UNIVERSIDAD DE
COSTA RICA

Facultad de Ingeniería

Escuela de Ciencias de la Computación e Informática

Proyecto Integrador de Sistemas Operativos y Redes de Comunicación

CI-0123

Grupo 01

Diseño del balanceador de carga

Equipo Andrómeda:

Tatiana Briones López

María Chaves Salas

Johnn Hernández

Eric Ríos Morales

I ciclo 2019

Introducción

El propósito de este proyecto es crear un balanceador de carga, encargado de dividir de manera equitativa la carga de trabajo que se le asigna a cada servidor, esto con el fin de mejorar el desempeño de cada uno de los servidores al aprovechar de una buena manera la capacidad de estos, con la ventaja de disminuir el tiempo de respuesta a cada una de las solicitudes que se le manden a estos.

Dicho proyecto estará implementado en el lenguaje de programación Python. El motivo por el cuál se escogió este es su simplicidad al momento de programar, al ser un lenguaje de alto nivel permite que la forma de aprender a usarlo sea fácil, además de que es sumamente versátil. Además Python cuenta con bibliotecas estándar muy amplias, que son de gran utilidad para aspectos que este proyecto necesita.

Además, como se utilizará un Raspberry Pi para que el sistema funcione, este tiene un sistema operativo llamado Raspbian que contiene muchas herramientas de desarrollo que utilizan Python.

Una parte importante del balanceador es ver la manera en la que la carga se distribuirá a cada uno de los servidores. Para el propósito de este proyecto se escogió el algoritmo de “Round Robin”, este consiste darle a cada servidor funcional cierto intervalo de tiempo para que reciba, procese y transmita la información. Las peticiones son distribuidas entre los servidores de forma cíclica, y con este tipo de rotación se le permite al servidor que tenga cierto tiempo para descansar y que no le caigan todas las peticiones al mismo momento.

Diseño del balanceador de carga

I. Cliente

Para propósitos de prueba, el cliente o los clientes realizan constantes solicitudes web a la IP del balanceador, la única a la que tienen visibilidad. Para simular de manera sencilla estas solicitudes, el cliente se implementa utilizando la biblioteca de Python llamada Requests.

A lo largo del desarrollo del balanceador se usa un número distinto de clientes y distintos tipos de solicitudes para probar las capacidades del producto, acorde a los requerimientos a cumplir.

II. Balanceador

Es el componente fundamental del modelo. Se implementa en el lenguaje Python, aprovechando cuanto se pueda las bibliotecas ya existentes que simplifican la implementación de sus funciones.

II-A. Comunicación entre cliente y servidor

El balanceador de carga debe sacar provecho de los recursos de su red privada, en este caso de todos los servidores que tenga disponibles, de manera que mejoren significativamente la disponibilidad del servicio y los tiempos de respuesta. Partiendo de la suposición de que las solicitudes pueden responderse en un tiempo similar, el algoritmo de *Round Robin* es adecuado, pues asegura que todos los servidores están siendo aprovechados y que reciben carga a un ritmo semejante.

Para llevar control de las solicitudes enviadas por clientes y los servidores a los que fueron asignadas, el balanceador activo maneja una tabla de direccionamiento. El tamaño de esta tabla varía acorde a la cantidad de servidores disponibles en un momento dado, y con el objetivo de reusabilidad funciona similar a una lista circular. Cada entrada en la tabla tiene los siguientes espacios:

- **Identificador de conexión:** Valor entero que se añade a los paquetes enviados en la red privada, y permite que, en caso de que las respuestas de un servidor llegan en desorden, estas se envíen al cliente correcto.
- **Cliente:** Contiene la información para redireccionar la respuesta del servidor.
- **Servidor:** Información del servidor al que fue asignada la solicitud, de acuerdo al algoritmo utilizado. Espacio importante en caso de detectar caída del servidor, para reasignar su carga.

II-B. Manejo de servidores

Una de las características del balanceador de carga es poseer la capacidad de agregar de manera dinámica servidores a su *server pool*, además de detectar con prontitud cualquier posible caída de un servidor, y actuar acordemente en cada una de estas situaciones.

Cuando un servidor quiera ser agregado realiza *broadcast* a toda la red privada, donde el balanceador activo estará escuchando y procederá a modificar el tamaño de la tabla de direccionamiento, agregar el servidor a su *server pool* y enviar una respuesta al servidor indicando que ha sido agregado correctamente. Como se mencionó en la subsección II-A, la comunicación en la red privada incluye un identificador. Dicho identificador tendrá un valor especial para indicar que se trata de un servidor que busca ser agregado, y también para la respuesta del balanceador.

Con el fin de evitar pérdidas en una cola de solicitudes enviadas al servidor, la cola se mantiene del lado del balanceador; en el momento en que llega la respuesta del servidor de la última petición se le envía la siguiente. Esto hace más viable la recuperación en caso de que un servidor deje de funcionar, pues estaba atendiendo una sola solicitud y se tiene un registro de cuál era. Conforme llegan las respuestas del servidor de nuevo al balanceador y se envía hacia el cliente se elimina la entrada de esa conexión en la tabla de direccionamiento, que indica también que la cola del servidor tiene un nuevo espacio disponible.

Para la detección de caídas se hace uso de *timeouts* en las respuestas de cada servidor. Cuando efectivamente se detecte error se procede a reasignar equitativamente la cola de trabajo del servidor caído entre los que queden disponibles, se reduce el tamaño de la tabla de direccionamiento y se saca el servidor del *server pool*.

II-C. Manejo de balanceadores de carga

Es importante considerar que es posible tener varios balanceadores de carga listos para trabajar, de los cuales solamente uno está activo. El objetivo de contar con n balanceadores es evitar un *Single Point Of Failure* en el servicio, que sería el balanceador de carga y poder recuperar el funcionamiento con alguno de los otros disponibles. Es necesario entonces que todos los balanceadores tengan una noción de lo que sucede en el que esté activo, debe ser posible agregar y quitar balanceadores y, lo más importante, que el siguiente balanceador posea la capacidad de recuperar el estado del anterior y mantener el funcionamiento esperado del servicio.

Para mantener la información clave contenida en el balanceador activo, como lo es la tabla de direccionamiento, cada cierto tiempo este se comunica con los pasivos para actualizar dichos datos. Además, el activo lleva una bitácora lo suficientemente detallada como para detectar errores y recuperar cierta información.

Con las comunicaciones periódicas entre balanceadores de carga, los pasivos monitorean que el activo sigue en funcionamiento. Cuando un error sea detectado por los pasivos, el siguiente balanceador toma control de la IP del anterior y continúa atendiendo solicitudes de clientes. Quien es ahora el balanceador activo tiene una versión reciente de la tabla de direccionamiento, colas de servidores, entre otros, pero muy difícilmente sea el último estado del anterior. Es aquí cuando entra más activamente la bitácora que mantuvo el otro balanceador, pues es posible revisar (con la ayuda de scripts) las entradas más recientes y actualizar la información como corresponde.

II-D. Hilos de ejecución

Considerando que cada balanceador de cargas será un proceso independiente, se busca la mayor eficiencia del servicio haciendo uso de varios hilos de ejecución con distintas tareas.

- **Receptor de clientes:** Escucha las solicitudes entrantes y las pasa al procesador de solicitudes nuevas.
- **Procesador de solicitudes nuevas:** Asigna la nueva solicitud acorde al algoritmo utilizado, agrega a la tabla de direccionamiento y añade en la cola del servidor o envía directamente.
- **Receptor de respuestas del servidor:** Con cada respuesta entrante hace uso de la tabla de direccionamiento para enviar al cliente que corresponde, elimina la entrada de la tabla y si existen más solicitudes en la cola de servidor envía la siguiente. Cuando un servidor comunica que quiere ser agregado actúa en consecuencia.
- **Comunicador con balanceadores de carga:** Se encarga de actualizar la información en los demás balanceadores cuando corresponde. Del lado de los pasivos, es quien monitorea el funcionamiento del activo.
- **Manejador de errores:** Se activa en caso de errores, ya sea caída de servidor o de balanceador activo, y su fin es recuperar el funcionamiento del sistema sin detener las demás operaciones.

Como se hace uso de varios hilos de ejecución es crucial el manejo correcto de zonas críticas, como lo son la tabla de direccionamiento o colas de trabajo.

III. Servidores

En el caso de la implementación de los servidores, se hace uso de Apache, con varios dominios web. Una vez está configurado correctamente dentro de la red privada, hace *broadcast* a toda la red para indicar que está disponible. En cuanto recibe la confirmación del balanceador de cargas se prepara para atender solicitudes.

IV. Raspbian

Otra de las funcionalidades primordiales del sistema es ser capaz de ejecutarse cuando se inicia el Raspberry Pi y además ser capaz de recuperarse de manera automática sin necesidad de intervenir manualmente. Es por esto que se considera agregar el producto como un servicio del sistema operativo Raspbian. Una forma de implementar esta funcionalidad es mediante el uso de *scripts* que logren inscribir al sistema como un servicio más.

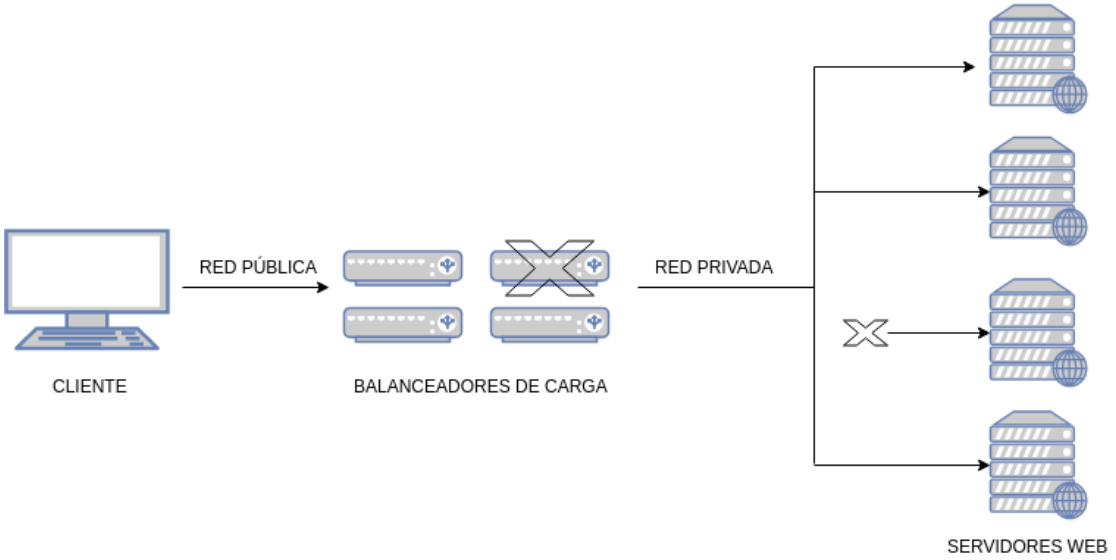

Figura 1. Vista general del modelo

Cronograma de fechas y requerimientos

Fecha	Tarea o entrega a realizar
21 de mayo	Diseño del proyecto
Sin definir	Balanceador solo como intermediario entre un cliente y un servidor
Sin definir	Balance entre cantidad estática de servidores
4 de junio	Conexión de cliente con balanceador y balanceador con servidor (como servicio recuperable)
Sin definir	Varios clientes
Sin definir	Agregar dinámicamente servidores y balanceadores
25 de junio o 2 de julio	Recuperación ante caídas de servidores o balanceadores
4 de julio	Entrega final