Aplicación de Ventas: GoShop

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
Examen Final

Abstract

En este trabajo presentamos una aplicación cliente/servidor con una base de datos compartida la cual trabajará con cuatro servidores activos y un número indefenido de clientes. Este aplicativo tiene tolerancia fallos con respecto a la caída de un servidor, ya que la conexión no se verá afectada y no se generan datos corruptos.

Keywords: Blockingqueue, Balanceador carga, Colas.

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 varios 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]. Además nuestro servidor de balanceo tambien se encargará de la gestión de conexión con los otros servidores y se encargará de evaluar su disponibilidad para poder ejecutar operaciones.

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 Tolerancia a Fallos

La tolerancia a fallos es considerada la principal característica que debe de tener un sistema distribuido para alcanzar el principio de transferencia. Para lograr la tolerancia a fallos se necesita de una buena comunicación entre procesos distribuidos y sobre todo de una correcta coordinación entre ellos. Un sistema distribuido en base a la coordinación de sus procesos puede ser:

- Asíncrono : no hay coordinación en el tiempo.

- Síncrono : se suponen límites máximos para el retraso de mensajes.

El primer factor a tomar en cuenta es que el canal de comunicación este libre de errores (canal confiable). Para garantizar que el canal sea confiable se debe tener QoS (Calidad en el servicio) que implica realizar lo siguiente:

- Retransmisión de mensajes.
- Establecer redundancia de canales.
- Poner límite al tiempo de entrega de un paquete en lapso especificado.

En general, se considera que los canales de comunicación son fiables y que cuando falla la comunicación es debido a la caída del proceso, sin embargo algunos fallos en el funcionamiento de un sistema distribuido pueden originarse por :

- Especificaciones impropias o con errores.
- Diseño deficiente del software o hardware.
- Deterioro o averías de Hardware

2.3 Prevención en la tolerancia a Fallos

Existen dos formas de aumentar la fiabilidad de un sistema.

1. Prevención de fallos: Se trata de evitar que se implementen sistemas que pueden introducir fallos.
2. Tolerancia a fallos: Se trata de conseguir que el sistema continué funcionando correctamente aunque se presenten algunos fallos.

Un sistema que sea tolerante a fallos debería de tener disponibilidad, confiabilidad, seguridad y un programa de mantenimiento.

- Disponibilidad : La cualidad de un sistema de estar preparado en todo momento para operar.
- Confiabilidad : La garantía de que el sistema puede llevar a cabo su trabajo con muy bajas probabilidades de una caída repentina.
- Seguridad : La característica de que el sistema puede recuperarse o repararse a si mismo en caso de presentarse algún tipo de fallo.
- Mantenimiento : Se refiere a que el sistema puede ser remplazado o reparado rápidamente mediante los lineamientos de un programa preventivo y un plan de contigencia.

Tipo de Fallo	Descripción
Fallo Crash	El servidor se detiene pero estaba operando correctamente.
Fallo por Omisión -Omisión de recibido -Omisión de envio	Un servidor falla en responder a las peticiones. El servidor falla en recibir mensajes. El servidor falla en mandar mensajes.
Fallo de tiempo	La respuesta del servidor no esta en el intervalo de tiempo especificado.
Fallo de respuesta -Fallo de valor -Fallo de estado de transición	La respuesta del servidor es incorrecta. El valor de la respuesta es incorrecto. El servidor se desvia del flujo de control.
Fallo arbitrario	El servidor produce fallos arbitrarios en tiempo indefenidos.

2.4 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.

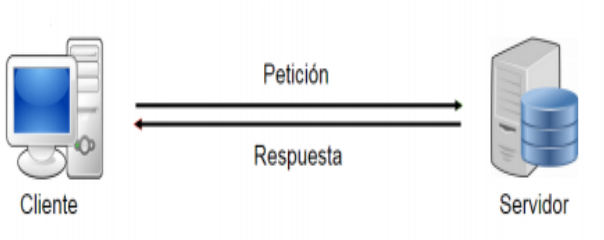


Figure 1: Funcionamiento del modelo Cliente-Servidor

2.5 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.

2.6 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.

2.7 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]

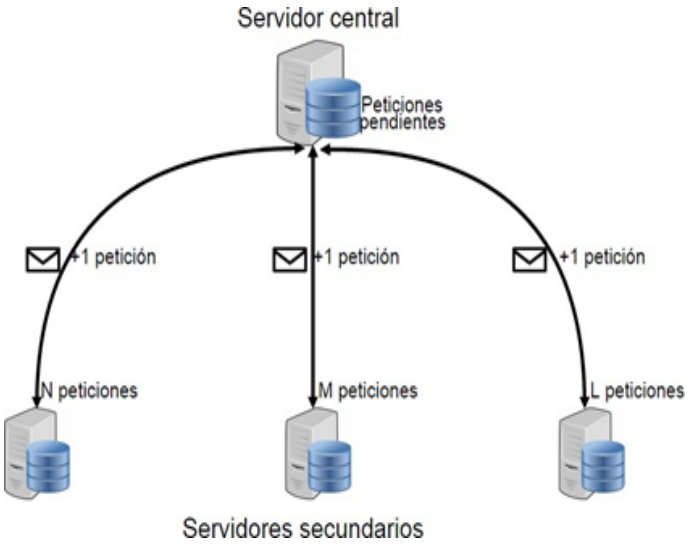


Figure 2: Distribución de carga

2.7.1 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]

2.8 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.

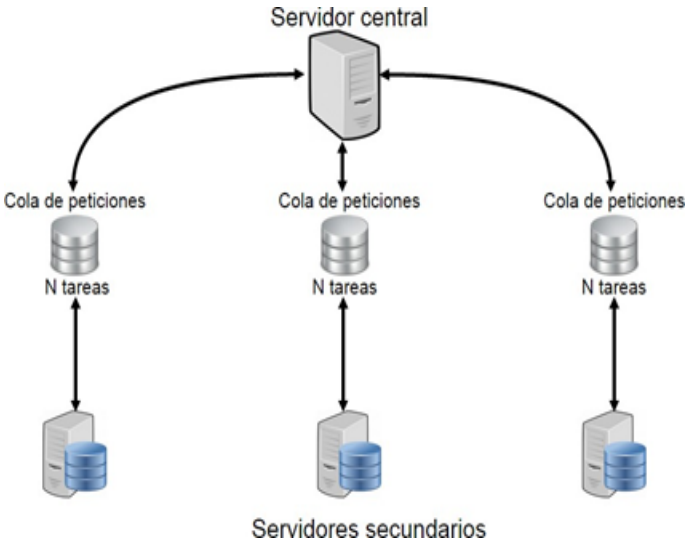


Figure 3: Distribución de tareas encoladas

3 Metodología

Creación de la Base de Datos MySQL e implemetación dos programas de comunicación.

- Cliente
- Balanceador-Replica

3.1 Cliente

Este programa esta desarrollado para comunicarse con el balanceador por medio de sockets. Utiliza metodos para enviar y recibir mensajes con formato : "id-cliente;#operacion;mensaje".

3.2 Balanceador - Replica

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

3.3 Base de datos

MySQL es un sistema de gestión de base de datos (SGBD) de código abierto. El SGBD MySQL pertenece actualmente a Oracle. Funciona con un modelo cliente-servidor. Eso quiere decir que los ordenadores que instalan y ejecutan el software de gestión de base de datos se denominan clientes.

3.4 Caída de un servicio

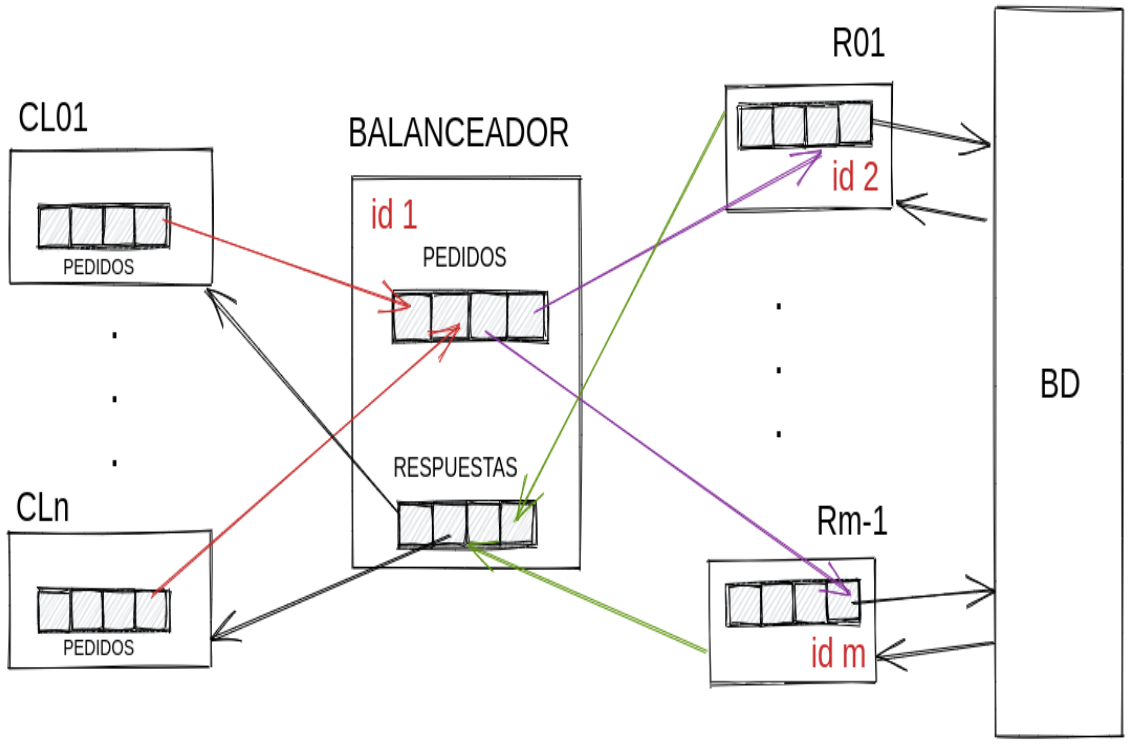


Figure 4: Proceso normal

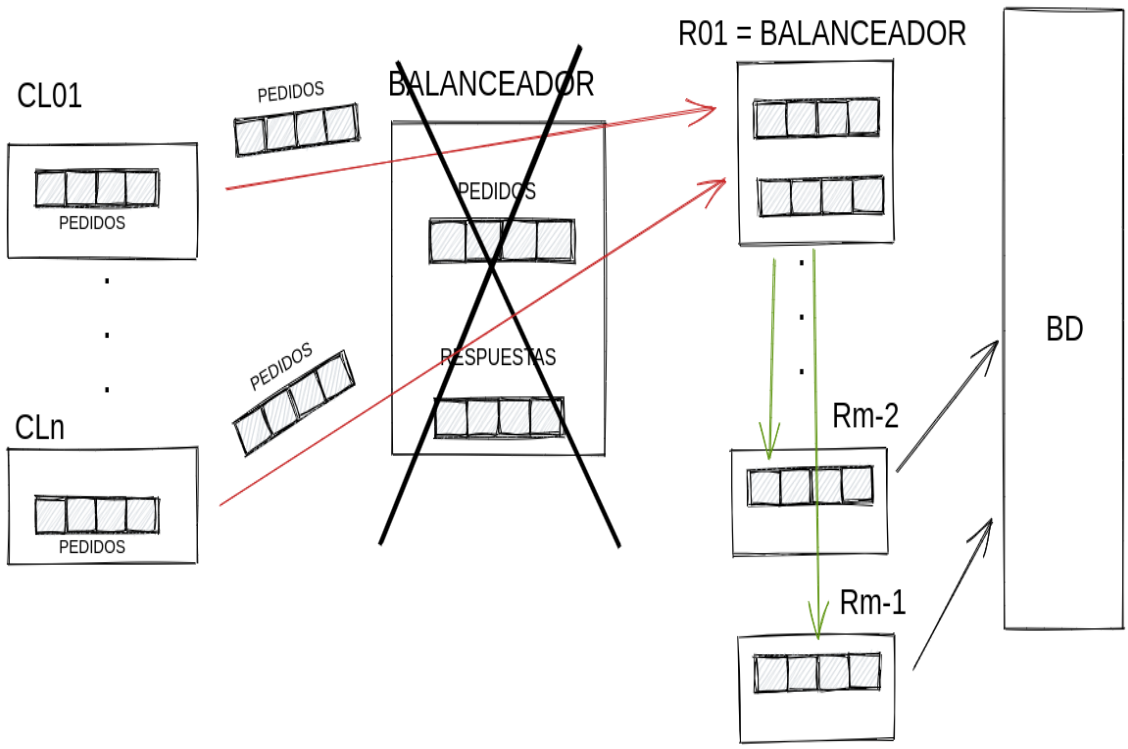


Figure 5: Fallo en el Balanceador

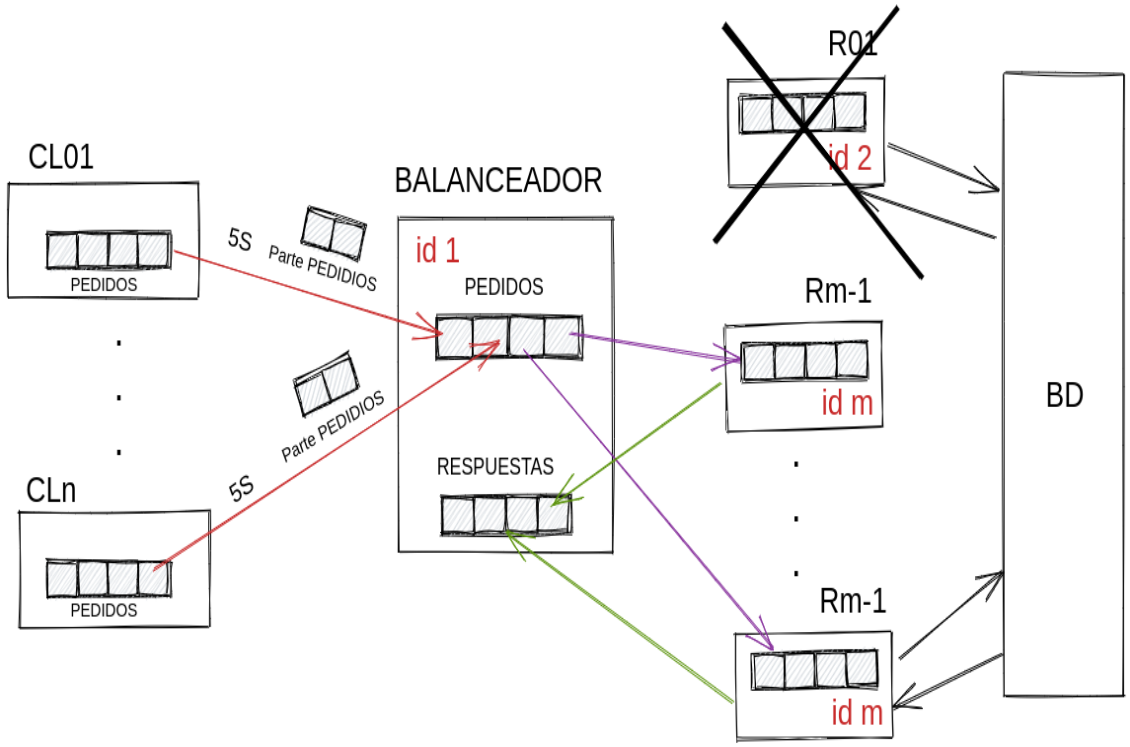


Figure 6: Fallo de la Replica

4 Desarrollo de la Investigación

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

- Cliente
- Balanceador-Replica

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

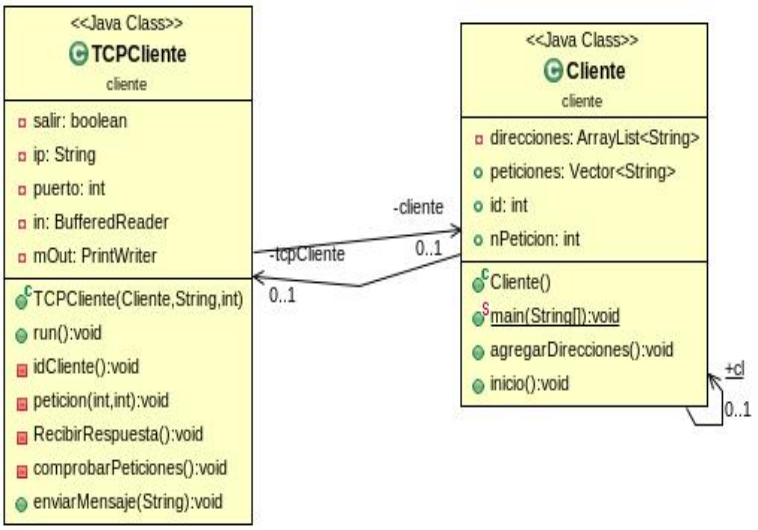


Figure 7: Diagrama de clases de cliente

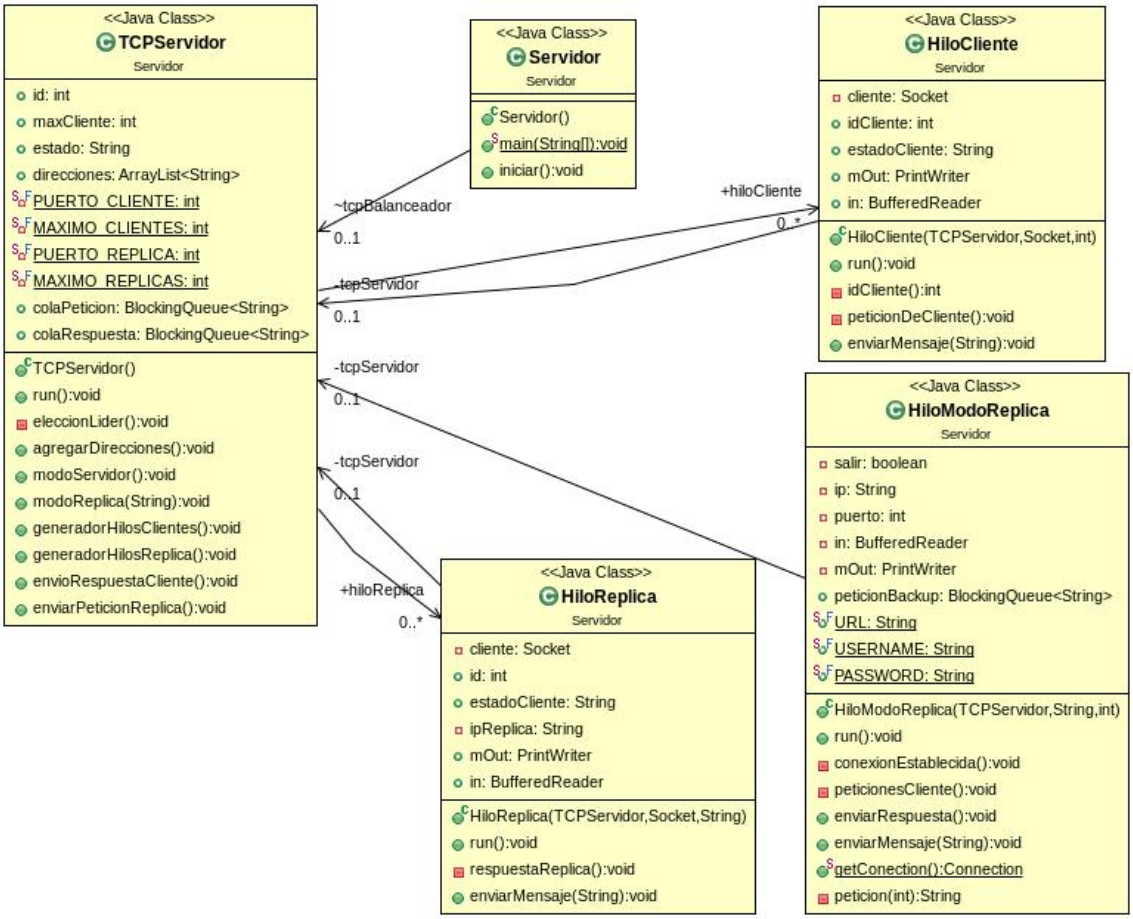
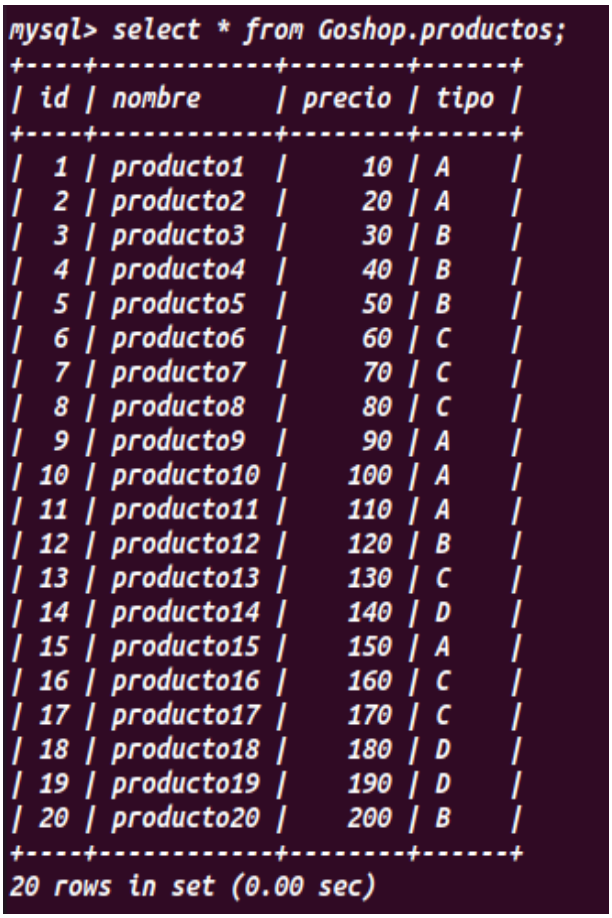


Figure 8: Diagrama de clases de servidor

4.1 Base de Datos

La base de datos relacional Goshop alojada en un servidor MySQL se encuentra en un máquina diferente a las máquinas que hacen de servidores, balanceador y réplica. Ésta contiene la tabla de productos, con los campos ; id, nombre, precio y tipo. Las réplicas harán consultas a esta tabla. Para poder conectarse las replicas con la base de datos; se creo un usuario "replica" con los permisos para poder visualizar todas las tablas. Además de permitir el paso de paquetes TCP por el puerto 3306 de la máquina que contiene la base de datos y cuya ip es 194.168.1.6. Las réplicas asi mismo consultan con un conector jar de mysql especialmente para conexiones java-mysql.



```
mysql> select * from Goshop.productos;
+----+------------+--------+------+
| id | nombre     | precio | tipo |
+----+------------+--------+------+
|  1 | producto1  |     10 | A    |
|  2 | producto2  |     20 | A    |
|  3 | producto3  |     30 | B    |
|  4 | producto4  |     40 | B    |
|  5 | producto5  |     50 | B    |
|  6 | producto6  |     60 | C    |
|  7 | producto7  |     70 | C    |
|  8 | producto8  |     80 | C    |
|  9 | producto9  |     90 | A    |
| 10 | producto10 |    100 | A    |
| 11 | producto11 |    110 | A    |
| 12 | producto12 |    120 | B    |
| 13 | producto13 |    130 | C    |
| 14 | producto14 |    140 | D    |
| 15 | producto15 |    150 | A    |
| 16 | producto16 |    160 | C    |
| 17 | producto17 |    170 | C    |
| 18 | producto18 |    180 | D    |
| 19 | producto19 |    190 | D    |
| 20 | producto20 |    200 | B    |
+----+------------+--------+------+
20 rows in set (0.00 sec)
```

Figure 9: Tabla productos

5 Resultados

El laboratorio fue ejecutado en cuatro maquinas virtuales con las ip's siguientes:

1. 192.168.1.101
2. 192.168.1.102
3. 192.168.1.103
4. 192.168.1.104

Los clientes se conectan al balanceador , si no encuentra uno busca en su lista de ip para encontrar al servidor que hace de balanceador. Los resultados del laboratorio se muestran a continuación:

Figure 10: Primer server conectado

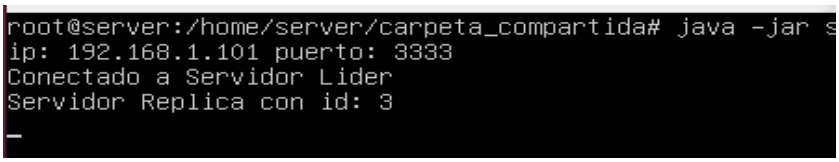


Figure 11: Server 3 escogido como replica

Figure 12: Cliente 1 enviando peticiones

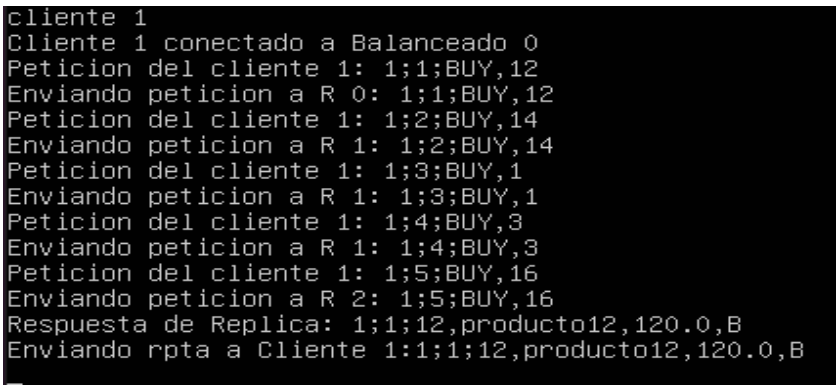


Figure 13: Balanceador recibiendo y enviando a cliente 1

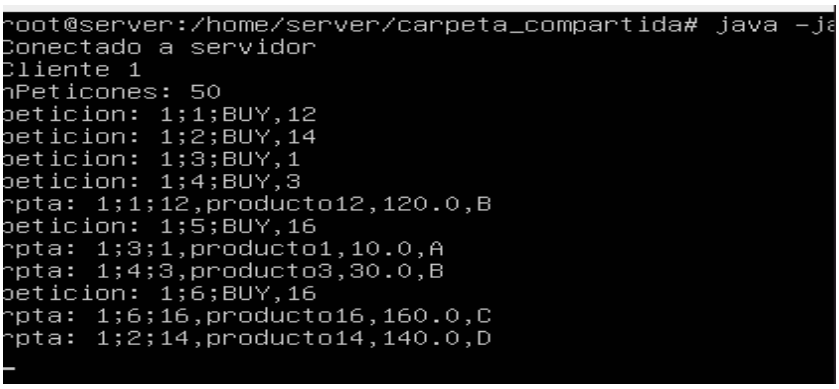


Figure 14: Cliente 1 enviando y recibiendo

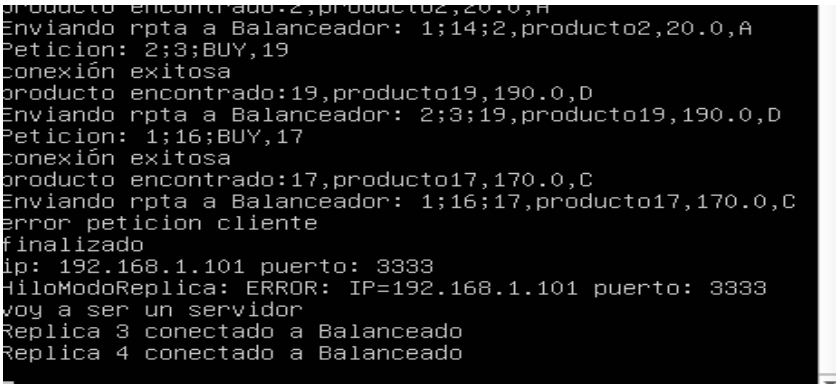


Figure 15: Replica 2 en modo Balanceador

Figure 16: Servidor 2 trabajando como balanceador

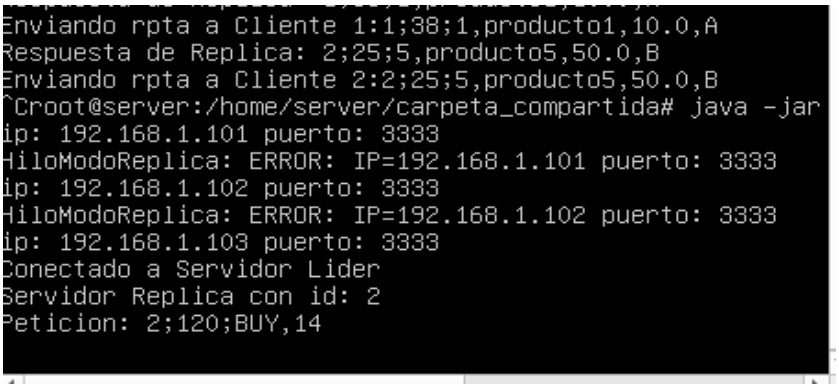


Figure 17: Servidor 2 en modo replica

6 Conclusiones

- Los resultados obtenidos al implementar los conceptos enunciados, permitieron escalar un sistema para adaptarse al crecimiento de los requerimientos del servicio brindado.
- Mostramos que la implementación de replicas-balanceadores soluciona una de las fallas más comunes del modelo cliente-servidor en sistemas distribuidos.
- Se mencionaron algunos conceptos básicos de balance de carga. En este sentido se pueden ampliar las mejoras según los requerimientos de la aplicación en el futuro verificando los resultados obtenidos con otras alternativas de distribución de carga a la propuesta.

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
7. https://sites.google.com/site/mrtripus/home/sistemas-operativos-2/2-1-comunicacion-comunicacion-con-cliente-servidor-comunicacion-con-llamada-a-procedimiento-remoto-comunicacion-en-grupo-tolerancia-a-fallos
8. https://dev.mysql.com/doc/ , documentación de mysql