Memoria

Alumno: Álvaro Pereda Sánchez.

Funcionamiento del proyecto

Stack Prime está diseñado con una arquitectura distribuida compuesta por dos componentes principales: un coordinador (coordinator) y múltiples trabajadores (workers).

- El coordinador recibe un número proporcionado por el usuario y lo divide en partes iguales, asignando un rango específico a cada trabajador.
- Cada trabajador recibe su rango mediante una petición POST enviada por el coordinador, y se encarga de calcular los números primos dentro de ese rango.
- Esta división de tareas permite paralelizar el trabajo, optimizando así el tiempo de procesamiento.
- Una vez que los trabajadores terminan sus cálculos, envían los resultados nuevamente al coordinador mediante otra petición POST.
- Finalmente, el coordinador recoge todas las respuestas y presenta los resultados al usuario.

Tecnologías usadas

Stack Prime utiliza las siguientes tecnologías para su desarrollo y ejecución:

- ### Docker

Docker se emplea para la gestión de contenedores, que proporcionan entornos virtualizados ligeros con todas las dependencias necesarias para la ejecución del proyecto. Esto permite una portabilidad sencilla del proyecto, sin preocuparse por errores de compatibilidad entre diferentes sistemas.

Se utiliza un archivo Dockerfile para generar una imagen personalizada que incluye las librerías de Python requeridas por el proyecto.

- Python

El lenguaje principal del proyecto es Python, por su simplicidad y amplia disponibilidad de librerías. Algunas librerías destacadas son:

- **Flask**: Permite crear rutas web de manera sencilla, sin necesidad de utilizar frameworks más complejos.
- **ThreadPoolExecutor**: Se utiliza para lanzar peticiones paralelas a los trabajadores, lo que mejora el rendimiento al evitar esperas secuenciales.

- Docker Swarm

La tecnología más importante en Stack Prime es Docker Swarm, una herramienta de orquestación de contenedores. Su lógica se define en el archivo docker-compose.yaml, donde se especifican:

- Las imágenes que deben montarse.
- La creación de una red interna que facilita la comunicación entre contenedores mediante DNS interno (sin necesidad de especificar direcciones IP manualmente).
- Variables de entorno necesarias.
- La cantidad de réplicas para cada contenedor.

Gracias a Docker Swarm, se pueden tener N trabajadores activos simultáneamente. Si uno de los contenedores falla, Docker Swarm lo reemplaza automáticamente, garantizando así la disponibilidad del sistema.

Además, Swarm gestiona la distribución de las peticiones del coordinador hacia los trabajadores. Si una réplica está ocupada, la petición se redirige automáticamente a otra réplica disponible, permitiendo un reparto eficiente del trabajo sin necesidad de una lógica compleja en el código.

Diferencias de Docker Swarm y Docker Compose

Aunque Docker Compose y Docker Swarm tienen similitudes, ambos permiten gestionar múltiples contenedores y redes internas, presentan diferencias clave:

- Docker Compose es ideal para entornos de desarrollo, mientras que Docker Swarm está orientado a producción.
- Con Docker Swarm se pueden escalar los servicios fácilmente, permitiendo tener un número variable de trabajadores.
- En caso de caída de un contenedor, Docker Swarm lo reinicia automáticamente, mejorando la tolerancia a fallos.
- Gracias a su sistema de orquestación, Docker Swarm facilita el balanceo de carga entre réplicas activas, sin necesidad de implementar esta lógica manualmente.