Trabajo de CBD
Análisis de Datos con MapReduce

Manuel José Martínez Baños

May 25, 2021

Contents

20 Mayo 2021

Abstract

En en siguiente documento, se expone el trabajo final realizado para la asignatura CBD sobre la utilización del modelo de programación MapReduce sobre Apache Hadoop.

1 Introducción

El paradigma **MapReduce** es una forma completamente diferente de resolver un cierto subconjunto de problemas paralelizables que evita el cuello de botella de ingerir datos de entrada desde el disco. Mientras que el paralelismo tradicional lleva los datos al cálculo, MapReduce hace lo contrario: lleva el cálculo a los datos.

Apache Hadoop es una implementación real de este paradigma MapReduce, almacenando datos de forma replicada y distribuida en **HDFS**, logra el objetivo de MapReduce: llevar el cálculo a los datos. Siendo MapReduce ideal para operar en conjuntos de datos planos (no estructurados) muy grandes y realizar operaciones trivialmente paralelas en ellos.

Los trabajos realizados en Hadoop deben pasar por una etapa de mapeo y otra de reducción, en la que el mapeador (*mapper*) transforma los datos de entrada sin procesar en pares clave-valor donde pueden ocurrir múltiples valores para la misma clave y el reductor (*reducer*) transforma todos los pares clave-valor que comparten una clave común en una clave única con un valor único. Este diagrama ejemplificaría la implementación de Maprdeuce en Hadoop:

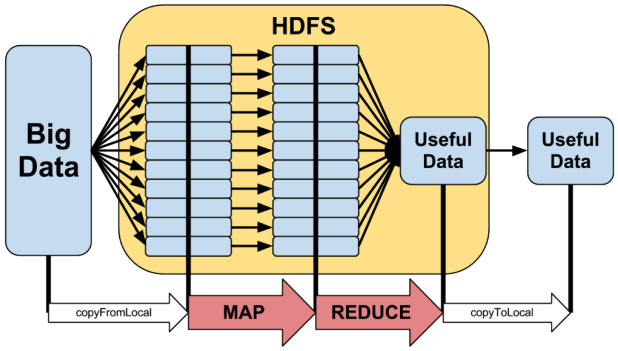


Figure 1: Mapreduce Workflow[1]

2 Objetivos

El objetivo principal de este trabajo es implementar varios programas de análisis de datos usando el modelo de programación MapReduce sobre Apache Hadoop. Los datos que van a ser utilizados para el análisis son un conjunto de reseñas de videojuegos vendidos en Amazon, cuya fuente es la siguiente: `http://snap.stanford.edu/data/`

[1] https://www.glennklockwood.com/data-intensive/hadoop/overview.html

`web-Amazon-links.html`. Por último, como objetivo de mejorar la visualización de los resultados, los datos son procesados para generar una serie de gráficos.

Para llevar acabo los objetivos comentados, han sido utilizadas las siguientes tecnologías:

- Modelo de Programación MapReduce
- Framework Apache Hadoop
- Implementación mediante Hadoop Streaming
- Librería matplotlib.pyplot en Python 3.5

3 Estructura

Este apartado solo tiene la finalidad de mostrar como ha sido estructurado todo el código realizado, cada una de las tareas que han sido realizadas han sido enumeradas, siendo la carpeta llamada **1-task** la primera tarea(*Número de Reviews por juego*) que se detallará más adelante. Seguidamente, se muestra de forma más visual dicha estructura:

```
$ tree final-task/
final-task/
|----1-task/
    |----mapper.py
    |----part-00000
    +----reducer.py
|----2-task/
    |----mapper.py
    |----part-00000
    +----reducer.py
|----3-task/
    |----mapper.py
    |----part-00000
    |----part-00000-2
    |----reducer.py
    +----top-reducer.py
|----4-task/
    |----mapper.py
    |----part-00000
    |----part-00000-2
    |----reducer.py
    +----top-reducer.py
|----5-task/
    |----mapper.py
    |----part-00000
    +----reducer.py
|----converter.py
|----matplot/
    |----matplot-best-rated.py
    |----matplot-bestrated-ranking.py
    |----matplot-history-game.py
    |----matplot-top-revieweds.py
    |----matplot-top-reviewers.py
    |----matplot-top-reviewers-ranking.py
    +----matplot-total-rewies.py
|----Video_Games.json
+----Video_Games.txt.gz
```

4 Trabajos con Hadoop Streaming

Hadoop Streaming es una utilidad que viene con la distribución de Hadoop. La utilidad permite crear y ejecutar trabajos MapReduce con cualquier ejecutable o script como mapeador(*mapper*) y reductor(*reducer*). Con Hadoop Streaming, es necesario escribir un programa que actúe como "**mapper**" y un programa que actúe como "**reducer**". Estas aplicaciones deben interactuar con los flujos de entrada/salida de una manera equivalente a la siguiente:

```
$ cat data.txt | ./mapper.py | order | ./reducer.py> salida.txt
```

Por tanto, Hadoop Streaming envía datos sin procesar a través de stdin al "*mapper*", luego envía los pares clave/valor asignados al "*reducer*" a través de stdin. En medio de este proceso, los datos se agregan por claves durante la etapa intermedia llamada "*shuffle & sort*", como se muestra en la siguiente imagen:

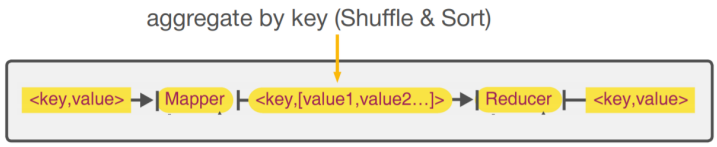


Figure 2: shuffle & sort[2]

4.1 Preparación

Para poder empezar a realizar programas para Hadoop Streaming, estos serían los pasos a seguir antes de poder empezar a trabajar:

1. Crear nuestra carpeta de trabajo en HDFS.
2. Pre-procesar el Dataset para una mejor comodidad y versatilidad de los datos.
3. Definir el script de ejecución para utilizar Hadoop Sreming en el cluster Hadoop con mayor comodidad.

En los siguientes apartados, son detallados brevemente como han sido realizados cada uno de los puntos anteriores, logrando de ese modo, un mayor flujo de trabajo.

4.1.1 HDFS

Para poder tener los datos en un entorno accesible para Hadoop, se ha creado una carpeta en el sistema de fichero HDFS, esta carpeta contendrá el Dataset de reseñas de videojuego. Ha sido creada la carpeta videogames-alucloud31 de la siguiente manera:

```
$ hadoop fs -mkdir folder1
```

[2]https://nancyyanyu.github.io/posts/f53c188b/

4.1.2 Dataset Reseña de Videojuegos

Una vez ha sido descargado el conjunto de datos sobre reseñas de videojuego en Amazon, se visualiza parte del contenido para conocer como esta estructurado:

```
$ cat Video_Games.txt | head -n 20
product/productId: B000068VBQ
product/title: Fisher-Price Rescue Heroes: Lava Landslide
product/price: 8.88
review/userId: unknown
review/profileName: unknown
review/helpfulness: 11/11
review/score: 2.0
review/time: 1042070400
review/summary: Requires too much coordination
review/text: I bought this software for my 5 year .....

product/productId: B000068VBQ
product/title: Fisher-Price Rescue Heroes: Lava Landslide
product/price: 8.88
review/userId: A10P44U29RNOT6
review/profileName: D. Jones
review/helpfulness: 6/6
.....
```

Como podemos observar, la separación entre productos no es por líneas, sino que cada reseña de producto es separado por una línea en blanco. Esta estructura no es óptima ni cómoda a la hora de trabajar con Hadoop Streaming, por tanto para el trabajo, se ha pre-procesado el Dataset para lograr que cada línea sea una reseña de un producto en formato JSON. Para lograr dicho pre-procesado, se ha utilizado el siguiente script en python2:

Listing 1: converter.py

```
#!/usr/bin/python2
import gzip
import json

def parse(filename):
  f = gzip.open(filename, 'r')
  entry = {}
  for l in f:
    l = l.strip()
    colonPos = l.find(':')
    if colonPos == -1:
      yield entry
      entry = {}
      continue
    eName = l[:colonPos]
    rest = l[colonPos+2:]
    entry[eName] = rest
  yield entry

target = open("Video_Games.json", 'w+')
for e in parse("Video_Games.txt.gz"):
  target.writelines(json.dumps(e)+'\n')
target.close()
```

Para la ejecución de **converter.py**, previamente debe descargarse en la misma ubicación el fichero Video_Games.txt.gz[3]. Este script genera un fichero **Video_Games.json**, donde cada línea es una reseña de un producto(videojuego), como es mostrado a continuación:

```
$ ./converter.py

$ cat Video_Games.json | head -n 2
{"review/profileName": "unknown", "product/price": "8.88", "review/time": "1042070400",
      "product/productId": "B000068VBQ", "review/helpfulness": "11/11", "review/summary
     ": "Requires too much coordination", "review/userId": "unknown", "product/title":
     "Fisher-Price Rescue Heroes: Lava Landslide", "review/score": "2.0", "review/text"
     : "I bought this software for my 5 year old. ..."}
{"review/profileName": "unknown", "product/price": "8.88", "review/time": "1041552000",
      "product/productId": "B000068VBQ", "review/helpfulness": "9/10", "review/summary"
     : "You can't pick which parts you want to play!", "review/userId": "unknown", "
     product/title": "Fisher-Price Rescue Heroes: Lava Landslide", "review/score": "2.0
     ", "review/text": "I got this f......."}
```

Una vez se ha realizado el pre-procesado, es turno de subir el Dataset **Video_Games.json** a la carpeta HDFS creada:

```
$ hadoop fs -put Video_Games.json videogames-alucloud31

$ hadoop fs -text videogames-alucloud31/Video_Games.json
```

4.1.3 Script de Lanzamiento

Una vez ha sido copiado el Dataset en el sistema de archivos HDFS del cluster Hadood, los trabajos de Python MapReduce en el clúster de Hadoop pueden ser lanzados. Para ello, se utiliza la API de transmisión de Hadoop para ayudarnos a pasar datos entre nuestro "*mapper*" y el código "*reducer*" a través de stdin y stdout.

Para este trabaja se ha utilizado **hs**, un breve script para ayudar a la invocación de los trabajos en en Hadoop Streaming. Este script contiene el siguiente código:

```
$ more `which hs`
/opt/hadoop/bin/hadoop jar /opt/hadoop/share/hadoop/tools/lib/hadoop-streaming-*.jar -
     mapper $1 -reducer $2 -file $1 -file $2 -input $3 -output $4
```

Por tanto, todas las implementaciones relizadas han sido son lanzadas siguiendo el siguiente patrón:

```
#Limpiar la carpeta de salida
$ hadoop fs -rm -r videogames-alucloud31/output

#Lanzar trabajo
1-tarea$ hs mapper.py reducer.py videogames-alucloud31/Video_Games.json
     videogames-alucloud31/output

#Copiar resultado a una carpeta local determinada
hadoop fs -text videogames-alucloud31/output/part-00000 > part-00000
```

Ahora que el entorno de trabajo esta preparado, en los siguientes apartados se van a presentar las tareas realizadas sobre el Dataset sobre **reseñas de videojuegos de Amazon**.

[3]http://snap.stanford.edu/data/amazon/Video_Games.txt.gz

4.2 Número de Reseñas por juego

En esta primera implantación, se ha realizado dos scripts mapper.py y reducer.py para obtener un listado de todos los juegos con el total de reseñas(*reviews*) que han sido realizados. Además han sido realizados dos scripts, para la visualización de los datos de dos maneras distintas.

4.2.1 Solución

En este aparatado, se muestra el código implementado para la resolución, en concreto, dos scripts mapper.py y reducer.py.

Código 2: mapper.py

```
#!/usr/bin/python2
import sys
import json
entry = {}
for line in sys.stdin:
    line = line.strip()
    # parse the incoming json
    j = json.loads(line)
    if j != None:
        try:
                print "{0}\t{1}".format(j['product/title'], j['review/
                    score'])
        except Exception as e:
                continue
```

Código 3: reducer.py

```
#!/usr/bin/python
import sys

TotalScore = 0
oldKey = None

for line in sys.stdin:
    data_mapped = line.strip().split("\t")
    if len(data_mapped) != 2:
        continue

    thisKey, thisScore = data_mapped
    if oldKey and oldKey != thisKey:
        print oldKey, "\t", TotalScore
        oldKey = thisKey;
        TotalScore = 0

    oldKey = thisKey
    TotalScore += 1

if oldKey != None:
    print oldKey, "\t", TotalScore
```

4.2.2 Resultados

Primero se comprueban los resultados sobre un fragmento del Dataset, sin hacer uso de Hadoop. Esta primera comprobación se reproducirá en todas las demás tareas, pero ha sido omitido para agilizar el documento:

```
$ cat Video_Games.json | head -n 300 |  ./1-task/mapper.py | sort -k 1,1
     |  ./1-task/reducer.py | head -n 5
Barbie Sparkling Ice Show       24
Barbie as Rapunzel: A Creative Adventure        55
Fisher-Price Rescue Heroes: Lava Landslide      7
FlatOut         12
Kelly Club Pet Parade   15
```

Ahora es mostrado el resultado obtenido sobre el Dataset **Video_Games.json** completo:

```
cat 1-task/part-00000 | head -n 5
"PSP, 3 PACK SCREEN PROTECTOR" 2
$100,000 Pyramid     22
'N Sync Hotline Fantasy Phone and CD-ROM Game 15
'Nam (Jewel Case)    6
(2 Pack)GameShark 2.4 Ghz Wireless MicroCON- PS2 and PS2 (slim)  2
```

Los resultados obtenidos son un fragmento pequeño del total, si se desea enriquecer de mayor utilidad los datos obtenidos, se puede utilizar algún tipo de visualizador de datos. Por ello, en este trabajo se ha manejado la librería **matplot** en Python3.5 para generar una serie de gráficos. Para este apartado se han generado dos gráficos.

El primer gráfico muestra los juegos con un número de reseñas considerables (550 reseñas o más) y el segundo gráfico muestra los 15 juegos más comentados. A continuación son expuestos los dos gráficos comentados, siendo generados con los scripts **matplot-total-rewies** y **matplot-top-revieweds.py**:

Figure 3: Juegos con más de 550 reseñas

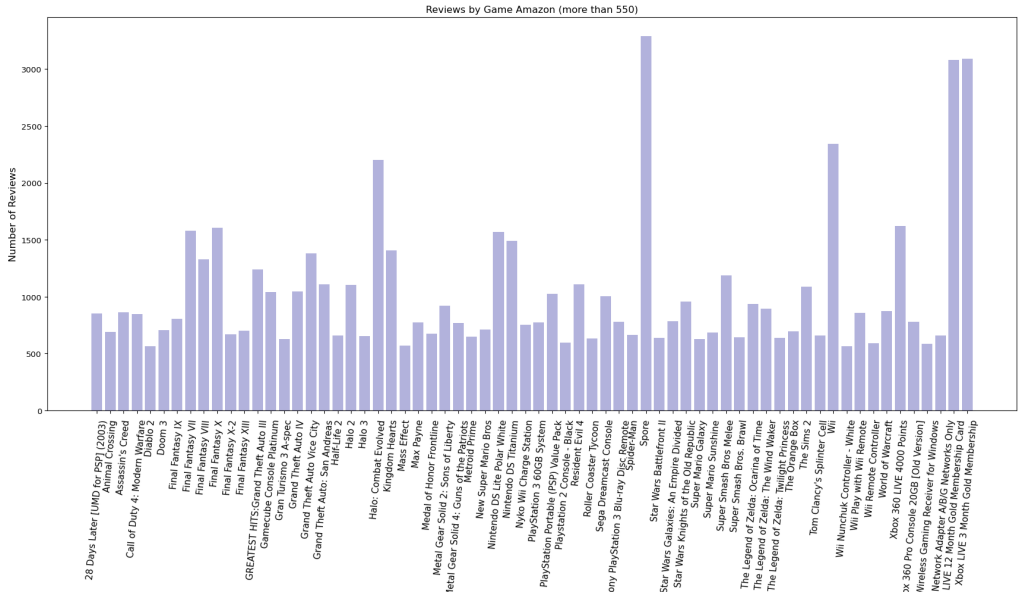
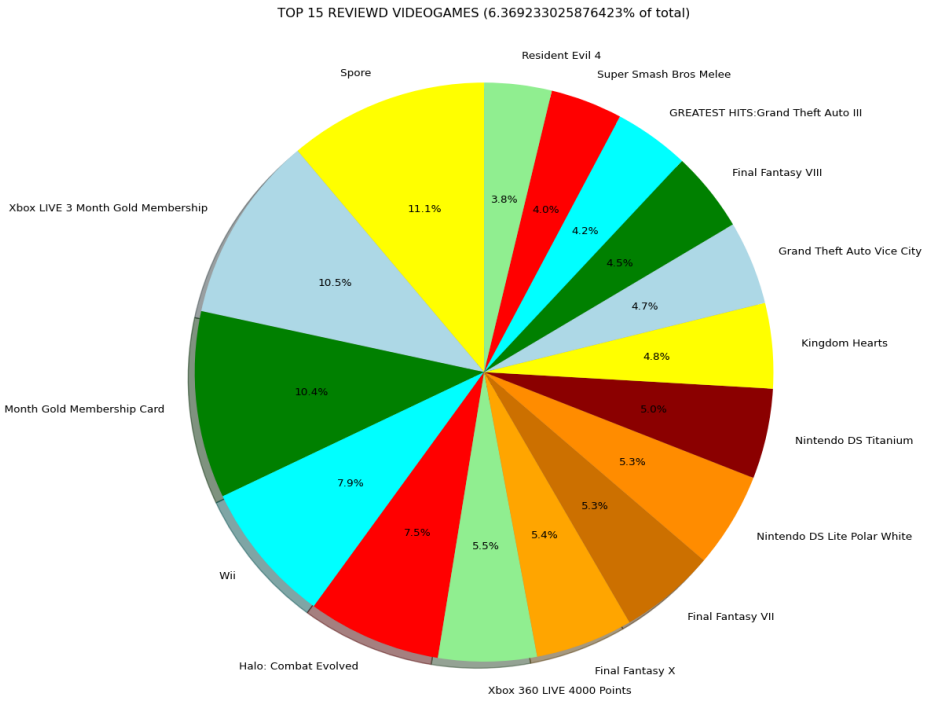


Figure 4: Top 15 juegos reseñados



Como se observa en este último gráfico, los 15 juegos más comentados solo representan un 6.36% del total de juegos, además muestra como el Spore es el juego más comentado o reseñado con un 11.1% de las reseñas del top 15.

4.3 El juego con más reseñas

Como adición a la tarea anterior, se ha creado un script reduce.py y mapper.py para obtener el juego con más reseñas. En este caso la key de ordenación es por ID del producto.

4.3.1 Solución

En este aparatado se muestra el código implementado para la resolución, en concreto, dos scripts mapper.py y reducer.py.

Código 4: mapper.py

```
#!/usr/bin/python2
import sys
import json
entry = {}
for line in sys.stdin:
    line = line.strip()
    j = json.loads(line)
    if j != None:
        try:
                print "{0}\t{1}".format(j['product/productId'], j['
                    product/title'])
        except Exception as e:
                continue
```

Código 5: reducer.py

```
#!/usr/bin/python
import sys

totalReview = 0
maxtotalReview = 0
topGameReview= None
topIdProduct= None
oldKey = None
oldGame = None

for line in sys.stdin:
    data_mapped = line.strip().split("\t")
    if len(data_mapped) != 2:
        continue

    thisKey, thisGame = data_mapped
    if oldKey and oldKey != thisKey:
        if totalReview > maxtotalReview:
            maxtotalReview=totalReview
            topIdProduct=oldKey
            topGameReview=oldGame
        oldKey = thisKey;
        totalReview = 0

    oldGame =  thisGame
    oldKey = thisKey
    totalReview += 1

if oldKey != None and totalReview > maxtotalReview:
        maxtotalReview=totalReview
        topGameReview=oldGame
        topIdProduct=oldKey

if topGameReview != None:
    print topIdProduct, "\t",topGameReview, "\t",maxtotalReview
```

4.3.2 Resultados

El resultado del lanzamiento de la tarea en Hadoop es la siguiente:

```
$ hadoop fs -text videogames-alucloud31/output/part-00000
B000FKBCX4    Spore  3292
```

El resultado obtenido es el correcto, debido a que, si es contrastado con los gráficos anteriores, se puede observar que Spore es el juego con más reseñas.

4.4 Lista de juegos y sus estadísticas

Esta tarea tiene como objetivo perfeccionar la obtención de los resultados de la primera tarea. La valoración final de un producto en la tienda de Amazon, es el resultado de la media de todas las reseñas, siendo la peor reseña una estrella y la mejor 5 estrellas, como se puede observar en la siguiente imagen:

Figure 5: Valoración media en la tienda de Amazon

Por tanto, se desea conseguir por cada juego su puntuación media, así como el recuento de puntuaciones de cada tipo, dado que la puntación solo puede ser de 1,2,3,4 o 5.

4.4.1 Solución

Para realizar la media, se han sumado todos los valores de la variable "*review/score*" y dividido por el total reseñas obtenidas. Por último, para el recuento de puntuaciones, solo ha sido necesario crear un array donde cada una de sus posiciones representa cada tipo de puntación(1,2,3,4 y 5), siendo la posición 1 del array el contador de la puntuación de valor "1" y así sucesivamente. A continuación, los scripts implementados:

Código 6: mapper.py

```
#!/usr/bin/python2
import sys
import json

entry = {}
for line in sys.stdin:
    line = line.strip()
    j = json.loads(line)
    if j != None:
        try:
                print "{0}\t{1}".format(j['product/title'], j['review/
                    score'])
        except Exception as e:
                continue
```

Código 7: reducer.py

```
#!/usr/bin/python

import sys
totalScore = 0
totalReview=0
oldKey = None
#Score has only 5 values (1.0,2.0,3.0,4.0,5.0)
sc = [0 for i in range(6)]
for line in sys.stdin:
    data_mapped = line.strip().split("\t")
    if len(data_mapped) != 2:
        continue

    thisKey, thisScore = data_mapped
    if oldKey and oldKey != thisKey:
        AVGScore=totalScore/totalReview
        print oldKey,"\t",AVGScore,"\t",totalReview,"\t",sc[1],":",sc
            [2],":",sc[3],":",sc[4],":",sc[5]
        oldKey = thisKey;
        sc = [0 for i in range(6)]
        totalScore = 0
        totalReview = 0

    oldKey = thisKey
    totalReview += 1
    sc[int(float(thisScore))] +=1
    totalScore += float(thisScore)

if oldKey != None:
        AVGScore=totalScore/totalReview
        sc[int(float(thisScore))] +=1
        print oldKey,"\t",AVGScore,"\t",totalReview,"\t",sc[1],":",sc
            [2],":",sc[3],":",sc[4],":",sc[5]
```

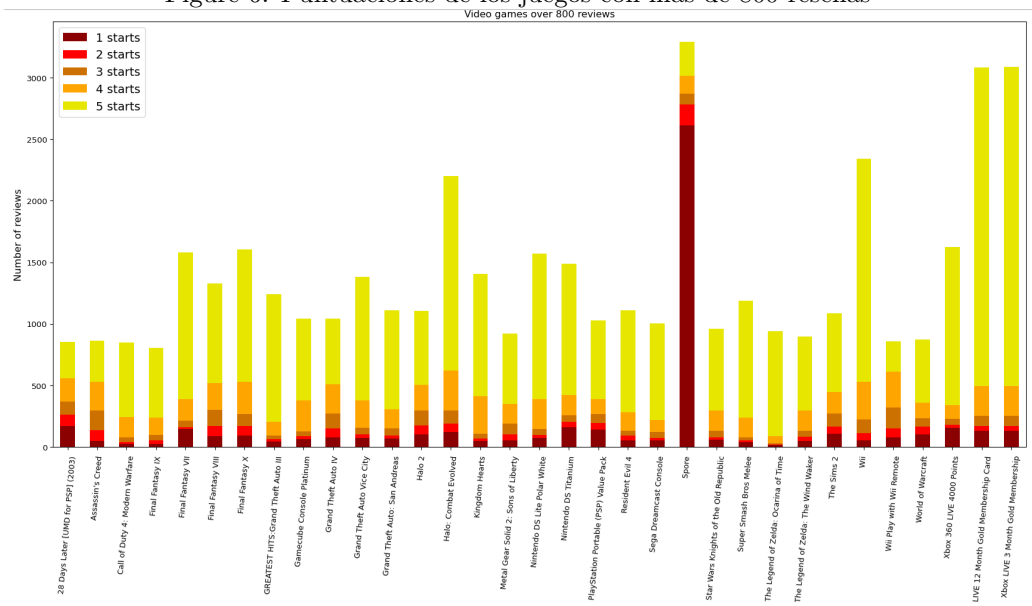
4.4.2 Resultados

Los resultado obtenidos en el lanzamiento tiene el siguiente aspecto:

```
final-task cat 3-task/part-00000 | head -n 5
"PSP, 3 PACK SCREEN PROTECTOR" 3.0  2      1 : 0 : 0 : 0 : 1
$100,000 Pyramid      4.18181818182 22     0 : 1 : 4 : 7 : 10
'N Sync Hotline Fantasy Phone and CD-ROM Game 3.8 15     3 : 0 : 1 : 4 : 7
'Nam (Jewel Case)     3.0    6      2 : 1 : 0 : 1 : 2
(2 Pack)GameShark 2.4 Ghz Wireless MicroCON- PS2 and PS2 (slim)  2.5    2      1 : 0 :
    0 : 1 : 0
```

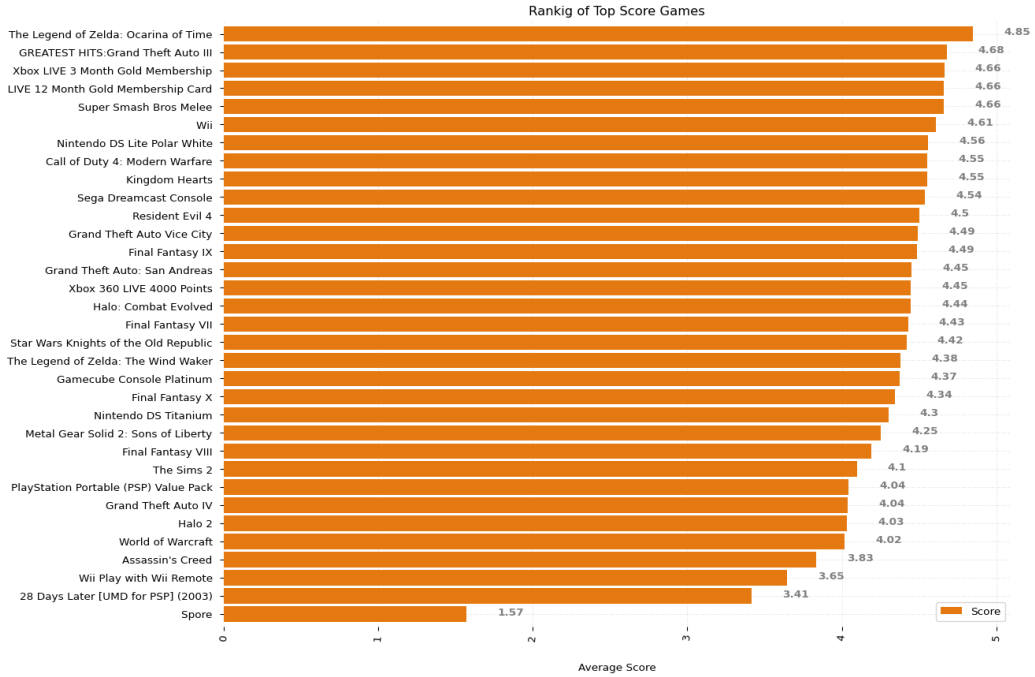
Como se puede observar, se obtiene por cada juego el título, la puntación media, el total de reseñas y el recuento de cada puntuación obtenida. Para mejorar la visualización, al igual que en la primera tarea, se han utilizado una serie de gráficos. Seguidamente, se muestra las estadísticas de los juegos con más de 800 reseñas y un ranking, permitiendo conocer el juego con ratio de puntación más alto. Se ha limitado a partir de 800 para disminuir el tamaño del gráfico resultante y facilitar su entendimiento. Este gráfico ha sido realizado con el script **matplot-best-rated.py**.

Figure 6: Puntuaciones de los juegos con más de 800 reseñas



En este gráfico es similar al primer gráfico (Figura 1), pero ahora el total de reseñas de un juego se compone de distintos colores representado las distintas notas(estrellas) obtenidas. Para finalizar este apartado se muestra un ranking, siendo **The Legend of Zelda: Ocarina of Time** con una puntación media de **4.84648187633** de un total de 938 reseñas. Este ranking ha sido generado con el script **matplot-bestrated-ranking.py**.

Figure 7: Puntuaciones de los juegos con más de 800 reseñas



4.5 Lista de usuarios y su número total reseñas

Esta tarea, tiene como objetivo obtener una la lista de usuarios con el número de reseñas que han realizado, además también se incluye la puntación más alta dada y la más baja. Con los datos obtenidos, se puede determinar los usuarios más dispuestos a realizar reseñas o si han realizado al menos una valoración muy positiva.

4.5.1 Solución

Para poder lograr el objetivo correctamente, se han excluido todas aquellas reseñas con un nombre de usuario(*review/profileName*) con el valor "unknown" o una longitud de nombre menor a 2. Además la puntación más alta y más baja están acotadas entre los valores del 1 al 5, dando un valor 0 a la puntación más baja en caso de empate o solo existir una sola reseña. Las puntuaciones que se asignan en las reseñas de Amazon solo tienen este conjunto de valores: 1.0, 2.0, 3.0, 4.0, 5.0.

A continuación, se expone el código desarrollado como en las tareas anteriores, han sido creados dos scripts mapper.py y reducer.py.

Código 8: mapper.py

```
#!/usr/bin/python2

import sys
import json

entry = {}

for line in sys.stdin:
    line = line.strip()
    j = json.loads(line)

    if j != None:
        try:
                if  len(str(j['review/profileName']))>1 and j['review/
                    profileName'] != 'unknown':
                        print "{0}\t{1}".format(j['review/profileName'],
                            j['review/score'])
        except Exception as e:
                continue
```

Código 9: reducer.py

```
#!/usr/bin/python
import sys

totalScore = 0
highestScore= 0
lowestScore=0
oldKey = None

for line in sys.stdin:
    data_mapped = line.strip().split("\t")
    if len(data_mapped) != 2:
        continue

    thisUser, thisScore = data_mapped
    if oldKey and oldKey != thisUser:
        print oldKey, "\t",totalScore, "\t",lowestScore, "\t",
            highestScore
        oldKey = thisUser;
        totalScore = 0
        lowestScore = 0
        highestScore = 0

    oldKey = thisUser
    totalScore += 1
    if float(lowestScore) > float(thisScore):
        lowestScore = thisScore
    elif float(highestScore) < float(thisScore):
        highestScore = thisScore

if oldKey != None:
    print oldKey, "\t",totalScore, "\t",lowestScore, "\t",highestScore
```

4.5.2 Resultados

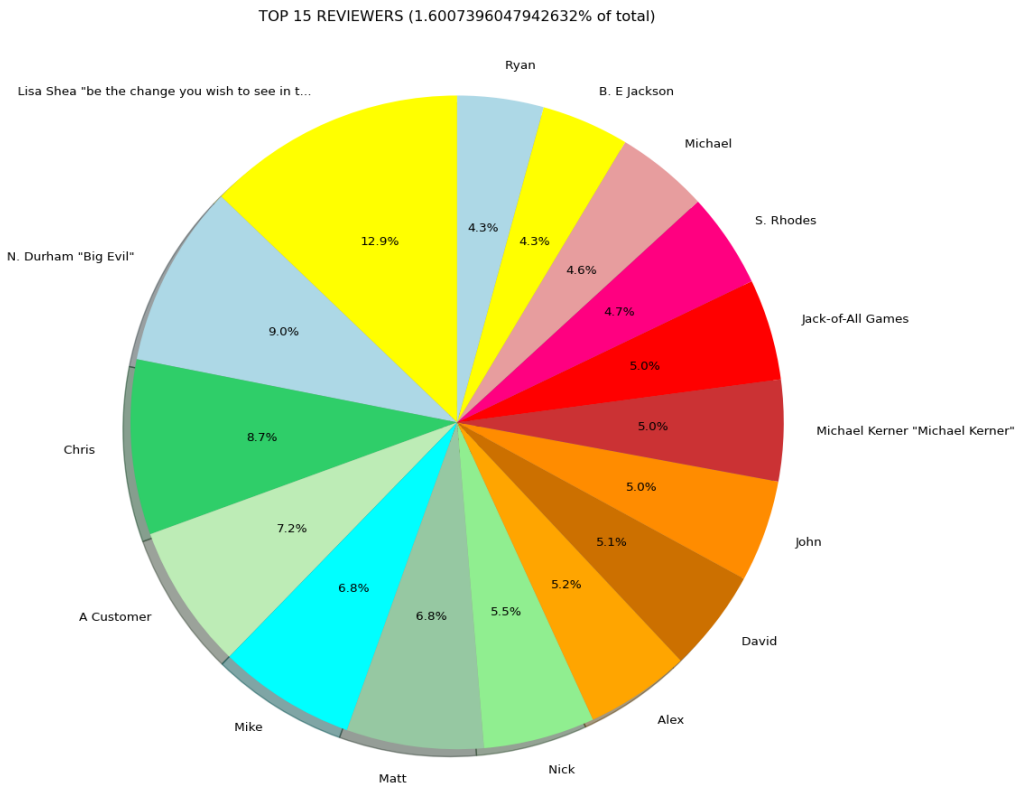
Seguidamente se muestran los resultado obtenidos de lanzar dichos scripts contra el Dataset completo con ayuda del comando **hs**:

```
 hadoop fs -text videogames-alucloud31/output/part-00000 | head -n 4
! MR. KNOW IT ALL ;-b "DR SHOCK"    5      0      5.0
!!:. VaNesSa .:!!     1      0      5.0
!!Davvid!! "Dave"     6      0      5.0
!3 Year old gamer     2      0      5.0
```

En este resultado se muestra la lista de usuarios, el total reseñas y por último, la nota más baja y la nota más alta.

Como en los apartados anteriores, se ha hecho uso de un gráfico para un mejor análisis de los datos y por ejemplo, determinar el usuario con más reseñas. Ahora se muestra el gráfico obtenido de lanzar el script **matplot-top-reviewers.py**:

Figure 8: Los 15 usuarios que han realizado más reseñas



Para facilitar la visualización del resultado, los datos para el gráfico ha sido limitado al "top" 15 usuarios con más reseñas, este top 15 solo representa un 1.6% del total de reseñas. Como se observa en dicho gráfico el usuario "Lisa Shea "be the change you wish to see in t..." es quien ha realizado más reseñas.

Además para verificar el resultado, se ha creado otra versión del script reducer.py llamado **top-reducer.py**, cuyo objetivo es obtener el usuario con mayor número de reseñas. El contenido del script es el siguiente:

Código 10: top-reducer.py

```
#!/usr/bin/python

import sys
TotalScore = 0
maxTotalScore = 0
topUserReviews= None
oldKey = None

for line in sys.stdin:
    data_mapped = line.strip().split("\t")
    if len(data_mapped) != 2:
        continue
    thisKey, thisScore = data_mapped
    if oldKey and oldKey != thisKey:
        if TotalScore > maxTotalScore:
```

```
            maxTotalScore=TotalScore
            topUserReviews=oldKey
        oldKey = thisKey;
        TotalScore = 0

    oldKey = thisKey
    TotalScore += 1


if oldKey != None and TotalScore > maxTotalScore:
        maxTotalScore=TotalScore
        topUserReviews=oldKey

if topUserReviews != None:
    print topUserReviews, "\t",maxTotalScore, "\t"
```

Este script genera el resultado previsto y concuerda con la información aportada por el gráfico anterior:

```
final-task cat 4-task/part-00000-2
Lisa Shea "be the change you wish to see in t...  750
```

4.6 Historial de puntaciones de un juego

En el mundo de los videojuegos, con el paso del tiempo, un videojuego puede verse afectado a cambios drásticos, haciéndolo muy diferente a como era en los primeros meses o primer año de lanzamiento. Esto es debido a que un videojuego no deja de ser un software que se vende en formato físico, y por tanto, este software puede ser actualizado con el paso del tiempo. Existen casos recientes como el del videojuego **No Man's Sky**, donde con el pasar del tiempo, su recepción a mejorado a pasos agigantados (*La redención definitiva de No Man's Sky*[4]).

Por este motivo, es interesante conocer el historial de puntaciones de un determinado videojuego y observar si ha ido cambiando la recepción de la gente con el pasar del tiempo.

4.6.1 Solución

Al igual que las tareas anteriores, han sido creados dos scripts "mapper.py" y "reducer.py" pero con la diferencia de que el script "reducer.py" acepta entrada de parámetros, en caso de no recibir ningún parámetro, se realiza la búsqueda de todas las puntaciones sobre un juego denominado "Age of Empires 2: Age of Kings", por defecto.

Código 11: mapper.py

```
#!/usr/bin/python2
import sys
import json
entry = {}
for line in sys.stdin:
    line = line.strip()
    # parse the incoming json
    j = json.loads(line)

    if j != None:
        try:
```

[4]https://blogs.larioja.com/partida-guardada/2019/08/24/la-redencion-definitiva-de-no-mans-sky/

```
            print "{0}\t{1}\t{2}".format( j['product/title'], j['review/
                score'],j['review/time'])
        except Exception as e:
            continue
```

Código 12: reducer.py

```
#!/usr/bin/python

import sys

def reduce(findItem,separator='\t'):
    oldKey = None
    titleName=findItem
    findItem= findItem.lower()
    for line in sys.stdin:
        data_mapped = line.strip().split(separator)
        if len(data_mapped) != 3:
            continue
        #title, score, unixtime
        thisKey, thisScore, thisTime = data_mapped

        if oldKey and oldKey != thisKey:
            oldKey = thisKey;

        oldKey = thisKey.lower()
        if oldKey == findItem:
            print titleName, "\t", thisScore, "\t", thisTime


if __name__ == "__main__":
    arguments = len(sys.argv) - 1
    if (arguments >= 1):
        reduce(sys.argv[1])
    else:
        reduce("Age of Empires 2: Age of Kings")
```

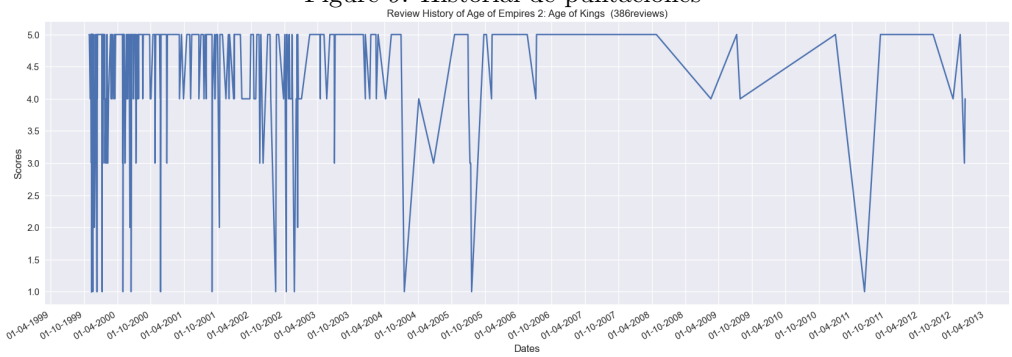
4.6.2 Resultados

Una vez son lanzados los scripts anteriormente expuesto sobre el Dataset completo en el cluster Hadoop, se ha generado un fichero resultante cuyo contenido es el siguiente:

```
$ hadoop fs -text videogames-alucloud31/output/part-00000 | head -n 4
Age of Empires 2: Age of Kings  5.0  1144022400
Age of Empires 2: Age of Kings  4.0  1081296000
Age of Empires 2: Age of Kings  5.0  1037059200
Age of Empires 2: Age of Kings  5.0  1026691200
```

Como se puede apreciar en el resultado, cada línea esta formada por el título del videojuego, la puntación y la fecha en formato unix. Para facilitar la compresión del resultado obtenido, se ha generado el siguiente gráfico con ayuda del script denominado **matplot-history-game.py**.

Figure 9: Historial de puntaciones



El el gráfico anterior se visualiza como a la largo de los años el juego *Age of Empires 2: Age of Kings* se ha mantenido con una recepción muy buena por los usuarios. Este juego es un clásico entre los jugadores de estrategia y actualmente todavía sigue siendo jugado por muchos jugadores.

El código implementado en el script **reduce.py** permite el paso de parámetros, permitiendo cambiar el videojuego que se desea filtrar. Como el pequeño script **hs** no permite suministrar el nombre del script junto a los parámetros, si se desea suministrar el nombre del videojuego, se debe lanzar el trabajo directamente utilizando la librería de hadoop-streaming-*.jar:

```
/opt/hadoop/bin/hadoop jar /opt/hadoop/share/hadoop/tools/lib/hadoop-streaming-*.jar \
-mapper mapper.py \
-reducer 'reducer.py "World of Warcraft" ' \
-file mapper.py -file reducer.py \
-input videogames-alucloud31/Video_Games.json \
-output videogames-alucloud31/output
```

Para concluir, se procede a mostrar dos nuevos gráficos pero con juegos totalmente distintos, los videojuegos **Halo 3** y **World of Warcrast**. Es interesante conocer su historial, puesto que al ser juegos enfocados al multijugador son más propensos a tener más número actualizaciones y de mayor impacto en el juego.

Figure 10: Historial del Videojuego Halo 3

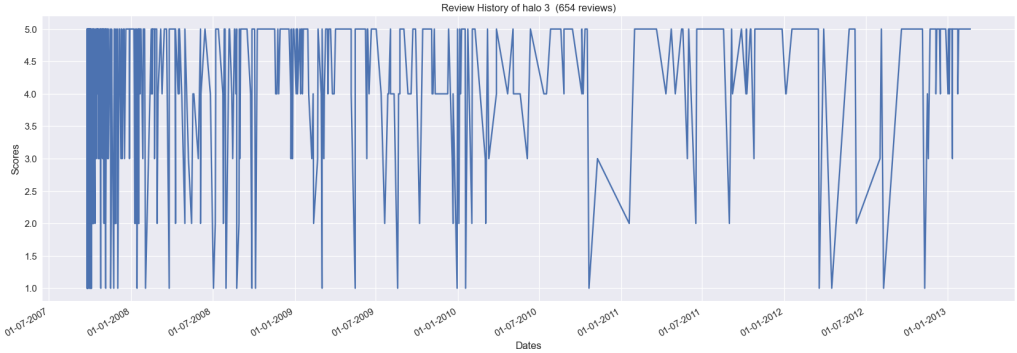
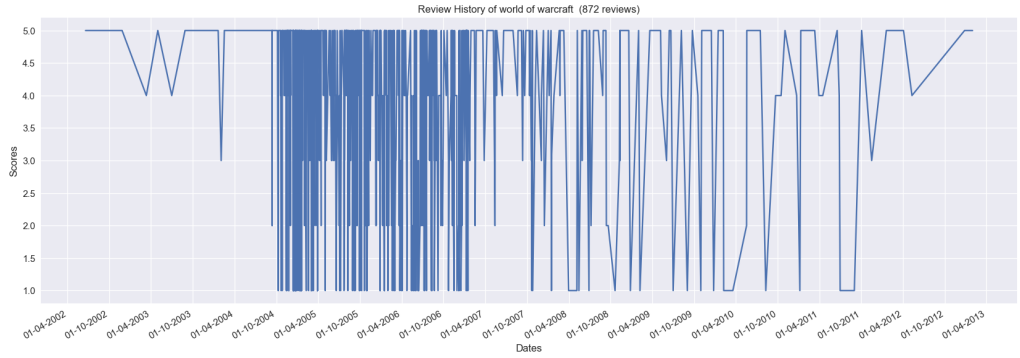


Figure 11: Historial del Videojuego World of Warcraft



Como se observa en los gráficos, el videojuego **Halo3** tiene mayormente puntuaciones elevadas a sus inicios del lanzamiento, estas puntuaciones se van estabilizando positivamente conforme pasa el tiempo. Pero en el caso de **World of Warcraft** se aprecia como pasado un tiempo, existe un aumento notorio en el número de puntuaciones tanto positivas como negativas, esto puede deberse a la salida de una gran actualización como por ejemplo, una nueva expansión.

5 Conclusiones

Durante el desarrollo del trabajo, se ha logrado tratar de manera más profunda con el cluster Hadoop utilizando Hadoop Streaming y de ese modo conocer las ventajas y limitaciones del paradigma de programación MapReduce. Existen otras implementaciones de MapReduce mas óptimas como Apache Spark, pero con Hadoop Streaming ha sido suficiente para lograr obtener información de mayor utilidad a partir de un Dataset de gran tamaño, que a priori es difícil poder sacar alguna información al respecto. Además, esta información obtenida puede ser tratada para visualizarse mediante gráficos, como los que han sido realizados en este trabajo y entender mejor los resultados obtenidos.