HTTP multiplexación

Nicolás Reyes

Taller 2 - 10.000 usuarios concurrentes usando HTTP, Ingenieria de sistemas,
Pontificia Universidad Javeriana, Bogotá, Colombia

Abstract: Este documento muestra las pruebas y conclusiones obtenidas luego de ejecutar un servidor que intenta atender la solicitud de 10.000 usuarios de forma concurrente, usando HTTP en el puerto 8080.

1. Introducción

El protocolo HTTP ha ido evolucionando a lo largo del tiempo para cumplir con las necesaidades de la creciente demanda y uso de este protocolo. Actualmente en la versión HTTP/2, el protocolo soporta multiplexación, es decir que se pueden enviar múltiples solicitudes de manera simultánea sobre una única conexión TCP .

2. Problema

Hoy en día, no todos los servidores soportan el protocolo HTTP2, lo cual hace que la multiplexación no esté pérmitida, de forma que el navegador envía una petición al servidor y debe esperar la respuesta para enviar la siguiente solicitud.

3. Protocolo de Pruebas

Se presenta el siguiente plan de pruebas:

3.1. Alcance

Lo que se pretende es medir el tiempo de respuesta, de latencia, bajo la carga de 10.000 usuarios concurrentes:

- Tiempo de respuesta: Tiempo total que tarda una solicitud en ser procesada desde el momento en que se envía hasta el momento en que se recibe la respuesta

3.2. Descripción de las pruebas

Herramienta: Locust
Especificaciones de máquina usada:

- RAM: 32,0 GB
- CPU: 12th Gen Intel(R) Core(TM) i7-12700KF 3.60 GHz

Descripción: Con ayuda de las herramientas que proporciona locust se ajustaron las pruebas de la siguiente forma:

- Número de usuarios: 10.000
- Ramp up (Second): 500

En pocas palabras, esto quiere decir que cada segundo, se incrementan los usuarios concurrentes en 500 hasta llegar a los 10.000.

3.3. Ejecución de Pruebas

Fue necesario ejecutar el servidor primero para que escuchara las solicitudes. Adicionalmente el servidor tiene un contador de solicitudes.

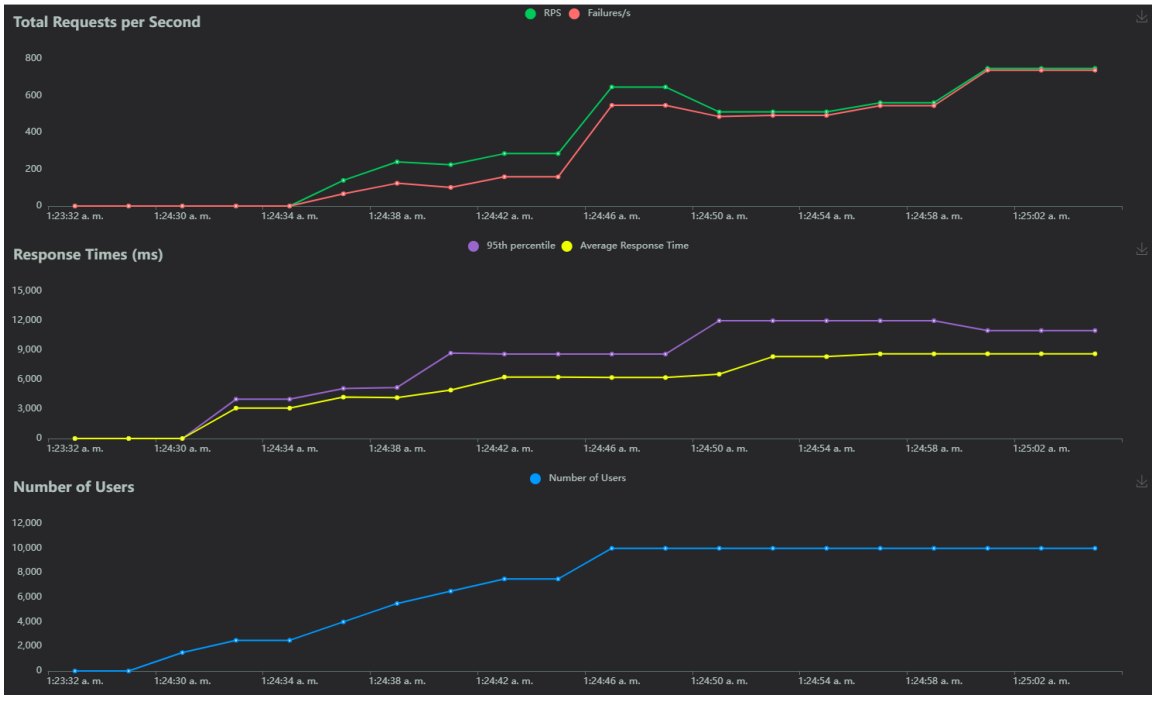

Fig. 1. Métricas

Estas métricas me permitieron percatarme que el servidor no estaba configurado con el protocolo HTTP/2, ya que como se puede evidenciar en la primera gráfica, las peticiones prácticamente se igualan con los fallos, particularmente el siguiente error: ”ConnectionRefusedError(10061, ’[WinError 10061] No se puede establecer una conexión ya que el equipo de destino denegó expresamente dicha conexión.’) ” De esta forma se pudo observar que era necesario actualizar el protocolo HTTP del servidor, puesto que no estaba permitida la multiplexación.

4. Resultados

Una vez obtenidos los datos, se sacó un promedio del tiempo de respuesta para cada técnica de integración, obteniendo los siguientes resultados, representados en una tabla:

REST	Queues	Topics
6147,4 ms	10,73 ms	9,94 ms

TABLE I
Promedio de tiempos de respuesta

Como se ve reflejado en la tabla hay una gran diferencia en REST con respecto a queues y topics, resultados que tienen sentido por la forma en la que funciona el envío y procesamiento de mensajes. En REST hay un flujo síncrono, lo que resulta en un tiempo de respuesta mayor, ya que se envía el mensaje desde ”producer” y se espera a que los otros 3 procesos respondan, a diferencia de las colas y los tópicos que envían los mensajes sin esperar respuesta.

References

[1] N. Ford, M. Richards, Pramod Sadalage, and Z. Dehghani, Software Architecture: The Hard Parts. “O’Reilly Media, Inc.,” 2021.
[2] “REST - MDN Web Docs Glossary: Definitions of Web-related terms | MDN,” developer.mozilla.org. https://developer.mozilla.org/en-US/docs/Glossary/REST