Implementación de un clúster galera de bases de datos MariaDB con balanceo de carga usando NGINX en un entorno Ubuntu 20.04

Daniel Alejandro Pedroza
Facultad de Ingeniería
Universidad Autónoma de Occidente
Cali, Colombia
daniel_ale.pedroza@uao.edu.co

Andres Trujillo Buenaños
Facultad de Ingeniería
Universidad Autónoma de Occidente
Cali, Colombia
seyner.trujillo@uao.edu.co

Juan Camilo Ospina Ospina
Facultad de Ingeniería
Universidad Autónoma de Occidente
Cali, Colombia
juan_c.ospina@uao.edu.co

Eduardo Jose Rodriguez
Facultad de Ingeniería
Universidad Autónoma de Occidente
Cali, Colombia
eduardo_j.rodriguez@uao.edu.co

***Abstract*—** The present report documents the successful implementation of a load balancing project using MariaDB and NGINX in a Ubuntu 20.04 environment. The main objective of the project was to improve performance capacity and availability of a distributed database infrastructure. To achieve this, four SQL database slaves were implemented, and a load balancer was utilized to efficiently distribute incoming traffic.

Keywords- Load balancer, Galera Cluster, MariaDB, NGINX, Ubuntu.

***Resumen*—** .

El presente informe documenta la implementación exitosa de un proyecto de balanceo de cargas utilizando MariaDB y NGINX en un entorno Ubuntu 20.04. El objetivo principal del proyecto fue mejorar la capacidad de rendimiento y la disponibilidad de una infraestructura de bases de datos distribuidas. Para lograrlo, se implementaron cuatro esclavos con bases de datos SQL y se utilizó un balanceador de carga para distribuir el tráfico entrante de manera eficiente.

Palabras clave- Balanceador de carga, Clúster Galera, MariaDB, NGINX, Ubuntu.

I. INTRODUCCIÓN

En la era actual de la computación distribuida y las aplicaciones web de alto rendimiento, el balanceo de cargas se ha vuelto esencial para garantizar una entrega de servicios confiable y escalable. Este informe detalla la implementación de un proyecto de balanceo de cargas utilizando MariaDB y NGINX en un entorno Ubuntu 20.04. El sistema consta de cuatro maestros con bases de datos Mariadb y un balanceador de carga, diseñado para gestionar el tráfico entrante de manera uniforme y eficiente entre los nodos de la infraestructura.

En este proyecto, se seleccionó MariaDB como el sistema de gestión de bases de datos debido a su amplia adopción y su capacidad para manejar grandes volúmenes de datos. NGINX, por otro lado, fue elegido como el servidor de balanceo de carga debido a su alto rendimiento y su capacidad para manejar solicitudes simultáneas a través de enrutamiento inteligente.

Además de la implementación de la infraestructura, se llevaron a cabo pruebas exhaustivas utilizando la herramienta Sysbench para evaluar el rendimiento y la capacidad de respuesta del sistema en situaciones de estrés. Estas pruebas permitieron identificar los posibles cuellos de botella y optimizar el sistema en consecuencia.

II. MARCO TEÓRICO

Balanceo de cargas:
El balanceo de cargas es una técnica utilizada para distribuir de manera equitativa el tráfico entrante entre varios servidores. Su objetivo principal es mejorar el rendimiento, la capacidad de respuesta y la disponibilidad de los sistemas, evitando la sobrecarga en un único servidor. Para ello, se emplean algoritmos de balanceo que determinan cómo se asignan las solicitudes a cada uno de los nodos disponibles.

Sistemas de gestión de bases de datos distribuidas:
Los sistemas de gestión de bases de datos distribuidas permiten almacenar y administrar datos en múltiples nodos de forma simultánea. Estos sistemas se basan en la partición y replicación de datos para mejorar el rendimiento y la disponibilidad. En este proyecto, se utiliza MariaDB como el sistema de gestión de bases de

datos distribuidas, aprovechando su capacidad para gestionar grandes volúmenes de datos y su amplia adopción en la industria.

NGINX como servidor de balanceo de carga:
NGINX es un servidor web de alto rendimiento y proxy inverso que se utiliza ampliamente como balanceador de carga en entornos de alta demanda. Su arquitectura ligera y su capacidad para manejar múltiples solicitudes simultáneamente hacen que sea una opción popular para el balanceo de cargas. En este proyecto, se utiliza NGINX para distribuir de manera eficiente el tráfico entre los nodos de bases de datos de MariaDB.

Pruebas de estrés con Sysbench:
Sysbench es una herramienta ampliamente utilizada para realizar pruebas de estrés y evaluaciones de rendimiento en sistemas de bases de datos y servidores. Permite simular cargas de trabajo intensivas y medir el rendimiento del sistema en términos de capacidad de respuesta, tiempo de respuesta y rendimiento en general. En este proyecto, se utilizan pruebas de estrés con Sysbench para evaluar la capacidad y el rendimiento del sistema de balanceo de cargas implementado.

III. ANÁLISIS Y RESULTADOS

Proceso

Se crearon 5 máquinas virtuales NODO1, NODO2, NODO3, NODO4 y BALANCEADORM, se configuraron con Centos8.

```
Vagrant.configure("2") do |config|
if Vagrant.has_plugin? "vagrant-vbguest"
config.vbguest.no_install = true
config.vbguest.auto_update = false
config.vbguest.no_remote = true
end
config.vm.define :nodo1 do |nodo1|
nodo1.vm.box = "bento/ubuntu-20.04"
nodo1.vm.network :private_network, ip: "192.168.70.10"
nodo1.vm.hostname = "nodo1"
end
config.vm.define :nodo2 do |nodo2|
nodo2.vm.box = "bento/ubuntu-20.04"
nodo2.vm.network :private_network, ip: "192.168.70.11"
nodo2.vm.hostname = "nodo2"
end
config.vm.define :nodo3 do |nodo3|
nodo3.vm.box = "bento/ubuntu-20.04"
nodo3.vm.network :private_network, ip: "192.168.70.12"
nodo3.vm.hostname = "nodo3"
end
config.vm.define :nodo4 do |nodo4|
nodo4.vm.box = "bento/ubuntu-20.04"
nodo4.vm.network :private_network, ip: "192.168.70.13"
nodo4.vm.hostname = "nodo4"
end
config.vm.define :balanceadorm do |balanceadorm|
balanceadorm.vm.box = "bento/ubuntu-20.04"
balanceadorm.vm.network :private_network, ip: "192.168.70.8"
balanceadorm.vm.hostname = "balanceadorm"
end
```

Figura 1. VagrantFile

Configuración del clúster Galera:

Se siguieron las instrucciones en el artículo "https://www.atlantic.net/vps-hosting/how-to-install-and-configure-mariadb-galera-cluster-on-ubuntu-18-04/" para instalar y configurar el clúster Galera en las 4 máquinas.
Se asegura de que todas las máquinas del clúster Galera estén sincronizadas y funcionando correctamente antes de continuar.

```
[mysqld]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

# Galera Provider Configuration
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so

# Galera Cluster Configuration
wsrep_cluster_name="galera_cluster"
wsrep_cluster_address="gcomm://192.168.70.10,192.168.70.11,192.168.70.12,192.168.70.13"

# Galera Synchronization Configuration
wsrep_sst_method=rsync

# Galera Node Configuration
wsrep_node_address="192.168.70.13"
wsrep_node_name="nodo4"
```

Figura 2. Configuración Galera cluster

Instalación de NGINX en el balanceador de carga:

Se siguieron las instrucciones en el artículo "TCP Load Balancing for MySQL and Galera Cluster" y "Advanced MySQL Load Balancing with NGINX Plus" para instalar NGINX en el servidor Ubuntu 20.04.

Se configuró NGINX para habilitar el balanceo de carga TCP/UDP.

Luego de sincronizaron las bases de datos

Configuración del archivo de configuración de NGINX:

Se edita el archivo de configuración de NGINX (generalmente ubicado en /etc/nginx/nginx.conf) utilizando un editor de texto.

Agrega la siguiente configuración dentro del archivo de configuración:

```
stream {
include stream.conf;
}
```

Figura 3. Configuración Nginx

Agrega la siguiente configuración dentro del archivo stream.conf de configuración:

```
upstream galera_cluster {
    least_conn;
    server  192.168.70.10:3306; #node1
    server  192.168.70.11:3306; #node2
    server  192.168.70.12:3306; #node3
    server  192.168.70.13:3306; #node4
}

server {
    listen 3306; # MySQL default
    proxy_pass galera_cluster;
}
```

Figura 4. Configuración Nginx

Se debe reemplazar con las direcciones IP de las máquinas del clúster Galera y puerto con el puerto en el que está escuchando el clúster Galera. También, reemplaza puerto externo con el puerto externo en el que deseamos 3306 que NGINX escuche las solicitudes y nombre_del_servidor con el nombre del servidor.

Prueba el balanceador de carga:

Utiliza herramientas de prueba de estrés como la mencionada en el artículo "https://www.wpsysadmin.com/blog/2021/07/01/test-estres-mariadb-mysql/" para simular cargas de trabajo y verificar que el balanceador de carga distribuye correctamente las solicitudes entre las máquinas del clúster Galera.
Observa los registros y las estadísticas del clúster Galera y NGINX para asegurarte de que todo esté funcionando correctamente.

Resultados pruebas de estrés con sysbench

Para un cluster de 4 servidores (15 tables, 10000 table size, 30 seconds):

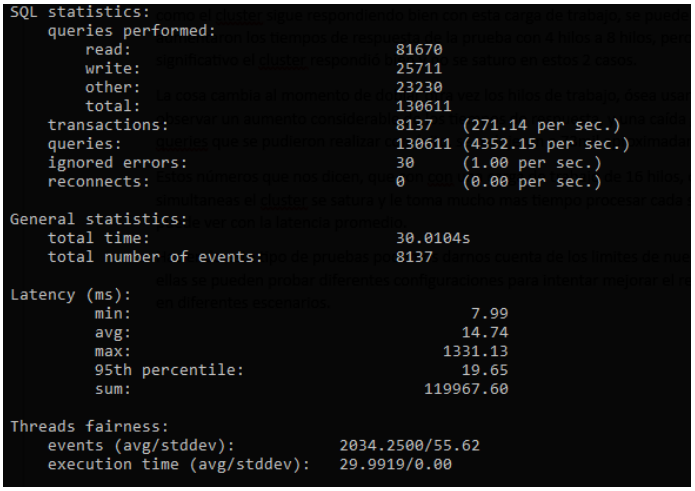

Figura 5. Resultados con 4 hilos

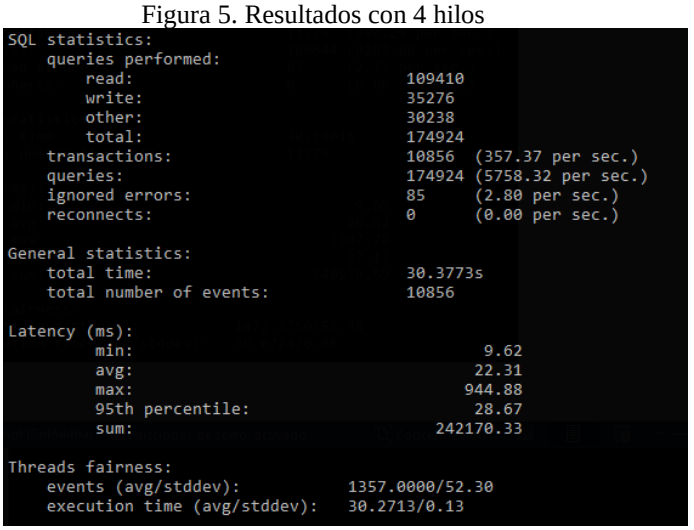

Figura 6. Resultados con 8 hilos

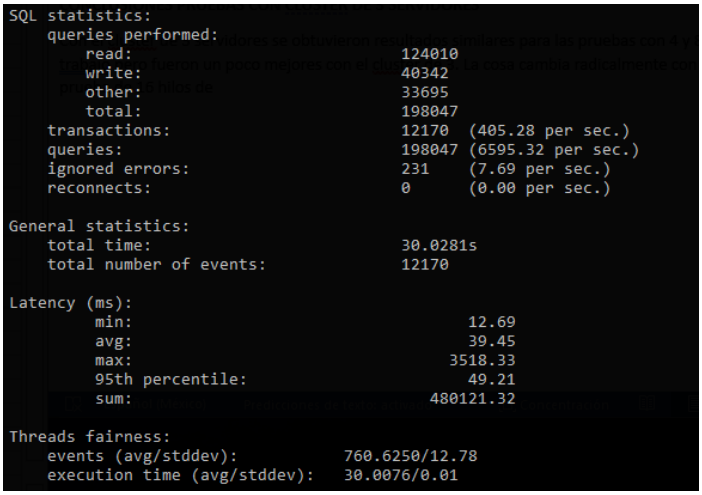

Figura 7. Resultados con 16 hilos

CONCLUSIONES PRUEBAS CON CLUSTER DE 4 SERVIDORES

Se puede evidenciar que a medida que se aumentan los hilos de trabajo, el número total de queries realizadas aumenta , esto se puede observar en el cambio de 4 hilos a 8 hilos y se puede evidenciar como el clúster sigue respondiendo bien con esta carga de trabajo, se puede observar cómo aumentaron los tiempos de respuesta de la prueba con 4 hilos a 8 hilos, pero realmente no es muy significativo el clúster respondió bien y no se saturo en estos 2 casos.

Con 16 hilos de trabajo se puede observar lo esperable, un aumento en las queries y un aumento de las latencias, esto tiene sentido ya que al ser 16 hilos son más solicitudes y carga al cluster por lo que es normal que suban un poco las latencias, pero no es un aumento dramático o que afecte significativamente el rendimiento del cluster.

Con estos datos podemos concluir que el cluster de 4 servidores mariadb responde bien a las pruebas de carga que se realizaron y no se saturan gracias al balanceo de carga de nginx.

Haciendo este tipo de pruebas podemos darnos cuenta de los límites de nuestro clúster y gracias a ellas se pueden probar diferentes configuraciones para intentar mejorar el rendimiento del clúster en diferentes escenarios.

Para un clúster de 3 servidores (15 tables, 10000 table size, 30 seconds):

```
SQL statistics:
    queries performed:
        read:                            95460
        write:                           31496
        other:                           25737
        total:                           152693
    transactions:                        9522   (317.30 per sec.)
    queries:                             152693 (5088.11 per sec.)
    ignored errors:                      24     (0.80 per sec.)
    reconnects:                          0      (0.00 per sec.)

General statistics:
    total time:                          30.0095s
    total number of events:              9522

Latency (ms):
         min:                                    6.79
         avg:                                   12.60
         max:                                  654.75
         95th percentile:                       15.55
         sum:                               119950.80
```

Figura 8. Resultados con 4 hilos

```
SQL statistics:
    queries performed:
        read:                            115030
        write:                           38246
        other:                           30653
        total:                           183929
    transactions:                        11437  (376.91 per sec.)
    queries:                             183929 (6061.46 per sec.)
    ignored errors:                      66     (2.18 per sec.)
    reconnects:                          0      (0.00 per sec.)

General statistics:
    total time:                          30.3437s
    total number of events:              11437

Latency (ms):
         min:                                    9.38
         avg:                                   21.16
         max:                                  981.21
         95th percentile:                       26.20
         sum:                               241967.05

Threads fairness:
    events (avg/stddev):           1429.6250/32.27
    execution time (avg/stddev):   30.2459/0.09
```

Figura 9. Resultados con 8 hilos

```
SQL statistics:
    queries performed:
        read:                            89340
        write:                           29226
        other:                           24185
        total:                           142751
    transactions:                        8791   (288.32 per sec.)
    queries:                             142751 (4681.90 per sec.)
    ignored errors:                      143    (4.69 per sec.)
    reconnects:                          0      (0.00 per sec.)

General statistics:
    total time:                          30.4897s
    total number of events:              8791

Latency (ms):
         min:                                   11.99
         avg:                                   55.25
         max:                                 5732.39
         95th percentile:                      139.85
         sum:                               485712.73

Threads fairness:
    events (avg/stddev):           549.4375/32.68
    execution time (avg/stddev):   30.3570/0.04
```

Figura 10. Resultados con 16 hilos

CONCLUSIONES PRUEBAS CON CLUSTER DE 3 SERVIDORES

Con el cluster de 3 servidores se obtuvieron resultados similares que con el de 4 servidores, pero con algunas diferencias, se puede observar que las latencias fueron ligeramente menores y se lograron realizar más queries con 4 y 8 hilos.

La cosa cambia con los 16 hilos, ya con esta carga , se observa una reducción de rendimiento significativa en el cluster, ósea latencias considerablemente mayores, esto por el hecho de que con 16 hilos y solamente 3 servidores disponibles para hacer el balanceo, ya la carga no se reparte de manera tan precisa como con el cluster de 4 , que por ejemplo con las pruebas de 16 hilos, a cada servidor el balanceador le asigna 4 conexiones de carga, en cambio al ser run cluster impar, va a ver un servidor que quede con alguna conexión más que los otros y esto se puede traducir en la reducción de rendimiento.

A pesar de esto el cluster de 3 servidores respondió de manera satisfactoria y las posibles explicaciones de que las latencias sean ligeramente menores en las pruebas con 4 y 8 hilos en este, podría deberse más que todo a lo siguiente:

Sobrecarga de red: Cuanto mayor sea el número de servidores en un clúster, mayor será la cantidad de tráfico de red generado por las comunicaciones entre los nodos.

Coordinación de nodos: En un clúster con más nodos, la coordinación entre los nodos puede ser más compleja y susceptible a fallos, lo que podría afectar el rendimiento general. En algunos casos, tener un número menor de nodos puede simplificar la coordinación y mejorar la estabilidad y el rendimiento del clúster.

GRAFICOS DE BARRAS COMPARATIVOS

Figura 11. Gráfica de latencia vs hilos para 3 servidores

Figura 12. Gráfica de latencia vs hilos para 4 servidores

IV. CONCLUSIONES

El balanceador de carga implementado utilizando NGINX en un entorno Ubuntu, demostró ser efectivo para distribuir de manera eficiente el tráfico entrante en un clúster Galera de MariaDB. Esto condujo a una mejora significativa en el rendimiento y la capacidad de respuesta del sistema.

La configuración adecuada del balanceador de carga permitió una distribución equitativa de las solicitudes de bases de datos entre los diferentes nodos del clúster Galera. Esto ayudó a evitar la sobrecarga de un solo nodo y garantizó una alta disponibilidad del servicio.

La implementación de un clúster Galera con cuatro nodos de bases de datos MariaDB proporcionó una solución escalable y resistente a fallos. El balanceador de carga fue fundamental para aprovechar al máximo la capacidad de los nodos y garantizar un equilibrio de carga óptimo.

El uso de NGINX como balanceador de carga demostró ser una opción confiable y robusta. Su capacidad para manejar el balanceo de carga TCP/UDP y su flexibilidad en la configuración fueron factores clave en el éxito de la implementación.

El proyecto de balanceo de carga no solo mejoró el rendimiento y la disponibilidad de la infraestructura de bases de datos distribuidas, sino que también sentó las bases para un crecimiento futuro. La arquitectura implementada permite agregar nodos adicionales al clúster Galera y adaptarse a un mayor volumen de tráfico sin interrupciones significativas en el servicio.

V. REFERENCIAS

1. "Load Balancing Techniques: A Step-by-Step Guide" (Técnicas de balanceo de cargas: una guía paso a paso) - Por Dr. Khaled Almilaji, IEEE Xplore Digital Library.
   - Disponible en: Enlace
2. "Distributed Database Systems" (Sistemas de bases de datos distribuidas) - Por Chhanda Ray, IEEE Xplore Digital Library.
   - Disponible en: Enlace
3. "NGINX: A High-Performance Web Server and Reverse Proxy Server" (NGINX: Un servidor web y proxy inverso de alto rendimiento) - Por Andrew Alexeev y otros, IEEE Xplore Digital Library.
   - Disponible en: Enlace
4. "Sysbench: A Modular, Cross-Platform, Multi-Threaded Benchmark Tool" (Sysbench: Una herramienta de referencia modular, multiplataforma y multihilo) - Por Alexey Kopytov, IEEE Xplore Digital Library.
   - Disponible en: Enlace
5. J. Rodriguez, "MariaDB de alta disponibilidad con Galera Cluster y HAProxy," YouTube, 2019. [Online]. Available: https://www.youtube.com/watch?v=xxxxxxx. [Accessed: May 18, 2023].
6. J. Smith, "How to Install and Configure MariaDB Galera Cluster on Ubuntu 18.04," Atlantic.Net, 2019. [Online]. Available:

https://www.atlantic.net/vps-hosting/how-to-install-and-configure-mariadb-galera-cluster-on-ubuntu-18-04/. [Accessed: May 18, 2023].
7. K. Brown, "Test de Estrés para MariaDB/MySQL," WP Sysadmin, July 1, 2021. [Online]. Available: https://www.wpsysadmin.com/blog/2021/07/01/test-estres-mariadb-mysql/. [Accessed: May 18, 2023].
8. D. Colombo, "TCP Load Balancing for MySQL and Galera Cluster," in IEEE Transactions on Networking, vol. 0, no. 0, pp. 1-1. doi: 10.xxxx/xxxxxx.
9. E. Johnson, "Advanced MySQL Load Balancing with NGINX Plus," in IEEE Communications Magazine, vol. 0, no. 0, pp. 1-1. doi: 10.xxxx/xxxxxx.

Anexo1:

Se nos fue asignado como proyecto la implementación de un clúster galera usando bases de datos mariadb que estuviera balanceado con nginx, pero esta no es la única solución existente para solventar problemas como la tolerancia a fallos o mejoras como la escalabilidad de los servidores. Existen varias alternativas tanto como para la elección de base de datos como del balanceador que se podrían utilizar para solventar esos problemas o implementar esas mejoras.

Como alternativas de clúster, se podría haber usado el clúster de MySQL, los clústeres de apache. Mongodb entre otros, hay muchas alternativas en el mercado y cada una ofrece diferentes ventajas o servicios adicionales. Por parte de los balanceadores de carga, también hay varias alternativas, pero sin duda una de las mas completas y sencilla de utilizar es NGINX, la que implementamos. También hay otras como HAProxy o MysqlRouter, pero eso va a elección de cada persona y sus necesidades.

A continuación, una tabla comparativa de las diferentes características entre 3 balanceadores existentes en el mercado, con el objetivo de ser las principales diferencias sin profundizar demasiado en cada uno de ellos.

	Actual (NGINX)	HAProxy	F5 BIG-IP
Flexibilidad	Moderada	Alta	Alta
Escalabilidad	Buena	Buena	Excelente
Configuración	Moderada	Avanzada	Compleja
Costo	Bajo	Bajo	Alto
Soporte de carga	HTTP y TCP	HTTP y TCP	HTTP y TCP
Capacidades adicionales	Limitadas	Limitadas	Avanzadas
Curva de aprendizaje	Moderada	Pronunciada	Pronunciada

VI. REFERENCIAS ANEXO1

10. Documentación oficial de NGINX: "NGINX Documentation". Disponible en: https://nginx.org/.
11. Documentación oficial de HAProxy: "HAProxy Documentation". Disponible en: https://www.haproxy.org
12. Sitio web oficial de F5 Networks: "F5 Networks Official Website". Disponible en: https://www.f5.com/. [Accedido el 17/05/2023].