TP3 - Algoritmos en sistemas distribuidos

Sistemas Operativos - Primer Cuatrimestre de 2014

Límite de entrega: Martes 8 de Julio a las 23:59

Introducción

Compañías como *Amazon* generan y almancenan a diario una cantidad de datos colosal y por ende necesitan algoritmos distribuidos para poder procesar esos datos en tiempos razonables. El área denominada con el nombre de *Big Data* ha tenido una gran repercusión durante los últimos años proponiendo diversas maneras de abordar esta problemática. Dado que en la Escuela de Ciencias Informáticas del corriente año la facultad proveerá varios cursos sobre el área en cuestión, en la materia decidimos experimentar con algoritmos distribuidos y conjuntos de datos reales.

Con el fin de poder realizar una prueba de concepto obtuvimos un set de datos abierto de *Amazon*. El objetivo del trabajo es realizar experimentos sobre este conjunto de datos con algoritmos para sistemas distribuidos mediante la utilización de la técnica *Map-Reduce*. Mediante la utilización de un servidor de base de datos *MongoDB*, emulamos la distribución real de los procesos.

El trabajo se dividirá en dos secciones, una donde se implementarán pequeños algorítmos de análisis de datos y otra sección donde se hará un análisis de escalabilidad y performance de la arquitectura.

Datos

El subconjunto de datos de *Amazon* que eligió la materia consiste en más de quinientos mil reseñas reales de películas que se venden (o vendieron) en la tienda virtual. Cada reseña contiene los siguientes campos:

- `_id`: Id de la reseña.
- `productId`: Id del producto en la base de Amazon.
- `userId`: Id del usuario en la base de Amazon.
- `profileName`: Nickname del usuario.
- `helpfulness`: Valoración de la reseña por otros usuarios.
- `score`: Valoración de la reseña para el producto.
- `time`: Timestamp de la reseña.
- `summary`: Resumen textual de la reseña.
- `text`: Texto completo de la reseña.

Implementación Map-Reduce

El lote de datos se encuentra almacenado en db.tp3, para acceder desde la terminal:

```
$ mongo
> use tp3
```

En la colección *reviews* [1] se encuentra la información como un Json que tiene la estructura definida previamente:

Para ver el primer elemento en lote de datos deberán hacer:

[1] Para que puedan ejercitar con tiempos de ejecución menores, creamos dos bases de datos con menor cantidad de registros llamadas *reviewsFew* y *reviewsVeryFew* respectivamente.

```
> db.reviews.find()[0]
{
"_id" : ObjectId("539ccb72c87ae5335cc1abd2"),
"text" : "As if Ginger didn't have enough to deal with between PMS and Puberty!
 Things really fall apart when she's bitten by a werewolf. I love the bits of oddly
  placed humor, and the extra dose of teenage angst gives the movie an edge
   over your typical 'teen horror' film. This is a great movie for that Halloween
    party (though you might not want to eat in front of it). I enjoyed the third in
     the series (Ginger snaps back) the most, but this is a great watch too.<br
      /><br />Chrissy K. McVay - Author",
"userId" : "A1I7QGUDP043DG",
"summary" : "Great Halloween flick!",
"score" : "5.0",
"helpfulness" : "1/2",
"time" : "1162252800",
"profileName" : "Chrissy K. McVay \"Writer\"",
"productId" : "B008X1O7KM"
}
```

Utilizando estos datos deberán realizar los siguientes análisis:

1. Encontrar las doce películas mejor rankeadas de la colección de reseñas con al meños veinte reseñas. Buscando por el *productId* en el sitio de Amazon se encuentra el título.
2. Para cada puntaje de review (1-5) encontrar las palabras más frecuentes en sus respectivas reseñas sin contar stop words (and, the, for...).
3. Para cada puntaje de helpfulness, calcular la longitud promedio del review text.
4. A partir del resultado del punto anterior, calcular la correlación entre ambas magnitudes.

Para implementar cada uno de los análisis, deberán crear la función *Map* en un archivo `map.js` y la función *Reduce* en un archivo `reduce.js` y de ser necesario la función *Finalize* en un archivo `finalize.js`. Como su extensión lo indica, estas funciones deben contener código Javascript. Para ejecutar los análisis cuentan con el script de Python `runner.py`.

```
$ python runner.py
```

Para mas información consultar:

- El código fuente.
- Map-Reduce en Mongo: `http://docs.mongodb.org/manual/core/map-reduce/`
- JavaScript: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide.

Se pide que entreguen un informe completo con la explicación detallada y la justificación de las funciones que utilicen en cada uno de los ejercicio. Además deben presentar los resultado obtenidos en cada caso. Por último deben presentar un informe sobre una propuesta para aumentar la escalabilidad del sistema que se detalla en la siguiente sección.

Análisis de escalabilidad

Está claro que al crecer los datos y aumentar la complejidad de los análisis, una sola máquina no es suficiente para procesar los algoritmos en tiempos razonables. Una posible solución es contratar los servicios de plataformas del estilo de *HaaS* (Hardware as a Service) o *Cloud Computing* para usuarios que necesitan aumentar su poder de computo y/o almacenamiento. Se solicita entonces, en primer lugar, una búsqueda bibliográfica de esta alternativa. Los docentes de la materia sugerimos que analicen los servicios que brinda el cloud de Amazon, pero tienen libertad de elección. Luego

se pide que elaboren un informe, de no más de una carilla, proponiendo la implementación de esta alternativa. Debe considerarse que el tamaño de los datos tendrá un crecimiento lineal anual. El análisis debe incluir un presupuesto económico, una breve descripción de cómo se implementaría y también una evaluación informal del eventual impacto en el rendimiento.