Ecosistema de soporte a proyectos Big Data

Práctica 1 - Casos de uso y persistencia

José Manuel Bustos Muñoz

Índice

1. Ejemplo de persistencia

- **Datos escogidos, análisis e hipótesis, y preparación de los datos.**

Los datos escogidos para la práctica corresponden a la liga de baloncesto NBA. Se dividen en 3 ficheros csv:

- "**players.csv**" con información de los jugadores que han pasado por la liga como podría ser el año de nacimiento, universidad o la altura y peso del jugador.
- "**player_data.csv**" que tiene información también de los jugadores donde la información relevante al anterior csv sería que añade el año inicio y fin de la carrera del jugador en la liga y la posición en la que juega.
- "**Seasons_stats.csv**" que contiene todas las estadísticas en los distintos apartados del juego de cada jugador en cada temporada desde el inicio de la liga.

Lo que se pretende analizar en la práctica es la influencia de la entrada en la temporada 1979-1980 de la línea de 3 en la liga, y cómo podría haber cambiado el propio juego. Para ello nos vamos a quedar con el último csv "Seasons_stats.csv" que es el que contiene mayor información y sobre todo la información que pensamos que puede ser útil para poder analizar y llegar a alguna conclusión.

Para dar un primer vistazo a los datos utilizamos un notebook de jupyter con Python, donde cargamos los datos y realizamos unos primeros análisis de los dataset (se adjunta con la práctica un archivo html con el notebook).

```
# Carga de los datos escogidos para la práctica
# Datos de seasons stats
df_season_stats = pd.read_csv('Seasons_stats.csv', sep=',')

print("num_rows: %d\tColumnas: %d\n" % (df_season_stats.shape[0], df_season_stats.shape[1]) )
print("Columnas:\n", list(df_season_stats.columns))
```

```
num_rows: 24691 Columnas: 53

Columnas:
 ['Unnamed: 0', 'Year', 'Player', 'Pos', 'Age', 'Tm', 'G', 'GS', 'MP', 'PER', 'TS%', '3PAr', 'FTr', 'ORB%', 'DRB%', '
TRB%', 'AST%', 'STL%', 'BLK%', 'TOV%', 'USG%', 'blanl', 'OWS', 'DWS', 'WS', 'WS/48', 'blank2', 'OBPM', 'DBPM', 'BPM',
'VORP', 'FG', 'FGA', 'FG%', '3P', '3PA', '3P%', '2P', '2PA', '2P%', 'eFG%', 'FT', 'FTA', 'FT%', 'ORB', 'DRB', 'TRB',
'AST', 'STL', 'BLK', 'TOV', 'PF', 'PTS']
```

```
# primer vistazo a los datos de seasons stats
df_season_stats.head()
```

	Unnamed: 0	Year	Player	Pos	Age	Tm	G	GS	MP	PER	...	FT%	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS
0	0	1950.0	Curly Armstrong	G-F	31.0	FTW	63.0	NaN	NaN	NaN	...	0.705	NaN	NaN	NaN	176.0	NaN	NaN	NaN	217.0	458.0
1	1	1950.0	Cliff Barker	SG	29.0	INO	49.0	NaN	NaN	NaN	...	0.708	NaN	NaN	NaN	109.0	NaN	NaN	NaN	99.0	279.0
2	2	1950.0	Leo Barnhorst	SF	25.0	CHS	67.0	NaN	NaN	NaN	...	0.698	NaN	NaN	NaN	140.0	NaN	NaN	NaN	192.0	438.0
3	3	1950.0	Ed Bartels	F	24.0	TOT	15.0	NaN	NaN	NaN	...	0.559	NaN	NaN	NaN	20.0	NaN	NaN	NaN	29.0	63.0
4	4	1950.0	Ed Bartels	F	24.0	DNN	13.0	NaN	NaN	NaN	...	0.548	NaN	NaN	NaN	20.0	NaN	NaN	NaN	27.0	59.0

Como se quiere analizar el impacto de la llegada de la linea de 3 puntos a la NBA, se divide el dataset escogido en 2, creando un dataframe con los datos anteriores a dicha fecha y otro dataframe con los datos posteriores.
Además en ambos dataframes se ven los datos vacíos para en este caso sustituirlos por el valor de la mediana para dicho atributo, y así evitar tratar en los análisis posteriores con missing values.

```
# sustituir los valores vacíos que haya en las columnas del DF por el valor más adecuado. En este caso se piensa que
# la mejor opción para los datos vacíos es rellenarlos con la mediana de ese atributo y así influirá menos en los
# futuros análisis.
df_season_NO3_final = df_season_NO3.fillna(df_season_NO3.median())
df_season_NO3_final.head()
```

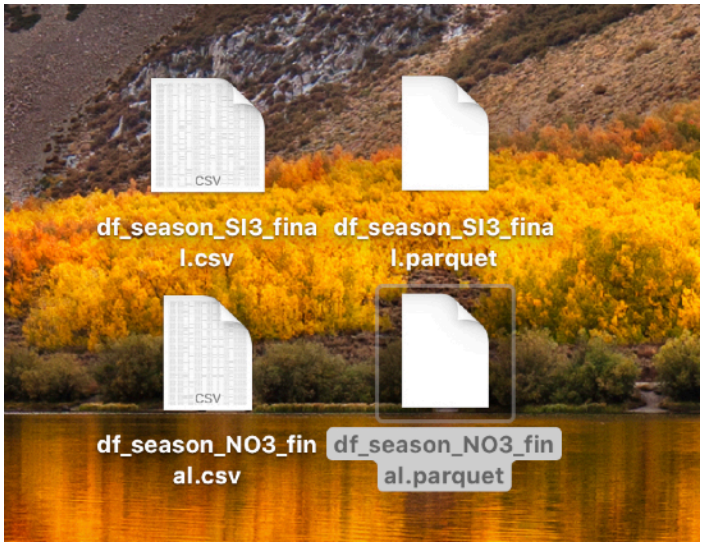
	Year	Player	MP	PER	TS%	FTr	USG%	FG	FGA	FG%	...	FT	FTA	FT%	TRB	AST	STL	BLK	TOV	PF	PTS
0	1950.0	Curly Armstrong	1308.0	12.8	0.368	0.467	19.3	144.0	516.0	0.279	...	170.0	241.0	0.705	221.0	176.0	41.0	14.0	95.5	217.0	458.0
1	1950.0	Cliff Barker	1308.0	12.8	0.435	0.387	19.3	102.0	274.0	0.372	...	75.0	106.0	0.708	221.0	109.0	41.0	14.0	95.5	99.0	279.0
2	1950.0	Leo Barnhorst	1308.0	12.8	0.394	0.259	19.3	174.0	499.0	0.349	...	90.0	129.0	0.698	221.0	140.0	41.0	14.0	95.5	192.0	438.0
3	1950.0	Ed Bartels	1308.0	12.8	0.312	0.395	19.3	22.0	86.0	0.256	...	19.0	34.0	0.559	221.0	20.0	41.0	14.0	95.5	29.0	63.0
4	1950.0	Ed Bartels	1308.0	12.8	0.308	0.378	19.3	21.0	82.0	0.256	...	17.0	31.0	0.548	221.0	20.0	41.0	14.0	95.5	27.0	59.0

Cuando se tienen ambos dataframes con el formato adecuado, se genera un fichero .csv por cada uno, y además generamos también un fichero en formato .parquet.

Una vez tenemos los dos DF finales, de datos anteriores a 1980 y los datos posteriores, generamos ambos .csv y parquet para subirlos al HDFS, almacenarlos y analizarlos.

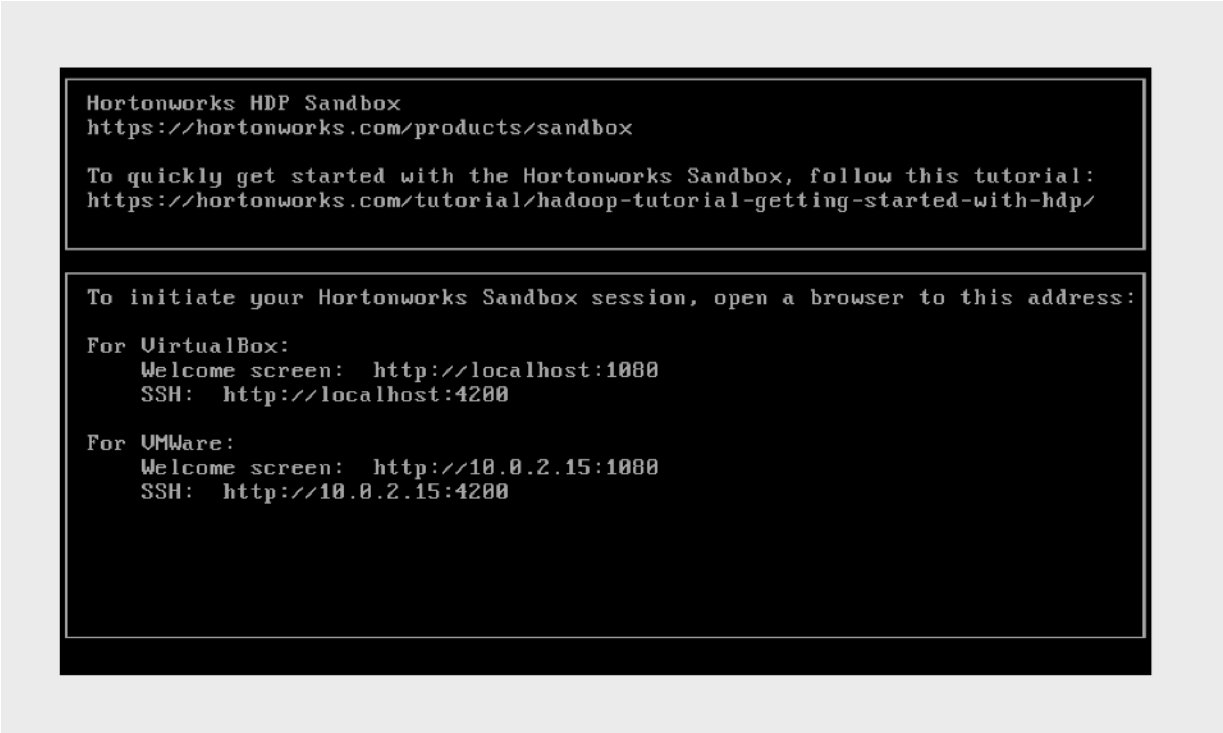
```
df_season_NO3_final.to_csv('df_season_NO3_final.csv')
table = pa.Table.from_pandas(df_season_NO3_final, preserve_index=True)
pq.write_table(table, 'df_season_NO3_final.parquet')
df_season_SI3_final.to_csv('df_season_SI3_final.csv')
table_si = pa.Table.from_pandas(df_season_SI3_final, preserve_index=True)
pq.write_table(table_si, 'df_season_SI3_final.parquet')
```

Se generan los 4 ficheros comentados, que serán los que suban al HDFS para realizar los siguientes pasos de la práctica.

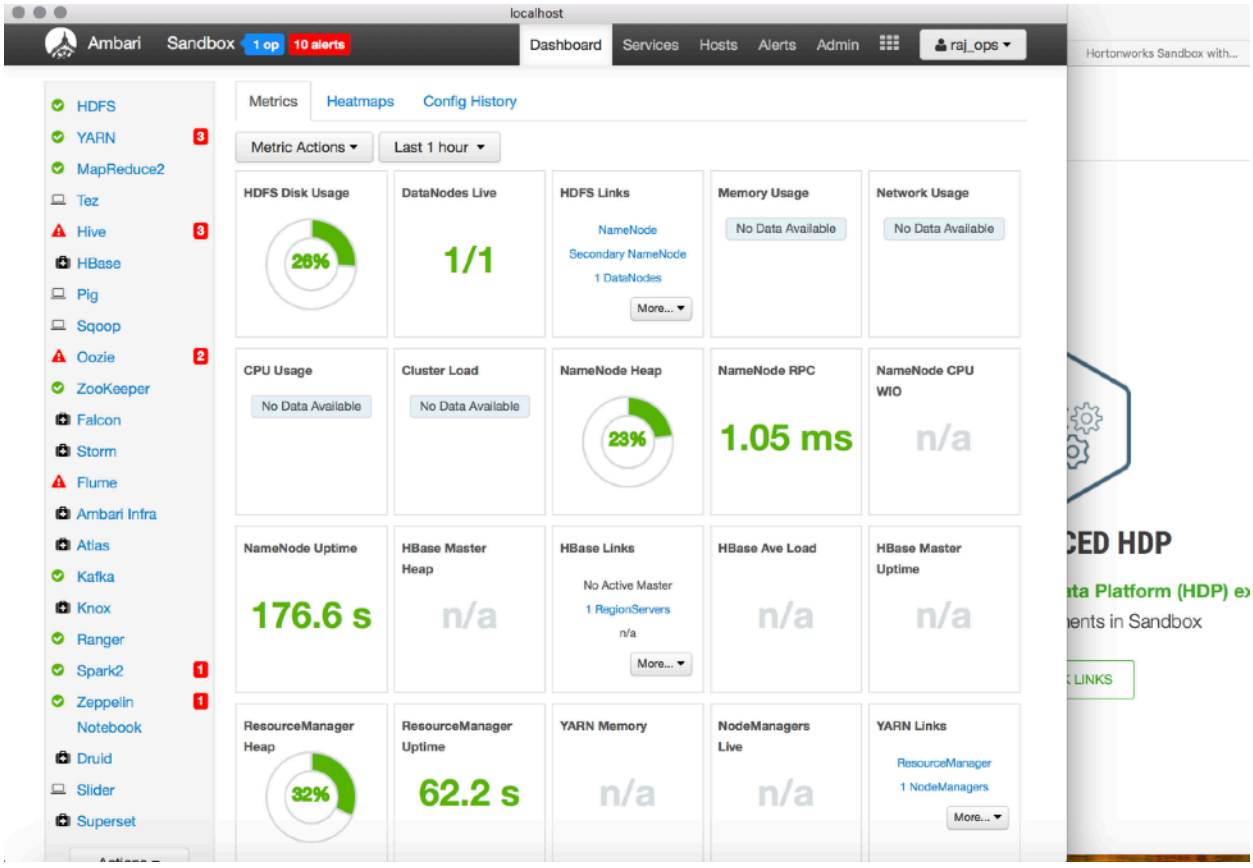


- **Almacenamiento de los datos.**

Para la práctica usamos la sandbox de Hortonworks con virtualbox.



Se accede a través de la url: “http://localhost:1080" al Ambari donde se tiene el panel de administración para arrancar, parar y administrar todos los servicios, o acceder al gestor visual de Hive o el sistema de archivos HDFS.
Con la url: “http://localhost:4200" accedemos en el navegador a la terminal de la máquina virtual.



Por terminal del pc local mandamos a la máquina virtual los ficheros generados en el punto anterior mediante scp.

```
MBP-de-Jose:practica 1 ecosistema josemanuel$ scp -P 2222 df_season_NO3_final.parquet root@127.0.0.1:
root@127.0.0.1's password:
df_season_NO3_final.parquet                                   100%  257KB  55.7MB/s   00:00
MBP-de-Jose:practica 1 ecosistema josemanuel$ scp -P 2222 df_season_SI3_final.parquet root@127.0.0.1:
root@127.0.0.1's password:
Permission denied, please try again.
root@127.0.0.1's password:
Permission denied, please try again.
root@127.0.0.1's password:
df_season_SI3_final.parquet                                   100%  846KB  77.7MB/s   00:00
MBP-de-Jose:practica 1 ecosistema josemanuel$ scp -P 2222 df_season_SI3_final.csv root@127.0.0.1:
root@127.0.0.1's password:
df_season_SI3_final.csv                                       100% 3534KB  90.5MB/s   00:00
MBP-de-Jose:practica 1 ecosistema josemanuel$ scp -P 2222 df_season_NO3_final.csv root@127.0.0.1:
root@127.0.0.1's password:
df_season_NO3_final.csv                                       100%  938KB  81.6MB/s   00:00
MBP-de-Jose:practica 1 ecosistema josemanuel$
```

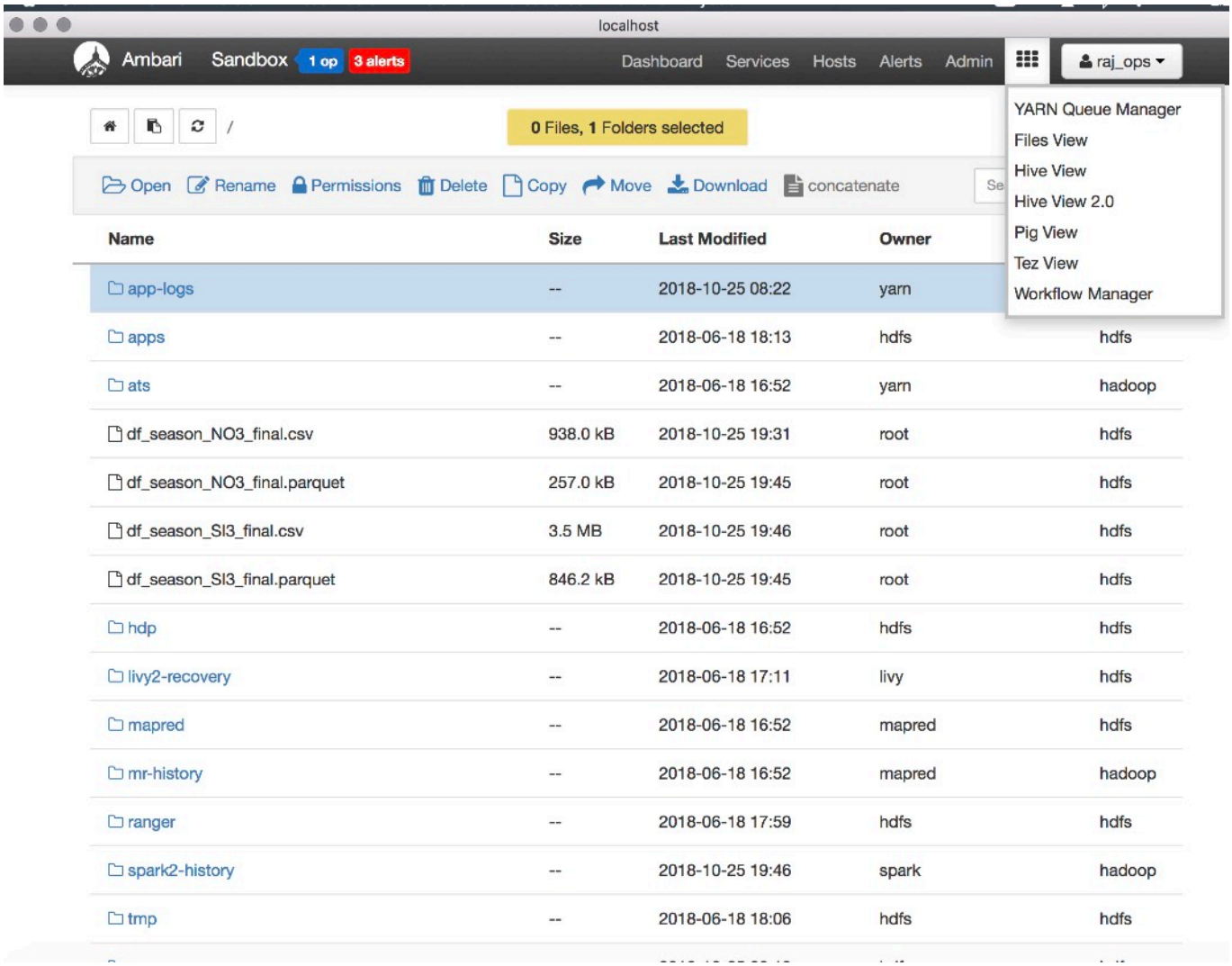
Desde la terminal de la sandbox en el navegador podemos hacer un ‘ls’ y comprobar que se han subido los archivos.

```
[root@sandbox-hdp ~]# ls
anaconda-ks.cfg  df_season_NO3_final.csv  df_season_NO3_final.parquet  df_season_SI3_final.csv  df_season_SI3_final.parquet  root
[root@sandbox-hdp ~]#
```

Una vez tenemos los archivos en la máquina virtual de Hortonworks, los subimos al sistema de HDFS mediante el comando ‘hdfs dfs -put’. Así hacemos con los 4 ficheros.

```
[root@sandbox-hdp ~]# hdfs dfs -put df_season_NO3_final.parquet /
[root@sandbox-hdp ~]# hdfs dfs -put df_season_SI3_final.parquet /
[root@sandbox-hdp ~]# hdfs dfs -put df_season_SI3_final.csv /
```

Podemos comprobar desde Ambari accediendo a la opción de 'Files View' que efectivamente los archivos ya están subidos en HDFS.



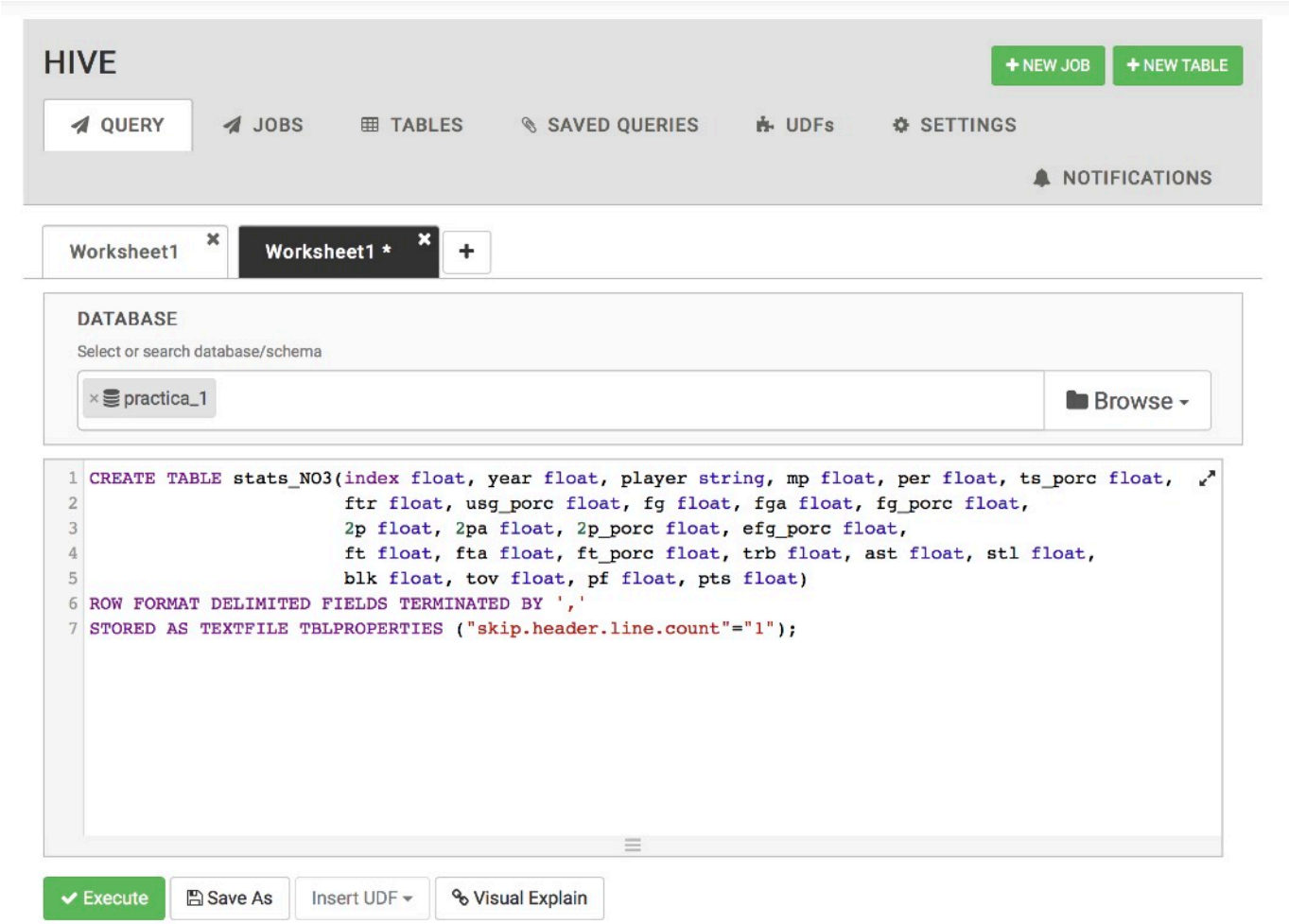
Una vez tenemos los ficheros para la práctica en HDFS, creamos una BBDD para trabajar, y luego accediendo a Hive creamos las tablas necesarias.

Creamos la BBDD en Hive desde la terminal de la máquina virtual.

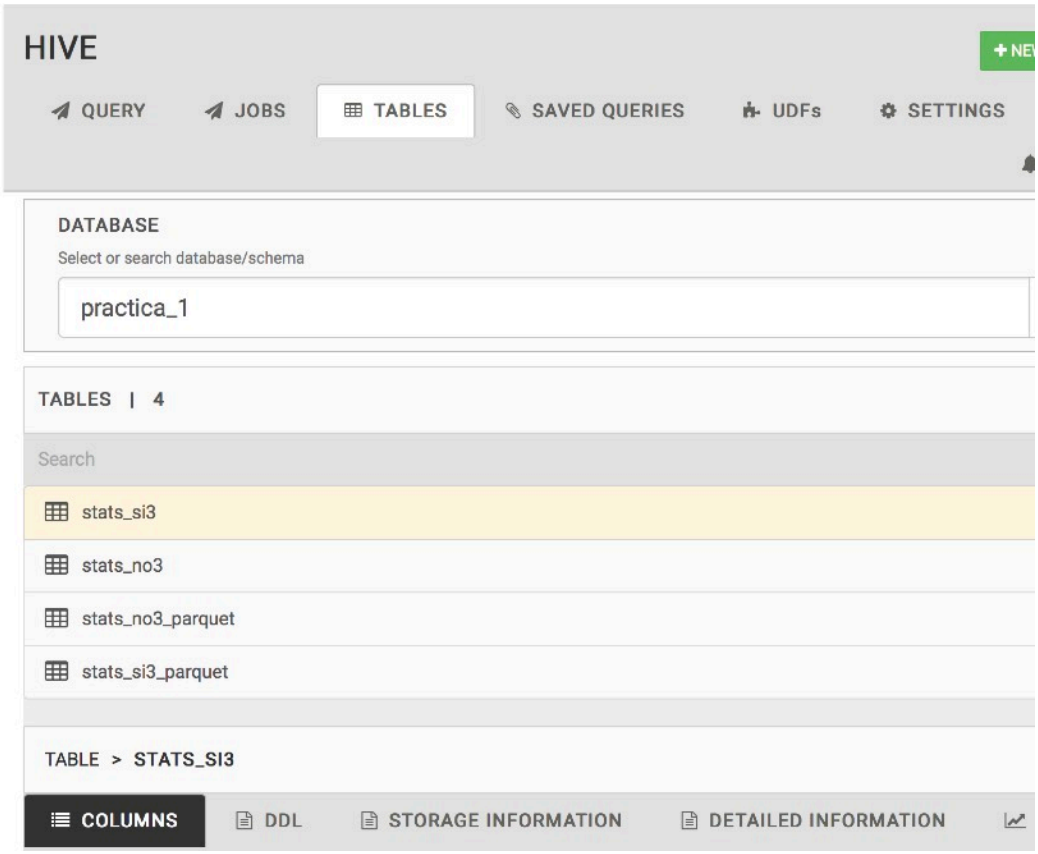
```
Creates a table. Pass a table name, and a set of column family
[root@sandbox-hdp ~]# ls
anaconda-ks.cfg  df_season_NO3_final.csv  df_season_NO3_final.parquet  df_season_SI3_final.csv  df_season_SI3_final.parquet  root
[root@sandbox-hdp ~]# hive
log4j:WARN No such property [maxFileSize] in org.apache.log4j.DailyRollingFileAppender.

Logging initialized using configuration in file:/etc/hive/2.6.5.0-292/0/hive-log4j.properties
hive> create database practica_1;
OK
Time taken: 3.918 seconds
hive> show databases;
OK
default
foodmart
practica_1
Time taken: 0.165 seconds, Fetched: 3 row(s)
hive> use practica_1;
OK
Time taken: 0.251 seconds
hive>
```

Para crear las tablas utilizamos la interfaz de Hive en Ambari. Vamos a crear 4 tablas, dos para los ficheros csv, y otras dos para los ficheros parquet.

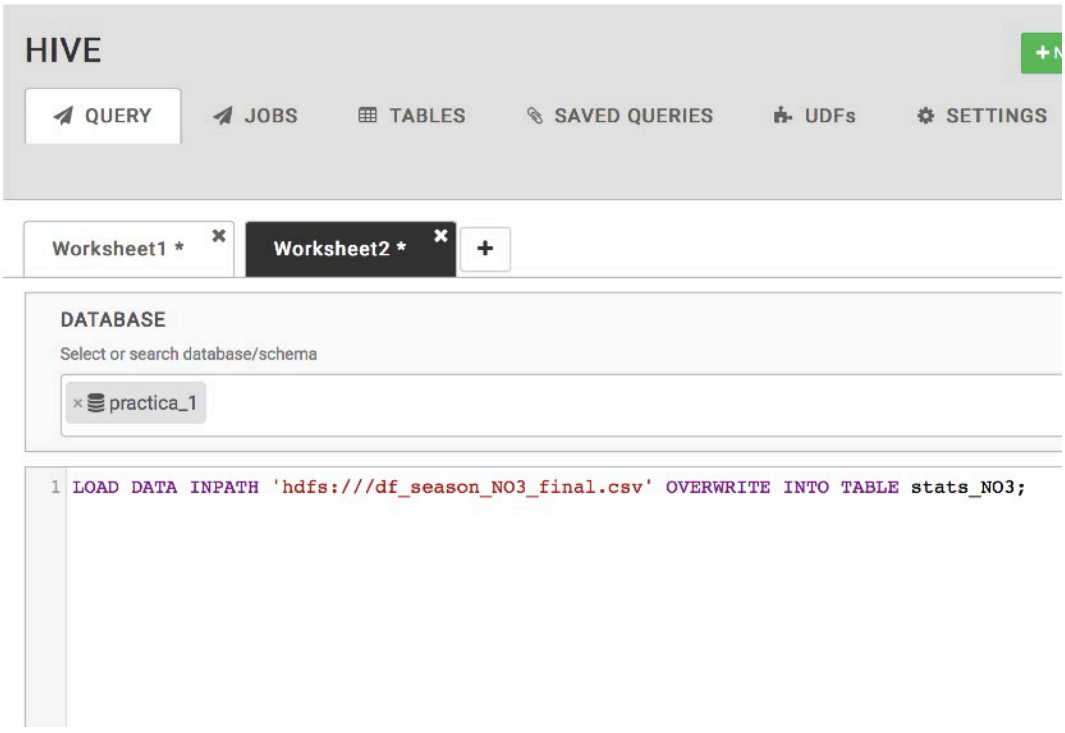
Vemos en la imagen la creación de una de las tablas, en este caso para los datos anteriores a la introducción de la linea de 3. Cada tabla tendrá definidos en su creación los campos necesarios según el dataset generado para cargarla posteriormente.



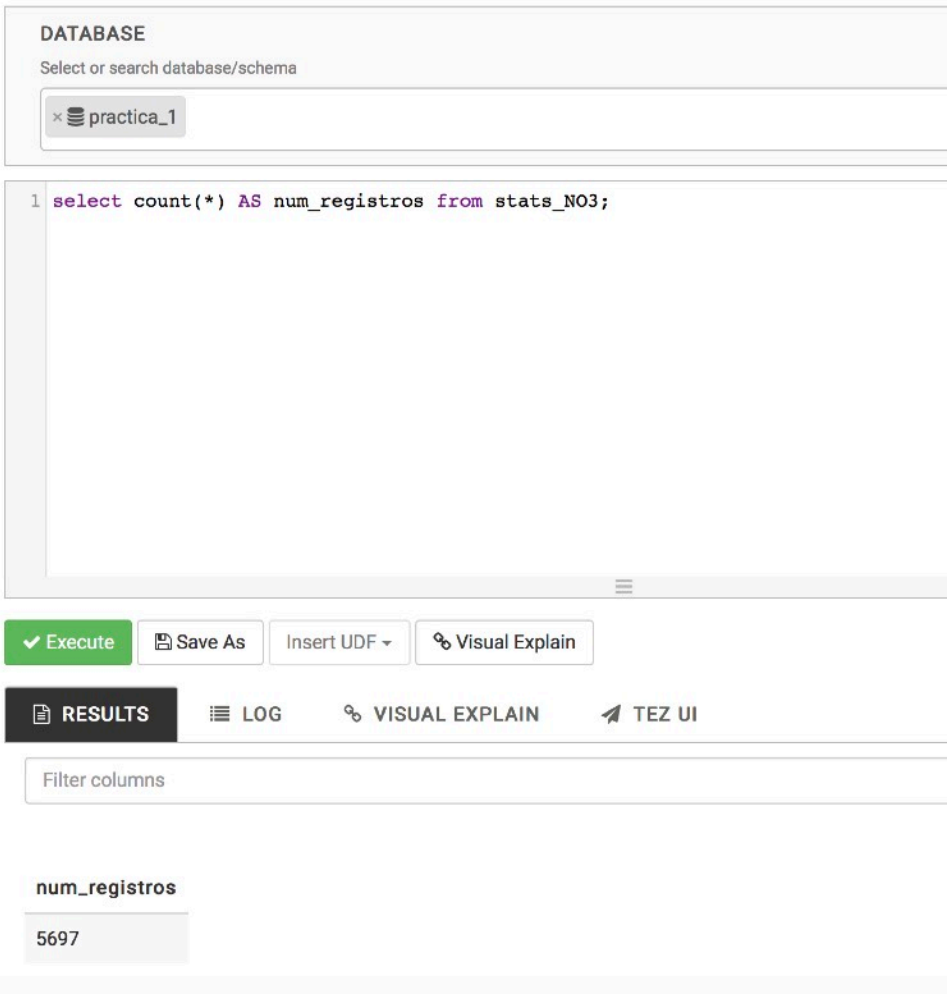
Se generan las 4 tablas comentadas, y puede verse dicha creación en la misma interfaz de Hive en la pestaña de 'TABLES' seleccionando luego la base de datos donde se hayan generado estas tablas. En la siguiente imagen se pueden apreciar que las 4 están creadas.

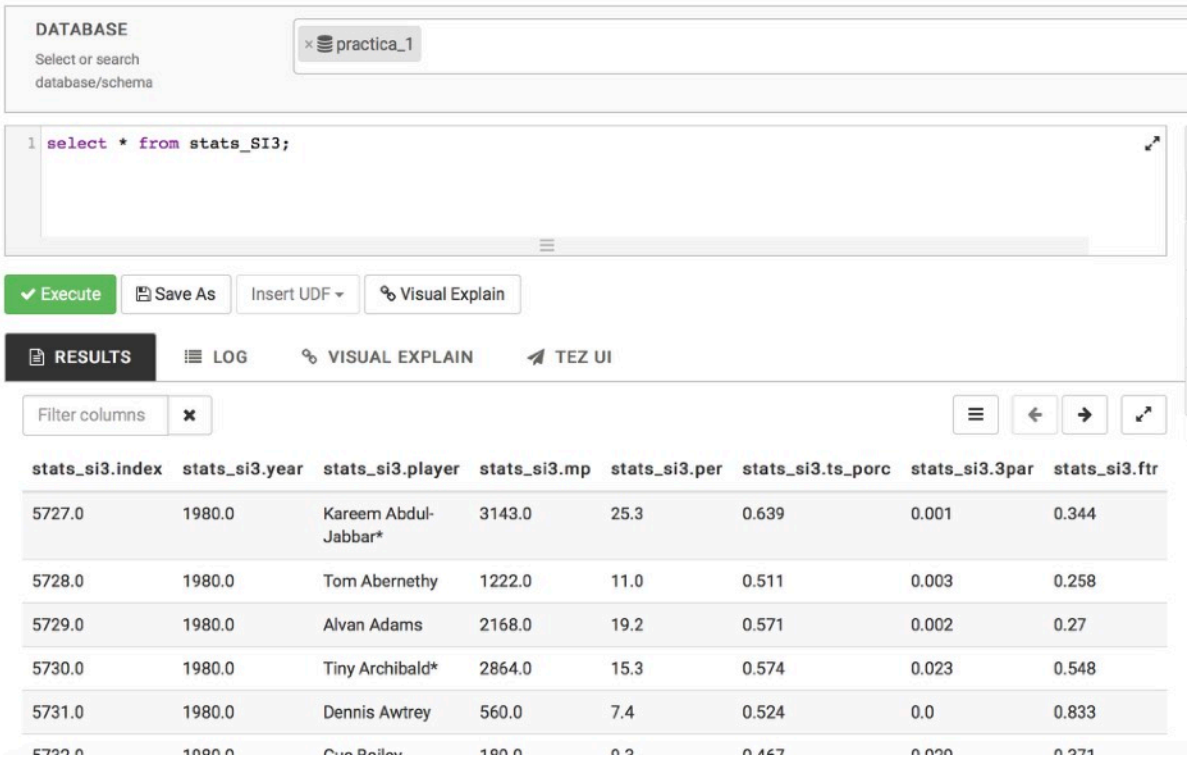


Una vez creadas las tablas, vamos a rellenarlas con los datos del archivo que les corresponde. En la imagen está el ejemplo de carga de la tabla para los datos sin tiro de 3. Se realiza con el siguiente comando desde Hive: “LOAD DATA INPATH ‘hdfs:///df_season_NO3_final.csv' OVERWRITE INTO TABLE stats_NO3;”



Cuando están cargadas las tablas tiramos alguna query para asegurarnos de que los datos se han cargados correctamente, como podría ser hacerle un Count a una de las tablas, o un select a una tabla para visualizar todos los datos incluidos.





- **Análisis realizado y conclusiones.**

Lo primero que vamos a realizar es sacar la media en los apartados del juego más importantes del dataset con las estadísticas de los años anteriores a la implantación del triple. Para ello usamos la query de la siguiente imagen.
Cómo no vamos a realizar un análisis muy exhaustivo, redondeamos la salida de los datos por comodidad visual.
La query lo que hace es sacar de la tabla en cuestión la media redondeada de varios apartados del juego, para saber por ejemplo la media de puntos anotados en el periodo de años que comprende dicha tabla.

```
select round(avg(pts)) as Puntos, round(avg(trb)) as Rebotes, round(avg(ast)) as Asistencias,
round(avg(stl)) as Robos, round(avg(blk)) as Tapones, round(avg(pf)) as Faltas,
round(avg(tov)) as Perdidas, round(avg(fga)) as Tiros_intentados,
round(avg(2pa)) as Intentos_2, round(avg(fta)) as Tiros_libres_intentados
from stats_no3;
```

A continuación vemos la salida de la query anterior, con las medias de este dataset en los siguientes apartados del juego: puntos, rebotes, asistencias, robos, tapones, faltas personales, pérdidas de balón, tiros intentados, tiros de 2 intentados y tiros libres intentados.

RESULTS | LOG | VISUAL EXPLAIN | TEZ UI

Filter columns

puntos	rebotes	asistencias	robos	tapones	faltas	perdidas	tiros_intentados	intentos_2	tiros_libres_intentados
590.0	293.0	129.0	44.0	18.0	143.0	97.0	529.0	529.0	178.0

Realizamos la misma query para los datos del dataset que contiene las estadísticas ya con el tiro de 3 puntos. Usamos la siguiente query:

```
select round(avg(pts)) as Puntos, round(avg(trb)) as Rebotes, round(avg(ast)) as Asistencias,
round(avg(stl)) as Robos, round(avg(blk)) as Tapones, round(avg(pf)) as Faltas,
round(avg(tov)) as Perdidas, round(avg(fga)) as Tiros_intentados,
round(avg(2pa)) as Intentos_2, round(avg(fta)) as Tiros_libres_intentados
from stats_si3;
```

Obtenemos los siguientes resultados:


RESULTS LOG VISUAL EXPLAIN TEZ UI

Filter columns

puntos	rebotes	asistencias	robos	tapones	faltas	perdidas	tiros_intentados	intentos_2	tiros_libres_intentados
486.0	204.0	111.0	39.0	24.0	108.0	73.0	401.0	338.0	124.0

Como se puede observar, a partir de 1980 y la implantación de la línea de 3, prácticamente todos los apartados del juego se vieron disminuidos (partimos de la base de que el dataset original tiene unos datos coherentes y válidos): menos puntos, menos rebotes, menos asistencias, y por ejemplo menos tiros intentados.

¿Esto que puede significar? Lo que significa es que a partir del tiro de 3 bajo el ritmo del juego, por lo tanto cada equipo fue disminuyendo las posesiones en un partido, los intentos de tiro, y en consecuencia se disminuyen las estadísticas en casi todos los apartados del juego.

Una posible explicación es que obviamente lanzar de lejos como puede ser el tiro de 3 es más complicado que lanzar cerca del aro, por tanto a medida que el tiro de 3 se fue convirtiendo en un arma más del juego, los equipos tuvieron que ir centrándose en hacer un juego con el fin de encontrar los tiros más librados posibles, aquellos donde el tirador de lejos estuviera sólo.
Para ello el juego se ralentizó, y fue hacia una deriva más táctica.

Evidentemente esta es una conclusión a la que se ha llegado tras un análisis muy superficial, habría que profundizar mucho más y cruzar con más datos para poder hacer un análisis más certero.
Ahora para intentar validar esta conclusión, vamos a analizar los datos de la tabla que tiene las estadísticas de 1980 a 2017, ya con el tiro de 3, viendo como el uso del tiro de 3 a lo largo de los años ha podido influir.

En la siguiente query sacamos las estadísticas para esta tabla, esta vez sacando también los intentos de tiros de 3 porque queremos ver si el aumento de este tiro influye. Para verlo de forma adecuada vamos a agrupar los datos por año y así ver la evolución desde 1980 hasta ahora.

```
1 select year, round(avg(pts)) as Puntos, round(avg(fga)) as Tiros_Intentados,
2 round(avg(3pa)) as Intentos_de3, round(avg(2pa)) as Intentos_de2, round(avg(fta)) as Intentos_de1,
3 round(avg(trb)) as Rebotes, round(avg(ast)) as Asistencias, round(avg(tov)) as Perdidas
4 from stats_si3
5 group by year order by year asc;
```

Mostramos los resultados obtenidos agrupando por año:

RESULTS | LOG | VISUAL EXPLAIN | TEZ UI

Filter columns

year	puntos	tiros_intentados	intentos_de3	intentos_de2	intentos_de1	rebotes	asistencias	perdidas
1980.0	612.0	509.0	15.0	494.0	156.0	253.0	144.0	105.0
1981.0	593.0	487.0	11.0	476.0	158.0	237.0	142.0	102.0
1982.0	592.0	483.0	12.0	470.0	155.0	237.0	134.0	95.0
1983.0	569.0	471.0	12.0	460.0	149.0	236.0	135.0	100.0
1984.0	617.0	495.0	13.0	482.0	167.0	241.0	147.0	99.0
1985.0	589.0	474.0	17.0	457.0	156.0	233.0	139.0	94.0
1986.0	569.0	458.0	18.0	441.0	156.0	225.0	135.0	91.0
1987.0	570.0	461.0	24.0	437.0	158.0	229.0	136.0	87.0
1988.0	529.0	430.0	25.0	405.0	142.0	213.0	127.0	81.0
1989.0	550.0	449.0	33.0	416.0	146.0	222.0	128.0	85.0
1990.0	547.0	448.0	33.0	414.0	146.0	222.0	129.0	81.0
1991.0	566.0	465.0	38.0	427.0	148.0	230.0	129.0	83.0
1992.0	525.0	437.0	38.0	399.0	133.0	218.0	124.0	76.0
1993.0	534.0	437.0	45.0	391.0	141.0	221.0	125.0	79.0
1994.0	498.0	415.0	48.0	368.0	130.0	211.0	118.0	76.0
1995.0	521.0	419.0	79.0	340.0	139.0	214.0	119.0	79.0
1996.0	477.0	386.0	76.0	310.0	127.0	199.0	110.0	74.0
1997.0	437.0	359.0	77.0	282.0	114.0	187.0	101.0	69.0

1998.0	458.0	383.0	62.0	321.0	125.0	198.0	107.0	72.0
1999.0	277.0	236.0	40.0	197.0	78.0	126.0	63.0	44.0
2000.0	485.0	410.0	68.0	341.0	126.0	212.0	111.0	74.0
2001.0	451.0	384.0	64.0	320.0	118.0	205.0	104.0	69.0
2002.0	483.0	412.0	75.0	337.0	120.0	213.0	112.0	70.0
2003.0	494.0	421.0	77.0	344.0	127.0	219.0	113.0	74.0
2004.0	428.0	368.0	68.0	300.0	110.0	195.0	98.0	66.0
2005.0	452.0	376.0	75.0	301.0	120.0	194.0	100.0	65.0
2006.0	458.0	374.0	76.0	298.0	124.0	193.0	97.0	66.0
2007.0	497.0	402.0	86.0	316.0	131.0	205.0	109.0	73.0
2008.0	450.0	369.0	81.0	287.0	112.0	192.0	99.0	62.0
2009.0	468.0	381.0	87.0	294.0	115.0	193.0	99.0	63.0
2010.0	468.0	383.0	85.0	298.0	115.0	196.0	99.0	64.0
2011.0	450.0	368.0	85.0	284.0	110.0	185.0	98.0	62.0
2012.0	365.0	309.0	69.0	240.0	85.0	161.0	80.0	53.0
2013.0	448.0	376.0	92.0	284.0	101.0	192.0	102.0	64.0
2014.0	442.0	365.0	96.0	269.0	103.0	187.0	97.0	62.0
2015.0	428.0	360.0	97.0	262.0	97.0	184.0	96.0	60.0
2016.0	466.0	385.0	111.0	274.0	106.0	198.0	102.0	63.0
2017.0	475.0	385.0	123.0	262.0	104.0	196.0	101.0	60.0

Si se miran los resultados obtenidos, se puede apreciar como de forma bastante clara a medida que pasan los años va disminuyendo la anotación y otros apartados estadísticos.
Y sobre todo, si vemos los intentos de 3 puntos, apreciamos como desde la implantación del tiro en 1980 prácticamente aumentan los intentos año a año, y por tanto el aumento del tiro de 3 influye en la disminución en otros aspectos del juego.

Se comprueba la hipótesis que ya al analizar los datos en ambas tablas comparadas se había visto.

2. Caso de uso 1

Entorno:
- Recepción de información de dispositivos.
- Muy diferente información (presión, temperatura, ph, peso). Los dispositivos pueden cambiar en el tiempo.
- De cada sensor hay información adicional permanente (poco volátil).

Objetivo:
- Analizar la información con la intención de: predecir probabilidad de fallo, y comportamiento operacional (funcionamiento) en casi tiempo real.
- Almacenar la información para a medida que mejoremos en nuestras capacidades analíticas podamos aplicarlas a datos históricos.

Solución propuesta:
Como solución se ha pensado en realizar la ingesta a través de **Kafka** para volcar los datos en una BBDD NoSQL como seria **MongoDB** en tiempo real. Se utilizaría **Spark** tanto para los análisis y modelado para ser capaces de predecir la probabilidad de fallo, como para su aplicación en tiempo real. Para ello utilizamos **Spark Streaming** y **Spark MLib**.
Finalmente se podría utilizar una herramienta de visualización como **Watson Analytics**, para poder analizar los datos resultantes y tomar conclusiones además de mostrarlo de forma clara.

El uso de MongoDB sería para almacenamiento, y para realizar comparaciones como uso de histórico. Con el uso de Kafka que puede tener varias fuentes y destinos, combinado con el sharding de MongoDB, podría escribirse en paralelo y ganar capacidad y eficiencia.

Con Spark ML conectando a MongoDB podrían construirse los modelos y realizar los análisis necesarios, en busca de conseguir predecir la probabilidad de fallo.
Después, utilizando Spark Streaming con los modelos construidos y entrenados, y en tiempo real, ser capaces de conseguir analizar los datos live y predecir la probabilidad de fallo prácticamente en tiempo real.

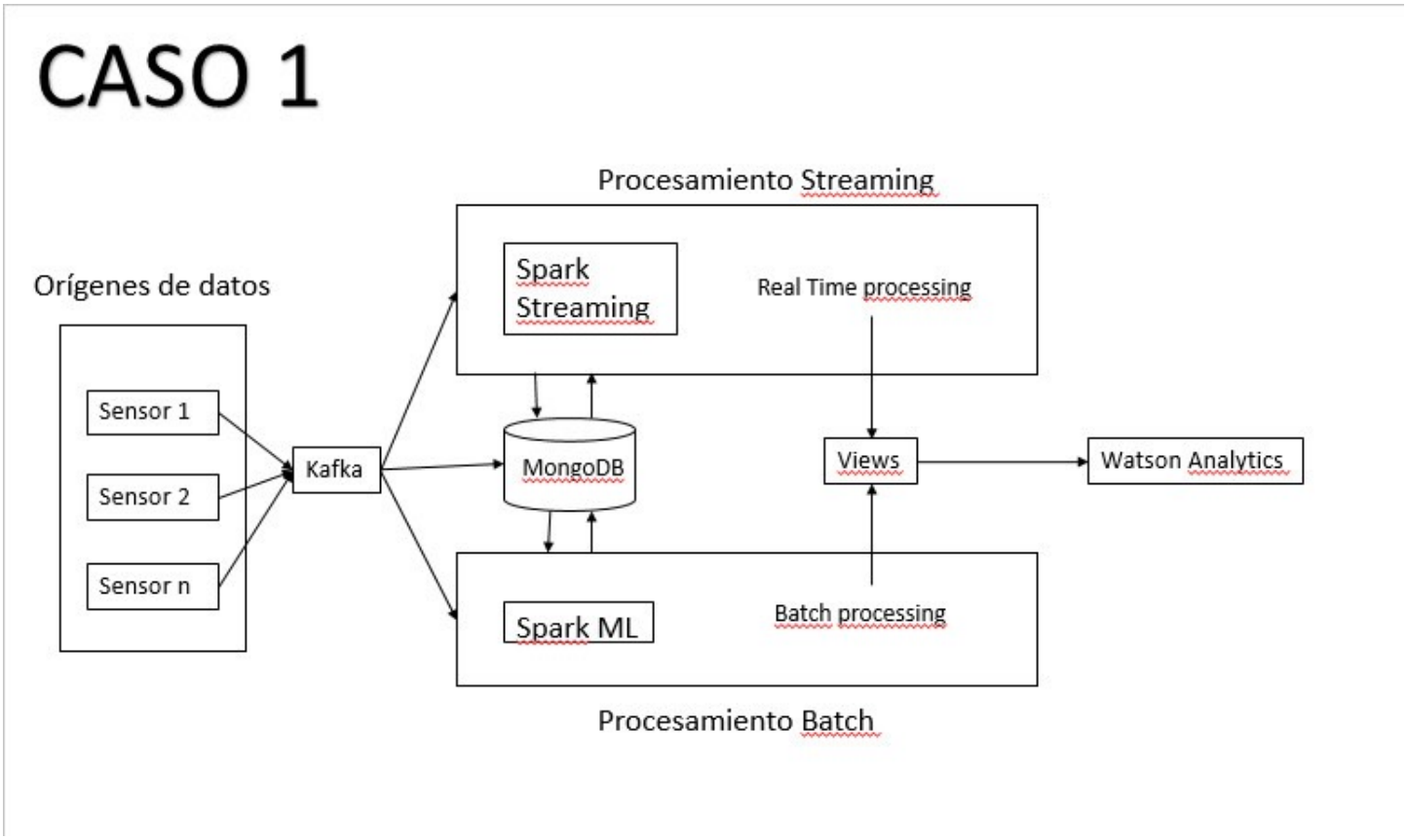
Podría acabar generándose una tabla final ya con los indicadores pertinentes tras aplicar los modelos. A esta tabla podría conectarse por ejemplo con la herramienta de Watson Analytics, que es una herramienta que otorga gran capacidad para hacer análisis visuales y crear historias a partir de los datos que nos ayuden a comprenderlos y sacar conclusiones. Además Watson también permite realizar analítica, y en torno a una variable a predecir se pueden sacar las variables que más influyen en ella y facilitar la toma de decisiones o conclusiones.

Seria una arquitectura de tipo Lambda que se caracteriza por el balanceo de la carga de procesado en dos ramas independientes, una destinada al procesado en Batch, y la otra al Streaming.

Razones por las que se ha escogido Spark:
- Usabilidad: puede ser importante para el desarrollo contar con todas las APIs que trae Spark que pueden ayudar a facilitar el desarrollo.
- Rendimiento: pensamos que la ganancia de rendimiento de Spark al trabajar en memoria puede ser importante para realizar las operaciones con los datos en tiempo real y los análisis.
- Spark Streaming y Spark ML: por el trabajo que hay que realizar en tiempo real, con varias fuentes y datos cambiantes, análisis predictivo sobre los datos, y las posibilidades de conexión de Spark con Kafka o MongoDB en este caso, pensamos que es la solución adecuada. Y tanto Spark Streaming como Spark ML cumplen perfectamente con lo que debemos hacer.

Razones para usar NoSQL:

- Se ha pensado que el uso de NoSQL como podría ser MongoDB sería una buena opción para recibir y almacenar los datos, también teniendo en cuenta que los datos vienen de distintos dispositivos y con cierta variación en el tiempo.
- La flexibilidad que otorgan las BBDD NoSQL respecto a las tradicionales SQL y la facilidad para escalar horizontalmente, y además usar sharding para ganar mayor eficiencia en paralelo, son otros de los motivos para seleccionar esta BBDD.



3. Caso de uso 2

Entorno:
- Negocio basado en unas oficinas centrales y unas sucursales repartidas a nivel nacional.
- Se recibe la información en bloque con frecuencia horaria en formato XML.
- La información corresponde a aplicaciones comunes a la mayor parte de las sucursales.

Objetivo:
- Consolidar la información a nivel central.
- Analizar la información con la intención de: mostrar cuadros de mando con la evolución del negocio por sucursal, región y a nivel corporativo (no nos han dicho cuando hay que tenerlos preparados); ser capaz de realizar predicciones a medio y largo plazo.

Solución propuesta:
Para este caso se ha pensado utilizar el ecosistema **Hadoop**. Al no requerir tiempo real, ni gran velocidad de procesamiento, pensamos que Hadoop y su ecosistema puede ser una buena opción para consolidar la información a nivel global.

Para la recolección de datos podrían usarse tanto **Flume** como **Sqoop**. Flume se usaría para recoger los datos que se hayan dejado en un servidor SFTP, y se configura Flume para ir cogiéndolos de esa dirección y en un tiempo determinado si es necesario. Y con Sqoop también se configuraría la ingesta de los datos, en este caso desde una BBDD relacional para llevarlos a HDFS y Hadoop. Si los datos de las distintas oficinas se centralizaran en una BBDD Oracle por ejemplo, se llevarían mediante Sqoop al HDFS.

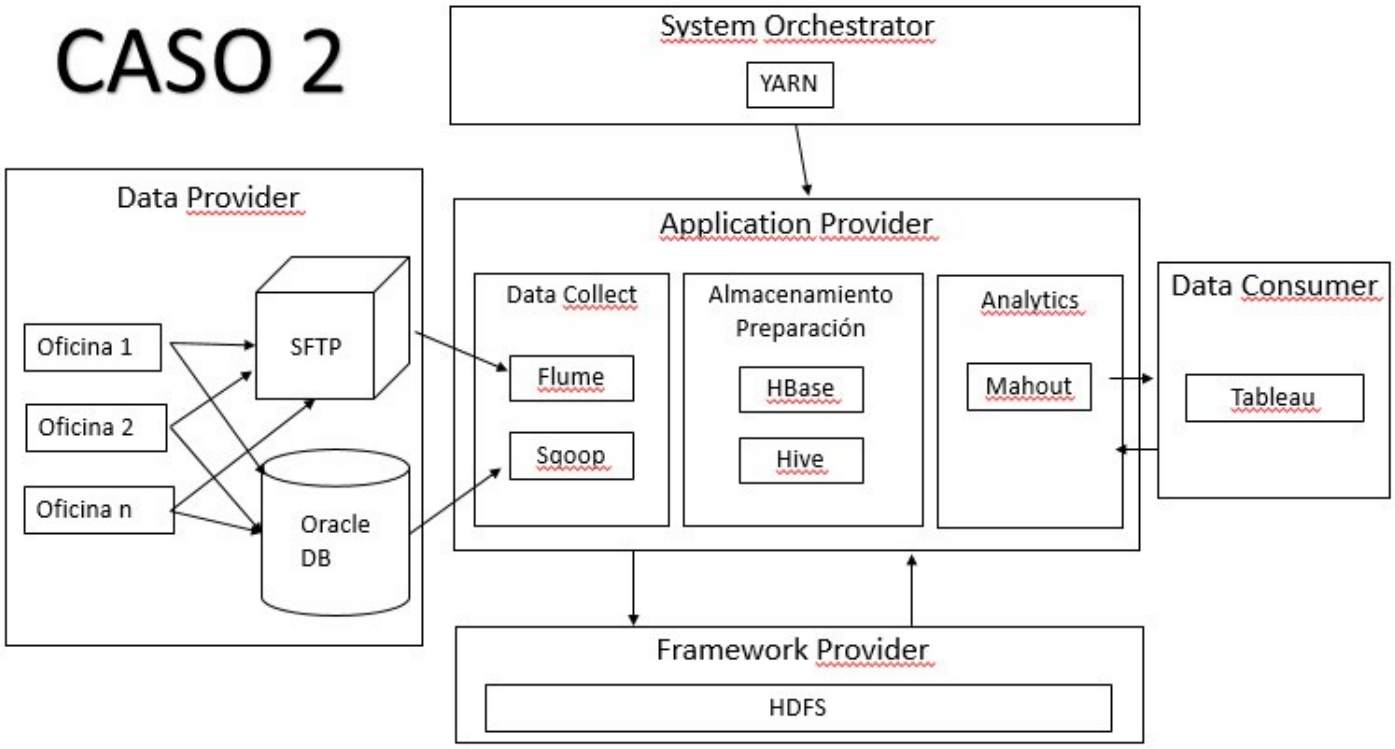
El almacenamiento se haría en **HBase** para consolidar toda la información de la empresa, y en **Hive** se crearían tablas específicas para realizar los procesados y primeros análisis que se requieran hacer a los datos recibidos y almacenados, con la intención final de generar los cuadros de mandos que la parte de negocio de la empresa requiera.
En HBase se activarían los snapshots para ir generando backups de los datos y cambios realizados.

Como se requiere también un análisis predictivo, para ser capaces de en base a los datos que se tienen almacenados predecir un comportamiento futuro, se utilizaría el proyecto **Mahout** del ecosistema Hadoop para tal fin. Con Mahout se pueden realizar algoritmos de machine Learning sobre Hadoop como podrían ser de clasificación o regresión, para una vez construidos y entrenados aplicarlos sobre los datos y poder predecir comportamientos futuros.

Finalmente para visualizar los datos procesados y analizados en Hive se usaría **Tableau**, que es una herramienta de visualización muy potente para generar Dashboards, y que tiene conexión con Hive para poder acceder a sus datos.
Tableau trabaja muy bien además con datos geolocalizados, lo cual podría venir muy bien para este desarrollo ya que si se tuvieran datos donde estuviera la geolocalización se podrían visualizar y analizar los datos sobre un mapa, pintando para cada oficina sus valores y podría ser clave para la gente de negocio para llegar a tomar decisiones.

Razones por las que se ha escogido Hadoop:
- Hadoop está diseñado para procesar grandes volúmenes de datos de manera distribuida, lo cual puede adaptarse a la definición de este problema.
- Está diseñado para ejecutarse en equipos de bajo coste, lo cual en el tema económico puede ser una gran noticia.
- Tiene facilidad para escalar al poder añadir nodos al cluster de forma rápida y sencilla.
- Por ser muy confiable. Da seguridad en la consolidación de la información.
- Hadoop se puede utilizar en teoría para casi cualquier tipo de trabajo batch, mejor que para trabajos en tiempo real, ya que son más fáciles de dividir y ejecutar en paralelo. Además una de las tareas habituales para las que funciona bien es para procesamiento de información en XML como es este caso.

CASO 2


System Orchestrator
YARN
Data Provider
Oficina 1
Oficina 2
Oficina n
SFTP
Oracle DB
Application Provider
Data Collect
Flume
Sqoop
Almacenamiento Preparación
HBase
Hive
Analytics
Mahout
Data Consumer
Tableau
Framework Provider
HDFS