**Implementación De Un Sistema De Balanceo De Carga Utilizando MySQL Y MySQL Router Para Mejorar El Rendimiento De Aplicaciones Distribuidas**

David Stivens Ordoñez Ojeda, 2180756

david\_sti.ordonez@uao.edu.co

Javier Andrés Vásquez, 2205042

javier.vasquez@uao.edu.co

Santiago Castaño Henao, 2195304

santiago.castano\_hen@uao.edu.co

Juan David Duque Echeverry, 2200388

juan\_david.duque@uao.edu.co

**Abstract**

In this project, the implementation of a load balancing system was carried out using MySQL and MySQL Router as key components. The approach involved the use of a total of five machines to establish the necessary infrastructure: a Docker machine for container management, a MySQL Router machine responsible for directing incoming requests, and three machines configured for Group Replication, consisting of one master machine and two secondary machines. Through extensive testing conducted with Sysbench on the master machine, the system's performance was evaluated under various workload scenarios. The main objective was to achieve an equitable distribution of the load among the database nodes, ensuring optimal performance and high availability for end users. The project stood out for its focus on scalability, fault tolerance, and performance optimization, thus ensuring a smooth and efficient experience for users of the database system.

**Resumen**

En este proyecto, se llevó a cabo la implementación de un sistema de balanceo de carga utilizando MySQL y MySQL Router como componentes clave. El enfoque consistió en utilizar un total de cinco máquinas para establecer la infraestructura necesaria: una máquina Docker para la gestión de los contenedores, una máquina MySQL Router encargada de direccionar las solicitudes entrantes y tres máquinas configuradas para el Group Replication, con una máquina maestra y dos máquinas secundarias. Mediante pruebas exhaustivas realizadas con Sysbench en la máquina maestra, se evaluó el rendimiento del sistema en diversos escenarios de carga de trabajo. El principal objetivo era lograr una distribución equitativa de la carga entre los nodos de base de datos, garantizando un rendimiento óptimo y una alta disponibilidad para los usuarios finales. El proyecto se destacó por su enfoque en la escalabilidad, la tolerancia a fallos y la optimización del rendimiento, asegurando así una experiencia fluida y eficiente para los usuarios del sistema de base de datos.

**Introducción**

Las aplicaciones web que ofrecen servicios importantes y de gran utilidad a menudo se enfrentan a una situación particular y es la de recibir muchas peticiones de diferentes usuarios al mismo tiempo. Esta situación si no es prevista con tiempo, al momento de la producción la aplicación no podrá responder y podría generar perdidas bastante altas. Por eso es importante conocer el balanceo de cargas en un servidor implementado en MySQL, aprendiendo a utilizar la herramienta MySQL Cluster que hace parte de la herramienta principal llamada InnoDB Cluster.

**Contexto**

El balanceo de carga es un mecanismo esencial para mejorar el rendimiento y la capacidad de los servicios web en entornos de alta demanda y tráfico intenso. En la era digital actual, donde las empresas dependen cada vez más de sus aplicaciones y servicios en línea, mantener un rendimiento óptimo y una disponibilidad constante es fundamental para satisfacer las expectativas de los usuarios y garantizar la satisfacción del cliente.

Este proceso de balanceo de carga se lleva a cabo mediante el uso de un balanceador, una pieza de software o hardware especializado que actúa como intermediario entre los clientes y los servidores que alojan los servicios web. Su principal función es distribuir la carga de trabajo entrante de manera equitativa y eficiente entre varios servidores, evitando así que un solo dispositivo se sature o se convierta en un cuello de botella.

El objetivo principal del balanceo de carga es aumentar la capacidad y escalabilidad de un servicio web, permitiendo que más usuarios accedan al servicio al mismo tiempo y reduciendo los tiempos de respuesta [1].

**Propuestas de solución**

**Galera Cluster:**

En Galera Cluster, el balanceo de carga a nivel de base de datos se refiere a la capacidad de distribuir automáticamente las solicitudes de lectura entre los nodos del clúster sin la necesidad de un balanceador de carga externo. Esto se logra utilizando un mecanismo llamado "read load balancing" o balanceo de carga de lectura.

Cuando se habilita el balanceo de carga de lectura en Galera Cluster, los nodos del clúster tienen un componente llamado "arbitrator" o árbitro que se encarga de tomar decisiones sobre cómo distribuir las solicitudes de lectura entre los nodos. El árbitro utiliza un algoritmo interno para determinar qué nodo debe manejar cada solicitud de lectura. [5].

**Percona XtraDB Cluster:**

Percona XtraDB Cluster es una solución de clustering de bases de datos basada en el motor de almacenamiento Percona XtraDB de Percona Server para MySQL. Ofrece replicación síncrona multi-master y permite el balanceo de carga mediante la distribución de las consultas entre los nodos del clúster. Percona XtraDB Cluster proporciona alta disponibilidad y rendimiento escalable. [6].

**MaxScale:**

MaxScale es una plataforma de enrutamiento y balanceo de carga desarrollada por MariaDB Corporation. Ofrece una amplia gama de funciones para el balanceo de carga, como enrutamiento basado en reglas, caché de consultas, filtrado de consultas, y gestión de alta disponibilidad y conmutación por error. MaxScale es compatible tanto con MariaDB como con MySQL. [7].

**Descripción del problema**

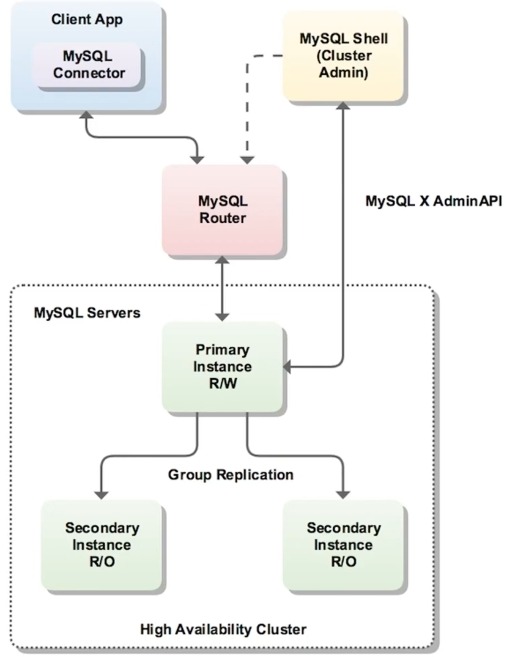
A medida que una empresa o plataforma experimenta un crecimiento notable en la cantidad de usuarios y transacciones, su infraestructura de bases de datos existente basada en un solo servidor se enfrenta a desafíos significativos. La sobrecarga resultante de la creciente carga de trabajo provoca tiempos de respuesta lentos y una experiencia insatisfactoria para los usuarios. Los recursos limitados del servidor no son suficientes para manejar eficientemente el aumento en la demanda, lo que afecta negativamente la capacidad de la empresa para brindar un servicio rápido y confiable. Además, la falta de redundancia y tolerancia a fallos puede generar interrupciones y pérdida de datos en caso de una falla del servidor único. Es evidente que se requiere una solución que distribuya la carga de manera equitativa, mejore el rendimiento y la disponibilidad, y permita una escalabilidad adecuada para satisfacer las necesidades en constante evolución de la empresa.

**Solución Implementada**

Una de las posibles soluciones era el “Group Replication”, una característica que proporciona el mismo MySQL con el nombre de MySQL InnoDB Cluster (versiones 5.7.17. en adelante). Este consta de componentes esenciales para su funcionamiento [2]:

1. MySQL Shell: Es el componente que permite la administración del clúster
2. MySQL Router: Es el encargado de enrutar las peticiones y de distribuirlas a lo largo del clúster de datos.
3. MySQL Servers: Son las maquinas encargadas del almacenamiento de datos. Tienen la característica de que hay un servidor maestro (Tiene el poder de lectura/escritura) y el resto son instancias secundarias de respaldo (solo tienen el poder de lectura).

El esquema de esta solución se puede presentar de la siguiente manera (figura sacada de la documentación oficial de MySQL):



**Pruebas**

Para la realización de las pruebas del proyecto, se utilizaron los siguientes softwares: Vagrant con CentOS 8 Generic el cual tenía 5 máquinas virtuales para realizar los respectivos testeos. La primera máquina es la que contiene el MySQL Router el cual se le instalo MySQL y MySQL Router, una vez la máquina virtual ya tiene exitosamente estos servicios se le establece un Cluster denominado FUTURESOFT\_CLS, después se revisan los protocolos clásicos de MySQL los cuales tienen establecidos puertos específicos para la escritura y lectura: 6446 por otro lado, se tiene el de solo lectura por el puerto 6447. Se revisa de forma minuciosa que el puerto no este interfiriendo con otro servicio porque podría traer futuros problemas como colisiones de datos.

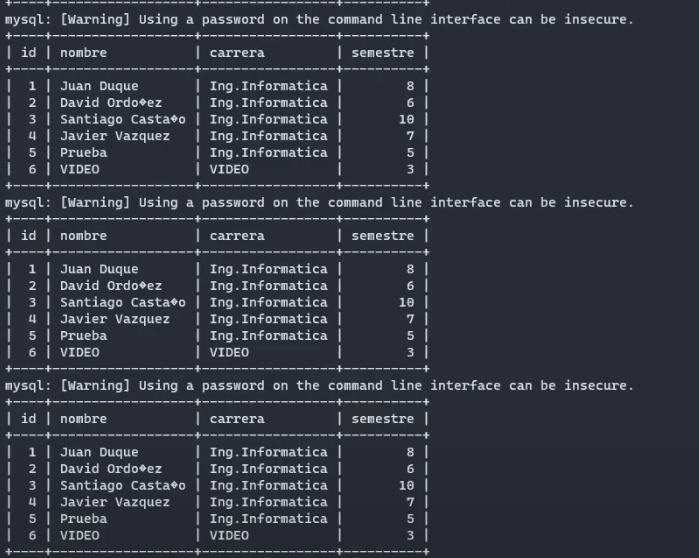
Se cuenta con maquina con el nombre “MySQL1” la cual es la “master” y es la que te tiene todos los privilegios (escritura y lectura). Además, se tiene “MySQL2” que en este caso sería una maquina esclava con solo lectura y “MySQL3” cuenta con el mismo privilegio que la mencionada anteriormente. Por último, la maquina número 5 esta implementada con Docker el cual es un software que permite la creación, implementación de aplicaciones en entornos aislados llamados contenedores. Se crean 3 contenedores para los cuales cada uno de ellos tendrá asignado una máquina, la "master" y las dos esclavas.

Una vez teniendo todas las maquinas corriendo simultáneamente se procede a crear una base de datos de prueba llamada UAO en la maquina “master” donde cuenta con una tabla estudiantes el cual se tiene 5 inserciones almacenadas anteriormente, esto con el fin de verificar que la base de datos este correctamente creada y la máquina cuenta con los privilegios respectivos.

Después, se ingresa a la maquina esclava “MySQL2” con el fin de comprobar que únicamente debería consultar al digitar una sentencia SQL de select, al introducirla efectivamente nos muestra la información la cual inserto la maquina “master” (las 5 inserciones) pero a la hora de intentar insertar un registro no lo permite puesto a que no tiene permisos apropiados.

La primera prueba se hace en la maquina donde está el Docker, para observar la tabla estudiantes de la base de datos “UAO”, para observarla se digita un ciclo while, siempre se mostrará la tabla hasta que se interrumpan las 3 máquinas al mismo tiempo(MySQL1, MySQL2 y MySQL3), en caso tal de que la maquina “master” se caiga por diferentes motivos, una de las dos máquinas quedaría como la master (primaria), esto con el fin de descentralizar el proceso, para que no suceda que si se cae la “master” se caigan las demás máquinas esclavas. Siempre va a estar mostrando la consulta a pesar de que ocurra un error repentino con la “master". Este es el comando el cual se utilizó: *while [ 1 ] do sleep 1 docker exec -it mysql1 mysql -h 192.168.50.2 -P 6446 -uinnodbcluster -pcontraseña\_fuerte1! -e "SELECT \* FROM UAO.estudiantes;" done*

Aquí las tablas:



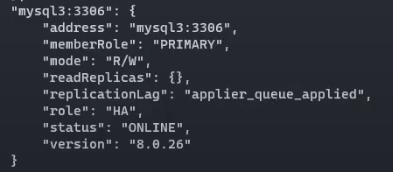
Se procederá a detener la maquina “master” para revisar que ocurre, puesto a que se está intentado simular como si fuera un caso en la realidad de una empresa, ¿qué pasaría si se detiene por completo?, para esto simplemente detenemos el servicio dentro de la máquina de “master”-> MySQL1:

*sudo systemctl stop mysqld*

Una vez la maquina fue apagada, comprobamos el estado del Cluster donde nos permite visualizar toda la información de las maquinas que están siendo ejecutadas en el momento.



Al verificar el estado se puede visualizar que no se puede establecer una conexión con la maquina” master” entonces lo que hace el Cluster es asignar a cualquiera de las dos máquinas esclavas, en este caso le dio privilegio de read and write a la máquina MySQL3. Al realizar esta prueba se puede deducir que se cuenta con una alta disponibilidad porque al interrumpir la máquina” master” continúan los servicios y las esclavas sin perder información.

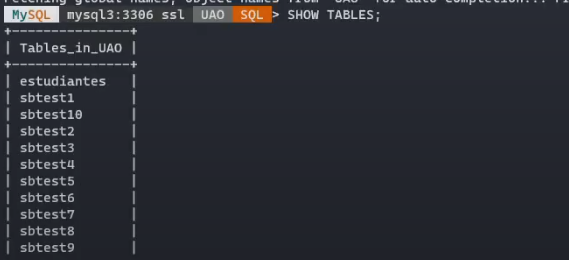


Como se puede observar ahora la “MySQL3” es la “master”, en caso tal de que se detenga la máquina actual primaria pasaría la esclava “MySQL2” a tener privilegios de read and write y se convierte en la “master”.

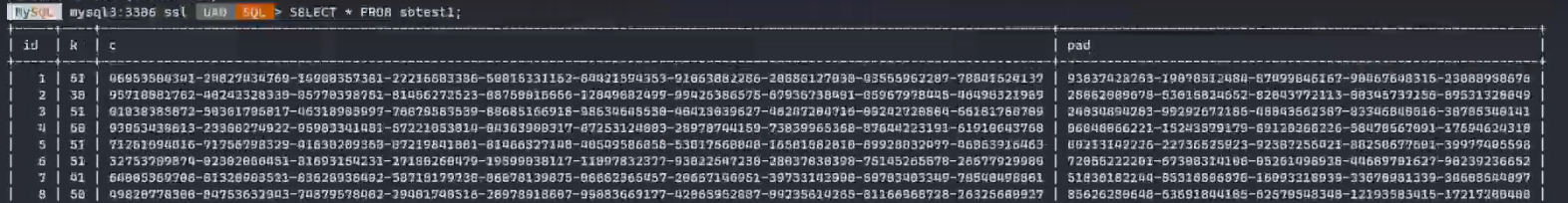
Una vez hecho las pruebas de disponibilidad de las maquinas, se realizaron las respectivas pruebas de estrés para percatarse de un posible error por n número de inserciones al mismo tiempo. Se utilizó sysbench con 10 hilos esto se refiere en palabras cotidianas que 10 personas están ejecutando sentencias sql de inserción al mismo tiempo.

*sysbench oltp\_write\_only --tables=10 --table-size=100 --db-driver=mysql --mysql-host=localhost --mysql-db=UAO --mysql-user=root --mysql-port=6446 --time=120 --threads=10 prepare.*

Al realizar el Select de las tablas, nos generó 10 tablas:



el cual cuenta con 100 registros cada una:



La siguiente prueba será de 100 tablas y cada tabla que contendrá 1000 registros para comprobar si realmente cuenta con el rendimiento oportuno:

*sysbench oltp\_write\_only --tables=100 --table-size=1000 --db-driver=mysql --mysql-host=localhost --mysql-db=UAO --mysql-user=root --mysql-port=6446 --time=120 --threads=64 prepare.*

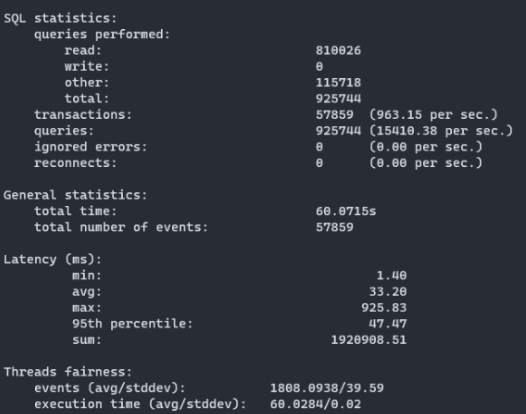
Cuando se realizó esta prueba tardo un poco más que el anterior puesto a que se estaba registrando más información. Y a la hora de imprimir los 1000 registros se puede visualizar que cada tabla de los hilos contiene los registros insertados.

También se realizaron pruebas más exhaustivas donde se crearon 200 tablas y cada una con 10000 registros, se puede percatar que el tiempo de las inserciones aumentaron, se puede decir que se tiene una alta disponibilidad porque a pesar de estas pruebas de estrés las maquinas no sufrieron ningún tipo de error ni se detuvieron.

Además se realizaron pruebas de **lectura intensiva**, los comandos para realizar esta prueba y comprobar el rendimiento (únicamente es lectura):

*sysbench oltp\_read\_only --tables=50 --table-size=5000 --db-driver=mysql --mysql-host=localhost --mysql-db=UAO --mysql-user=root --mysql-port=6446 --time=60 --threads=32 prepare*

*sysbench oltp\_read\_only --tables=50 --table-size=5000 --db-driver=mysql --mysql-host=localhost --mysql-db=UAO --mysql-user=root --mysql-port=6446 --time=60 --threads=32 run*

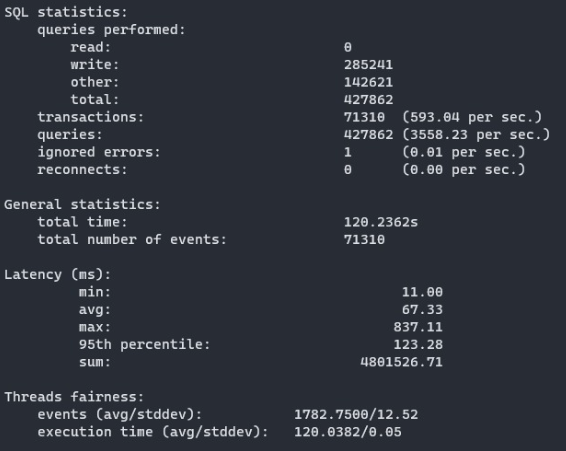


Se puede observar que se realizaron 810026 consultas.

Las pruebas de **escritura intensiva** se llevaron a cabo con los siguientes comandos, donde crean 100 tablas y cada una cuenta con 2000 registros (únicamente escritura):

*sysbench oltp\_write\_only --tables=100 --table-size=2000 --db-driver=mysql --mysql-host=localhost --mysql-db=UAO --mysql-user=root --mysql-port=6446 --time=120 --threads=64 prepare*

*sysbench oltp\_write\_only --tables=100 --table-size=2000 --db-driver=mysql --mysql-host=localhost --mysql-db=UAO --mysql-user=root --mysql-port=6446 --time=120 --threads=40 run*

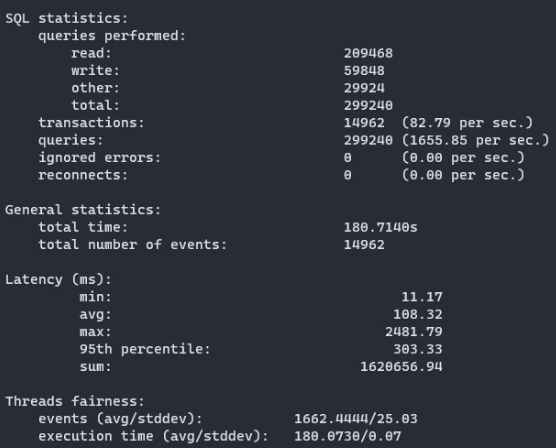


Se observa que se insertaron 285241 datos. A la hora de realizar este tipo de pruebas solo dejaba hasta un máximo de 40 hilos, si se sobrepasa este límite genera un error.

Luego, el equipo de trabajo decidido realizar **pruebas de carga mixta** es decir lectura y escritura, todo esto simultáneamente:

*sysbench oltp\_read\_write --tables=200 --table-size=10000 --db-driver=mysql --mysql-host=localhost --mysql-db=UAO --mysql-user=root --mysql-port=6446 --time=180 --threads=128 prepare*

*sysbench oltp\_read\_write --tables=200 --table-size=10000 --db-driver=mysql --mysql-host=localhost --mysql-db=UAO --mysql-user=root --mysql-port=6446 --time=180 --threads=9 run*

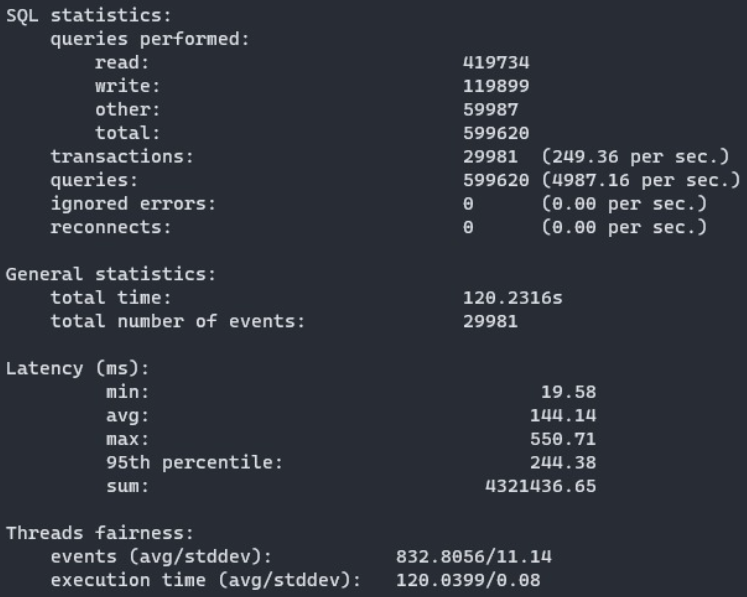


Se observa que 209468 fueron consultas y 59848 fueron inserciones. Deja ejecutar hasta 9 hilos.

La siguiente prueba es la de **concurrencia**

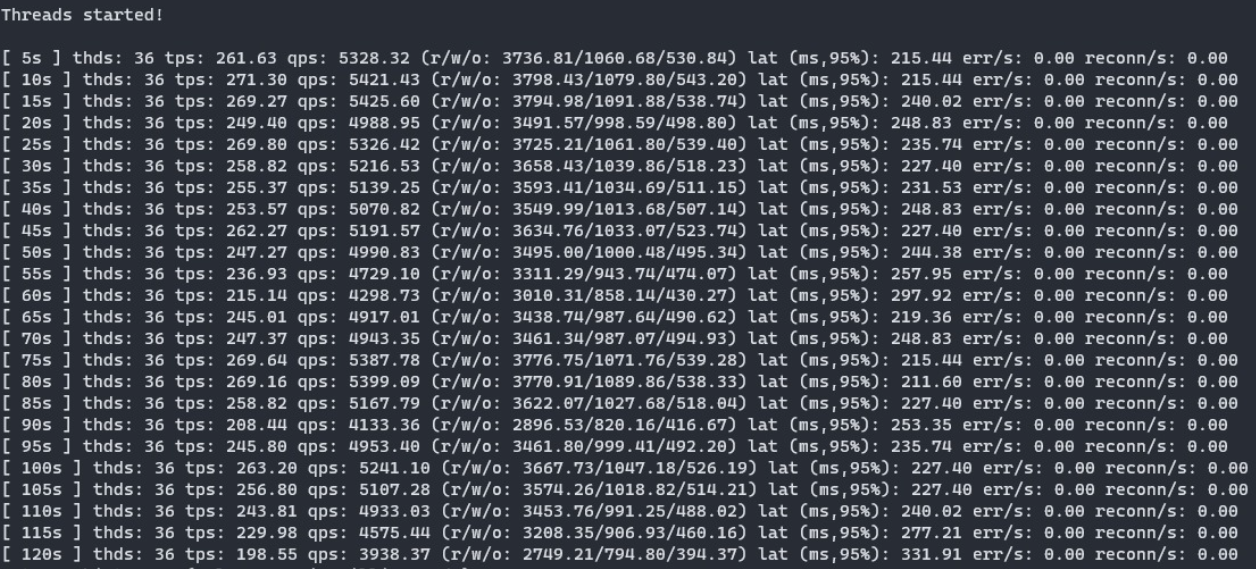
*sysbench oltp\_read\_write --tables=50 --table-size=5000 --db-driver=mysql --mysql-host=localhost --mysql-db=UAO --mysql-user=root --mysql-port=6446 --time=120 --threads=64 --report-interval=5 --histogram prepare*

*sysbench oltp\_read\_write --tables=50 --table-size=5000 --db-driver=mysql --mysql-host=localhost --mysql-db=UAO --mysql-user=root --mysql-port=6446 --time=120 --threads=36 --report-interval=5 --histogram run*

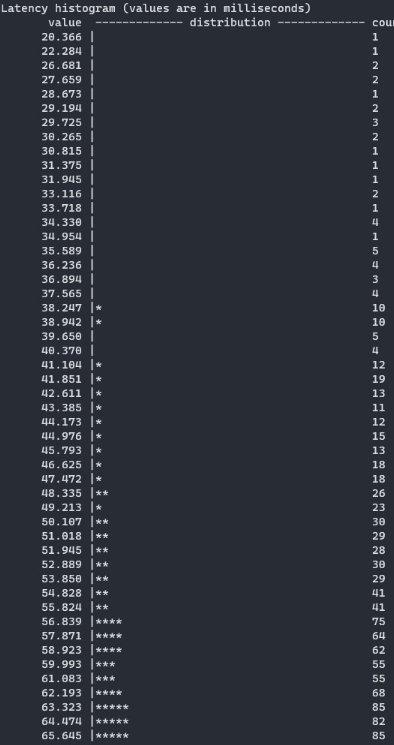
**

Al ejecutar los comandos se tiene un máximo de 36 hilos, una vez se sobrepase ese valor generara un error. Como se puede observar en el comando cuenta con dos parámetros nuevos los cuales es el reporte de intervalo e histograma.

**Reporte de intervalo:**

****

**Histograma:**

****

**Conclusión**

En conclusión, implementar la solución de balanceo de carga utilizando MySQL y MySQL Router ofrece varios beneficios significativos. Esta implementación permite distribuir la carga de trabajo de manera equitativa entre múltiples servidores de bases de datos, lo que mejora el rendimiento general del sistema al evitar la sobrecarga en un solo servidor. Además, el uso de MySQL Router facilita la gestión y dirección de las solicitudes entrantes, optimizando el enrutamiento y garantizando una distribución eficiente de la carga.

Al implementar esta solución, se logra una mayor disponibilidad y tolerancia a fallos gracias a la configuración de replicación y la redundancia de los datos. En caso de un fallo en uno de los servidores, el sistema continúa funcionando sin interrupciones, lo que garantiza una alta disponibilidad y una experiencia ininterrumpida para los usuarios.

Además, al distribuir la carga entre múltiples servidores, se mejora la escalabilidad del sistema, ya que se puede agregar capacidad adicional según sea necesario para manejar el crecimiento futuro de la empresa. Esto proporciona flexibilidad y la capacidad de adaptarse a las demandas cambiantes del negocio.

Este proyecto de servicios telemáticos nos ha brindado una valiosa experiencia práctica en la implementación de soluciones de balanceo de carga utilizando MySQL y MySQL Router. Hemos fortalecido nuestras habilidades en comunicación, coordinación y gestión de proyectos, mientras adquirimos conocimientos técnicos sobre configuración de servidores de base de datos y distribución equitativa de solicitudes. Estamos satisfechos con los logros alcanzados y confiamos en que estas habilidades nos serán útiles en nuestra futura carrera como ingenieros informáticos.

**Referencias**

[1] Byte. "Balanceo de carga". Redes Teleco. [En línea]. Disponible en: <https://redesteleco.com/balanceo_de_Carga> Accedido el 13 de mayo de 2023.

[2] MySQL. "InnoDB Cluster". MySQL. [En línea]. Disponible en: <https://dev.mysql.com/doc/refman/8.0/en/mysql-innodb-cluster-introduction.html> Accedido el 13 de mayo de 2023.

[3] MySQL. "MySQL router 8.0". MySQL. [En línea]. Disponible en: <https://dev.mysql.com/doc/mysql-router/8.0/en/> Accedido el 13 de mayo de 2023.

[4] Tilsor S.A. "Tilsor S.A.". [En línea]. Disponible en: <https://www.tilsor.com.uy/index.php/blog/72-mysql> Accedido el 13 de mayo de 2023.

[5] Cluster de balanceo de carga. (s.f.). [En línea]. Disponible en: <https://enciclopedia_universal.es-academic.com/43711/Cluster_de_balanceo_de_carga> Accedido el 17 de mayo de 2023.

[6] Percona. (2019, 12 de septiembre). How to Build Percona XtraDB Cluster from Sources. Percona. [En línea]. Disponible en: <https://www.percona.com/blog/how-to-build-percona-xtradb-cluster-from-sources/> Accedido el 17 de mayo de 2023.

[7] MariaDB MaxScale - MariaDB Knowledge Base. (n.d.). [En línea]. Disponible en: <https://mariadb.com/kb/en/maxscale/> Accedido el 17 de mayo de 2023.