

Balanceador de carga usando NGINX



Equipo

Andres Higuera Lozano

Santiago Cortés Galvis

Yusef Gabriel Aquil Tezna

Nicolas Gonzalez Franco

Menú

- Introducción
- Contexto
- Alternativas de Solución
- Diseño de la solución
- Implementación
- Pruebas y conclusiones

Introducción

En la actualidad, la escalabilidad y la disponibilidad constante de los servicios web son fundamentales para garantizar una buena experiencia de usuario. A medida que crecen en uso y complejidad, se hace necesario distribuir eficientemente la carga de trabajo entre varios servidores para evitar cuellos de botella y mejorar el rendimiento general.

Este proyecto implementa un sistema de balanceo de carga con el servidor NGINX, utilizando el algoritmo Least Connections, que asigna nuevas solicitudes al servidor con menor número de conexiones activas, logrando así una distribución dinámica e inteligente del tráfico.

Contexto

Los servicios web actuales requieren infraestructuras que garanticen alto rendimiento, disponibilidad constante y escalabilidad ante aumentos de demanda. Un enfoque centralizado, con un solo servidor, rápidamente se vuelve un cuello de botella y presenta riesgo de fallos bajo sobrecarga.

La virtualización y las arquitecturas distribuidas han impulsado el uso de balanceadores de carga, que distribuyen solicitudes de forma inteligente entre múltiples servidores, ofreciendo alta disponibilidad, tolerancia a fallos y adaptación dinámica.

La solución desarrollada responde a esta necesidad mediante el uso de tecnologías modernas como NGINX, Vagrant, VirtualBox y Artillery, integradas en una práctica con enfoque profesional y formativo.



Alternativas de solución

Existen múltiples algoritmos y plataformas para realizar balanceo de carga, desde opciones básicas como Round Robin o Least Connections hasta soluciones avanzadas que consideran latencia o métricas de uso de recursos.

Además de NGINX, existen tecnologías como HAProxy, Traefik o servicios gestionados en la nube como Azure Application Gateway, AWS ALB o Google Cloud Load Balancer, que ofrecen balanceo dinámico, monitoreo integrado y escalabilidad global.

Round Robin

El algoritmo Round Robin es una técnica simple y ampliamente adoptada, en la que las solicitudes se reparten secuencialmente entre los servidores del grupo.

IP Hash (Source)

El algoritmo IP Hash (también llamado Source) asigna a cada cliente un servidor específico basado en su dirección IP. Este método tiene la ventaja de mantener la afinidad de sesión (sticky session).

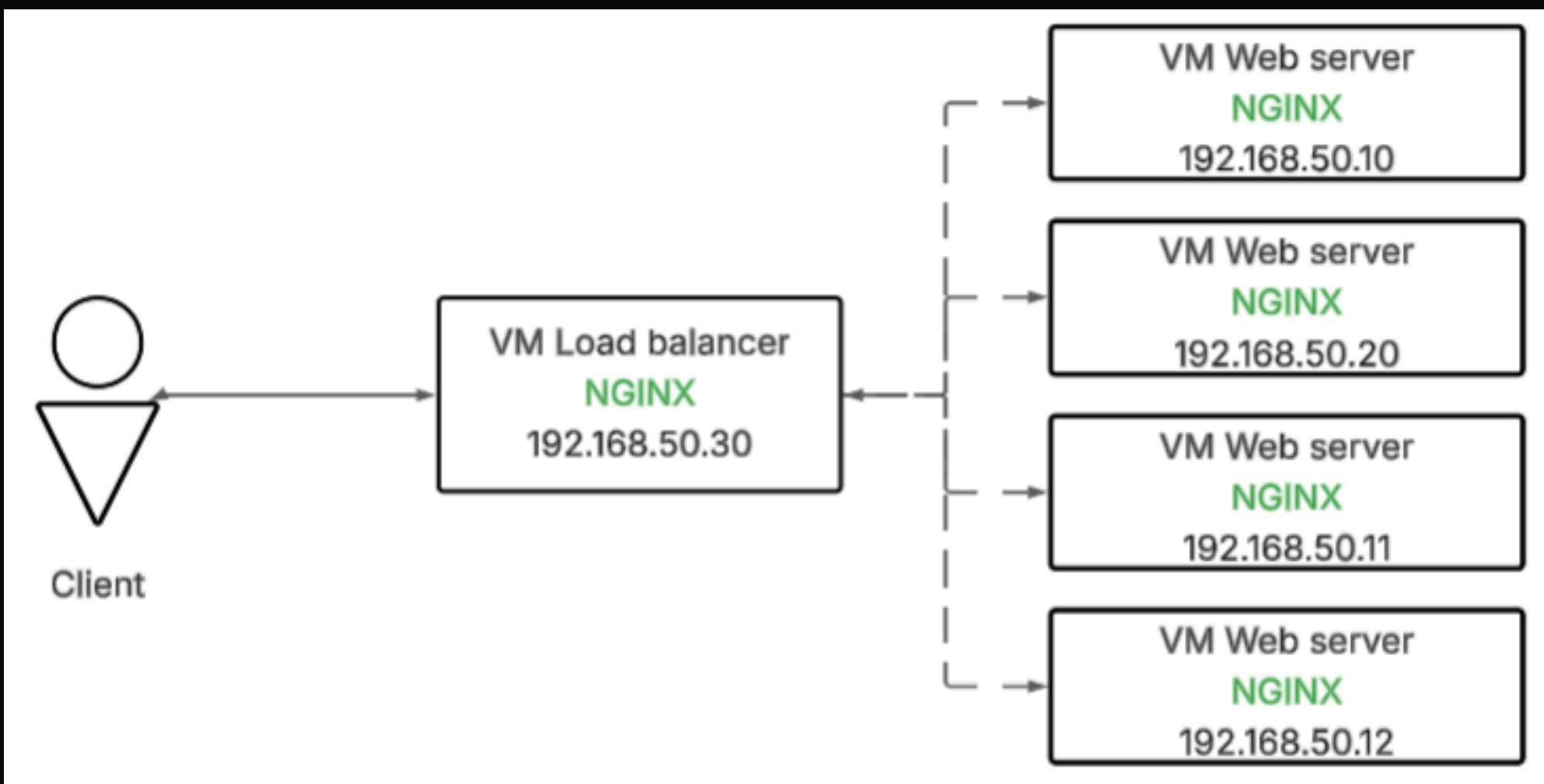
Least Connections (least_conn)

La estrategia Least Connections fue finalmente seleccionada por ser la más eficiente para nuestro escenario. Este algoritmo analiza activamente el número de conexiones actuales en cada servidor backend y dirige nuevas peticiones al que tenga menos carga activa en ese momento.

Diseño de solución

El diseño de solucion consta de 6 maquinas: cliente, balanceador y 4 servidores web.
La maquina cliente se encargara de realizar las peticiones.

El balanceador tendra la configuracion para realizar su trabajo de el manejo de carga.
Por ultimo las 4 servidores restantes tendran las configuraciones para el sitio web.



```
vagrant@cliente:~          X  vagrant@balancer:~          X + 
Use of this system is acceptance of the OS vendor EULA and License Agreements.
Last login: Mon May 19 23:38:10 2025 from 10.0.2.2
vagrant@balancer:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
        inet6 fe80::a00:27ff:fed8:9864 prefixlen 64 scopeid 0x20<link>
        inet6 fd00::a00:27ff:fed8:9864 prefixlen 64 scopeid 0x0<global>
            ether 08:00:27:c8:98:64 txqueuelen 1000 (Ethernet)
            RX packets 930 bytes 107134 (107.1 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 735 bytes 117383 (117.3 KB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.50.30 netmask 255.255.255.0 broadcast 192.168.50.255
        inet6 fe80::a00:27ff:feb7:e7bd prefixlen 64 scopeid 0x20<link>
            ether 08:00:27:bf:e7:bd txqueuelen 1000 (Ethernet)
            RX packets 28810 bytes 28846467 (28.8 MB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 28838 bytes 28149851 (28.1 MB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
            loop txqueuelen 1000 (Local Loopback)
            RX packets 0 bytes 0 (0.0 B)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 0 bytes 0 (0.0 B)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

vagrant@balancer:~$ cat /etc/nginx/nginx.conf
user www-data;
worker_processes auto;
```



22°C 9:33 PM
Mayorm. nublado ESP 5/19/2025

Pruebas

Con el fin de evaluar el rendimiento del balanceador de carga NGINX bajo diferentes algoritmos, se llevaron a cabo múltiples pruebas utilizando Artillery, una herramienta de código abierto para pruebas de carga HTTP. Las pruebas consistieron en el envío de solicitudes concurrentes desde la máquina cliente hacia el balanceador, quien a su vez distribuyó la carga entre los servidores backend.

PRUEBA 1

Round Robin

PRUEBA 3

Least Connection

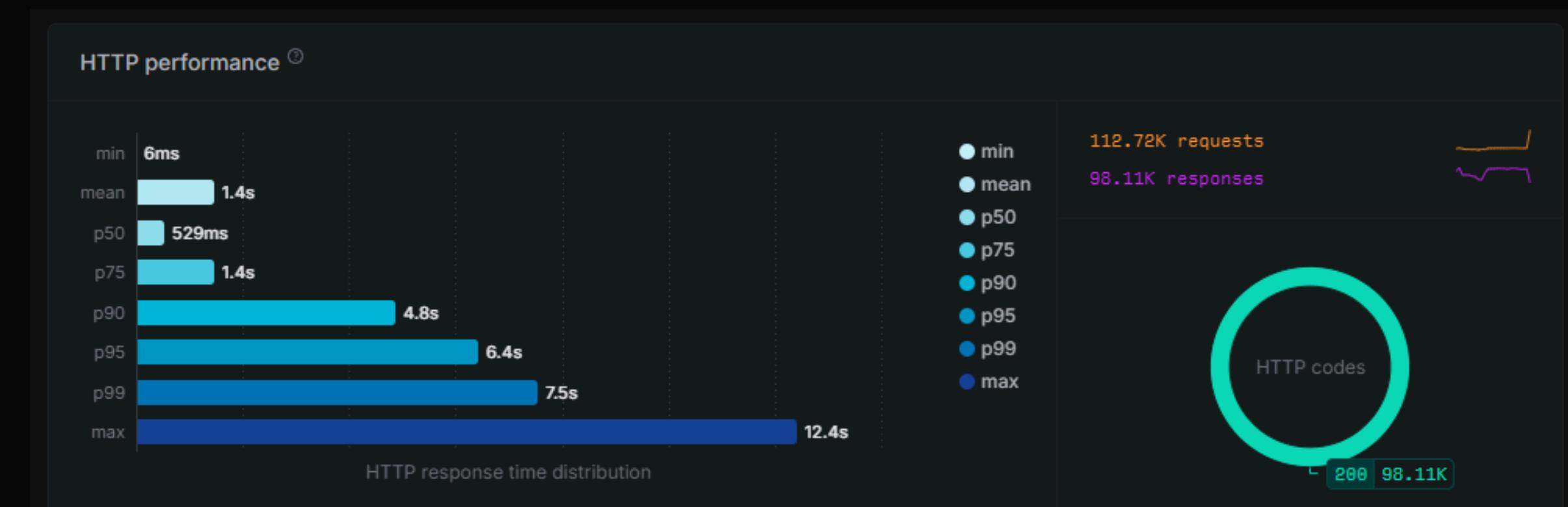
PRUEBA 2

IP Hash

Round Robin

El algoritmo Round Robin evidenció una latencia promedio baja (1.4 segundos) y una alta tasa de éxito 87.06% (98.11K respuestas exitosas de 112.7K solicitudes). Esta eficiencia sugiere que la carga fue distribuida equitativamente entre los servidores backend, lo que resulta adecuado en entornos donde los recursos son homogéneos y el tráfico es constante.

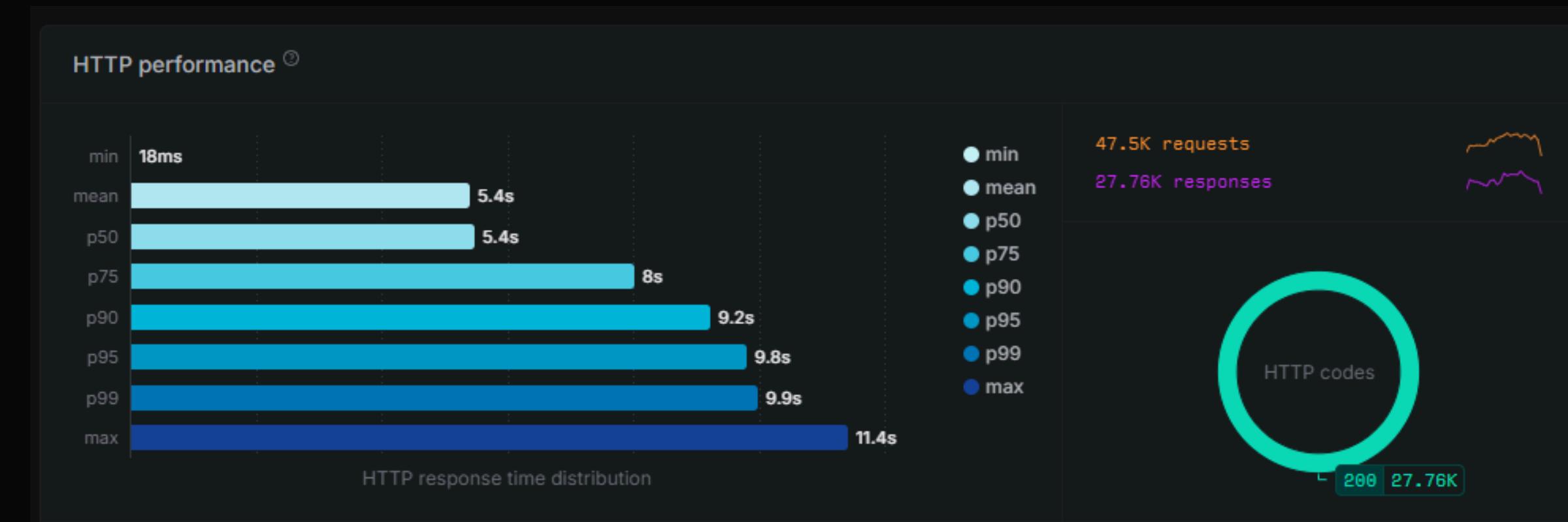
No obstante, el tiempo máximo de respuesta alcanzó los 12.4 segundos, lo que podría estar relacionado con picos transitorios de carga o diferencias mínimas en el rendimiento de los servidores. A pesar de ello, el algoritmo se comportó de forma robusta, confirmando su idoneidad para escenarios básicos de balanceo con baja variabilidad.



IP Hash

La estrategia basada en IP Hash mostró una latencia media elevada (5.4 segundos) y una baja proporción de respuestas exitosas dandonos una tasa de exito del 58.46% (27.76K de 47.5K solicitudes). Este comportamiento se explica por la naturaleza del algoritmo, que asigna siempre la misma IP de origen al mismo servidor backend. Dado que en este entorno de pruebas la totalidad de las solicitudes se originaron desde una única máquina cliente, la carga fue canalizada completamente hacia un único servidor, provocando un claro desbalance.

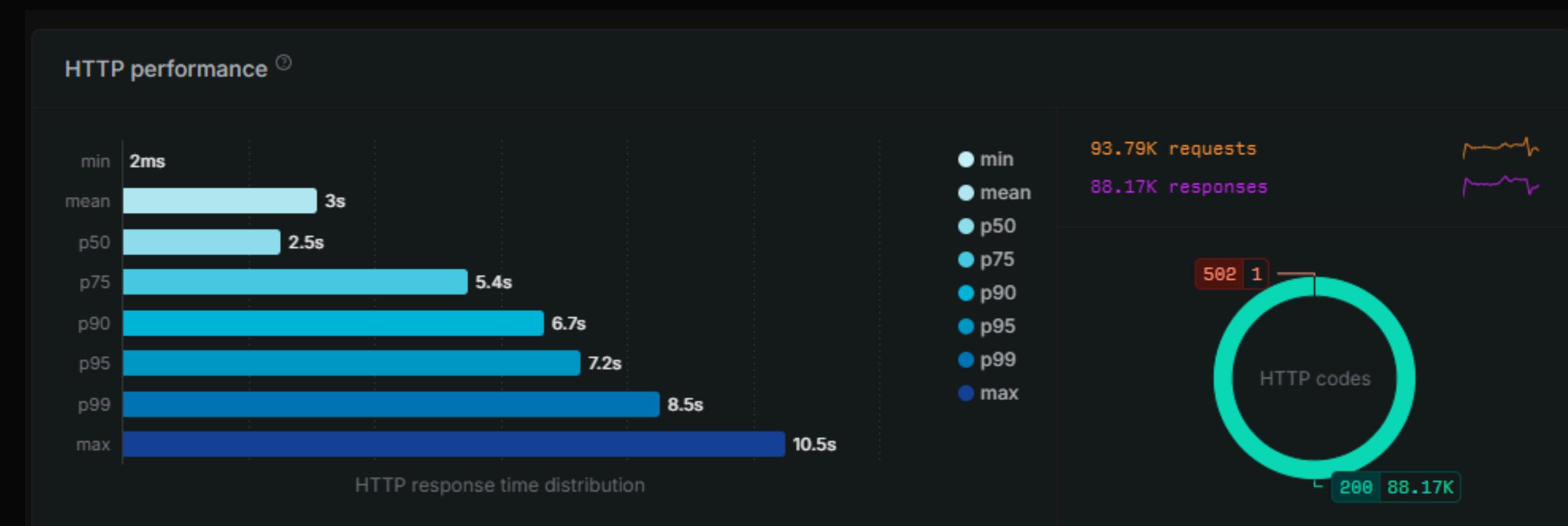
Como resultado, se evidenció **saturación de un solo servidor**, lo que afectó tanto la capacidad de respuesta como la disponibilidad. Este escenario refleja una de las principales limitaciones del algoritmo IP Hash, especialmente en entornos de prueba donde no se simulan múltiples direcciones IP de origen.



Least Connections

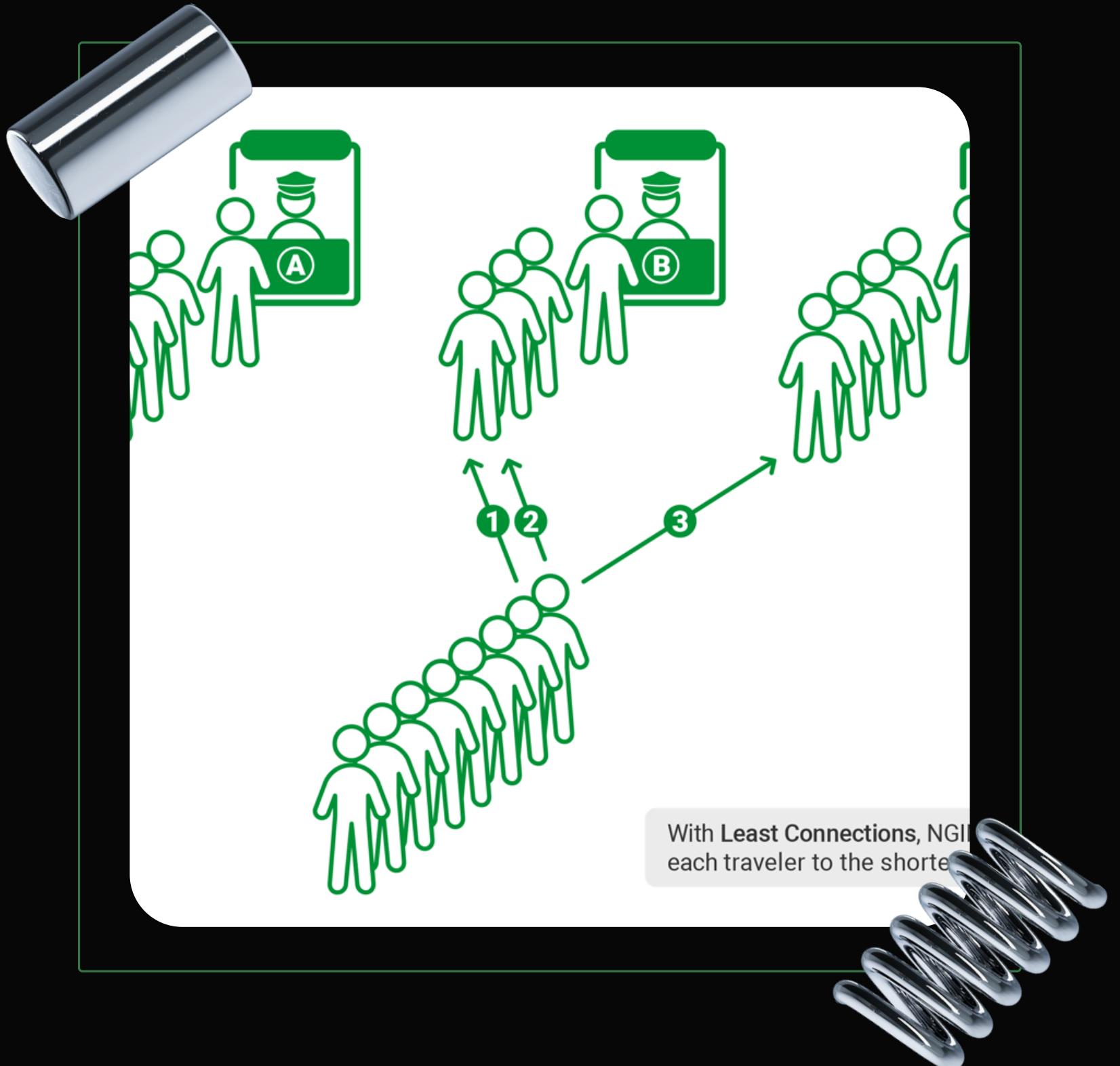
El algoritmo Least Connections presentó un rendimiento equilibrado, con una latencia media de 3 segundos y una tasa de éxito superior al 94% (88.17K de 93.79K solicitudes). Este algoritmo asigna dinámicamente nuevas conexiones al servidor con menor número de conexiones activas, permitiendo una distribución adaptativa de la carga.

Aunque se registró un único error de tipo HTTP 502, el comportamiento general **fue estable y eficiente**, destacándose como la opción más flexible para manejar cargas variables o desbalanceadas. Este algoritmo demostró ser especialmente útil en entornos donde los tiempos de procesamiento entre servidores pueden diferir o donde se espera una carga dinámica.



Conclusiones

El algoritmo de balanceo seleccionado fue **Least Connection**, ya que tras realizar pruebas comparativas con otros métodos, se observó que ofreció el mejor desempeño en la gestión de peticiones generadas por Artillery. A diferencia de algoritmos como Round Robin o IP Hash, Least Connection distribuye las solicitudes según la cantidad actual de conexiones activas en cada servidor, lo que permite una asignación más eficiente de los recursos. Esto se traduce en una menor sobrecarga, mejor tiempo de respuesta y mayor estabilidad del sistema, especialmente en escenarios con tráfico variable o solicitudes de diferente duración.



Muchas gracias

