*PONTIFICIA UNIVERSIDAD JAVERIANA*
*FACULTAD DE INGENIERIA*
*DEPARTAMENTO INGENIERIA DE SISTEMAS*

**Introducción a Sistemas Distribuidos – Proyecto**
**Período Académico 2022-3**

**Entrega y Sustentación – Semana 10**

**Balanceador de cargas**

**Objetivos Generales**

- Poner en práctica conceptos de sistemas distribuidos en un problema práctico del mundo real.
- Implementar, mediante el uso de la librería ZMQ, una aplicación funcional, utilizando patrones de comunicación síncronos y asíncronos.
- Afianzar los conceptos de control de control de fallos y concurrencia de un Sistema Distribuido
- Realizar métricas de desempeño y calidad asociados a la implementación de un sistema distribuido.
- Analizar el rendimiento de una aplicación en un entorno distribuido

**Objetivo Específico**

Apropiar el uso de ZMQ, a través de la implementación de un sistema de balanceo de carga por software, para distribuir las cargas recibidas por parte de diversos clientes, realizando un manejo equilibrado con el algoritmo Round Robin.

**Descripción del Sistema a desarrollar**

El proyecto se focalizará en la implementación y uso de mecanismos de invocación de métodos remotos, para ofrecer un servicio de balanceo de cargas en un sistema distribuidos de una tienda de comercio electrónico. Las conexiones se harán utilizando la librería ZMQ y los patrones de comunicación cliente/servidor y publicador/suscriptor.

**Contexto**
Un balanceador de carga es un proxy inverso que se sitúa delante de un conjunto de servidores y gestiona las solicitudes entrantes, con el fin de repartir el trabajo solicitado por diferentes clientes entre diferentes servidores. Los propósitos de un balanceador de cargas son, entre otros, gestionar la carga entrante para distribuirla de forma equitativa entre los diferentes servidores, permitir mayor eficiencia de los dispositivos de seguridad al enfrentar ataques virtuales, ofrecer transparencia a los usuarios en caso de fallas en alguno de los servidores.

El proyecto consiste en desarrollar e implementar un balanceador de cargas que permita hacer la distribución correspondiente entre diferentes servidores de las peticiones de compra de los clientes de una tienda virtual. Los usuarios aceden al servicio web a través de una única URL, asociada a una dirección IP.

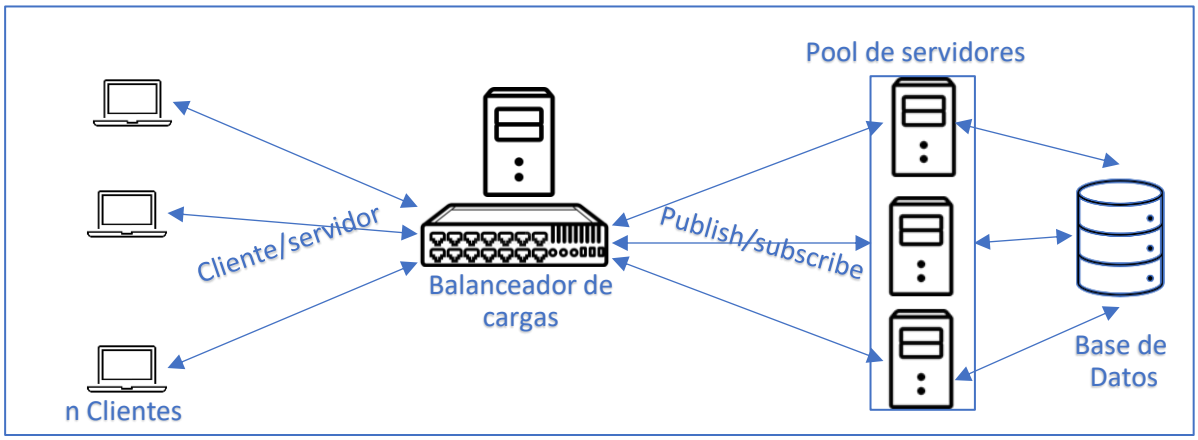


Fig. 1. Arquitectura general del sistema de tienda virtual

**Clientes**

Los clientes pueden realizar operaciones de consulta o compra de artículos. En la operación de consulta se le debe mostrar al cliente los artículos disponibles y con base en esta información el cliente decide que artículos comprar. Los clientes se pueden simular utilizando hilos con el fin de realizar las operaciones de consulta o compra de artículos y pueden provenir de una única máquina física. La comunicación entre los clientes y el balanceador de cargas debe hacerse por medio del **patrón cliente/servidor** de la librería ZMQ.

**Balanceador de cargas**

El balanceador de cargas recibe las peticiones de cada cliente y las asigna a los servidores de acuerdo con el algoritmo Round Robin. La comunicación entre el balanceador de cargas y los servidores debe hacerse por medio del **patrón publicador/suscriptor** de la librería ZMQ[1]. El balanceador de cargas recibe las respuestas procesadas de los servidores y las entrega a los clientes.

**Servidor**

Recibe las solicitudes del balanceador de cargas y realiza la operación correspondiente (Lectura o escritura) de la base de datos y la envía al balanceador de cargas. Después de que el servidor realiza la operación correspondiente la comunica al balanceador de cargas.

**Base de datos**

En la base de datos se mantiene el estado de cada uno de los artículos de la tienda virtual y la información debe ser consistente. Se debe tener en cuenta que varios clientes pueden solicitar el mismo artículo al mismo tiempo (concurrencia) y en algún momento no se podría satisfacer la demanda. Si esto ocurre, se le debe informar al usuario y mostrarle nuevamente los artículos disponibles y sus cantidades.

Tolerancia a Fallas

Se debe implementar una arquitectura tolerante a fallas, considerando una posible falla de uno de los servidores, de manera que debe existir un proceso encargado de chequear que todos los monitores están o no en funcionamiento (health check). Cuando se detecte la falla, se debe asignar esta carga a otro servidor de acuerdo con el algoritmo Round Robin de forma automática, ofreciendo transparencia al usuario, quien no se dará cuenta de lo ocurrido.

[1] https://zeromq.org/socket-api/#publish-subscribe-pattern

**Sustentación**

El día de la sustentación es importante que se pueda observar:

- Operaciones que van realizando cada uno de los procesos y resultado de la operación.
- Información guardada en la BD

**RENDIMIENTO DEL SISTEMA**

**Variables para medir el rendimiento del sistema**

1. Se deben definir variables que permitan medir el rendimiento del sistema, por ejemplo: tiempos de almacenamiento en la base de datos, tiempo que tarda en llegar una solicitud a los servidores (desde el cliente hace una solicitud, hasta que recibe la respuesta), utilización de los procesadores, etc. Debe definir al menos dos tipos de variables.

2. Establezca el procedimiento para tomar el valor de dichas variables, por ejemplo, va a instrumentar el programa, va a usar ciertas herramientas especiales de monitoreo, ¿cuáles serán?, etc.

**Elementos que afectan el rendimiento del sistema**

Existen muchos elementos que afectan el desempeño del sistema, por ejemplo: la carga de trabajo, los patrones de comunicación utilizados, la congestión en la red o en los servidores, etc.

La siguiente actividad que Ud. va a realizar es incorporar un elemento a su sistema que afecte su desempeño. Llamaremos sistema A al sistema sin este elemento (el sistema inicial que implementó) y sistema B, aquel que tienen un elemento que afecta su desempeño. A continuación, se dan ejemplos de factores que pueden afectar el rendimiento:

- Aumentar la carga: Número de peticiones por parte de los clientes para verificar el comportamiento del balanceador de cargas y generar las gráficas correspondientes.
- Patrones de comunicación: en lugar del patrón publicador-suscriptor usar otro patrón como el push/pull de la librería (https://programmerclick.com/article/915097237/) para la comunicación de los clientes al balanceador de cargas.
- Congestión en el balanceador de carga: genere números elevados de peticiones para medir el comportamiento.

**Comparación de los sistemas**

Una vez que el grupo decida las variables a medir, cómo las va a medir y qué elemento va a incorporar para afectar el rendimiento del sistema, realice los siguientes pasos del método científico para comparar el sistema A con el sistema B

- Plantee una hipótesis.
- Realice medidas (experimentación) para comparar el sistema A con el B
- Analice los resultados
- Describa las conclusiones

A continuación, algunos enlaces que explican los diferentes pasos del método científico:

- https://www.todamateria.com/pasos-del-metodo-cientifico/, https://www.universidadviu.com/int/actualidad/nuestros-expertos/cuales-son-las-fases-del-metodo-cientifico.

**PRIMERA ENTREGA**

La primera entrega se realizará en la semana 10 a través de la plataforma. Las sustentaciones serán el día correspondiente a la asignatura de acuerdo con el grupo.

La primera entrega consta de un informe donde se debe especificar:

- Utilizar como guía el modelo de vistas de arquitectura 4+1[2]
- Diseño del sistema: Este diseño debe incluir el o los componentes para enmascarar las fallas del sistema.
- Modelo de Interacción, de fallos y de seguridad.
    - Para el modelo de seguridad se sugiere utilizar como mínimo control de acceso lógico (Usuario y contraseña), las contraseñas se deben almacenar en **memoria permanente** utilizando una función hash.
- El protocolo de pruebas que utilizará para la entrega final
- Describa cómo afectará el rendimiento de su sistema y cómo va a comparar el rendimiento de los sistemas A y B: hipótesis, variables a medir y herramientas de medición.
- Implementación inicial: Se debe presentar una implementación preliminar en el que haya algún tipo de comunicación entre las máquinas.

Se deben implementar los requerimientos de acuerdo con el enunciado y la rúbrica.

El día de la sustentación, cada equipo tendrá como máximo 15 minutos para mostrar sus resultados. Posteriormente responderá las preguntas que tengan los profesores.

**SEGUNDA ENTREGA**

La entrega se realizará en la semana 18 a través de la plataforma. Las sustentaciones se harán los días que correspondan para cada curso. El día de la sustentación los integrantes del equipo deben mostrar la funcionalidad del proyecto. Deben estar presentes todos los integrantes del grupo.

La entrega se compone de:

- En un archivo .zip código fuente de los programas que conforman el sistema y un archivo Readme donde indique cómo ejecutarlo.
- Se debe complementar la documentación de la primera entrega
- Un video de máximo 10 minutos donde muestra la topología implementada y explicar los siguientes aspectos de su proyecto:
    a. Distribución de componentes en máquinas. **Se debe evidenciar que utilizan direcciones IP diferentes**
    b. Librerías y patrones usados
    c. Cómo se implementó la arquitectura confiable (tolerante a fallas)
- Un informe de máximo 4 páginas donde explican: hipótesis, experimentos realizados (especificaciones de hw y sw donde se realizaron las medidas, variables e instrumentos de medición), resultados obtenidos acompañados por gráficos y/o tablas, conclusiones.

Equipos de Trabajo

[2] https://es.wikipedia.org/wiki/Modelo_de_Vistas_de_Arquitectura_4%2B1

El proyecto se realizará en grupos de trabajo de máximo tres personas

No puede existir replicación de documentos ni de código fuente entre grupos, lo cual se consideraría plagio.