Universidad de los Andes > FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

Tarea N°1
Almacenamiento y Procesamiento Masivo de Datos
"Sistemas SQL y NoSQL"

Alumno:
Juan Pablo Arévalo

Profesor: Ignacio Pérez

20/08/2017

Resumen

El objetivo de esta tarea fue familiarizarse con los sistemas SQL y NoSQL para almacenamiento y manejo de datos. Además de aprender también sobre arquitecturas utilizadas actualmente por grandes compañías.
La tarea consiste de dos partes:

1) Primero se trabajará con la API Twitter para descargar tweets y almacenarlos en alguna base de datos. Se procederá a estudiar los datos y calcular métricas por medio de un sistema SQL y uno NoSQL, para luego poder comparar su rendimiento y efectividad. Los resultados obtenidos se presentarán más adelante en este informe.

2) La segunda parte consiste de investigar y analizar la infraestructura de datos de alguna empresa conocida/exitosa, comparando con los conceptos vistos en clases.

Índice:

Parte 1 - Sistemas SQL y NoSQL:

Como se mencionó anteriormente, se trabajó con la API de Twitter para descargar tweets, con el objetivo de descargar alrededor de 2 Gb de tweets para luego analizarlos.

Para este ejercicio, se comenzó descargando tweets relacionados al término "rickandmorty"[1]. Luego de aproximadamente 6 horas descargando, se había descargado alrededor de 200 mb de tweets relacionados a dicho término. Al darme cuenta que para descargar los 2 Gb deseados me tomaría casi 10 veces el tiempo ya transcurrido, decidí cambiar el término de búsqueda por el que venía proporcionado por defecto por parte del profesor: "trump"[2].
Con el nuevo término logré descargar los 2 Gb de tweets (1.059.271 tweets en total) en alrededor de 5 horas, y pude luego proceder a analizar y estudiar los datos obtenidos. Cabe mencionar que los datos comenzaron almacenandose en un sistema NoSQL (MongoDB, específicamente) y luego se exportaron a formato .CSV para luego ser importados a un sistema SQL (PostgreSQL).

Una vez migrados los datos se pudo proceder a calcular distintas métricas por medio de consultas a ambas bases de datos (SQL y NoSQL).

Las métricas que se calcularon fueron las siguientes:

1) Tweets agrupados por hora de un determinado día.
2) Tweets agrupados por lenguaje
3) Tweets agrupados por país de emisión (ubicación)
4) Tweets agrupados por presencia de distintas palabras

Personalmente, en un comienzo pensé que MongoDB sería mucho más rápido que PostgreSQL en las consultas, ya que había escuchado que este tipo de bases de datos son especiales para un alto número de datos, y en esta tarea existía una gran cantidad de datos a analizar.

A continuación se explicará cada métrica y se presentarán los resultados obtenidos en los dos sistemas mencionados anteriormente:

[1] https://en.wikipedia.org/wiki/Rick_and_Morty
[2] https://en.wikipedia.org/wiki/Donald_Trump

1) **Tweets Agrupados por Hora de un Determinado Día:**

Esta métrica consiste en agrupar todos los tweets descargados y agruparlos por la hora en la que fueron publicados, realizando finalmente un conteo del número de tweets que se publicaron en cada intervalo de hora, para cada día por separado. Para hacer esto, se utilizó el campo "timestamp_ms" que indica el año, día, fecha y hora en que se publicó el tweet, este valor viene en formato Timestamp (Epoch/Unix)[3], por lo que se debió convertir a Date.

Resultados:

Como se puede ver en la tabla a continuación, es bastante mayor el número de tweets durante horarios diurnos (13:00-16:00), siendo muy bajos en horario de madrugada (03:00), comportamiento que a mi parecer es bastante normal, considerando el horario de actividad de las personas.

tweets	hora
84510	2017-08-09 12:00:00-04
133019	2017-08-09 13:00:00-04
127724	2017-08-09 14:00:00-04
134054	2017-08-09 15:00:00-04
104892	2017-08-09 16:00:00-04
37196	2017-08-09 17:00:00-04
14732	2017-08-10 00:00:00-04
62104	2017-08-10 01:00:00-04
44953	2017-08-10 02:00:00-04
9719	2017-08-10 03:00:00-04
114596	2017-08-10 12:00:00-04
37232	2017-08-10 13:00:00-04
109759	2017-08-11 11:00:00-04
25646	2017-08-11 12:00:00-04
19135	2017-08-17 07:00:00-04
1059271	total

[3] https://en.wikipedia.org/wiki/Unix_time

- PostgreSQL:

    - Tiempo de ejecución: 2654 ms
    - Observaciones: El tiempo de ejecución fue bastante rápido considerando que se trabajó con más de 1.000.000 de tweets. No se consumió un porcentaje importante de memoria (~10%)

- MongoDB:

    - Tiempo de ejecución: 59319 ms
    - Observaciones: El tiempo fue bastante lento, casi un minuto de espera, y la consulta fué bastante más complicada de formular, aún así los resultados fueron favorables y no se consumió un porcentaje importante de memoria (~20%)

2) **Tweets Agrupados por Lenguaje:**

Esta métrica consiste en agrupar todos los tweets descargados y agruparlos por el lenguaje en el que fueron escritos. Se utilizó el capo “lang” para determinar el lenguaje en el que fue escrito cada tweet. Para agrupar mejor los resultados, se juntaron los lenguajes con números de tweets pocos significativos en la categoría “Otros”.

Resultados:

Como se puede ver, el lenguaje predominante es el Inglés, alcanzando aproximadamente un 90% de los lenguajes encontrados, seguido por el español. Es bastante razonable que el lenguaje predominante sea inglés, dado que Donald Trump es el actual presidente de los Estados Unidos. Existe una gran cantidad de “indefinidos”, probablemente debido a la no especificación o reconocimiento del lenguaje en algunos tweets (muy posible que sean tweets con GIFs, links o imágenes y que no presenten texto).

tweets	lang	
37809	undefined	Undefined
2713	pt	Portuguese
4587	de	German
109	ru	Russian
11130	fr	French
22020	es	Spanish
1929	nl	Dutch
982	in	Indonesian
718	tl	Tagalog
520	sv	Swedish
1357	it	Italian
2020	tr	Turkish
969937	en	English
3440	other	Other
1059271	total	

- PostgreSQL:

    - Tiempo de ejecución: 294 ms
    - Observaciones: La consulta se ejecutó rápidamente, sin ocupar un porcentaje perceptible de memoria.

- MongoDB:

    - Tiempo de ejecución: + 1500000 ms
    - Observaciones: La consulta no logró completarse, se ejecutó durante aproximadamente 25 minutos finalizando en una desconexión del servidor de MongoDB (timeout). Durante este tiempo, el computador quedó pegado/congelado, utilizando el 100% de la memoria. A mi parecer fue una consulta muy ineficiente, que ni siquiera logró completar.

3) Tweets Agrupados por País de Emisión:

Esta métrica consiste en agrupar todos los tweets descargados y agruparlos basándose en el país de emisión del tweet. Para realizar esto, se utilizó el campo “place”, el cual en sí representa una estructura de datos (estilo JSON[4]), que contiene dentro el país, ciudad, código de país, estado, etc. Para realizar esta métrica se consideró particularmente el atributo “Country” (país) del campo “place”.

Resultados

Como se ve a continuación, se habla de Trump en todo el mundo, pero nuevamente predominantemente en Estados Unidos. Es importante notar que aproximadamente un 99% de los resultados se presentan como “indefinido”, probablemente debido a que este parámetro no se utiliza mucho o no presenta gran importancia para Twitter. También es probable que este parámetro requiere de conexión a GPS para registrarse, lo cual limitaría mucho los resultados ya que la gran mayoría de la gente podría preferir no guardar su ubicación cada vez que hace un tweet o simplemente no tendría su GPS encendido.

tweets	country
13	Costa Rica
31	Nigeria
27	España
60	Australia
11	Argentina
47	Brazil
12	Israel
13	Chile
10	New Zealand
35	Mexico
5	Peru
22	Venezuela
61	The Netherlands
8	Uruguay
10	Spain
47	Ireland
3	Hong Kong
17	Türkiye
13	Portugal
20	North Korea
83	México
409	Canada
22	Austria
31	South Africa
598	United Kingdom
32	Italy
28	Sweden
33	India
2	Paraguay
49	Deutschland
1	Bolivia
17	Singapur
28	Germany
83	France
2	Iraq
3	Islamic Republic of Iran
7	Colombia
7	Japan
7	Ecuador
8	Denmark
10	Russia
8339	United States
1049007	Undefined
1059271	total

[4] https://en.wikipedia.org/wiki/JSON

- PostgreSQL:

    - Tiempo de ejecución: 154 ms
    - Observaciones: La consulta se ejecutó rápidamente sin ocupar un porcentaje perceptible de la memoria del computador.

- MongoDB:

    - Tiempo de ejecución: 42115 ms
    - Observaciones: La consulta fue algo lenta en comparación a psql, pero se ejecutó sin problemas y sin ocupar un porcentaje significativo de la memoria (~15%)

4) Tweets Agrupados por Presencia de Distintas Palabras:

Esta métrica consiste en agrupar todos los tweets descargados que presentaran la aparición de una determinada palabra. Para eso, se utilizó el campo “text” de los tweets, comparando con una expresión regular para revisar la aparición de una determinada palabra. Esta expresión regular encuentra una determinada palabra en cualquier parte del texto (puede considerar también una subpalabra dentro de otra palabra, pero la mayoría de las palabras elegidas no se prestan como para ser consideradas subpalabras de otras). La búsqueda es Case Insensitive, es decir, no se vé afectada por mayúsculas o minúsculas.

Resultados:

Esta métrica se llevó a cabo simplemente por curiosidad, intentando buscar una relación entre “trump” y otras palabras que se repitieran a menudo en los tweets descargados. Para esto, se investigó en internet y se confeccionó un set de palabras generalmente vinculadas al Presidente de los Estados Unidos, Donald Trump. A continuación se puede ver una tabla donde se muestra el número de tweets que relacionan a Trump con una determinada palabra. No es una sorpresa que la palabra más mencionada fuera Korea (en aproximadamente un 10% de los tweets) ya que durante la fecha de descarga de los datos ocurrió el conflicto entre el presidente Trump y Kim Jong-Un donde se habló sobre un posible ataque nuclear. “Korea”, “War” (guerra) y “Nuclear” fueron las palabras que presentaron mayor relación con los tweets que mencionan al actual presidente de los Estados Unidos.

tweets	word
3285	racist/racism
36769	obama
14876	hillary
10905	f**k
6527	stupid
82377	war
9591	bomb
8456	die
44506	america
72103	president
3921	wall
1275	terrorist/terrorism
808	mexican
1853	immigrant
4094	idiot
442	genius
3925	hitler
108557	korea
37801	kim (jong-un)
39826	threat
67248	nuclear

- PostgreSQL:

    - Tiempo de ejecución: ~2000 ms (por palabra)
    - Observaciones: Para cada palabra, el tiempo fue algo lento, pero nada fuera de lo normal. Se ocupó poca memoria (~10%)

- MongoDB:

    - Tiempo de ejecución: ~38600 ms (por palabra)
    - Observaciones: El tiempo de ejecución fue nuevamente mayor al de PostgreSQL, pero la consulta se completó finalmente sin ocupar un porcentaje preocupante de memoria (~25%).

Índices

MongoDB:

En MongoDB, los índices ayudan a que las consultas se ejecuten más eficientemente, evitando que se revise cada documento uno por uno para revisar la condición de la consulta que se está realizando.

Por lo tanto, al tener un índice, MongoDB puede usarlo para limitar el número de documentos a inspeccionar y hacer más eficiente el proceso.

PostgreSQL:

En PostgreSQL, los índices, al igual que en MongoDB, se utilizan para ejecutar de manera más eficiente las consultas, ayudando al servidor a encontrar filas de una manera más rápida.

Estos índices se asignan a columnas de la tabla para facilitar la búsqueda de elementos en base a dichas columnas.

Conclusiones Parte 1

En conclusión, la tarea fue muy fructífera para aprender sobre el funcionamiento de bases de datos SQL y NoSQL, pero me encontré con varios problemas, los cuales terminaron por ocupar gran parte de mi tiempo.

Por ejemplo, comencé intentando ocupar MySQL en lugar de PostgreSQL, lo cual nunca logré a pesar de dedicarle varias horas a intentar migrar los datos desde MongoDB. Finalmente cuando ya intenté todas las formas de importación decidí cambiarme a PostgreSQL y la importación funcionó en el primer intento.

Por otra parte, como se mencionó anteriormente, perdí gran parte del tiempo descargando tweets de “rickandmorty”, los cuales descargaban a una velocidad muy lenta, de lo cual no me pude percatar ya que no conocía el funcionamiento correcto del programa ni de la API, por lo que lo asumí como normal.
Una vez que me di cuenta que mis compañeros descargaban varios Gigabytes en cuestión de horas, decidí cambiar al parámetro por defecto “trump” y logré finalmente guardar 2 Gb de datos.

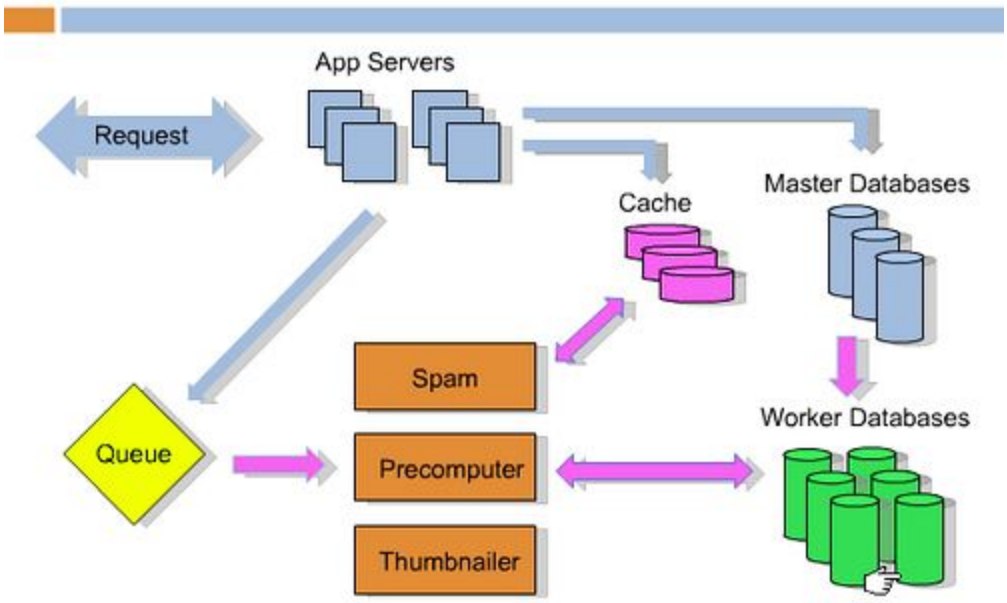
Durante el proceso de desarrollo, pude notar que las consultas en MongoDB se ejecutaban de manera menos eficientes que en PostgreSQL, con una diferencia bastante significativa. Pero luego de aprender sobre índices logré entender cómo ese proceso se podría optimizar, para obtener así tiempos más aceptables en las consultas de búsqueda.

A modo de concluir esta parte de la tarea, puedo decir que logré experimentar en primera persona lo complicado que puede ser lidiar con un set grande de datos, donde una equivocación en el código, o una consulta mal escrita puede significar agotar los recursos de tu computador y obligar a un reinicio forzado.

Comprendí lo complejo de manejar un gran número de datos y lo pesado que puede ser para un computador esta tarea, pero también aprendí cómo optimizar esto y llegar a resultados en cuestión de segundos, cosa que un humano no podría conseguir por su cuenta ni con varios días de análisis.

Parte 2 - Infraestructura de Datos

REDDIT:



Para esta parte de la tarea, elegí estudiar la infraestructura de datos de la página web Reddit.

La imagen mostrada más arriba es un bosquejo de la arquitectura de Reddit.
Explicada de manera rápida, los servers de Reddit reciben un request (acción de algún usuario, cambio, publicación, etc.), el cual es SIEMPRE guardado en caché (memcache) y al mismo tiempo se envía a una base de datos "master" y a una cola para ser procesado y almacenado en la base de datos de reddit.

La arquitectura de Reddit se concentra en 7 puntos importantes[5]:

1) Reiniciar automáticamente servicios fallidos.
2) Separar procesos y datos en distintos "boxes" o servicios.
3) No complejizar mucho el schema (base de datos)
4) Mantenerla sin estado (stateless/REST)
5) Almacenar TODO en memcache
6) Almacenar todos los datos, pre computarlos y guardarlos en caché.
7) Hacer el mínimo de trabajo en backend y mostrar los resultados al usuario. el resto hacerlo después.

[5] http://highscalability.com/blog/2010/5/17/7-lessons-learned-while-building-reddit-to-270-million-page.html

En resumen, en Reddit se concentran en almacenar y procesar todos los datos y requests provenientes de los usuarios, guardando todo en caché y procesando todos los datos en el backend luego de entregar una respuesta al usuario, ya que lo más importante es procesar rápidamente el request y entregar una respuesta simple y rápida (es decir, con una baja latencia).

Con un poco más de investigación[6], logré averiguar qué Reddit utiliza una base de datos en PostgreSQL, almacenando todos sus objetos (usuarios, threads, comentarios, etc.) como "things" en una tabla Thing, con atributos compartidos. Además tiene también una tabla de "datos" donde se almacenan todos los atributos de los objetos "thing" que no se consideran en la tabla Thing.

Con su modelo de base de datos se puede ver el énfasis en el punto 3 previamente mencionado, manteniendo lo más simple posible su modelo de datos.
Esta simplicidad ayuda a utilizar este modelo de datos con una arquitectura de tipo lambda, es decir, procesa los datos como cadenas de ítems individuales o en forma de batch simultáneamente, para poder así mejorar la latencia, throughput y tener una mayor tolerancia a fallas.

Utilizan además el datastore Cassandra, el cual realiza sharding, lo cual, como vimos en clases, es una separación (partición) de datos en distintos servidores (bases de datos). Esto agrega una precaución ante cualquier posible falla del hardware, aumentando aún más la tolerancia a fallas del sistema.

A pesar de que aún no comprendo mucho sobre el tema de cache, vale la pena mencionar que en el sistema de Reddit se concentran fuertemente en realizar memcache, principalmente utilizando sgm, pagecache, rendercache y memoize.

Es también interesante mencionar que los requests recibidos por el sistema se almacenan en una cola de RabbitMQ la cual luego distribuye los diferentes mensajes para ser procesados por los diferentes trabajos en backend.

En conclusión, Reddit aplica varios de los principios estudiados en clases, pero con varias singularidades que sobresaltan, a mi parecer, de lo convencional. Como por ejemplo la simplicidad de su modelo de datos o la tendencia a almacenar todo en Caché.
Sin embargo, con el éxito de Reddit actualmente, un sitio web visitado millones de veces por día desde todo el mundo, no cabe duda de que esta infraestructura se adecua completamente a lo que se requiere para un sistema de tal envergadura.

[6] https://github.com/reddit/reddit/wiki/Architecture-Overview