GPS Spoofing Detection

Jing Xuan Selwyn Ang

Vilnius University, National University of Singapore

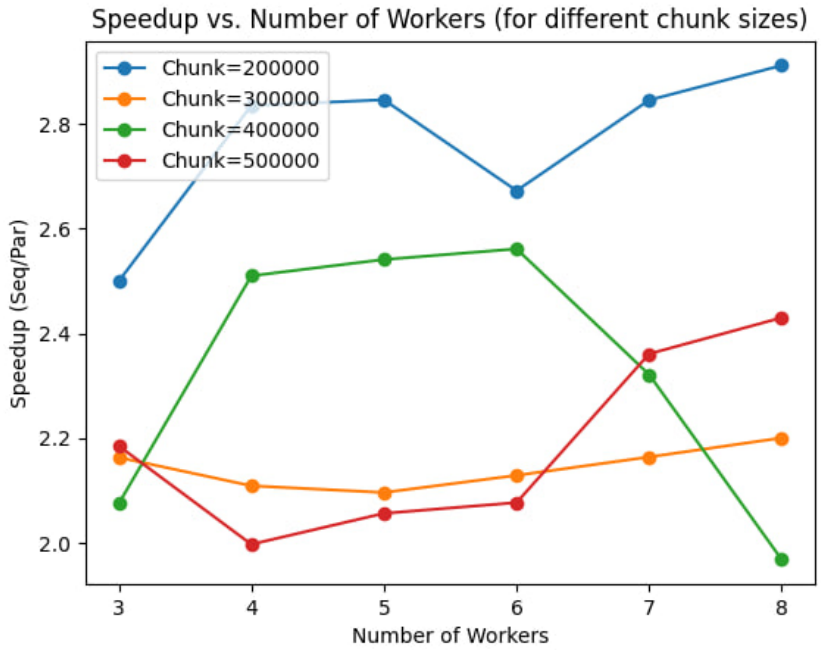
March 15, 2025

Areas to parallelise

- **Data Parallelism:** CSV data can be divided into chunks and each chunk can be sent to the pool of workers to be processed by a specific worker.
  - Parallelism is done via Python's multiprocessing module with the apply_async method.
  - Experiment was conducted to see which specific configuration (chunk size and number of workers) causes the most speed-up over sequential processing.
- **Task Parallelism:** Decided to opt against task parallelism techniques due to a few reasons:
  - May need to pass intermediate results from one set of tasks (e.g. location anomaly detection) to another (e.g. cross-vessel checks). On large datasets, that can become expensive and complicated.
  - Some workers may become overloaded if they are assigned more computationally-heavy tasks.

Parallel processing performance

- More workers does not imply faster performance!
  (More workers may lead to increased overhead due to spawning of more processes, increased splitting of data & concatenating of data across more workers).
- Chunk size does not really correlate with speed-up.



Short-comings of parallel processing

1. Data parallelism involves breaking up data across different chunks, which results in some calculations that require the whole dataset to be inaccurate (eg. Different chunk sizes will lead to different number of anomalous rows being discovered).
2. Sequential processing is actually faster than parallel processing on smaller, less complex datasets.

Experience executing code on the HPC server

- HPC server can be accessed via SSH (Instructions can be found in README).
- Installed conda in HPC to manage Python dependencies and packages.
- Created a shell script to send the job to the SLURM manager in the HPC server, so that code can run independently on the HPC server.
- Results are saved to an output text file and CSV file.

```
#!/bin/bash
#SBATCH -p main             # use the 'cpu' partition
#SBATCH -n8                 # request 8 CPU cores
#SBATCH -t 24:00:00         # up to 24 hour of runtime
#SBATCH -J bda_assignment_1     # job name
#SBATCH -o bda_assignment_1.out # output file name

python3 main.py
```