EECE 5640: High Performance Computing

Assignment 2

Jiayun Xin

NUID: 001563582

College of Engineering

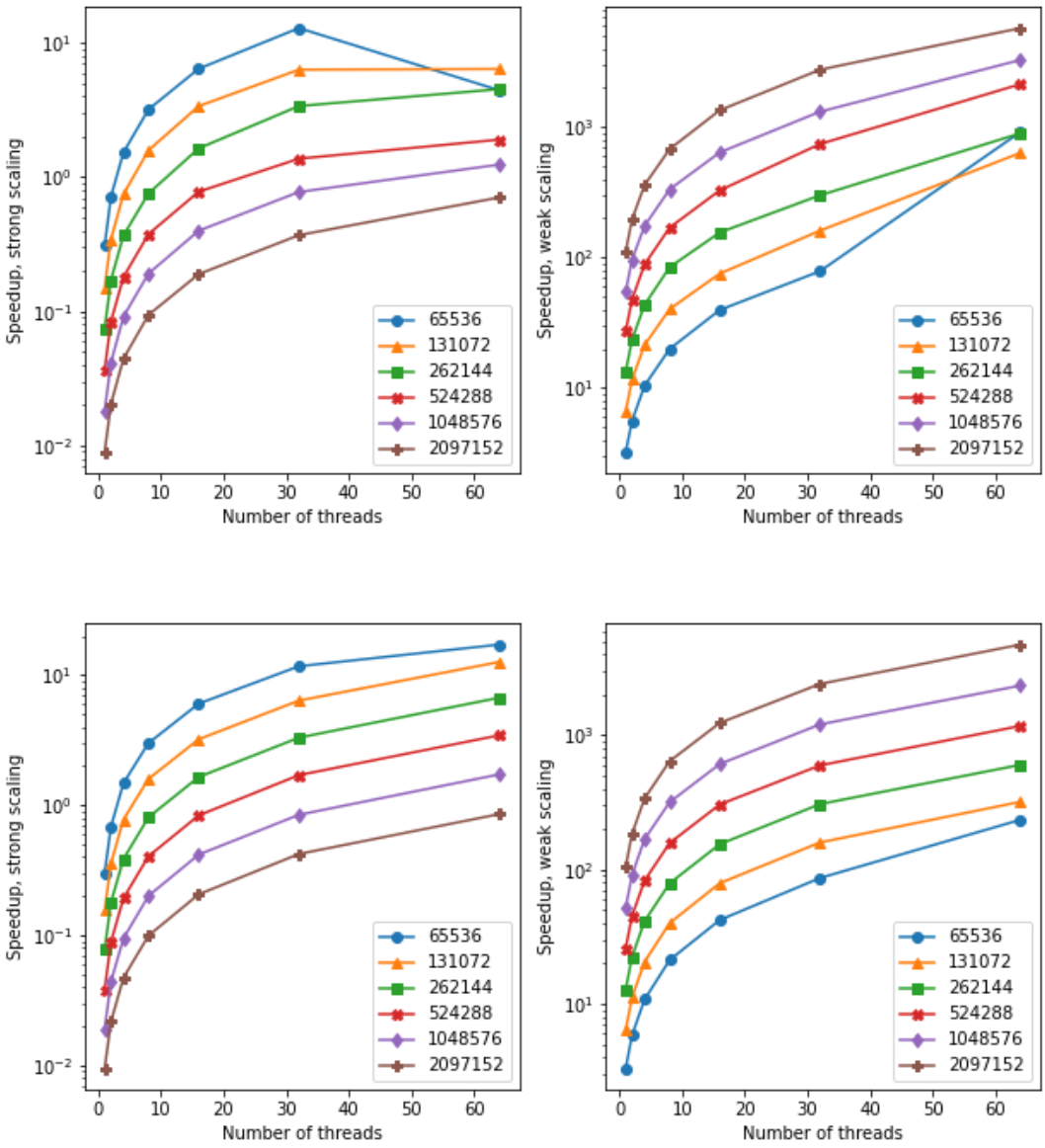
Northeastern University Boston, Massachusetts

Fall, 2021

1.
Edgar Dijkstra is a computer scientist, programmer, software engineer, systems scientist, science essayist and pioneer in computing science in Dutch. His contributions refer to compiler construction, operating systems, distributed systems and concurrent programming etc., in which his study of concurrent computer pioneered the field of concurrency, semaphores, mutual, deadlock and finding shortest paths in graphs. He wrote the first paper to identify and solve the problem of mutual exclusion problem.
Dining philosophy problem can be abstracted as concurrent algorithm to avoid dead lock. Philosophies are views as threads and forks are mutex lock. In read world, different programs should avoid access to shared memory simultaneously. Some complex systems may use thousands of locks and synchronization, which should strictly follow protocols to avoid deadlock and starvation.
Dijkstra's algorithm is a method to find the shortest path between 2 given nodes in a graph. Each time, the lowest distance neighbor node will be pick until the final node is reached. This algorithm is used with priority queue or priority heap for optimization.

a) The number of philosophers does not impact my solution.

2.
c.)



The above 4 screenshots show the results of executing Sieve of Eratosthenes by using pthread and openmp. The up 2 pictutes are the results of openmp and the down 2 pictures are the results of pthread. Those lines with different color refer to the different size of problem scale.
Strong scaling is defied that a fixed problem size has a limit speedup. As we can see, both results of pthread and openmp lead to a constant speedup as number of threads increase, but the speedup of using openmp has a sudden decrease when the size of problem scale is 65536 and thread number is 64.
Weak scaling states there is a linear relation between speedup and the number of threads when the problem size is fixed. Actually, the relation on the pictures above is not obvious linear. The trends of two implementations are almost the same but the blue one. The speedup has a sudden increase for using openmp when the number of threads is 64 and problem size is 65536.

The system is shown below:

```
Architecture:                    x86_64
CPU op-mode(s):                  32-bit, 64-bit
Byte Order:                      Little Endian
Address sizes:                   43 bits physical, 48 bits virtual
CPU(s):                          64
On-line CPU(s) list:             0-63
Thread(s) per core:              2
Core(s) per socket:              16
Socket(s):                       2
NUMA node(s):                    2
Vendor ID:                       AuthenticAMD
CPU family:                      23
Model:                           49
Model name:                      AMD EPYC 7282 16-Core Processor
Stepping:                        0
CPU MHz:                         2794.823
BogoMIPS:                        5589.64
Virtualization:                  AMD-V
L1d cache:                       1 MiB
L1i cache:                       1 MiB
L2 cache:                        16 MiB
L3 cache:                        128 MiB
```

Strong scaling: speedup = 1 / (s + p / N)

Weak scaling: speedup = s + p * N

Extra questions

a)

Using light-weight threading library can accelerate the execution of codes of using open MP, but different LWT library have different PM and the specific functions in library may vary.

b)

I choose Argobot. I use this library to create my own instance by dividing the whole vector into many pieces based on the number of threads I can use. Using these threads to add two vectors simultaneously.

c)

The BLAS-1 supergrams include a series of mathematical operations on a single vector or a pair of vectors, for example, vector arguments, dot product subprograms, scalar-vector multiplication, set up given rotation, , plane rotation and transformation.

d)

Nested parallel structures and nested tasks can arrange each thread to execute work dynamically and it also has an advantage of dynamic allocation of memory depending on threads to increase the efficiency of memory usage.