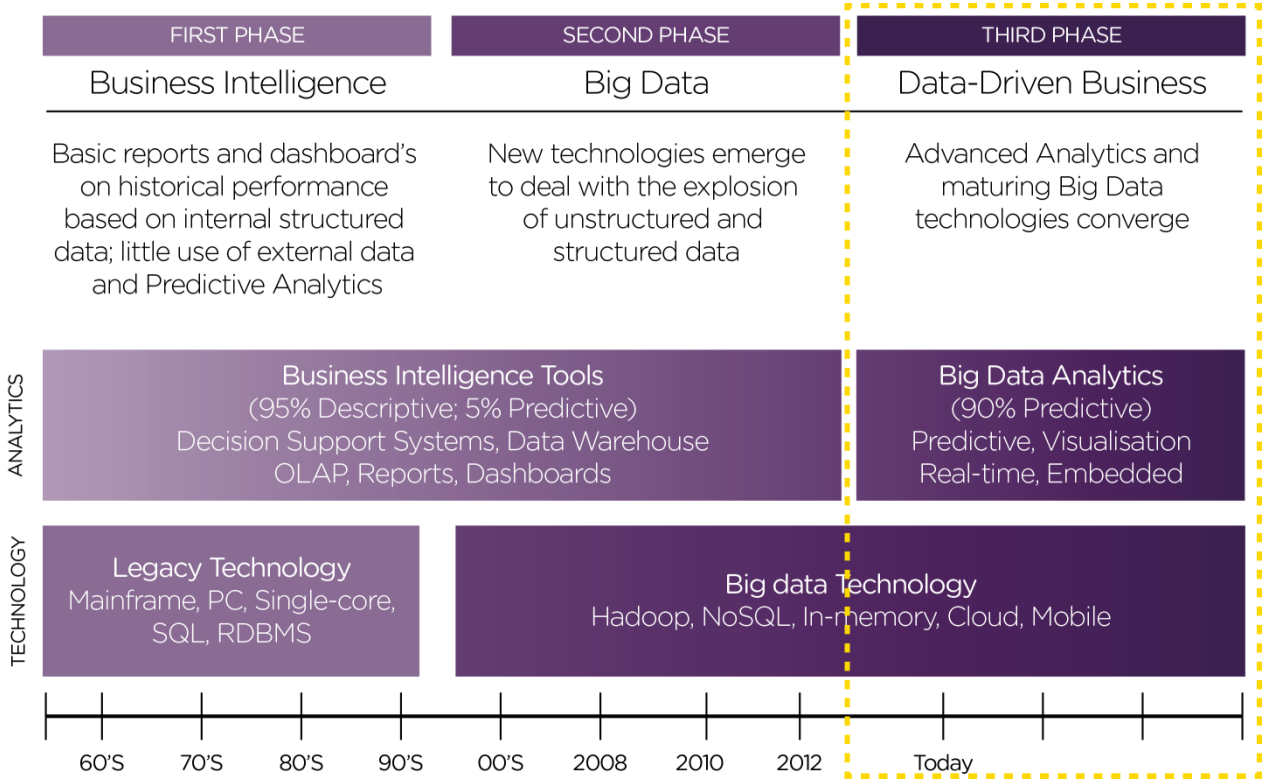
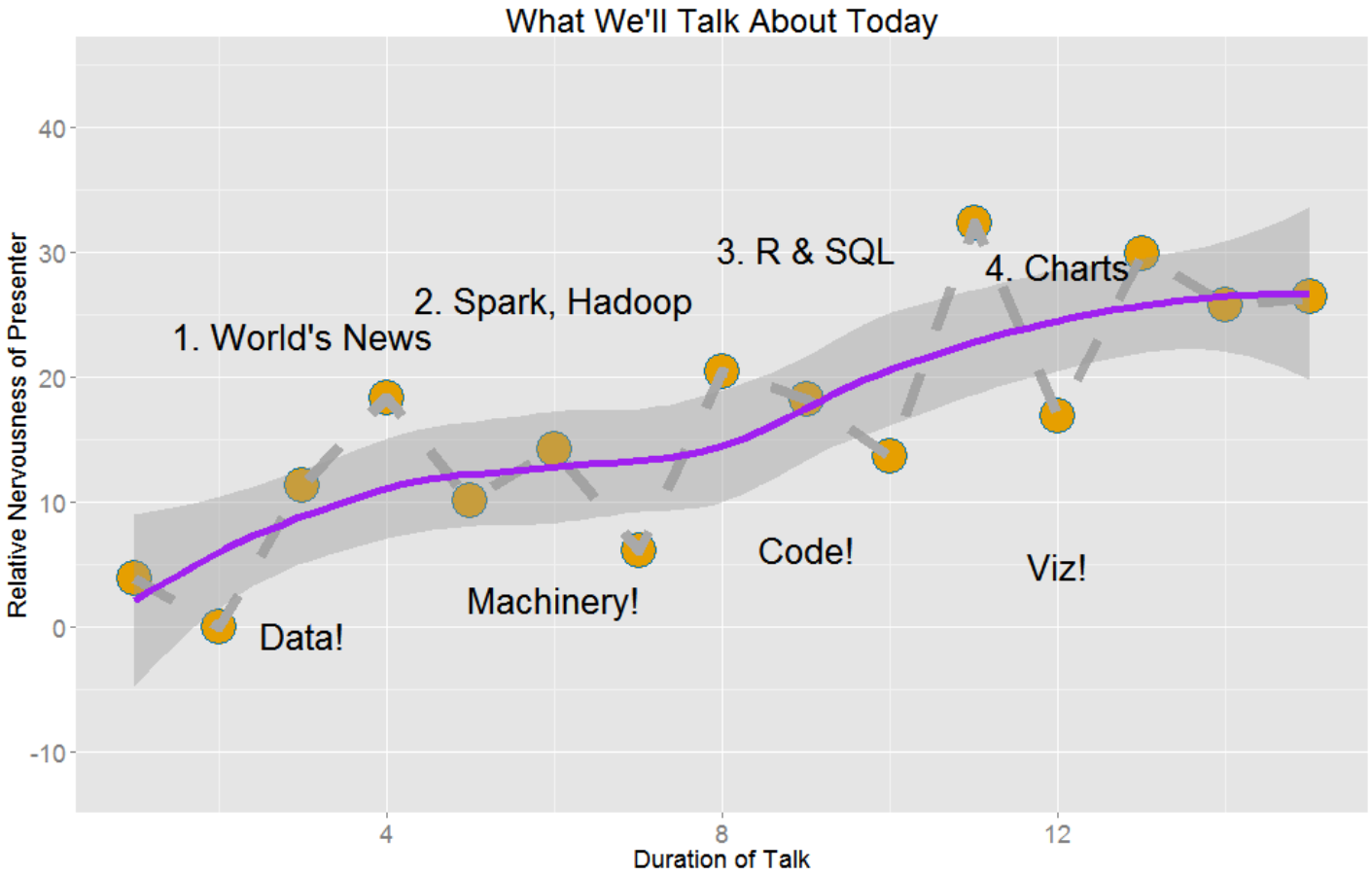
AQUILAINSIGHT
A new model for analytics

Distributed Analytics with R, Spark & Hadoop (And SQL)

Analytics entering its third phase



AQUILAINSIGHT
A new model for analytics



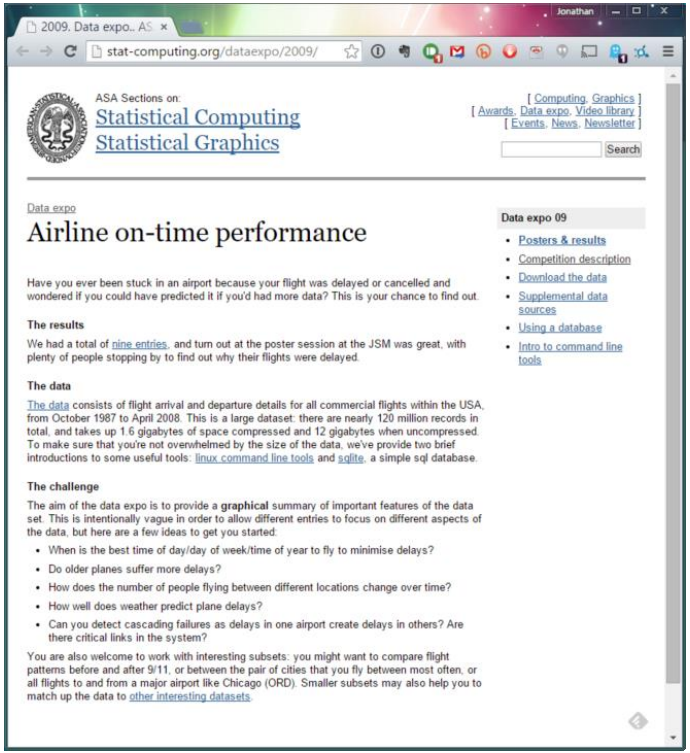
We're going to...

- Collect together chunks from all the world's newsworthy events from 1979
- Collect on-time performance data for US commercial flights between 1987 to 2008
- Find something interesting in them and...
- Draw some charts and graphics.

... by storing, processing and analysing it with ...

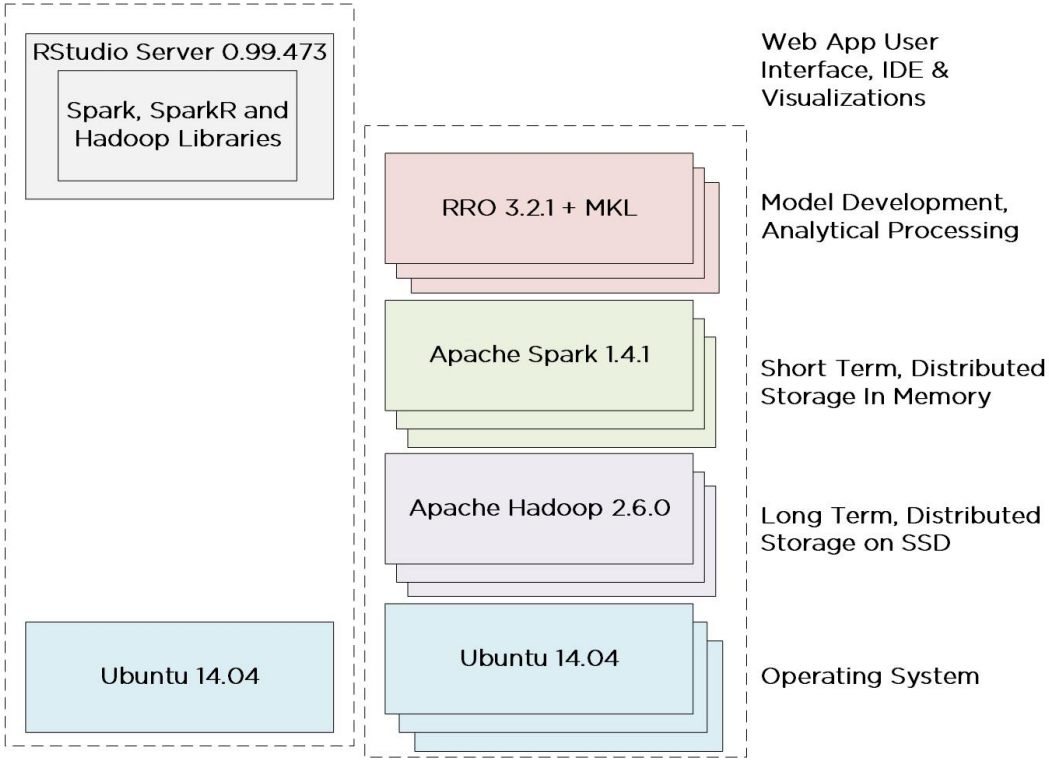
- R and RStudio
- Apache Spark
- Apache Hadoop
- A handy cluster of servers running on AWS

Our Data Sources



AQUILAINSIGHT
A new model for analytics

Our Building Blocks



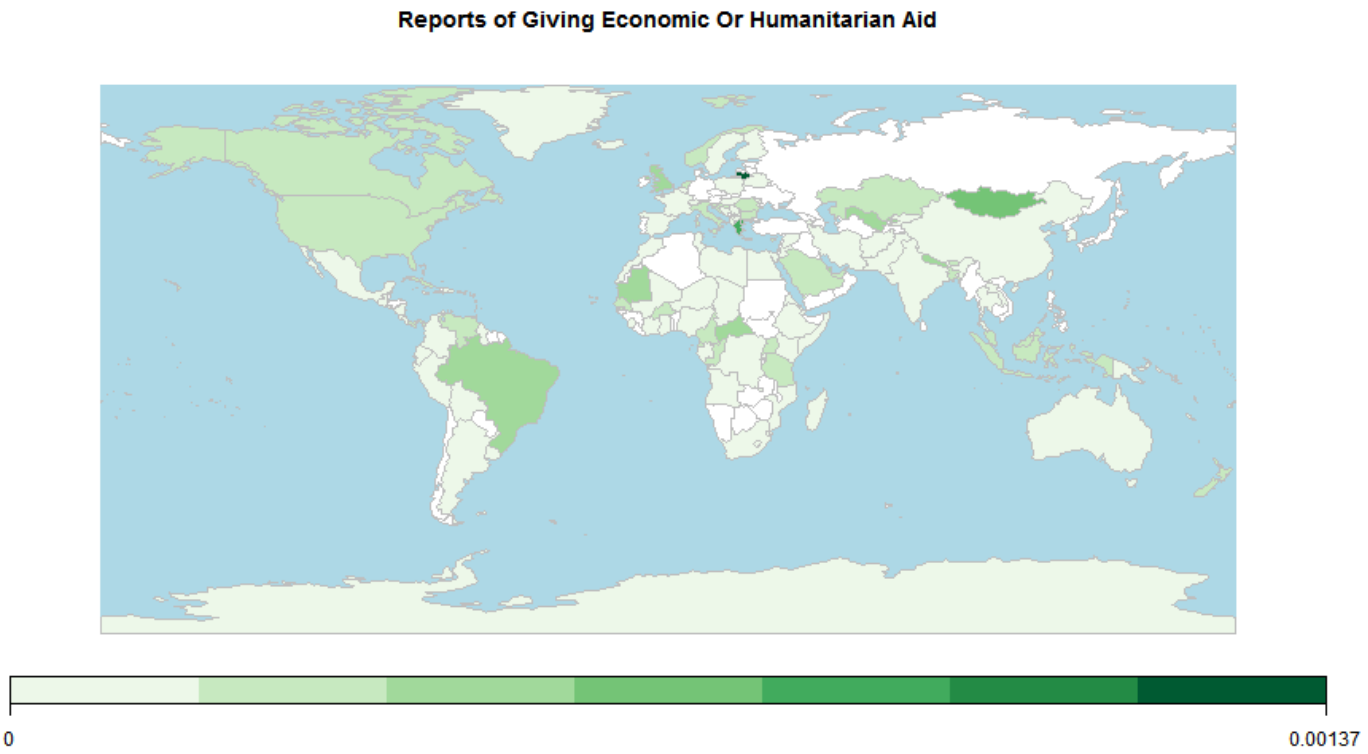
	Hadoop MR Record	Spark Record	Spark 1 PB
Data Size	102.5 TB	100 TB	1000 TB
Elapsed Time	72 mins	23 mins	234 mins
# Nodes	2100	206	190
# Cores	50400 physical	6592 virtualized	6080 virtualized
Cluster disk throughput	3150 GB/s (est.)	618 GB/s	570 GB/s
Sort Benchmark Daytona Rules	Yes	Yes	No
Network	dedicated data center, 10Gbps	virtualized (EC2) 10Gbps network	virtualized (EC2) 10Gbps network
Sort rate	**1.42 TB/min**	**4.27 TB/min**	**4.27 TB/min**
Sort rate/node	**0.67 GB/min**	**20.7 GB/min**	**22.5 GB/min**

GDELT Reports of Giving Economic or Humanitarian Aid Between 1st July 2015 and 2nd July 2015

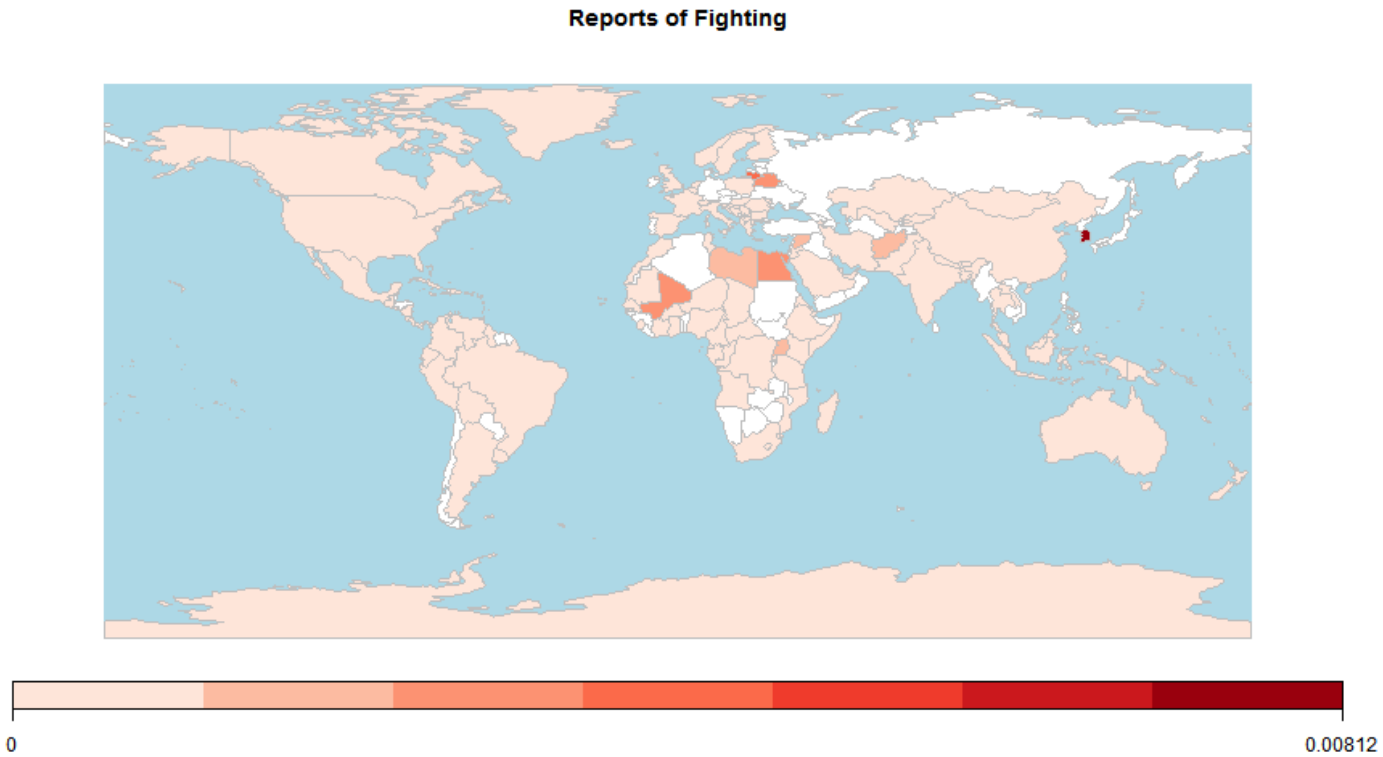
```
library(RPushbullet)

library(GDELTtools)

start.time <- Sys.time()

gdelt.aid <- GetGDELT(start.date="2015-07-01", end.date="2015-07-02", local.folder="E:\\datasci\\scotlandis2015\\data", filter=list(EventCode=c("071","073")))

normed.gdelt.aid <- NormEventCounts(gdelt.aid, unit.analysis="country.year",var.name="reports.of.aid")

library("rworldmap")

map.data.aid <- joinCountryData2Map(normed.gdelt.aid, joinCode="ISO2", nameJoinColumn="country")

colourPalette <- brewer.pal(7,'Greens')

mapCountryData(map.data.aid, nameColumnToPlot="reports.of.aid.norm", mapTitle="Reports of Giving Economic Or Humanitarian Aid", colourPalette = colourPalette, oceanCol="lightBlue", missingCountryCol="white", cat="fixedWidth")

# Alert job completion

end.time <- Sys.time()

time.taken <- end.time - start.time

time.taken

message <- paste("Time was ", time.taken, " mins")

pbPost("note", "Econ/Humanitarian plot done", message)
```

AQUILAINSIGHT
A new model for analytics

GDELT Reports of Giving Economic or Humanitarian Aid Between 1st July 2015 and 2nd July 2015



GDELT Reports of Fighting Between 1st July 2015 and 2nd July 2015



Now onto Airline On-Time Performance…all commercial flights in US between 1987 and 2006

```
# Set up Spark Context inside R
# Set env variables for Spark & Hadoop
Sys.setenv(SPARK_HOME="/home/spark-user/spark")
Sys.setenv(HADOOP_HOME="/home/spark-user/hadoop")

# This line loads SparkR from the installed
directory
.libPaths(c(file.path(Sys.getenv("SPARK_HOME"),
"R", "lib"), .libPaths()))
.libPaths(c(file.path(Sys.getenv("HADOOP_HOME"),
"lib"), .libPaths()))

# Prepare to read in CSV format files from HDFS
Sys.setenv('SPARKR_SUBMIT_ARGS'='"--packages"
"com.databricks:spark-csv_2.10:1.0.3" "sparkr-
shell"')

library(SparkR)
sc <- sparkR.init(master="spark://ip-172-31-10-
150:7077")
sqlContext <- sparkRSQL.init(sc)
```

```
# Load up files
airports <- read.df(sqlContext, "hdfs://ip-172-31-
10-150:50050/data/aotp/airports.csv",
"com.databricks.spark.csv", header="true")

carriers <- read.df(sqlContext, "hdfs://ip-172-31-
10-150:50050/data/aotp/carriers.csv",
"com.databricks.spark.csv", header="true")

plane_data <- read.df(sqlContext, "hdfs://ip-172-
31-10-150:50050/data/aotp/plane-data.csv",
"com.databricks.spark.csv", header="true")

aotp <- read.df(sqlContext, "hdfs://ip-172-31-10-
150:50050/data/aotp/csv/all_flights.csv",
"com.databricks.spark.csv", header="true")
#
# Show e.g. schema info
printSchema(aotp)
# Turn them into temp tables
registerTempTable(airports, "airports")
registerTempTable(carriers,  "carriers")
registerTempTable(plane_data, "plane_data")
registerTempTable(aotp,  "aotp")
```

Schema info – aotp

```
> printSchema(aotp)
root
 |-- Year: string (nullable = true)
 |-- Month: string (nullable = true)
 |-- DayofMonth: string (nullable = true)
 |-- DayOfWeek: string (nullable = true)
 |-- DepTime: string (nullable = true)
 |-- CRSDepTime: string (nullable = true)
 |-- ArrTime: string (nullable = true)
 |-- CRSArrTime: string (nullable = true)
 |-- UniqueCarrier: string (nullable = true)
 |-- FlightNum: string (nullable = true)
 |-- TailNum: string (nullable = true)
 |-- ActualElapsedTime: string (nullable = true)
 |-- CRSElapsedTime: string (nullable = true)
 |-- AirTime: string (nullable = true)
 |-- ArrDelay: string (nullable = true)
 |-- DepDelay: string (nullable = true)
 |-- Origin: string (nullable = true)
 |-- Dest: string (nullable = true)
 |-- Distance: string (nullable = true)
 |-- TaxiIn: string (nullable = true)
 |-- TaxiOut: string (nullable = true)
 |-- Cancelled: string (nullable = true)
 |-- CancellationCode: string (nullable = true)
 |-- Diverted: string (nullable = true)
 |-- CarrierDelay: string (nullable = true)
 |-- WeatherDelay: string (nullable = true)
 |-- NASDelay: string (nullable = true)
 |-- SecurityDelay: string (nullable = true)
 |-- LateAircraftDelay: string (nullable = true)
```

Schema info – airports, carriers and plane_data

```
> printSchema(airports)
root
 |-- iata: string (nullable = true)
 |-- airport: string (nullable = true)
 |-- city: string (nullable = true)
 |-- state: string (nullable = true)
 |-- country: string (nullable = true)
 |-- lat: string (nullable = true)
 |-- long: string (nullable = true)

> printSchema(carriers)
root
 |-- Code: string (nullable = true)
 |-- Description: string (nullable = true)

> printSchema(plane_data)
root
 |-- tailnum: string (nullable = true)
 |-- type: string (nullable = true)
 |-- manufacturer: string (nullable = true)
 |-- issue_date: string (nullable = true)
 |-- model: string (nullable = true)
 |-- status: string (nullable = true)
 |-- aircraft_type: string (nullable = true)
 |-- engine_type: string (nullable = true)
 |-- year: string (nullable = true)
```

Now run a simple SQL query…

```
#
# Set up a simple query
simple_query <- sql(sqlContext,
"SELECT Year, count(FlightNum)
 FROM aotp
 GROUP BY Year")
#
start.time <- Sys.time()
showDF(simple_query)
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken
#
# Alert job completion
message <- paste("Time was ", time.taken, " mins")
pbPost("note", "SQL Job Done", message)
#
#sparkR.stop()
```

```
Console ~/scotsoft2015/
15/09/05 11:42:14 INFO TaskSetManager: Finished task 197.0
15/09/05 11:42:14 INFO TaskSetManager: Finished task 196.0
15/09/05 11:42:14 INFO TaskSetManager: Finished task 198.0
15/09/05 11:42:14 INFO TaskSchedulerImpl: Removed TaskSet 7
15/09/05 11:42:14 INFO DAGScheduler: ResultStage 7 (showStr
15/09/05 11:42:14 INFO DAGScheduler: Job 5 finished: showSt
+----+-------+
|Year|     c1|
+----+-------+
|1987|1311826|
|1988|5202096|
|1989|5041200|
|1990|5270893|
|1991|5076925|
|1992|5092157|
|1993|5070501|
|1994|5180048|
|1995|5327435|
|1996|5351983|
|1997|5411843|
|1998|5384721|
|1999|5527884|
|2000|5683047|
|2001|5967780|
|2002|5271359|
|2003|6488540|
|2004|7129270|
|2005|7140596|
|2006|7141922|
+----+-------+
> end.time <- Sys.time()
> time.taken <- end.time - start.time
> time.taken
Time difference of 4.793702 mins
> #
> # Alert job completion
> message <- paste("Time was ", time.taken, " mins")
> pbPost("note", "SQL Job Done", message)
```

Now run a join across Spark data frames…

```
#
# Set up a query across plane_data and aotp
#
simple_query <- sql(sqlContext, "SELECT
plane_data.tailnum, plane_data.aircraft_type,
plane_data.model, plane_data.year,
aotp.Origin, aotp.Dest
FROM plane_data, aotp
WHERE plane_data.tailnum = aotp.TailNum
AND   plane_data.aircraft_type is not NULL
AND   aotp.Year = 2006
AND   aotp.Month = 10")
#
start.time <- Sys.time()
showDF(simple_query)
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken
#
# Alert job completion
message <- paste("Time was ", time.taken, " mins")
pbPost("note", "SparkR aotp join_query job done",
message)
```

```
Console ~/scotsoft2015/
15/09/05 12:01:47 INFO MapOutputTrackerMasterEndpoint: Asked t
15/09/05 12:01:47 INFO MapOutputTrackerMaster: Size of output
15/09/05 12:01:47 INFO TaskSetManager: Finished task 0.0 in st
15/09/05 12:01:47 INFO TaskSchedulerImpl: Removed TaskSet 13.6
15/09/05 12:01:47 INFO DAGScheduler: ResultStage 13 (showStrin
15/09/05 12:01:47 INFO DAGScheduler: Job 7 finished: showStrin
+-------+--------------------+---------+----+------+----+
|tailnum|       aircraft_type|    model|year|Origin|Dest|
+-------+--------------------+---------+----+------+----+
| N11150|Fixed Wing Multi-...|EMB-145XR|2003|   DAB| EWR|
| N11150|Fixed Wing Multi-...|EMB-145XR|2003|   MCI| IAH|
| N11150|Fixed Wing Multi-...|EMB-145XR|2003|   GRR| EWR|
| N11150|Fixed Wing Multi-...|EMB-145XR|2003|   OKC| IAH|
| N11150|Fixed Wing Multi-...|EMB-145XR|2003|   EWR| LEX|
| N11150|Fixed Wing Multi-...|EMB-145XR|2003|   IND| CLE|
| N11150|Fixed Wing Multi-...|EMB-145XR|2003|   IAH| SLC|
| N11150|Fixed Wing Multi-...|EMB-145XR|2003|   BTR| IAH|
| N11150|Fixed Wing Multi-...|EMB-145XR|2003|   IAH| GRR|
| N11150|Fixed Wing Multi-...|EMB-145XR|2003|   EWR| IND|
| N11150|Fixed Wing Multi-...|EMB-145XR|2003|   CLE| ORF|
| N11150|Fixed Wing Multi-...|EMB-145XR|2003|   IAH| GRR|
| N11150|Fixed Wing Multi-...|EMB-145XR|2003|   RIC| IAH|
| N11150|Fixed Wing Multi-...|EMB-145XR|2003|   BPT| IAH|
| N11150|Fixed Wing Multi-...|EMB-145XR|2003|   LCH| IAH|
| N11150|Fixed Wing Multi-...|EMB-145XR|2003|   PSP| IAH|
| N11150|Fixed Wing Multi-...|EMB-145XR|2003|   IAH| PSP|
| N11150|Fixed Wing Multi-...|EMB-145XR|2003|   IAH| MSP|
| N11150|Fixed Wing Multi-...|EMB-145XR|2003|   GRK| IAH|
| N11150|Fixed Wing Multi-...|EMB-145XR|2003|   AVL| IAH|
+-------+--------------------+---------+----+------+----+
> end.time <- Sys.time()
> time.taken <- end.time - start.time
> time.taken
Time difference of 4.77374 mins
> #
> # Alert job completion
> message <- paste("Time was ", time.taken, " mins")
> pbPost("note", "SparkR aotp query done", message)
>
```

Now run a join across Spark data frames…

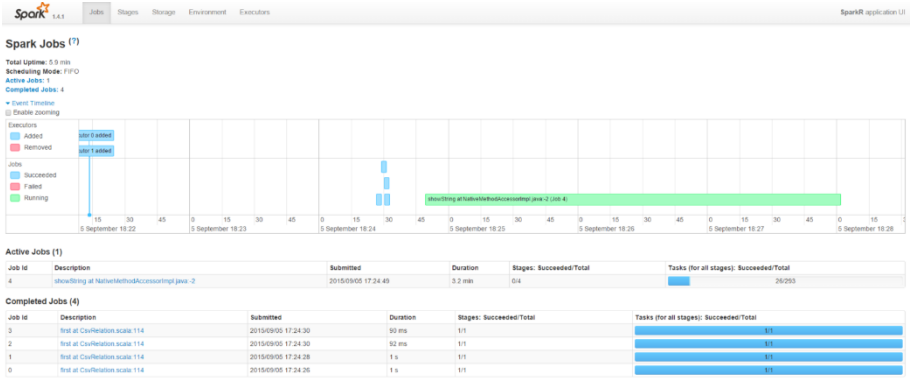
```
#
# Set up a query across plane_data and aotp
simple_query <- sql(sqlContext, "SELECT aotp.Year,
plane_data.aircraft_type,
COUNT(plane_data.aircraft_type)
FROM plane_data, aotp                    WHERE
plane_data.tailnum = aotp.TailNum
AND   plane_data.aircraft_type is not NULL
AND   aotp.Year = 2006
AND   aotp.Month = 10
GROUP BY aotp.Year, plane_data.aircraft_type")
#
start.time <- Sys.time()
showDF(simple_query)
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken
#
# Alert job completion
message <- paste("Time was ", time.taken, " mins")
pbPost("note", "SparkR aotp simple_query job
done", message)
#
```

```
15/09/05 12:12:35 INFO TaskSchedulerImpl: Removed Task
15/09/05 12:12:35 INFO DAGScheduler: ResultStage 21 (s
15/09/05 12:12:35 INFO DAGScheduler: Job 9 finished: s
+----+--------------------+------+
|Year|       aircraft_type|    c2|
+----+--------------------+------+
|2006|             Balloon|   340|
|2006|Fixed Wing Multi-...|475953|
|2006|           Rotorcraft|   252|
|2006|Fixed Wing Single...|  2313|
+----+--------------------+------+
> end.time <- Sys.time()
> time.taken <- end.time - start.time
> time.taken
Time difference of 4.947701 mins
> #
> # Alert job completion
> message <- paste("Time was ", time.taken, " mins")
> pbPost("note", "SparkR aotp query done", message)
>
```

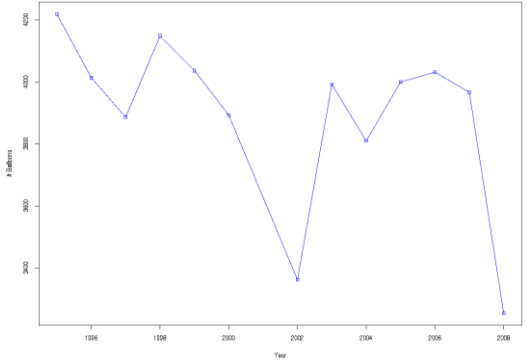
AQUILAINSIGHT
A new model for analytics

Now run a join across Spark data frames…

```
#
# Set up a query across plane_data and aotp
simple_query <- sql(sqlContext, "SELECT aotp.Year,
COUNT(plane_data.aircraft_type)
FROM plane_data, aotp
WHERE plane_data.tailnum = aotp.TailNum
AND   plane_data.aircraft_type = 'Balloon'
GROUP BY aotp.Year, plane_data.aircraft_type
ORDER BY aotp.Year ASC")
#
start.time <- Sys.time()
showDF(simple_query)
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken
#
# Alert job completion
message <- paste("Time was ", time.taken, " mins")
pbPost("note", "SparkR aotp simple_query job
done", message)
#
```



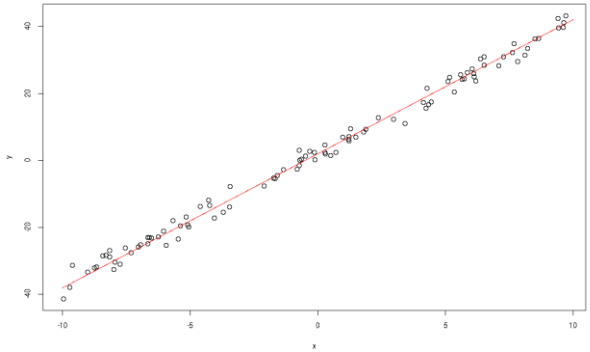

```
15/09/05 17:49:21 INFO TaskSchedulerImpl: Removed TaskSet 1
15/09/05 17:49:21 INFO DAGScheduler: ResultStage 15 (showSt
15/09/05 17:49:21 INFO DAGScheduler: Job 6 finished: showSt
+----+----+
|Year|  c1|
+----+----+
|1995|4219|
|1996|4014|
|1997|3887|
|1998|4149|
|1999|4037|
|2000|3894|
|2002|3363|
|2003|3993|
|2004|3810|
|2005|4000|
|2006|4032|
|2007|3967|
|2008|3255|
+----+----+
> end.time <- Sys.time()
> time.taken <- end.time - start.time
> time.taken
Time difference of 11.04948 mins
> #
> # Alert job completion
> message <- paste("Time was ", time.taken, " mins")
> pbPost("note", "SparkR aotp query done", message)
>
```



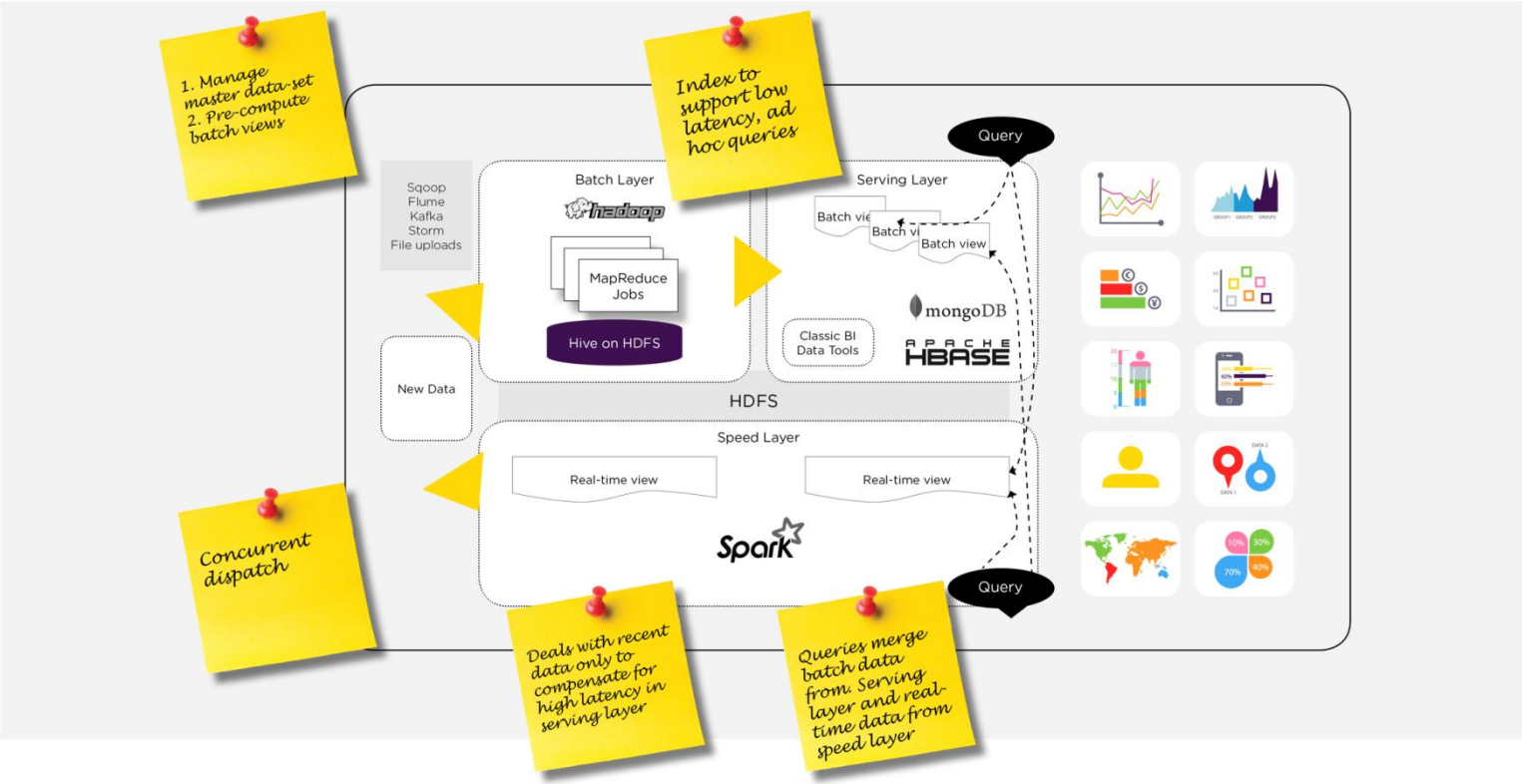
What about modelling?

- Spark 1.4.1 doesn't support native R functions like glm
  - An occasional workaround is to render Spark dataframes to local R dataframes and use native R packages on that
  - `localDf <- collect(simple_query)`
- Spark 1.5 starts to make in-roads on that gap
  - glm with Gaussian and Binomial distributions are supported

- But you can of course still run e.g. a linear regression in R using a Spark Context:
  - See NativeX's James Shanahan and Liang Dai's KDD 2015 tutorial at http://kdd2015-sparktutorial.droppages.com/



Aquila Insights Platform Architecture



Conclusion

- 3rd Phase Advanced Analytics is here and it works – SQL + R + Spark + Hadoop
- People that are predominantly skilled with SQL and R can reuse those skills quickly to query and analyse using Spark the very large datasets that can reside on Hadoop.
- It's hidden here but operating it really benefits from having a mix of tech skills and data analysis skills working closely together
  - Self-tuning a cluster's config to suit the analyses you're doing is powerful.

Credits and thanks!

- Canonical for Ubuntu
- The Apache Hadoop and Apache Spark teams
- The RStudio team and the Core R team
- Revolution Analytics
- R package developers
- Kalev H. Leetaru and Google for GDELT
- Ramnath Vaidyanathan for Slidify
- Andy South for rworldmap and Rolf Fredheim/QuantifyingMemory for examples
- James Shanahan and Liang Dai of NativeX for presenting regression analysis example at KDD 2015
- My company Aquila Insight; John Brodie, Warwick Beresford-Jones and our amazing analysts

Happy to talk!

- Why do you need to provide a SQL capability in your NoSQL world?
- Is this being used for production operations?
- Is the overall architecture production-ready?
- How much of the architecture is open source versus proprietary?
- How long did it take to build it?
- How performant is it?
- What are its limits?
- How much does it cost to run?
- Code at Github elektrifi/scotsoft2015

AQUILAINSIGHT
A new model for analytics

Edinburgh Office:
Argyle House, 3 Lady Lawson Street, Edinburgh EH3 9DR

London Office:
18 King William Street, Monument, London, EC4N 7BP

0131 290 2300

info@aquilainsight.com

/aquilainsight

@aquilainsight

www.aquilainsight.com