151214 20151216 20151218 20151220 20151222 20151224 20151226 20151228 20151231

Stacked Optimized Unstacked

blue

blue

The Analytics OS

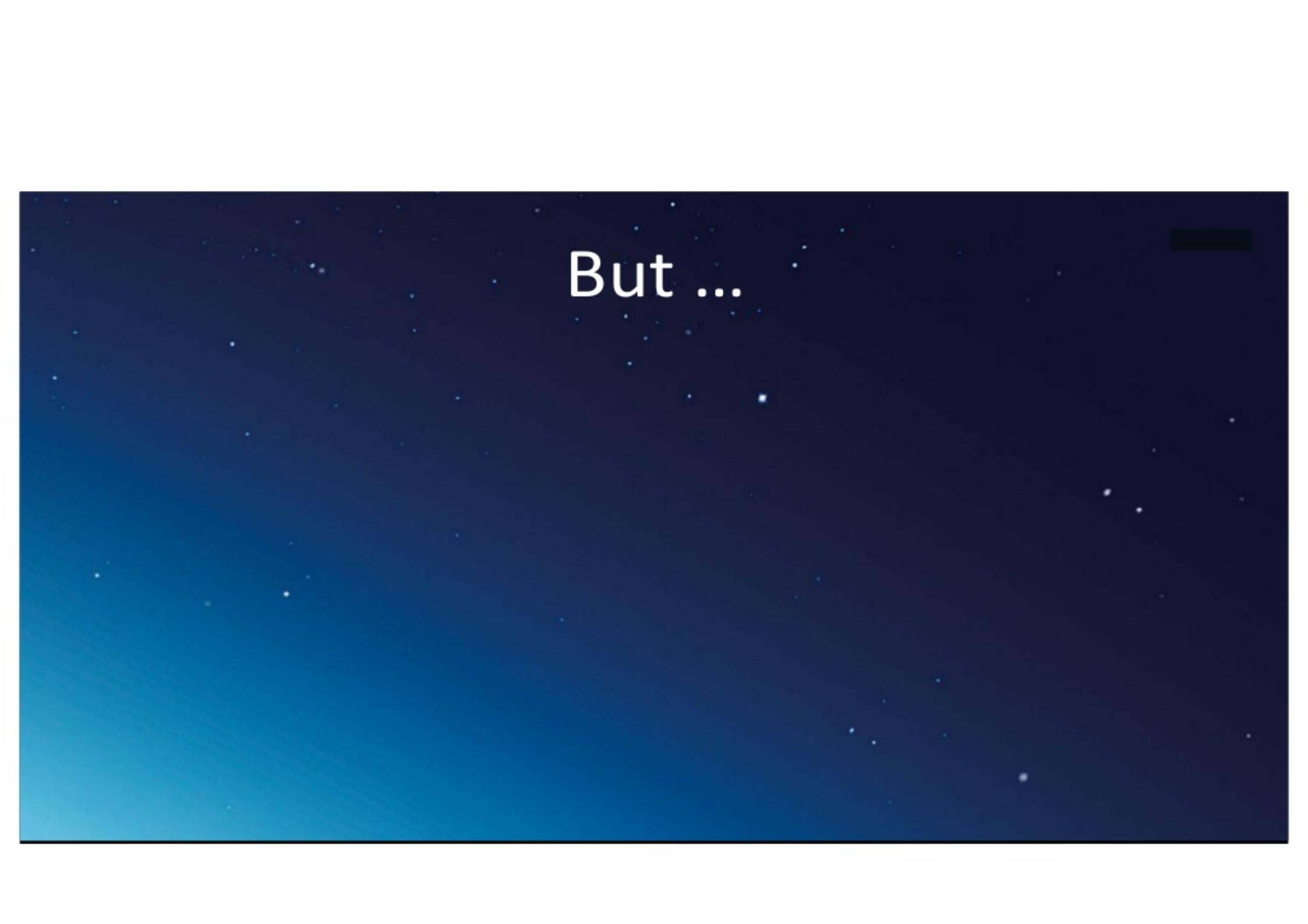


Notebooks

Stream
Analytics

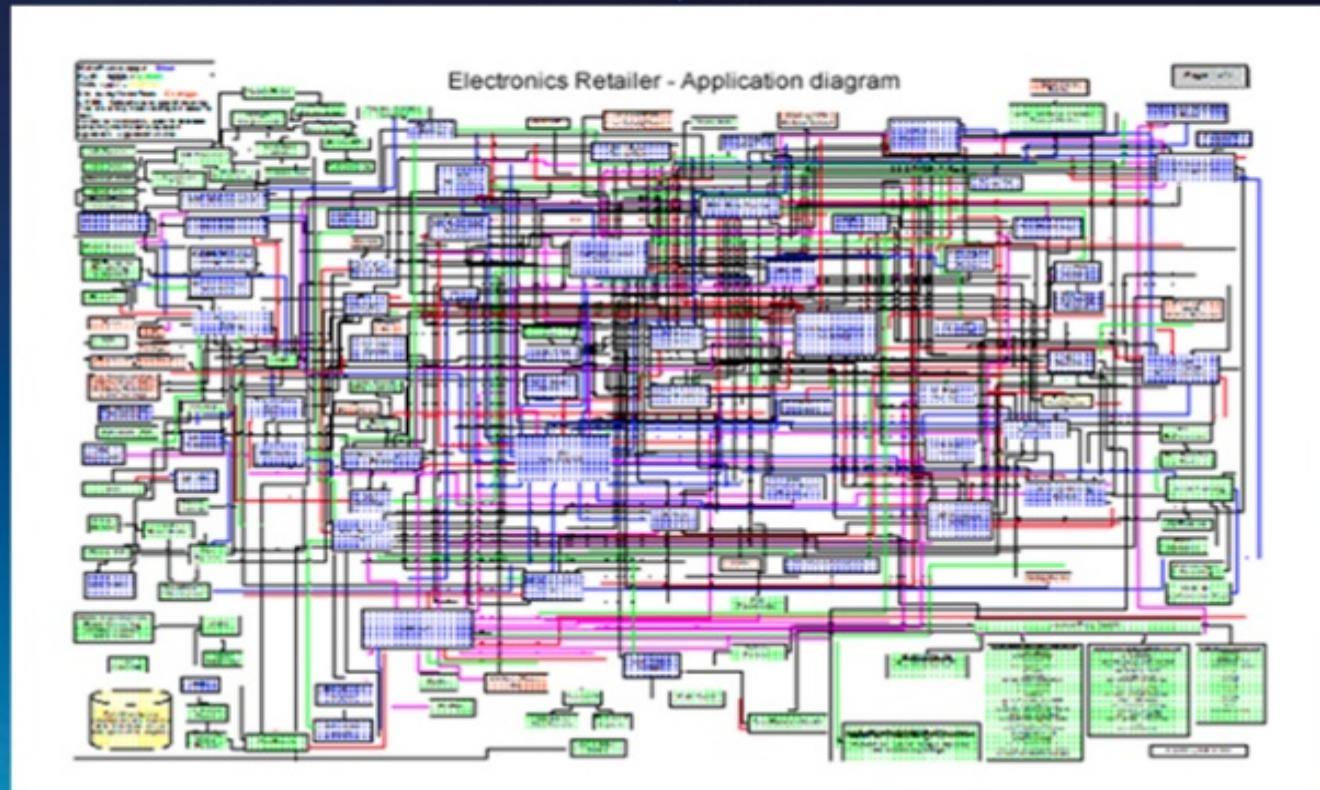
Batch
Analytics

Spark

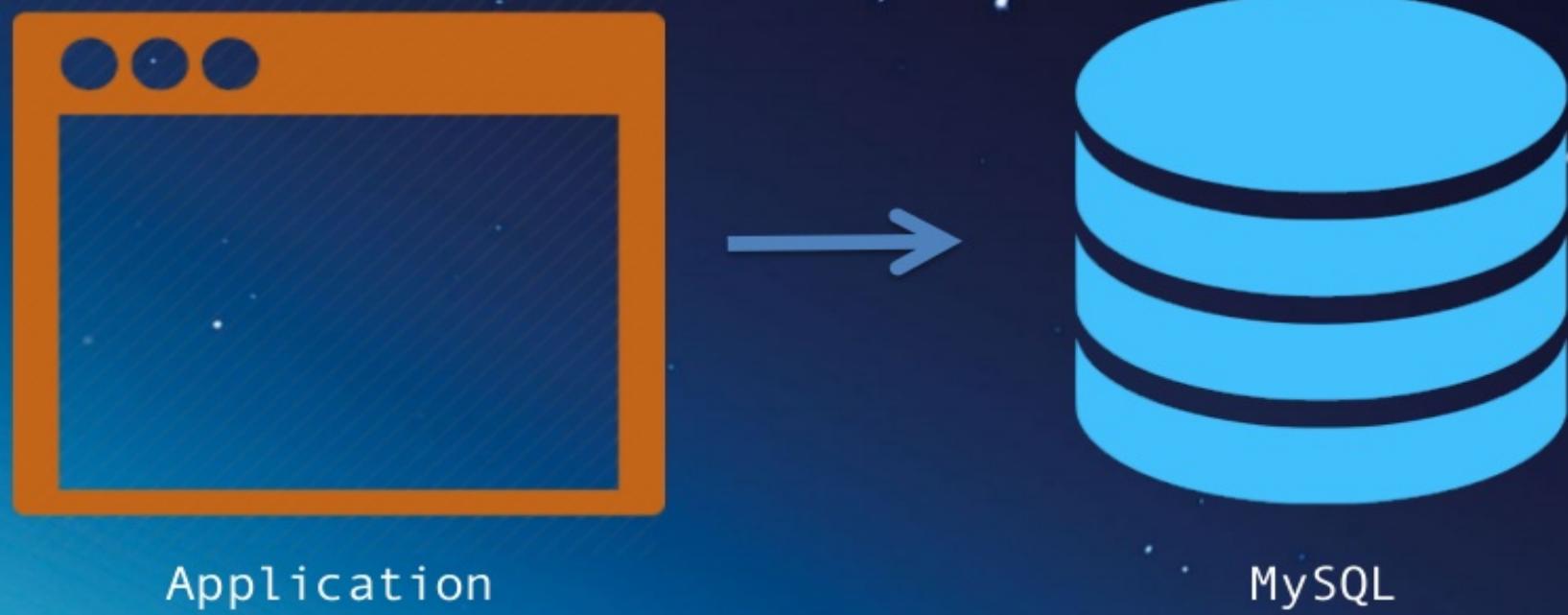


But ...

The Real World (Enterprise Version)



The Real World (Startup Version)



Step 1: Pick a Problem to Solve



Step 2: Build a Data Lake





Step 3: Set up Spark

- Direct download
- Hadoop distribution
(Hortonworks, Cloudera, etc)
- Managed service (Elastic
MapReduce, Databricks,
BlueMix, etc)

Step 4: Start Collecting Data

- Options:
 - Sqoop to move RDBMS tables
 - Flume/FluentD to move logs
 - Import from Spark-supported data sources
 - Using Spark Streaming attached to a queue
 - ...

Step 5: Use a Notebook

Jupyter Spark - Scala Test Last Checkpoint: 12/07/2016 (autosaved)

File Edit View Insert Cell Kernel Help

Control Panel Logout

Spark 1.4.1 (Scala 2.10.4)

In [1]:

```
import com.databricks.spark.connector_
import org.apache.spark._

case class LUE(userid: String, timestamp

val sqlContext = kernel.sqlContext
import sqlContext.implicits_
sc.cassandraTable[LUE]("prod_analytics_e

In [1]: sqlContext.sql("select * from lues limit

In [1]: %%spark
sqlContext <- sparkSQL.init(sc)
df <- sql(sqlContext, "select * from lue
showDF(df)

In [1]: %%sql select * from lues limit 10

In [1]: %elemagic

In [1]: %showtypes

In [1]: %%html
<h1>Hello</h1>
<p>world</p>

In [1]: %%adddpsp com.databricks spark-csv_2.10 1

In [1]: import org.apache.spark_
val sqlContext = kernel.sqlContext
import sc.cassandraTable
```

Zeppelin Notebook Interpreter Connected

Verifying Akamai Aggregation

```
val fn = "s3n://luc/prod/akamai/lat-analysis/cleansed.akamai.parquet/file=dsx_prod/latDate=20151227"
sqlContext.read.parquet(fn).registerTempTable("dsxakamai")
fn: String = s3n://luc/prod/akamai/lat-analysis/cleansed.akamai.parquet/file=dsx_prod/latDate=20151227
Task 00 seconds.
```

FINISHED > 23 0 0

```
%%sql
select logDate, count(*) as bounces
from dsxakamai
where httpVerb = 'POST'
and requestURI like '/dsx/weather/convs/locnck'
group by logDate
order by logDate
```

FINISHED > 33 0 0

201,197.00
180,000.00
160,000.00
140,000.00
120,000.00
100,000.00
80,000.00
60,000.00
40,000.00
20,000.00
0.00

20151201 20151204 20151206 20151208 20151210 20151212 20151214 20151216 20151218 20151220 20151222 20151224 20151226 20151228 20151231

Stacked Optimized Optimized

blue

blue

Final Thoughts



Thank You!



Robbie Strickland
@rs_atl

(we're hiring!)