COMP90024
Assignment 1 Report

Student: Rong Wang

Student ID: 1373126

Email: rongwang@student.unimelb.edu.au

1. Artacthure

One master, others workers.

My design to utilize multiple cores is to first divide the JSON file into small parts, evenly distributing them across the cores. Then, each core parses its assigned JSON segment, extracts the necessary information, and writes it to the relevant file. Once all parsing is complete, the master core reads the parse results into a hash table to determine which is the happiest.

Note that the master also parses JSON; the difference is that it is responsible for splitting the JSON file and collecting the results.

2. Result

```
Executing script: 2node8core.slum
rm -f sample_mpi *.o *.tmp
rm -f *.tmp
mpicxx -std=c++17 -O2 -I./simdjson -o sample_mpi sample_mpi.cpp simdjson/simdjson.cpp
The happiest hour are:
1st happiest hour is 2025.1.1 at 0 with score: 206.152
2st happiest hour is 2024.12.31 at 23 with score: 187.465
3st happiest hour is 2025.1.1 at 5 with score: 135.925
4st happiest hour is 2024.12.24 at 22 with score: 117.286
5st happiest hour is 2024.12.25 at 15 with score: 114.602
The saddest hour are:
1st saddest hour is 2024.11.6 at 7 with score: -373.767
2st saddest hour is 2024.9.11 at 1 with score: -305.576
3st saddest hour is 2025.1.30 at 17 with score: -256.568
4st saddest hour is 2025.1.30 at 18 with score: -226.91
5st saddest hour is 2025.2.3 at 16 with score: -223.833
The happiest people are:
1st happiest people is id:110237351908820391 gameoflife with score: 9105.74
2st happiest people is id:112728392129924952 TheFigen_ with score: 2541.47
3st happiest people is id:113441357479871732 EmojiAquarium with score: 2086.46
4st happiest people is id:113541715572887376 choochoo with score: 1978.74
5st happiest people is id:110006430181118061 hnbot with score: 1917.79
The saddest people are:
1st saddest people is id:109521050152429017 realTuckFrumper with score: -9094.08
2st saddest people is id:109471254002284799 uavideos with score: -5901.96
3st saddest people is id:113443732887173058 TheHindu with score: -5710.72
4st saddest people is id:111873606251312850 uutisbot with score: -4066.14
5st saddest people is id:112071598249945636 MissingYou with score: -3341.6

Total time: 35.9436 seconds
```

1 node 1 core execute time:

1. 176.49s
2. 187.48s

3. 187.93s

Average 183.97 seconds.

1 node 8 core execute time:

1. 43.94s
2. 37.12s
3. 43.70s

Average 41.59 seconds.

2 node 8 core execute time:

1. 37.35s
2. 36.17s
3. 35.94s

Average 36.49 seconds.

For exploring Amdahl's law, I recorded the parsing time in a 1-node 8-core environment. The results show that parsing takes around 27 seconds, while the final result collection and hash map building take 9 seconds. Therefore, eight cores achieve a 6.4× speedup compared to one core. This speedup is not surprising, as my code parses JSON much faster; thus, certain hardware elements may become bottlenecks as the number of cores increases, such as disk read speed and memory bandwidth.

I cannot explain why two nodes are faster than one node—maybe it's due to the memory bandwidth? I have no detailed knowledge of Spartan hardware, so this is just a guess. But it is not surprising that I designed to minimize communications with cores.

Note that it is obvious that the final parse result collection and hash table building must be done on a single core (ignoring hash table building for multiple threads). There is also a way to improve the program's speed: pipelining—building the hash table while parsing JSON. This is achievable because the JSON file has been split and parse results become available

while parsing is ongoing.

3. Implement concern

My goal is to make the most use of the hardware and employ various tricks to achieve maximum performance. The report must be brief, so I have no space to elaborate here. Also, it seems like no one cares.

4. Usage

Just run the sbatch SLURM scripts in the script/ directory. For example:

sbatch 1node8core.slum

This works on Spartan. Note that the JSON file path is hard-coded in the C++ file; you have to change it at line 30 in sample_mpi.cpp.