Experimentos concurrentes

Autor: Jeymar Vega Bustos
Email: jeymar.vega@mail.escuelaing.edu.co
Escuela Colombiana de ingeniería Julio Garavito

***Resumen*— Este articulo muestra los resultados de experimentos, enfocados a medir el rendimiento de un servidor concurrente que recibe muchas peticiones simultaneas.**

***Abstract*-- This article shows the results of experiments, focused on measuring the performance of a concurrent server that receives many simultaneous requests.**

I. INTRODUCCIÓN

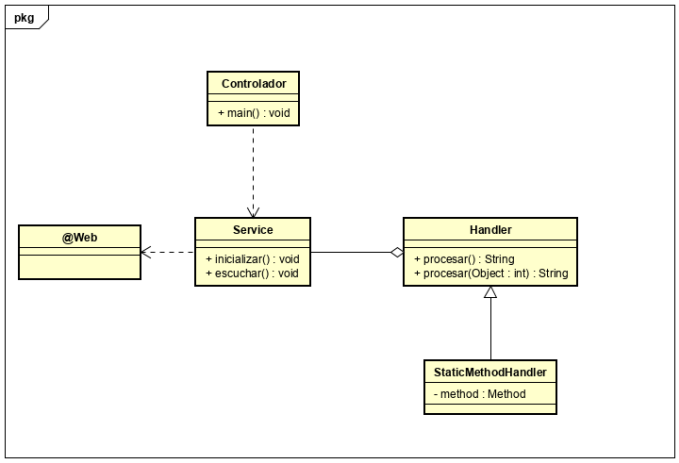
En este experimento se realizo un montaje experimental de servidor web, un servidor web es aquel que haciendo uso del protocolo HTTP (Hypertext Transfer Protocol) para responder a las solicitudes que se le realizan por sus clientes, los cuales generalmente son Browsers, pero no están limitados a ellos.
Este servidor fue desplegado en heroku, contando con una lógica que le permite atender múltiples solicitudes de manera simultanea o concurrente.
En este articulo expondremos resultados de unos experimentos de rendimiento realizados sobre el mismo, para permitir hacer un análisis sobre su comportamiento.

II. CONTEXTO

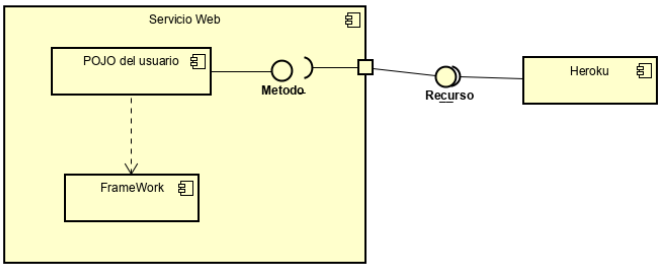
El servidor está diseñado para ser un "Framework" que permita al usuario añadir sus propias clases, con métodos ejecutables siempre y cuando estén marcados con nuestra etiqueta "Web", a la cual se le debe asignar un "Nombre" que hará las veces de URI para el método dentro del despliegue. Es decir, si el método es llamado "test" la URL será "apps/test" y si solicita parámetro la manera de enviarlo será "apps/test?param=prueba".
En este orden de ideas, se debe aclarar que el servidor tiene dos tipos de servicios:

- **Estáticos:** Son archivos almacenados dentro del servidor, es decir, son archivos que se encuentran ahí por "defecto".
- **Dinámicos:** Estos son el resultado de la ejecución de los métodos que hacen uso de nuestro framework. Estos métodos pueden o no solicitar datos del cliente como parámetros. Dichos métodos están directamente ligados a el nombre que haya sido colocado en su etiqueta.

El servidor cuenta con la siguiente arquitectura:



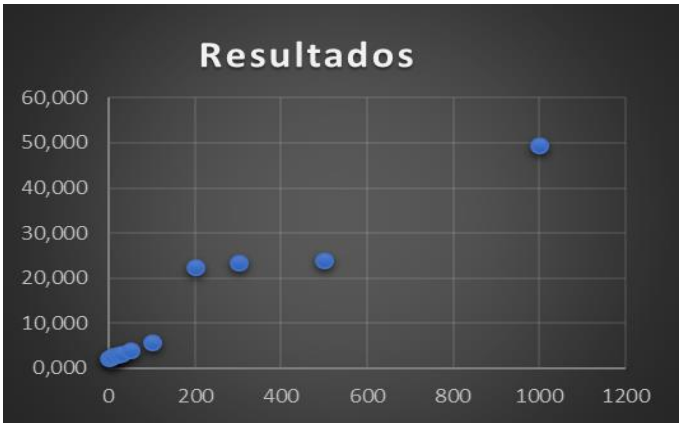
Y esta diseñado para desplegarse de la siguiente manera:



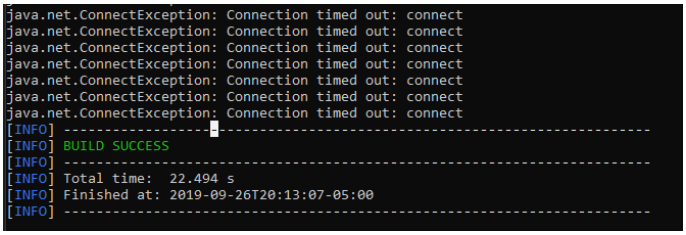
III. PRUEBAS DE CONCURRENCIA

Estas pruebas serán realizadas desde una maquina local, realizando un determinado número de peticiones y viendo su comportamiento en el tiempo si dicha cantidad de peticiones va en aumento. Por comodidad, se utilizará la herramienta Maven para la obtención de los tiempos.

Las pruebas serán realizadas sobre el index del servidor, para confirmar los tiempos de respuesta.



En la gráfica anterior el eje X es el numero de peticiones realizadas y el eje Y resulta ser el tiempo.
Como se puede ver al inicio el crecimiento es relativamente “Lineal”, pues a medida que ibamos aumentando las peticiones aumentaba el tiempo, sin embargo llegando a las 100 peticiones, las cuales representan el limite del pool dentro del servidor, los tiempos aumentaron de manera bastante agresiva.
Por otro lado dichos tiempos se mantuvieron durante las siguientes pruebas, es decir sobre las 200, 300 y 500. Sin embargo, cabe aclarar que sobre dicha cantidad de peticiones se hicieron mas de una prueba, pues algunas veces el servidor era incapaz de responder y tiraba “Conecction time out”.



Así que dichos experimentos se repitieron hasta que el servidor fuese capaz de responder.

Para terminar, en las 1000 peticiones el resultado en tiempo resulto ser relativamente “lo esperado” pues resultaba ser mas o menos el doble del tiempo necesitado por las 500 peticiones. Por otro lado en las 1000 peticiones no se presento el problema de los time out.

IV. CONCLUSIONES

Respecto al rendimiento del servidor, y teniendo en cuenta las configuraciones del mismo, los resultados resultaron ser algo confusos, pues, aunque el número de solicitudes aumento bastante durante algunos tramos, los tiempos resultaron ser prácticamente iguales. Pues la diferencia estaba cerca al segundo por encima o por debajo. Además, luego se demuestra una relativa “normalidad” o un comportamiento esperado, sobre todo con las diferencias entre las 100 y 200 peticiones, y las diferencias sobre las 500 y las 1000, donde pese a que no conociéramos el contexto se podría prever “por lógica” que los tiempos resultarían el doble (en el caso de 500 a 1000 por ejemplo). En definitiva, puedo afirmar que el servidor funciona de la manera esperada, pero que sin embargo no sé si por el comportamiento interno de Heroku, mi internet, mi maquina o algún otro objeto en relación a estas pruebas, algunas veces el comportamiento de las respuestas resulta ser algo confuso. Estas pruebas en general parecen verse afectadas por otros factores aparte del numero de peticiones que es capaz de soportar el servidor de manera simultánea, en este sentido sabemos que puede ser afectado por las latencias entre el servidor y el cliente.