Single-Node Apache Spark + Twitter

Jason B. Hill - CA Technologies

May 19, 2016 - Datapalooza, Denver

https://github.com/hilljb/spark-jupyter

Agenda

1. Introduction to the (free) Twitter APIs
2. Examples of Previous Analysis on Twitter Data
3. What Was Lacking
4. Java, Spark, Python, Anaconda, Jupyter - Putting It All Together
5. Let’s Analyze Some Presidential Candidate Data From Twitter
6. Questions

Twitter: You (usually) don’t need the firehose

Firehose

- Average volume: 5,700 tweets per second (342K per minute)
- Peak volume: 254,644 tweets per second

Firehose Access

- Only given to certain companies (e.g., Crimson Hexagon, GNIP)
- An option: Pay Crimson Hexagon $24K per year for firehose analytics

Twitter: The (free) public APIs

REST API

- 15 requests every 15 minutes, each limited to 200 responses
- User info, tweets, followers, searches, lots more...

Streaming APIs

- 1% sample stream
- Filter stream

Twitter: The Filtered Streaming API

Some Details:

- Track users or terms
- Rate limited (at least 3K tweets per minute, often more)
- Often stays connected for months
- JSON
- Caveat: Many times a line response isn’t an entire JSON entity
- Caveat: Rate limit responses track total undelivered tweets since connection

Twitter: How to Get Access

- Twitter uses OAuth
  - Tokens do not expire
  - Apps are authenticated via a signature in an http request
  - Python packages: requests + requests_oauthlib
  - github.com/bear/python-twitter
- Log in to Twitter and go to apps.twitter.com
- For documentation and resources: dev.twitter.com

Examples

Example 1: NCAA March Hashtag Madness

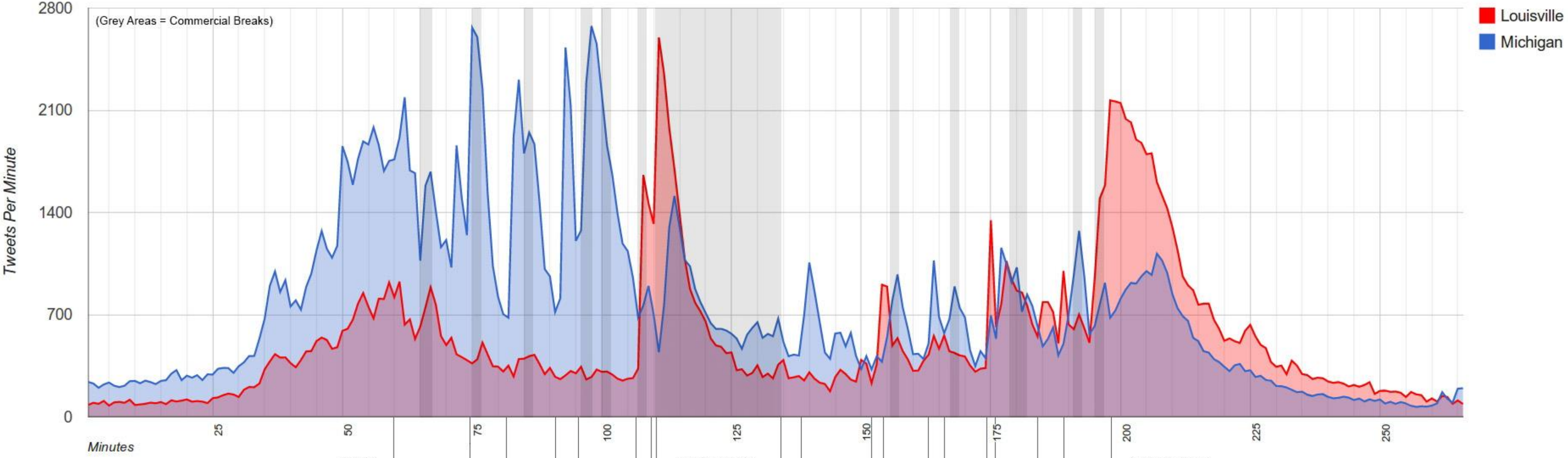
In 2013, a media agency client wanted to know:

- “How much data can we collect for free to analyze ourselves?”
- “Can we get minute-by-minute granularity?”

We decided to analyze the NCAA Tournament

- We used the filtered streaming API
- Every school, every game, every minute

2013 NCAA Championship: #1 Louisville (#l1c4, #uofl, #louisville) vs. #4 Michigan (#goblue, #michigan), Hashtagged Tweets Per Minute



Tip Off

Spike Albrecht (M) makes his 3rd 3-pointer in 3 minutes

Mitch McGary (M) block + Nik Stauskas (M) 3-pointer

Spike Albrecht (M) makes his 4th 3-pointer

Spike Albrecht (M) layup + Louisville T/O

Luke Hancock (L) makes 3 3-pointers in 1 minute

Luke Hancock (L) makes his 4th 3-pointer in 2 minutes, Peyton Siva (L) to Montrezl Harrell (L) alley-oop
Louisville takes the lead for the first time

Half Time Starts

Half Time Ends

Louisville Wins

Trey Burke (M) and Glenn Robinson III (M) make 6 free throws in 1 minute,
Louisville leads by 4

Luck Hancock (L) 3-pointer, Louisville leads by 10

Trey Burke (M) foul (clean block)

Luke Hancock (L) to Peyton Siva (L) alley-oop, Louisville leads by 5

Trey Burke (M) 2-point jumper, fouled, makes free throw, Louisville leads by 3

Tim Hardaway Jr. (M) dunks, Louisville leads by 2

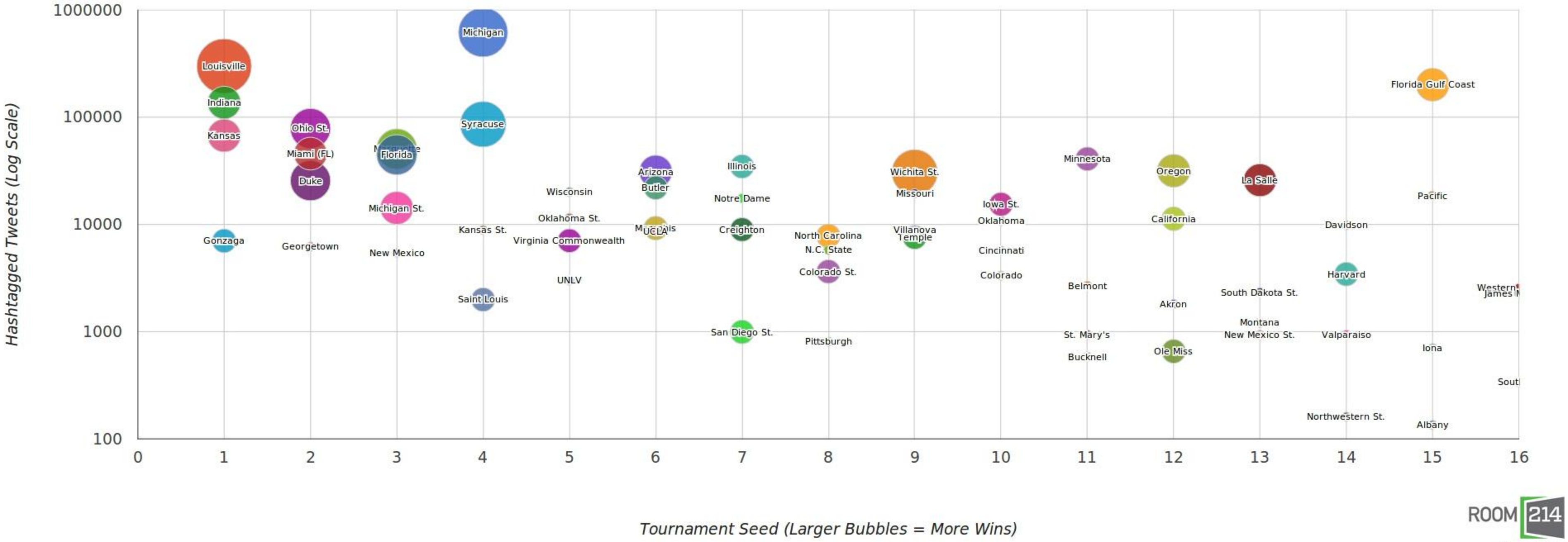
Trey Burke (M) makes 3-pointer, Louisville leads by 2

Chane Behanan (L) makes 2 free throws, Louisville leads by 2

Trey Burke (M) makes 3-pointer, Michigan leads by 2

ROOM 214

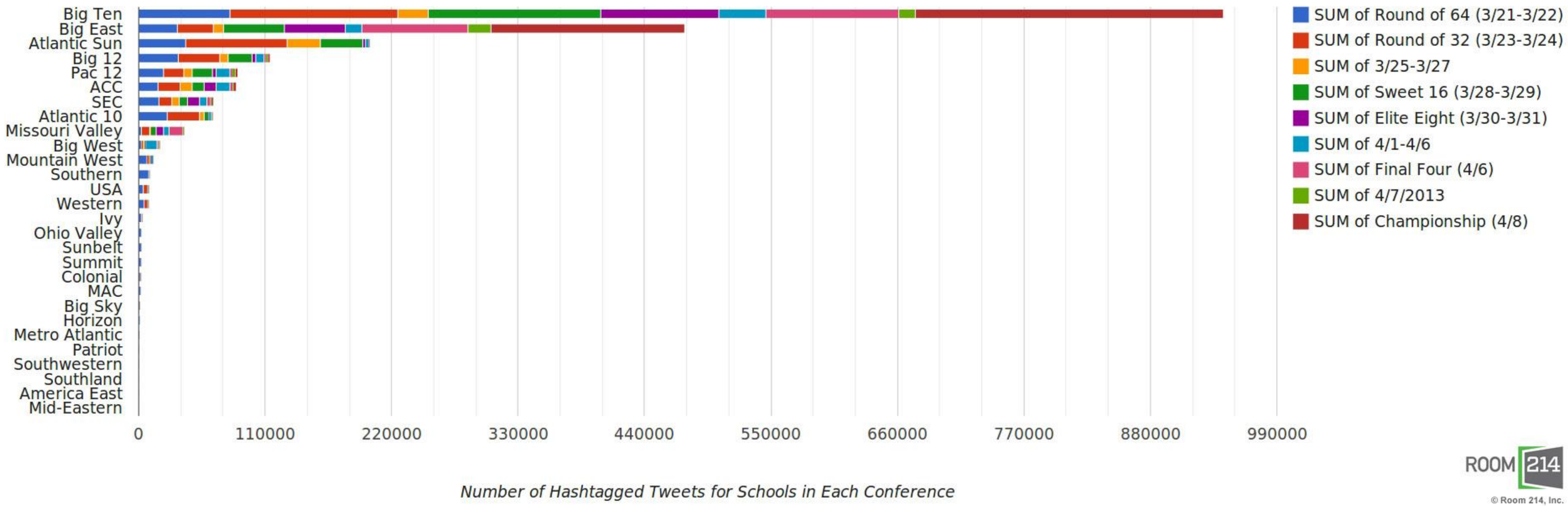
Tournament Seed vs. Wins vs. Hashtagged Tweets on Twitter During the 2013 NCAA Tournament



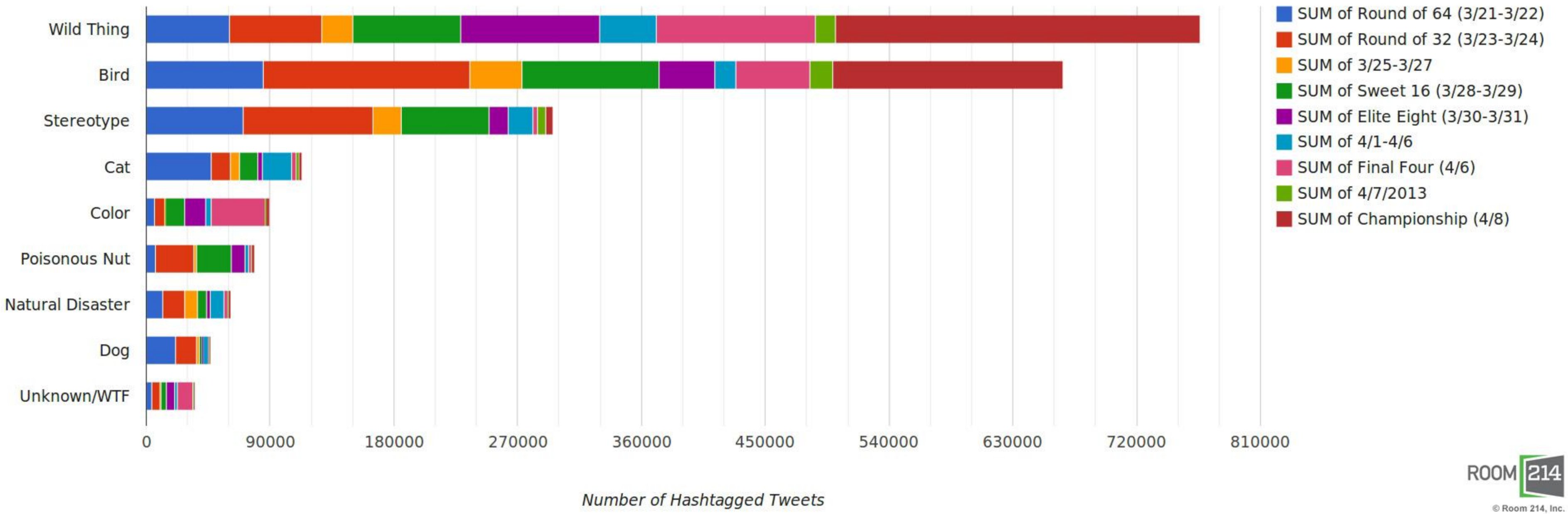
ROOM 214

© Room 214, Inc.

Hashtag Use on Twitter by Athletic Conference During the 2013 NCAA Tournament



Hashtagged Tournament Tweets vs. Mascot Type During the 2013 NCAA Tournament



ROOM 214

Example 2: 2014 Super Bowl

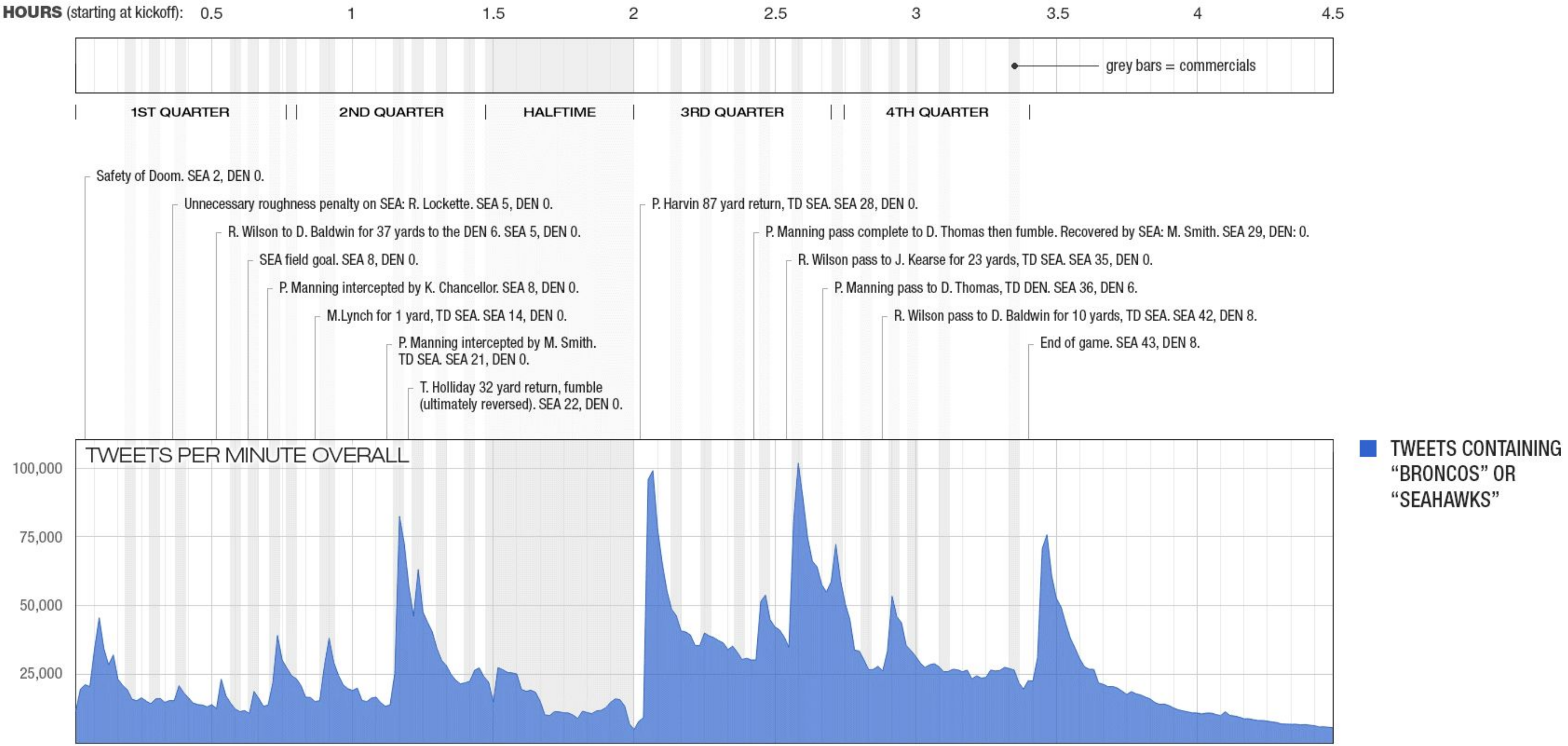
In 2014, the Denver Broncos lost to Seattle:

- 22-0 at halftime, 43-8 overall

We collected tweets mentioning both teams:

- 4,109,946 tweets containing “Broncos”
- 2,502,952 tweets containing “Seahawks”

Twitter During the 2014 Super Bowl



Safety of Doom. SEA 2, DEN 0.

Unnecessary roughness penalty on SEA: R. Lockette. SEA 5, DEN 0.

R. Wilson to D. Baldwin for 37 yards to the DEN 6. SEA 5, DEN 0.

SEA field goal. SEA 8, DEN 0.

P. Manning intercepted by K. Chancellor. SEA 8, DEN 0.

M.Lynch for 1 yard, TD SEA. SEA 14, DEN 0.

P. Manning intercepted by M. Smith. TD SEA. SEA 21, DEN 0.

T. Holliday 32 yard return, fumble (ultimately reversed). SEA 22, DEN 0.

P. Harvin 87 yard return, TD SEA. SEA 28, DEN 0.

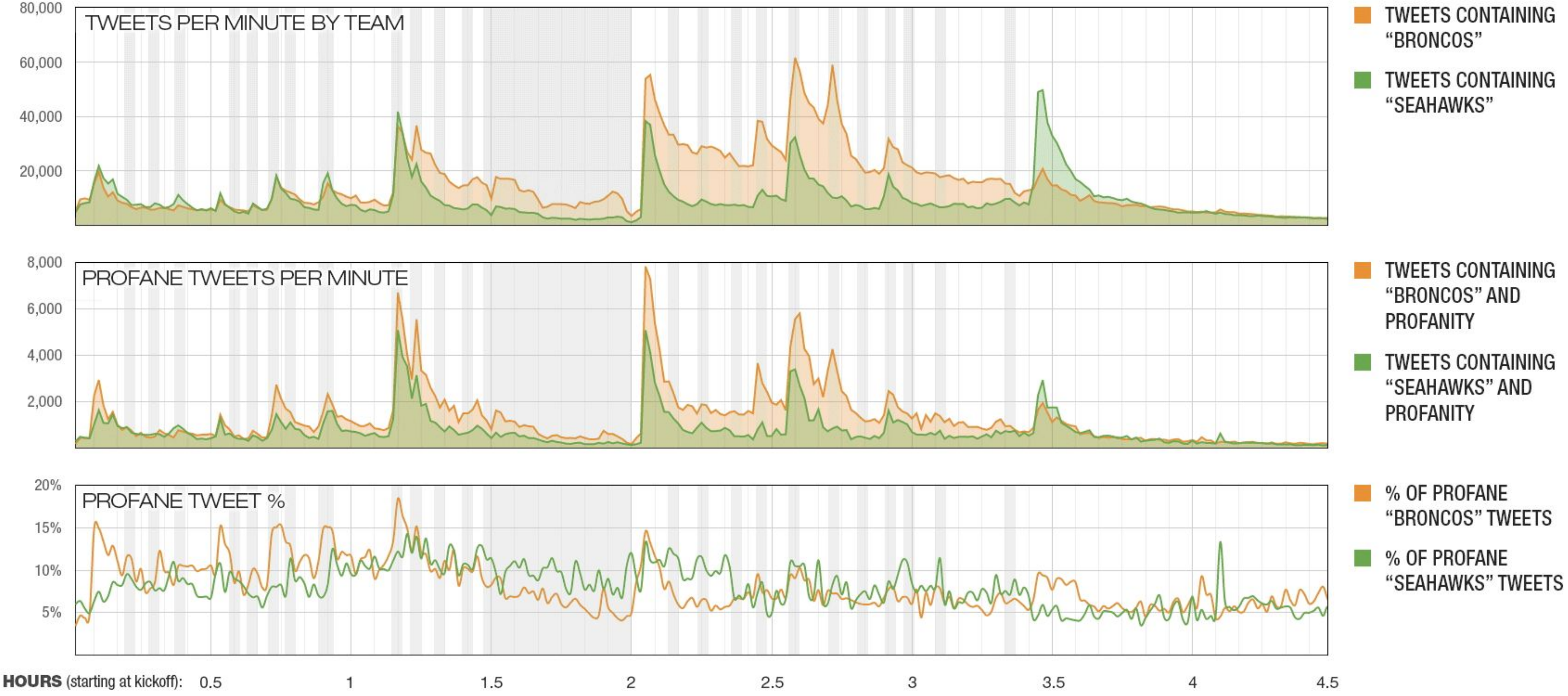
P. Manning pass complete to D. Thomas then fumble. Recovered by SEA: M. Smith. SEA 29, DEN: 0.

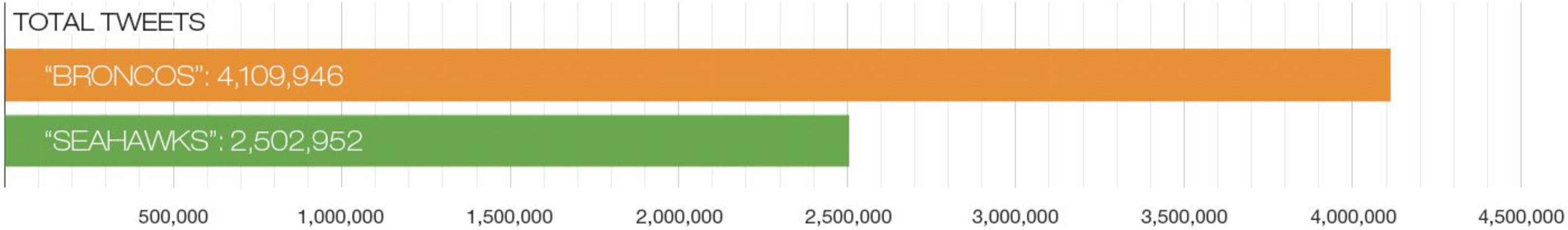
R. Wilson pass to J. Kearse for 23 yards, TD SEA. SEA 35, DEN 0.

P. Manning pass to D. Thomas, TD DEN. SEA 36, DEN 6.

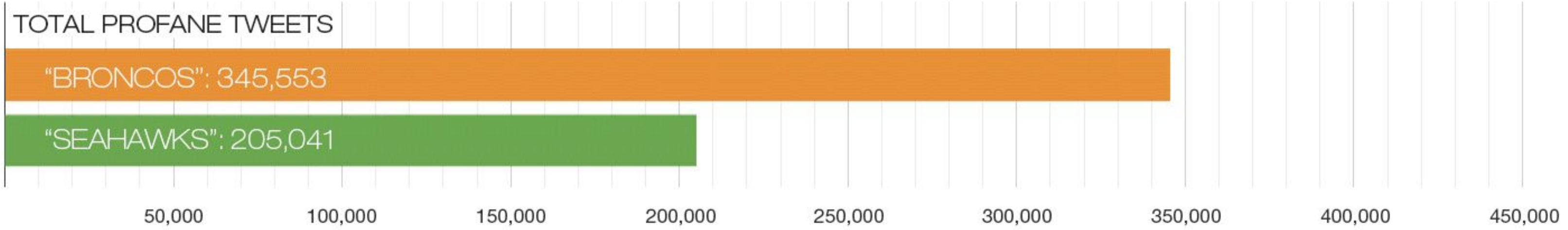
R. Wilson pass to D. Baldwin for 10 yards, TD SEA. SEA 42, DEN 8.

End of game. SEA 43, DEN 8.


TWEETS PER MINUTE BY TEAM
80,000
60,000
40,000
20,000
🟧 TWEETS CONTAINING "BRONCOS"
🟩 TWEETS CONTAINING "SEAHAWKS"
PROFANE TWEETS PER MINUTE
8,000
6,000
4,000
2,000
🟧 TWEETS CONTAINING "BRONCOS" AND PROFANITY
🟩 TWEETS CONTAINING "SEAHAWKS" AND PROFANITY
PROFANE TWEET %
20%
15%
10%
5%
🟧 % OF PROFANE "BRONCOS" TWEETS
🟩 % OF PROFANE "SEAHAWKS" TWEETS
HOURS (starting at kickoff):
0.5
1
1.5
2
2.5
3
3.5
4
4.5



TOTAL TWEETS



TOTAL PROFANE TWEETS

PERCENTAGE OF PROFANE TWEETS:

"BRONCOS": **7.88%**
"SEAHAWKS": **7.68%**

Example 3: Gooooooaaaaallllll!!!!!!!1

The 2014 World Cup on Twitter:

- 400 GiB of gzipped data
- 100 million (hashtagged) tweets
- The streaming API fed at rates over 30K tweets per minute
- Very few rate limit responses

goal

2,352,714 tweets

GOAL

522,353 tweets

Goal

505,513 tweets

GOAL!

125,084 tweets

goal!

52,194 tweets

goall

34,445 tweets

Goal!

16,868 tweets

GOAL!!!

11,034 tweets

Goall

5,947 tweets

goal!!

5,840 tweets

goal!!!

5,399 tweets

goooooaaaallllllll!!!!!

5,127 tweets

How many different spellings of “goal”?

- 3,997,679 tweets contained some variant of “goal”
- 22,430 distinct spellings
- 12,531 (55.9%) spellings were only tweeted once

What Was Lacking

Python Worked, But…

400 GiB of gzipped JSON isn’t quick to analyze

- json/simplejson is slow
- Pandas dataframes are nice, but they also…
  - Can have memory issues for large datasets
  - Pandasql isn’t as developed as other SQL environments
- Python is still largely single-threaded (see Dask, Cython with OpenMP, etc.)
- For me at least, SparkSQL has solved these problems

Let’s Build It

What Do We Need?

Java

- Visit Oracle’s Java SE Download page
- I’m using jdk-8u91-linux-x64.tar.gz
- JAVA_HOME=”/opt/jdk1.8.0_91”

What Do We Need?

Spark

- Visit the Apache Spark download page
- I’m using version 1.6.0 “pre-build for Hadoop 2.6 and later”
    - There seems to be a memory bug in Spark 1.6.1 that halts the Python/Jupyter kernel
- SPARK_HOME=”/opt/spark-1.6.0-bin-hadoop2.6”

What Do We Need?

Anaconda

- Visit the Continuum Analytics Anaconda download page
- Use the install script for your machine
- ‘conda’ should be in your PATH

What Do We Need?

Conda Environment

- I called my conda environment “spark-jupyter”
- pip/conda install: matplotlib, pandas, seaborn, jupyter, simplejson, numpy, requests, requests_oauthlib
    - Mac OS-X with matplotlib may require extra steps

What Do We Need?

Bash Script

- See the bin directory of the referenced repository
- Set your JAVA_HOME, and SPARK_HOME as needed
- Optionally: Set “driver-memory” to control available RAM for Spark
- This script can be modified slightly to run against a cluster