DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo práctico 3

Sistemas Operativos

Integrante	LU	Correo electrónico
Fernando Gasperi Jabalera	56/09	fgasperijabalera@gmail.com
Martín Matías Monti	727/10	martinmatiasmonti@gmail.com
Diego Alejandro Amil	68/09	amildie@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Índice

1. Implementación

Contamos con una colección de **MongoDB** llamada *posts*, que contiene documentos con un formato específico, y debemos extraer cierta información de la misma utilizando el modelo de programación *MapReduce*. Cada documento de la colección *posts* se corresponde con un post en una red social llamada *Rededit*.

1.1. Ejercicio 1

En este ejercicio debemos encontrar el subreddit con mayor score promedio. Los documentos de la colección tienen un campo llamado subreddit, el mismo se corresponde con el nombre de un canal dentro de la red social que agrupa mensajes con algún tipo de similitud. Además, cada post tiene asignado un score, un puntaje representado con un entero. El objetivo es para cada subreddit tomar todos sus posts y calcular el score promedio entre todos ellos. Luego obtener el subreddit que tenga el mayor promedio.

map queremos agrupar por subreddit por lo cual hacemos que el subreddit de cada documento sea nuestra key y como value vamos necesitar por un lado el score, ya que lo que finalmente queremos calcular es el promedio de ellos, y por otro el count. El count en este momento parece trivial ya que siempre se le asigna 1, sin embargo, tomará relevancia en el reduce que lo va a preparar para poder tomar el promedio:

```
emit(this.subreddit, {count: 1, score: this.score});
```

reduce el reduce se encarga de contar la cantidad de posts que tiene cada subreddit y calcular la suma total de los puntajes de ellos. Necesitamos esos dos valores para poder calcular el puntaje promedio del subreddit:

```
var reducedVal = {count: 0, score: 0};
for (var i = 0; i < values.length; i++) {
  reducedVal.score += values[i].score;
  reducedVal.count += values[i].count;
}

return reducedVal;
```

finalize finalmente con la función finalize calculamos el puntaje promedio de cada subreddit realizando la división entre el puntaje total y la cantidad de posts.

Luego de ejecutar éste MapReduce sobre *posts* tenemos una nueva colección que tiene por id el nombre del subreddit y como value el puntaje promedio del mismo. Para obtener el que mayor puntaje promedio tiene basta con correr la siguiente consulta:

```
db.subreddit_puntaje_promedio.find().sort({value: -1}).limit(1)
```

con esta última consulta obtenemos todos los documentos de la nueva colección generada, los ordenamos decrecientemente por su value, el puntaje promedio en este caso, e imponemos un límite de 1 a los resultados mostrados, lo cual resulta en que sólo obtengamos el primero:

```
{ "\_id" : "GirlGamers", "value" : 2483 }
```

1.2. Ejercicio 2

En este caso debemos encontrar los doce títulos con mayor score que cuenten con al menos 2000 votos. Para entontrarlos utilizaremos los campos score, total_votes y title. El orden en el que haremos esto será:

1. quedarnos con los posts que tengan más de 2000 votos.
2. los ordenaremos decrecientemente por score.
3. nos quedaremos con los 12 primeros.

Puede ser que los títulos sean un índice único dentro de la colección posts, en ese caso el procedimiento se podría simplificar mucho. Sin embargo, como comprobamos que existen distintos posts con el mismo título decidimos tomar un enfoque obtiene la respuesta correcta en los dos casos, es decir, sean los títulos únicos o no. Veamos en qué parte hacemos cada uno de estos pasos:

map Como queremos encontrar los títulos, utilizaremos ese campo como key y el score y los votos como values para luego poder ordenarlos y filtrarlos por esos campos respectivamente: este campo:

```
emit(this.title, {score: this.score, votes: this.total_votes});
```

reduce el reduce se encargará de sumar los votos y los score de los posts que tengan el mismo título. De esta forma obtendremos para cada título la cantidad total de votos que tiene entre todos los posts que tienen ese título y los mismo con el score:

```
var reducedValue = {score: 0, votes: 0};
for (var i = 0; i < values.length; i++) {
  reducedValue.score += values[i].score;
  reducedValue.votes += values[i].votes;
}

return reducedValue;
```

finalize finalmente lo que hacemos es un filtrado semántico de los documentos asignándoles un score de -1 a los que tengan un cantidad de votos menor a 2000. De esta forma cuando los ordenemos por score éstos necesariamente van a quedar últimos y será fácil identificarlos:

```
if (value.votes < 2000) {
  return -1;
}

return value.score;
```

Por último, para poder obtener los 12 con mayor score nos resta ordenarlos decrecientemente por score y quedarnos con los 12 mayores:

```
db.title_score.find().sort({'value':-1}).limit(12);
  { "_id" : "The Bus Knight", "value" : 21953 }
  { "_id" : "Haters gonna hate", "value" : 14098 }
  { "_id" : "So my little cousin posted on FB that he was bored and gave
     everyone his new phone number... (pic)", "value" : 12333 }
```

```
{ "_id" : "My friend calls him &quot;Mr Ridiculously Photogenic Guy&quot;
    ", "value" : 11908 }
{ "_id" : "This is called humanity.", "value" : 10263 }
{ "_id" : "Genius", "value" : 9980 }
{ "_id" : "Seems legit.", "value" : 9530 }
{ "_id" : "President Obama's new campaign poster", "value" : 8942 }
{ "_id" : "Poster ad for the Canadian Paralympics", "value" : 8752 }
{ "_id" : "Fuck the police", "value" : 8746 }
{ "_id" : "Soon...", "value" : 8669 }
{ "_id" : "I'm sorry pinata bro", "value" : 8314 }
```

1.3. Ejercicio 3

Este ejercicio es muy parecido al primero. Debemos obtener la cantidad de comentarios promedio de los 10 mejores scores. Lo único que cambiará con respecto al ejercicio 1 será el campo utilizado como key y el campo sobre el cual se realizará el promedio pero en cuanto el formato del MapReduce se preserva casi intacto:

map utilizamos el score como key, ya éste es el campo por el cual nos interesa agrupar los resultados y como value utilizaremos los comments ya que sobre ellos calcularemos el promedio:

```
emit(this.score, {comments: this.number_of_comments, qty: 1});
```

reduce el reduce una vez más lo que hace es calcular la suma total de comentarios por un lado y por otro la cantidad de posts sobre los cuales se obtuvieron esos comentarios para luego poder realizar el promedio:

```
var reducedValue = {comments: 0, qty: 0};
for (var i = 0; i < values.length; i++) {
  reducedValue.comments += values[i].comments;
  reducedValue.qty += values[i].qty;
}

return reducedValue;
```

finalize el finalize simplemente calcula el promedio:

```
return value.comments/value.qty;
```

Finalmente, para poder obtener los 10 mejores scores y sus respectivos promedios de comentarios los ordenamos y nos quedamos con los 10 primeros:

```
db.scores_comments_average.find().sort({_id:-1}).limit(10);
  { "_id" : 20570, "value" : 1463 }
  { "_id" : 12333, "value" : 1612 }
  { "_id" : 11908, "value" : 2681 }
  { "_id" : 10262, "value" : 1514 }
  { "_id" : 8935, "value" : 480 }
  { "_id" : 8835, "value" : 1716 }
  { "_id" : 8699, "value" : 934 }
  { "_id" : 8241, "value" : 571 }
  { "_id" : 7297, "value" : 1110 }
  { "_id" : 6741, "value" : 2204 }
```

1.4. Ejercicio 4

En esta ocasión debemos encontrar al usuario con al menos 5 envíos que tenga mayor cantidad de upvotes. Primero, calcularemos para cada usuario su cantidad de envíos y de upvotes. Luego, nos quedaremos con aquellos que tengan al menos 5 envíos y para encontrar el que mayor cantidad de upvotes tiene los ordenaremos decrecientemente por cantidad de upvotes.

map Indexamos por el campo username y utilizamos upvotes para llevar cuenta de la cantidad de upvotes y submissions para luego poder contar la cantidad de envíos que tiene ese usuario:

```
emit(this.username, {upvotes: this.number_of_upvotes, submissions:
    1});
```

reduce como ya hicimos en otros ejercicios contamos con dos campos, uno que tendrá la sumatoria total de upvotes y otro para realizar el conteo de cuántos posts comprende el conteo. El mismo formato fue utilizado a la hora de calcular promedios:

```
var reducedValue = {upvotes: 0, submissions: 0};
for (var i = 0; i < values.length; i++) {
  reducedValue.upvotes += values[i].upvotes;
  reducedValue.submissions += values[i].submissions;
}

return reducedValue;
```

finalize aquí utilizaremos una vez más un filtrado semántico en el cual asignaremos una cantidad de upvotes igual a -1 a los users que cuenten con una cantidad menor a 5 envíos para que sea fácil identificarlos y se ubiquen últimos con respecto a un orden decrececiente de upvotes:

```
if (value.submissions > 5) {
  return -1;
}

return value.upvotes;
```

Por último, desde la consola cliente de Mongo, ordenamos a los users por upvotes y nos quedamos con el primero:

```
db.users\_upvotes.find().sort({value: -1}).limit(1);
  { "\_id" : "lepry", "value" : 90396 }
```

1.5. Ejercicio 5

Este último ejercicio requiere que para todos los subreddit que posean un score entre 280 y 300 indiquemos la cantidad de palabras presentes en sus títulos. Este ejercicio sigue un formato muy parecido al 4. En este caso agruparemos por subreddit en lugar de por usuario, nos interesa contar la cantidad de palabras en los títulos en lugar de los upvotes y filtraremos por score en lugar de por envíos. Primero contaremos la cantidad de palabras presentes en los títulos de cada subreddit. Luego filtraremos a los que tengan un score que no caiga dentro del rango [280, 300].

map aquí agrupamos por subreddit, utilizándolo como key, y utilizamos el score y la cantidad de palabras del título como values:

```
emit(this.subreddit, {score: this.score, title_word_count: this.
    title.split(" ").length});
```

reduce en el reduce una vez más nos dedicamos a acumular campos, en este caso la cantidad de palabras del título y el score.

finalize aquí relizamos nuevamente el filtrado semántico señalando a aquellos que no caigan dentro del rango de nuestro interés asignándoles un -1 como value:

```
if (value.score >= 280 && value.score <= 300) {
  return value.title_word_count;
}

return -1;
```

En este caso, a diferencia del resto, sólo queremos filtrar a aquellos que el finalize nos señaló como inválidos:

```
db.default.find({value: {$gt: -1}})
  { "\_id" : "Feminism", "value" : 6 }
  { "\_id" : "Firearms", "value" : 24 }
  { "\_id" : "HeroesofNewerth", "value" : 6 }
  { "\_id" : "Sexy", "value" : 13 }
  { "\_id" : "TheRealZachAnner", "value" : 4 }
  { "\_id" : "anime", "value" : 18 }
  { "\_id" : "ragecomics", "value" : 4 }
  { "\_id" : "xkcd", "value" : 4 }
```

2. Investigación