Proyecto Alpha

Sistemas Distribuidos
23 de Marzo de 2020

Integrantes

Diego Villalvazo 155844
Mariella Revah 167125
Francisco Aramburu 170920

Estresamiento

La estrategia de estresamiento para nuestro proyecto fue únicamente orientada al servidor. Hicimos pruebas para 20, 50, 100, 200, 300, 400, 500, 600 y 750 jugadores con el fin de medir el tiempo que el servidor se tardaba en contestar a los clientes. Cada prueba se realizó 10 veces para disminuir los datos atípicos al máximo.

El tiempo medido fue aquel que transcurre entre que el usuario le hace click al topo correcto y el usuario recibe la respuesta del servidor indicando que ya hizo un ajuste en los puntos. En nuestro proyecto, es el cliente el que checa si el usuario hizo click al topo correcto, y de ser así manda una solicitud al servidor para aumentar los puntos de dicho jugador. En el proyecto original, el servidor no le contestaba al cliente después de hacer el cambio, pero para fines del estresador se modificó la sección de mensajes vía Sockets del servidor para que envíe una respuesta al cliente después de modificar los puntos en el método sincronizado que usan los hilos. Se midió el tiempo de respuesta del servidor ante un envío de mensaje con Sockets, esto fue fácil ya que el receive() es bloqueante en éste protocolo de comunicación.

Otro cambio que tuvimos que hacer fue crear un cliente sin interfaz gráfica, pues ésta complicaba el estresamiento del servidor. El único inconveniente de quitar la interfaz gráfica fue que el cliente checaba si el jugador había hecho click en el botón correcto mediante una comparación de colores de la interfaz gráfica. Una vez eliminada la interfaz gráfica se hizo un aleatorio entre 1 y 3 simulando que el jugador le atina al 33% de los topos que le aparecen.
Al principio intentamos hacer el estresador con todo y las conexiones al servicio multicast que anuncia los topos que deben aparecer. Sin embargo, tuvimos que quitarlas pues Java tiene un máximo de 25 sockets activos, por lo que éste era nuestro límite de jugadores soportados.

Las pruebas se realizaron en una computadora con procesador Intel i5-6300U @ 2.40GHz con 2 Cores y 4 Procesadores Lógicos en el IDE NetBeans 8.1 con JVM 1.8.0_72.

Cada cliente tiene un sleep de 10ms dentro del while. En este “modo de juego” no hay un ganador sino que todos los clientes están vivos hasta que mandan 5 golpes. Para medir el tiempo que tarda el servidor en procesar un golpe() se agregó un readUTF de Sockets del lado del cliente y un writeUTF del lado del servidor. El read del cliente es bloqueante entonces cuando recibe algo es porque el servidor ya acabó de procesar su solicitud. Cada cliente imprimía el promedio de tiempo de sus 5 solicitudes.

La prueba se finalizó con 750 jugadores pues el servidor ya no los aguantaba y nos mandaba un error, el cual se muestra en la Figura 1.

```
Mar 22, 2020 11:47:51 PM Estresador.Clente golpe
SEVERE: null
java.net.ConnectException: Connection refused: connect
        at java.net.DualStackPlainSocketImpl.connect0(Native Method)
382.0
342.0
        at java.net.DualStackPlainSocketImpl.socketConnect(DualStackPlainSocketImpl.java:79)
        at java.net.AbstractPlainSocketImpl.doConnect(AbstractPlainSocketImpl.java:350)
        at java.net.AbstractPlainSocketImpl.connectToAddress(AbstractPlainSocketImpl.java:206)
        at java.net.AbstractPlainSocketImpl.connect(AbstractPlainSocketImpl.java:188)
        at java.net.PlainSocketImpl.connect(PlainSocketImpl.java:172)
        at java.net.SocksSocketImpl.connect(SocksSocketImpl.java:392)
        at java.net.Socket.connect(Socket.java:589)
        at java.net.Socket.connect(Socket.java:538)
385.0
        at java.net.Socket.<init>(Socket.java:434)
        at java.net.Socket.<init>(Socket.java:211)
        at Estresador.Clente.golpe(Clente.java:127)
        at Estresador.Clente.run(Clente.java:100)

Mar 22, 2020 11:47:51 PM Estresador.Clente golpe
SEVERE: null
java.net.ConnectException: Connection refused: connect
        at java.net.DualStackPlainSocketImpl.connect0(Native Method)
        at java.net.DualStackPlainSocketImpl.socketConnect(DualStackPlainSocketImpl.java:79)
        at java.net.AbstractPlainSocketImpl.doConnect(AbstractPlainSocketImpl.java:350)
        at java.net.AbstractPlainSocketImpl.connectToAddress(AbstractPlainSocketImpl.java:206)
        at java.net.AbstractPlainSocketImpl.connect(AbstractPlainSocketImpl.java:188)
```

Figura 1. Mensaje de error con 750 clientes conectados al servidor.

Resultados y análisis

Encontramos que conforme incrementamos el número de usuarios, incrementó el tiempo de respuesta del servidor. En la Figura 2 se muestra tanto el promedio como la desviación estándar para el promedio de corridas para cada diferente N. Se encontró que tanto el promedio como la desviación estándar aumentaron al incrementar el número de usuarios concurrentes. El sistema se comportó como era esperado.

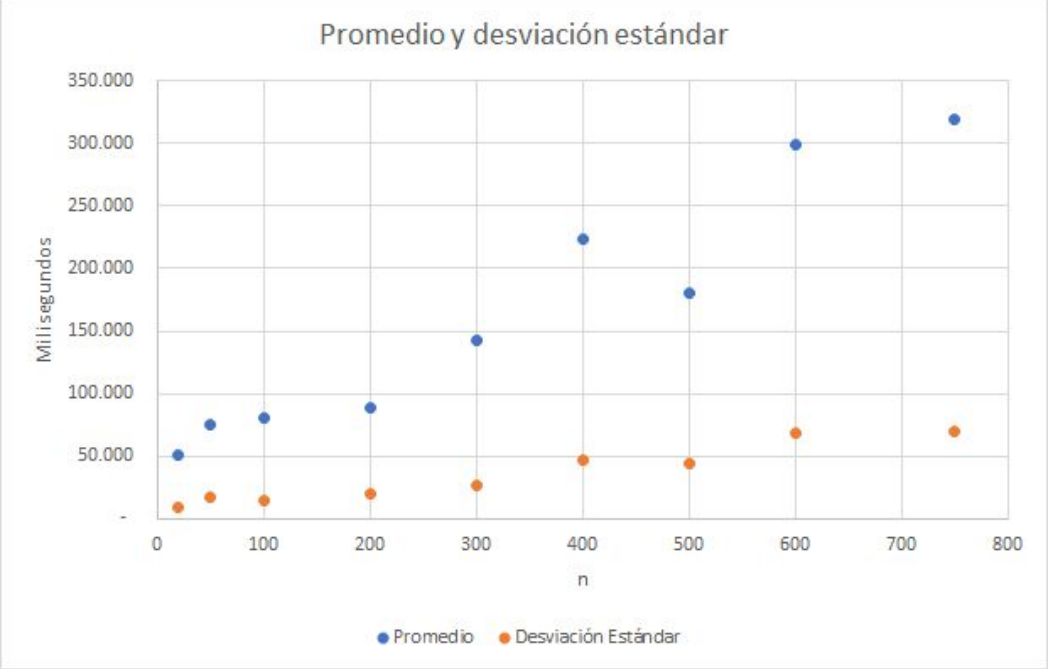

Figura 2. Gráfica de promedio y desviación estándar vs número de clientes.

Podemos observar que el único punto que no sigue la tendencia es en N=500. Esto puede deberse a que el experimento se hizo en un día diferente. Intentamos que las situaciones fueran idénticas a la de los demás experimentos, pero pudo haber variables que no pudimos controlar que hayan podido afectar el tiempo de respuesta del servidor.

Nuestros resultados son consistentes con lo que pensábamos que iba a pasar, y tiene mucho sentido que conforme aumenta el número de clientes que atiende un servidor, también aumenta el tiempo de respuesta.

Para obtener los promedios y desviaciones estándar de los tiempos de respuesta, hicimos cada experimento 10 veces para poder obtener resultados confiables. Encontramos que nuestros resultados son consistentes. Por ejemplo, la mayoría de nuestros resultados se parecen a la gráfica de la Figura 3. Sólo encontramos una anomalía mostrada en la Figura 4. Se pueden consultar todas las gráficas y todos los resultados en el archivo *Graficas_ProyectoAlpha.xlsx*.

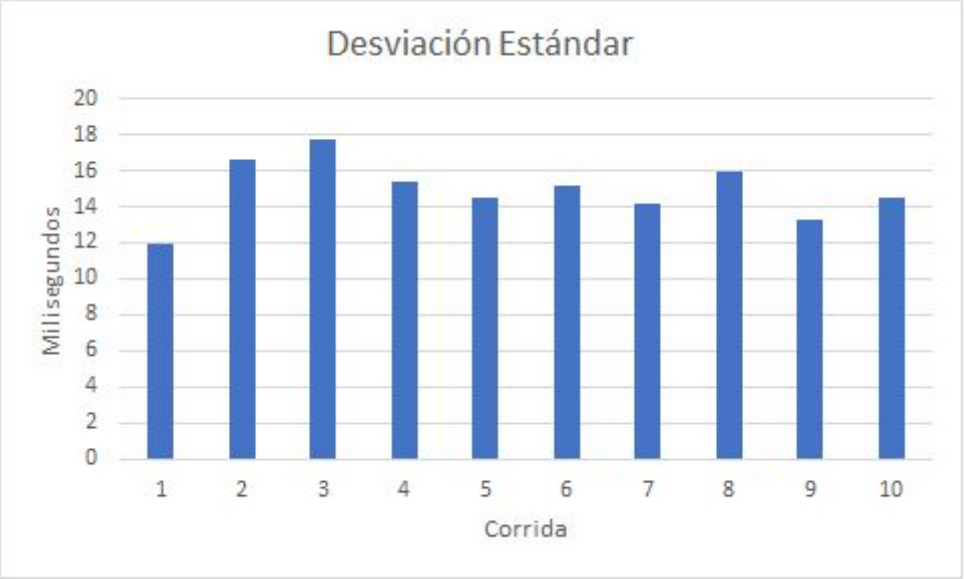

Figura 3. Gráfica desviación estándar para N=100.

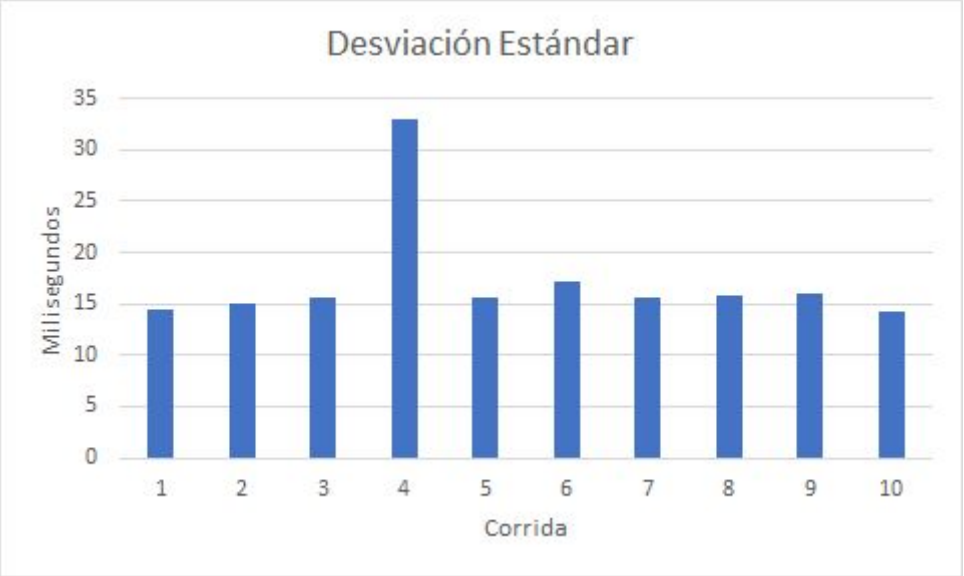

Figura 4. Gráfica desviación estándar para N=50.

Por último, como se reportó más arriba, encontramos que los clientes comenzaron a tener problemas de conexión con N=750. Cabe mencionar que no en todas las corridas que hicimos con esta N existieron problemas de conexión, sólo en algunas ocasiones. Sin embargo, no podemos permitir que el servidor a veces tenga problemas.

Manual de Usuario

Para poder correr nuestro programa lo único que se debe cambiar es el path de los .policy. Hay 3: uno en el archivo Login.java, otro en Administrador.java y el último en Clente.java. Este último únicamente sirve para la prueba de estresamiento.

Para el juego con interfáz gráfica

Una vez que los .policy estén listo, primero debe correrse el archivo Administrador y después el de Login. Al correr éste último aparece una ventana en donde se debe escribir el nombre de usuario y hacer click en Login. Para agregar más usuarios no es necesario correr otra vez el archivo Login (aunque también se puede), únicamente debe registrar un nuevo usuario en la misma ventana que se abrió al principio.

Si un usuario se desea retirar del juego debe hacer click en el botón Salir. Si hace click en el tache para cerrar la ventana, el programa no funcionará adecuadamente cuando el usuario quiera regresar a su partida.

Todos los usuarios deben hacer click en el botón de Start para que empiece una partida. La partida no empezará hasta que todos los usuarios conectados hagan este click, pues nuestro juego no quiere darle ventaja a ningún jugador.

Para el estresador

Una vez modificados los paths de los .policy únicamente es necesario determinar cuántos usuarios desea tener para el estresamiento. Esto se puede cambiar en el main del archivo Clente.java del paquete Estresador. Se debe cambiar el valor máximo de la i en el for de la línea 133. Después de elegir una N, debe correr el Administrador y después Clente. En el output se imprimirán los promedios de cada cliente. Es decir, si N=100 se imprimirán 100 valores, cada uno correspondiente a cada uno de los 100 clientes. Si se desea estresar al servidor siga las instrucciones en el Readme que se encuentra en el siguiente repositorio. https://github.com/FranciscoBuru/ProyectoAlpha