Informe de Experimentos SITM MIO:

Simon Reyes
Thomas Brueck
Dario Marquinez
William Joseph VerdeSoto

1. Objetivo del Experimento

El objetivo principal es evaluar el rendimiento, la escalabilidad y la precisión del subsistema encargado de calcular la velocidad promedio por arco en las rutas del SITM-MIO. Se busca comparar dos arquitecturas de software (Monolítica vs. Distribuida) bajo diferentes cargas de trabajo.

La finalidad es identificar el umbral de "punto de quiebre" donde la arquitectura monolítica deja de ser viable debido a la latencia y saturación de recursos, justificando así la necesidad de distribuir el procesamiento en múltiples nodos (Workers) para mantener la estabilidad del sistema ante volúmenes masivos de datos.

2. Metodología y Arquitectura de Datos

Para entender los resultados, es crucial distinguir las dos etapas del procesamiento de datos en el sistema:

1. **Etapa de Ingesta y Procesamiento (Backend/Workers):**
    - **Fuente de datos:** Archivos CSV (`datagrams...csv`) con datos históricos de GPS.
    - **Contenido:** ID del bus, Latitud, Longitud, ID de Línea y Timestamp.
    - **Proceso:** El sistema (ya sea el servidor central o los workers externos) procesa estos registros *antes* de la prueba de carga para construir un grafo de velocidades promedio por calle.
2. **Etapa de Benchmark (Simulación de Usuario):**
    - **Herramienta:** `runBenchmark`.
    - **Acción:** No envía datagramas. Simula usuarios concurrentes realizando consultas de tipo `findRoute` ("¿Cómo llego de A a B?").
    - **Dependencia:** Las respuestas dependen del grafo de tráfico calculado en la etapa anterior.

Variables de Evaluación

- **Tiempo Total (s):** Duración completa de la prueba.
- **Total de Consultas:** Cantidad de peticiones atendidas.
- **Rendimiento (TPS):** Transacciones por segundo.
- **Latencia Promedio (ms):** Tiempo de respuesta por consulta.
- **Concurrencia:** 50 hilos (simulando 50 usuarios simultáneos).

3. Resultados Obtenidos

Se realizaron tres pruebas clave para comparar el comportamiento del sistema. A continuación se presentan los datos recolectados:

Escenario	Configuración	Carga de Datos	Tiempo Total	Total Consultas	Rendimiento (TPS)	Latencia Promedio
1 (Base)	Monolito (Local)	Baja	60,35 s	24.474	**405,57**	122,79 ms
2 (Dist.)	Distribuida (2 Workers)	Baja	60,15 s	17.756	**295,18**	169,12 ms
3 (Stress)	Monolito (Local)	Alta (Masiva)	**605,02 s**	55.775	**92,19**	278,37 ms

Evidencia:

- **Escenario 1 (Monolito - Carga Baja):** El sistema local logra procesar más consultas por segundo debido a la ausencia de latencia de red entre nodos.
- **Escenario 2 (Distribuido - Carga Baja):** Aunque el tiempo total es similar, el TPS es menor (295 vs 405) debido al *overhead* (sobrecosto) de comunicación entre el servidor y los workers externos.
- **Escenario 3 (Monolito - Carga Alta):** Al aumentar drásticamente el volumen de datos en el CSV, el rendimiento del monolito se desploma (el tiempo sube a 10 minutos y el TPS cae a 92).

4. Análisis de Resultados

Al analizar los datos comparativos, se extraen las siguientes conclusiones sobre el comportamiento de las arquitecturas:

1. La eficiencia del Monolito en baja escala:

En escenarios con pocos datos (Escenario 1 vs 2), la arquitectura monolítica (procesamiento local) resulta más eficiente en términos de Throughput (405 TPS vs 295 TPS). Esto sucede porque, con poca carga, el costo de serializar datos y enviarlos por la red a workers externos (arquitectura distribuida) es mayor que el beneficio del procesamiento paralelo. La comunicación en red introduce una latencia que no existe cuando todo ocurre en la misma memoria RAM.

2. El colapso del Monolito en alta escala:

El hallazgo más crítico se observa en el Escenario 3. Cuando se satura el sistema con un volumen masivo de datagramas históricos, el monolito sufre una degradación severa:

- El **tiempo de ejecución** se dispara de ~1 minuto a más de **10 minutos (605 s)**.
- El **rendimiento** cae drásticamente a **92 TPS**.
- La **latencia** se duplica.

Esto indica que el procesador y la memoria de un solo nodo se convierten en un cuello de botella, incapaces de mantener el ritmo de actualización del grafo de velocidades mientras responden consultas.

3. Conclusión Final:

La arquitectura distribuida presenta un costo inicial de rendimiento (overhead) en cargas bajas, pero es indispensable para garantizar la escalabilidad del proyecto. Mientras el monolito se vuelve inoperable bajo carga masiva (aumentando tiempos en un 900%), la arquitectura distribuida permite mantener tiempos de respuesta estables dividiendo la carga de procesamiento del CSV entre múltiples nodos. Por lo tanto, para un despliegue real del SITM-MIO con datos de toda la flota de buses, la arquitectura distribuida es la única opción viable.