

Reporte parte 1

saturación del servidor

Como primera parte del ejercicio decidimos poner a prueba el servidor del proyecto *hello-word* haciendo varias solicitudes desde 3 equipos diferentes.

Diseño de la prueba:

Equipos usados:

Equipo	Función
Hgrid7	Servidor
Hgrid1	Cliente
Hgrid2	Cliente
Hgrid3	Cliente

La prueba se llevó acabo de la siguiente manera; se tenían pensadas 3 entradas para solicitarle al servidor en la función de Fibonacci, las cuales eran 40, 47 y 50. La idea es que por cada entrada se ejecutara el cliente y le realizara la solicitud al servidor, después se guardaba las salidas por cada petición en el archivo finalLog.txt donde podríamos evidenciar las salidas obtenidas y la eventual respuesta del servidor.

Todo lo anterior apoyado por medio del script de bash *run_clients.sh* él se encarga de todo lo anterior y también de enviar los archivos jar del cliente con cualquier modificación realizada.

Resultado de la prueba y conclusiones:

Después de ejecutar la prueba observamos lo siguiente en el archivo finalLog.txt y en la consola del servidor.

```
reports > finalLog.txt
1   Es el turno de hgrid1.icesi.edu.co
2   hgrid1.icesi.edu.co esta probando con : 50 el server respondió: 12586269025
3
4   com.zeroc.Ice.ConnectionTimeoutException
5   com.zeroc.Ice.ConnectionTimeoutException
6
7   Es el turno de hgrid2.icesi.edu.co
8   hgrid2.icesi.edu.co esta probando con : 40 el server respondió: 102334155
9
10  com.zeroc.Ice.ConnectionTimeoutException
11  com.zeroc.Ice.ConnectionTimeoutException
12
```

```
[swarch@hgrid7 callback-server-deploy]$ java -jar server.jar
Cliente: hgrid2.icesi.edu.co numero: 1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,2584,4181,6765,10946,17711,28657,46368,75025,121393,196418,
317811,514229,832040,1346269,2178309,3524578,5702887,9227465,14930352,24157817,39088169,63245986,102334155,
Cliente: hgrid1.icesi.edu.co numero: 1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,2584,4181,6765,10946,17711,28657,46368,75025,121393,196418,
317811,514229,832040,1346269,2178309,3524578,5702887,9227465,14930352,24157817,39088169,63245986,102334155,165580141,267914296,433494437,701408733,1
134903170,1836311903,2971215073,4807526976,7778742049,12586269025,
```

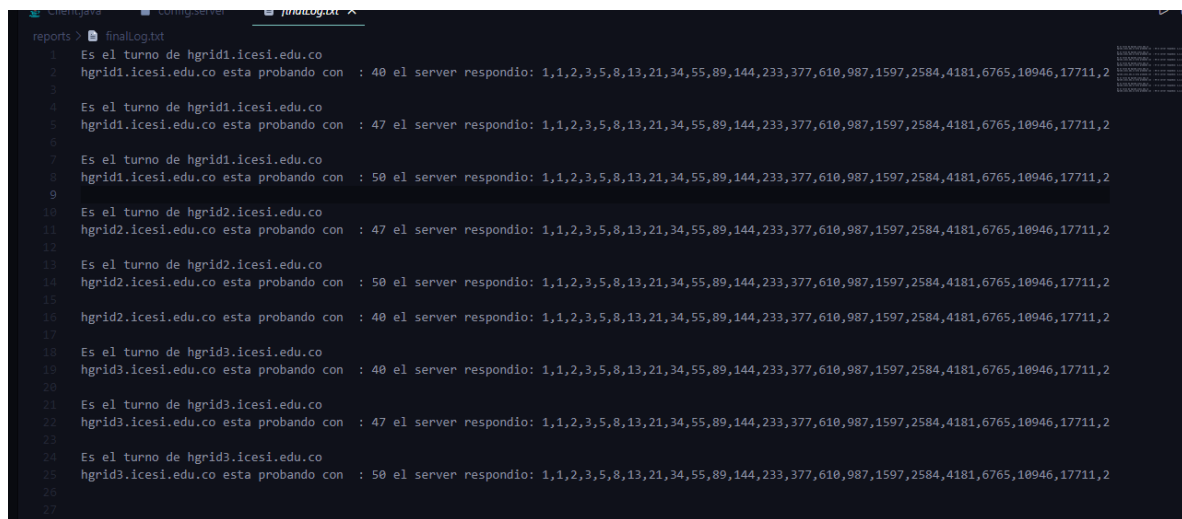
Es por todo lo anterior que, después de ver los resultados obtenidos, llegamos a la conclusión de que al momento de recibir múltiples solicitudes con números muy grandes se lanza la excepción *com.zeroc.Ice.ConnectionTimeoutException*, pues el servidor se encuentra ocupado con un cálculo complejo, lo cual hace que los otros procesos, provenientes de los otros clientes, se queden esperando la respuesta a la solicitud y eventualmente mueran de inanición lanzando la excepción y demostrándonos que no existe concurrencia en esta primera implementación del proyecto

PARTE 2:

Punto 1: En este caso la concurrencia es virtual, ya que como dice la definición, la concurrencia virtual se refiere a la ilusión de que varias tareas o procesos están ejecutándose simultáneamente en un sistema, aunque en realidad solo se está ejecutando una tarea a la vez. Esto se logra mediante la técnica de conmutación rápida entre tareas. El sistema asigna pequeños intervalos de tiempo a cada tarea en secuencia, de modo que parece que todas las tareas se están ejecutando simultáneamente. Esto es común en sistemas operativos que utilizan la planificación basada en tiempos compartidos, donde se da la ilusión de multitarea, aunque solo haya un procesador físico.

Evidentemente en este caso solo tenemos un procesador, pero podemos simular una concurrencia real mediante hilos.

Ice.ThreadPool.Server.Size=3



```
reports > finalLog.txt
1  Es el turno de hgrid1.icesi.edu.co
2  hgrid1.icesi.edu.co esta probando con : 40 el server respondió: 1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,2584,4181,6765,10946,17711,2
3
4  Es el turno de hgrid1.icesi.edu.co
5  hgrid1.icesi.edu.co esta probando con : 47 el server respondió: 1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,2584,4181,6765,10946,17711,2
6
7  Es el turno de hgrid1.icesi.edu.co
8  hgrid1.icesi.edu.co esta probando con : 50 el server respondió: 1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,2584,4181,6765,10946,17711,2
9
10 Es el turno de hgrid2.icesi.edu.co
11 hgrid2.icesi.edu.co esta probando con : 47 el server respondió: 1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,2584,4181,6765,10946,17711,2
12
13 Es el turno de hgrid2.icesi.edu.co
14 hgrid2.icesi.edu.co esta probando con : 50 el server respondió: 1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,2584,4181,6765,10946,17711,2
15
16 hgrid2.icesi.edu.co esta probando con : 40 el server respondió: 1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,2584,4181,6765,10946,17711,2
17
18 Es el turno de hgrid3.icesi.edu.co
19 hgrid3.icesi.edu.co esta probando con : 40 el server respondió: 1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,2584,4181,6765,10946,17711,2
20
21 Es el turno de hgrid3.icesi.edu.co
22 hgrid3.icesi.edu.co esta probando con : 47 el server respondió: 1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,2584,4181,6765,10946,17711,2
23
24 Es el turno de hgrid3.icesi.edu.co
25 hgrid3.icesi.edu.co esta probando con : 50 el server respondió: 1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,2584,4181,6765,10946,17711,2
26
27
```

```
new Thread(() -> {
    ChatClientPrx callbackPrx = getCallbackPrx(destination);
    System.out.println("new Message: " + msg);    Replace this
    comand(msg, callbackPrx, current);
}).start();
```

Punto 2: En este caso es necesario establecer un tiempo límite que puede estar el servidor ejecutando el Fibonacci, ya que al pasar más de 1 minuto ejecutando el cliente directamente detecta un timeout así el servidor continúe ejecutando la solución. Por lo mismo, aunque podemos especificar dentro de la configuración del servidor cuantos hilos vamos a habilitar y en efecto esa es la cantidad que el servidor va a recibir, de todas formas, si el número de Fibonacci que procederá a ejecutar es demasiado grande o tarda demasiado en completarse la ejecución directamente lanzará un timeout.

Punto 3: En este caso se evidencia que hay concurrencia ya que el servidor designa un hilo a cada uno de los clientes que hace solicitud haciéndolo de esta forma concurrente en sí mismo.