Computing Weak and Strong Scalability

Nandhana Sakthivel

Exercise 1

Aim

The aim of this exercise is to get a deep knlowledge about the weak and strong scalability. Here we are using a toy application: a Monte-Carlo integration of a quarter of a unit circle to compute the number PI (given as area computed*4). We provide a basic implementation of the algorithm (program pi.c) and we also give a parallel MPI implementation (MPIpi.c) that computes PI by the same algorithm using MPI to compute the algorithm in parallel.

Introduction

This exercise aims for a deeper understanding of how the above mentioned application scales up to the total number of cores of one node. It is done by using OpenMPI which is an open source MPI-2 implementation for performing parallel processing using a distributed memory model.

Serial Vs Parallel Implementation

We run the above mentioned application for the calculation of PI in both serial and parallel implementation to compare and understand the overhead associated with both. The application in serial implementation took a wall time of **0.0200s** for 10 million iterations where as the application in parallel implementation took a wall time of **0.02025s** for 10 million iterations using the MPI code. The time taken by both the implementation is more or less the same but we can see some overhead in parallel implementation.

Scalability

Scalability is the measure of a parallel system's capacity to increase the speedup in proportion to the number of processors.

The speedup in parallel computing can be straightforwardly defined as

Speedup = T(1) / T(N)

where T(1) is the computational time for running the application using one processor, and T(N) is the computational time for running the same application with N processors.

Strong Scalability

Strong scaling concerns the speedup for a fixed problem size with respect to the number of processors, and is governed by Amdahl's law. Amdahl's law can be formulated as follows

Speedup = 1 / (s + p / N)

where s is the proportion of execution time spent on the serial part, p is the proportion of execution time spent on the part that can be parallelized, and N is the number of processors. Amdahl's law states that, for a fixed problem, the upper limit of speedup is determined by the serial fraction of the code.

The strong scaling test is carried out for N = $10^6, 10^7, 10^8$

Figure 1 shows that the run time decreases as the number of processors increases with a fixed problem size and the **Figure 2** shows that the speedup increases with increase in the number of processors.

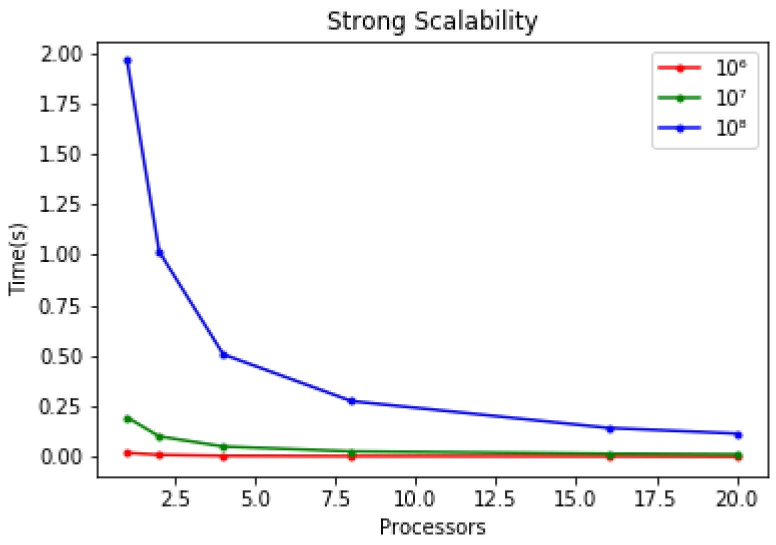

Figure 1: Strong Scalability: Processors vs Time(s)

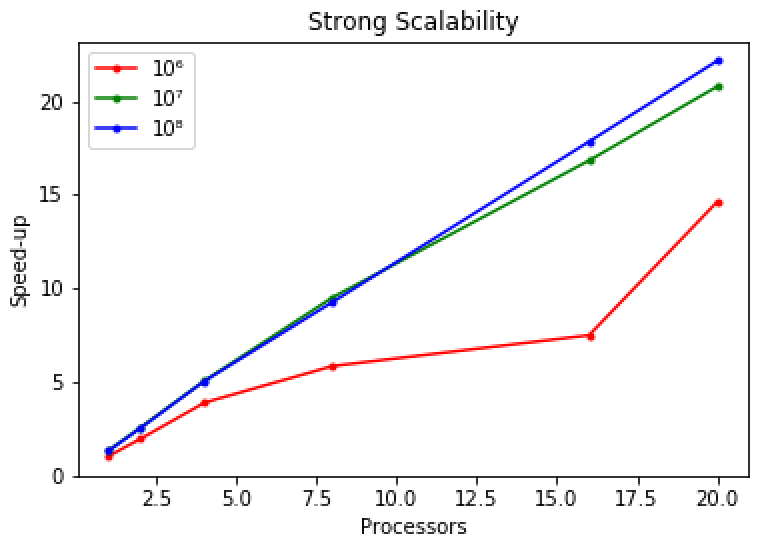

Figure 2: Strong Scalability: Processors vs Speedup

Weak Scalability

Weak scaling concerns the speedup for a scaled problem size with respect to the number of processors, and is governed by Gustafson's law. Gustafson's law is based on the approximations that the parallel part scales linearly with the amount of resources, and that the serial part does not increase with respect to the size of the problem. It provides the formula for scaled speedup

Scaled speedup = s + p N

where s, p and N have the same meaning as in Amdahl's law. With Gustafson's law the scaled speedup increases linearly with respect to the number of processors, and there is no upper limit for the scaled speedup.

The weak scaling test is carried out for N = $10^6, 10^7, 10^8$

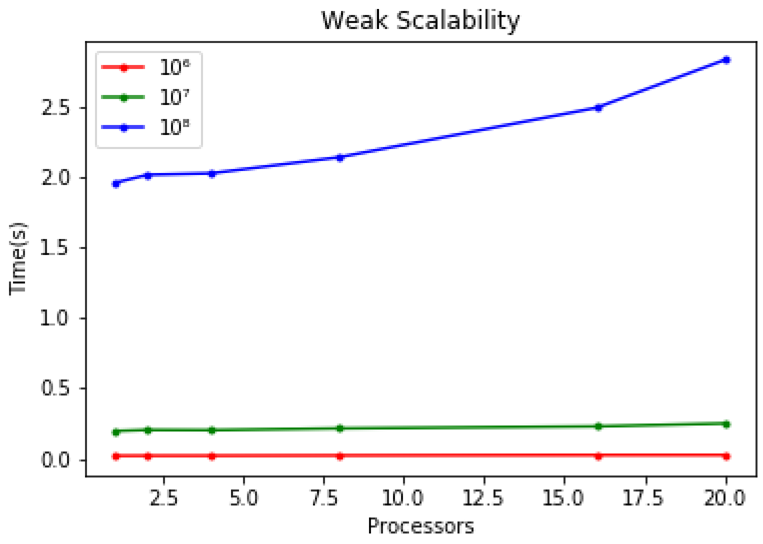

Figure 3: Weak Scalability: Processors vs Time(s)

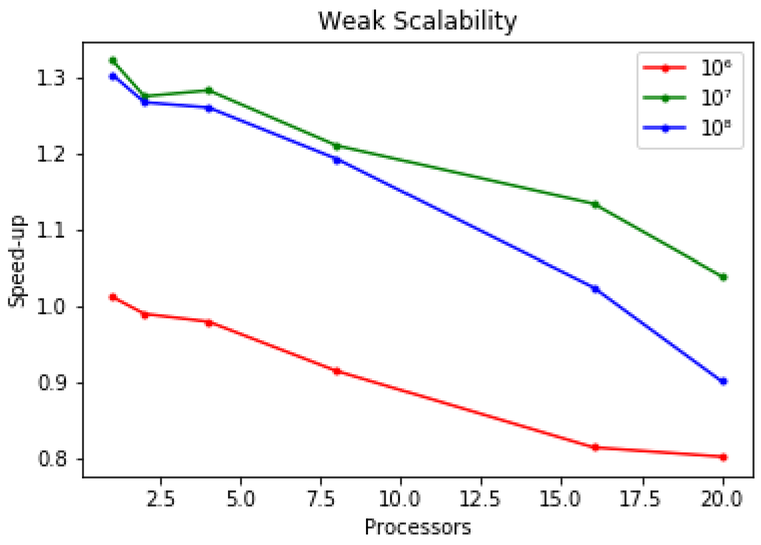

Figure 4: Weak Scalability: Processors vs Speedup

Figure 3 shows that the run time remains almost constant as the problem size and the number of compute nodes increase in proportion. and the **Figure 4** shows the speedup for weak scaling and the decrease in speedup can be explained by the communication overhead.

Conclusion

Strong Scaling Efficiency

If the amount of time to complete a work unit with 1 processing element is T(1), and the amount of time to complete the same unit of work with N processing elements is T(N), the strong scaling efficiency (as a percentage of linear) is given as:

$$T(1) \,/\, (\, N * T(N) \,) * 100$$

Weak Scaling Efficiency

If the amount of time to complete a work unit with 1 processing element is T(1), and the amount of time to complete N of the same work units with N processing elements is T(N), the weak scaling efficiency (as a percentage of linear) is given as:

$$(T(1)\ /\ T(N)\ )\ *\ 100$$

Here in **Figure 5** we are comparing the efficiency of weak and strong scaling achieved for this application. We can see that the efficiency of weak scaling decreases slower than strong scaling efficiency for larger problems with respect to the number of processors. This is because according to Gustafon's law the increased problem size only affects the parallel part of the program and we can handle bigger problems with more parallelism. Hence, weak scaling is better than strong scaling for bigger problems.

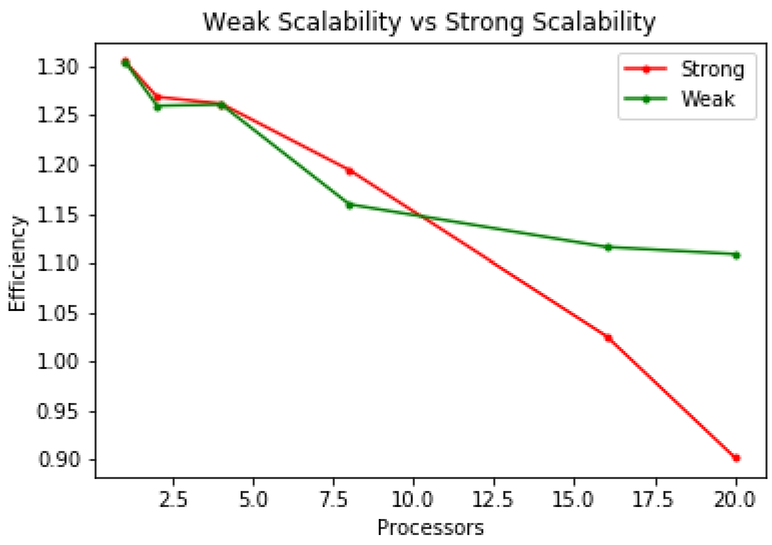

Figure 5: Strong Vs Weak Scaling Efficiency