# Assignment 1

Jiakai Ni jiakain@student.unimelb.edu.au 988303
ZIRAN GU zirang1@student.unimelb.edu.au 1038782

## Implementation

### Tool

Python is used to develop 'The Happiest City' project. In order to parallel programming on the distributed system, MPIe for python (mpi4py) is used in this project.

### More than one core

When more than one process is available, the master and slave model is implemented. Process 0 will be treated as master, and all the rest processes will be treated as slave. And here is how the application work:

1. Master process reads 'melbGrid' and sentiment words information, then broadcasts the information to the slave processes.

2. Each slave has a counter, which could count the number of Tweets and sentimental score for each cell.

3. Master process reads a batch of tweets data each time, the data will be sent to the slave node.

4. Each slave receives and processes the tweets, then calculates the number of Tweets and sentimental scores.

5. When all the data is processed, the master process sum up the information gathered from the slave processes.

### 1 node and 1 core

When only 1 node and 1 core is used, the whole task will be processed in that core, hence there is no need to pass information through the process. And here is how the application work:

1. process reads 'melbGrid' and sentiment words information

2. process reads one tweets data each time, updates the number of Tweets and sentimental scores based on the data.

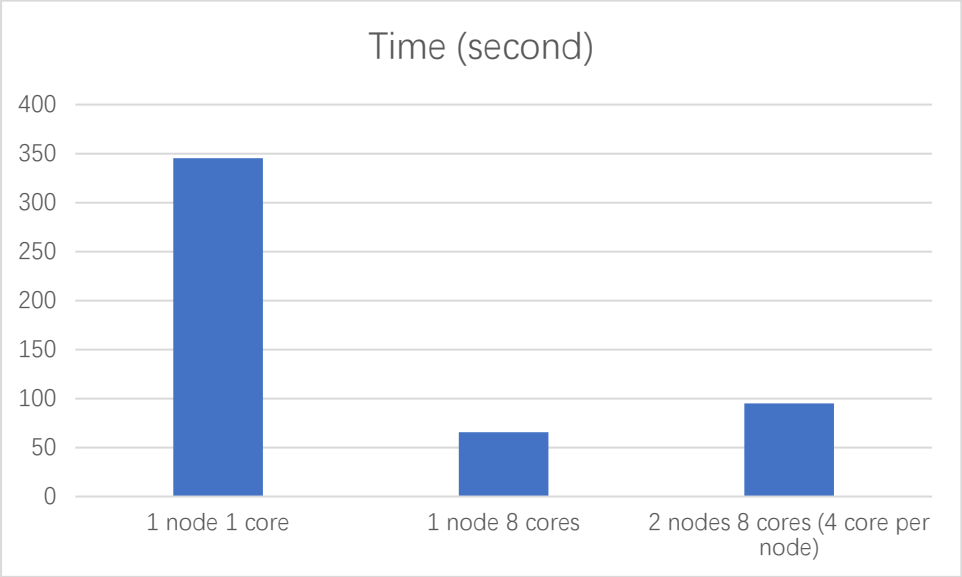
## Slurm

```
#!/bin/bash
# Created by the University of Melbourne job script generator for
SLURM
# Wed Mar 31 2021 13:15:04 GMT+1100 (Australian Eastern Daylight
Time)

# Partition for the job:
#SBATCH --partition=physical

# The name of the job:
#SBATCH --job-name="MelbTwitter_1_1"

# The project ID which this job should run under:
#SBATCH --account="COMP90024"

#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --time=0-0:10:00

# check that the script is launched with sbatch
if [ "x$SLURM_JOB_ID" == "x" ]; then
   echo "You need to submit your job to the queuing system with
sbatch"
   exit 1
fi

# Run the job from the directory where it was launched (default)

# The modules to load:
module load gcc/8.3.0 openmpi/3.1.4 python/3.7.4

echo "nodes=1 ntasks=1"
# The job command(s):
echo "tinyTwitter output:"
time mpirun python3 test.py melbGrid.json AFINN.txt tinyTwitter.json

echo "smallTwitter output:"
time mpirun python3 test.py melbGrid.json AFINN.txt smallTwitter.json
```

```
echo "bigTwitter output:"
time mpirun python3 test.py melbGrid.json AFINN.txt bigTwitter.json
```

*If 1node1cores, set nodes=1, ntasks=1;*
*If 1node8cores, set nodes=1, ntasks=8;*
*If 2node8cores, set nodes=2, ntasks=4.*

## Analysis




| Recourse | Time (second) |
|---|---|
| 1 node 1 core | 345.352 |
| 1 node 8 cores | 65.706 |
| 2 nodes 8 cores (4 core per node) | 94.985 |

When only 1 node 1 core is used, no communicate between nodes or cores is required, hence all the time is spent on calculation. When more than one cores is used, the speed improvement will highly related to communication cost and how many tasks can be processed paralleling. Since master and slave is implemented, the master core will read all information and send them to slave cores to processed (calculate total number of Tweets and Sentiment Score). Hence the information reading part can't be done parallelly, only processed part can be done parallelly. Therefore, compare to using one core, multi-core could save processed time, but communication cost will increase. In addition, the communication between nodes depends on the network. In the case of poor network conditions, the communication will take more time.

Therefore, 1node8cores as a single node is more suitable for this job situation. According to the above, 1 node 8 cores should faster than 2 node 8 cores which faster than 1 node 1 core.

from the experimental data, we could see that, in the case of 1 node 1 code, time is mainly spent on processed data, since it couldn't be done parallelly. ; in the case of 1 node 8 cores, there will be more time for 1 code to broadcast data, and reading Twitter data and processing Twitter data is done in parallel by other 7 codes, parallel processed the time deduct on process in parallel in significantly more than time spend on communication. Hence 1 node 8 cores is significantly faster than1 node 1 core; in the case of 2 node 8 cores, it's little bit slower than 1 node 8 cores. This is because it will take longer time to communicate between the two nodes than 1 node 8 core.
It is not difficult to find that node 1 code 8 is about five to six times faster than 1 node 1 code. This is in line with our initial guess. This is also in line with Amdahl's Law. Using 8 codes will not be 8 times faster. These conclusions can be proved through The theorem formula of Amdahl's Law:

$$T(1) = \sigma + \pi \approx 345s$$
$$T(8) = \sigma + \pi/8 \approx 65s$$
$$Speedup(p) = \frac{T(1)}{T(8)} \approx 5.307$$

T(1) represents 1node1core and T(8) represents 1 node 8cores. It can be found is that in the time calculation, the time to read the data cannot be ignored, so even with 8 times the core, the speed will not increase by 8 times.
Comparing 1 node 8 cores and 2 node 8 cores, although the number of cores is the same, 1 node 8 cores is faster than 2 node 8 cores. According to the above, this is because the communication between nodes will take extra time and the data needs to be merged between the two nodes.

Future improve performance
Currently, one core is used as the master to read data and broadcast tasks. In future research, all cores can read and process the data together, which may be faster.