Texto

Descripción generada automáticamente con confianza media

Instituto Tecnológico de Costa Rica  
Campus Tecnológico Central Cartago  
Escuela de Ingeniería en Computación

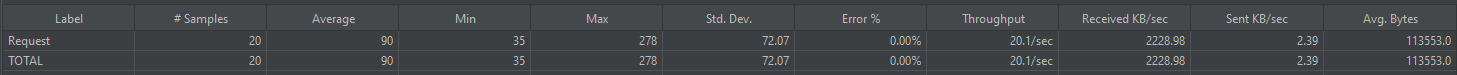
**Bases de datos I.**  
Prof. Rodrigo Núñez  
Semestre I  
Alonso Josué Navarro Carrillo, c.2022236435  
Dominic Casares Aguirre, c. 2022085016  
Fecha de entrega: Lunes 15 de mayo del 2023

Análisis de pruebas de stress implementando endpoints.

Endpoint pool

Interfaz de usuario gráfica

Descripción generada automáticamente

Lo primero que notamos es que el endpoint con pool ejecuta un aproximada de 6 solicitudes de lotes por segundo. También es importante resaltar que esas 5 conexiones que se muestran se mantuvieron durante todas las pruebas, hasta que se finalizara la ejecución del API.

Notamos luego que el tiempo promedio de respuesta según el summary report de JMeter es de 90ms, aunque su desviación estándar es relativamente alta por lo que tenemos medidas con mucha variabilidad respecto al promedio. Importante recalcar un valor máximo de 278ms y un mínimo de 35ms.

Interfaz de usuario gráfica, Texto

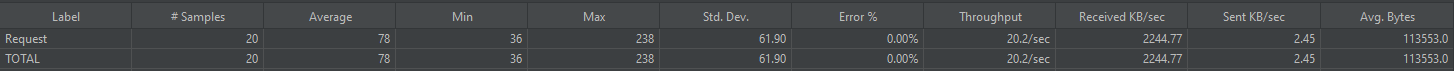
Descripción generada automáticamenteInterfaz de usuario gráfica, Texto

Descripción generada automáticamenteEn la imagen de la izquierda vemos que el valor máximo se encuentra en el thread 1. Posiblemente, por el hecho de tener que crear las 5 conexiones mínimas del pool apenas se envía el request. En el reporte de la derecha vemos un valor realmente mínimo en uno de los últimos threads; entendemos que aquí ya se tienen las conexiones necesarias y el proceso es más directo.

Endpoint no pool

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Al ejecutar desde endpoint sin pool, vemos una mejoría de 2,5 solicitudes de lote más que lo que ejecutaba el endpoint con pool por segundo. Todas las conexiones abiertas por el endpoint sin pool fueron cerradas poco tiempo después de devolver los datos.

Vemos que el promedio de ms de respuesta es 12 puntos menor que el endpoint pool. Además, la desviación estándar también se reduce. El valor máximo es menor, pero su valor mínimo es un punto mayor.

Texto

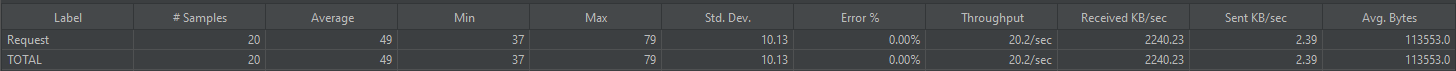
Descripción generada automáticamente con confianza mediaTexto

Descripción generada automáticamenteEn la imagen de la izquierda volvemos a ver al mayor tiempo en el thread 1. Pero ahora el menor tiempo se encuentra en el thread 15 y su comportamiento en los últimos threads, aunque es bajo, no es tan marcado como el endpoint pool.

Endpoint ORM

Interfaz de usuario gráfica

Descripción generada automáticamente

Por último, el endpoint ORM denota una gran potencia al poder ejecutar más de 10 solicitudes de lotes por segundo. Un rendimiento bastante superior a los endpoints anteriores.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto

Descripción generada automáticamenteVemos que su promedio de tiempo es de 49ms y que su desviación estándar es sumamente baja. Su valor máximo es 79ms y su mínimo de 37ms. Se sigue el patrón de que el primer thread sea el que dure más y que los últimos sea más rápidos.

Análisis General

Concluimos que el endpoint ORM tiene un rendimiento realmente eficiente comparado con los demás. Inferimos que esto ocurre por que el ORM implementa optimizaciones internas como usos de caché para las consultas y estrategias de carga que mejoran el rendimiento y la cantidad de consultas que se le solicitan a la base de datos.

Aunque en internet se encuentra que los endpoints con conexiones pool suelen ser más eficiente, esto solo se puede detectar de mejor manera en ambientes con alto tráfico. En ambientes limitados, como es nuestro caso de máximo 20 threads, tenemos que los endpoints sin conexiones pool son más eficientes, pues en realidad no hay una sobrecarga significativa que afecte el rendimiento del sistema.