Procesamiento paralelo de retrasos en los recorridos de las lineas del sistema de trasporte Metropolitano para la ciudad de Montevideo

Gabriel Acosta y Emiliano Gonzalez
Facultad de Ingeniería, Universidad de la República
E-mail {gabriel.acosta.soria, gonzalez.emiliano}@fing.edu.uy

***Resumen*—El presente artículo trata sobre el análisis de los retrasos en los horarios de las líneas de autobuses, del sistema de transporte en la ciudad de Montevideo, durante el mes de junio del año 2024. Se propone entregar una serie de indicadores sobre la demora en los servicios y la cantidad de personas que afecta este atraso. Como es un gran volumen de datos a procesar, se utilizan técnicas de programación paralela y se utilizarán para ello, las maquinas existentes en las salas de computación de la Facultad de Ingeniería.**

I. Descripción del problema

A partir de los principios para el manejo de los datos abiertos en el gobierno [1] traducción de la intendencia de Montevideo (IM) [2] , pone a disposición de la ciudadanía la mayor parte de los datos públicos que esta maneja, para que puedan ser procesados por cualquier ciudadano, investigador, periodista, universidades y organizaciones nacionales y extranjeras [3], estos datos figuran tanto en la página de la IM [4], como en el catálogo de datos abiertos del Gobierno uruguayo [5].

A partir de estos mismos datos, la IM publica datos estadísticos sobre el desarrollo de ciertas áreas de interés [6], donde la movilidad en todos su aspectos ha sido relevante el los últimos años [7], en particular con la inclusión del sistema de transporte público metropolitano STM y su consiguiente digitalización y utilización de tarjetas de tipo MIFARE [8] y la utilización de GPS para el posicionamiento de los vehículos, han ofrecido muchas posibilidades de procesamiento posterior a partir de las ventas y el seguimiento de los autobuses en su recorrido por la ciudad.

A partir de los datos abiertos publicados y de la captura de datos mediante la API propuesta por la IM [9], se propone calcular el retraso de los buses del sistema STM en Montevideo durante el mes de junio de 2024, luego relacionar estos datos con los datos históricos de la ventas en junio de años anteriores para lograr inferir a qué cantidad de personas afecta ese retraso, otras posibles métricas a mostrar podrían ser las paradas con más atraso, la distribución de los atrasos en el correr del día o en el mes, entre otras.

Realizamos a partir de la API de la IM, una captura cada 20 segundos en todo el mes de junio de 2024, de la ubicación de los buses en tiempo real. A su vez, capturamos diariamente los datos de los horarios estimados de pasada, por cada parada para cada variante y por tipo de día (hábil, sábado y domingo), esto lo hacemos por cualquier cambio eventual en los horarios al correr del mes.

Además, vamos a utilizar los datos de ventas de boletos en junio de 2022 y 2023, ya que no contaremos de forma inmediata con los datos del presente año. Con estos datos y el procesamiento de retrasos de buses, podemos saber cuántas personas por parada por día son afectadas por este retraso.

II. Implementación de la Solución

II-A. Justificación de usar HPC

Cómo se realizará una captura cada 20 segundos durante 30 días. Esto nos da un total de 129600 capturas en total, como existe un promedio de unos 400 buses por captura (máximos de 1500 buses mínimos de 50). Por lo tanto, se necesita procesar unos 52 millones de registros aproximadamente para el mes de junio. En estos registros figuran la variante de la línea, la hora de salida de la terminal (llamado por la IM frecuencia), el número de coche, la ubicación geográfica entre otros datos relevantes. Recorriendo estos datos, se pretende calcular el atraso con la hora de pasada prevista para cada parada, este dato de los horarios, se encuentran en otro archivo que cuenta con unas 2.5 millones de líneas. Encontrar la hora de pasada por cada parada, para determinada variante frecuencia y día en particular, implica un esfuerzo computacional considerable, lo que lleva a una tiempo prolongado de ejecución, si esta se realiza de forma secuencial. Otro detalle adicional a considerar es que la ubicación geográfica de las capturas de GPS se encuentra en formato WGS84 (EPSG:4326) mientras que los datos de los horarios junto con las ubicaciones de las paradas se encuentra en formato UTM 21S (EPSG:32721). Esta transformación necesaria, agrega más tiempo al procesamiento .

II-B. Pre-procesamiento de los datos

El archivo que la IM brinda para los horarios por paradas, no están geo-referenciados, para no aumentar el cómputo, se agregan a este archivo los campos de las coordenadas que brinda la IM para cada parada, que figura en en los datos abiertos de la IM [11]

II-C. Estrategia de resolución

Para el análisis utilizaremos estos dos términos:

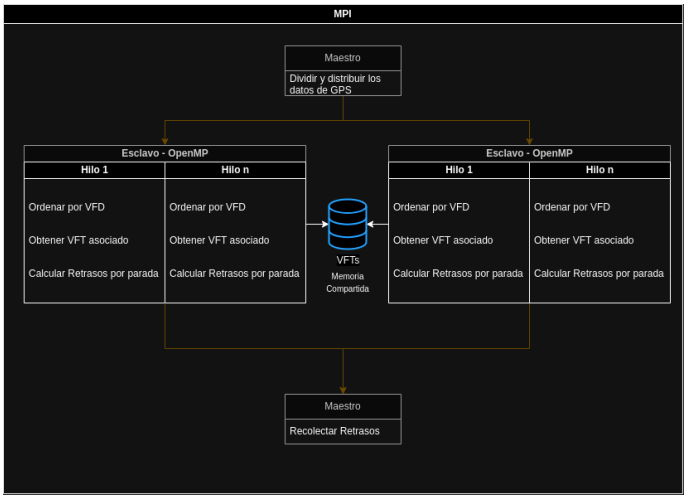

Figura 1. Esquema de resolución

- VFD (Variante,frecuencia, dia):
  Representa de forma única, una variante de bus, una hora de partida desde el origen (llamada por la IM frecuencia) y una fecha determinada. Por ejemplo: variante #4735 (191 - Punta Carretas), 19020 (19:02 hrs), 01/06/2024, corresponde al VFD 4735-19020-01/06/2024, estos datos se procesarán desde el archivo de capturas.
- VFT (Variante, frecuencia, tipo de día):
  Representa de forma única una variante de bus, una hora de partida desde el origen, y un tipo de día (habil=1, sábado=2, domingo=3). Por ejemplo: 4735-19020-1, este dato se procesará desde el archivo de horarios.

El esquema de la figura 1 retrata la estrategia de resolución. En este esquema:
El proceso master divide inicialmente el conjunto de datos entre los procesos esclavos utilizando MPI. Cada esclavo divide los datos asignados equitativamente entre los hilos utilizando OpenMP, para luego realizar el análisis y procesamiento de los mismos. Existe colaboración entre esclavos para el cálculo de ciertos datos, los cuales son almacenados en memoria compartida. Al final los esclavos retornan sus resultados al master.

Para la implementación se utilizará un modelo maestro/esclavo usando un esquema de descomposición de dominio. Esta implementación se hará en C con la bibliotecas MPI, y los algoritmos de los nodos en OpenMP.

Cada proceso esclavo realiza un procesamiento de los datos asignados agrupando por VFD y así obteniendo los retrasos para las paradas correspondientes. Por otra parte un grupo de procesos se encargará del análisis de los datos de los que se obtendrá la venta de boletos por parada durante los meses de junio de los pasados dos años.

Los datos obtenidos desde la API serán divididos por bloques. Estos bloques serán distribuidos dependiendo de la cantidad de procesos con los que se cuente, pero para poder controlar el correcto funcionamiento de los nodos, se harán pruebas, para determinar qué cantidad de bloques se dará por vez a cada uno, esto en vistas de analizar las posibles caídas de nodos en el sistema, y qué repercusión tendrá sobre el funcionamiento de éste. Debido a que el menor flujo de información de los buses reportados sucede a la 3 AM (aproximadamente), los datos serán divididos entre bloques de 24 horas desde las 3 AM de un día hasta las 3 AM del día siguiente, con lo cuál disminuirá la cantidad de VFDs parciales que se repartirán, entre los procesos con los que se ejecute el sistema.

Por otro lado, los datos de los horarios de pasada por día y por parada, serán compartidos entre los procesos, con estos se generarán los VFTs, que en conjunto con los VFD, se utilizarán para calcular el retraso en las paradas por las que pasa la línea para los datos dentro del VFD.

Cada proceso deberá ordenar los datos por VFD antes de procesarlos. Esto hará que cada uno de ellos cuente con datos totales o parciales para determinado VFD. Luego del ordenamiento se realizará el procesamiento de los datos.

Cada proceso deberá obtener un VFT (a demanda), correspondiente a cada VFD. Cómo existirá más de un proceso que necesite utilizar el mismo VFT, se deberá implementar un mecanismo para evitar, en lo posible, la redundancia en el cálculo de los VFT, y a su vez hacer colaborativa la realización de esta tarea. Para esto, se pueden utilizar ventanas de memoria compartida donde cada proceso, le comunica al resto, mediante un broadcast, que realizará el cálculo de determinado VFT y reserva un segmento de memoria donde se va a almacenar este dato, para que luego pueda ser accedido por otros procesos que los necesiten.

Se implementarán dos algoritmos: uno para procesar el conjunto de datos en cada proceso (ordenar en VFDs y obtener VFTs), esta implementación se hará en C con la bibliotecas OpenMP, luego se utilizará para el cálculo de la demora, un script de Python utilizando librerías de QGIS [10], que realizará las siguientes tareas: transformar las coordenadas de las paradas del VFT de los sistemas WGS84 a UTM 21S, determinar qué paradas del VFT serán cubiertas por el recorrido del VFD (ya que pueden ser parciales), luego para estas paradas válidas se calculará la hora de pasada utilizando para ellos los dos registros más cercanos, retornando así la demora en minutos para cada parada e incluyendo datos relevantes para un posterior análisis .

II-D. Post-procesamiento de los resultados

Una vez el maestro obtiene los resultados, hace un análisis posterior para devolver la demora para cada parada para cada VFD, y enlaza estos datos con la cantidad de personas por parada, que se estima afecta este retraso.

III. Análisis experimental

III-A. Rendimiento

Para evaluar el rendimiento del sistema se calcularán diferentes métricas:

- tiempo de ejecución.
- speedup logarítmico.
- eficiencia.
- paralelicidad.

Cada una de estas métricas se van a calcular sobre diversas cantidades de nodos, e hilos dentro de cada nodo. En la siguiente tabla se observan los resultados de los datos:

Nodo	Hilos	TE	Speedup	Eficiencia	Paralelicidad
1	1	X	X	X	X
1	2	X	X	X	X
1	4	X	X	X	X
2	4	X	X	X	X
3	4	X	X	X	X
4	4	X	X	X	X
5	4	X	X	X	X
6	4	X	X	X	X

Figura 2. Tabla de resultado de los datos

III-B. Escalabilidad

- Comunicación entre procesos
  Con el fin de minimizar el overhead existente para la comunicación entre procesos, se podría utilizar comunicación asincrónica, para reducir los tiempos ociosos.
- Balance de carga.
  Se utilizará una arquitectura de maestro-esclavo, donde el maestro distribuye las tareas de forma dinámica entre los esclavos utilizando MPI. Cada esclavo utilizará OpenMP para paralelizar la tarea en múltiples hilos.
- Granularidad
  Según se experimente, se hará la división y distribución con una granularidad que no utilice de forma intensiva la red, para no comprometer la eficiencia y tampoco que los procesos analicen bloques muy grandes de información por vez.

III-C. Tolerancia a fallos

Se implementarán checkpoints para que el maestro esté al tanto del estado de cada esclavo o procesos dentro de este, para poder reasignar recursos hacia los esclavos que se van liberando o incluso matar procesos y asignar esa tarea a uno nuevo, en caso de ser necesario.

III-D. Evaluación experimental

III-E. Discusión de resultados experimentales

IV. Resultados obtenidos y Conclusiones

IV-A. Resultados del procesamiento

IV-B. Conclusiones

Referencias

[1] Principios para el manejo de datos abiertos en el gobierno — Intendencia de Montevideo.
URL: https://montevideo.gub.uy/areas-tematicas/servicios-digitales/datos-abiertos/principios-para-el-manejo-de-datos-abiertos-en-el-gobierno
[2] 8 Principles of Open Government Data.
URL: https://public.resource.org/8_principles.html
[3] Servicios digitales — Intendencia de Montevideo.
URL: https://montevideo.gub.uy/areas-tematicas/servicios-digitales
[4] Portal de Datos Abiertos — Intendencia de Montevideo.
URL: https://ckan.montevideo.gub.uy/
[5] Catálogo de Datos Abiertos — GUB.UY
URL: https://catalogodatos.gub.uy/
[6] Datos Abiertos Ambientales — Intendencia de Montevideo.
URL: https://montevidata.montevideo.gub.uy/
[7] Movilidad — Intendencia de Montevideo.
URL: https://montevidata.montevideo.gub.uy/movilidad
[8] MIFARE: Contactless Chips.
URL: https://es.wikipedia.org/wiki/Mifare
[9] API de Transporte Publico — Intendencia de Montevideo.
URL: https://api.montevideo.gub.uy/apidocs/publictransport
[10] QGIS — Sistema de Información Geográfica libre y de Código Abierto.
URL: https://www.qgis.org/es/site/about/index.html
[11] Transporte colectivo: paradas, puntos de control y recorridos de omnibus. URL: https://catalogodatos.gub.uy/dataset/transporte-colectivo-paradas-puntos-de-control-y-recorridos-de-omnibus