Modulo 2: Balanceador de Carga (Round Robin)

Este documento detalla la implementacion de un simulador de balanceador de carga que utiliza el algoritmo Round Robin, de acuerdo con los requisitos del ejercicio.

---

1. Descripcion del Algoritmo Round Robin

El algoritmo Round Robin es una de las tecnicas mas sencillas y utilizadas para la distribucion de carga entre un conjunto de servidores. Su funcionamiento se basa en un principio de rotacion secuencial y equitativa.

Cuando una nueva solicitud de cliente llega al balanceador, este la asigna al primer servidor de su lista. La siguiente solicitud se envia al segundo servidor, y asi sucesivamente, recorriendo la lista de forma ordenada. Una vez que se ha asignado una solicitud al ultimo servidor de la lista, el ciclo vuelve a comenzar desde el primer servidor.

Este metodo no tiene en cuenta la carga actual o la capacidad de procesamiento de los servidores; los trata a todos como si fueran identicos. Su principal ventaja es la simplicidad de implementacion y la garantia de que, en condiciones de flujo constante de solicitudes, la carga se distribuira de manera uniforme a lo largo del tiempo.

2. Justificacion del Modelo Implementado

Para la implementacion en Python, se opto por un modelo que combina la programacion orientada a objetos con estructuras de datos eficientes, priorizando la claridad y el rendimiento.

Clase Servidor: Se creo una clase para representar a cada servidor, encapsulando su estado individual, como el nombre y el contador de solicitudes recibidas. Esto facilita la gestion y el seguimiento de las estadisticas por servidor.

Modelo de Lista Circular con collections.deque: En lugar de implementar una lista enlazada circular desde cero, se utilizo el objeto deque (cola de dos extremos) de la libreria estandar de Python. Esta eleccion se justifica por las siguientes razones:

- Eficiencia: deque esta implementado en C y ofrece operaciones de anadir y eliminar elementos de ambos extremos con una complejidad de tiempo de O(1).
- Funcionalidad rotate(): El metodo deque.rotate(-1) permite mover el elemento del principio al final de la cola en una sola operacion atomica y altamente optimizada (O(1)). Esto simula perfectamente el comportamiento circular del algoritmo Round

Robin de una manera limpia y Pythonica, ya que el servidor que acaba de recibir una solicitud se mueve al final del turno.

- Simplicidad: El uso de deque reduce la cantidad de codigo necesario y evita la gestion manual de punteros, lo que disminuye la probabilidad de errores.

Este enfoque cumple con la recomendacion de usar una Circular Linked List a nivel conceptual, pero aprovechando las herramientas optimizadas que ofrece el propio lenguaje.

3. Complejidad del Proceso de Asignacion

La complejidad computacional del proceso de asignacion de una unica solicitud en el modelo implementado es de O(1), es decir, tiempo constante.

El analisis se desglosa de la siguiente manera:

1. Seleccion del Servidor: Obtener el proximo servidor al que se le asignara la solicitud implica acceder al primer elemento de la deque (servidores[0]). Esta es una operacion de tiempo constante, O(1).
2. Procesamiento de la Solicitud: Incrementar el contador de solicitudes del servidor seleccionado es una operacion aritmetica basica, tambien de tiempo constante, O(1).
3. Rotacion de la Cola: Mover el servidor utilizado al final de la cola usando servidores.rotate(-1) es la operacion clave que simula el Round Robin. Como se menciono anteriormente, esta operacion tiene una complejidad de tiempo constante, O(1), independientemente del numero de servidores en la cola.

Dado que todos los pasos involucrados en la asignacion de una solicitud se ejecutan en tiempo constante, la complejidad total del proceso no depende del numero de servidores ni del numero total de solicitudes procesadas. Esto hace que el algoritmo Round Robin sea extremadamente rapido y escalable para gestionar un gran volumen de trafico.