NBA Player Tracking Data

Ryan Speed
Felipe Chamma
Felipe Ferreira
Alex Morris

Data

- SportVU Tracking Data
- High frequency, high density tracking information
- Developed By Israeli Missile Tracking Experts
- Later Sold to NBA & Soccer Leagues

Experimental Environment

Distributed Environment

Configuration Details

Release label: emr-4.6.0

Hadoop distribution: Amazon 2.7.2

Applications: Hive 1.0.0, Spark 1.6.1, Zeppelin-Sandbox 0.5.6

Log URI: s3://aws-logs-876682794419-us-west-2/elasticmapreduce/

EMRFS consistent view: Disabled

Network and Hardware

Availability zone: us-west-2b

Subnet ID: subnet-33186356

Master: Running 1 m3.2xlarge (Spot: 0.15)

Core: Running 5 m3.2xlarge (Spot: 0.15)

Task: --

Data Pipeline

- Data originally scraped from stats.nba.com on a per play basis (500 separate JSON files per game) and stored on S3
  - Much overlap between plays
  - Not suitable for HIVE (not flat)
- Boto/Python Script pulls raw data from S3 and convert to 2 flat CSV files of player metadata and location data
- Data queried in both HIVE and Spark on EMR using Apache Zeppelin

Final Process

Raw Scraped JSON Files (500 per game)

Python/Boto script

Deduplication

2x Flat CSV Files (player meta, location moments)

HIVE

Aggregate (x, y) by player

Cluster players

Spark

Aggregate (x, y) by player

Cluster players

Processed Data Example

Can be loaded directly from S3 with wildcards in Spark

All Buckets / dcproject / 2014-10-29

Name	Storage Class	Size
0021400004_moments.csv.gz	Standard	10.8 MB
0021400004_players.csv	Standard	1002 bytes
0021400005_moments.csv.gz	Standard	10.3 MB
0021400005_players.csv	Standard	941 bytes
0021400006_moments.csv.gz	Standard	8.1 MB
0021400006_players.csv	Standard	992 bytes
0021400007_moments.csv.gz	Standard	9.7 MB
0021400007_players.csv	Standard	953 bytes
0021400008_moments.csv.gz	Standard	10.1 MB
0021400008_players.csv	Standard	990 bytes
0021400009_moments.csv.gz	Standard	9.8 MB
0021400009_players.csv	Standard	969 bytes
0021400010_moments.csv.gz	Standard	5.4 MB
0021400010_players.csv	Standard	964 bytes
0021400011_moments.csv.gz	Standard	10.2 MB
0021400011_players.csv	Standard	975 bytes
0021400012_moments.csv.gz	Standard	10.2 MB
0021400012_players.csv	Standard	971 bytes

Moments Data

FINISHED

```
%sql
select * from location_table limit 10
```

game_id	team_id	player_id	x	y
21,400,004	1,610,612,749	201,162	42.62491	18.10144
21,400,004	1,610,612,749	202,336	46.89332	24.26152
21,400,004	1,610,612,749	203,114	45.87236	31.90785
21,400,004	1,610,612,749	202,688	23.55151	24.61808
21,400,004	1,610,612,749	203,953	34.7505	24.22319
21,400,004	-1	-1	64.09499	24.57804
21,400,004	1,610,612,766	2,744	43.09823	22.30679
21,400,004	1,610,612,766	101,107	51.64268	21.98719
21,400,004	1,610,612,766	202,689	58.20318	23.41105
21,400,004	1,610,612,766	202,362	31.76621	44.52863

HIVE

```
CREATE EXTERNAL TABLE players (player_id STRING, team STRING,
first_name STRING, last_name STRING, position STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION 's3n://dcproject-public/players/';

CREATE EXTERNAL TABLE locations (row_number STRING, game_id STRING,
unix_time STRING, quarter INT, team_id STRING, player_id STRING,
game_clock FLOAT, shot_clock FLOAT, x FLOAT, y FLOAT, z FLOAT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION ‘s3n://dcproject-public/locations/';

CREATE TABLE locations_part (game_id STRING, quarter INT, team_id
STRING, game_clock FLOAT, shot_clock FLOAT, x FLOAT, y FLOAT, z FLOAT)
PARTITIONED BY (player_id STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
LOCATION 's3n://dcproject-public/locations_part/';
```

HIVE Final Query

```
select concat(players.first_name,'_', players.last_name) name, l.x, l.y, l.cnt

from

(select player_id, x, y, count(*) cnt from (select player_id, round(x, 0) x, round(y, 0) y
from locations

where

player_id <> -1) a group by player_id, x, y) l left join players

on l.player_id = players.player_id order by name, x, y limit 20;
```

Spark

- Create empty 50 length RDD cross joined with 94 length RDD and convert to data frame of blank x, y coordinates

- Cross join this player table to get empty entires for each player

- Left join this with rounded and grouped data to get full counts of players per square foot

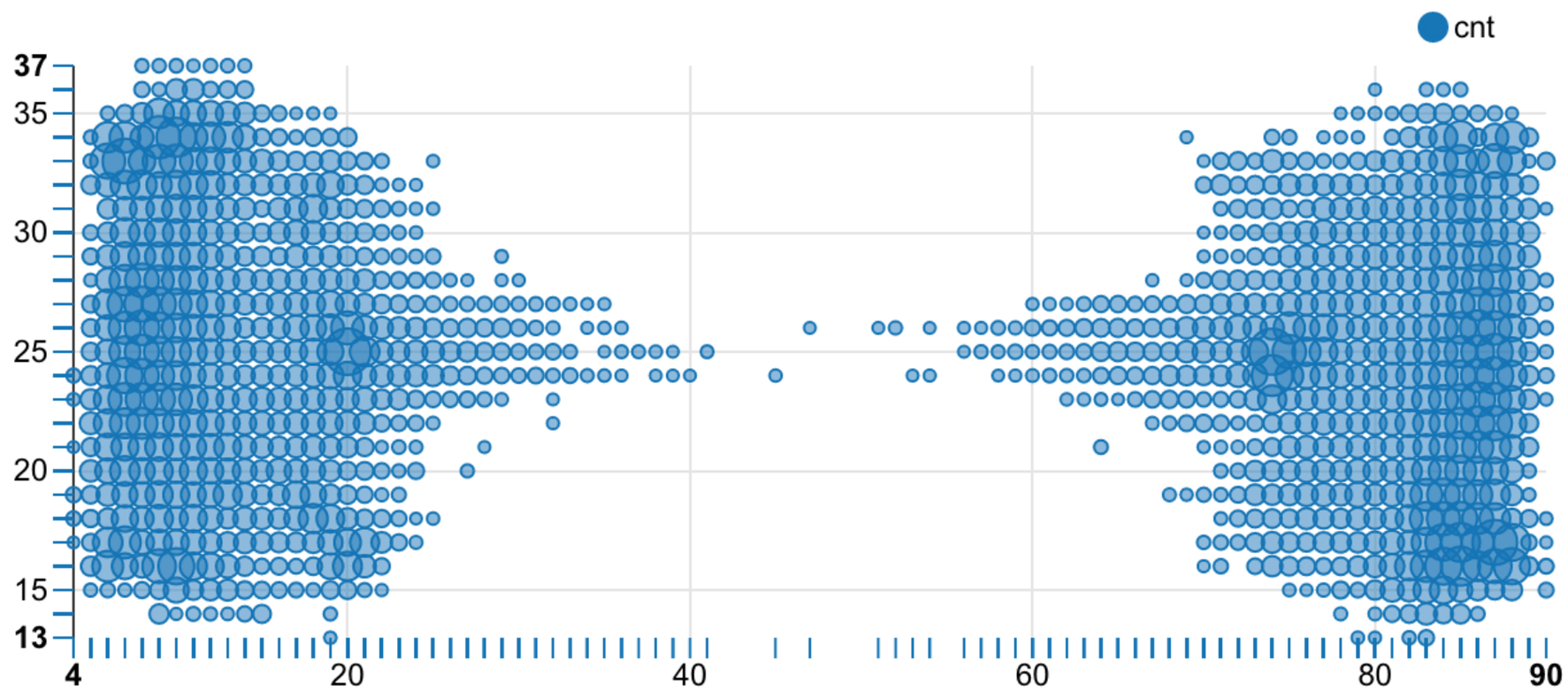
- Query takes 583s to run

Spark

```
%sql
select * from (select avg(cnt) cnt, x, y, position from player_loc_count group by x, y,
position) l where position = 'C' order by cnt desc limit 1000
```

FINISHED

settings



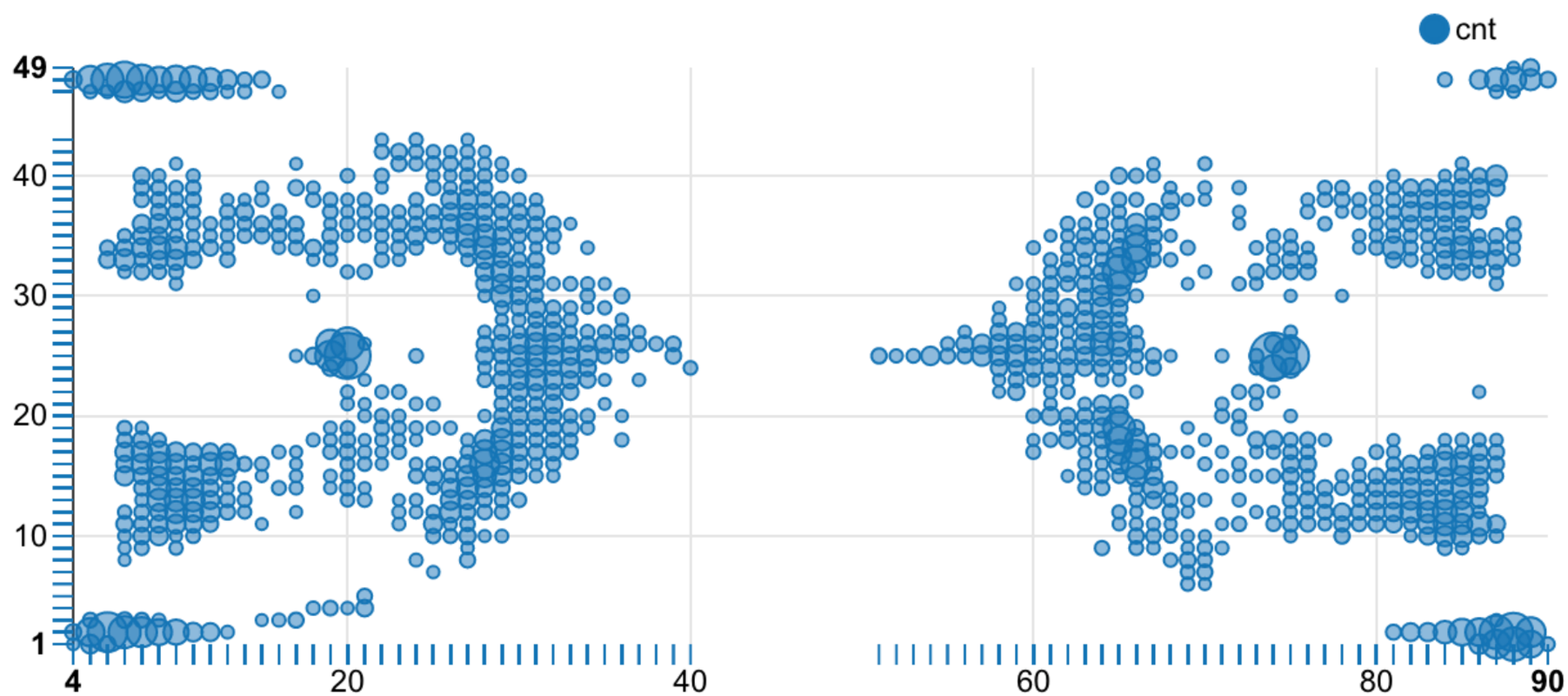
Centre's aggregated top 1000 coordinates

Spark

```
%sql
select * from (select avg(cnt) cnt, x, y, position from player_loc_count group by x, y,
position) l where position = 'G' order by cnt desc limit 1000
```

FINISHED

settings

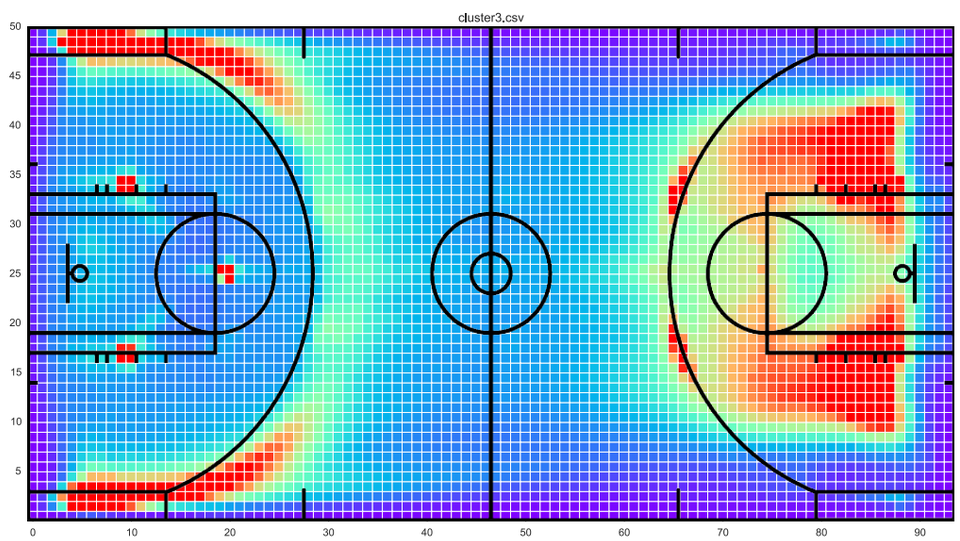


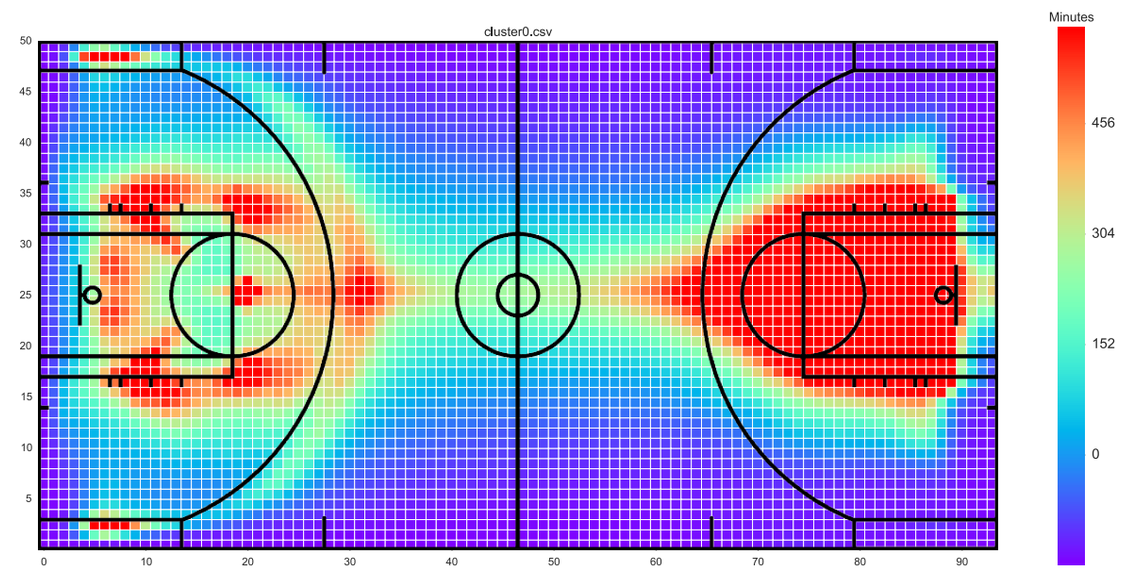
Guards aggregated top 1000 coordinates

Clustering

- Map to 55 x 91 matrix for each player with elements containing counts per moment spent in particular square foot of court
- Flatten matrix into 1 dimensional Numpy array
- Using Spark clustering algorithms to cluster groups of similar players

Final Cluster Examples





Future Work

- Try a different number of clusters
- Subset the data to compare plays with different outcomes or players with different line ups

Challenges

- Flipping the coordinates
- Tuning Spark Jobs for optimum performance
- Programming (clustering for example) using a tool thats designed for querying

Resources

- Tutorial to work with S3 and hive directly:
  - https://blog.mustardgrain.com/2010/09/30/using-hive-with-existing-files-on-s3/


Questions?