

UNIVERSIDAD DIEGO PORTALES

Escuela de Ingeniería Informática y Telecomunicaciones



Tarea 2 Sistemas Distribuidos

Integrantes:

Alex Marambio Leyton, Joaquín Silva Sánchez

Sistemas Distribuidos.

Profesor de Cátedra: Nicolas Hidalgo

Fecha de entrega: 03/06/2025

Índice

1. Introducción	3
2. Metodología	4
2.1. Convertidor	4
2.2. Filtering y Homogeneización	4
2.3. Processing	4
3. Resultados y Análisis	5
3.1. Convertidor	5
3.2. Filtering y Homogeneización	5
3.3. Processing	5
4. Conclusiones	6
5. Anexos	6

1. Introducción

En el presente trabajo, se hará un sistema de procesamiento de información de manera distribuida, el cual se hará uso de un filtering, homogeneización y un proccesing, con el trabajo anterior, donde se reutilizará el scraper de tráfico de calles en la región Metropolitana y el almacenamiento de este, pero en este caso, el filtering tiene que filtrar los datos recibidos por la base de datos, y luego, tiene que ser procesado por el componente de proccesing, esto es con el objetivo de de crear un sistema donde pueda procesar información de manera distribuida.

En este informe se documentará a detalle la segunda entrega de tres a lo largo de este semestre. Se contará con las siguientes funcionalidades: Convertidor de JSON a CSV, el filtrado y la homogenización, y el procesado. Cabe mencionar que se recopilarán más consultas que la anterior, siendo un total de 100.000 eventos obtenidos del sistema de scraping, con el objetivo de tener un resultado más cercana a la realidad.

Con los resultados obtenidos de este laboratorio, se tiene el objetivo de analizar los resultados detallados útiles para las Unidades de Control de Tránsito y entidades pertinentes encargados de este ámbito, ya que su utilidad reside en la efectividad de las medidas de respuesta a incidente que con este sistema de procesamiento de datos, la distribución de recursos y la identificación de áreas de alta congestión o peligrosidad.

2. Metodología

En esta sección se comentará acerca de las metodologías hechas para la implementación del sistema de filtrado, homogeneización y procesamiento y las decisiones tomadas para la realización de este trabajo.

2.1. Convertidor

El convertidor tiene la función básica de pedir las consultas a la base de datos MongoDB en JSON y convertirlo a CSV, ya que facilita el funcionamiento a la hora de utilizar apache pig con hadoop.

El script Python que realiza esta tarea utiliza la biblioteca `pymongo` para conectarse al contenedor MongoDB, recupera todos los documentos de la colección. Durante este proceso se realiza un preprocesamiento inicial. Esta estructura estandarizada permite un tratamiento uniforme en etapas posteriores del flujo de datos.

2.2. Filtering y Homogeneización

El filtering se hará con Apache pig y Hadoop, donde se dividirá bajo esquemas unificados que incluirá los tipos de incidentes, comunas, marca de tiempo y descripción, esto se hará dentro de un archivo pig llamado `process.pig`, el cual tiene un `entrypoint.sh`, para ver el estado y mostrando como va avanzando al hacer el filtering a los datos.

El script Pig, definido en el archivo `process.pig`, carga los datos CSV usando `PigStorage`, y los organiza. Se usa `DISTINCT` para eliminar duplicados y se prepara para posteriores análisis. Este archivo es ejecutado mediante `entrypoint.sh`, que además se encarga de iniciar los servicios de Hadoop, montar el sistema HDFS, y supervisar el avance del procesamiento.

2.3. Processing

El processing se hará, al igual que el punto anterior, con apache pig y Hadoop, donde se hará una agrupación de incidentes por comunas para identificar patrones geográficos, además se contará la frecuencia de incidentes y la evolución temporal de este. Este procesamiento se hará después del filtrado, así que una parte del proceso ya se habrá hecho para no tener tanta sobrecarga en el filtrado.

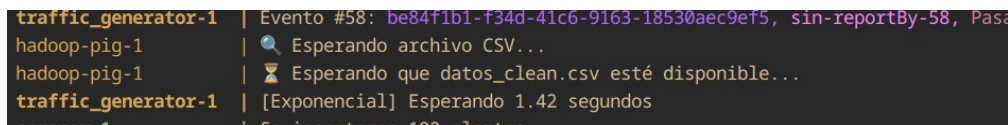
3. Resultados y Análisis

3.1. Convertidor

El convertidor es un script en python que tiene la utilidad de llamar la base de datos, tomar las alertas en JSON y transformarlos en CSV y exportarlo al filtrado y normalización. Este script obtiene los fieldname del primer documento y elimino el `_id`, Además, transforma la estructura anidada de las coordenadas geográficas (`location.x` y `location.y`) en dos campos planos (`location_x`, `location_y`), y convierte el valor en marca temporal `pubMillis` en una cadena de texto para facilitar la lectura y análisis posterior.

3.2. Filtering y Homogeneización

El filtramiento de las alertas extraídas desde Waze permitió transformar un conjunto de datos crudos y potencialmente redundante en información estructurada y analizable. A través del uso de APache pig, se aplicaron operaciones de carga y filtrado que habilitan un primer nivel de análisis cuantitativo. El paso de `DISTINCT` sobre los datos eliminó duplicados basándose en el campo `uuid`, que representa el identificador único de cada alerta. Esto fue crucial para asegurar que cada incidente fuese considerado solo una vez, evitando sesgos en los conteos. La existencia de registros duplicados puede atribuirse a múltiples recolecciones de la misma alerta debido al comportamiento asincrónico del scraper o a repeticiones en la fuente original.



```

traffic_generator-1 | Evento #58: be84f1b1-f34d-41c6-9163-18530aec9ef5, sin-reportBy-58, Pas
hadoop-pig-1       | ⏱ Esperando archivo CSV...
hadoop-pig-1       | ⌚ Esperando que datos_clean.csv esté disponible...
traffic_generator-1 | [Exponencial] Esperando 1.42 segundos

```

Figura 1: hadoop-pig.

3.3. Processing

El módulo de procesamiento utiliza Apache Pig, con el objetivo de traducir operaciones de transformación y consultas analíticas en tareas ejecutables mediante el paradigma MapReduce sobre Hadoop. Este enfoque permitiría aprovechar la escalabilidad y tolerancia a fallos del ecosistema Hadoop para procesar grandes volúmenes de datos de tráfico extraídos desde Waze.

Se deben agrupar las alertas por tipo mediante `GROUP BY type`, lo que hace una segmentación lógica para el análisis. Esta estructura hace el conteo y posterior análisis por categoría de incidente.

4. Conclusiones

Al tener que realizar una comunicación de cuatro módulos con funciones distintas para obtener un mismo fin que es procesar, filtrar y agrupar alertas hacia un servidor, se pudo entender bien el funcionamiento de un sistema que reparte las funcionalidades para un objetivo en concreto, y la sincronización de este, donde se puede entender la descentralización con los contenedores, no solo con docker, si no que también con el uso de hadoop que tiene la funcionalidad de trabajar de manera distribuida gran cantidad de datos de forma distribuidas en clusters.

Se puede concluir que a pesar de que no se pudo terminar la tarea 2 de este proyecto, especialmente la etapa de *processing* mediante Pig, se logró comprender el objetivo principal que es la distribución de tareas en distintos contenedores y componentes para una mejor gestión de estos, haciendo que la distribución del sistema haga más efectiva el uso de los recursos y logre una arquitectura eficiente, modular y flexible

5. Anexos

- Github Tarea 2 Sistemas Distribuidos