

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



Moments Data

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

The Problem

- Aggregate counts of moments spent in each square foot of court
- Cluster players by location distribution to find similar players

Experimental Environment



Local Implementation

- Map Reduce framework in Python
 - **Process_game()**
 - Aggregate count of unique (x, y) per player
 - **Process_result()**
 - Sums the counts of each (x, y)


Distributed Environment

Configuration Details

Release label: emr-4.6.0

Hadoop Amazon 2.7.2
distribution:

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 Disabled
consistent view:

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: --

Final Process

Raw scraped JSON files (500 per game)

Deduplication
↓ Python/Boto script

2x flat CSV files (player meta, location moments)

HIVE

Aggregate (x, y) by player

Cluster players

Spark

Aggregate (x, y) by player

Cluster players



















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

Processed Data Example

Can be loaded directly from S3 with wildcards in Spark

[All Buckets](#) / [dcproject](#) / [2014-10-29](#)

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

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 Aggregated 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;
```

Aaron_Gordon	-1.0	22.0	16
Aaron_Gordon	0.0	13.0	1
Aaron_Gordon	0.0	14.0	2
Aaron_Gordon	0.0	15.0	2
Aaron_Gordon	0.0	16.0	7
Aaron_Gordon	0.0	17.0	7
Aaron_Gordon	0.0	18.0	5
Aaron_Gordon	0.0	19.0	2
Aaron_Gordon	0.0	20.0	2
Aaron_Gordon	0.0	21.0	13
Aaron_Gordon	0.0	22.0	20
Aaron_Gordon	0.0	24.0	1

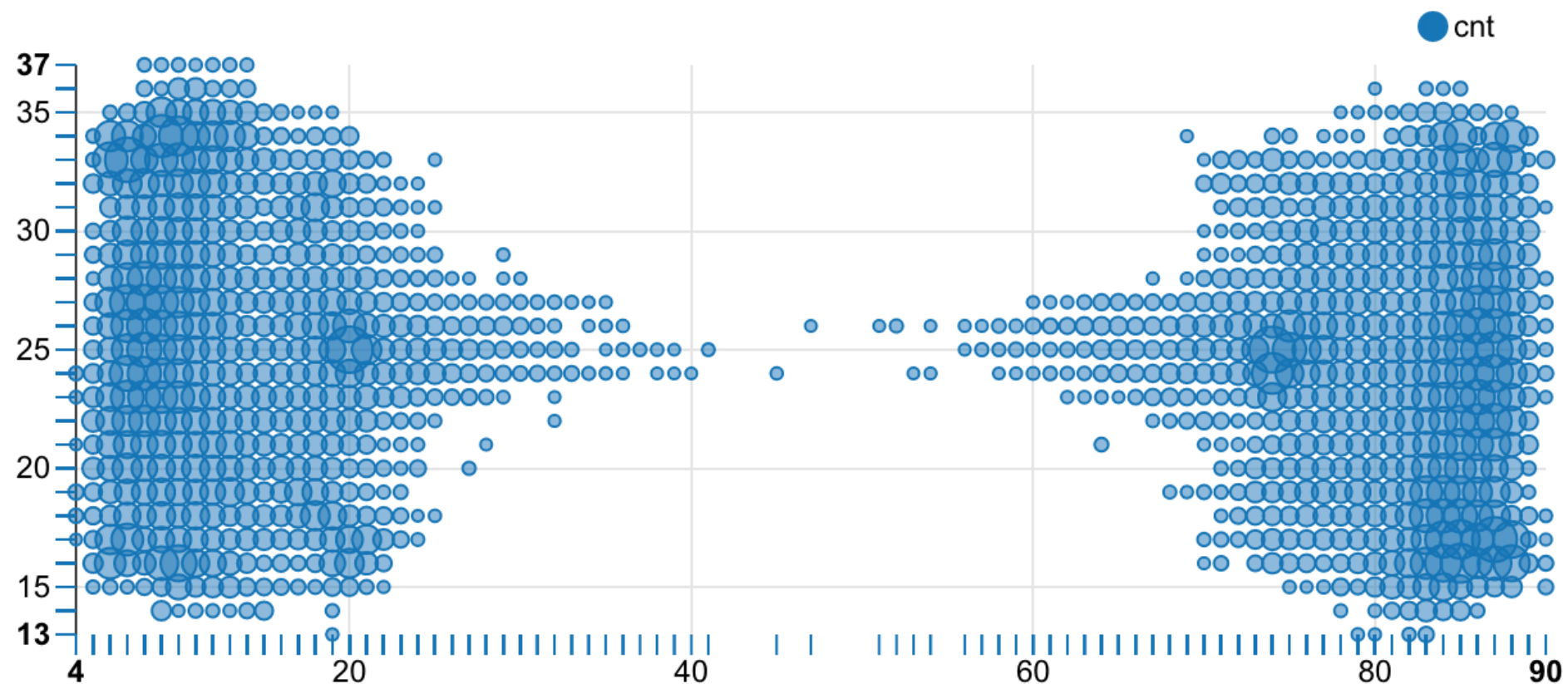
Spark

- 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 execute

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 ▶ 🔍 📖 ⚙️



Centre's aggregated top 1000 coordinates

Spark

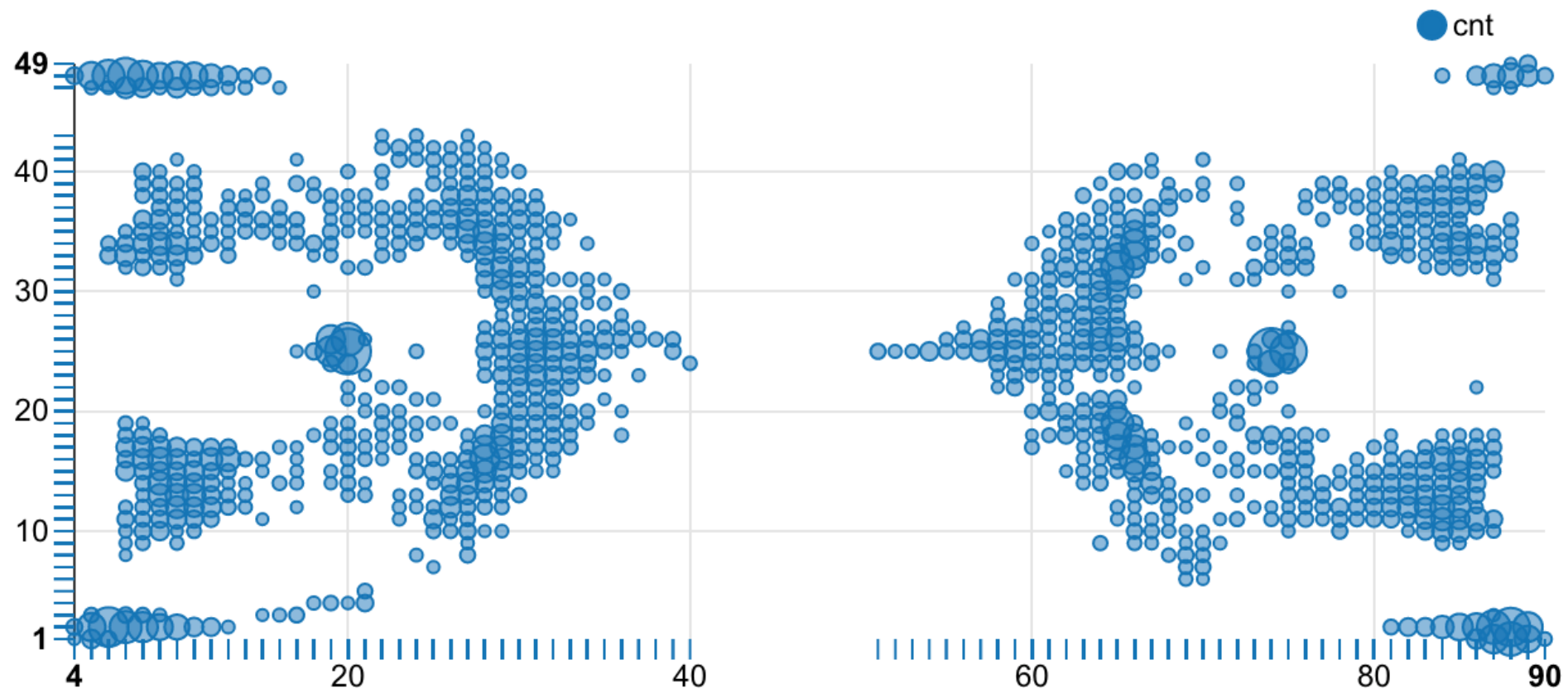
%sql

FINISHED ▶ ⌵ 📖 ⚙️

```
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
```



settings ▼



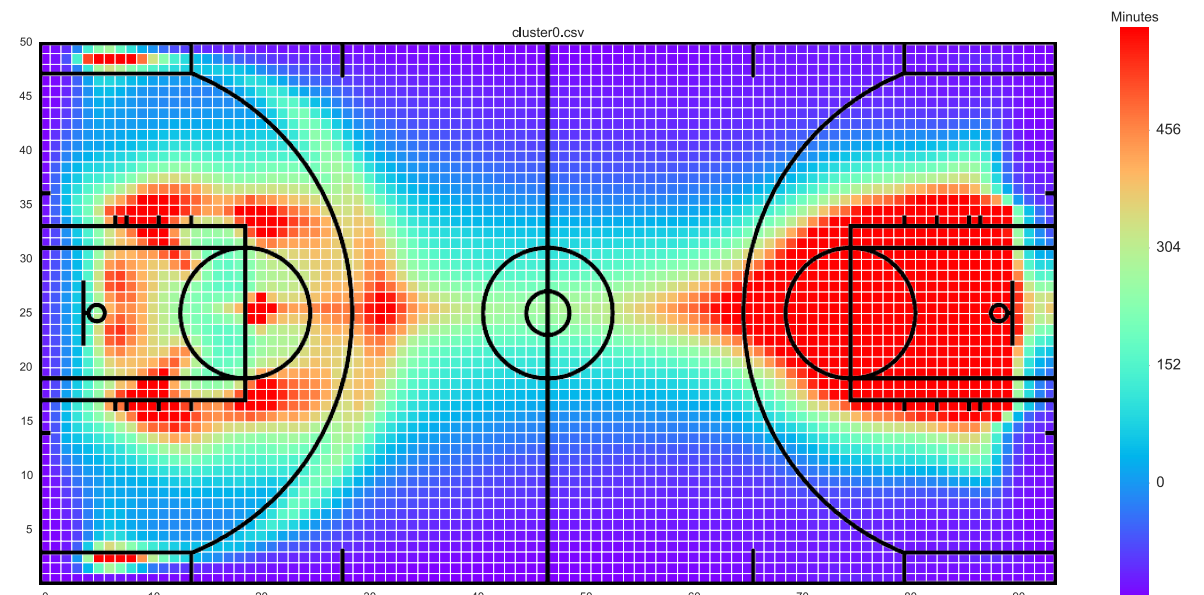
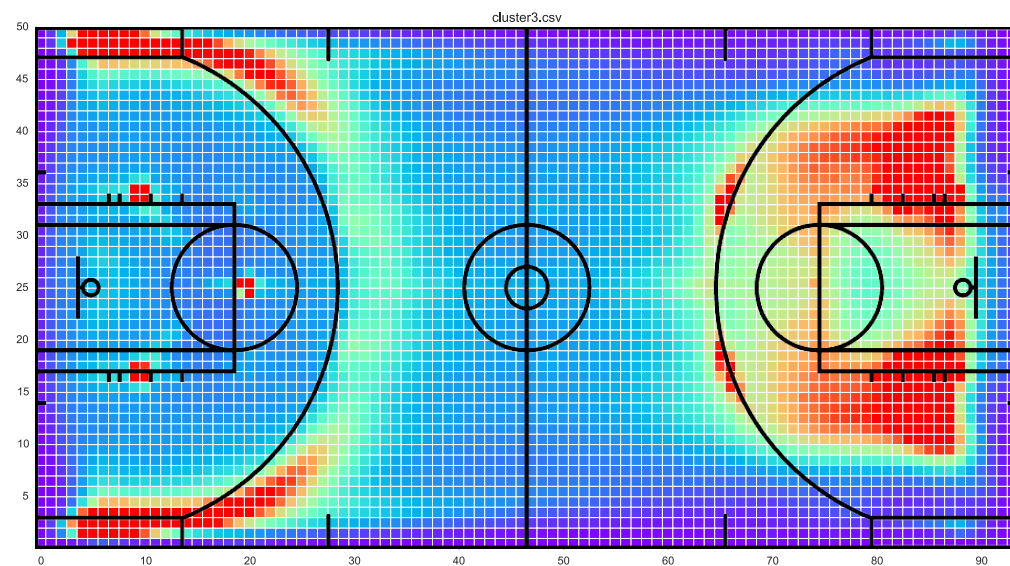
Guards aggregated top 1000 coordinates

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

- Aggregate player locations by their assigned cluster
- Heat map of 54 x 90 matrix plotted



Future Work

- Experiment with a different number of clusters
- Subset the data to compare plays with different outcomes or players with different line ups
- Clustering teams in order to see whether good teams and bad teams fall on different buckets

Challenges

- Flipping the coordinates correctly
 - Which way players shoot determined by coin toss
 - Data does not contain this information, must be inferred
- 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/>
- GitHub Repository
 - https://github.com/alexemorris/NBA_location_aggregation

Questions?

