Sistemas Distribuidos I
(75.74)

Flights Optimizer HA

TP Tolerancia a Fallos

Docentes

- Pablo D. Roca
- Guido Albarello
- Gabriel Robles
- Franco Barreneche
- Tomás Nocetti
- Nicolás Zulaica

Requerimientos Funcionales

- Se solicita un sistema distribuido que analice 6 meses de registros de precios de vuelos de avión p/proponer mejoras en la oferta a clientes.
- Los registros poseen trayectos (aeropuertos origen-destino), tarifa total, distancia total, duración, cada segmento con escalas y aerolíneas.
- Los registros se ingresan progresivamente al ser escaneados de internet.
- Se debe obtener:
  - ID, trayecto, precio y escalas de vuelos de 3 escalas o más.
  - ID, trayecto y distancia total de vuelos cuya distancia total sea mayor a cuatro veces la distancia directa entre puntos origen-destino.
  - ID, trayecto, escalas y duración de los 2 vuelos más rápidos para cada trayecto entre todos los vuelos de 3 escalas o más.
  - El precio avg y max por trayecto de los vuelos con precio mayor a la media general de precios.

Requerimientos No Funcionales

- Para construir una simulación realista se define la serie de datos:
  - https://kaggle.com/datasets/dilwong/flightprices
  - https://kaggle.com/datasets/pablodroca/airports-opendatahub
  - https://kaggle.com/datasets/pablodroca/flight-prices-2M (extract)
- Usar la librería *geopy.distance* para calcular distancias.
- El sistema debe estar optimizado para entornos multicomputadoras
- Se debe soportar el incremento de los elementos de cómputo para escalar los volúmenes de información a procesar
- Se requiere del desarrollo de un Middleware para abstraer la comunicación basada en grupos.
- Se debe soportar una única ejecución del procesamiento y proveer *graceful quit* frente a señales SIGTERM.

Nuevos Requerimientos

- El sistema debe permitir la ejecución de múltiples análisis de clientes en paralelo si ser reiniciado.
- Los clientes deben enviar el conjunto completo de datos a analizar, permitiendo futuras extensiones en las consultas implementadas por el sistema.
- El sistema debe mostrar alta disponibilidad hacia los clientes
- El sistema debe ser tolerante a fallos por caídas de procesos
- En caso de usar un algoritmo de consenso, el mismo tiene que ser implementado por los alumnos
- Se puede utilizar docker-in-docker para levantar procesos caídos
- No está permitido utilizar la API de docker para verificar si un nodo está disponible.

Normas de Trabajo

Se espera del alumno:

- Empleo del tiempo de consultas en clase para resolver dudas y clarificar el negocio del sistema a construir previo a su diseño
- Exposición y verificación en clase de la arquitectura propuesta antes de iniciar su implementación
- Empleo del grupo de correos para realizar consultas que no pudieran ser resueltas en clase
- Consideración de prácticas distribuidas según lo estudiado en clase para elaborar una arquitectura flexible, escalable y robusta
- Aprobación del cuerpo docente para el uso de cualquier librería.
- Demo del sistema en funcionamiento previamente ensayada

Entrega

- Fecha de entrega: 07/12/2023
- Formato de entrega:
  - Entrega digital mediante correo personal.
  - Demostración del sistema en funcionamiento.
  - Documento de arquitectura 4+1 Views o C4Model incluyendo al menos: diagramas de robustez, despliegue, actividades, paquetes, secuencia y DAG.