****

Instituto Tecnológico Autónomo de México

Sistemas Distribuidos

Proyecto Alpha

**¡PÉGALE AL MONSTRUO!**

Elaborado por:

Dan Jinich - 167907

Pablo Martínez - 149952

Larissa Trasviña -166280

**Profesor: José Octavio Gutiérrez García**

Entrega: 17 de marzo de 2021

## Introducción

Los estresadores son una forma de pruebas deliberadamente intensas o exhaustivas que se utilizan para determinar la estabilidad de un sistema, una infraestructura crítica o una entidad determinada. En particular, los estresadores permiten observar el tiempo de respuesta de los servidores dependiendo de los clientes que acceden y las peticiones que se le hacen. Con los estresadores se puede ir cambiando las características y configuraciones para saber la carga de trabajo que puede soportar el programa.

Estas pruebas ponen un mayor énfasis en la robustez, la disponibilidad y el manejo de errores bajo una carga pesada, en lugar de lo que se consideraría un comportamiento correcto en circunstancias normales. [1]

## Desarrollo

La estructura del proyecto está dividida en cuatro grandes partes, de la siguiente manera:

* Cliente
* Estresador
* Interfaces
* Servidor

Esta división nos permite diferenciar concretamente las clases que fueron implementadas.

El manual de replicación se presenta al final de este documento.

### Pruebas de estrés

Las pruebas que a continuación se describen fueron centradas en el servidor e implementadas con un procesador AMD Ryzen 7-3700-U @2.3 GHz con 4 procesadores principales y 8 lógicos. El IDE utilizado fue InteliJ IDEA Community edition 2020.3.1.

Es importante mencionar que estas fueron realizadas en bloques de 50 jugadores hasta alcanzar un máximo de 1000 (50, 100, 150, …, 1000), en particular, se realizaron diez pruebas para cada población. Resultó sencillo medir el tiempo de respuesta del servidor al enviar un mensaje con Sockets, debido a que la función **recieve**() es bloqueante. Otra de las observaciones destacables es que se creó un cliente sin la implementación de su interfaz, esto con la finalidad de facilitar las pruebas del estresador.

El tiempo se midió desde el momento en que el jugador hace clic en el monstruo correcto y el juego le avisa al jugador los puntos que ha acumulado. En el proyecto implementado es el cliente quien está a cargo de revisar si el jugador le pegó al monstruo correcto, en dado caso, este manda una solicitud al Servidor para realizar la suma de los puntos.

El tiempo que tardaba el servidor en procesar un *golpe* se midió con **readUTF** de Sockets para el Cliente y en el lado del servidor con un **WriteUTF**.

Uno de los retos con los que nos encontramos fue el siguiente:

|  |
| --- |
| Socket: maximum number **of** datagramsockets reached |

por lo que se pudo notar que java tiene solo 25 sockets activos. Para resolver este problema se limitó el número de jugadores activos al mismo.

El estresamiento terminó con 1000 jugadores.

## Resultados

Una de las primeras observaciones notorias fue el enorme incremento en el tiempo de respuesta del servidor conforme aumentaba el número de usuarios. Esto lo podemos observar en la figura 1, donde la línea es la media y la zona es la desviación estándar.

Podemos ver una clara correlación positiva entre el número de jugadores y el tiempo. Sin embargo, es curioso el caso de la zona 400-600 ya que se puede notar una caída del tiempo, al igual que en la zona 600-800 donde la caída es aún más notoria. Este patrón parece repetirse, aunque no de la misma magnitud, en la zona 800-1000 y en zonas anteriores. Lo anterior puede deberse a diversos factores que se desconocen, ya que se realizaron pruebas en diferentes momentos y la gráfica parecía comportarse de la misma manera.

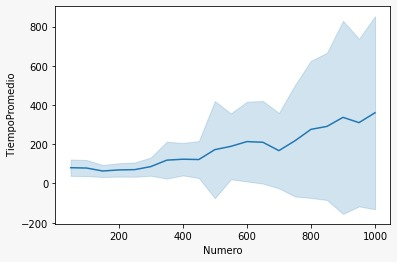


Figura 1. Promedio y desviación estándar del número de jugadores contra el tiempo.

En la Figura 2 podemos observar la distribución al tener 100 usuarios la gráfica se distribuye, en su mayoría, normal sesgada ligeramente a la izquierda, y con un comportamiento un tanto extraño en la cola. Esto pudo deberse a diversos factores como pudieron ser las caídas observadas en la gráfica anterior. Inclusive podría verse la distribución como una normal logarítmica.

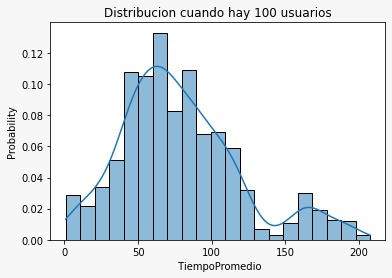


Figura 2. Distribución de 100 usuarios

Al observar la Figura 3 podemos notar como el comportamiento en la gráfica continua como una normal logarítmica, o normal sesgada, sin embargo, este sesgo es menos notorio que en la gráfica anterior. Esto debido al gran incremento en el número de usuarios y por consecuencia en el tiempo. Esto es coherente con los resultados obtenidos al realizar las pruebas.

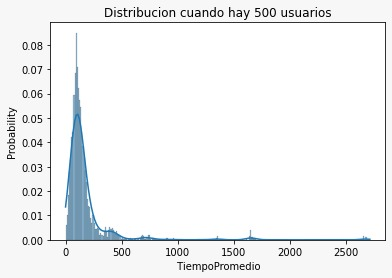


Figura 3. Distribución de 500 usuarios

En la Figura 4 se puede observar claramente un comportamiento similar al de la Figura 3. El sesgo continúa firmemente hacia la izquierda en esta figura. Se puede notar como la probabilidad aumenta ligeramente en la zona 1500-2000 para luego seguir con su tendencia decreciente.

Esto es coherente con las observaciones en las figuras anteriores.

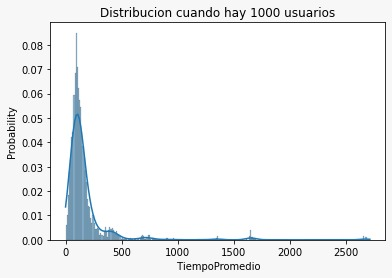


Figura 4. Distribución de 1000 usuarios

*Nota: Las gráficas fueron realizadas en Python, y se anexan en la carpeta del proyecto.*

## Conclusiones

Los resultados presentados anteriormente son consistentes con la respuesta esperada, a mayor cantidad de usuarios, mayor cantidad de tiempo de respuesta.

Algunos de los retos en la realización del proyecto fue el recordar la manera de realizar una interfaz gráfica en Java completamente manual, aprender a manejar los desbordamientos de UDP, así como el gran problema del número máximo de Sockets activos de Java.

## Manual de usuario

Para ejecutar el programa se debe cambiar la dirección de los archivos .policy en las clases login, Administrador y en el ClienteEstresado.

Para el estresador se debe elegir el número de usuarios que se desea en el entrenamiento, este se encuentra en la clase ClienteEstresado.java. Se debe ejecutar primeramente el Administrador y posteriormente el ClienteEstresado.java.