Steven Tarazona Bernal

201617841

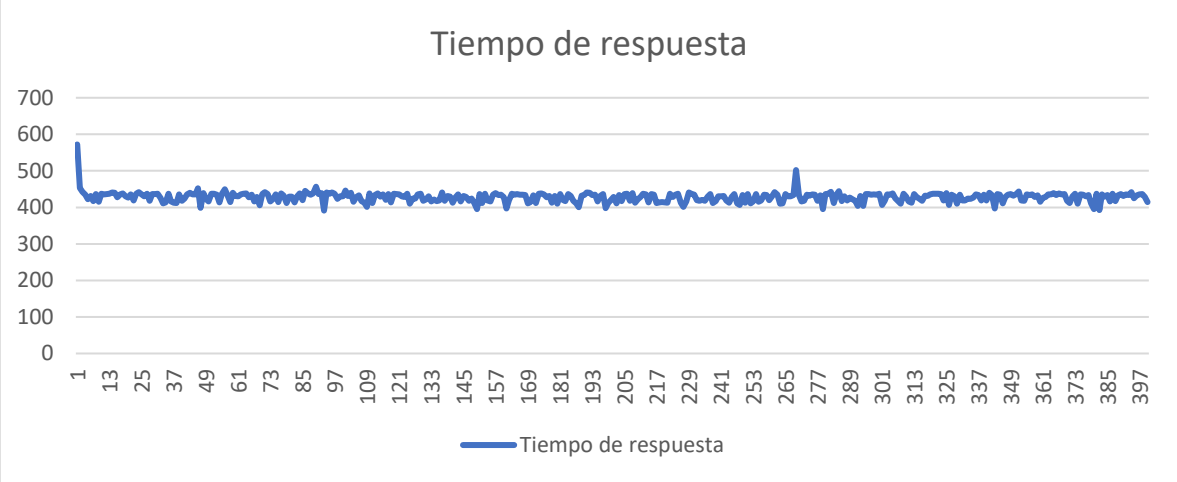
Caso 3

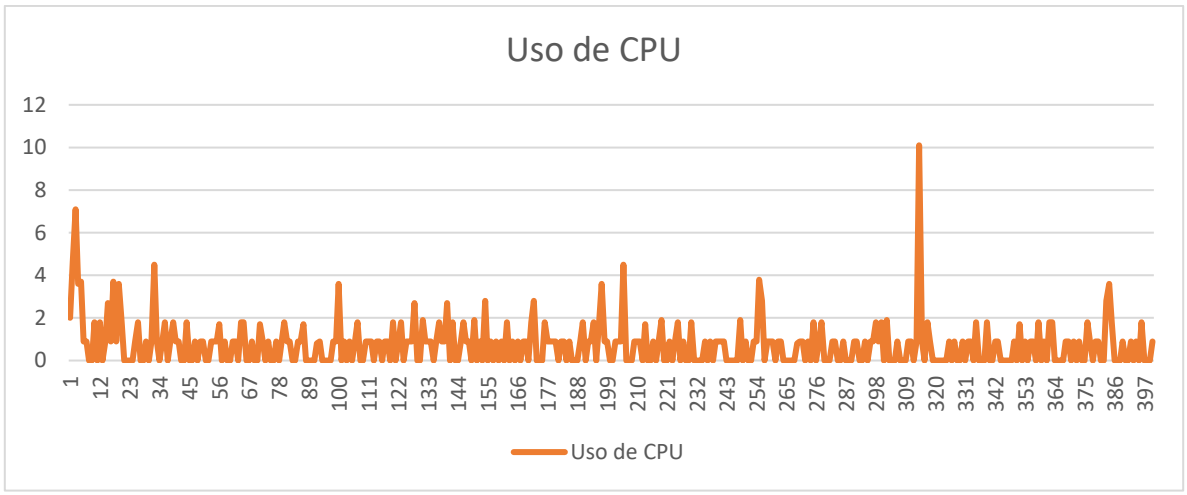
a) Los monitores se implementaron de la siguiente forma:
Para calcular el tiempo de respuesta de una transacción de cada transacción se gizo uso del método System.nanoTime(), tomando así el tiempo del sistema antes de tomar la lectura de la llave simétrica y nuevamente al escribir en el socket el HMAC del valor, el tiempo final se calculo a partir de la resta.
Para calcular el uso de CPU se hizo uso del método proporcionado en el enunciado getSystemCpuLoad().
El número de transacciones perdidas se calculo por medio de un contador al cual se le iba sumando cada vez que ocurría una excepción y la transacción no finalizaba correctamente.
Todos estos datos se recopilaron en un archivo csv para posterior visualización y análisis de los datos.

b) Identificación de la plataforma:
Arquitectura: 64 bits
Numero de núcleos: 1
Velocidad de procesador: 2.0 GHz
Tamaño de la memoria RAM: 3,75 GB
Espacio de memoria asignado a la JVM:
InitialHeapSize= 60817408
MaxHeapSize=968884224

c)

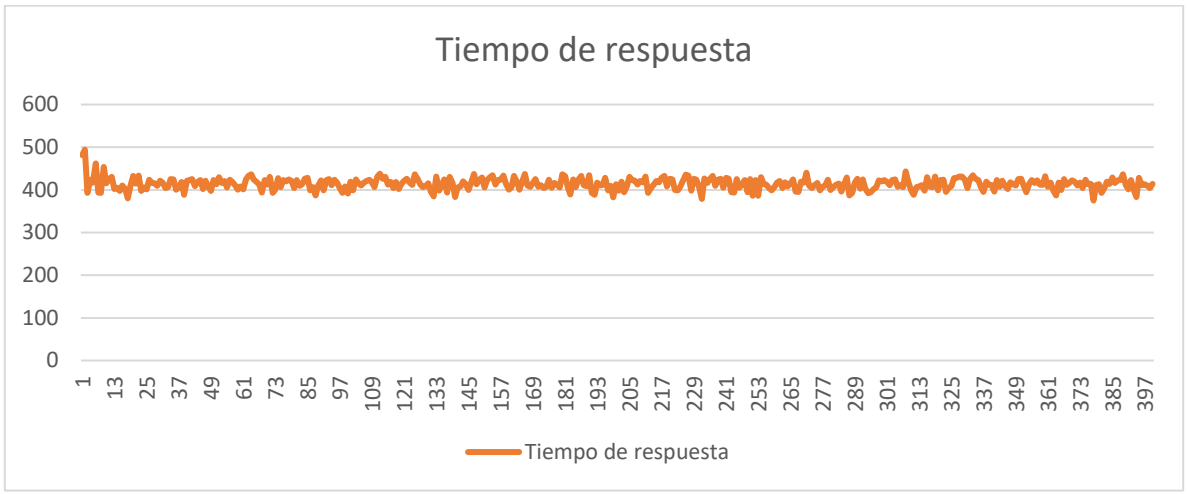
Escenario1.
Threads:1
Carga:400
Retardo:20

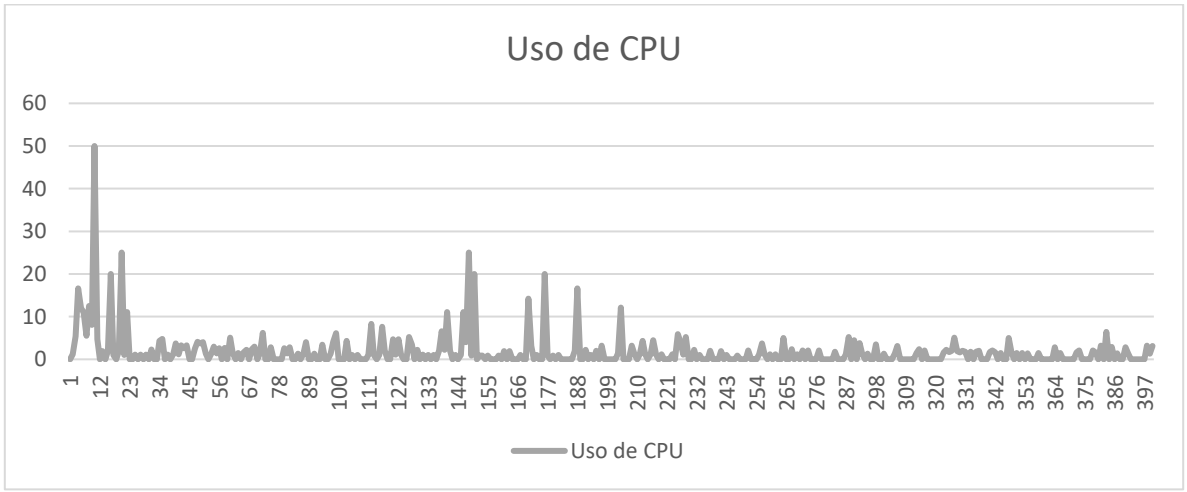




Transacciones perdidas: 0

Escenario2.
Threads:2
Carga:400
Retardo:20





Transacciones perdidas: 0

Ya que la toma de estos datos tomo mucho tiempo, no me fue posible consolidar todos los datos y graficarlos, pero los datos tomados se pueden encontrar en la carpeta /monitor dentro la carpeta del servidor.

En cuanto tiempo de respuesta se puede ver que no se ve realmente afectado por el cambio en el numero de threads, por otro lado se ve un gran aumento en el uso de CPU al aumentar el numero de threads.