Servidor Web Concurrente en Java y cliente AWS (Septiembre 2019)

Santiago Rocha Durán

I. Introducción

Un servidor Web es un programa que utiliza HTTP (Hypertext Transfer Protocol) para servir los archivos que forman páginas Web a los usuarios, en respuesta a sus solicitudes, que son reenviados por los clientes HTTP. Un servidor web opera en un ordenador aguardando las solicitudes de parte del navegador web de un cliente, brindando los datos solicitados para componer una página web o, en su defecto, un mensaje de error. Los servidores web pueden ser de dos clases:

- **Estáticos:** los archivos se envían tal y como están almacenados.
- **Dinámicos:** Realizan operaciones con los datos antes de enviar respuesta al cliente

Típicamente se usan servidores web como Apache para el desarrollo de aplicaciones más grande pero en esta ocasión se va a describir el desarrollo de un servidor web en java.

II. Arquitectura de Software

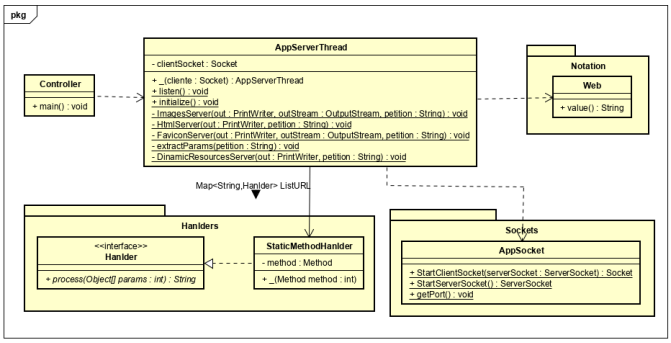

Fig. 1. Diagrama que describe la arquitectura del proyecto.

A continuación se detalla la arquitectura del servidor web, esta arquitectura busca el bajo acoplamiento de sus componentes con el objetivo de continuar agregando características y funcionalidades a este. El servidor actualmente funciona con un modelo de ThreadPool, con un limite flexible de hilos, es decir que a medida que el servidor vaya necesitando responder a más peticiones irá creando hilos, sin embargo si hay hilos ya creados los re utilizara siempre y cuando estén libres

- **Hanlder:** El objetivo de este paquete es tener una interfaz que permita manejar e invocar los métodos de los POJOS que el usuario cargue sin necesidad de conocer concretamente el tipo o sus parámetros
- **Notation:** El objetivo de este paquete es crear anotaciones sobre el código que le permitan al usuario poner el nombre por el cual se va a buscar los recursos dinámicos que desea cargar
- **Sockets:** El objetivo de este paquete es iniciar los sockets tanto de cliente como servidor para comenzar a escuchar las peticiones del browser o de cualquier otro cliente por un puerto definido
- **Threads:** El objetivo de este paquete es separar en una capa el servicio que funciona con el pool de hilos, es decir que allí se encuentra toda la lógica para que el servidor sea concurrente

El usuario del servidor web debe cargar sus archivos en la carpeta app con el fin de que sean encontrados y ejecutados por el servidor

Finalmente el flujo del programa seria el siguiente:

1) El cliente envía una petición que es recibida por los sockets
2) ThreadPool tiene el socket en el cual viene la petición, al recibirla delega la ejecución al pool de hilos que decide si crear un nuevo hilo o usar uno ya existente.
3) AppServer recibe el socket cliente con la petición, revisa que la petición este bien formada y selecciona el recurso solicitado
4) En caso de ser un recurso estático lo devuelve con el encabezado apropiado de los contrario delega el trabajo al Hanlder para que maneje los parámetros y la respuesta
5) El recurso es enviado como respuesta al cliente y mostrado en el browser

III. Arquitectura de Despliegue

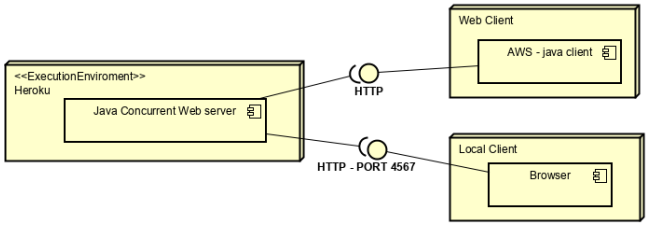

Fig. 2. Diagrama que describe como se despliega el proyecto.

IV. Pruebas de Concurrencia

Las pruebas que se presentan a continuación son realizadas a un servidor hosteado en heroku con un pool de hilos flexible, como se describe anteriormente en este documento.

Se realizaron dos pruebas, una con un cliente local y otra con un cliente en AWS. Las pruebas constan de realizar N numero de peticiones concurrentes al servidor y calcular su tiempo de respuesta. Los resultados son descritos a continuación

A. Cliente Local

```
estudiante@linux-12:~/Documents/AWS-TALLER/ClientAWStoHeroku/dist$ java -jar ClientAWStoHeroku.jar 10 https://arem-lab4.herokuapp.com/
0.7423180608
estudiante@linux-12:~/Documents/AWS-TALLER/ClientAWStoHeroku/dist$ java -jar ClientAWStoHeroku.jar 50 https://arem-lab4.herokuapp.com/
0.7510739131799998
estudiante@linux-12:~/Documents/AWS-TALLER/ClientAWStoHeroku/dist$ java -jar ClientAWStoHeroku.jar 100 https://arem-lab4.herokuapp.com/
1.3618054352000002
estudiante@linux-12:~/Documents/AWS-TALLER/ClientAWStoHeroku/dist$ java -jar ClientAWStoHeroku.jar 500 https://arem-lab4.herokuapp.com/
6.6043104300760008
estudiante@linux-12:~/Documents/AWS-TALLER/ClientAWStoHeroku/dist$ java -jar ClientAWStoHeroku.jar 1000 https://arem-lab4.herokuapp.com/
9.031750505450992
estudiante@linux-12:~/Documents/AWS-TALLER/ClientAWStoHeroku/dist$
```

Fig. 3. Resultado de las pruebas utilizando un cliente local.

Con los tiempos tomados en esta prueba se realiza una gráfica para dimensionar en función de que crece el tiempo a medida que se aumentan las peticiones

Fig. 4. Gráfica peticiones/tiempo utilizando cliente local.

B. Cliente AWS

Con los tiempos tomados en esta prueba se realiza una gráfica para dimensionar en función de que crece el tiempo a medida que se aumentan las peticiones

```
[ec2-user@ip-172-31-31-180 ~]$ java -jar ClientAWStoHeroku.jar 10 https://arem-lab4.herokuapp.com/
1.0824847011
[ec2-user@ip-172-31-31-180 ~]$ java -jar ClientAWStoHeroku.jar 50 https://arem-lab4.herokuapp.com/
2.296379679712
[ec2-user@ip-172-31-31-180 ~]$ java -jar ClientAWStoHeroku.jar 100 https://arem-lab4.herokuapp.com/
5.0911157634
[ec2-user@ip-172-31-31-180 ~]$ java -jar ClientAWStoHeroku.jar 500 https://arem-lab4.herokuapp.com/
28.897374689148
[ec2-user@ip-172-31-31-180 ~]$ java -jar ClientAWStoHeroku.jar 1000 https://arem-lab4.herokuapp.com/
31.242680503679992
[ec2-user@ip-172-31-31-180 ~]$
```

Fig. 5. Resultado de las pruebas utilizando un cliente en AWS.

Una vez concluidas las pruebas podemos observar que hay una notable diferencia entre los tiempo de respuesta para un cliente local y para un cliente en AWS, sin embargo el patron de crecimiento del tiempo es bastante similar.

Fig. 6. Gráfica peticiones/tiempo utilizando cliente en AWS.

V. Conclusión

El uso de POJOS y anotaciones permiten estructurar mejor el uso y administración de un servidor web a nivel de código. Adicionalmente le brinda al usuario una guía para manejar los recursos que quiere cargar al servidor y como dicho servidor esta esperando que estos sean cargados para que el flujo del software sea correcto y la salida sea la esperada por el usuario.

La respuesta a un cliente, no solo depende de la cantidad de hilos que pueda manejar un servidor, también se ve afectado por otras factores como la latencia entre el servidor y la ubicación del cliente, las características hardware del cliente, entre otras.

References

[1] María Estela Raffino. (2019). Servidor Web. 06 de septiembre del 2019, de Conceptos Sitio web: https://concepto.de/servidor-web/

[2] osgroup. Que es un servidor web. 06 de septiembre del 2019, de osgroup Sitio web: https://www.osgroup.co/que-es-un-servidor-web/