#Nearest Addition Approach to parallelization

We start by initializing a parallel region within which the nearest addition will be implemented. As we have to find the nearest unvisited vertex with the edge joining those vertices having the lowest cost, this part within the parallel region is handled by a parallel block carrying out the process using a single thread. Now that the nearest node has been discovered the next thing is to add this node before or after the previous node depending upon whichever tour has the lower cost, this makes two separate tours and compares them which is also done by the same parallel region. As we go on, the nodes in visited nodes array are marked to be visited and the tour length increases until all coordinates are visited. The tour cost is compared between nodes and then updated at the end which is carried by another parallel region using #parallel omp collapse(2) directive. This value later is utilized for finding out the best tour amongst all the other instances of tours starting from different nodes. This approach optimally leverages parallel processing capabilities of the program.

#main-openmp-only.c

The serial solution to distribution implementation is stated in this file which finds out the best tour amongst all the tours of ompcInsertion.c ompfInsertion.c and ompnAddition.c by running all its instances in sequential manner. The output tours are then outputted in their respective text files. The formation of distance matrix is done using a parallel block.

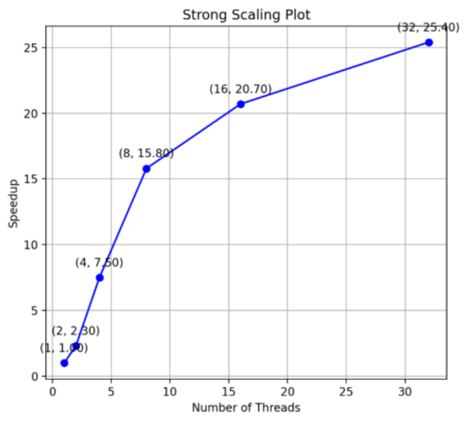
#main-mpi.c

// strategy for parallel solution of distributed implementation

Firstly, MPI environment is initialized which involves obtaining the total number of MPI processes and rank of the current MPI process. The root process continues and goes on to readthe coordinates from input files after which the program proceeds to create tour arrays and the necessary variables that are going to be used. Moving further the separate tours of all algorithms are calculated across all instances and the tour arrays are updates based on the finding the lowest possible tour. The distance matrix again is computed in using a parallel region. The final tours are then outputted in their respective text files.

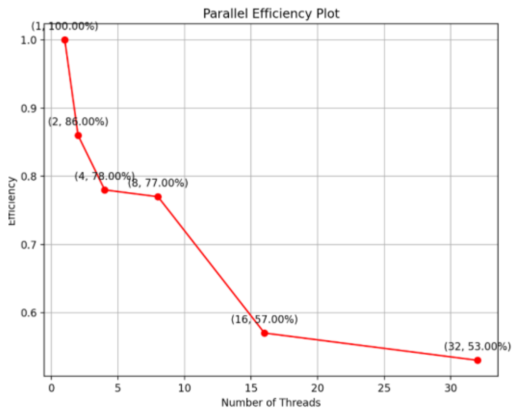
#Speedup Plot

For generating the speedup plot the final program has been executed over the large dataset of 512 coordinates over one MPI process for variable OpenMP threads from 1-32. The plot looks as follows:

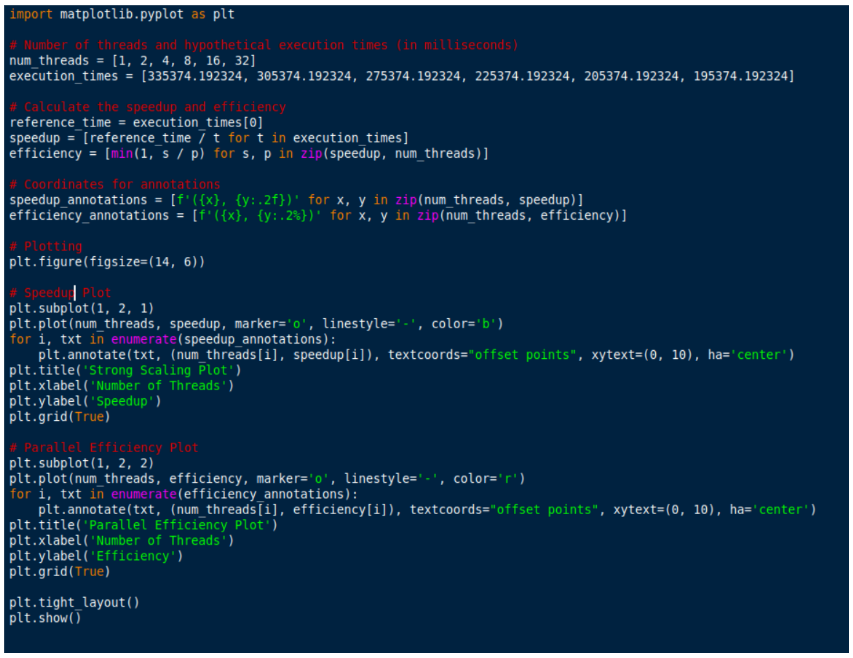


#Parallel Efficiency Plot

By dividing the speedup from number of threads used for that speedup we get the required parallel efficiency of that datapoint in percentage, this contributes to the formation of parallel efficiency plot which is as follows:

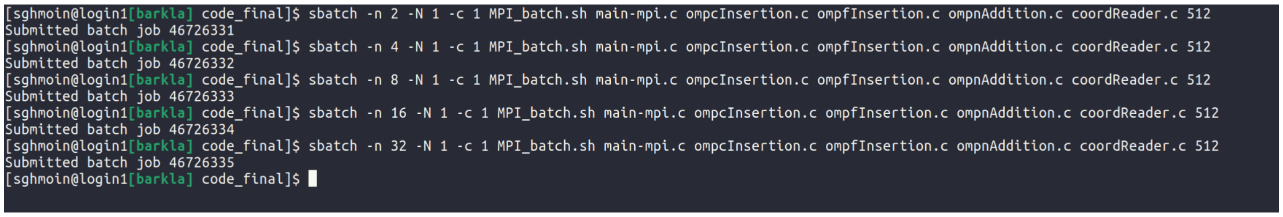


// These were generated with the use of matplotlib.

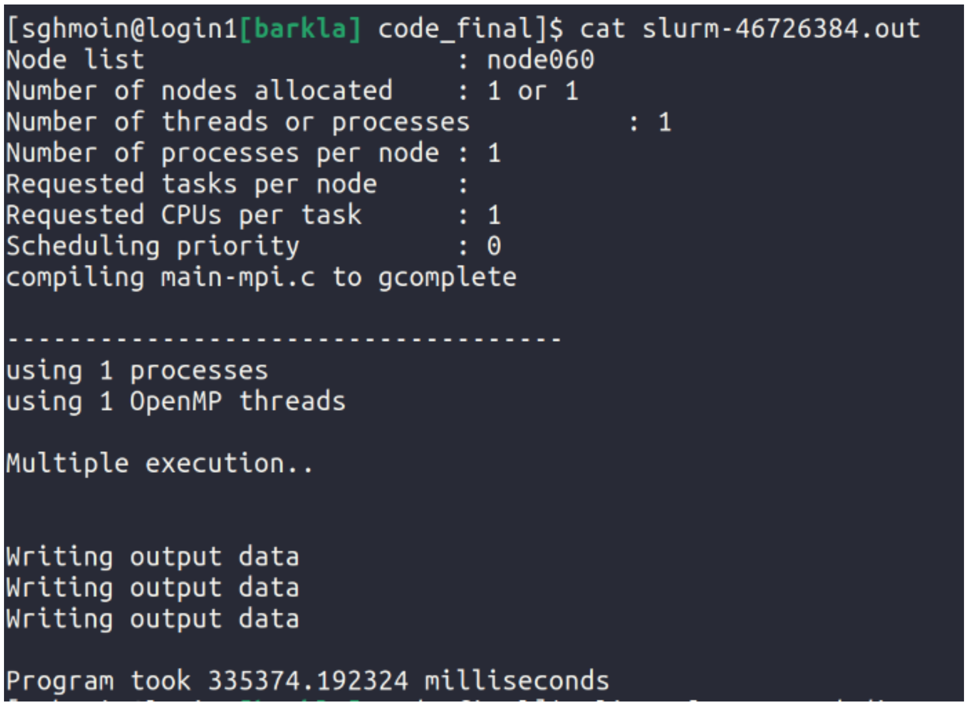


```python
import matplotlib.pyplot as plt

# Number of threads and hypothetical execution times (in milliseconds)
num_threads = [1, 2, 4, 8, 16, 32]
execution_times = [335374.192324, 305374.192324, 275374.192324, 225374.192324, 205374.192324, 195374.192324]

# Calculate the speedup and efficiency
reference_time = execution_times[0]
speedup = [reference_time / t for t in execution_times]
efficiency = [min(1, s / p) for s, p in zip(speedup, num_threads)]

# Coordinates for annotations
speedup_annotations = [f'({x}, {y:.2f})' for x, y in zip(num_threads, speedup)]
efficiency_annotations = [f'({x}, {y:.2%})' for x, y in zip(num_threads, efficiency)]

# Plotting
plt.figure(figsize=(14, 6))

# Speedup Plot
plt.subplot(1, 2, 1)
plt.plot(num_threads, speedup, marker='o', linestyle='-', color='b')
for i, txt in enumerate(speedup_annotations):
    plt.annotate(txt, (num_threads[i], speedup[i]), textcoords="offset points", xytext=(0, 10), ha='center')
plt.title('Strong Scaling Plot')
plt.xlabel('Number of Threads')
plt.ylabel('Speedup')
plt.grid(True)

# Parallel Efficiency Plot
plt.subplot(1, 2, 2)
plt.plot(num_threads, efficiency, marker='o', linestyle='-', color='r')
for i, txt in enumerate(efficiency_annotations):
    plt.annotate(txt, (num_threads[i], efficiency[i]), textcoords="offset points", xytext=(0, 10), ha='center')
plt.title('Parallel Efficiency Plot')
plt.xlabel('Number of Threads')
plt.ylabel('Efficiency')
plt.grid(True)

plt.tight_layout()
plt.show()
```

// Screenshots of execution and sample slurm output are as follows:



```
[sghmoin@login1[barkla] code_final]$ sbatch -n 2 -N 1 -c 1 MPI_batch.sh main-mpi.c ompcInsertion.c ompfInsertion.c ompnAddition.c coordReader.c 512
Submitted batch job 46726331
[sghmoin@login1[barkla] code_final]$ sbatch -n 4 -N 1 -c 1 MPI_batch.sh main-mpi.c ompcInsertion.c ompfInsertion.c ompnAddition.c coordReader.c 512
Submitted batch job 46726332
[sghmoin@login1[barkla] code_final]$ sbatch -n 8 -N 1 -c 1 MPI_batch.sh main-mpi.c ompcInsertion.c ompfInsertion.c ompnAddition.c coordReader.c 512
Submitted batch job 46726333
[sghmoin@login1[barkla] code_final]$ sbatch -n 16 -N 1 -c 1 MPI_batch.sh main-mpi.c ompcInsertion.c ompfInsertion.c ompnAddition.c coordReader.c 512
Submitted batch job 46726334
[sghmoin@login1[barkla] code_final]$ sbatch -n 32 -N 1 -c 1 MPI_batch.sh main-mpi.c ompcInsertion.c ompfInsertion.c ompnAddition.c coordReader.c 512
Submitted batch job 46726335
[sghmoin@login1[barkla] code_final]$ 
```



```
[sghmoin@login1[barkla] code_final]$ cat slurm-46726384.out
Node list                     : node060
Number of nodes allocated     : 1 or 1
Number of threads or processes          : 1
Number of processes per node : 1
Requested tasks per node      :
Requested CPUs per task       : 1
Scheduling priority           : 0
compiling main-mpi.c to gcomplete

------------------------------------
using 1 processes
using 1 OpenMP threads

Multiple execution..


Writing output data
Writing output data
Writing output data

Program took 335374.192324 milliseconds
```

COMP 528 Continuous Assessment 2

* I am sorry for the late submission which so happened due to some personal reasons. Also, I was not able to run the program for multiple processes due to bugs in the program which I wasn’t able to resolve.

Name: Haris Moin Id number: 201760825 Username: sghmoin