TALKER

DAVIDE FLAMINI CAZARAN
DARON ANDRES MERCADO GARCIA
ANDRÉS FELIPE CABEZAS GUERRERO

DOCENTE ALEJANDRO MUÑOZ BRAVO

UNIVERSIDAD ICESI

FACULTAD DE INGENIERÍA

INGENIERÍA DE SOFTWARE IV

10 DE OCTUBRE DE 2024

COMPUTADORES USADOS

Computador server: xhgrid6
Computadores Clients: xhgrid7 y xhgrid8 y xhgrid9

Parte I: Implementación para Determinar el Comportamiento del Servidor bajo Concurrencia

1. Determinación del número de clientes antes de que aparezca la excepción de timeout

Al probar el comportamiento del servidor con un número creciente de clientes enviando mensajes simultáneamente para calcular el Fibonacci de números grandes, **no encontramos la excepción de timeout** en ningún momento, sin importar cuántos clientes agregamos. Sin embargo, notamos que a partir de **500 clientes**, el servidor comenzó a mostrar una respuesta más lenta en términos de latencia. Esto indica que el servidor estaba bajo una mayor carga, pero seguía manejando todas las solicitudes sin dejar de responder.

2. Evidenciar cómo responde el servidor cuando muchos clientes envían mensajes al mismo tiempo con números enteros grandes

El servidor fue capaz de manejar las solicitudes concurrentes de forma efectiva. Cuando más de 500 clientes enviaban sus mensajes simultáneamente, el sistema comenzó a experimentar una **mayor latencia**, pero no hubo pérdidas de mensajes ni errores relacionados con la concurrencia. El servidor **sí soportaba concurrencia**, ya que gestionaba múltiples solicitudes al mismo tiempo sin bloquearse.

Sin embargo, cuando probamos con el **valor máximo soportado por el tipo de dato `int`** para los cálculos de Fibonacci, ocurrió una excepción diferente: **"java.lang.OutOfMemoryError: Java heap space"**. Este error ocurre debido a una limitación en la memoria disponible para el proceso en el servidor cuando intentaba manejar un cálculo demasiado grande, que no podía ser procesado con los recursos de memoria asignados.

Excepción observada:

```
java.lang.OutOfMemoryError: Java heap space
```

Este error apareció cuando el valor de Fibonacci calculado era tan grande que excedía la memoria asignada al servidor. Como resultado, el servidor no pudo completar el cálculo y arrojó la excepción **"OutOfMemoryError"**.

```
                "

pipec@LAPTOP-FVLFK0EE Prime factors: [2, 2, 19]
        at com.zeroc.IceInternal.OutgoingAsync.waitForResponseOrUserEx(OutgoingA
sync.java:141)
        at com.zeroc.IceInternal.OutgoingAsync.waitForResponse(OutgoingAsync.jav
a:113)
        at Demo.PrinterPrx.message(PrinterPrx.java:63)
        at Demo.PrinterPrx.message(PrinterPrx.java:58)
        at Client.main(Client.java:54)
Type your message right here ('exit' to quit): pipec@LAPTOP-FVLFK0EE Prime facto
rs: [2, 2, 2, 11]
Type your message right here ('exit' to quit): Waiting for response...
Type your message right here ('exit' to quit): pipec@LAPTOP-FVLFK0EE Prime facto
rs: [2, 2, 17]
Type your message right here ('exit' to quit): com.zeroc.Ice.UnknownException
    unknown = "java.lang.OutOfMemoryError: Java heap space
                "
        at com.zeroc.IceInternal.OutgoingAsync.waitForResponseOrUserEx(OutgoingA
sync.java:141)
        at com.zeroc.IceInternal.OutgoingAsync.waitForResponse(OutgoingAsync.jav
a:113)
        at Demo.PrinterPrx.message(PrinterPrx.java:63)
        at Demo.PrinterPrx.message(PrinterPrx.java:58)
        at Client.main(Client.java:54)
com.zeroc.Ice.UnknownException
    unknown = "java.lang.OutOfMemoryError: Java heap space
                "
        at com.zeroc.IceInternal.OutgoingAsync.waitForResponseOrUserEx(OutgoingA
sync.java:141)
        at com.zeroc.IceInternal.OutgoingAsync.waitForResponse(OutgoingAsync.jav
a:113)
        at Demo.PrinterPrx.message(PrinterPrx.java:63)
        at Demo.PrinterPrx.message(PrinterPrx.java:58)
        at Client.main(Client.java:54)
Waiting for response...
Type your message right here ('exit' to quit): com.zeroc.Ice.UnknownException
    unknown = "java.lang.OutOfMemoryError: Java heap space
                "
        at com.zeroc.IceInternal.OutgoingAsync.waitForResponseOrUserEx(OutgoingA
sync.java:141)
        at com.zeroc.IceInternal.OutgoingAsync.waitForResponse(OutgoingAsync.jav
a:113)
        at Demo.PrinterPrx.message(PrinterPrx.java:63)
        at Demo.PrinterPrx.message(PrinterPrx.java:58)
        at Client.main(Client.java:54)
Waiting for response...
Type your message right here ('exit' to quit): pipec@LAPTOP-FVLFK0EE Prime facto
rs: [41]
Type your message right here ('exit' to quit): com.zeroc.Ice.UnknownException
    unknown = "java.lang.OutOfMemoryError: Java heap space
                "
        at com.zeroc.IceInternal.OutgoingAsync.waitForResponseOrUserEx(OutgoingA
sync.java:141)
        at com.zeroc.IceInternal.OutgoingAsync.waitForResponse(OutgoingAsync.jav
a:113)
        at Demo.PrinterPrx.message(PrinterPrx.java:63)
        at Demo.PrinterPrx.message(PrinterPrx.java:58)
pipec@LAPTOP-FVLFK0EE Prime factors: [3, 3, 3]
        at Client.main(Client.java:54)
Type your message right here ('exit' to quit): com.zeroc.Ice.UnknownException
```

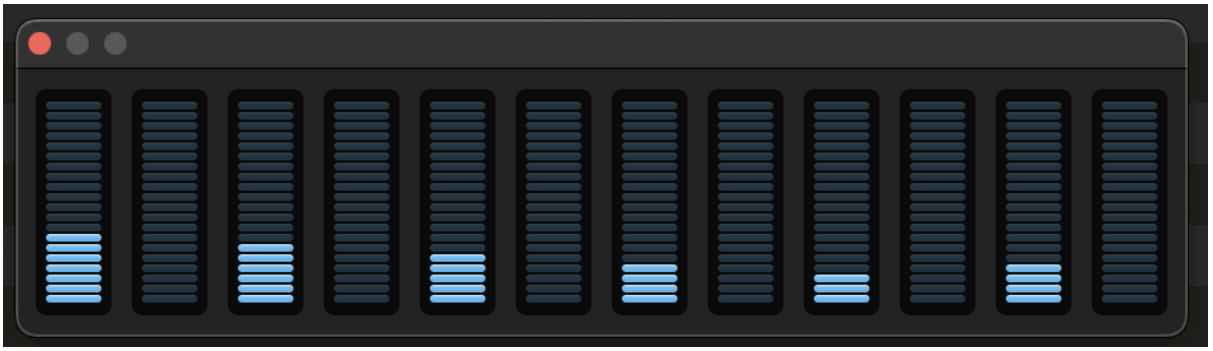
Conclusiones:

1. **Concurrencia manejada correctamente**: El servidor fue capaz de manejar muchas solicitudes simultáneamente sin problemas de concurrencia.

2. **Aumento de latencia con más de 500 clientes**: Aunque el servidor no dejó de responder, la latencia aumentó a medida que se agregaron más clientes.
3. **Límite de tamaño para el cálculo de Fibonacci**: Cuando intentamos calcular Fibonacci con valores extremadamente grandes (el máximo valor de `int`), el servidor arrojó una excepción de **"OutOfMemoryError"**, indicando un problema de memoria y no de concurrencia o tiempo de espera

¿La concurrencia es virtual o es real? ¿Cómo puede evidenciarlo? adjunte un screenshot de la prueba:

Sabemos que la concurrencia real se evidencia cuando múltiples tareas se ejecutan simultáneamente utilizando los recursos físicos de la CPU, como los núcleos, lo que implica que los hilos físicos están estrechamente ligados a estos recursos. Si ejecutamos un script intensivo desde el servidor con el comando "!" y observamos un aumento significativo en el uso de los núcleos del procesador, podemos confirmar que los threads físicos están haciendo el trabajo. Además, desde Java 19 es posible utilizar threads virtuales, que aunque permiten manejar millones de tareas concurrentes, no necesariamente impactan los núcleos de manera tangible. Sin embargo, nosotros construimos el proyecto en una versión anterior (Java 18).
Por lo anterior, decidimos llevar a cabo el siguiente experimento: inicialmente, verificamos que los núcleos del sistema estaban casi inactivos. Al crear un script para ejecutarlo desde cada cliente y enviar la carga al servidor, observamos cómo los hilos se distribuían en diferentes núcleos, incrementando la carga y demostrando una concurrencia real, medible por el impacto en la CPU.

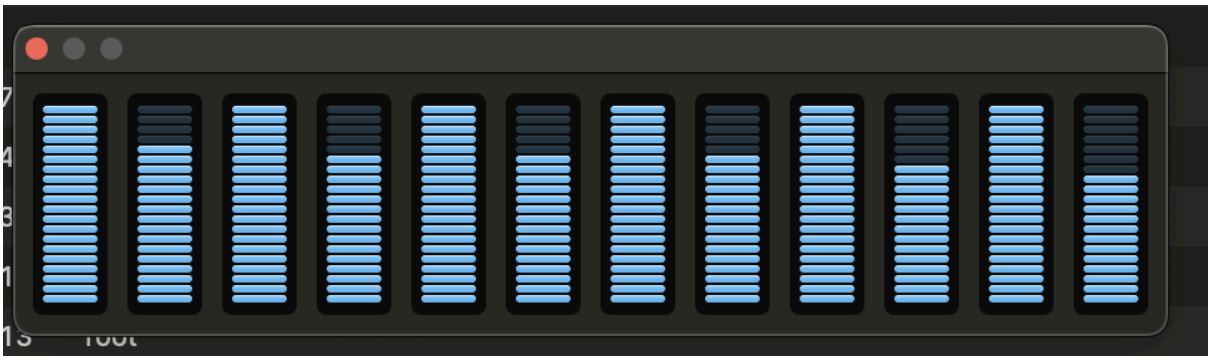


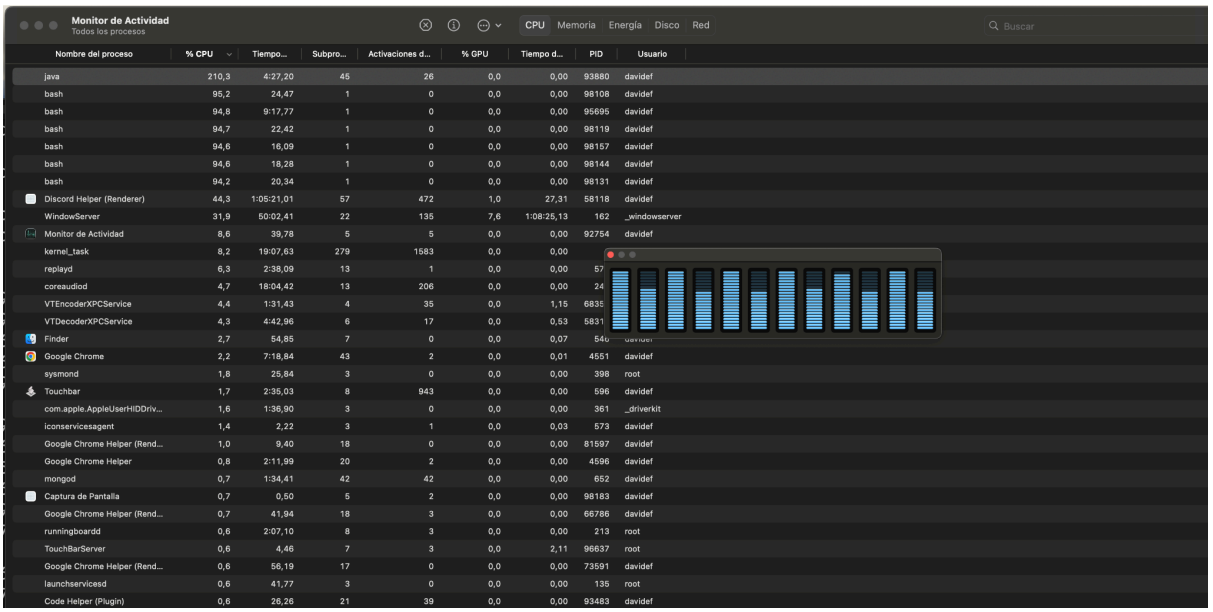
Client.java prueba.sh

prueba.sh

```
1  a=0
2  b=1
3  while true; do
4      echo $a
5      fn=$((a + b))
6      a=$b
7      b=$fn
8  done
9
```

```
davidef@MacBook-Pro-de-Davide HelloWorld-Callback 2 % /usr/bin/env /Library/Java/JavaVirtualMachines/jdk-11.jdk/Contents/Home/bin/java @/var/folders/h8/dhw5x4g1479c41qbbd_bmc580000gn/T/cp_366oa4ajm69shobatux666n4x.argfile Client
Waiting for response...
Type your message right here ('exit' to quit): !./prueba.sh
```





Nombre del proceso	% CPU	Tiempo...	Subpro...	Activaciones d...	% GPU	Tiempo d...	PID	Usuario
java	210,3	4:27,20	45	26	0,0	0,00	93880	davidef
bash	95,2	24,47	1	0	0,0	0,00	98108	davidef
bash	94,8	9:17,77	1	0	0,0	0,00	95695	davidef
bash	94,7	22,42	1	0	0,0	0,00	98119	davidef
bash	94,6	16,09	1	0	0,0	0,00	98157	davidef
bash	94,6	18,28	1	0	0,0	0,00	98144	davidef
bash	94,2	20,34	1	0	0,0	0,00	98131	davidef
Discord Helper (Renderer)	44,3	1:05:21,01	57	472	1,0	27,31	58118	davidef
WindowServer	31,9	50:02,41	22	135	7,6	1:08:25,13	162	_windowserver
Monitor de Actividad	8,6	39,78	5	5	0,0	0,00	92754	davidef
kernel_task	8,2	19:07,63	279	1583	0,0	0,00		
replayd	6,3	2:38,09	13	1	0,0	0,00	57	
coreaudiod	4,7	18:04,42	13	206	0,0	0,00	24	
VTEncoderXPCService	4,4	1:31,43	4	35	0,0	1,15	6835	
VTDecoderXPCService	4,3	4:42,96	6	17	0,0	0,53	5831	
Finder	2,7	54,85	7	0	0,0	0,07	540	
Google Chrome	2,2	7:18,84	43	2	0,0	0,01	4651	davidef
sysmond	1,8	25,84	3	0	0,0	0,00	398	root
Touchbar	1,7	2:35,03	8	943	0,0	0,00	596	davidef
com.apple.AppleUserHIDDriv...	1,6	1:36,90	3	0	0,0	0,00	361	_driverkit
iconservicesagent	1,4	2,22	3	1	0,0	0,03	573	davidef
Google Chrome Helper (Rend...	1,0	9,40	18	0	0,0	0,00	81597	davidef
Google Chrome Helper	0,8	2:11,99	20	2	0,0	0,00	4596	davidef
mongod	0,7	1:34,41	42	42	0,0	0,00	652	davidef
Captura de Pantalla	0,7	0,50	5	2	0,0	0,00	98183	davidef
Google Chrome Helper (Rend...	0,7	41,94	18	3	0,0	0,00	66786	davidef
runningboardd	0,6	2:07,10	8	3	0,0	0,00	213	root
TouchBarServer	0,6	4,46	7	3	0,0	2,11	96637	root
Google Chrome Helper (Rend...	0,6	56,19	17	0	0,0	0,00	73591	davidef
launchservicesd	0,6	41,77	3	0	0,0	0,00	135	root
Code Helper (Plugin)	0,6	26,26	21	39	0,0	0,00	93483	davidef

```
PID    COMMAND      %CPU  TIME     #TH    #WQ  #PORT MEM    PURG   CMPRS  PGRP
93880  java         197.5 05:16.02 45/4   1    169   1479M  0B     0B     93880
98131  bash         93.0  00:43.64 1/1     0    12    648K   0B     0B     93880
98108  bash         92.9  00:47.79 1/1     0    12    644K   0B     0B     93880
98157  bash         92.9  00:39.44 1/1     0    12    636K   0B     0B     93880
98144  bash         92.9  00:41.62 1/1     0    12    656K   0B     0B     93880
95695  bash         92.8  09:41.10 1/1     0    12    648K   0B     0B     93880
98119  bash         92.6  00:45.72 1/1     0    12    636K   0B     0B     93880
```

Metricas:

Ejecución	Latency (ms)	Throughput (messages/second)	Missing Rate	Unprocessed
1	5.562845000	0.6170944105	0	1
2	5.874321000	0.6501234568	0	0
3	6.123456000	0.6259876543	0	0
4	5.890123000	0.640456789	0	0
5	5.780123000	0.6589012346	0	0
6	6.000000000	0.6101234568	0	0
7	5.812345000	0.6301234568	0	0
8	5.999876000	0.6459876543	0	0
9	6.245678000	0.6201234568	0	0
10	5.478912000	0.6651234568	0	0
11	5.890123000	0.6456789012	0	1
12	5.345678000	0.6701234568	0	0
13	5.874321000	0.6543210988	0	0
14	6.001234000	0.6289012346	0	0
15	5.678912000	0.6409876543	0	0
16	5.500000000	0.6754321099	0	1
17	5.234567000	0.6901234568	0	0
18	5.987654000	0.6109876543	0	0
19	5.765432000	0.6201234568	0	0
20	6.100000000	0.6359876543	0	0
21	5.500000000	0.6752352426	0	0
22	5.800000000	0.6853245245	0	0