Evaluación y comparación del rendimiento de bases de datos no relacionales

Martín Rotti
Facultad de Ingeniería, Universidad de la República
Montevideo, Uruguay
martin.rotti@fing.edu.uy

Resumen

Las bases de datos no relacionales han venido en constante crecimiento en su uso debido a ser una opción diferente a las relacionales, brindando muchas ventajas cuando se trabaja con grandes volúmenes de datos. Las mismas tienen un enfoque distinto a las relacionales, siendo más flexibles en cuanto al modelo lógico e implementación. Dentro del esquema no relacional podemos tener distintos enfoques, como bases de datos basadas en documentos, tabulares, clave-valor o grafos. La elección de usar una base de datos no relacional para un sistema depende de varios factores: los datos a almacenar, arquitectura, precio, rendimiento esperado en ciertos flujos. Por esto, es importante a la hora de decidir que base de datos utilizar, tener una clara comparación entre diferentes enfoques que se basen en datos empíricos y permitan ayudar a elegir la mejor opción. El objetivo de este trabajo es realizar una evaluación de rendimiento entre las bases de datos MongoDB y Cassandra utilizando una herramienta que brinda ayuda en este proceso complejo de evaluación y comparación. Como resultados concluimos que MongoDB se comporta mejor que Cassandra, superándola en el rendimiento en todas las pruebas realizadas.

I. Introducción

Históricamente, los datos de un sistema de información se han almacenado en una estructura de tablas basándose fuertemente en el concepto de relación. Las bases de datos relacionales siguen este conocido modelo relacional y son, por lejos, las más elegidas a la hora de tener un sistema de información que requiere almacenar datos. Sin embargo, en la última década se ha producido un auge de las bases de datos no relacionales, siendo una clara opción diferente. Este tipo de bases de datos permiten tener más flexibilidad a la hora de guardar los datos, pudiendo tener más posibilidades para tomar decisiones en cuanto a diseño. Las mismas también proporcionan una capacidad de escalar más alta que las relacionales ya que al estar distribuidas se comportan mejor cuando se trabaja con grandes volúmenes de datos.

Al momento de elegir una base de datos para un sistema de información, es clave tener claro cuál es la que mejor se adapta a la realidad que se desea modelar. Para ello es muy importante conocer previamente cuáles son las ventajas y desventajas de las opciones y poder hacer una comparación entre ellas. Sin embargo, diseñar una evaluación de rendimiento de una base de datos y llevarla a cabo puede ser muy costoso en cuanto a tiempo y precio. En este trabajo, utilizaremos la herramienta NoSQLBench[1] que nos permitirá realizar una evaluación de rendimiento de distintas bases de datos no relacionales para poder compararlas entre sí. La utilización de esta herramienta permite que el proceso de evaluación y comparación sea mucho menos costoso ya que permite automatizar gran parte de las etapas para llevar a cabo el trabajo.

En la sección II hacemos una introducción a las herramientas que utilizaremos para llevar a cabo el trabajo, en la sección III hablamos sobre el proceso de diseño e implementación de cargas de trabajo, en la sección IV detallamos las pruebas realizadas y en la sección V analizamos y comparamos los resultados obtenidos y planteamos en qué aspectos se puede seguir avanzando para continuar este trabajo.

II. Trabajos relacionados

Existen diferentes tipos de bases de datos no relacionales, las documentales se basan en almacenar los datos en archivos (típicamente JSON o XML), las de grafos modelan la realidad con un grafo que tiene como nodos a los datos y las aristas representan las relaciones entre ellos, las de clave-valor guardan los datos como una colección de pares clave (única, permite recuperar el dato) y valor (representa al dato), las tabulares almacenan los datos en columnas y son un híbrido entre lo tabular y clave-valor.

II-A. MongoDB

MongoDB[2] es una base de datos que se compone de colecciones que poseen documentos. Los documentos se componen de parejas campo-valor y son de tipo Binary JSON (BSON), que siguen una notación similar a los JSON. Al no seguir un esquema predefinido, brinda una gran flexibilidad a la hora de modelar la manera en que se van a almacenar los datos, pudiendo tener documentos en la misma colección con diferentes estructuras. A su vez, permite tener documentos embebidos dentro de

[1] https://github.com/nosqlbench/nosqlbench
[2] https://www.mongodb.com

otros, pudiendo esto ser clave para lograr más eficiencia en las consultas. MongoDB tiene como característica que permite la escalabilidad tanto vertical (pudiendo agregar más recursos) como horizontal (pudiendo agregar más nodos), lo que hace que este gestor de bases de datos sea muy útil para el almacenamiento de grandes volúmenes de datos. Además cumple con tener una alta disponibilidad al incluir instancias de réplicas, lo que permite que al fallar una instancia, los datos están disponibles en otra. La estrategia de réplica de MongoDB se basa en tener un nodo primario (denominado maestro) y nodos secundarios. El nodo principal es el que recibe todas las operaciones de escritura y las replica en los nodos secundarios, para que el conjunto de datos se refleje en todos los nodos del sistema.

II-B. Cassandra

Cassandra[3] es una base de datos open source, distribuida y tabular. Está diseñada manejar grandes volúmenes de datos y generalmente la utilizan aplicaciones que requieren un gran rendimiento de operaciones de lectura y escritura. Cassandra garantiza la tolerancia a fallas y confiabilidad gracias a su estrategia de réplicas. Esta estrategia incluye tener múltiples nodos, es decir, muchas instancias de Cassandra, que se comunican entre sí a través del protocolo Gossip que es utilizado para comunicar máquinas entre pares. En Cassandra no existe un nodo maestro, todos los nodos proporcionan las mismas funcionalidades, sin existir una jerarquía entre ellos. Una de las grandes ventajas de esta arquitectura ampliamente distribuida es que permite la escalabilidad horizontal y es muy sencillo agregar más nodos al sistema sin la necesidad de dejarlo inactivo por un tiempo. La escalabilidad horizontal que posee Cassandra permite que no se dependa del aumento de CPU, RAM o disco, que se sabe que tiene limitaciones y es costoso. Si se desea tener más potencia simplemente se puede agregar más nodos de forma indefinida, lo que es una gran ventaja cuando se trabaja con grandes volúmenes de datos.

II-C. NoSQLBench

NoSQLBench es una herramienta *open source* que permite realizar una evaluación de rendimiento para el ecosistema NoSQL. Brinda un conjunto de cargas de trabajo y controladores genéricos que permiten ejecutar pruebas de rendimiento para cualquier base de datos no relacional. Además de las cargas de trabajo predefinidas, NoSQLBench permite al usuario diseñar pruebas a medida sin requerir un gran esfuerzo en codificación, definiéndola en un solo archivo y basándose en una plantilla de declaraciones. Con esta herramienta, se puede generar conjuntos de datos de un tamaño arbitrario y ser utilizados en las pruebas. En el momento de ejecución de una carga de trabajo, la herramienta brinda la opción de visualizar los resultados en una ventana aparte utilizando los software Grafana y Prometheus, automatizados por NoSQLBench. La herramienta es de muy fácil acceso y uso ya que no se requieren configuraciones complejas para ello.

Fue diseñada en principio por DataStax[4] para soportar el lenguaje de consulta CQL(Cassandra Query Language)[5], pero en los últimos años se extendió de manera genérica a otros tipos de sistemas no relacionales. Con respecto a la comunidad, si bien no es muy amplia, la misma tiene cada vez más colaboradores que aportan al proyecto y el mismo está en constante actualización, siendo que actualmente se encuentra en la versión 5. Es una herramienta que está desarrollada para ser utilizada en el sistema operativo Linux pero con un proceso un poco más complejo de configuración tiene soporte para Windows y Mac.

Para realizar una evaluación simplemente se debe clonar el proyecto, configurar la base de datos que se quiere evaluar y a través de la línea de comandos ejecutar una carga de trabajo. Luego de la evaluación, la herramienta genera tres archivos donde se reportan las acciones llevadas a cabo en la prueba y las estadísticas que se recogieron.

En el ejemplo 1 se muestra un comando básico para ejecutar una carga de trabajo para MongoDB.

Listado 1 Ejemplo de comando

```
./nb5 run driver=mongodb workload=mongodb-basic.yaml connection=mongodb://127.0.0.1 database=test --progress console:3s
```

Con el parámetro *run* le indicamos a la herramienta que ejecute la carga de trabajo definida en el archivo *mongodb-basic.yaml* utilizando el driver definido *mongodb*. Con el parámetro *connection* indicamos la dirección donde se ejecuta la instancia de MongoDB y en *database* indicamos el nombre de la base de datos utilizada para la prueba. Por último, con *–progress console:3s* indicamos que se muestre en la consola el estado de la ejecución cada 3 segundos.

III. Diseño de escenarios

Como se mencionó en II, la herramienta NoSQLBench brinda un conjunto de cargas de trabajo predefinidas y la posibilidad de diseñar otras a medida de las necesidades. Para este trabajo se tomó la decisión de diseñar distintas cargas de trabajo que varían entre sí en el número de operaciones de lectura, escritura y borrado que realizan.

Las cargas de trabajo se definen en un documento YAML y tienen un formato de configuración estándar que permite con facilidad crear nuevas. La parte más importante de una configuración son las operaciones, es allí donde efectivamente se va a

[3]https://cassandra.apache.org
[4]https://www.datastax.com
[5]https://cassandra.apache.org/doc/latest/cassandra/cql

interactuar con la base de datos, ya sea para crear el esquema, insertar, borrar o leer datos. Las plantillas se componen de tres secciones: *schema*, *rampup* y *main*.

En la sección *schema* es donde se crea el esquema en la base de datos a utilizar. Es decir, para el caso de Cassandra se crea el *keyspace* y las tablas, mientras que este paso no es necesario para MongoDB ya que no sigue un esquema específico.

En la sección *rampup* se crean datos base que servirán de apoyo para la siguiente sección. De esta manera, al iniciar el *main* ya se cuenta con un conjunto de datos insertados en la base de datos, los cuales se pueden leer, actualizar o borrar. El propósito del *rampup* es tener un número realista de datos de fondo en el sistema para que la prueba sea significativa.

En la sección *main* es donde se centra la prueba y allí se deben colocar las operaciones necesarias para llevarla a cabo y así poder obtener las métricas. Aquí se interactúa con el esquema y los datos creados en las secciones anteriores pudiendo accederlos, eliminarlos, actualizarlos o insertar más.

La generación de datos es una parte importante para las pruebas en NoSQLBench. Para esto la herramienta provee una biblioteca para crear conjuntos de datos virtuales, a través de los llamados *bindings*. De esta manera, se puede utilizar las funciones que provee para generar los datos que serán utilizados en las pruebas.

A continuación se muestra la definición de una carga de trabajo básica para Cassandra que consiste en crear la estructura de la base de datos, insertar valores con la ayuda de *bindings* para generar los datos y por último accederlos.

Listado 2 Ejemplo de una carga de trabajo

```
scenarios:
   default:
      schema: run driver=cql tags==block:schema threads==1 cycles==UNDEF
      rampup: run driver=cql tags==block:rampup cycles===TEMPLATE(rampup-cycles,10) threads=auto
      main: run driver=cql tags==block:"main.*" cycles===TEMPLATE(main-cycles,10) threads=auto

bindings:
  id: ElapsedNanoTime(); ToHashedUUID() -> java.util.UUID
  message: Discard(); FirstLines('data/cql-starter-message.txt');

blocks:
   schema:
      ops:
        create-keyspace: |
          create keyspace if not exists <<keyspace:example>>
        create-table: |
          create table if not exists <<keyspace:example>>.<<table:cqlexample>> (
          id UUID,
          message text,
          );
   rampup:
      ops:
        insert-rampup: |
          insert into <<keyspace:example>>.<<table:cqlexample>> (id, message)
          values ({id}, {message});
   main-read:
      ops:
        select-read: |
          select * from <<keyspace:example>>.<<table:cqlexample>>
          where id={id};
```

III-A. Escenarios

Para llevar a cabo las pruebas de evaluación del rendimiento de las bases de datos se decidió diseñar un conjunto de escenarios que varían entre sí en la cantidad de operaciones de lectura, escritura y borrado para abarcar distintos flujos de trabajo. El número total de operaciones en cada escenario fue de 500000. Cada carga de trabajo sigue el flujo definido, primero se ejecuta la sección *schema* para crear la estructura de la base de datos, luego sigue el *rampup* donde se hacen las inserciones de los datos base y luego se ejecuta el *main* para terminar con el proceso de operaciones. Cabe destacar que para el caso de MongoDB no es necesario ejecutar la sección *schema* debido a que no es necesario definir una estructura en esta base de datos ya que no sigue un esquema específico, por lo tanto las cargas de trabajo empezarán directamente con el *rampup*.

III-A1. Carga de trabajo C1: Esta primera carga de trabajo hace solo operaciones de escritura. Por lo tanto, al haber 500000 operaciones, la base de datos terminará con 500000 datos insertados al final de la ejecución. La estrategia de escritura fue de insertar 250000 datos en la fase de *rampup* y luego los restantes 250000 en la fase *main*.

III-A2. Carga de trabajo C2: En esta carga se llevan a cabo 375000 operaciones de escritura y luego 125000 operaciones de lectura. Todos los datos son insertados en la base de datos en la sección *rampup* y las operaciones de lectura se realizan en el *main*.

III-A3. Carga de trabajo C3: Esta carga realiza la misma cantidad de operaciones de escritura que de lectura. Es un escenario similar al anterior, variando el número de inserciones y lecturas, que en este caso son de 250000 cada una.

III-A4. Carga de trabajo C4: En esta carga de trabajo se realiza 125000 operaciones de escritura y luego 375000 operaciones de lectura. La estrategia es igual a la de las cargas anteriores, insertando los datos en la sección *rampup* y leyéndolos en la sección *main*.

III-A5. Carga de trabajo C5: En esta carga de trabajo se realizan las operaciones CRUD, siendo 250000 de ellas de escritura, 125000 de lectura y 125000 operaciones de borrado. En la sección de *rampup* se insertar 125000 datos, luego se accede a la misma cantidad, se actualizan 125000 y por último se eliminan 125000.

En el Cuadro I se puede ver la distribución, en porcentaje, de las operaciones para cada escenario.

IV. Experimentación

En esta sección se darán más detalles sobre el proceso de ejecución de las pruebas realizadas, el ambiente donde se llevaron a cabo y distintos parámetros configurados previo a correr las cargas de trabajo.

IV-A. Ambiente de trabajo

Las pruebas de rendimiento de las dos bases de datos se llevaron a cabo en una máquina con las siguientes características:

- Procesador: AMD Ryzen 7 5700U
- Núcleos: 16
- RAM: 16GB
- Sistema operativo: Ubuntu Server 22.04.2
- Disco: SSD 500GB

Se trabajó con las siguientes versiones:

- MongoDB: 6.0.6
- Cassandra: 4.1.2
- NoSQLBench: 5.17.3

IV-B. Ejecución de las pruebas

Todas las cargas de trabajo se corrieron 3 veces y se promedió el tiempo que llevaron en completarse. Para cada prueba se configuró que utilice un número de 160 hilos. Con respecto a los datos usados, los mismos se crearon a través de los *bindings* y consisten en *strings* de largo 20. En las figuras 1a, 1b, 2a, 2b y 3 se puede visualizar el resultado de las pruebas realizadas para cada carga de trabajo.

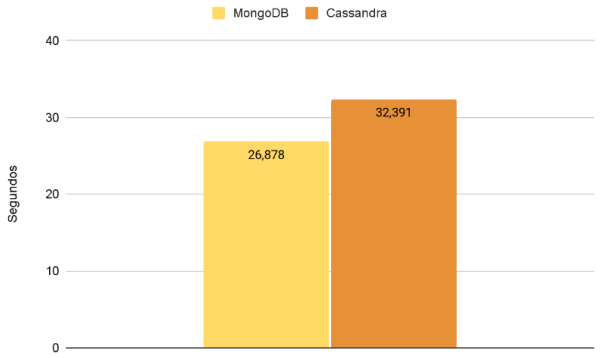
V. Conclusiones y trabajo futuro

Como resultado principal podemos concluir que MongoDB tuvo un mejor rendimiento que Cassandra para todos los casos de prueba llevados a cabo, superando notoriamente en el tiempo de ejecución de los escenarios. Por otra parte, se observa que a medida que va bajando el número de operaciones de escritura, también desciende el tiempo de ejecución, por lo tanto concluimos que esta operación es más costosa que la de lectura. Por último, siendo que el escenario C3 y C5 tienen la misma cantidad de operaciones de escritura y C5 incluye operaciones de borrado y un menor número de lecturas, se puede afirmar que la operación de borrar es más costosa que la de leer.

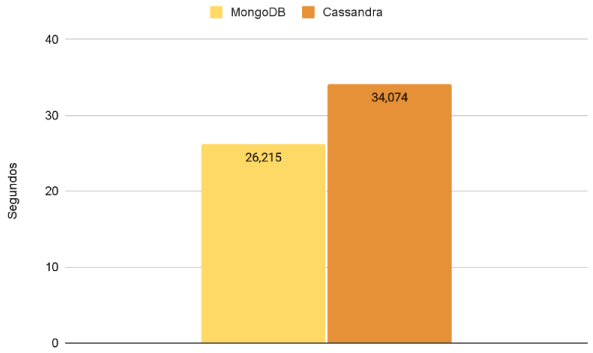
Con respecto a NoSQLBench, se concluye que es una herramienta muy útil para realizar evaluaciones de bases de datos no relacionales que permitan hacer una comparación entre distintas soluciones. Al contar con controladores genéricos, NoSQLBench permite evaluar a cualquier tipo de bases de datos no relacional. Para ello solo hace falta construir un cliente, elegir un driver que permita conectarse a él y ejecutar las cargas de trabajo.

Cuadro I: Porcentaje de operaciones por escenario

	C1	***C2***	***C3***	***C4***	***C5***
Escritura	100 %	75 %	50 %	25 %	50 %
Lectura	0 %	25 %	50 %	75 %	25 %
Borrado	0 %	0 %	0 %	0 %	25 %

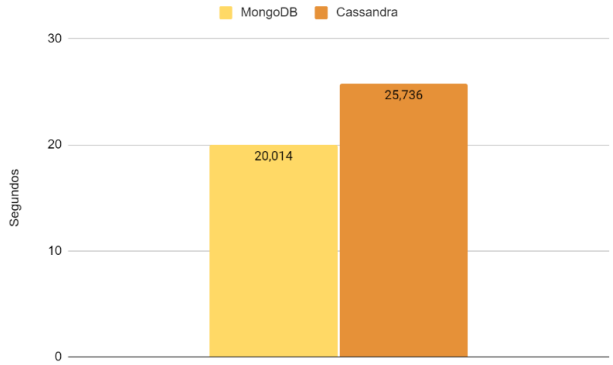


(a) Carga de trabajo C1

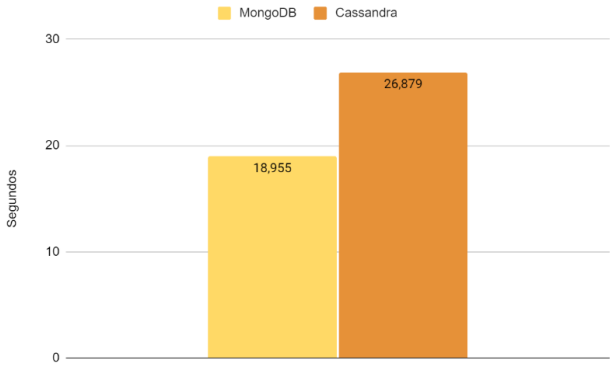


(b) Carga de trabajo C2

Figura 1



(a) Carga de trabajo C3



(b) Carga de trabajo C4

Figura 2

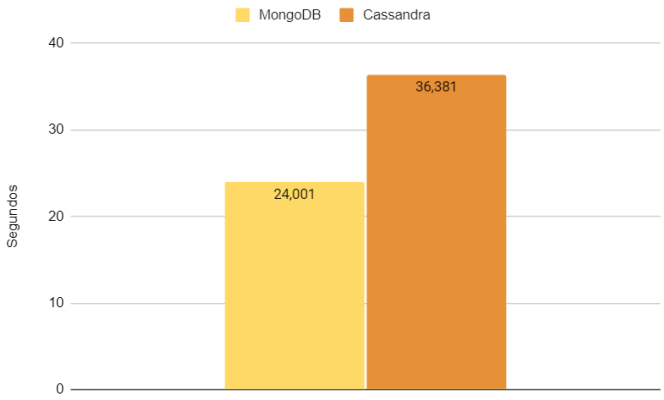


Figura 3: Carga de trabajo C5

Si bien lleva un cierto tiempo de aprendizaje al principio, la documentación[6] está muy completa y se mantiene actualizada constantemente. La herramienta tiene un conjunto activo de desarrolladores y *testers*, por lo tanto se puede intuir que la misma seguirá creciendo en cuanto a funcionalidades.

Como desventaja, al ser una herramienta relativamente nueva y que está en plena etapa de crecimiento, no se encontraron artículos académicos que la utilicen, como si los hay de Yahoo! Cloud Serving Benchmark (YCSB), la herramienta de facto para evaluar el rendimiento de bases de datos no relacionales.

Como trabajo a futuro, sería bueno incluir otras bases de datos a la comparación, como Redis y Couchbase. Además, si bien incluimos todas las operaciones CRUD en las pruebas realizadas, sería interesante crear escenarios donde las estructuras de

[6]https://docs.nosqlbench.io/

datos sean más complejas, por ejemplo agregar más colecciones en MongoDB o más columnas en Cassandra. Por último, sería deseable incluir al análisis otras métricas que recoge la herramienta para darle otro tipo de visión a la comparación, además del tiempo de ejecución. Alguna de ellas son el número total de operaciones y cuántas de ellas fueron exitosas, la cantidad de errores en operaciones y el número de intento y reintentos que se realizan por operación. Sería muy bueno visualizar estas estadísticas en las gráficas generadas por la herramienta en Grafana y Prometheus.