UNIDAD DE INGENIERÍA EN COMPUTACIÓN
Sede Central Cartago
Principios de Sistemas Operativos
Profesor: Kenneth Obando Rodríguez

TEC | Tecnológico de Costa Rica

Proyecto 2 (%) Fecha de Entrega: 12 de junio, 2024

Servidor HTTP Distribuido

1 Objetivo del Proyecto

El objetivo de este proyecto es ampliar el servidor HTTP desarrollado en el Proyecto 1, integrando procesamiento distribuido mediante la coordinación entre múltiples instancias del mismo servidor HTTP ejecutándose en contenedores Docker independientes. El sistema debe ser capaz de distribuir tareas computacionalmente intensivas, detectar caídas de workers mediante healthchecks periódicos y redistribuir tareas automáticamente, fomentando así el desarrollo de habilidades en diseño de sistemas distribuidos, concurrencia, tolerancia a fallos y despliegue en ambientes aislados.

2 Descripción General

Cada estudiante deberá utilizar el servidor HTTP desarrollado en el Proyecto 1 como base para implementar múltiples workers. Estos workers serán coordinados por un componente central llamado **Dispatcher**, que recibe solicitudes desde clientes y las redirige a los workers disponibles, monitorea su estado y redistribuye tareas en caso de fallo.

Se deberá implementar al menos dos problemas computacionalmente paralelizables, distribuidos entre los workers. Todo el sistema debe ser ejecutado sobre contenedores Docker.

3 Requerimientos Funcionales

1. **Distribución de Comandos HTTP Existentes:** El Dispatcher debe poder redirigir comandos como `/fibonacci`, `/hash`, `/simulate` a cualquiera de los workers.
2. **Problemas Paralelizables:** Se deben implementar al menos dos de los siguientes:
    - Cálculo de π vía método Monte Carlo
    - Búsqueda de hash con prefijo (Proof of Work)
    - Integración numérica
    - Multiplicación de matrices
    - Conteo de palabras en archivos grandes
3. **Healthcheck y Tolerancia a Fallos:** El Dispatcher debe realizar verificaciones periódicas a los workers mediante el endpoint `/ping`. Si un worker no responde:

    - Se marca como inactivo.
    - Las tareas asignadas se redistribuyen.

4. **Monitoreo de Workers:** Se debe implementar un endpoint `/workers` para mostrar el estado de cada worker (activo, carga, # tareas completadas, etc.).
5. **Redistribución de Tareas:** El Dispatcher debe usar una estrategia como Round Robin o Least Loaded y reintentar tareas fallidas.
6. **Ejecución en Docker:** Cada worker debe ejecutarse como un contenedor Docker que contiene el servidor HTTP del Proyecto 1.
7. **Scripts de despliegue:** Se debe incluir `docker-compose.yml` o scripts equivalentes para lanzar múltiples instancias.

4 Entregables

- Código fuente del Dispatcher y los workers.
- Dockerfiles y scripts de despliegue.
- Implementación funcional de al menos dos problemas paralelizables.
- Evidencia de ejecución distribuida (capturas, logs, pruebas).
- Informe técnico extendido (PDF) con:
    - Arquitectura del sistema distribuido
    - Protocolos de comunicación
    - Análisis de tolerancia a fallos y escalabilidad
    - Resultados comparativos con ejecución local

5 Rúbrica de Evaluación

Criterio	Descripción detallada	Subcriterios / Evidencias esperadas	Pts
Distribución de comandos HTTP	Dispatcher distribuye tareas y responde correctamente	Comunicación funcional, asincronía, retorno correcto	10
Problemas paralelizables	Implementación correcta de 2 problemas distribuidos	División de tarea, paralelización efectiva, combinación de resultados	10
Healthcheck y redistribución	Workers inactivos son detectados, tareas se reasignan	Endpoint `/ping`, reintento de ejecución, robustez ante fallas	10
Monitoreo de Workers	Visualización clara del estado de los workers	Endpoint `/workers`, PID, carga, tareas completadas	10
Contenerización con Docker	Cada worker corre en contenedor separado con el servidor HTTP	Dockerfile funcional, docker-compose, ejecución reproducible	10
Calidad del código y organización	Código limpio, modular, con buenas prácticas	Separación de responsabilidades, control de versiones, nombres claros	10
Pruebas y evidencia experimental	Resultados cuantitativos y cualitativos del sistema distribuido	Comparación local vs distribuido, capturas/logs, escalabilidad	10
Informe técnico	Documento con redacción técnica y formato académico	IEEE/ACM, estructura lógica, gráficos, interpretación de resultados	10
Robustez y manejo de errores	Validación de entradas, control de fallos, sistema no colapsa	Verificación de parámetros, manejo de excepciones, tiempo de respuesta	10
Estilo profesional y buenas prácticas	Estilo de codificación y documentación consistente	Convenciones del lenguaje, comentarios útiles, documentación externa	10
Total			**100**

Cuadro 1
Rúbrica detallada de evaluación del Proyecto 2 desagregada por criterios y evidencias