Proxy Inverso + Balanceador de Carga (PIBL)
Proyecto I
Telemática

1. Objetivo

Desarrollar habilidades en la programación de aplicaciones distribuidas, particularmente las que requieren de una arquitectura cliente/servidor utilizando la API sockets.

2. Introducción

Los proxies inversos, así como balanceadores de carga son elementos intermedios en el funcionamiento y operación de aplicaciones cliente/servidor.

En este sentido, un proxy inverso se entiende como aquel que recibe (intercepta) cada una de las peticiones del cliente y la envía a un servidor con la capacidad de procesar la petición para finalmente enviar la respuesta al cliente. Por otro lado, un balanceador de carga, es la entidad encargada de distribuir las peticiones entrantes por parte de los clientes hacia un conjunto de servidores. Para cada petición, debe posteriormente, debe retornar la respuesta al cliente.

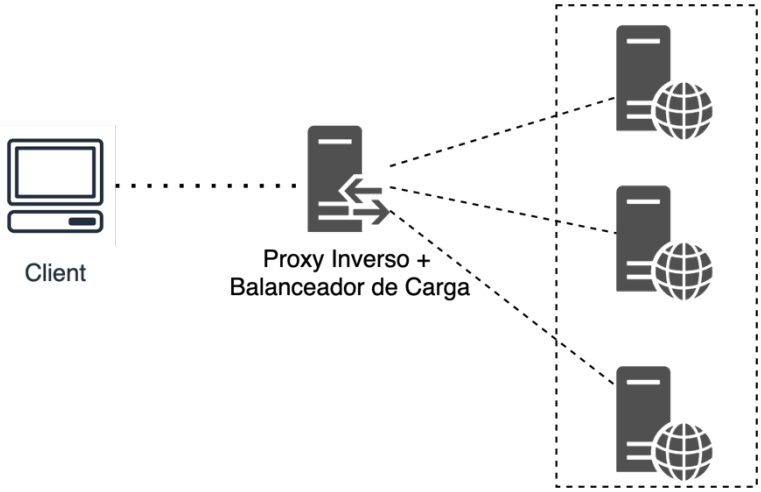
3. Arquitectura a Desplegar.

En la figura que se observa a continuación, se presenta la arquitectura de alto nivel que usted debe implementar para lograr el objetivo de la práctica.

La solución a implementar debe contar con los siguientes componentes:

- El servidor PIBL.
- Tres servidores de aplicación web. La aplicación se debe replicar en los tres servidores, es decir, es la misma. Para efectos de esta práctica, usted tiene la potestad de realizar su aplicación, para efectos de prueba, en cualquier lenguaje de programación.

Como se aprecia, se requiere que usted disponga de una máquina



4. **Recursos**:

   Toda la arquitectura detallada en el ítem anterior debe ser desplegada en la nube de Amazon Web Services utilizando la cuenta que se le ha asignado en el curso. Para esto debe utilizar emplear instancias EC2 para la instalación y configuración de su solución.

5. **Aspectos por considerar para el desarrollo de la práctica:**

   En este proyecto se requiere que usted implemente un Proxy Inverso + Balanceador de Carga (PIBL), que permita recibir las peticiones web del cliente, las procese, las envié a uno de los tres (o más) servidores que se encuentran detrás de éste para finalmente retornar la respuesta al cliente. Para lograr esto se requiere que usted considere los siguientes aspectos:

   1. Su PIBL debe ser escrito en lenguaje de programación C, Rust, Python. Debe justificar la elección de su lenguaje de programación.
      a. Si su elección es Python, debe trabajar la versión 3.X y solo puede emplear los siguientes “import”: sys, os, time, socket.
   2. Se debe emplear la API Sockets.
   3. Se debe soportar peticiones de forma concurrente desde diferentes tipos de clientes que envíen peticiones HTTP. Por favor debe implementar una estrategia para lograr esto.
      a. Dependiendo de la estrategia que escoja, para el lenguaje Python agregue a la lista del literal 1 el import a utilizar.
   4. Se requiere que se procesen peticiones para la versión de protocolo HTTP/1.1.
   5. Se requiere que su servidor escuche peticiones en el puerto 8080. Una vez reciba la petición de un cliente (p.ej., browser, terminal de consola, postman, etc), se debe iniciar un nuevo socket cliente para comunicarse con el servidor web destino elegido, y enviar la petición a éste.
   6. Una vez envié la petición al servidor, debe esperar la respuesta y enviarla al cliente web que la solicito que solicite el recurso web.

7. Se requiere que la aplicación PIBL implemente un proceso “log” donde se registren todas las peticiones que recibe. En este sentido, el log debe permitir registrar todas las peticiones que se reciben y debe visualizar la petición que se hace y la respuesta que se entrega. Esto se debe visualizar por la salida estándar, y de igual forma, se debe implementar el registro en un archivo.
8. La función de proxy debe permitir el caché para los diferentes recursos que se soliciten por parte de los clientes. Para esto debe considerar lo siguiente:
    a. Para todos los recursos solicitados en las peticiones hecha por los clientes, la respuesta (el recurso solicitado) debe ser almacenada en en un archivo en el disco del servidor. De esta forma se garantiza que el cache persista en caso tal se presente una falla en el servidor PIBL. Así, la próxima vez que se realice la petición de este recurso, se debe acceder desde al disco y enviar la respuesta desde aquí hacia el cliente.
    b. Los recursos para almacenar en el cache deben ser localizados en el directorio donde se ejecuta la aplicación principal del PIBL.
    c. Se debe implementar un mecanismo para implementar un TTL para cada recurso que se mantenga en el cache. Esto debe ser un parámetro que se pase al momento de lanzar la aplicación.
9. Para efectos de distribución de la carga de las peticiones, la política a implementar es Round Robin.
10. Su PIBL debe tener un archivo de configuración que permita parametrizar el puerto en el que se ejecuta (p.ej., por defecto es 8080) así como incluir la lista de servidores (backend) que contestan las peticiones.

6. **Grupos de trabajo:**
    a. El proyecto 1 debe ser desarrollado en grupos de tres personas.
7. **Fechas:**
    a. **Fecha de inicio:** 26/27 de septiembre.
    b. **Fecha de entrega:** viernes 28 de octubre hasta las 23:59.
    c. **Sustentaciones:** A partir del viernes 29 de octubre. Cada equipo dispondrá de 35 minutos para la presentación de los detalles de su proyecto.
8. **Mecanismo de Entrega:**
    a. La entrega se debe realizar por el buzón de entrega por interactiva virtual. La documentación se debe incluir en el repo en un archivo README.md. En este archivo se requiere que usted incluya los detalles de implementación donde como mínimo se esperan las siguientes secciones:
        i. Introducción.
        ii. Desarrollo
        iii. Conclusiones
        iv. Referencias
9. **Versión:**
    a. **Fecha de Creación:** septiembre 26.
    b. **Fecha de primera actualización:** octubre .