# CS3210 Assignment 3 Report

Richie Hsieh, A0252126N
Nguyen Quang Truong, A0293470B

## Table of Content

# Description

## Parallelisation Strategy and Design

The program uses Message Passing Interface (MPI) to distribute the computational workload across multiple processes by dividing the total stations among the available processes and MPI ranks, with each rank responsible for managing and simulating the stations that are in its allocated range. This helps to distribute the workload, minimising bottlenecks and improving scalability.
Communication between ranks is done to handle the movement of trains between stations managed by different processes. Trains are sent from one rank to another by a custom SentTrain struct using non-blocking MPI communication (MPI_Irecv, MPI_Isend).
Each rank manages an evenly distributed range of local stations, which include platforms, holding areas, links, trains, using queues and other supplementary structures for efficient state transitions.
Each tick is divided into the following steps:

1) **Train Spawning**: Processes containing the terminal stations will spawn new trains locally, adhering to the provided limits per line.
2) **Movement from Links to stations**: Movement between local links and stations are handled locally in the process, while movement of trains between stations of different processes are coordinated via MPI communication. Trains arriving at a station's holding area are sorted by their ID before being inserted into the respective holding areas.
3) **Station-Specific Movements**: Movement of trains from platforms to links and from holding areas to platforms are managed locally within the processes overseeing the stations.
4) **Result compilation**: Each process will compile the string results of their own stations. These results are transmitted to the master process (rank 0) via MPI communication functions (MPI_Gather and MPI_Gatherv).
5) **Final Output**: The master process sorts and prints the compiled results.

## Resolution of Deadlocks/Race conditions

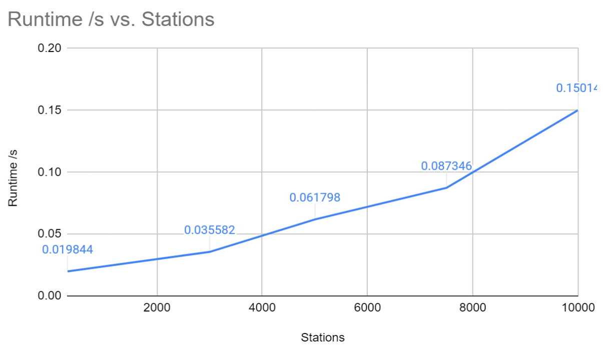
To prevent deadlocks, we use non-blocking MPI communication (MPI_Isend and MPI_Irecv) to facilitate the exchange of train data between ranks. This prevents ranks from being blocked while waiting for data. MPI_Alltoall is used to inform receiving ranks of the appropriate number of data items to receive beforehand. We then use MPI_Waitall to ensure that all outstanding communication operations are completed before proceeding to subsequent steps.
Each rank operates independently on its allocated local stations, except when trains are moving between stations managed by different processes. This minimises the need for synchronisation between stations on different ranks, reducing dependencies and helping to avoid race conditions.

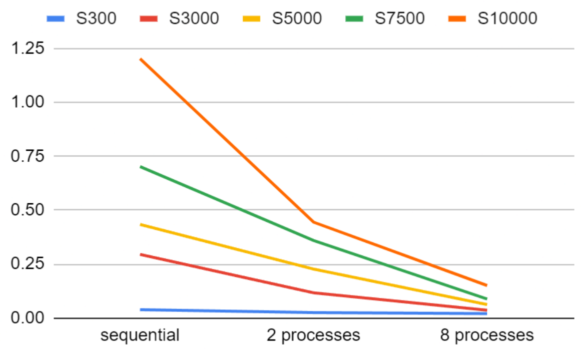
## MPI Constructs

**MPI_Type_create_struct:** Used to create custom MPI data types to enable the transfer of custom data structures (Train and SentTrain) between processes.
**MPI_Alltoall:** Used to exchange the counts of trains being sent between ranks. This ensures that ranks know the size of the messages sent between each other.
**MPI_Irecv and MPI_Isend:** Enables asynchronous communication, allowing ranks to perform computation while waiting for data exchanges.
**MPI_Waitall**: Used to ensure that all outstanding Irecv and Isend operations are completed before proceeding with further computation or communication.
**MPI_Gather**: Collects the size of string outputs from all ranks and sends them to the master process (rank 0) for string result compilation.
**MPI_Gatherv**: Collects train position data from all ranks to the root process for consolidated output.

## Analysis

### Number of Stations





The results (Appendix) show that with an increasing number of stations, there is an increase in the runtime of the program. This is likely due to the increasing size and complexity of the train network. A larger number of stations results in an increased number of messages exchanged between processes to handle train movements across stations of different processes, thereby increasing the communication overhead and the complexity of the problem. It can also be seen that this problem benefits significantly from parallelism. With a small number of stations (300 stations), the improvement is present and linear. However, as the number of stations grows bigger, the improvement of using more processes becomes drastic, offering close to 8x in performance on the xs-4114 processor when using 8 processes.

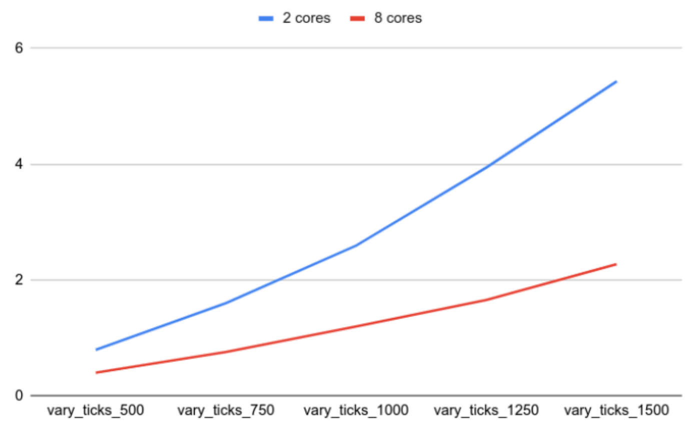
## Number of Links

The results (Appendix) show that varying the number of links has minimal impact on runtime. This may be due to the fact that computations for links are done efficiently in the local processes, hence do not require the additional overhead of message passing between processes. This suggests that the complexity of the problem is primarily determined by the number of stations in the train network rather than the number of links.

## Number of Trains

The results (Appendix) show that varying the number of trains has minimal impact on runtime. This may be due to the fact that computations for trains are done efficiently in the local processes, hence do not require the additional overhead of message passing between processes. This suggests that the complexity of the problem is primarily determined by the number of stations in the train network rather than the number of trains.

## Number of Ticks

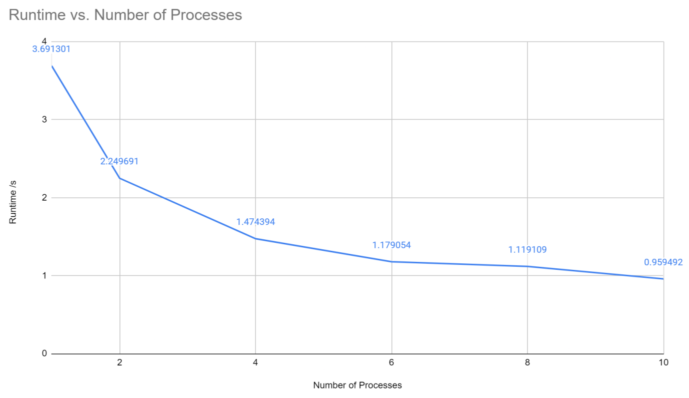


The results (Appendix) show that as the number of ticks grows, the runtime also grows almost linearly, as the program has to run for more iterations.

## Type of Cores

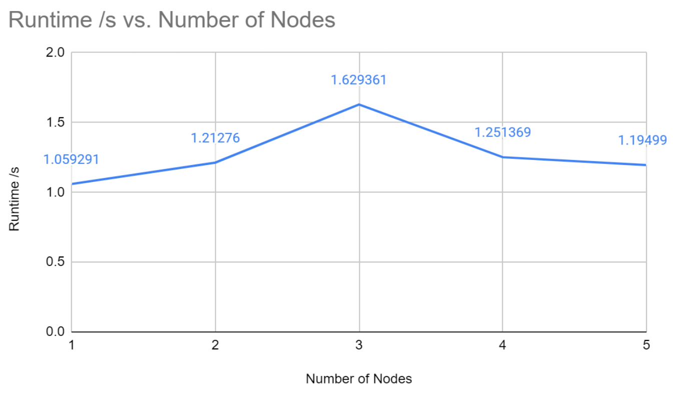
The results (Appendix) show that all processors benefit from parallelism, with lower runtimes using 8 processes compared to 2. Differences in performance across cores may be due to their architectures. The xs-4114 shows the largest runtime increase with fewer processes, becoming slower than the i7-7700, likely due to the i7-7700's faster clock speed.

## Number of Processes



The results (Appendix) show that as the number of processes increases, the runtime generally decreases, indicating that the parallelisation is effectively speeding up the program. However, the rate of improvement slows down with more processes, as seen from the diminishing gains when more processes are added. This may be due to the increasing overhead from communication between processes. Additionally, according to Amdahl's Law, speedup of parallel execution is limited by the fraction of the algorithm that cannot be parallelized. This can explain why the performance improvements diminish as more processes are added, with the non-parallelizable portions of the program becoming a limiting factor.

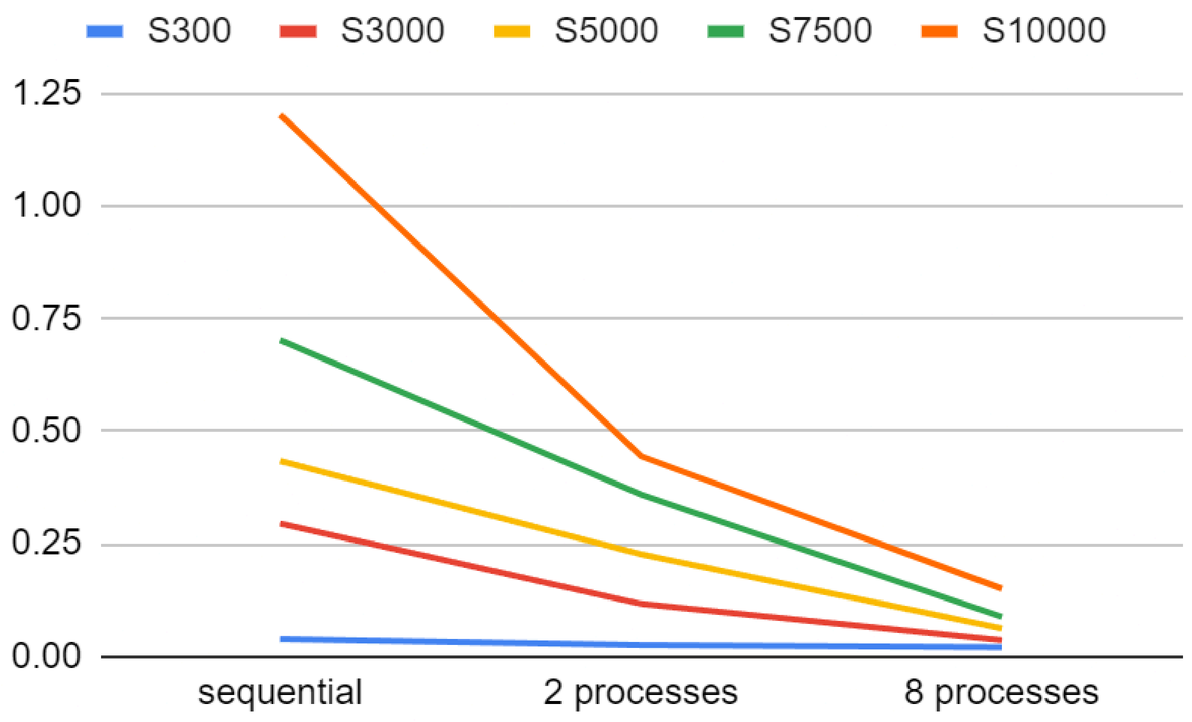
## Number of Nodes

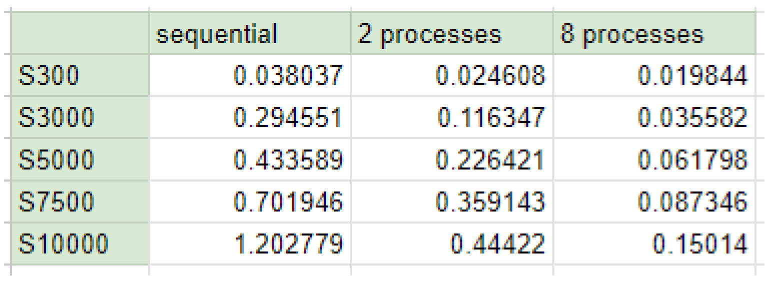


The results (Appendix) suggest that increasing the number of nodes can adversely affect the runtime of the program, with a noticeable spike in the runtime when using 3 nodes. These results may be attributed to the additional communication overhead of adding more nodes, as the time spent between communicating and synchronising data across more nodes becomes significant. This can thus lead to longer runtimes as the associated overhead may outweigh the benefits of increasing the number of nodes.

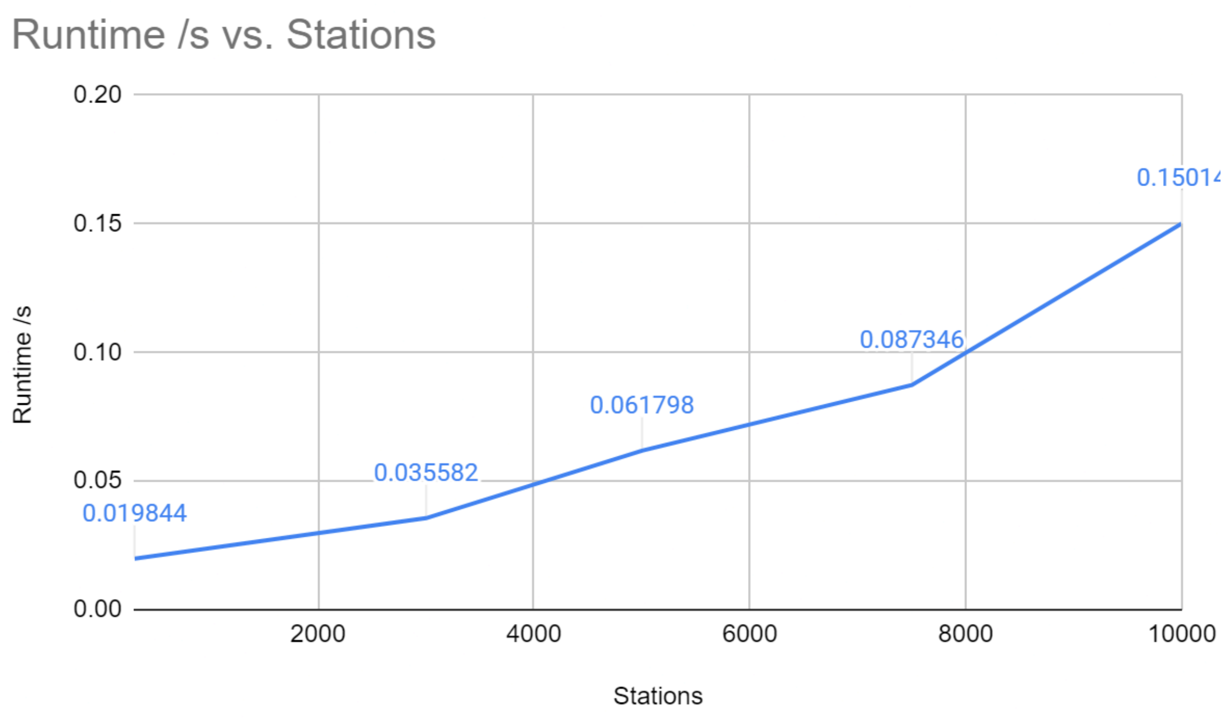
# Appendix

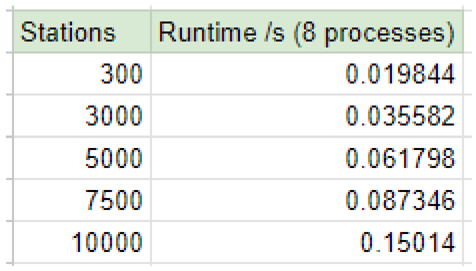
Number of Stations





| | sequential | 2 processes | 8 processes |
|---|---|---|---|
| S300 | 0.038037 | 0.024608 | 0.019844 |
| S3000 | 0.294551 | 0.116347 | 0.035582 |
| S5000 | 0.433589 | 0.226421 | 0.061798 |
| S7500 | 0.701946 | 0.359143 | 0.087346 |
| S10000 | 1.202779 | 0.44422 | 0.15014 |





| Stations | Runtime /s (8 processes) |
|---|---|
| 300 | 0.019844 |
| 3000 | 0.035582 |
| 5000 | 0.061798 |
| 7500 | 0.087346 |
| 10000 | 0.15014 |

Test Cases:

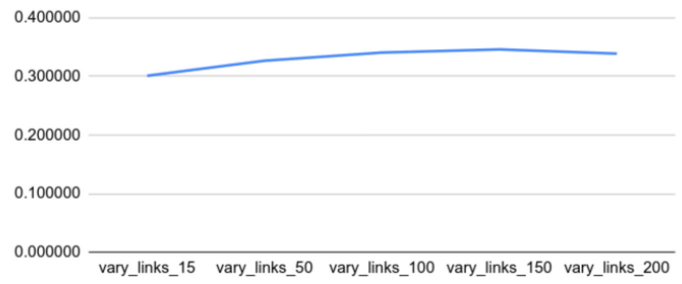
```
python3 gen_test.py 300 10 500 500 150 500 > testcases/performance/S300.in
python3 gen_test.py 3000 10 500 500 1500 500 > testcases/performance/S3000.in
python3 gen_test.py 5000 10 500 500 2500 500 > testcases/performance/S5000.in
python3 gen_test.py 7500 10 500 500 3750 500 > testcases/performance/S7500.in
python3 gen_test.py 10000 10 500 500 5000 500 > testcases/performance/S10000.in
```

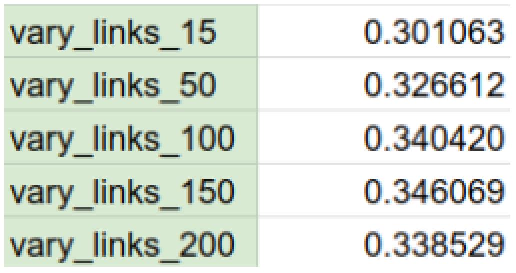
Command:

```
./bench_seq testcases/performance/S300.in > /dev/null
salloc --nodes 1 --ntasks 2 -p xs-4114 mpirun ./trains testcases/performance/S300.in > /dev/null
salloc --nodes 1 --ntasks 8 -p xs-4114 mpirun ./trains testcases/performance/S300.in > /dev/null

./bench_seq testcases/performance/S3000.in > /dev/null
salloc --nodes 1 --ntasks 2 -p xs-4114 mpirun ./trains testcases/performance/S3000.in >
/dev/null
```

```
salloc --nodes 1 --ntasks 8 -p xs-4114 mpirun ./trains testcases/performance/S3000.in >
/dev/null

./bench_seq testcases/performance/S5000.in > /dev/null
salloc --nodes 1 --ntasks 2 -p xs-4114 mpirun ./trains testcases/performance/S5000.in >
/dev/null
salloc --nodes 1 --ntasks 8 -p xs-4114 mpirun ./trains testcases/performance/S5000.in >
/dev/null

./bench_seq testcases/performance/S7500.in > /dev/null
salloc --nodes 1 --ntasks 2 -p xs-4114 mpirun ./trains testcases/performance/S7500.in >
/dev/null
salloc --nodes 1 --ntasks 8 -p xs-4114 mpirun ./trains testcases/performance/S7500.in >
/dev/null

./bench_seq testcases/performance/S10000.in > /dev/null
salloc --nodes 1 --ntasks 2 -p xs-4114 mpirun ./trains testcases/performance/S10000.in >
/dev/null
salloc --nodes 1 --ntasks 8 -p xs-4114 mpirun ./trains testcases/performance/S10000.in >
/dev/null
```

## Number of Links





| | |
|---|---|
| vary_links_15 | 0.301063 |
| vary_links_50 | 0.326612 |
| vary_links_100 | 0.340420 |
| vary_links_150 | 0.346069 |
| vary_links_200 | 0.338529 |

Test Cases:

```
python3 gen_test.py 30 10 10 500 15 500 > testcases/performance/vary_links_15.in
python3 gen_test.py 30 10 10 500 50 500 > testcases/performance/vary_links_50.in
python3 gen_test.py 30 10 10 500 100 500 > testcases/performance/vary_links_100.in
```

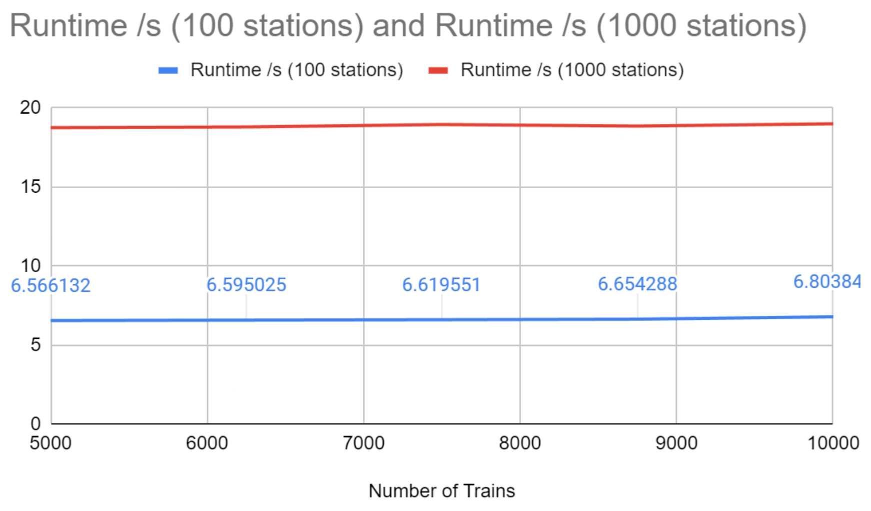
```
python3 gen_test.py 30 10 10 500 150 500 > testcases/performance/vary_links_150.in
python3 gen_test.py 30 10 10 500 200 500 > testcases/performance/vary_links_200.in
```

Commands:

```
salloc --nodes 1 --ntasks 8 -p xs-4114 mpirun ./trains testcases/performance/vary_links_15.in >
testcases/performance/vary_links_15.out
salloc --nodes 1 --ntasks 8 -p xs-4114 mpirun ./trains testcases/performance/vary_links_50.in >
testcases/performance/vary_links_50.out
salloc --nodes 1 --ntasks 8 -p xs-4114 mpirun ./trains testcases/performance/vary_links_100.in >
testcases/performance/vary_links_100.out
salloc --nodes 1 --ntasks 8 -p xs-4114 mpirun ./trains testcases/performance/vary_links_150.in >
testcases/performance/vary_links_150.out
salloc --nodes 1 --ntasks 8 -p xs-4114 mpirun ./trains testcases/performance/vary_links_200.in >
testcases/performance/vary_links_200.out
```

## Number of Trains

| Number of Trains | Runtime /s (100 stations) | Runtime /s (1000 stations) |
|---:|---:|---:|
| 5000 | 6.566132 | 18.746897 |
| 6250 | 6.595025 | 18.790194 |
| 7500 | 6.619551 | 18.944495 |
| 8750 | 6.654288 | 18.847813 |
| 10000 | 6.803849 | 18.991054 |



Test Cases:
100 stations:

```
python3 gen_test.py 100 10 10 5000 50 5000 > testcases/performance/vary_trains_500.in
```

```
python3 gen_test.py 100 10 10 6250 50 5000 > testcases/performance/vary_trains_625.in
python3 gen_test.py 100 10 10 7500 50 5000 > testcases/performance/vary_trains_750.in
python3 gen_test.py 100 10 10 8750 50 5000 > testcases/performance/vary_trains_875.in
python3 gen_test.py 100 10 10 10000 50 5000 > testcases/performance/vary_trains_1000.in
```

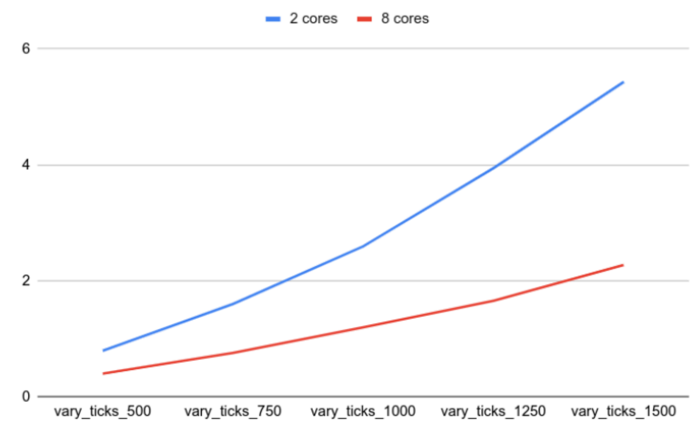
1000 stations:

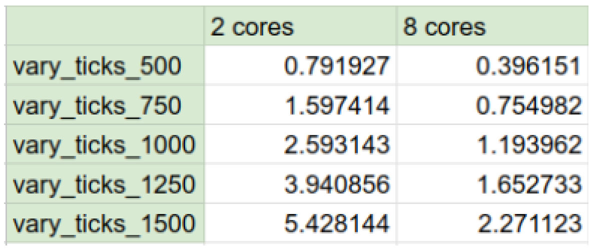
```
python3 gen_test.py 1000 10 10 5000 500 5000 > testcases/performance/vary_trains_500.in
python3 gen_test.py 1000 10 10 6250 500 5000 > testcases/performance/vary_trains_625.in
python3 gen_test.py 1000 10 10 7500 500 5000 > testcases/performance/vary_trains_750.in
python3 gen_test.py 1000 10 10 8750 500 5000 > testcases/performance/vary_trains_875.in
python3 gen_test.py 1000 10 10 10000 500 5000 > testcases/performance/vary_trains_1000.in
```

Commands:

```
salloc --nodes 1 --ntasks 8 -p xs-4114 mpirun ./trains testcases/performance/vary_trains_500.in
> testcases/performance/vary_trains_500.out
salloc --nodes 1 --ntasks 8 -p xs-4114 mpirun ./trains testcases/performance/vary_trains_625.in
> testcases/performance/vary_trains_625.out
salloc --nodes 1 --ntasks 8 -p xs-4114 mpirun ./trains testcases/performance/vary_trains_750.in
> testcases/performance/vary_trains_750.out
salloc --nodes 1 --ntasks 8 -p xs-4114 mpirun ./trains testcases/performance/vary_trains_875.in
> testcases/performance/vary_trains_875.out
salloc --nodes 1 --ntasks 8 -p xs-4114 mpirun ./trains
testcases/performance/vary_trains_1000.in > testcases/performance/vary_trains_1000.out
```

## Number of Ticks





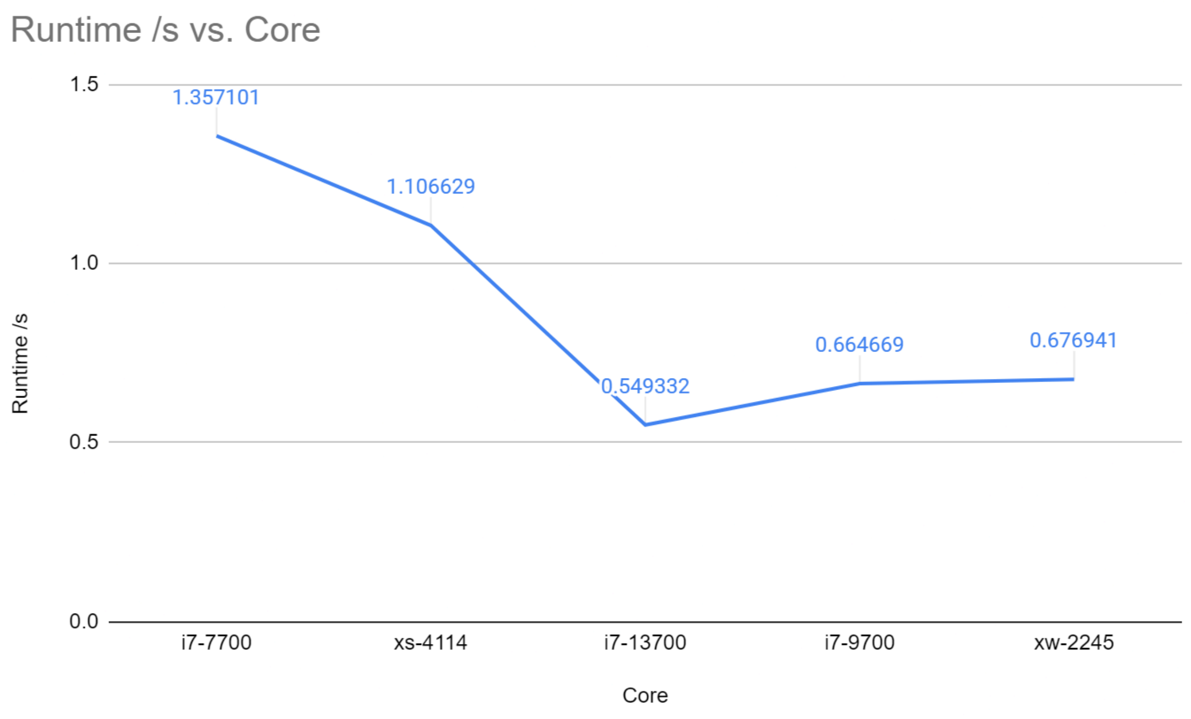
| | 2 cores | 8 cores |
|---|---|---|
| vary_ticks_500 | 0.791927 | 0.396151 |
| vary_ticks_750 | 1.597414 | 0.754982 |
| vary_ticks_1000 | 2.593143 | 1.193962 |
| vary_ticks_1250 | 3.940856 | 1.652733 |
| vary_ticks_1500 | 5.428144 | 2.271123 |

Test Cases:
```
python3 gen_test.py 300 10 10 500 150 500 > testcases/performance/vary_ticks_500.in
python3 gen_test.py 300 10 10 750 150 750 > testcases/performance/vary_ticks_750.in
python3 gen_test.py 300 10 10 1000 150 1000 > testcases/performance/vary_ticks_1000.in
python3 gen_test.py 300 10 10 1250 150 1250 > testcases/performance/vary_ticks_1250.in
python3 gen_test.py 300 10 10 1500 150 1500 > testcases/performance/vary_ticks_1500.in
```

Commands:
```
salloc --nodes 1 --ntasks 2 -p xs-4114 mpirun ./trains testcases/performance/vary_ticks_500.in >
testcases/performance/vary_ticks_500.out
salloc --nodes 1 --ntasks 2 -p xs-4114 mpirun ./trains testcases/performance/vary_ticks_750.in >
testcases/performance/vary_ticks_750.out
salloc --nodes 1 --ntasks 2 -p xs-4114 mpirun ./trains testcases/performance/vary_ticks_1000.in
> testcases/performance/vary_ticks_1000.out
salloc --nodes 1 --ntasks 2 -p xs-4114 mpirun ./trains testcases/performance/vary_ticks_1250.in
> testcases/performance/vary_ticks_1250.out
salloc --nodes 1 --ntasks 2 -p xs-4114 mpirun ./trains testcases/performance/vary_ticks_1500.in
> testcases/performance/vary_ticks_1500.out
salloc --nodes 1 --ntasks 8 -p xs-4114 mpirun ./trains testcases/performance/vary_ticks_500.in >
testcases/performance/vary_ticks_500.out
salloc --nodes 1 --ntasks 8 -p xs-4114 mpirun ./trains testcases/performance/vary_ticks_750.in >
testcases/performance/vary_ticks_750.out
salloc --nodes 1 --ntasks 8 -p xs-4114 mpirun ./trains testcases/performance/vary_ticks_1000.in
> testcases/performance/vary_ticks_1000.out
salloc --nodes 1 --ntasks 8 -p xs-4114 mpirun ./trains testcases/performance/vary_ticks_1250.in
> testcases/performance/vary_ticks_1250.out
salloc --nodes 1 --ntasks 8 -p xs-4114 mpirun ./trains testcases/performance/vary_ticks_1500.in
> testcases/performance/vary_ticks_1500.out
```

## Type of Cores

| Core | Runtime /s (8 processes) | Runtime /s (2 processes) |
|---|---|---|
| i7-7700 | 1.357101 | 1.601629 |
| xs-4114 | 1.106629 | 2.214022 |
| i7-13700 | 0.549332 | 1.230949 |
| i7-9700 | 0.664669 | 1.376151 |
| xw-2245 | 0.676941 | 1.461337 |



Test Cases:

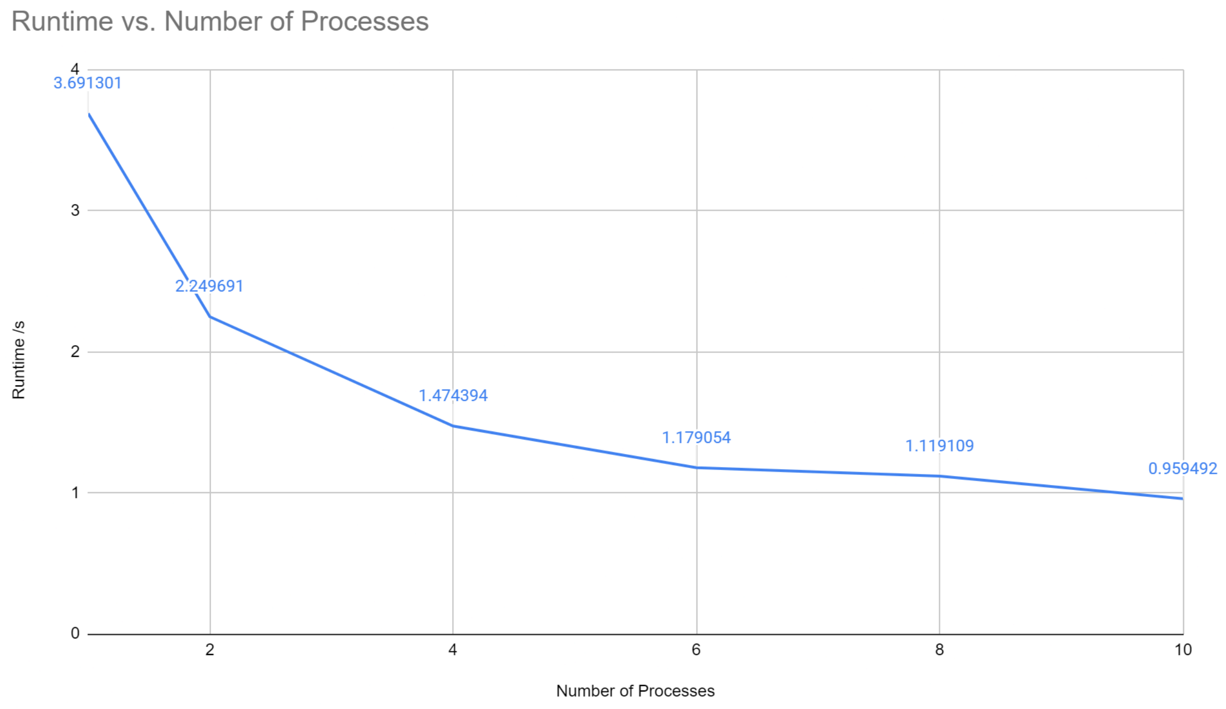
```
python3 gen_test.py 100 10 10 1000 50 1000 > testcases/performance/process_test.in
```

Commands (vary ntasks accordingly):

```
salloc --nodes 1 --ntasks 8 -p i7-7700 mpirun ./trains testcases/performance/process_test.in > process.out
salloc --nodes 1 --ntasks 8 -p xs-4114 mpirun ./trains testcases/performance/process_test.in > process.out
salloc --nodes 1 --ntasks 8 -p i7-13700 mpirun ./trains testcases/performance/process_test.in > process.out
salloc --nodes 1 --ntasks 8 -p i7-9700 mpirun ./trains testcases/performance/process_test.in > process.out
salloc --nodes 1 --ntasks 8 -p xw-2245 mpirun ./trains testcases/performance/process_test.in > process.out
```

## Number of Processes

| Number of Processes | Runtime /s |
|---|---|
| 1 | 3.691301 |
| 2 | 2.249691 |
| 4 | 1.474394 |
| 6 | 1.179054 |
| 8 | 1.119109 |
| 10 | 0.959492 |



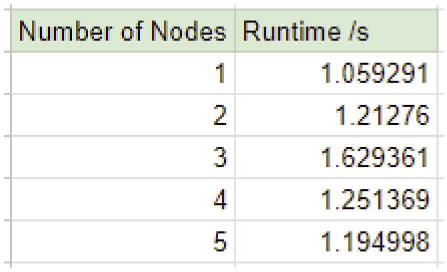
Test Cases:

```
python3 gen_test.py 100 10 10 1000 50 1000 > testcases/performance/process_test.in
```

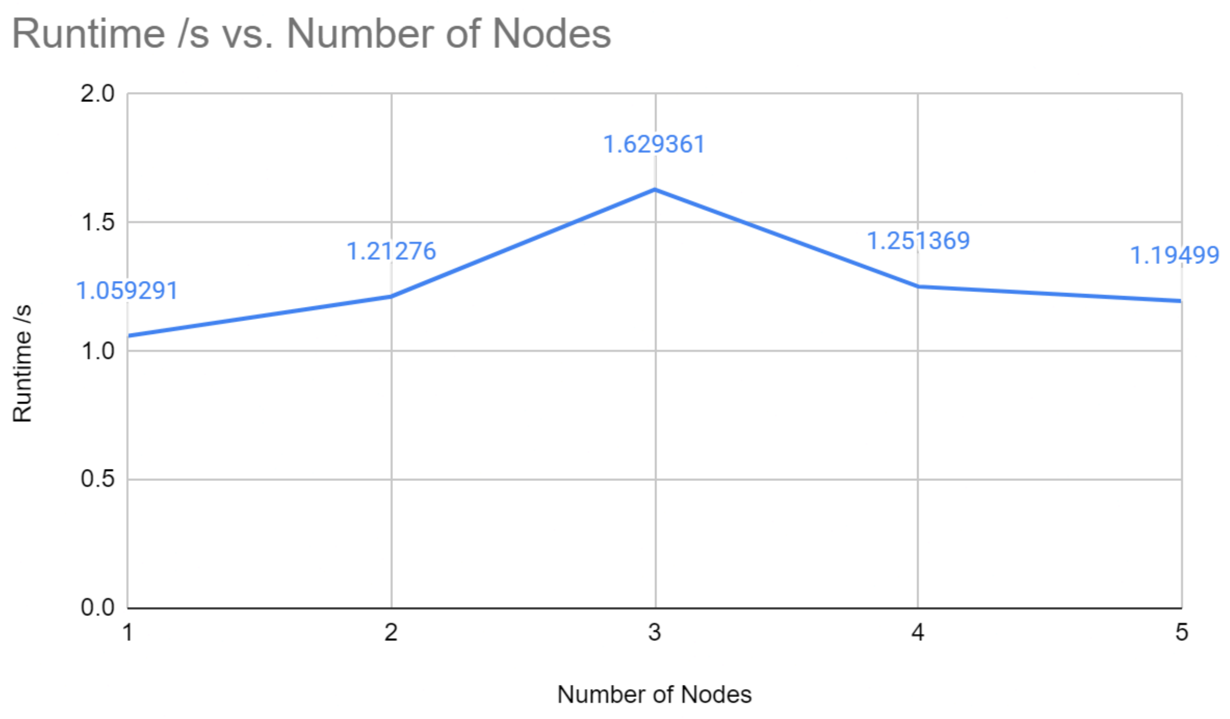
Commands:

```
salloc --nodes 1 --ntasks 1 -p xs-4114 mpirun ./trains testcases/performance/process_test.in > process.out
salloc --nodes 1 --ntasks 2 -p xs-4114 mpirun ./trains testcases/performance/process_test.in > process.out
salloc --nodes 1 --ntasks 4 -p xs-4114 mpirun ./trains testcases/performance/process_test.in > process.out
salloc --nodes 1 --ntasks 6 -p xs-4114 mpirun ./trains testcases/performance/process_test.in > process.out
salloc --nodes 1 --ntasks 8 -p xs-4114 mpirun ./trains testcases/performance/process_test.in > process.out
salloc --nodes 1 --ntasks 10 -p xs-4114 mpirun ./trains testcases/performance/process_test.in >
process.out
```

## Number of Nodes



| Number of Nodes | Runtime /s |
|---|---|
| 1 | 1.059291 |
| 2 | 1.21276 |
| 3 | 1.629361 |
| 4 | 1.251369 |
| 5 | 1.194998 |



Test Cases:

```
python3 gen_test.py 100 10 10 1000 50 1000 > testcases/performance/process_test.in
```

Commands:

```
salloc --nodes 1 --ntasks 8 -p xs-4114 mpirun ./trains testcases/performance/process_test.in >
process.out
salloc --nodes 2 --ntasks 8 -p xs-4114 mpirun ./trains testcases/performance/process_test.in >
process.out
```

```
salloc --nodes 3 --ntasks 8 -p xs-4114 mpirun ./trains testcases/performance/process_test.in >
process.out
salloc --nodes 4 --ntasks 8 -p xs-4114 mpirun ./trains testcases/performance/process_test.in >
process.out
salloc --nodes 5 --ntasks 8 -p xs-4114 mpirun ./trains testcases/performance/process_test.in >
process.out
```