Charla Tesis II

Paralelización de procesos de modelamiento de tráfico urbano por medio de la contenerización del software Simulation of Urban MObility (SUMO) para supercomputadores

Pablo Villar Mascaró

Universidad de Chile

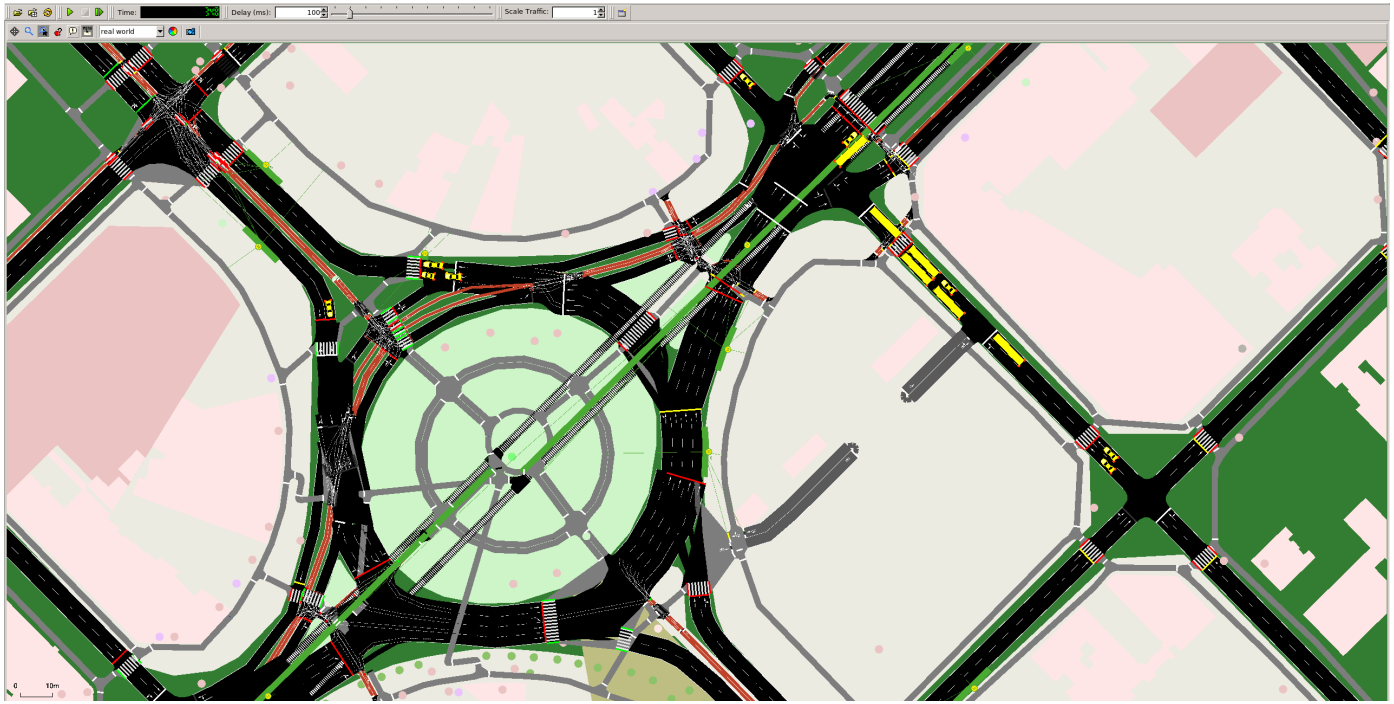
2025-01-28

Índice

Introducción

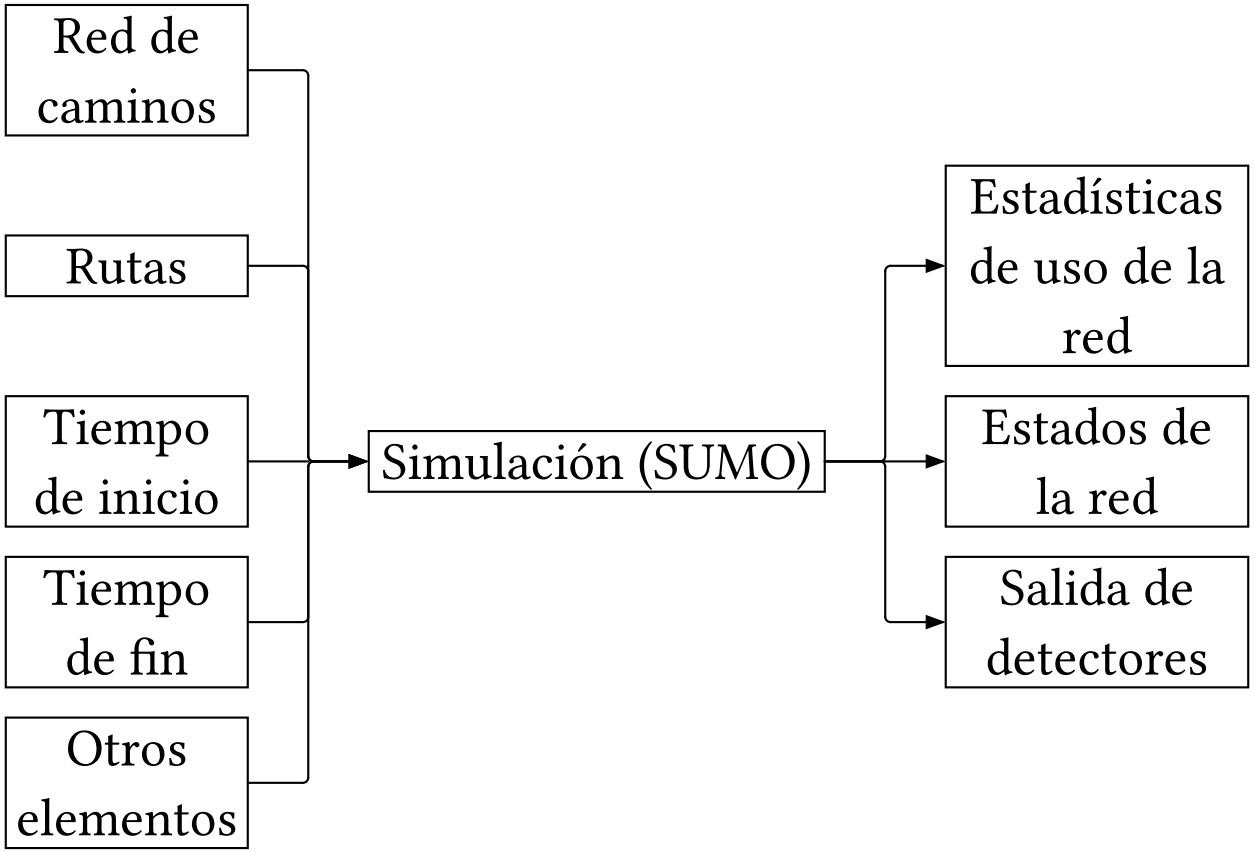
Motivación

Modelamiento de tráfico vehicular urbano para el desarrollo de *Gemelos Digitales* para ciudades.



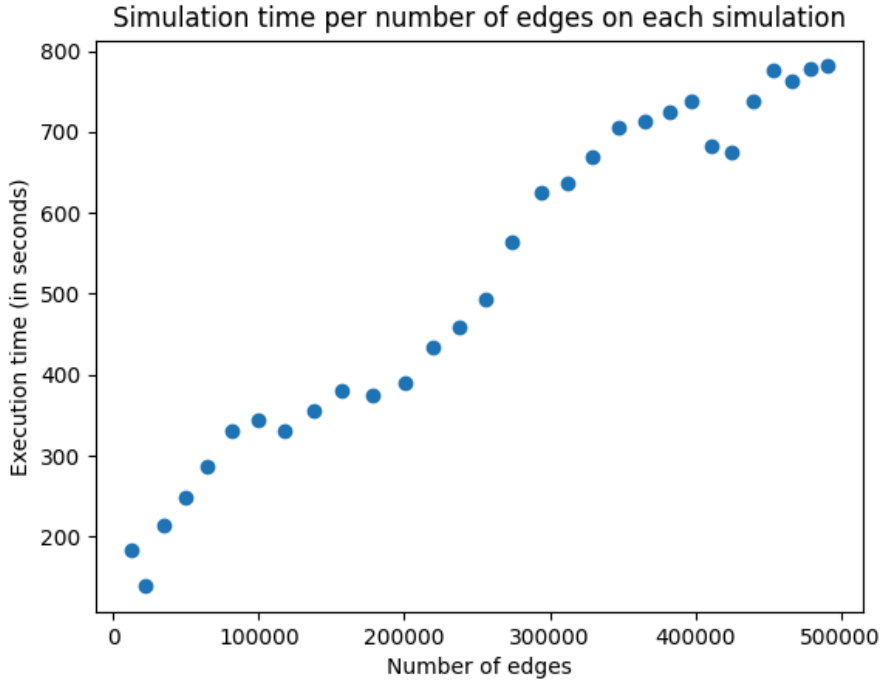
SUMO

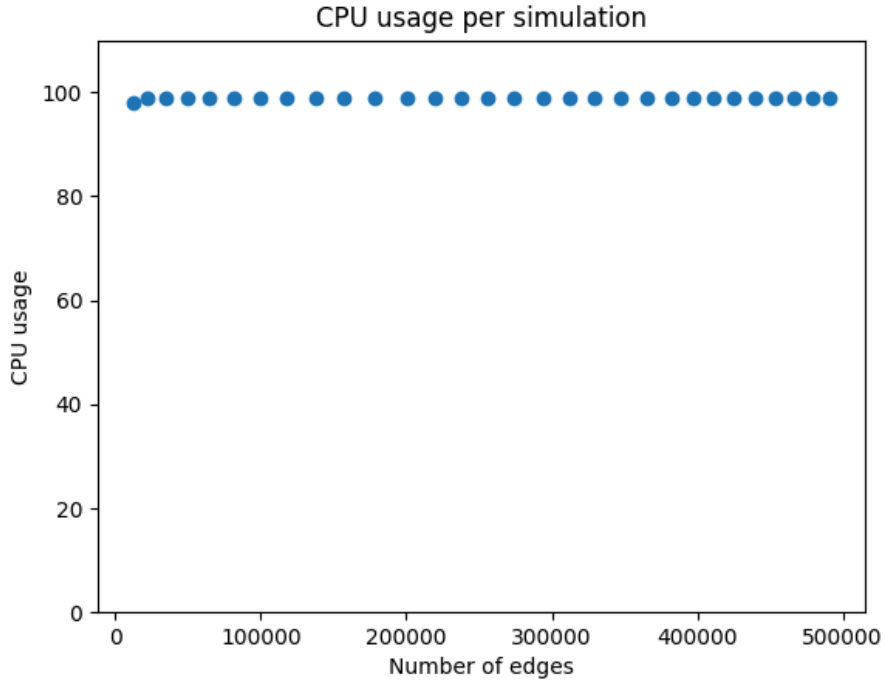
- Herramienta de simulación de tráfico continuo y microscópico diseñado para manejar grandes redes de caminos [1].
- Es portable, de código abierto y puede importar redes desde diversas fuentes.
- Las redes de caminos se representan como grafos de nodos y aristas unidireccionales que representan calles y veredas.


Red de caminos
Rutas
Tiempo de inicio
Tiempo de fin
Otros elementos
Simulación (SUMO)
Estadísticas de uso de la red
Estados de la red
Salida de detectores

Problema

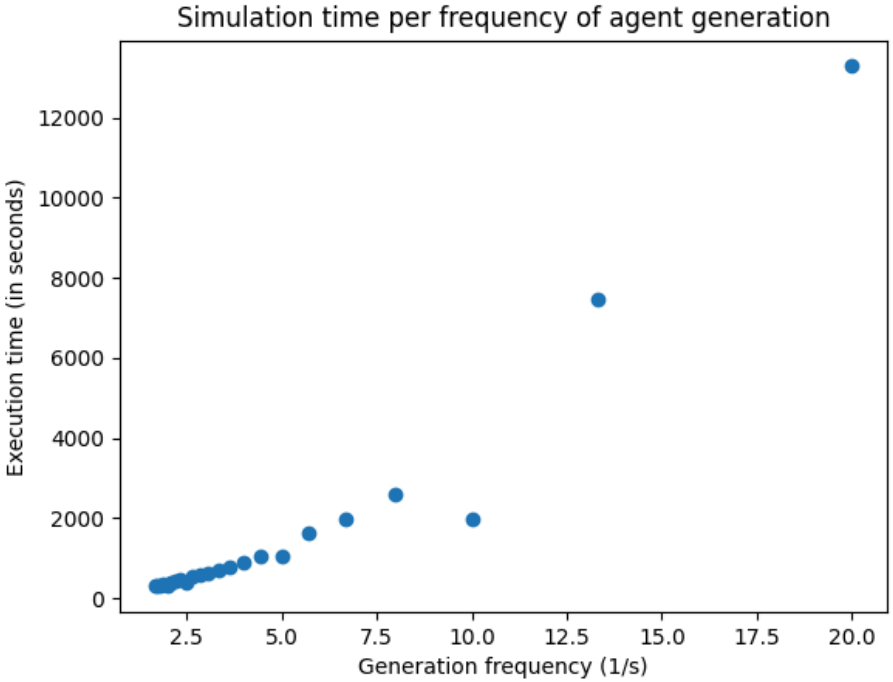
Pruebas de escalabilidad en SUMO basadas en el tamaño de la red

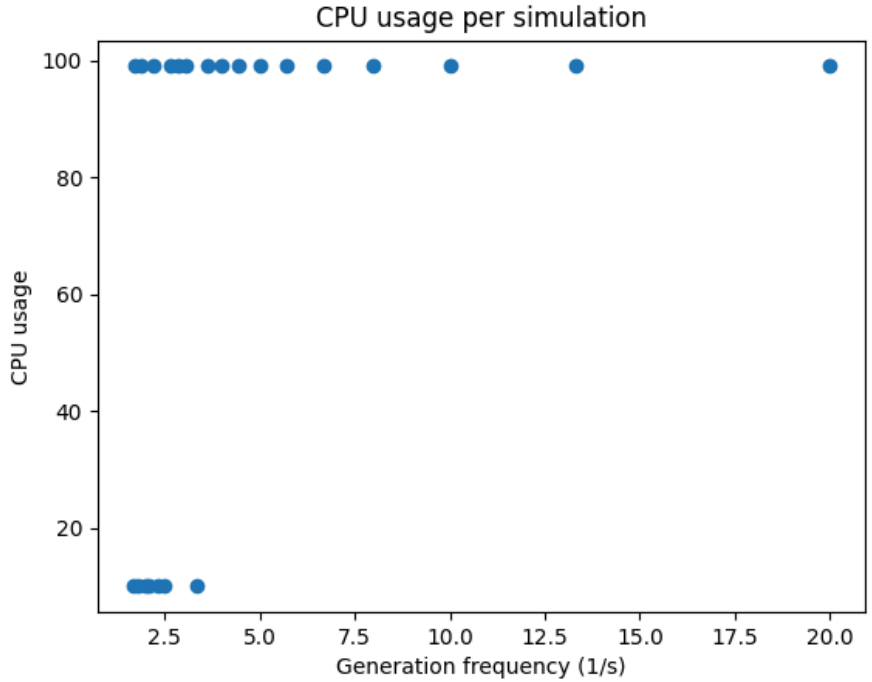




Problema

Pruebas de escalabilidad en SUMO basadas en la carga de tráfico vehicular





Problema

Planteamiento del Problema

- Existe un crecimiento exponencial respecto a los tiempos de ejecución en relación a la frecuencia de inserción de vehículos en las simulaciones (a mayor carga, mucho mayor el tiempo de simulación).
- En cada simulación, se ocupa el 100% de los recursos asignados.
- La escalabilidad se ve afectada para simulaciones que contemplen una gran cantidad de agentes en áreas extensas

Pero, **¿y si paralelizamos?**

Preguntas de investigación

- ¿Cómo se diferencian en cuanto a *performance* la versión secuencial de simulaciones en SUMO respecto a una versión paralelizada?
- ¿Es posible mejorar la *performance* de simulaciones de tráfico vehicular urbano en orden de poder simular áreas metropolitanas completas a nivel microscópico?
- ¿Cómo se comporta la influencia de la carga vehicular a simular en la *performance* de las simulaciones paralelizadas?

Es posible aumentar la escalabilidad de las simulaciones de alta granularidad mediante la paralelización de los procesos del *software* SUMO en un ambiente de supercomputación, incrementando el *speedup* de éstos en al menos un 5% de los tiempos de simulación secuenciales, optimizando así el uso de recursos para simulaciones que contemplen áreas metropolitanas de gran extensión y alta carga de tráfico vehicular.

Objetivos

Objetivo general

Avanzar en el estado del arte acerca de la paralelización de procesos de simulación de tráfico urbano para supercomputadores, estudiando la escalabilidad de dichos procesos luego de su paralelización.

Objetivos

Objetivos específicos

1. Aplicar la contenerización de SUMO en *Singularity Containers* para su ejecución en supercomputadores.
2. Diseñar e implementar un modelo de paralelización de los procesos contenerizados, determinando los parámetros necesarios para la efectiva comunicación y sincronización entre los procesos.
3. Medir la escalabilidad de la solución implementada y comparar con las mediciones previamente realizadas.

Solución propuesta

Trabajos relacionados

- Modelo de particiones de mapa simplificadas para la simulación en paralelo de tráfico en zonas urbanas [3]
  - Incrementa el nivel de error de la simulación de manera proporcional con el número de particiones realizadas.
- QarSUMO: una versión paralelizada de SUMO orientada a computadores personales [4]
  - Incrementa la eficiencia de las simulaciones.
  - Posee un alcance limitado al no estar orientado a entornos de *High-Performance Computing*.
  - Disminuye el nivel de granularidad de las simulaciones.

A partir de esto, se plantea un modelo de paralelización orientado a supercomputadores que haga uso de un algoritmo de particiones de grafos (*SPartSim*), y que a partir de la contenerización de instancias de SUMO ejecute distintos nodos de simulación con una partición asignada.

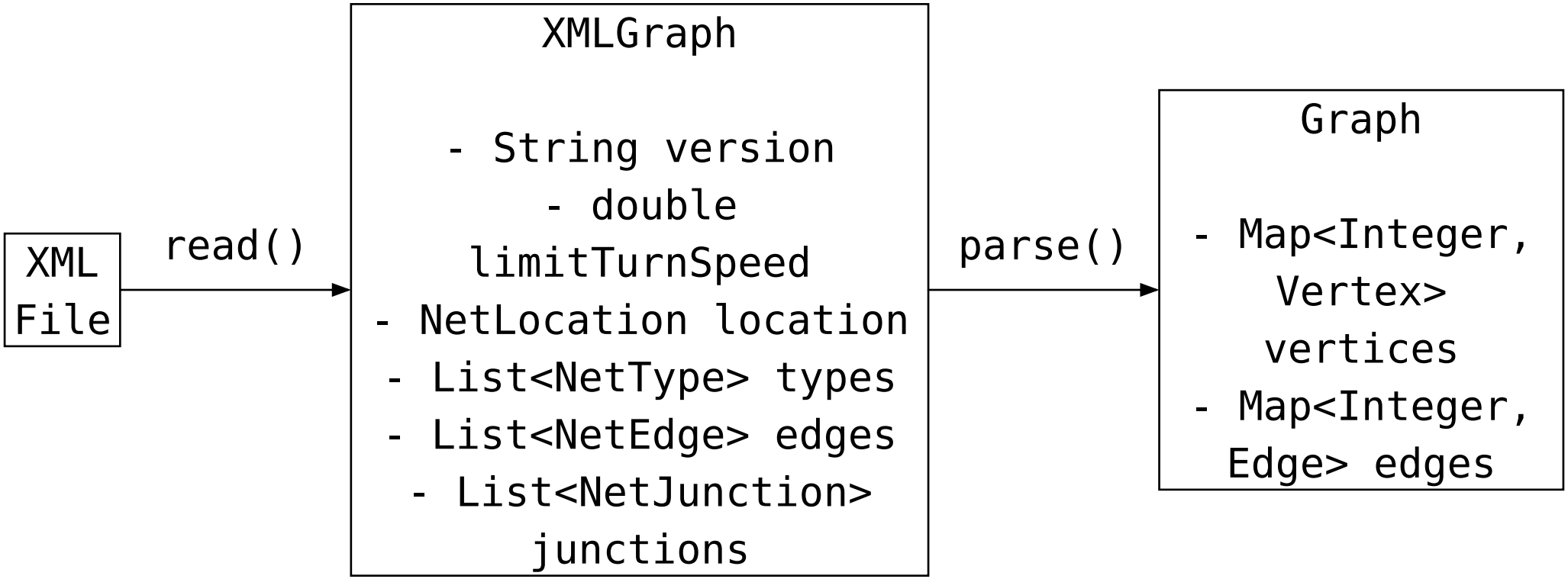
Diseño de la solución

Herramientas a utilizar

Partición de mapas

- Se hizo uso de una implementación del algoritmo *SPartSim* [5], el cual ejecuta una división por zonas geográficas.
- Se debió implementar un módulo de compatibilidad con archivos `XML` de SUMO, dado que originalmente la implementación sólo recibía archivos `GeoJSON`.

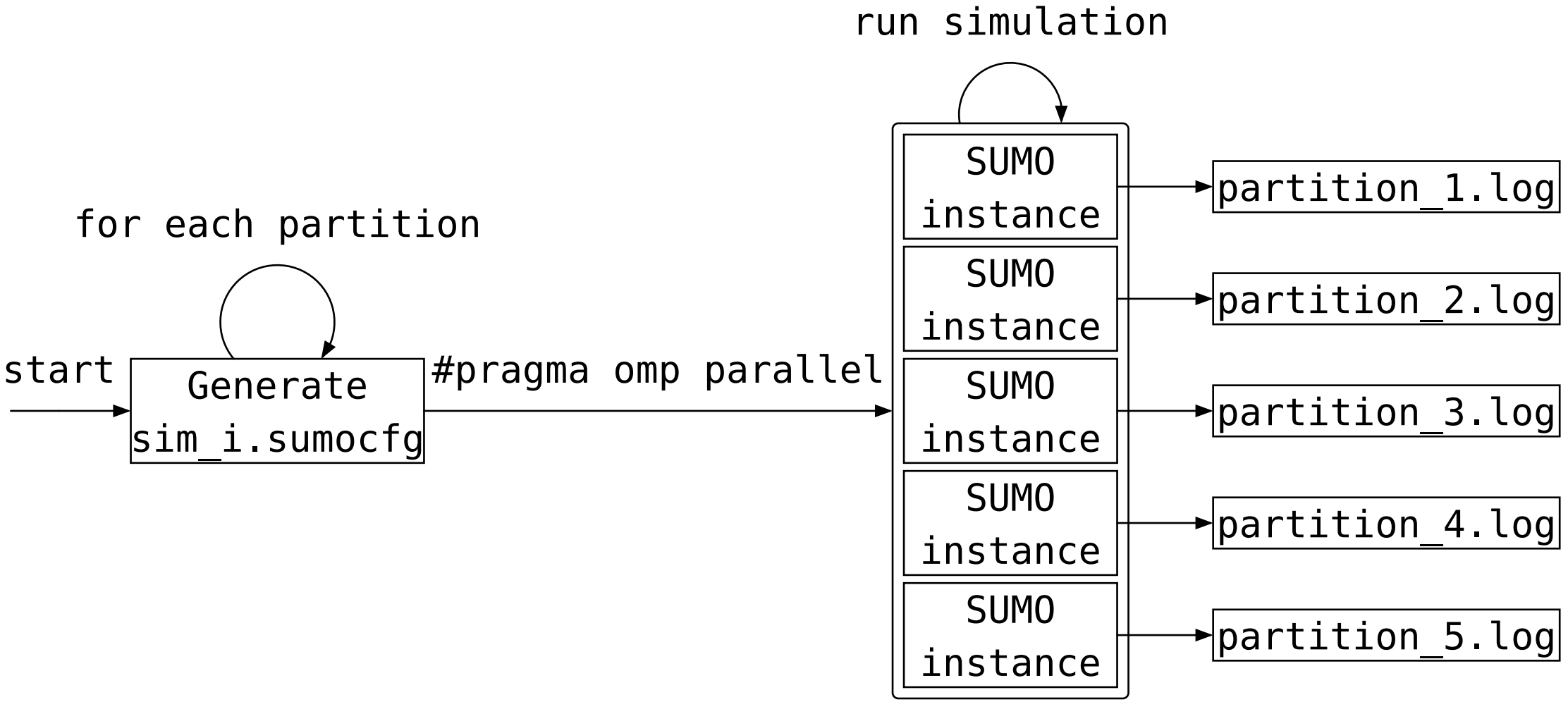
Partición de mapas



Generación y partición de rutas

- Se evaluaron dos opciones para la generación de rutas vehiculares:
  1. Generar rutas a partir de datos reales de movilidad urbana
  2. Generar rutas de manera pseudo-aleatoria por medio de *randomTrips.py* (herramienta provista por SUMO).
- Por simplicidad y para propósitos generales del trabajo, se eligió la segunda opción.
- Para la partición de las rutas generadas, se hizo uso de la herramienta *cutRoutes.py* provista por SUMO, además de las particiones de mapa generadas a partir de *SPartSim*.
  - No obstante, la herramienta presenta sus limitaciones en cuanto al cálculo de los tiempos de partida de los vehículos.

Paralelización de simulaciones



Paralelización de simulaciones

- Cada hilo de ejecución posee una instancia de SUMO contenerizada en *Singularity*.
- De esta manera, se evitan los conflictos de lectura/escritura para las simulaciones paralelizadas.
- Se llama a la directiva de compilación `#pragma omp parallel` para realizar la distribución de los nodos de ejecución con cada una de las particiones asignadas.

Reensamblaje de rutas

- Se implementó un módulo en `Python` que reconstruye las rutas particionadas para tener información de cuál es la ruta y partición que sigue a la salida de un vehículo.
- Si no existe una ruta siguiente, entonces el vehículo termina por salir de la simulación.
- Se logró una reconstrucción parcial de los viajes generados
  - Parcial, dado que *cutRoutes.py* no conserva las rutas tal como uno las entrega (hay pérdida de información).

Comunicación entre nodos

- Se implementó una cola global en la cual se escriben los vehículos salientes de cada partición con el nodo al que les corresponde entrar en el siguiente paso.
- De esta manera, cada nodo se encarga de insertar los vehículos correspondientes a su partición.
- Para evitar conflictos de lectura/escritura, esta cola se implementa por medio del uso de secciones críticas.

Impedimentos para la sincronización

- *cutRoutes.py* presenta limitaciones.
  - Principalmente, al momento de definir los tiempos de partida de los vehículos (*departure times*).
  - Lo que hace finalmente es redefinir aquellas rutas que cruzan más de una partición como rutas independientes con el mismo tiempo de partida.
  - Esto provoca que al ingreso de los vehículos en los nodos de simulación, éstos no se reconozcan dado que los mismos ya han salido de la simulación.
- Es posible solucionar estos problemas mediante la modificación del código fuente de la herramienta.

Test de carga

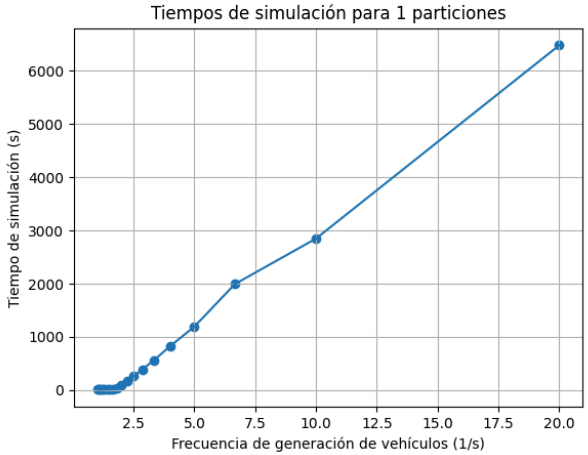
Versión secuencial

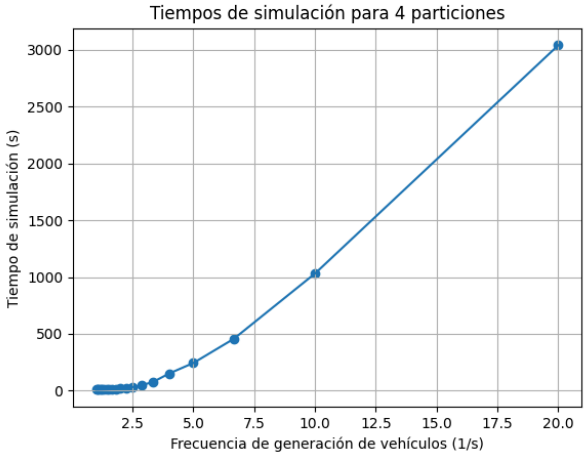
- ***Setup***: define los períodos de generación de vehículos y genera los archivos de rutas para el mapa definido.
  - Se definen 20 períodos y para cada período se crea el archivo de configuración correspondiente.
- **Ejecución**: cada simulación se ejecuta un total de 50 veces con una versión contenerizada de SUMO y se promedian los tiempos de ejecución para cada período.

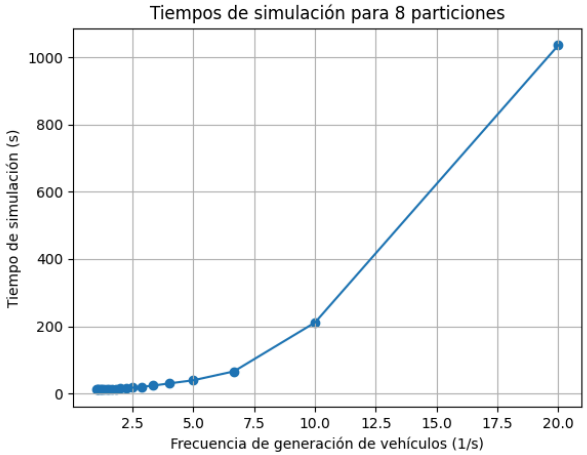
Versión paralelizada

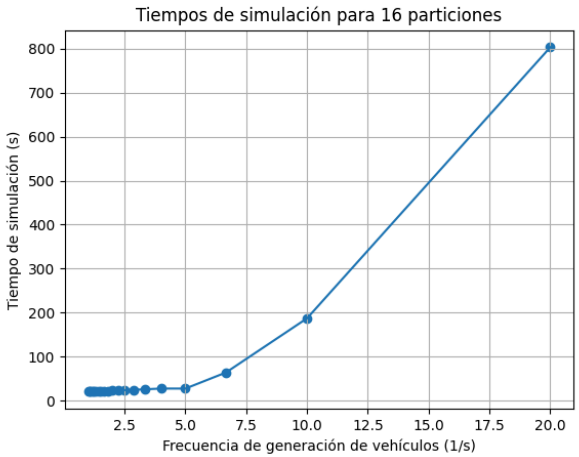
- ***Setup***: genera los períodos, rutas, particiones y archivos de configuración para cada nodo, y organiza los archivos en sus directorios correspondientes.
- ***start_test***: por cada *set* de particiones realizadas, se inicia el test de carga para cada período dado.
- ***load_test***: ejecuta las simulaciones llamando al módulo encargado de paralelizar los procesos de simulación según el número de particiones realizadas. Cada simulación es ejecutada 50 veces para luego promediar los tiempos de ejecución.

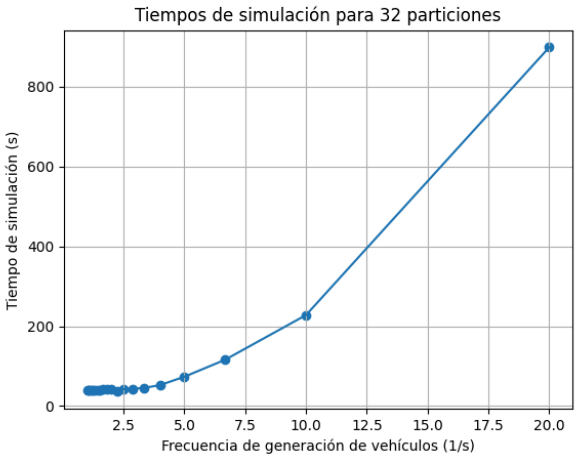
Resultados

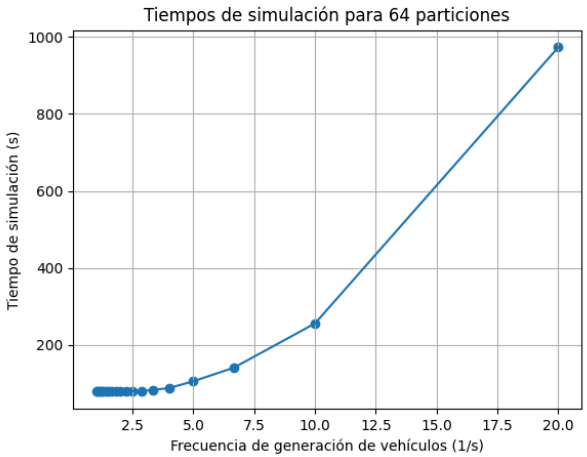

Tiempos de simulación para 1 particiones
Tiempo de simulación (s)
Frecuencia de generación de vehículos (1/s)

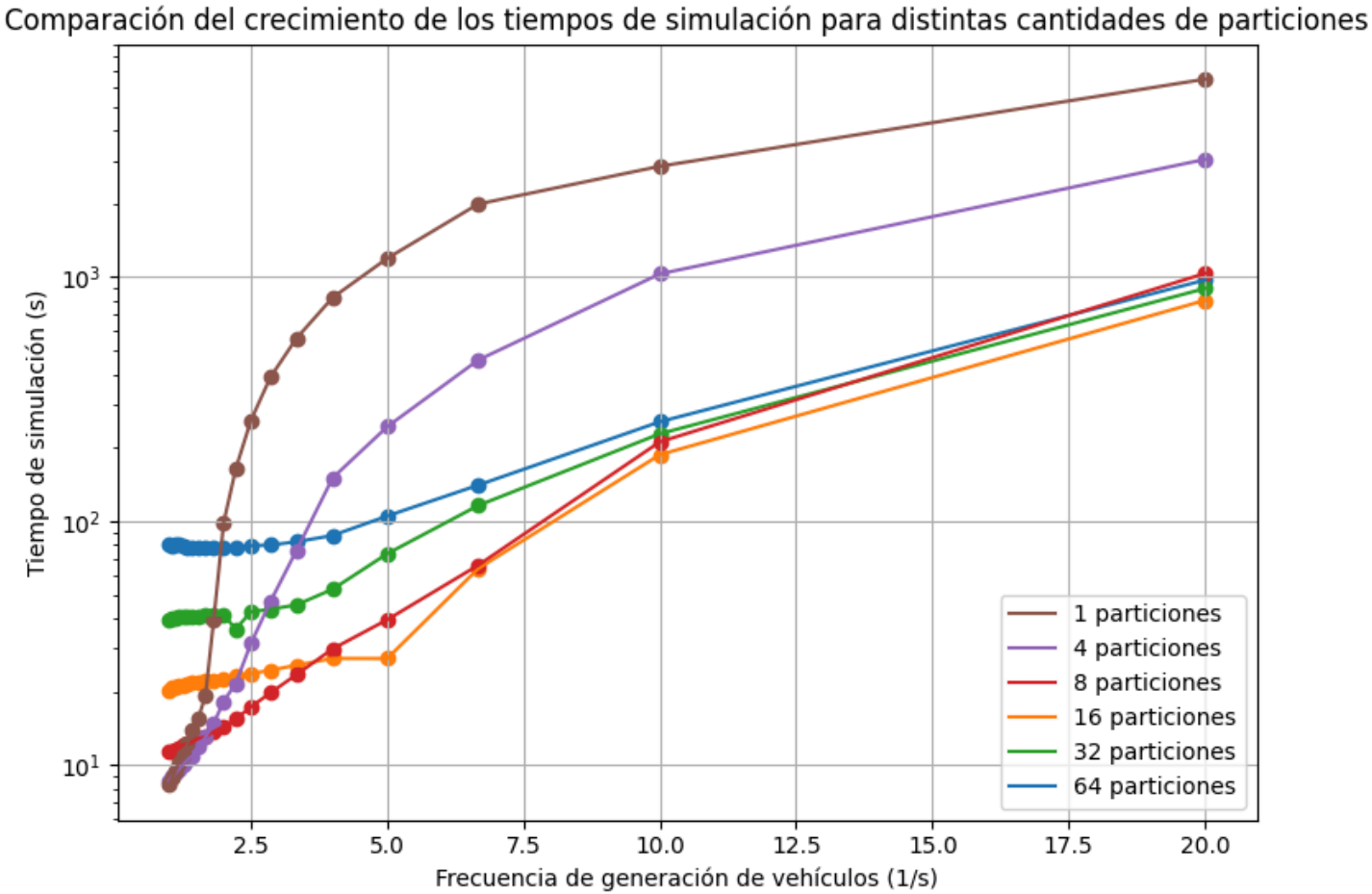

Tiempos de simulación para 4 particiones
Tiempo de simulación (s)
Frecuencia de generación de vehículos (1/s)


Tiempos de simulación para 8 particiones
Tiempo de simulación (s)
Frecuencia de generación de vehículos (1/s)

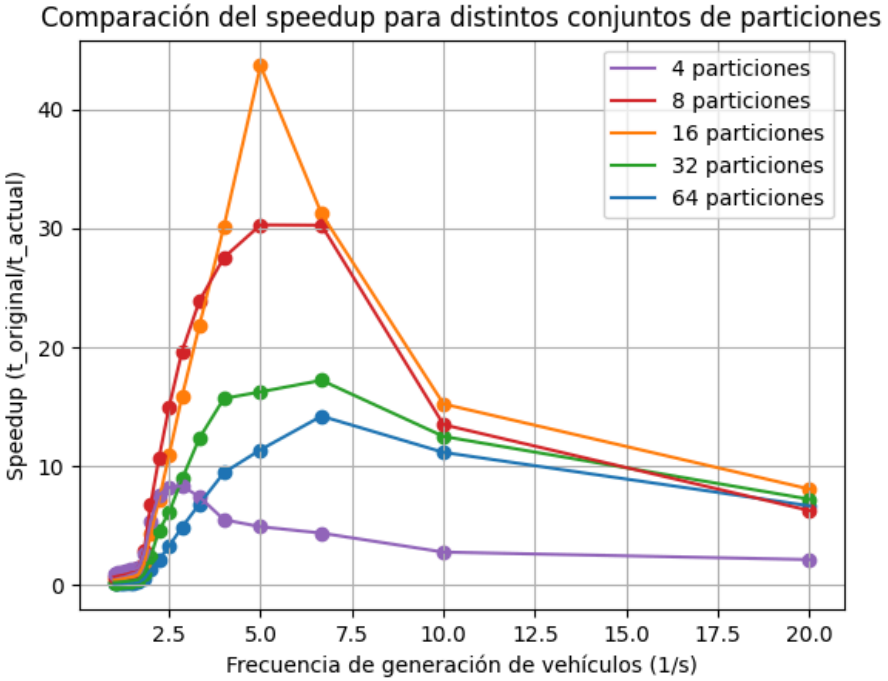

Tiempos de simulación para 16 particiones
Tiempo de simulación (s)
Frecuencia de generación de vehículos (1/s)

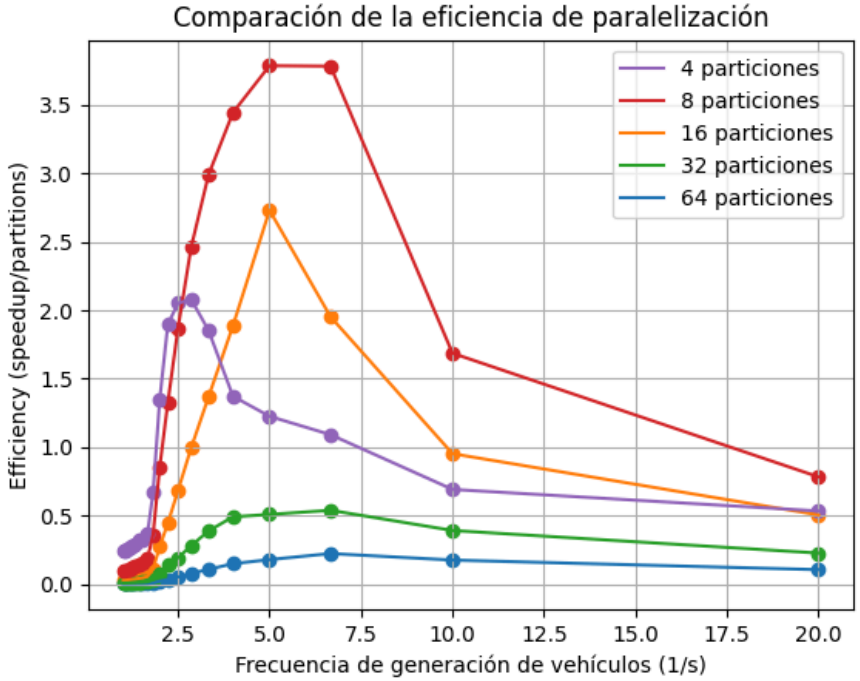

Tiempos de simulación para 32 particiones
Tiempo de simulación (s)
Frecuencia de generación de vehículos (1/s)


Tiempos de simulación para 64 particiones
Tiempo de simulación (s)
Frecuencia de generación de vehículos (1/s)

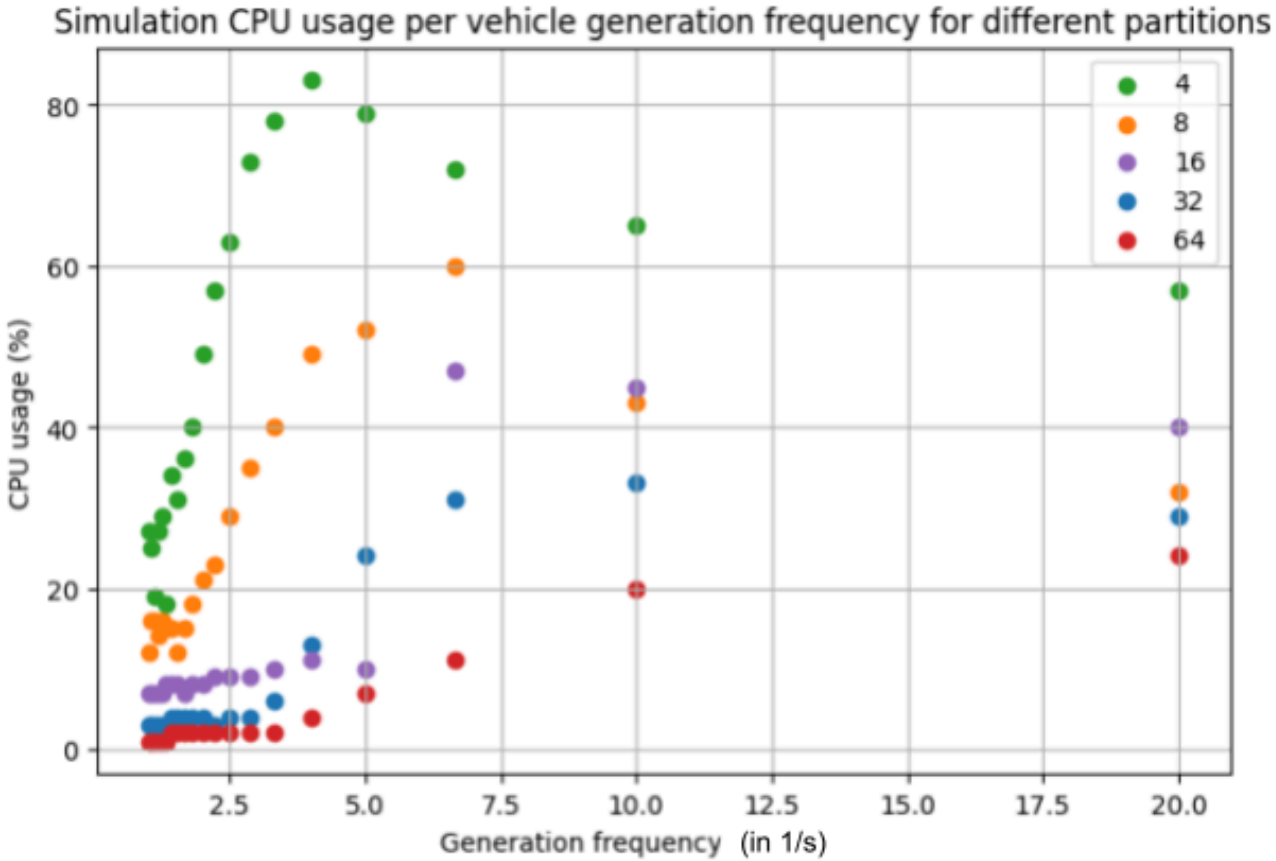

Comparación del crecimiento de los tiempos de simulación para distintas cantidades de particiones
Tiempo de simulación (s)
10^3
10^2
10^1
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Frecuencia de generación de vehículos (1/s)
1 particiones
4 particiones
8 particiones
16 particiones
32 particiones
64 particiones

Speedup y eficiencia





Uso de CPU


Simulation CPU usage per vehicle generation frequency for different partitions
CPU usage (%)
Generation frequency (in 1/s)
0
20
40
60
80
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
4
8
16
32
64

Conclusiones

Discusión

- Las simulaciones paralelizadas poseen una cota superior de tiempo de ejecución mucho menor a la versión secuencial (con un sólo nodo)
- A medida que se aumenta la cantidad de particiones, el escalamiento parece ser cada vez más lineal.
  - Las simulaciones con menor frecuencia de inserción de vehículos se ven afectadas.
- El aumento en el *speedup* puede superar el 5% de los tiempos de simulación, pero no en todos los casos.
  - Para los *sets* de 32 y 64 particiones, la eficiencia decrece.
- Es importante considerar en los resultados el **balance de las cargas** y la **pérdida de información** al realizar el corte de rutas.

Conclusión y trabajo futuro

- Se logró implementar un sistema de paralelización de simulaciones de tráfico vehicular urbano orientado a supercomputadores.
- Es posible mejorar la *performance* de las simulaciones de tráfico vehicular urbano con SUMO mediante la paralelización.
  - Sin embargo, es necesario considerar el *overhead* que implica la sincronización.
- El tamaño del área a simular no es un factor determinante en el escalamiento de las simulaciones. Aún así, se sugiere realizar mayores estudios que varíen los tamaños de los mapas así como la carga de tráfico vehicular.
- Asimismo, se sugiere realizar los experimentos variando los algoritmos de partición de grafos en búsqueda de la mayor eficiencia.

Referencias

[1] German Aerospace Center, “Introduction.” [Online]. Available: https://sumo.dlr.de/docs/index.html.

[2] Daniel Krajzewicz *et al.*, “The 'Simulation of Urban MObility' package: An open source traffic simulation.”

[3] Nicolas Arroyave, Andrés Acosta, Jorge Espinosa Oviedo, and Jairo Espinosa Oviedo, “A new strategy for synchronizing traffic flow on a distributed simulation using SUMO.”

[4] Hao Chen *et al.*, “QarSUMO: A Parallel, Congestion-optimized Traffic Simulator,” *ACM*.

[5] Anthony Ventresque *et al.*, “SPartSim : A Space Partitioning Guided by Road Network for Distributed Traffic Simulations.”