Concurrencia de Recursos Distribuidos

Students:
Anthony Daniel, Bautista León
Katheryn Ximena, Peralta Haro
Renzo Renato, Quispe Amao
Universidad Nacional de Ingeniería, Facultad de Ciencias,
e-mail: abautistal@uni.pe, katheryn.peralta.h@uni.pe, rrquispea@uni.pe

Curso:
CC4P1 Programación Concurrente y Distribuida
Laboratorio 04

Abstract

El objetivo de este trabajo es comprender como se comparte los recursos con la ayuda de un Balanceador, un dispositivo software, que atiende las solicitudes de los clientes y las reparte a los servidores. En este trabajo se implementó un balanceador para las transacciones de consulta y actualización de un conjunto de cuentas bancarias. Al finalizar el experimento el dinero total no ha variado en ningun servidor.

Keywords: Blockingqueue, Balanceador carga, Colas, formato UTP.

Contents

1 Introducción

El uso masivo de internet para todo tipo de transacciones, desde una cotidiana búsqueda de información a la compra semanal, produce una carga creciente de los servidores web responsables de la disponibilidad de tiendas electrónicas, portales informativos y páginas web. Paralelamente, también crecen las expectivas de los usuarios, que esperan, especialmente en el área de servicios, un funcionamiento rápido y seguro de las aplicaciones web. Es así como la disponibilidad de una página repercute en la transformación de un usuario interesado en un cliente.[1]

Un servidor sobrecargado puede ser perjudicial para cualquier proyecto web, pues puede frenar e incluso paralizar algunas áreas de cualquier negocio. Sin embargo, es posible prevenir este tipo de situaciones. Utilizando una distribución eficiente del flujo de datos. Así, entre arios servidores es posible controlar picos de tráfico en épocas de gran demanda y disminuir el riesgo de fallos en un servidor. Este procedimiento es lo que se conoce como Load Balancing o Balanceo de carga funciona gracias a un balanceador que distribuye las solicitudes de los clientes entre los diferentes servidores para garantizar una velocidad estable de respuesta.[1]

2 Marco Teórico

2.1 Sistemas Distribuidos

Es un colección de computadoras o máquinas conectadas por una red de comunicaciones, en la cual el usuario percibe la apariencia de interactuar con una sola computadora, debido a que el mismo puede acceder a los recursos remotos de la misma manera en que accede a los recursos locales.[2]

Los sistemas distribuidos tienen como meta principal mantener siempre una conexión entre los usuarios y los recursos que necesitan. Tener transparencia de distribución para que no sepa por parte del usuario que computadoras llevaron a cabo las tareas que este solicita.
Deben ser abiertos hasta un cierto grado, para que nuevos servicios de compartición de recursos tengan la capacidad de ser añadidos, siempre y cuando no perjudiquen ni dupliquen a los ya existentes. Tener escalabilidad para futuras optimizaciones tanto de software como de hardware para brindar servicio al mayor número de usuarios en el menor tiempo posible.[2]

Existen 3 diferentes formas de procesamiento distribuido:

- Basado en llamadas a procedimientos remotos: invocar procesos remotos como si fuera de manera local. Aquí, los detalles de envío-recepción de mensajes quedan ocultos para el usuario y se cumple con la transparencia.
- Basado en objetos distribuidos: cuando la invocación a los procesos remotos es similar a invocar métodos de objetos. Por ende, se tiene un enfoque de nivel más alto al anterior y se cumple por igual con la transparencia.
- Basado en memoria compartida: ocurre en casos en los cuales un proceso necesita leer o escribir datos en una memoria común para diferentes computadoras. En este sentido, se necesita contar con un mecanismo fiable de sincronización de procesos.

2.2 Modelo Cliente-Servidor

Este modelo visto como una arquitectura de comunicación en red que permite al usuario en una máquina, llamada el cliente, requerir algún tipo de servicio de una máquina a la que está unido, llamado el servidor, mediante una red de área local (LAN, de su nombre en inglés Local Area Network) o una red de área amplia o ancha (WAN, de su nombre en inglés Wide Area Network). Estos servicios pueden ser peticiones de datos de una base de datos, de información contenida en archivos o los archivos en sí mismos, mayormente peticiones de imprimir datos en una impresora asociada.[3]

Aunque clientes y servidores suelen verse como máquinas separadas, pueden, de hecho, ser dos áreas o procesos separados en la misma máquina. Por lo tanto, una única máquina puede ser al mismo tiempo cliente

y servidor. Además, una máquina cliente unida a un servidor puede ser a su vez servidor de otro cliente y el servidor puede ser un cliente de otro servidor en la red tal como se muestra en la figura 1.

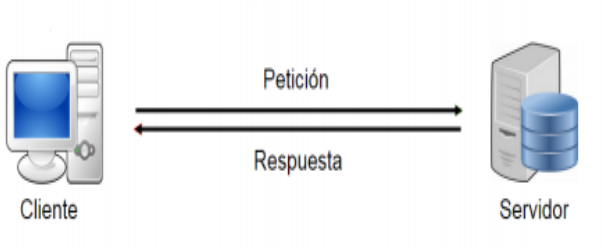

Figure 1: Funcionamiento del modelo Cliente-Servidor

2.3 Máquina Cliente

Es quien empaqueta los requerimientos que el usuario formuló en su petición para que sean enviados al servidor. Esto dice que el cliente es el encargado del manejo, manipulación y despliegue de los datos para con el usuario, dando paso a que permita una interacción con el usuario por medio de interfaces gráficas para tener acceso a los servicios distribuidos que se encuentren en cualquier componente o nodo de la red.

Las tareas principales que realiza el cliente son las siguientes: administrar la interfaz de usuario, mantener una interacción confiable con el usuario, procesar la lógica de la aplicación y hacer validaciones locales, generar requerimientos de bases de datos, recibir los resultados que fueron generados por el servidor y formatear dichos resultados.

2.4 Máquina Servidor

Es el opuesto al cliente, ya que es visto como el encargado de atender peticiones de manera secuencial. Es decir, el servidor normalmente maneja todas las funciones relacionadas con la mayoría de las reglas del negocio y los recursos de datos.

Las funciones que lleva a cabo el proceso servidor se resumen como sigue: aceptar los requerimientos de bases de datos que hacen los clientes, procesar requerimientos de bases de datos, formatear datos para trasmitirlos a los clientes, procesar la lógica de la aplicación y realizar validaciones a nivel de bases de datos.

2.5 Balanceo y distribución de Carga

El balanceo de carga de trabajo se puede definir como una técnica que incrementa los recursos, explotando el paralelismo y disminuyendo el tiempo de respuesta mediante la distribución de carga apropiada, o visto desde otro punto, es la división de la carga de trabajo de una computadora o servidor central que se le ha asignado, para que este sea realizado entre dos o más computadoras conectadas al servidor central, este tipo de división de carga permite realizar la petición o tarea en un intervalo de tiempo mucho más reducida, y así lograr realizar más carga de trabajo en el mismo intervalo de tiempo total.[4]

Un esquema de la distribución uniforme significa que las tareas se distribuyen de manera uniforme entre los servidores participantes de un clúster. Este esquema es, muy simple, hace que sea más fácil de implementar el esquema de la distribución de tareas, incluso también se conoce como Round Robin, es decir; los servidores reciben el trabajo en turnos rotativos (distribuida uniformemente).

La distribución de carga no toma en cuenta la diferencia que existe en el procesamiento que se requiere para procesar cada tarea, y aunque cada servidor se le dé el mismo rango de tareas, puede darse el caso que un servidor llegue a recibir un mayor número de tareas que requieran de una mayor cantidad de procesamiento que el resto. Así que, aunque el equilibrio de distribución de carga de continúa distribuyendo las tareas de manera uniforme en los servidores participantes, no se tiene la certeza de que se pueda dar lugar a una verdadera distribución uniforme de la carga.[5]

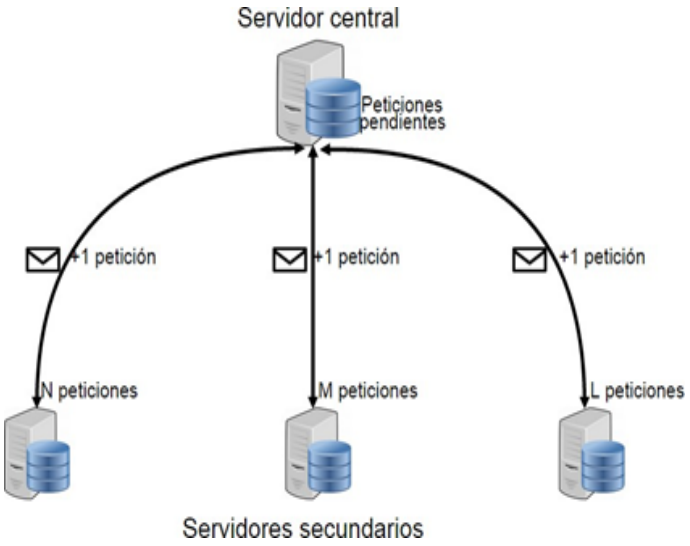

Figure 2: Distribución de carga

2.5.1 Método Round Robin

Existen diversos métodos para distribuir la carga, y el más simple de todos, es la solución Round Robin. Las peticiones clientes son distribuídas equitativamente entre todos los servidores existentes. Este método ciclico no tiene un cuenta las condiciones y carga de cada servidor. Esto puede llevarnos a tener servidores que reciben peticiones de carga mucho mayor, mientras tenemos servidores que apenas se encuentran utilizando recursos.[6]

2.6 Distribución de carga en cola

El equilibrio de la carga mantiene una cola de tareas para cada una de los servidores. Las colas de tareas contienen todas las peticiones que cada servidor que está procesando alguna tarea o que están en espera de alguna. Las tareas que aún se encuentran encoladas bajo los trabajos de esquema de distribución, aseguran de que cada cola de cada servidor tiene la misma cantidad de tareas correspondientes al mismo intervalo de tiempo, los servidores que cuentan con una mayor capacidad terminar´an las tareas en un menor intervalo de tiempo a diferencia de los servidores con baja capacidad, y así, las colas de tareas de los servidores de mayor capacidad se liberaran más rápido y por lo tanto liberar un espacio para las nuevas tareas en un menor intervalo de tiempo.

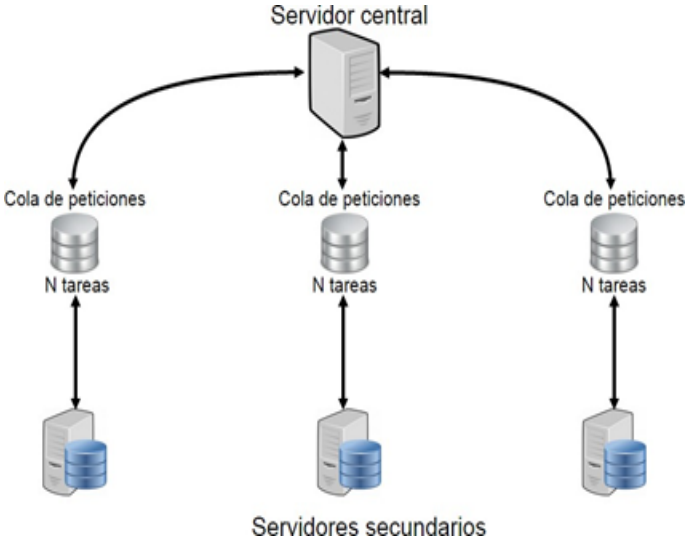

Figure 3: Distribución de tareas encoladas

3 Metodologia

Se implemetan tres programas de comunicación.

- Cliente
- Balanceador
- Replica (Servidor secundario)

3.1 Cliente

Este programa esta desarrollado para comunicarse con el balanceador por medio de sockets. Utiliza metodos para enviar y recibir mensajes.
Los mensajes son de la forma :
id_solicitud = id_cliente_operacion_NumeroOperacion
id_account= Numero de cuenta del cliente
de: cuenta 1 (de quien se pide dinero)
para: cuenta 2(quien recibe el dinero)

- Operacion==L
  Petición: id_solicitud;id_account
  Respuesta: id_solicitud;id_account;saldo de id_account

- Operacion==A
  Petición: id_solicitud;de;para;dinero_de
  Respuesta: id_solicitud;de:saldo_de;para;saldo_para

3.2 Balanceador

Este programa esta desarrollado para comunicarse con los clientes y replicas, implementado con listas de hilos para la comunicación por sockets con los clientes y replicas. Además que se utilizaron colas de bloqueo para evitar la corrupción de la información en el balanceador y su posterior sobreescritura en la base de datos. El balanceador utiliza los atributos de las replicas como libre y bloqueado para hacer las transacciones de lectura o actualización.

3.3 Replica o Servidor secundario

Programa encargado de resolver las peticiones reenviadas por el balanceador para ser consultadas en la base de datos de la replica.

3.4 Comunicación en las Peticiones

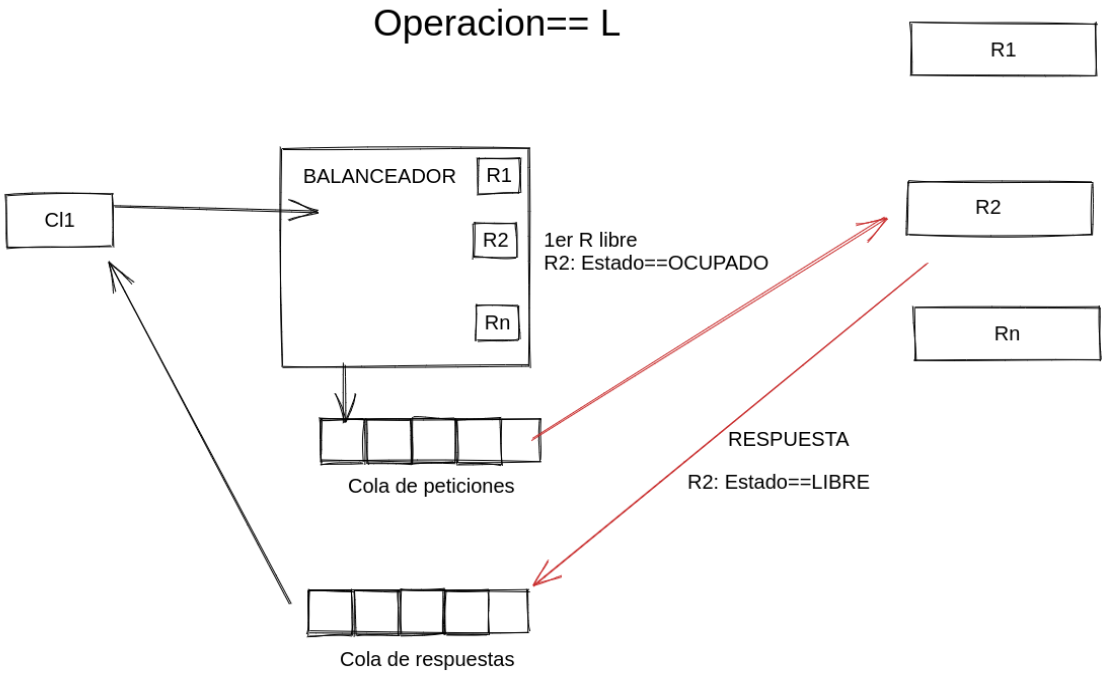

Figure 4: Petición de Lectura

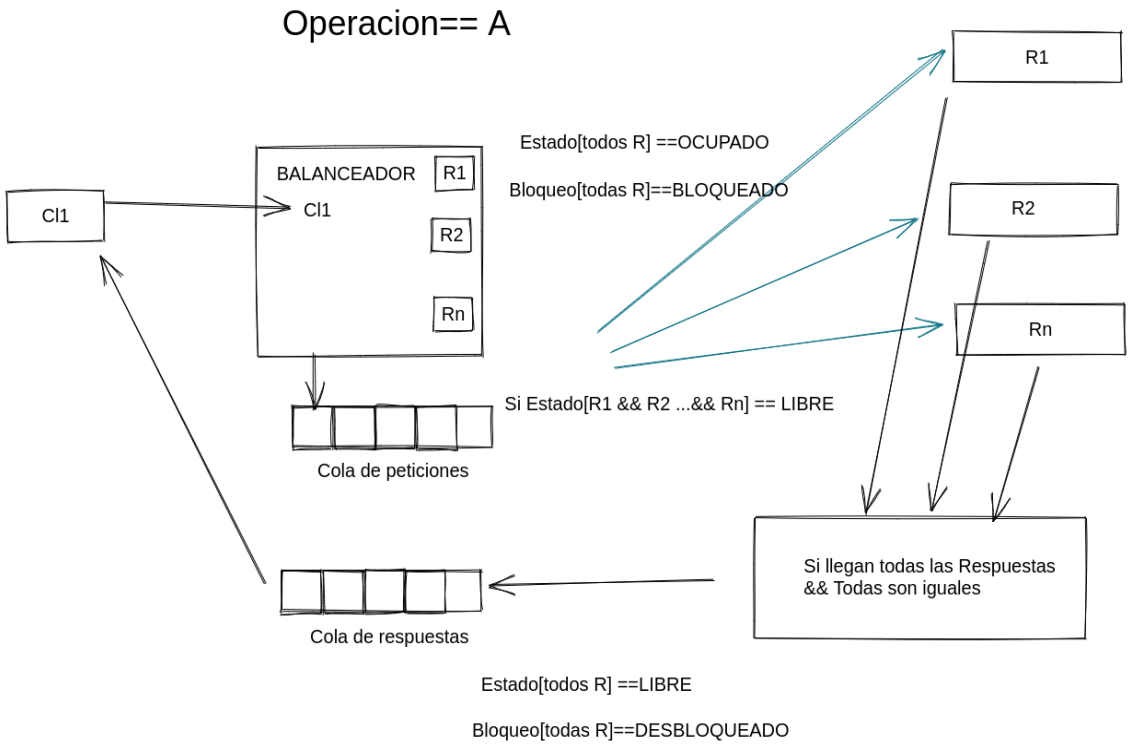

Figure 5: Petición de Actualización

4 Desarrollo de la Investigación

Para el desarrollo de este laboratorio de implemento tres programas en java:

- Cliente
- Balanceador
- Replica (Servidor secundario)

A continuación se presenta el diagrama de clases de cada programa.

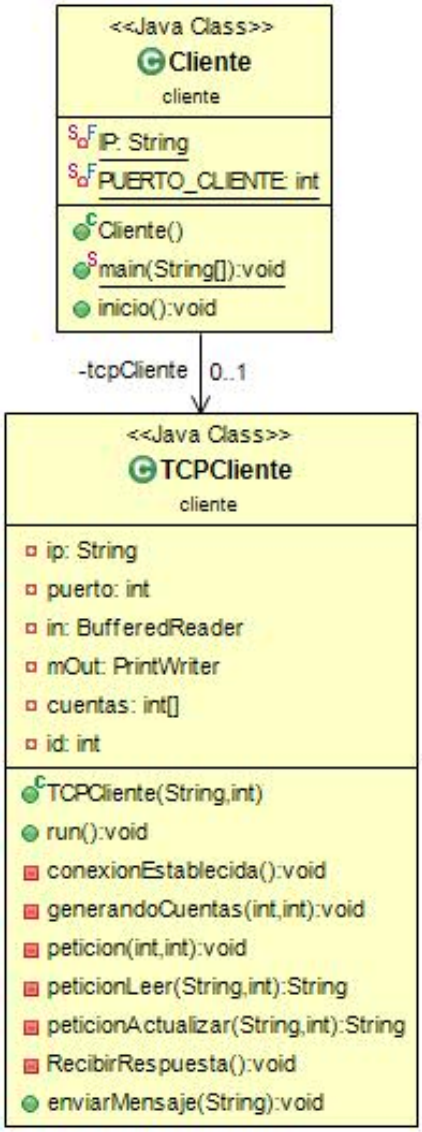

Figure 6: Diagrama de clases de cliente

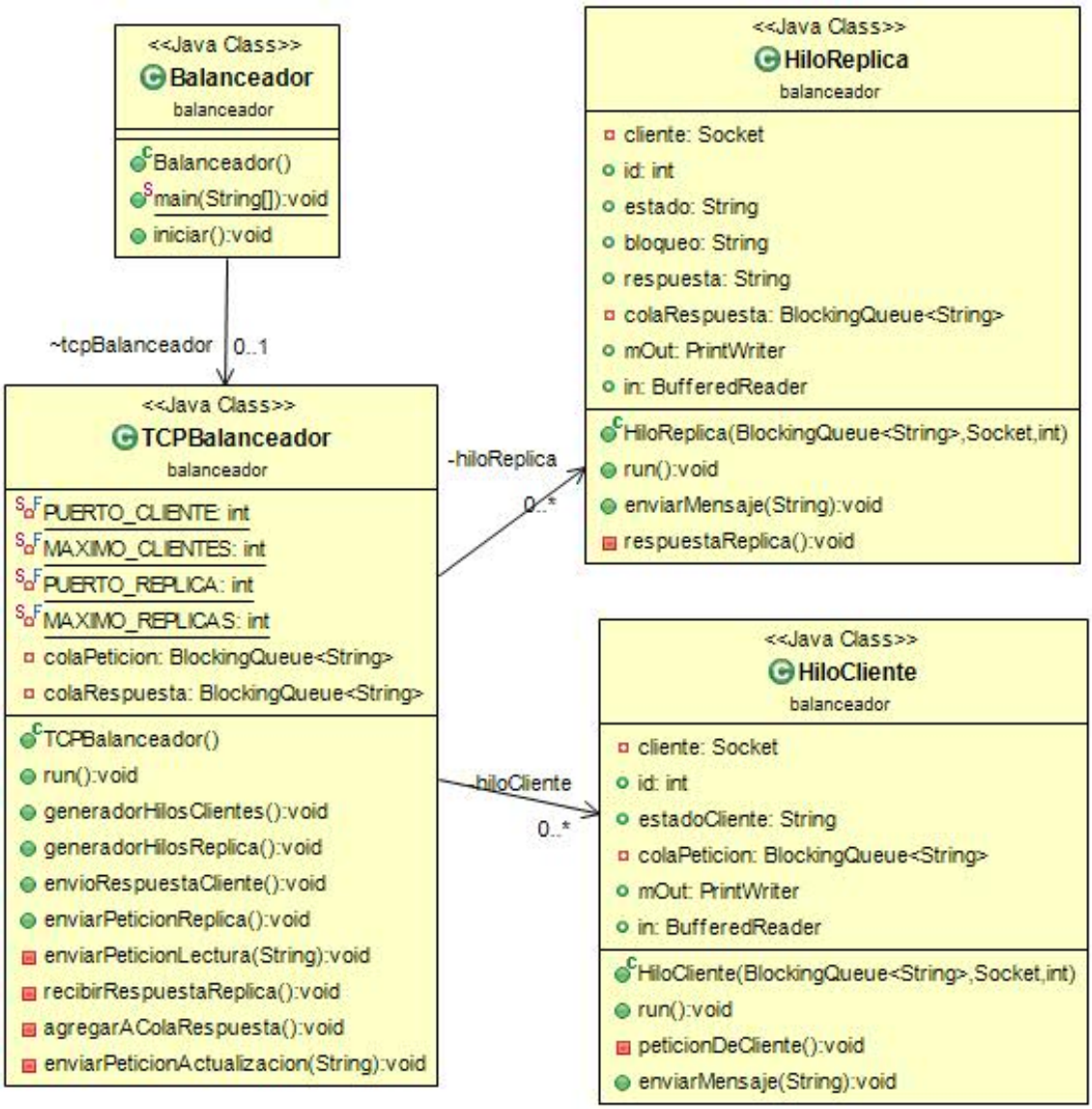

Figure 7: Diagrama de clases de Balanceador

Figure 8: Diagrama de clases de Replica

5 Resultados

- El laboratorio fue ejecutado con 5 clientes, 1 balanceador y 2 replicas (servidores secundarios). El balanceador genera los hilos para la comunicación con los clientes, replicas y espera la conexión de estos.
- Las replicas al iniciarse cada una genera las diez mil cuentas con una cantidad de mil cada una, y son almacenadas en un array del csv, y en cada petición esta sobrescribiendo el csv por completo.
- Los clientes estan programados para generar 500 peticiones cada $0,01$ segundos. La probabilidad de que la petición sea de lectura es de 0.6 y la de actualización es de 0.4.

Los resultados obtenidos fueron :

```
rpta: 1-L-490;16,35
petición: 1-L-491;16
rpta: 1-L-491;16,35
petición: 1-L-492;64
rpta: 1-L-492;64,28
petición: 1-L-493;100
rpta: 1-L-493;100,29
petición: 1-A-494;81;2484;110
rpta: 1-A-494,81,10,2484,1110
petición: 1-A-495;83;3217;331
rpta: 1-A-495, SALDO INSUFICIENTE
petición: 1-L-496;21
rpta: 1-L-496;21,1
petición: 1-L-497;64
rpta: 1-L-497;64,28
petición: 1-L-498;63
rpta: 1-L-498;63,109
petición: 1-L-499;78
rpta: 1-L-499;78,177
petición: 1-L-500;39
rpta: 1-L-500;39,13
FINALIZO PETICIONES DE CLIENTE 1
```

Figure 9: Resultados de Cliente 1

```
rpta: 4-L-490;301,51
petición: 4-A-491;389;9776;946
rpta: 4-A-491, SALDO INSUFICIENTE
petición: 4-A-492;367;6445;433
rpta: 4-A-492, SALDO INSUFICIENTE
petición: 4-L-493;372
rpta: 4-L-493;372,34
petición: 4-A-494;325;7708;996
rpta: 4-A-494, SALDO INSUFICIENTE
petición: 4-A-495;367;1391;915
rpta: 4-A-495, SALDO INSUFICIENTE
petición: 4-L-496;331
rpta: 4-L-496;331,62
petición: 4-A-497;388;3262;499
rpta: 4-A-497, SALDO INSUFICIENTE
petición: 4-A-498;325;7572;315
rpta: 4-A-498, SALDO INSUFICIENTE
petición: 4-L-499;318
rpta: 4-L-499;318,37
petición: 4-A-500;387;9026;905
rpta: 4-A-500, SALDO INSUFICIENTE
FINALIZO PETICIONES DE CLIENTE 4
```

Figure 10: Resultado de Cliente 4

```
<terminated> Replica [Java Application] C:\Program Files\Java\jdk-14.0.1\bin\javaw.exe
Recibe peticion: 3-L-498;209
Enviar Saldo: 3-L-498;209,42
Recibe peticion: 4-A-479;325;9377;733
Enviando Saldo: 4-A-479, SALDO INSUFICIENTE
Recibe peticion: 3-A-500;215;9665;854
Enviando Saldo: 3-A-500, SALDO INSUFICIENTE
Recibe peticion: 4-A-480;367;4530;680
Enviando Saldo: 4-A-480, SALDO INSUFICIENTE
Recibe peticion: 4-A-482;388;4655;967
Enviando Saldo: 4-A-482, SALDO INSUFICIENTE
Recibe peticion: 4-A-491;389;9776;946
Enviando Saldo: 4-A-491, SALDO INSUFICIENTE
Recibe peticion: 4-A-492;367;6445;433
Enviando Saldo: 4-A-492, SALDO INSUFICIENTE
Recibe peticion: 4-A-494;325;7708;996
Enviando Saldo: 4-A-494, SALDO INSUFICIENTE
Recibe peticion: 4-A-495;367;1391;915
Enviando Saldo: 4-A-495, SALDO INSUFICIENTE
Recibe peticion: 4-A-497;388;3262;499
Enviando Saldo: 4-A-497, SALDO INSUFICIENTE
Recibe peticion: 4-A-498;325;7572;315
Enviando Saldo: 4-A-498, SALDO INSUFICIENTE
Recibe peticion: 4-A-500;387;9026;905
Enviando Saldo: 4-A-500, SALDO INSUFICIENTE
Recibe peticion: FINALIZO
------------------------------------------
               SALDO TOTAL: 10000000
FINALIZADO OPERACIONES DE TODOS LOS CLIENTES
```

Figure 11: Resultado de Replica 1

```
<terminated> Replica [Java Application] C:\Program Files\Java\jdk-14.0.1\bin\javaw.exe
Recibe peticion: 4-L-489;400
Enviar Saldo: 4-L-489;400,46
Recibe peticion: 4-L-490;301
Enviar Saldo: 4-L-490;301,51
Recibe peticion: 4-A-491;389;9776;946
Enviando Saldo: 4-A-491, SALDO INSUFICIENTE
Recibe peticion: 4-A-492;367;6445;433
Enviando Saldo: 4-A-492, SALDO INSUFICIENTE
Recibe peticion: 4-L-493;372
Enviar Saldo: 4-L-493;372,34
Recibe peticion: 4-A-494;325;7708;996
Enviando Saldo: 4-A-494, SALDO INSUFICIENTE
Recibe peticion: 4-A-495;367;1391;915
Enviando Saldo: 4-A-495, SALDO INSUFICIENTE
Recibe peticion: 4-L-496;331
Enviar Saldo: 4-L-496;331,62
Recibe peticion: 4-A-497;388;3262;499
Enviando Saldo: 4-A-497, SALDO INSUFICIENTE
Recibe peticion: 4-A-498;325;7572;315
Enviando Saldo: 4-A-498, SALDO INSUFICIENTE
Recibe peticion: 4-L-499;318
Enviar Saldo: 4-L-499;318,37
Recibe peticion: 4-A-500;387;9026;905
Enviando Saldo: 4-A-500, SALDO INSUFICIENTE
Recibe peticion: FINALIZO
------------------------------------------
               SALDO TOTAL: 10000000
FINALIZADO OPERACIONES DE TODOS LOS CLIENTES
```

Figure 12: Resultado de Replica 2

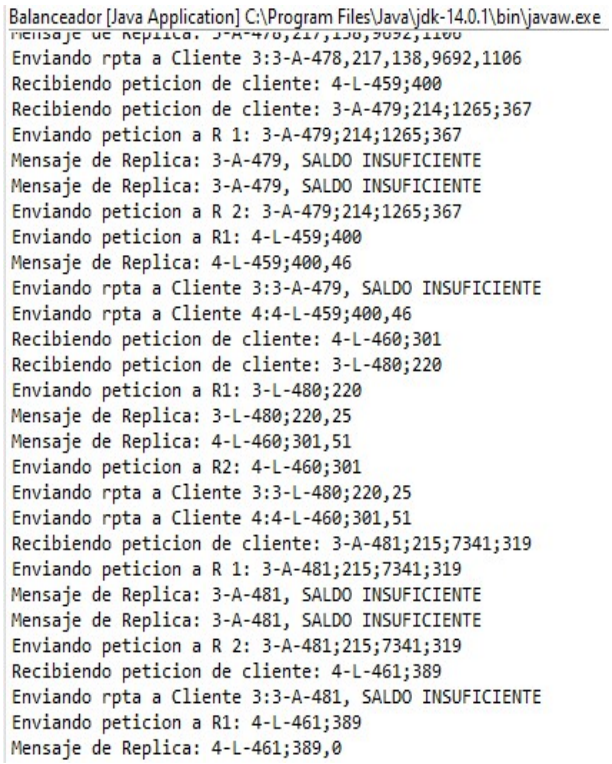

Figure 13: Resultado de Balanceador

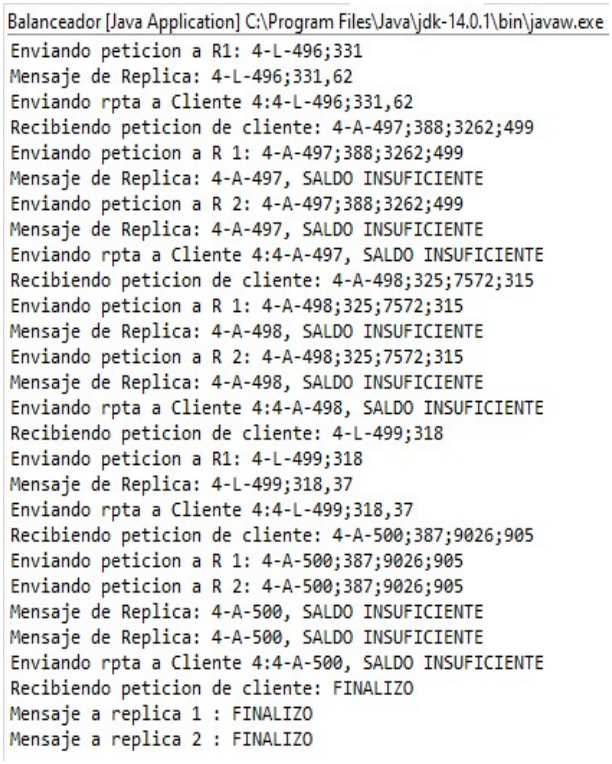

Figure 14: Resultado de Balanceador

6 Conclusiones

- Se implementó un balanceador de carga que recepciona las peticiones de los clientes y las envia a los servidores manteniendo una comunicación distribuida y sincronizada con la ayuda de colas de bloqueo en el balanceador de carga evitando así la corrupción los datos y que la capacidad del servidor sea sobrepasada y no halla una caída del servicio.

7 Anexo Código

`https://github.com/renzoqamao/CC4P1-ProgramacionConcurrente-Distribuida`

8 Anexo Documentación

1. Mejora el rendimiento de tu servidor con el balanceo de Carga, Digital Guide Ionos, https://www.ionos.mx/digitalgui how/balanceo-de-carga-conoce-a-fondo-sus-ventajas/
2. George F. Coulouris and Jean Dollimore. Distributed systems: concepts and design. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1994.
3. Alex Berson. Client-Server Architecture. 2nd. edition,McGraw-Hill, Universidad de Michigan, USA, 2007.
4. H. Castillo Zacatelco. Balance de carga en sistemas distribuidos. Vol. 7, pages 1–10, Octubre 2007.
5. Balanceo-Distribución de carga de trabajo y fragmentación-replicación de información para el servidor de aplicaciones web Apache Tomcat, C. Victor Hugo Carbajal Rosas,Centro Universitario UAEM Vale de México.
6. Aplicando Teoría de Colas en Dirección de Operaciones, José Pedro García Sabater, Universidad Politécnica de Valencia, http://personales.upv.es/jpgarcia/LinkedDocuments/Teoriadecolasdoc.pdf