*Tarea I*

*Almacenamiento y Procesamiento Masivo de Datos*

*Vicente Nazar Muñiz*

*ICC0012 | 20 de agosto del 2017*

Índice

[1 Primera Parte 2](#_Toc490957772)

[1.1 Herramientas y tecnologías 2](#_Toc490957773)

[1.2 Datos 2](#_Toc490957774)

[1.3 Métricas 3](#_Toc490957775)

[1.4 Índices 3](#_Toc490957776)

[1.5 Resultados 3](#_Toc490957777)

[1.6 Dificultades 4](#_Toc490957778)

[1.7 Conclusión y expectativas 5](#_Toc490957779)

[2 Segunda Parte 6](#_Toc490957780)

# Primera Parte

## Herramientas y tecnologías

Para el desarrollo de esta tarea se utilizaron las siguientes bases de datos:

* **Relacional:** PostgreSQL (versión 9.5.7)
* **No-Relacional:** MongoDB (versión 3.4.7)

Mientras que para la ejecución del script “tweepy.py” se utilizó:

* Virtualenv 15.1.0
* Python 2.7.12

Se utilizó como sistema operativo:

* Ubuntu Server 16.10

Para el monitoreo de las consultas se utilizó:

* **CPU/RAM:** htop
* **I/O Disco:** iostat

Finalmente, como hardware:

* **CPU:** 2 Cores (3.0GHz)
* **RAM:** 2 GB
* **Disco:** 50 GB (HDD)

## Datos

Para la tarea se descargaron 1,053,007 tweets (2.561GB) siendo estos filtrados por:

* “north korea”
* “northkorea
* “dprk”
* “kimjongun”
* “kim jong un”

Luego de descargar los tweets y guardarlos en la base de datos No-Relacional se hizo una transformación del campo “timestamp\_ms” para poder trabajar con él y extraer la hora y día.

Después de eso se exportaron los datos (solo los campos necesarios) en formato CSV de tal manera de ser importados a la base de datos relacional de forma sencilla. Para esto se utilizó la herramienta “mongoexport” incluida en MongoDB.

Luego se creó una tabla en la base de datos relacional con las columnas necesarias con los tipos de datos correspondientes.

**\* Todos los códigos y comandos utilizados se encuentran en el repositorio GitHub.**

## Métricas

Para el análisis de los datos se realizaron las siguientes métricas por medio de consultas SQL y No-SQL:

* **Tweets por hora (Query#01):** cantidad de tweet hechos por cada hora y día de la semana.
* **Tweets por ubicación geográfica (Query#02):** cantidad de tweets por ubicación y que se encuentre dentro de una zona europea (longitud: [-10.195312, 31.113281], latitud: [36.031332, 62.021528]).
* **Cantidad de retweets (Query#03):** Agrupa todos los tweets originales que han sido retuiteados y los cuenta, además solo considera los que han sido retuiteados mas de 1000 veces.

## Índices

El rol de los índices en las bases de datos es la de mapear los valores del campo/columna indicado(a), de tal manera de no hacer una búsqueda de valores uno por uno, sino que utilizando algoritmos óptimos (ej: B-tree) para encontrar los valores en un tiempo más acotado. De esta forma cuando se quiera acceder a un valor (el cual estará mapeado) se tendrá acceso a él sin tener que hacer una búsqueda de fila por fila (o documento por documento).

Para esta parte de la terea crearon los siguientes índices:

* **Query#01:** timestamp\_ms
* **Query#02:** coordinates.coordinates.0 y coordinates.coordinates.1
* **Query#03:**

## Resultados

Los siguientes resultados muestras el tiempo de ejecución para cada consulta con sus respectivas bases de datos. Los campos “Write” y “Read” señalan los MB escritos y leídos en el disco durante sus procesos respectivos, mientras que el campo CPU señala el porcentaje de CPU usado por el usuario (mongod/postgres) en el momento, siendo este siempre igual a 0% en un estado inicial y aumentado a lo señalado en cada consulta. Finalmente se muestra la columna RAM la cual indica los MB usados por el proceso (mongod/postgres) al momento de la ejecución de la consulta.

**\* Para cada una de las siguientes pruebas se reinició el servicio de la base de datos respectiva para cada consulta, de tal manera de hacer pruebas más limpias.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Consultas sin Índices (MongoDB)** | | | | | |
| **Consulta** | **Tiempo de ejecución (s)** | **Write (MB/s)** | **Read (MB/s)** | **CPU (%)** | **RAM (MB)** |
| Query#01 | 8.680 | 0.1 | 270.09 | 100% | 521 |
| Query#02 | 7.742 | 0.0 | 339.61 | 100% | 568 |
| Query#03 | 8.337 | 0.0 | 303.01 | 100% | 573 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Consultas sin Índices (PostgreSQL)** | | | | | |
| **Consulta** | **Tiempo de ejecución (s)** | **Write (MB/s)** | **Read (MB/s)** | **CPU (%)** | **RAM (MB)** |
| Query#01 | 1.236 | 0.1 | 0.0 | 61% | 21 |
| Query#02 | 0.990 | 0.0 | 0.57 | 4% | 22 |
| Query#03 | 0.222 | 4.0 | 0.0 | 12% | 27 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Creación de Índices (solo para consultas Query#02)** | | | | |
| **Base de Datos** | **Write (MB/s)** | **Read (MB/s)** | **CPU (%)** | **RAM (MB)** |
| MongoDB | 0.14 | 313.74 | 100% | 580 |
| PostgreSQL | 22 | 0 | 27% | 14 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Consultas con Índices (MongoDB)** | | | | | |
| **Consulta** | **Tiempo de ejecución (s)** | **Write (MB/s)** | **Read (MB/s)** | **CPU (%)** | **RAM (MB)** |
| Query#01 | 8.582 | 0.05 | 221.0 | 100% | 562 |
| Query#02 | 0.009 | 0.04 | 0.05 | 0.7% | 0 |
| Query#03 | 0.522 | 0.00 | 0.04 | 26% | 511 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Consultas con Índices (PostgreSQL)** | | | | | |
| **Consulta** | **Tiempo de ejecución (s)** | **Write (MB/s)** | **Read (MB/s)** | **CPU (%)** | **RAM (MB)** |
| Query#01 | 1.193 | 0.0 | 0.0 | 57% | 20 |
| Query#02 | 0.026 | 0.0 | 0.0 | 1% | 0 |
| Query#03 | 0.152 | 0.0 | 0.0 | 9.3% | 78 |

## Dificultades

En un principio las dificultades se dieron principalmente por:

* Manipulación de campo “timestamp\_ms” para la consulta Query#02 (en MongoDB).
* Manipular framework Agreggation de MongoDB.

Para el primer caso existió el problema de que el formato del campo “timestamp\_ms” entregado por la API de Twitter era en formato Unix Timestamp, de tal manera que la manipulación de este para la extracción de la hora y día de la semana no era posible. Siendo que para poder operar con ellos era necesario usar el formato Date (formato de BSON), fue necesario crear una consulta (mediante una búsqueda de consultas parecidas en Stackoverflow[[1]](#footnote-1)) para poder modificar este campo en todos los documentos para así operar con ellos. **La consulta se encuentra adjunta en el archivo timestampms\_conv.js**

Para el segundo caso fue más bien un problema por falta de conocimientos de la tecnología, de tal manera que se tuvo que realizar un estudio por medio de la documentación ofrecida por MongoDB[[2]](#footnote-2) (especialmente de la función aggregate). Finalmente se desarrolló una buena base sobre el funcionamiento de la funcionalidad de aggregation pipeline y de los destinos operadores que pueden ser integrados dentro de este.

## Conclusión y expectativas

Dentro de las complicaciones ya señaladas anteriormente, existió una tercera la cual fue relacionada con el cómo implementar de forma eficiente los índices para cada tipo de consulta, lo cual en un principio arrojaba resultados en largos tiempos de espera, pero luego de entender el funcionamiento haciendo distintas pruebas, fue posible llegar a un resultado aceptable.

Es importante destacar que para el buen aprovechamiento de los índices es necesario utilizarlos en los campos/columnas que son usados para buscar valores condicionales, ya sea por ejemplo dentro de un rango o igual a un valor dado. De esta manera se explica los resultados de la primera consulta (Query#01), la cual no presentó una mejora luego de haber creado un índice para el campo “timestamp\_ms”, debido a que, según lo señalado en la tarea misma, era solo necesario contar los tweets totales por cada hora y día. En el caso que se hubiese realizado una búsqueda o “match” (en el caso de MongoDB) sobre un intervalo de horas, entonces ahí si se hubiesen presentado resultados más optimistas dada la implementación del índice indicado.

Habiendo dicho esto, luego de haber estudiado el comportamiento de los índices, fue aceptable esperar y tener buenas expectativas en la rapidez de las consultas, pero al no tener una implementación adecuada de los índices, los resultados fueron llegaron a ser iguales o peores que sin la implementación de ellos. Luego de resolver este problema fue posible validar lo estudiado, y de presenciar excelentes resultados, entregando así tiempos de ejecución en milésimas de segundos.

**Finalmente …**

# Segunda Parte

1. https://stackoverflow.com/questions/9254351/how-can-i-rename-a-field-for-all-documents-in-mongodb [↑](#footnote-ref-1)
2. https://docs.mongodb.com/manual/aggregation/ [↑](#footnote-ref-2)