Proyecto Final

de Servicios Telemáticos

Ludy Agudelo - Edwin Guerrero -
Sergio Herrera - Sebastián Marinez

Introducción

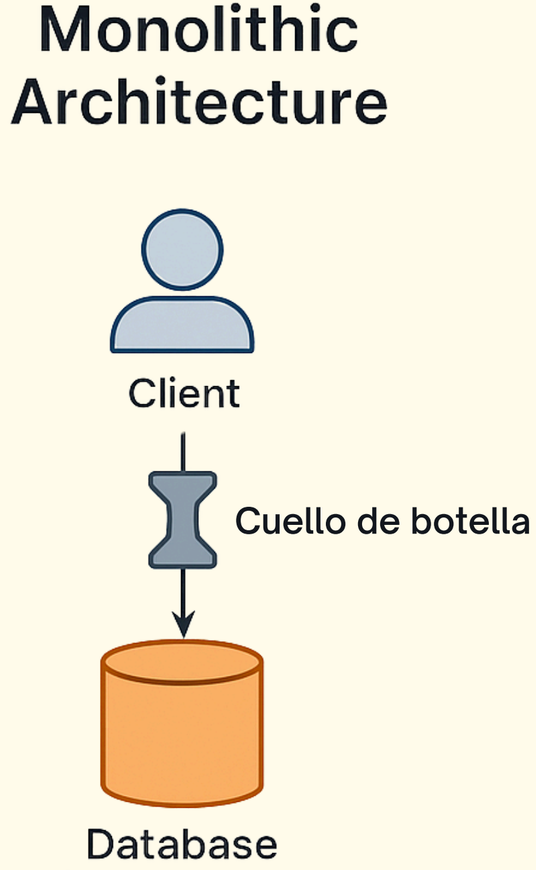
- Las arquitecturas monolíticas de bases de datos enfrentan serias limitaciones ante el crecimiento de aplicaciones web y la alta concurrencia de usuarios.

- Una sola instancia de MySQL que gestiona tanto lecturas como escrituras genera cuellos de botella, afectando directamente el rendimiento y la disponibilidad del sistema.

- Una solución eficaz es la arquitectura maestro-esclavo, donde el nodo maestro procesa las escrituras y los esclavos se encargan de las lecturas, mejorando la escalabilidad y la tolerancia a fallos.

- NGINX puede configurarse como balanceador de carga a nivel TCP, permitiendo distribuir dinámicamente las solicitudes de lectura entre varios nodos de base de datos.

- Objetivo: Diseñar e implementar esta arquitectura con NGINX y evaluar su rendimiento y tolerancia a fallos mediante pruebas con SysBench.



Contexto

Demanda digital y desafíos de escalabilidad

- El crecimiento acelerado de plataformas web ha generado una mayor demanda de servicios digitales, impulsada por la digitalización y el acceso masivo a internet.
- Según Business Research Insights (2023), en 2023, el mercado global de desarrollo web alcanzó USD 65.350 millones y se proyecta que llegará a USD 130.900 millones en 2032.
- Esta tendencia ha evidenciado las limitaciones de arquitecturas monolíticas:
  - Saturación de servidores
  - Largos tiempos de respuesta
  - Fallos en el acceso a datos
- EduConnect, plataforma educativa usada por 10+ instituciones en Colombia, sufrió interrupciones y errores críticos tras un aumento del 40% en usuarios. Su infraestructura, basada en una única instancia MySQL, no soportó la carga concurrente y carecía de mecanismos de balanceo y tolerancia a fallos.

Alternativas de Solución

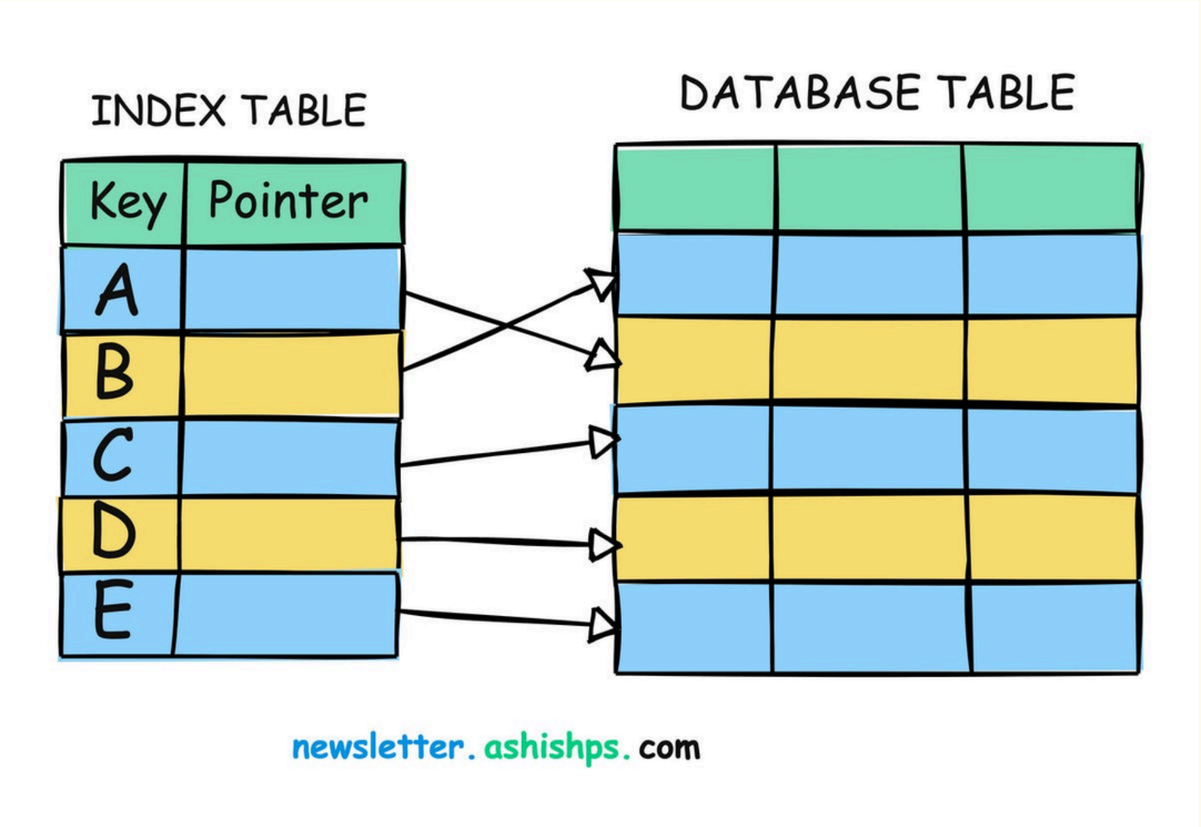
Base de Datos INDEXADA

VENTAJAS

- **Mejora inmediata del rendimiento sin cambiar arquitectura.**
- **Bajo costo.**
- **Requiere poco mantenimiento.**

DESVENTAJAS:

- **Alcance limitado ante crecimiento masivo.**
- **No resuelve cuellos de botella en concurrencia**



Extraído de Medium, DevCookies

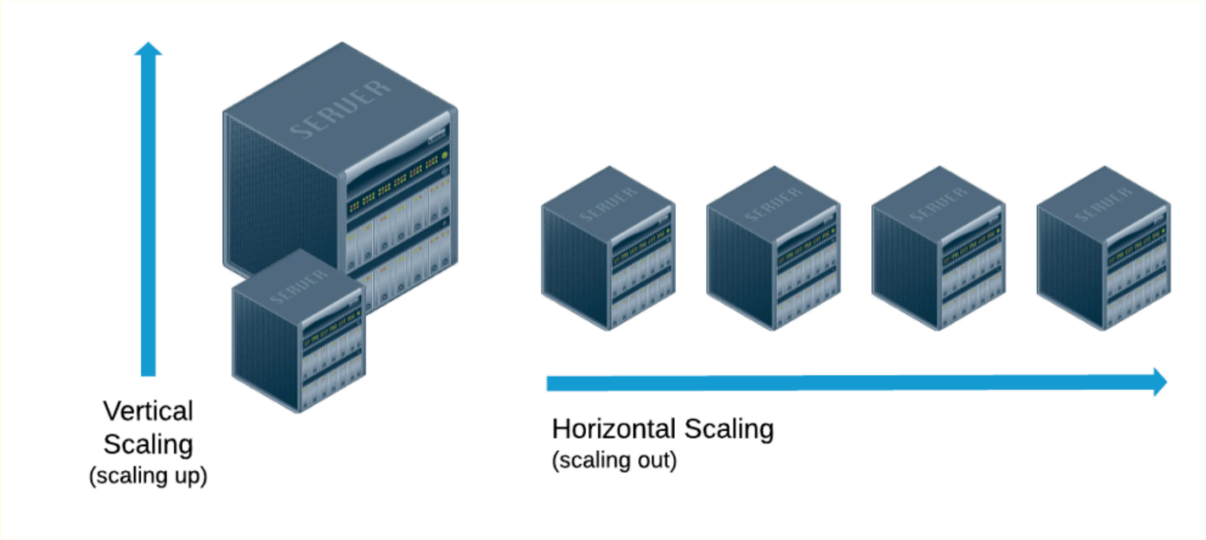
Escalamiento Vertical

VENTAJAS:

- **Rápida implementación.**
- **Sin complejidad en la arquitectura.**

DESVENTAJAS:

- **Costoso y con límite físico.**
- **Punto único de falla.**



Extraído de Webscale,

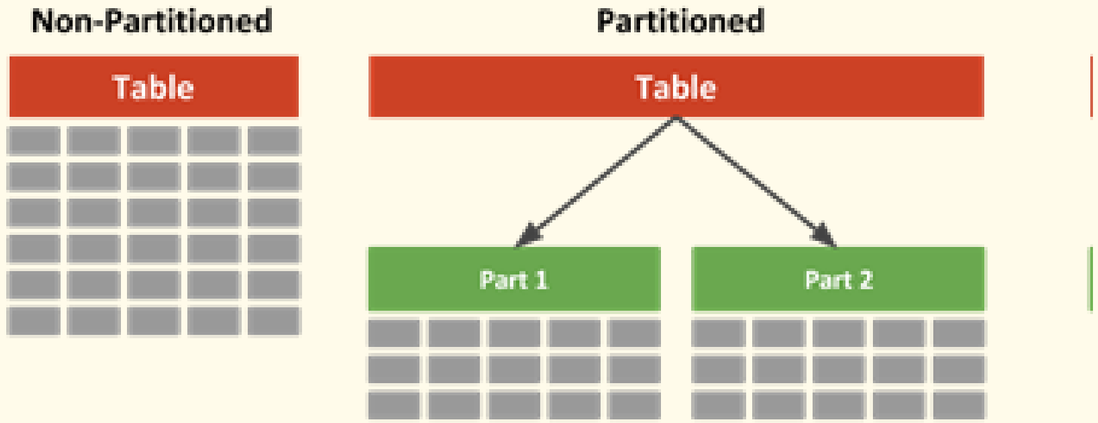
Particionamiento de Tablas

VENTAJAS

- **Mejora tiempos de consulta en datasets grandes.**

DESVENTAJAS:

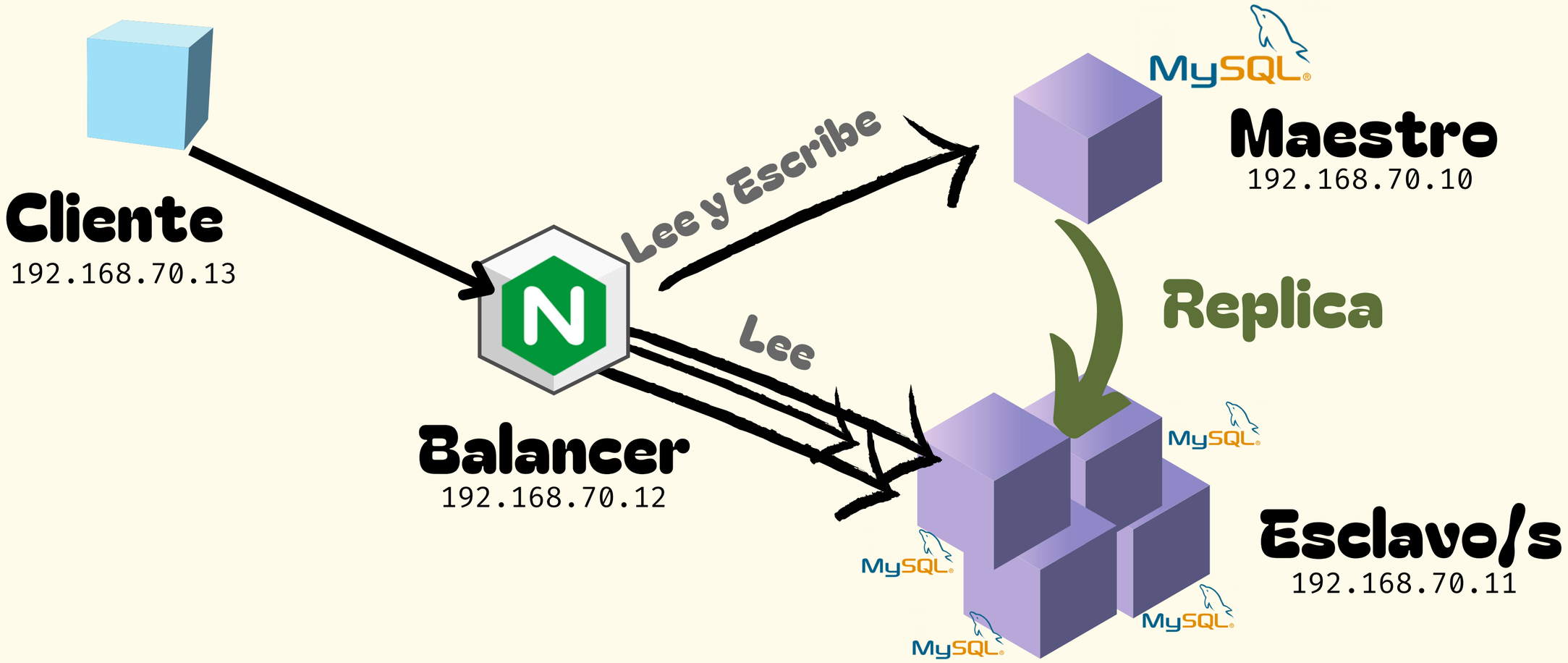
- **Complejidad en diseño y mantenimiento.**
- **No mejora concurrencia global si no se combina con otras técnicas.**



Extraído de Webscale,

Diseño de Solución

Balanceador de Carga



Ventajas

Escalable: Permite agregar más instancias de Bases de Datos

Rápido: Lecturas no son frenadas por escrituras.

Resiliente: La caída del Esclavo no afecta el comportamiento del sistema.

Seguro: acceso solo vía NGINX.

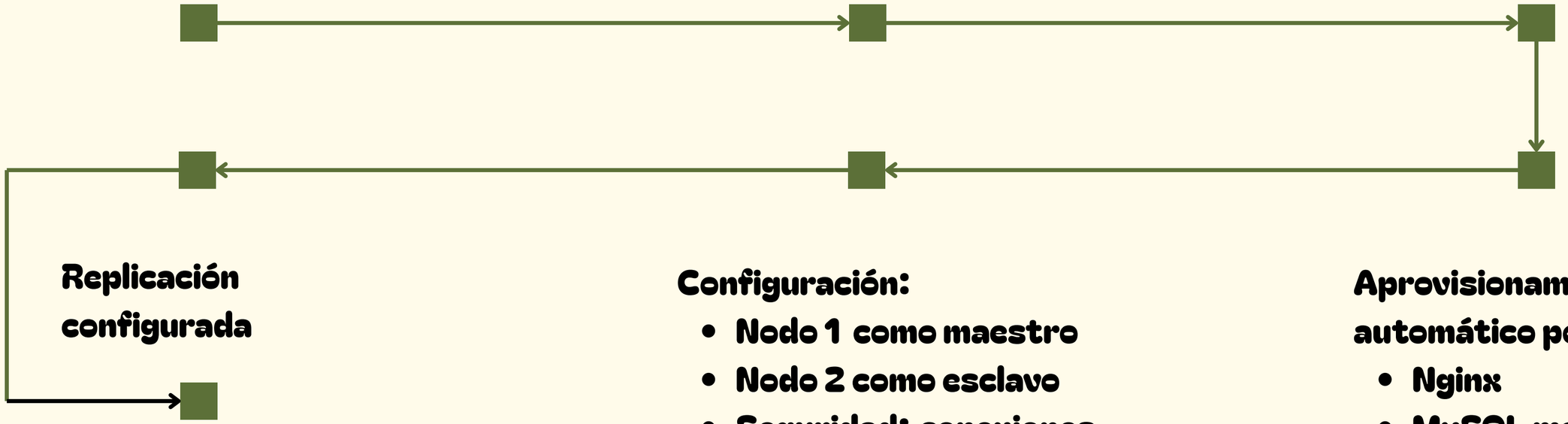
IMPLEMENTACIÓN

RUTA DE IMPLEMENTACIÓN

Instalación Vagrant/Virtualbox

Creacion del Vagrantfile

Levantamiento de máquinas + conexión SSH por nodo



Aprovisionamiento automático por nodo:

- Nginx
- MySQL maestro
- MySQL esclavo
- Cliente con Sysbench

Configuración:

- Nodo 1 como maestro
- Nodo 2 como esclavo
- Seguridad: conexiones solo desde IPs autorizadas

Replicación configurada

Pruebas de carga con Sysbench

Provisionamiento

Balanceador (Nginx): Instala y configura Nginx como proxy TCP con balanceo para lecturas y escrituras.

/etc/nginx/nginx.conf

```
stream {
    log_format proxy_logs '$remote_addr [$time_local] '
                          '$protocol $status $bytes_sent $bytes_received '
                          '$session_time "$upstream_addr"';


    access_log /var/log/nginx/mysql_access.log proxy_logs;

    upstream mysql_read {
        server 192.168.70.10:3306; # Maestro
        server 192.168.70.11:3306; # Esclavo
    }

    upstream mysql_write {
        server 192.168.70.10:3306; # Maestro
    }

    server {
        listen 3307;
        proxy_pass mysql_read;
        proxy_timeout 10s;
        proxy_connect_timeout 1s;
    }

    server {
        listen 3308;
        proxy_pass mysql_write;
        proxy_timeout 10s;
        proxy_connect_timeout 1s;
    }
}
```

Provisionamiento

Maestro MySQL: Activa binlog, crea usuario de replicación, y restringe acceso remoto por IP.

/etc/mysql/mysql.conf.d/mysqld.cnf

```
server-id               = 1
log_bin                 = /var/log/mysql/mysql-bin.log
binlog_format           = row
bind-address            = 0.0.0.0
```

Provisionamiento

Esclavo MySQL:Se conecta automáticamente al maestro y activa replicación (CHANGE MASTER TO).

/etc/mysql/mysql.conf.d/mysqld.cnf

```
server-id               = 2
log_bin                 = /var/log/mysql/mysql-bin.log
relay_log               = /var/log/mysql/mysql-relay-bin.log
binlog_format           = row
skip-networking = false
read_only = ON
```

Provisionamiento

Cliente: Instala Sysbench y genera datos de prueba.

```
apt-get update
apt-get install -y sysbench

echo "[INFO] Preparando los datos para la prueba en el cliente..."

sysbench /usr/share/sysbench/oltp_read_write.lua \
--mysql-host=192.168.70.12 \
--mysql-port=3308 \
--mysql-user=root \
--mysql-password=admin \
--mysql-db=sbtest \
--tables=4 \
--table-size=10000 \
prepare

echo "[✅] Todo listo para la prueba en el cliente."
```

Plan de Pruebas

Objetivo

Evaluar el rendimiento de lectura y escritura del sistema con distintos niveles de concurrencia, midiendo métricas clave como transacciones por segundo y latencia.

Tipos de pruebas

Habrán dos tipos de pruebas, las cuales son

- Pruebas de Escritura
  solo inserciones tipo INSERT
- Pruebas de Lectura
  solo consultas tipo SELECT

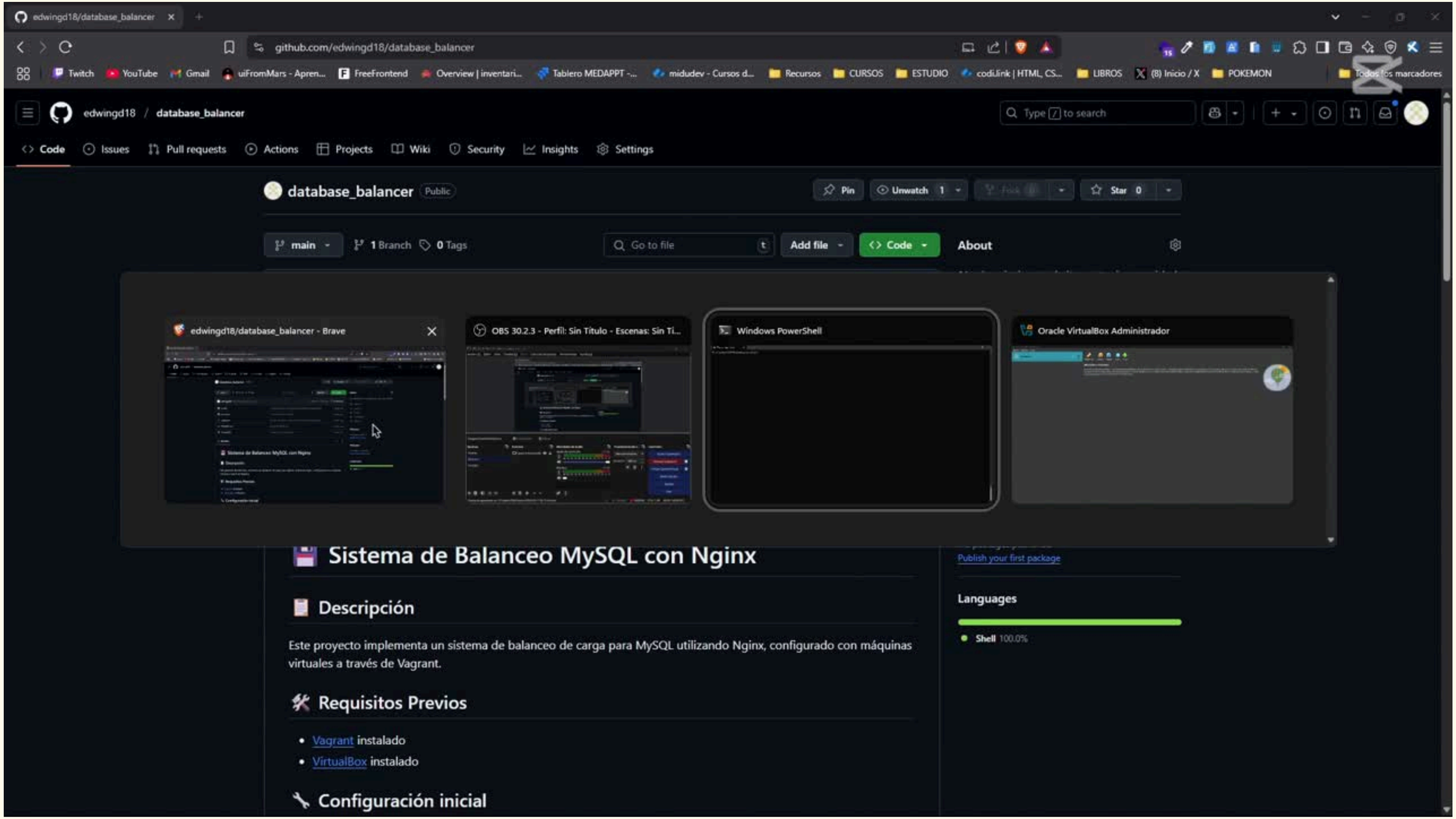
✅ **Fase 1:** Prueba Inicial (**baseline**)

- Se realiza una prueba controlada con parámetros por defecto:
  - Duración: 30 segundos
  - Número de hilos: 8
- Su propósito es obtener una línea base de rendimiento del sistema en condiciones normales.
- Este resultado sirve como referencia comparativa para todas las demás ejecuciones.

⚠ **Fase 2:** Pruebas de carga incremental

- Se incrementa la cantidad de hilos de 150 en 150 hasta que el sistema empieza a fallar.
- Se repite tanto para lectura como para escritura.
- El objetivo es identificar el punto de ruptura o el límite de capacidad del sistema, observando:
  - Aumento del tiempo total de ejecución
  - Disminución en el número de transacciones por segundo
  - Aparición de errores o reconexiones

DEMO


edwingd18/database_balancer
github.com/edwingd18/database_balancer
edwingd18 / database_balancer
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
database_balancer
Public
Pin
Unwatch 1
Star 0
main
1 Branch
0 Tags
Go to file
Add file
Code
About
edwingd18/database_balancer - Brave
OBS 30.2.3 - Perfil: Sin Título - Escenas: Sin Tí...
Windows PowerShell
Oracle VirtualBox Administrador
Sistema de Balanceo MySQL con Nginx
Descripción
Este proyecto implementa un sistema de balanceo de carga para MySQL utilizando Nginx, configurado con máquinas virtuales a través de Vagrant.
Requisitos Previos
Vagrant instalado
VirtualBox instalado
Configuración inicial
Publish your first package
Languages
Shell 100.0%

Discusión de Resultados

Fase 1

"obtener una línea base de rendimiento del sistema en condiciones normales."

Pruebas de Escritura

Métrica	Valor
Threads (hilos)	8
Duración total	30.21 s
TPS (avg)	48.82 transacciones/seg
QPS (avg)	293.02 consultas/seg
Transacciones totales	1,475
Consultas totales	8,853
• Lecturas	0
• Escrituras	5,902
• Otras	2,951
Latencia (ms)	
• Mínima	60.79
• Promedio	163.06
• Máxima	1,762.34
• Percentil 95	267.41
Errores ignorados	1 (0.03/s)
Reconexiones	0
Equidad de hilos	Eventos promedio: 184.38 ± 3.90
	Tiempo ejecución promedio: 30.06 s ± 0.06 s

Observaciones

- Buen rendimiento con 48.82 TPS y baja latencia promedio (163 ms).
- Latencia máxima alta sugiere picos ocasionales de bloqueo o espera.
- Carga bien distribuida entre hilos, sin reconexiones ni errores significativos.

Pruebas de Lectura

Métrica	Valor
Threads (hilos)	8
Duración total	30.14 s
TPS (avg)	69.77 transacciones/seg
QPS (avg)	1,116.28 consultas/seg
Transacciones totales	2,103
Consultas totales	33,648
• Lecturas	29,442
• Escrituras	0
• Otras	4,206
Latencia (ms)	
• Mínima	69.32
• Promedio	114.33
• Máxima	1,172.49
• Percentil 95	173.58
Errores ignorados	0
Reconexiones	0
Equidad de hilos	Eventos promedio: 262.88 ± 9.43
	Tiempo ejecución promedio: 30.05 s ± 0.04 s

Observaciones

- Buen rendimiento con 69.77 TPS y baja latencia promedio (114 ms).
- Latencia máxima más controlada que en escritura.
- Sin errores ni reconexiones, buena equidad de hilos y estabilidad.

Fase 2

"identificar el punto de ruptura o el límite de capacidad del sistema."

- **Pruebas de Escritura**

150 Hilos

Métrica	Valor
Threads (hilos)	150
Duración total	31.36 s
TPS (avg)	99.03 transacciones/seg
QPS (avg)	594.66 consultas/seg
Transacciones totales	3,106
Consultas totales	18,652
• Lecturas	0
• Escrituras	12,433
• Otras	6,219
Latencia (ms)	
• Mínima	123.69
• Promedio	1,476.82
• Máxima	8,066.43
• Percentil 95	3,911.79
Errores ignorados	7 (0.22/s)
Reconexiones	0
Equidad de hilos	Eventos promedio: 20.71 ± 3.16
	Tiempo ejecución promedio: 30.58 s ± 0.35 s

Observaciones

- Con 150 hilos, el sistema aumenta TPS (99) pero la latencia promedio se dispara (1,476 ms), mostrando saturación parcial.
- El percentil 95 alto (3,911 ms) indica variabilidad significativa y retrasos puntuales.
- Se registran algunos errores ignorados, pero sin reconexiones.

300 Hilos

Métrica	Valor
Threads solicitados	300
Resultado	✗ Fallo total – Too many connections (Error 1040)
Motivo	El servidor MySQL alcanzó el **límite máximo de conexiones permitidas.**
Ubicación del error	Durante la inicialización de los hilos (thread_init)

A 300 hilos, el servidor falla por límite de conexiones (Error 1040), señal clara de que no puede soportar esa carga simultánea.

- **Pruebas de Lectura**

150 Hilos

Métrica	Valor
Threads (hilos)	150
Duración total	30.99 s
TPS (avg)	130.71 transacciones/seg
QPS (avg)	2,091.30 consultas/seg
Transacciones totales	4,052
Consultas totales	64,832
• Lecturas	56,728
• Escrituras	0
• Otras	8,104
Latencia (ms)	
• Mínima	383.8
• Promedio	1,126.69
• Máxima	4,529.32
• Percentil 95	1,973.38
Errores ignorados	0 (0.00/s)
Reconexiones	0
Equidad de hilos	Eventos promedio: 27.01 ± 2.45
	Tiempo ejecución promedio: 30.43 s ± 0.25 s

A 150 hilos, el TPS sube a 130 y la latencia promedio es aceptable (~1,126 ms), aunque ya notablemente más alta que en fase 1.

300 Hilos

Métrica	Valor
Threads (hilos)	300
Duración total	32.36 s
TPS (avg)	120.15 transacciones/seg
QPS (avg)	1,922.33 consultas/seg
Transacciones totales	3,888
Consultas totales	62,208
• Lecturas	54,432
• Escrituras	0
• Otras	7,776
Latencia (ms)	
• Mínima	660.61
• Promedio	2,386.80
• Máxima	7,350.12
• Percentil 95	4,517.90
Errores ignorados	0 (0.00/s)
Reconexiones	0
Equidad de hilos	Eventos promedio: 12.96 ± 1.08
	Tiempo ejecución promedio: 30.93 s ± 0.66 s

A 300 hilos, el TPS baja ligeramente (120) pero la latencia promedio se dispara a más de 2,300 ms, con máximos que superan los 7 segundos, indicando saturación importante.

450 Hilos

Métrica	Valor
Threads solicitados	450
Resultado	✗ Fallo total – Too many connections (Error 1040)
Motivo	El servidor MySQL alcanzó el **límite máximo de conexiones permitidas.**
Ubicación del error	Durante la inicialización de los hilos (thread_init)

A 450 hilos, falla total por límite de conexiones (Error 1040), confirmando que el servidor no puede manejar tantas conexiones concurrentes.

Conclusiones

- La solución basada en NGINX y replicación maestro-esclavo permitió distribuir eficientemente las lecturas, reduciendo la carga del nodo maestro.
- Las pruebas con SysBench confirmaron mejoras en rendimiento, escalabilidad y tolerancia a fallos bajo cargas concurrentes.
- Es una alternativa viable en entornos locales con recursos limitados, ideal para instituciones educativas y pequeñas organizaciones.
- El uso de NGINX como balanceador TCP simplifica la administración, refuerza la seguridad y centraliza el acceso a la base de datos.
- Se concluye que esta arquitectura es una estrategia eficaz y flexible para afrontar el crecimiento y la demanda en plataformas digitales.

¡Gracias!