

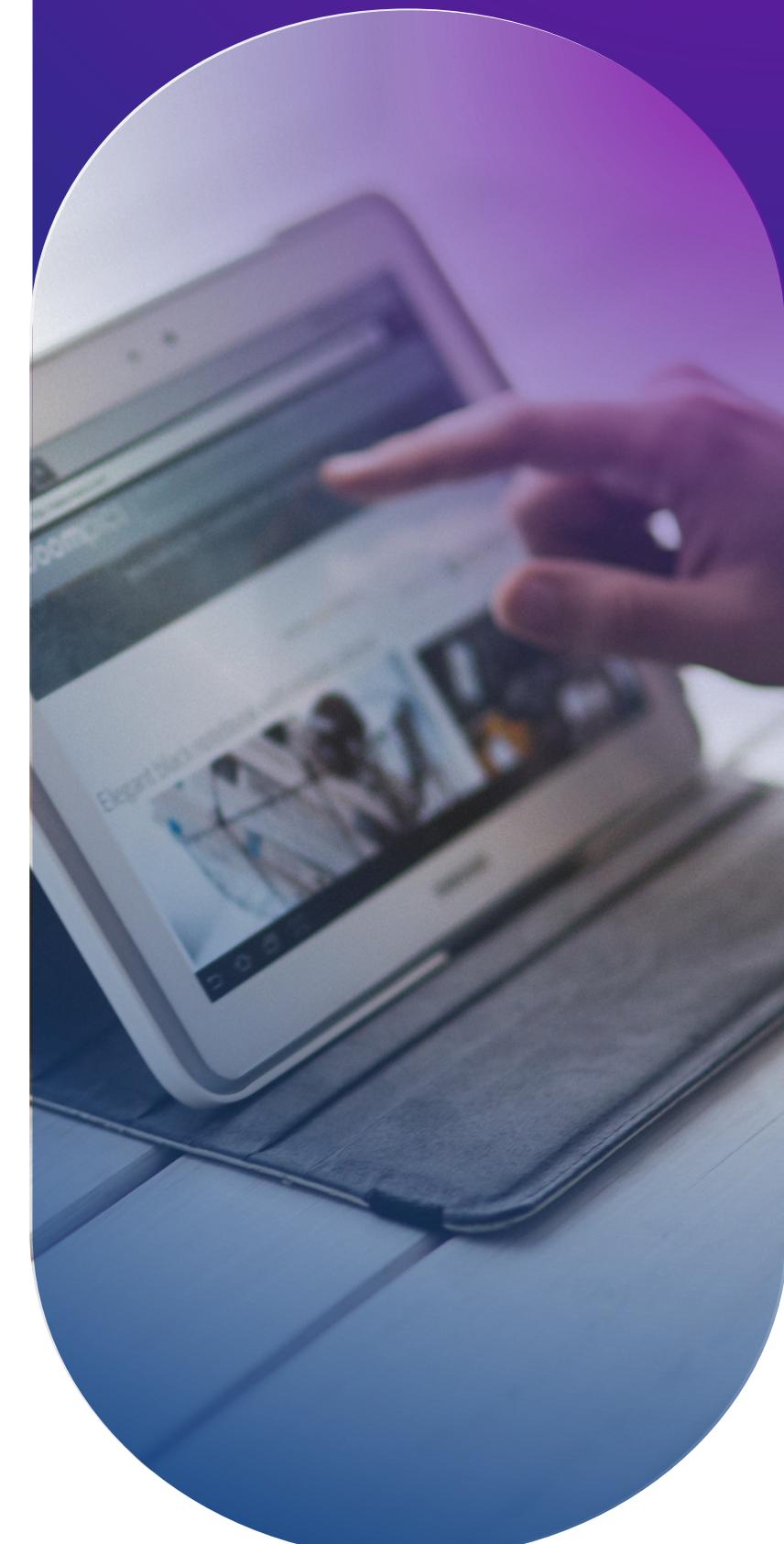
# BALANCEO DE CARGA DE BASES DE DATOS CON MYSQL Y NGINX

Alejandro Bravo Isajar  
Joan Sebastian Balanta  
Juan Eduardo Jaramillo  
Fernando Jose Cedeño

# INTRODUCCIÓN

- Las **arquitecturas monolíticas** con una sola **instancia MySQL** alcanzan rápidamente los **límites físicos**, afectando directamente la **disponibilidad** y el **rendimiento**.
- La **replicación maestro-esclavo** permite separar **operaciones**, mejorando **escalabilidad** y **tolerancia a fallos**.
- **NGINX** puede configurarse como **balanceador TCP** para **distribuir lecturas** entre **nodos esclavos**.

**Objetivo:** Diseñar e implementar un balanceador de carga para MySQL con NGINX y validar la efectividad de la solución mediante SysBench.



# CONTEXTO



- El crecimiento de plataformas web ha aumentado la demanda de servicios digitales. En **2023**, el **mercado global** de desarrollo web fue de **\$65.350M** y se espera que llegue a **\$130.900M** en **2032**.
- Este avance ha **expuesto** las limitaciones de las arquitecturas monolíticas, como la **saturación** de servidores y **fallos** en el acceso a datos.
- El **53%** de **usuarios móviles** abandona **sitios lentos** (>3s).

**Caso real:** EduConnect sufrió interrupciones tras un aumento del 40% en usuarios.

# ALTERNATIVAS DE SOLUCIÓN

- **Optimización de consultas:**

- Mejora índices y estructura de datos para menor tiempo de respuesta.
- Requiere análisis y posible rediseño de la base.

- **Escalamiento vertical:**

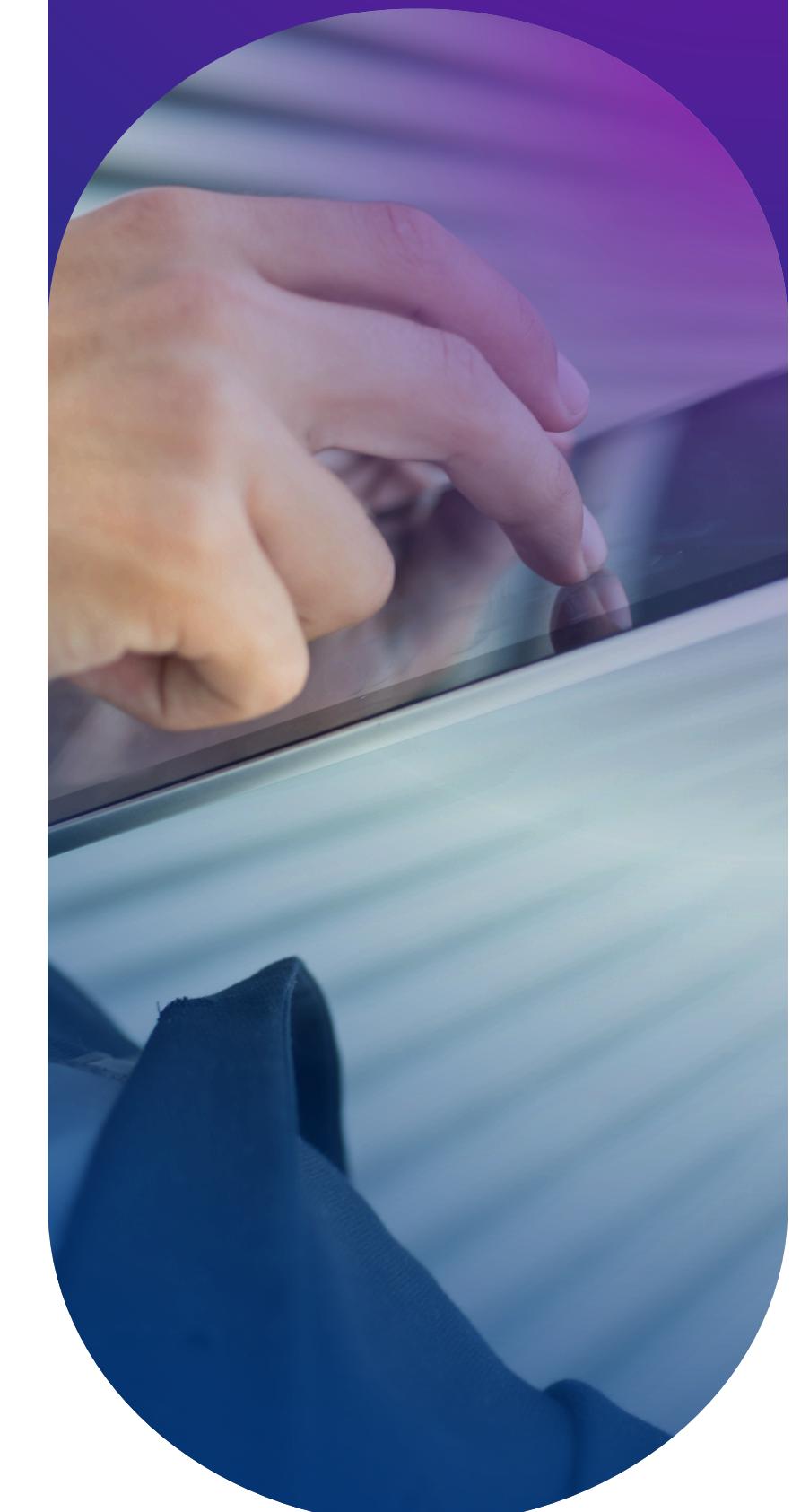
- Aumentar recursos del servidor (CPU, RAM, disco).
- Rápido pero costoso y limitado físicamente.

- **Particionado de tablas:**

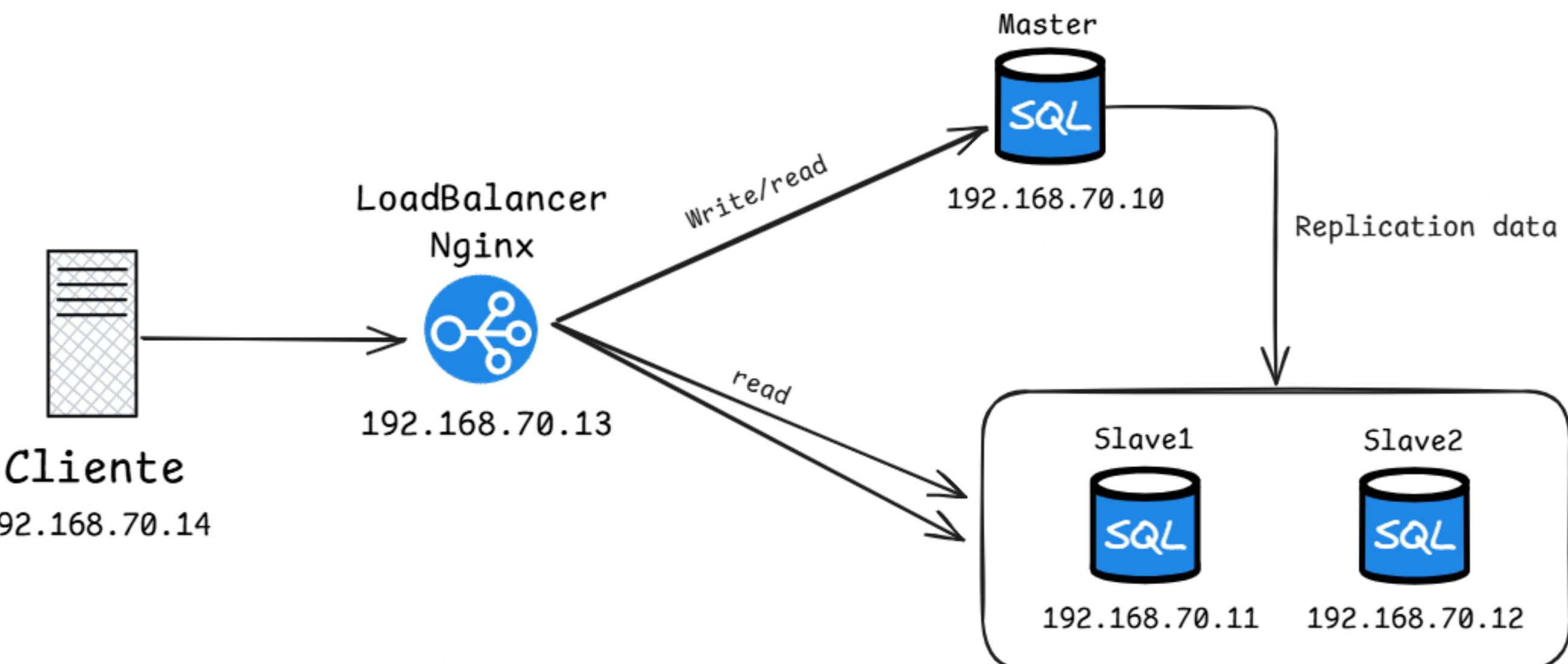
- Divide grandes tablas para consultas más rápidas.
- Mayor complejidad y mantenimiento.

- **Replicación maestro-esclavo:**

- Mejora disponibilidad y distribución de carga.
- Riesgo de lecturas desactualizadas.



# DISEÑO DE LA SOLUCIÓN

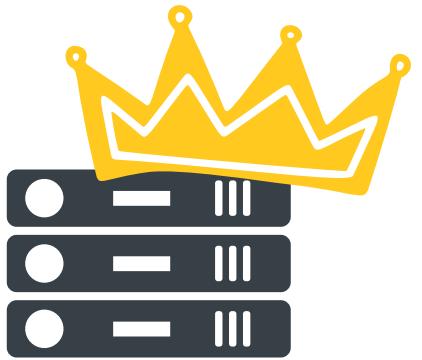


# VENTAJAS DE LA ARQUITECTURA



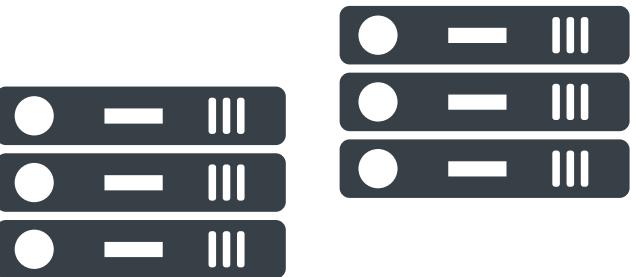
- El uso de réplicas de solo lectura permite escalar horizontalmente el sistema, simplemente añadiendo nuevos esclavos.
- Permite separar eficientemente la carga de trabajo. Las lecturas se reparten entre el maestro y los esclavos, reduciendo la presión sobre el maestro.
- NGINX centraliza el acceso, simplificando la gestión y mejorando la seguridad, ya que los nodos no quedan expuestos directamente.

# IMPLEMENTACIÓN



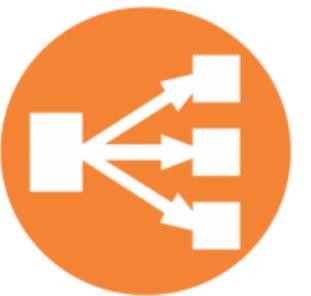
## Configuración master

Escrituras y fuente de replicación



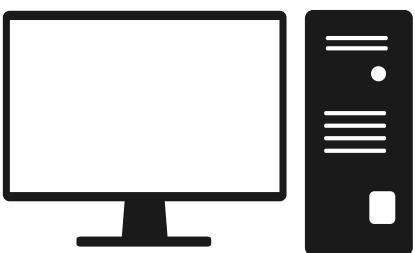
## Configuración esclavos

Réplica para lecturas



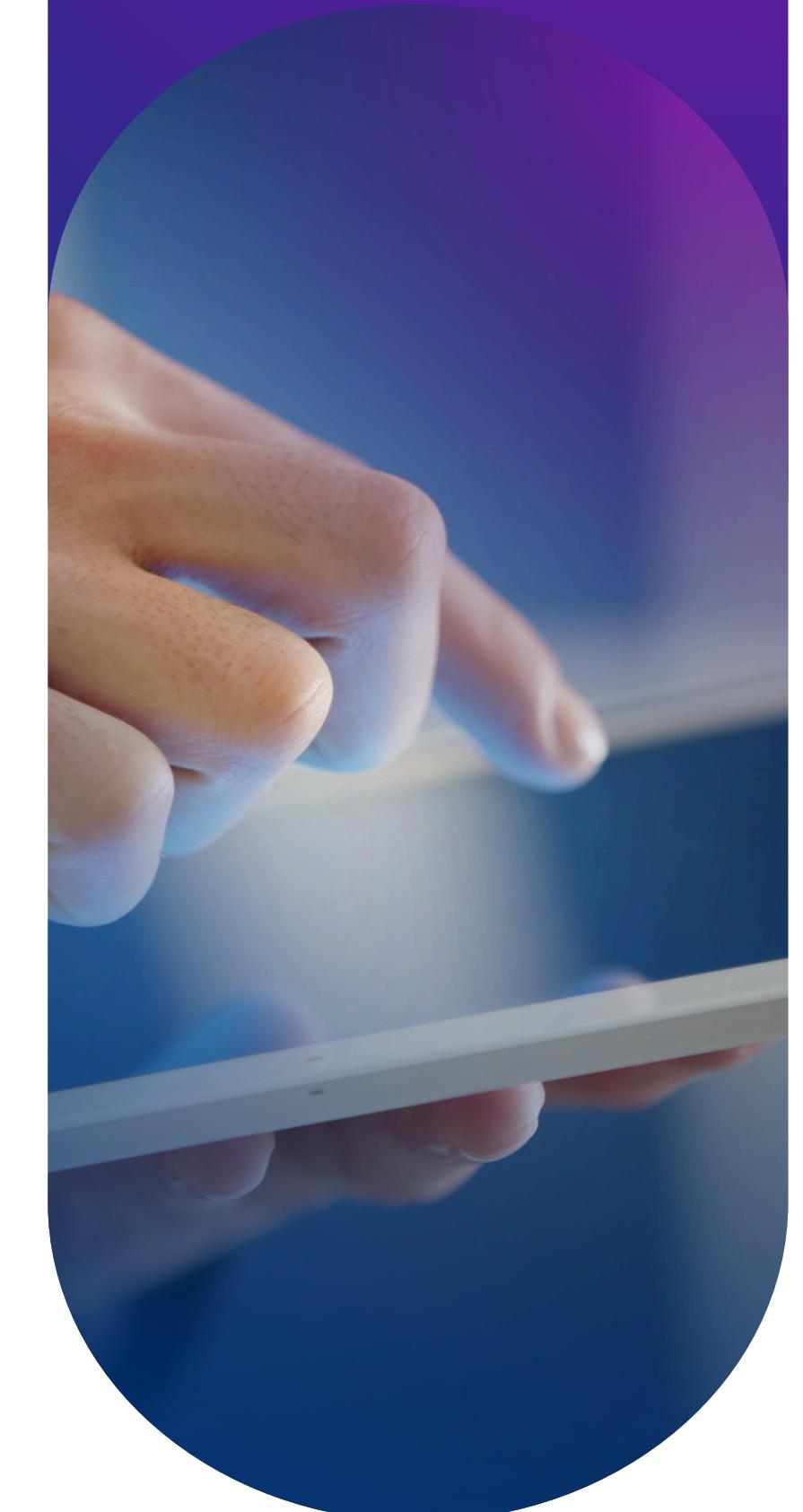
## Balanceador de carga (Nginx)

Proxy TCP (puertos 3307/3308)



## SysBench (Cliente)

Herramientas de prueba



# APROVISIONAMIENTO

## MySQL - MASTER

```
[mysql]      You, last week • Base del proyecto ...
# Configuración mínima para el servidor MySQL (maestro)
user = mysql

# Network: permitir conexiones de red (necesario para replicación en Vagrant)
skip-networking = false
mysqlx-bind-address = 127.0.0.1

# Performance tuning (valores modestos para entorno de pruebas)
key_buffer_size = 16M
myisam-recover-options = BACKUP

# Logging de errores
log_error = /var/log/mysql/error.log

# Binlog / Replicación (maestro)
# server-id único en la topología (maestro = 1)
max_binlog_size = 100M
server-id = 1
log_bin = /var/log/mysql/mysql-bin.log
binlog_format = row

# Escuchar en todas las interfaces para que otras VMs puedan conectarse
bind-address = 0.0.0.0
```

```
# Configuración de red DNS
sudo systemctl stop systemd-resolved
sudo systemctl disable systemd-resolved
sudo rm -f /etc/resolv.conf
echo "nameserver 8.8.8.8" | sudo tee /etc/resolv.conf > /dev/null

echo "Instalando MySQL Server en el maestro ..."

export DEBIAN_FRONTEND=noninteractive
apt-get update
apt-get install -y mysql-server

echo "Configurando MySQL como maestro ..."
systemctl stop mysql
cp /vagrant/config/my.conf.master /etc/mysql/mysql.conf.d/mysqld.cnf
systemctl start mysql

if systemctl is-active --quiet mysql; then
    echo "MySQL está corriendo como maestro."
else
    echo "MySQL no se pudo iniciar." >&2
    exit 1
fi

# Crear usuarios de replicación para ambos slaves
mysql -uroot -padmin <<EOF
-- Usuario de replicación para slave1 (192.168.70.11)
CREATE USER 'replicator'@'192.168.70.11' IDENTIFIED WITH mysql_native_password BY 'replicator_pass';
GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'replicator'@'192.168.70.11';

-- Usuario de replicación para slave2 (192.168.70.12)
CREATE USER 'replicator'@'192.168.70.12' IDENTIFIED WITH mysql_native_password BY 'replicator_pass';
GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'replicator'@'192.168.70.12';

FLUSH PRIVILEGES;
EOF

echo "Configurando acceso remoto desde balanceador y slaves ..."
mysql -uroot -padmin <<EOF
-- Acceso desde balanceador (192.168.70.13)
CREATE USER 'root'@'192.168.70.13' IDENTIFIED BY 'admin';
GRANT ALL PRIVILEGES ON *.* TO 'root'@'192.168.70.13' WITH GRANT OPTION;

-- Acceso desde slave1 (192.168.70.11)
CREATE USER 'root'@'192.168.70.11' IDENTIFIED BY 'admin';
GRANT ALL PRIVILEGES ON *.* TO 'root'@'192.168.70.11' WITH GRANT OPTION;

-- Acceso desde slave2 (192.168.70.12)
CREATE USER 'root'@'192.168.70.12' IDENTIFIED BY 'admin';
GRANT ALL PRIVILEGES ON *.* TO 'root'@'192.168.70.12' WITH GRANT OPTION;

FLUSH PRIVILEGES;
EOF
```

Configuration

Provisioning

# APROVISIONAMIENTO

## MySQL - SLAVES

```
[mysqld]      You, last week • Base del proyecto ...
# Configuración mínima para el servidor MySQL (esclavo)
user = mysql

# Network: permitir conexiones desde maestro y herramientas de administración
bind-address = 0.0.0.0
mysqlx-bind-address = 127.0.0.1

# Performance tuning (valores modestos para entorno de pruebas)
key_buffer_size = 16M
myisam-recover-options = BACKUP

# Logging de errores
log_error = /var/log/mysql/error.log

# Binlog / Replicación (esclavo)
# server-id debe ser único (esclavo = 2)
max_binlog_size = 100M
server-id = 2
log_bin = /var/log/mysql/mysql-bin.log
relay_log = /var/log/mysql/mysql-relay-bin.log
binlog_format = row

# Habilitar red y poner el esclavo en solo lectura
skip-networking = false
read_only = ON
```

```
# Configuración de red DNS
sudo systemctl stop systemd-resolved
sudo systemctl disable systemd-resolved
sudo rm -f /etc/resolv.conf
echo "nameserver 8.8.8.8" | sudo tee /etc/resolv.conf > /dev/null

echo "Instalando MySQL Server en el esclavo 1 ..."

export DEBIAN_FRONTEND=noninteractive
apt-get update
apt-get install -y mysql-server

echo "Configurando MySQL como esclavo 1 ..."
systemctl stop mysql
cp /vagrant/config/my.conf.slave /etc/mysql/mysql.conf.d/mysqld.cnf
systemctl start mysql

if systemctl is-active --quiet mysql; then
    echo "MySQL está corriendo como esclavo 1."
else
    echo "MySQL no se pudo iniciar." >&2
    exit 1
fi

# Conexión remota al maestro para obtener el log bin y la posición
read -r MASTER_LOG_FILE MASTER_LOG_POS <<< $(mysql -uroot -padmin -e "SHOW MASTER"
if [[ -z "$MASTER_LOG_FILE" || -z "$MASTER_LOG_POS" ]]; then
    echo "No se pudo obtener log binario y posición del maestro." >&2
    exit 1
fi

echo "MASTER_LOG_FILE = $MASTER_LOG_FILE"
echo "MASTER_LOG_POS = $MASTER_LOG_POS"

echo "Configurando esclavo 1 para replicar desde el maestro ..."
mysql -uroot -padmin <<< EOF
STOP SLAVE;
RESET SLAVE ALL;
CHANGE MASTER TO
    MASTER_HOST='192.168.70.10',
    MASTER_USER='replicator',
    MASTER_PASSWORD='replicator_pass',
    MASTER_LOG_FILE='$MASTER_LOG_FILE',
    MASTER_LOG_POS=$MASTER_LOG_POS;
START SLAVE;
EOF

echo "Verificando estado de la replicación ..."
sleep 3

SLAVE_IO_RUNNING=$(mysql -uroot -padmin -e "SHOW SLAVE STATUS\G" | grep "Slave_IO_Running:" | awk '{print $2}')
SLAVE_SQL_RUNNING=$(mysql -uroot -padmin -e "SHOW SLAVE STATUS\G" | grep "Slave_SQL_Running:" | awk '{print $2}')

if [[ "$SLAVE_IO_RUNNING" == "Yes" && "$SLAVE_SQL_RUNNING" == "Yes" ]]; then
    echo "La replicación está funcionando correctamente en esclavo 1."
else
    echo "La replicación NO está funcionando. Verifica configuración y logs." >&2
    mysql -uroot -padmin -e "SHOW SLAVE STATUS\G" | grep -E "Last_IO_Error|Last_SQL_Error"
    exit 1
fi
```

Configuration

Provisioning

# APROVISIONAMIENTO BALANCER

```
stream {
    # Formato de logs para conexiones TCP proxied (útil para MySQL)
    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 para lecturas (balanceo round-robin entre maestro y 2 esclavos)
    upstream mysql_read {
        server 192.168.70.10:3306; # Maestro
        server 192.168.70.11:3306; # Esclavo 1
        server 192.168.70.12:3306; # Esclavo 2
    }

    # Upstream para escrituras (solo maestro)
    upstream mysql_write {
        server 192.168.70.10:3306; # Maestro
    }

    # Puerto 3307 para lecturas: proxy hacia mysql_read
    server {
        listen 3307;
        proxy_pass mysql_read;
        proxy_timeout 10s;
        proxy_connect_timeout 1s;
    }

    # Puerto 3308 para escrituras: proxy hacia mysql_write
    server {
        listen 3308;
        proxy_pass mysql_write;
        proxy_timeout 10s;
        proxy_connect_timeout 1s;
    }
}
```

Configuration

```
sudo systemctl stop systemd-resolved
sudo systemctl disable systemd-resolved
sudo rm -f /etc/resolv.conf
echo "nameserver 8.8.8.8" | sudo tee /etc/resolv.conf > /dev/null

echo "Instalando nginx ..."

export DEBIAN_FRONTEND=noninteractive
apt-get update
apt-get install -y nginx

echo "Configurando nginx como balanceador de carga ... "
systemctl stop nginx
cp /vagrant/config/conf.balancer /etc/nginx/nginx.conf
systemctl start nginx

if systemctl is-active --quiet nginx; then
    echo "nginx está corriendo como balanceador de carga."
else
    echo "nginx no se pudo iniciar." >&2
    exit 1
fi
```

Provisioning

# APROVISIONAMIENTO

## CLIENT

```
# Configuración de red DNS
sudo systemctl stop systemd-resolved
sudo systemctl disable systemd-resolved
sudo rm -f /etc/resolv.conf
echo "nameserver 8.8.8.8" | sudo tee /etc/resolv.conf > /dev/null

echo "Instalando herramientas de prueba (mysql-client, sysbench) ..."

export DEBIAN_FRONTEND=noninteractive
apt-get update
apt-get install -y default-mysql-client sysbench
```

# VIDEO PRÁCTICA



<https://drive.google.com/file/d/1F1lftsG2RshVUO2gXBjHolrzsLQ6XfJi/view?usp=sharing>

# PRUEBAS

## Test 1: Inserción y búsqueda de datos

```
vagrant@client:~$ TIMESTAMP=$(date +%s)
mysql -uroot -padmin -h 192.168.70.13 -P3308 -e "INSERT INTO test.test (name) VALUES ('repl_test_$TIMESTAMP');"
echo "Insertado: repl_test_$TIMESTAMP"
mysql: [Warning] Using a password on the command line interface can be insecure.
Insertado: repl_test_1762816398
```

```
vagrant@client:~$ for i in {1..12}; do
    mysql -uroot -padmin -h 192.168.70.13 -P3307 -e "SELECT @@hostname, NOW();" 2>/dev/null
    sleep 0.5
done
+-----+
| @@hostname | NOW()           |
+-----+
| mysql-master | 2025-11-10 23:15:54 |
+-----+
+-----+
| @@hostname | NOW()           |
+-----+
| mysql-slave | 2025-11-10 23:15:54 |
+-----+
+-----+
| @@hostname | NOW()           |
+-----+
| mysql-slave2 | 2025-11-10 23:15:55 |
+-----+
+-----+
| @@hostname | NOW()           |
+-----+
| mysql-master | 2025-11-10 23:15:55 |
+-----+
+-----+
| @@hostname | NOW()           |
+-----+
| mysql-slave | 2025-11-10 23:15:56 |
+-----+
```

# PRUEBAS

## Test 2: Distribución de carga en los 3 nodos

```
vagrant@client:~$ mysql -uroot -padmin -h 192.168.70.13 -P3307 -e "SELECT @@hostname AS servidor, NOW() AS timestamp;"  
mysql: [Warning] Using a password on the command line interface can be insecure.  
+-----+  
| servidor | timestamp |  
+-----+  
| mysql-slave | 2025-11-11 00:48:42 |  
+-----+  
vagrant@client:~$ for i in {1..20}; do  
    mysql -uroot -padmin -h 192.168.70.13 -P3307 -e "SELECT @@hostname;" 2>/dev/null | grep -v "@@hostname"  
done | sort | uniq -c  
    7 mysql-master  
    6 mysql-slave  
    7 mysql-slave2
```

**Objetivo:** Confirmar que las conexiones de lectura realizadas a través de NGINX (puerto 3307) se distribuyen entre los tres nodos del clúster MySQL: mysql-master, mysql-slave, y mysql-slave2.

- Distribución aproximadamente equitativa entre los tres nodos.
- Confirma que NGINX está realizando balanceo de carga en modo round-robin o similar.

NGINX está balanceando correctamente las lecturas, adicionalmente mejora el rendimiento y la escalabilidad del sistema; y se garantiza alta disponibilidad y tolerancia a fallos

# PRUEBAS

```
vagrant@client:~$ sysbench /usr/share/sysbench/oltp_read_write.lua
--mysql-host=192.168.70.13 \
--mysql-port=3308 \
--mysql-user=root \
--mysql-password=admin \
--mysql-db=sbtest \
--tables=4 \
--table-size=10000 \
--threads=16 \
--time=60 \
--report-interval=10 \
run
sysbench 1.0.20 (using system LuaJIT 2.1.0-beta3)

Running the test with following options:
Number of threads: 16
Report intermediate results every 10 second(s)
Initializing random number generator from current time

Initializing worker threads ...

Threads started!      SQL statistics:
                        queries performed:
                           read:          67578
                           write:         19308
                           other:         9654
                           total:         96540
                           transactions: 4827  (80.19 per sec.)
                           queries:       96540 (1603.72 per sec.)
                           ignored errors: 0    (0.00 per sec.)
                           reconnects:    0    (0.00 per sec.)

                        General statistics:
                           total time:   60.1961s
                           total number of events: 4827

                        Latency (ms):
                           min:           62.11
                           avg:          199.31
                           max:          613.50
                           95th percentile: 331.91
                           sum:          962052.68

                        Threads fairness:
                           events (avg/stddev): 301.6875/28.85
                           execution time (avg/stddev): 60.1283/0.05
```

## Test 3: 70% lecturas 30% escrituras

**Objetivo:** Evaluar el rendimiento del clúster MySQL bajo carga mixta (lectura/escritura), midiendo TPS y latencia.

- Simula carga concurrente con 16 hilos durante 60 segundos.
- Realiza operaciones de lectura, escritura y otras (OLTP mixto).
- Reporta métricas cada 10 segundos.

Se identifica un buen throughput gracias al balanceo entre 3 nodos, latencia estable, sin picos extremos. Cero errores y sin reconexiones: entorno robusto.

# PRUEBAS

```
vagrant@client:~$ sysbench /usr/share/sysbench/oltp_read_only.lua \
--mysql-host=192.168.70.13 \
--mysql-port=3307 \
--mysql-user=root \
--mysql-password=admin \
--mysql-db=sbtest \
--threads=24 \
--time=60 \
--report-interval=10 \
run
```

```
SQL statistics:
  queries performed:
    read:          130620
    write:         0
    other:        18660
    total:        149280
  transactions:
    queries:      9330 (155.17 per sec.)
    total:        149280 (2482.67 per sec.)
  ignored errors: 0 (0.00 per sec.)
  reconnects:    0 (0.00 per sec.)

General statistics:
  total time:    60.1274s
  total number of events: 9330

Latency (ms):
  min:           25.72
  avg:          154.49
  max:         1236.65
  95th percentile: 320.17
  sum:        1441379.69

Threads fairness:
  events (avg/stddev): 388.7500/142.50
  execution time (avg/stddev): 60.0575/0.03
```

## Test 4: 100% lectura 3 nodos

**Objetivo:** Medir la capacidad de procesamiento de lecturas (queries/sec y TPS) en el clúster MySQL con los 3 nodos activos.

- Simula carga de solo lectura con 24 hilos concurrentes.
- Duración: 60 segundos.
- Reporte cada 10 segundos.

Se identifica alto throughput gracias al uso de los 3 nodos, latencia controlada incluso con 24 hilos. Sin errores ni reconexiones: entorno estable y eficiente.

# PRUEBAS

```
vagrant@client:~$ sysbench /usr/share/sysbench/oltp_read_only.lua \
--mysql-host=192.168.70.13 \
--mysql-port=3307 \
--mysql-user=root \
--mysql-password=admin \
--mysql-db=sbtest \
--threads=24 \
--time=60 \
--report-interval=10 \
run
```

```
Running the test with following options:
Number of threads: 24
Report intermediate results every 10 second(s)
Initializing random number generator from current time

Initializing worker threads ...
Threads started!

[ 10s ] thds: 24 tps: 229.77 qps: 3694.06 (r/w/o: 3232.22/0.00/461.83) lat (ms,95%): 170.48 err/s: 0.00 reconn/s: 0.00
[ 20s ] thds: 24 tps: 226.12 qps: 3617.32 (r/w/o: 3165.08/0.00/452.24) lat (ms,95%): 176.73 err/s: 0.00 reconn/s: 0.00
[ 30s ] thds: 24 tps: 212.46 qps: 3399.61 (r/w/o: 2974.80/0.00/424.81) lat (ms,95%): 204.11 err/s: 0.00 reconn/s: 0.00
[ 40s ] thds: 24 tps: 179.98 qps: 2882.84 (r/w/o: 2522.87/0.00/359.97) lat (ms,95%): 161.51 err/s: 0.00 reconn/s: 0.00
FATAL: mysql_stmt_execute() returned error 2013 (Lost connection to MySQL server during query) for query 'SELECT DISTINCT c FROM sbtest1 WHERE id BETWEEN ? AND ? ORDER BY c'
FATAL: 'thread_run' function failed: /usr/share/sysbench/oltp_common.lua:432: SQL error, errno = 2013, state = 'HY000': Lost connection to MySQL server during query
FATAL: mysql_stmt_execute() returned error 2013 (Lost connection to MySQL server during query) for query 'SELECT c FROM sbtest1 WHERE id BETWEEN ? AND ?'
FATAL: mysql_stmt_execute() returned error 2013 (Lost connection to MySQL server during query) for query 'SELECT c FROM sbtest1 WHERE id=?'
(Last message repeated 1 times)
FATAL: 'thread_run' function failed: /usr/share/sysbench/oltp_common.lua:419: SQL error, errno = 2013, state = 'HY000': Lost connection to MySQL server during query
FATAL: mysql_stmt_execute() returned error 2013 (Lost connection to MySQL server during query) for query 'SELECT c FROM sbtest1 WHERE id BETWEEN ? AND ?'
```

## Test 5: 100% lectura 2 nodos Simular caida de un Slave

**Objetivo:** Evaluar el impacto en rendimiento al reducir el número de nodos disponibles para lectura (de 3 a 2), simulando un fallo en slave2.

### Causas técnicas

- Recursos limitados de laboratorio
- Saturación de memoria
- Saturación de CPU
- Timeouts y desconexiones

### Recomendaciones para entornos productivos

- Memoria: mínimo 2-4 GB por nodo MySQL
- CPU: 2-4 cores por nodo
- max\_connections: ajustar según carga esperada (default: 151)
- Monitoreo: usar herramientas como PMM, Prometheus, Grafana

# COMPARATIVA TEST 4 VS 5



Configuración	TPS (aprox)	Latencia promedio (ms)	Latencia p95 (ms)
3 nodos (maestro + 2 slaves)	9330	154.49	320.17
2 nodos (maestro + 1 slave)	~33% menos (estimado)	más alta (~180-220 ms)	más variable (~350-450 ms)

## Interpretación técnica

Con 3 nodos:

- Lecturas distribuidas eficientemente → mayor TPS
- Latencia más baja y estable
- Menor riesgo de saturación

Con 2 nodos:

- Menor capacidad de procesamiento → menos TPS
- Mayor latencia por sobrecarga
- Posibles errores si se usan muchos hilos (24+)

## Recomendaciones para producción

- Mínimo 2-3 nodos para lectura distribuida
- Monitoreo activo para detectar caídas y redistribuir carga
- Escalabilidad horizontal para manejar concurrencia alta
- Simulaciones de fallo como parte del plan de pruebas

# CONCLUSIONES



La solución basada en replicación **maestro-esclavo** con balanceo de carga en NGINX mejora el **rendimiento, la disponibilidad y la escalabilidad** en sistemas con alta concurrencia. La arquitectura distribuye eficientemente las operaciones, **reduce la carga** del nodo maestro y **optimiza los tiempos** de respuesta. Además, SysBench confirmó su **tolerancia a fallos**, y NGINX simplifica la administración, **refuerza la seguridad** y centraliza el acceso, ofreciendo una **solución sólida y escalable** para plataformas digitales.