Analysis of Strong and Weak Scalability Of A HPC Application

Rabindra Khadka

DSSC, UNITS

Overview:

Parallel execution have limits; as the number of processor count increases for performing a parallel application, relative performance shrinks. This deviation from the expected increase in speedup for parallel computation highlights the significance of careful selection of number of processors to run a task or application. Depending upon the size of a task or application, the suitable number of processors differ so it is vital to measure the performance with different number of processors. Usually the bigger the task, better the performance of additional processors in parallel computing. So, in this exercise different sizes of the task were tested against a range of processor counts of one node and times were recorded to calculate the speedup at each processor number.

This report analyses the weak and strong scalability of an application that estimates the value of Pi by using Monte Carlo simulation which randomly selects the points inside a square and determines the ratio of points that satisfies the equation of a circle and the total points. Serial and Parallel MPI code were provided for this exercise.

Strong Scaling:

During the strong scaling test, our problem size was kept fixed and the core counts were increased up to 20. Some of the questions raised before performing the strong scaling tests were;

- Will my calculation gets halved if I double the number of processors?
- And similarly if I increase my processor by 4 fold, will the calculation time be reduced by one-fourth and so on. If not, what could be the reason behind the phenomena?

Basically, the aim was to see how the performance of the parallel application changes with increase in processor counts. While being aware the notion of ideal time, we used Speed up as the performance metric for our tests.

$$S(N,P) = \frac{T(N,1)}{T(N,P)}$$

Where,

S = Speed up
N = Task Size
P = No. of Processors

No of Processor	RealTime(N=10^4)	Speed Up (N=10^4)	RealTime(N=10^5)	Speed Up (N=10^5)	RealTime(N=10^8)	Speed Up (N=10^8)	RealTime(N=10^9)	Speed Up (N=10^9)	Ideal Speedup	Ideal Time(N=10^9)
1.0	1.688	1.000000	1.681	1.000000	3.469	1.000000	21.299	1.000000	1.0	21.299000
2.0	1.544	1.093264	1.547	1.086619	2.516	1.378776	11.643	1.829340	2.0	10.649500
4.0	1.589	1.062303	1.591	1.056568	2.038	1.702159	6.631	3.212034	4.0	5.324750
8.0	1.632	1.034314	1.637	1.026878	1.912	1.814331	4.323	4.926903	8.0	2.662375
16.0	1.773	0.952059	1.809	0.929243	1.921	1.805830	3.189	6.678896	16.0	1.331187
18.0	1.803	0.936217	1.819	0.924134	1.920	1.806771	3.052	6.978702	18.0	1.183278
20.0	1.832	0.921397	1.834	0.916576	1.885	1.840318	2.949	7.222448	20.0	1.064950

Table 1: Strong Scaling Data with the measure of Speed up for different task size

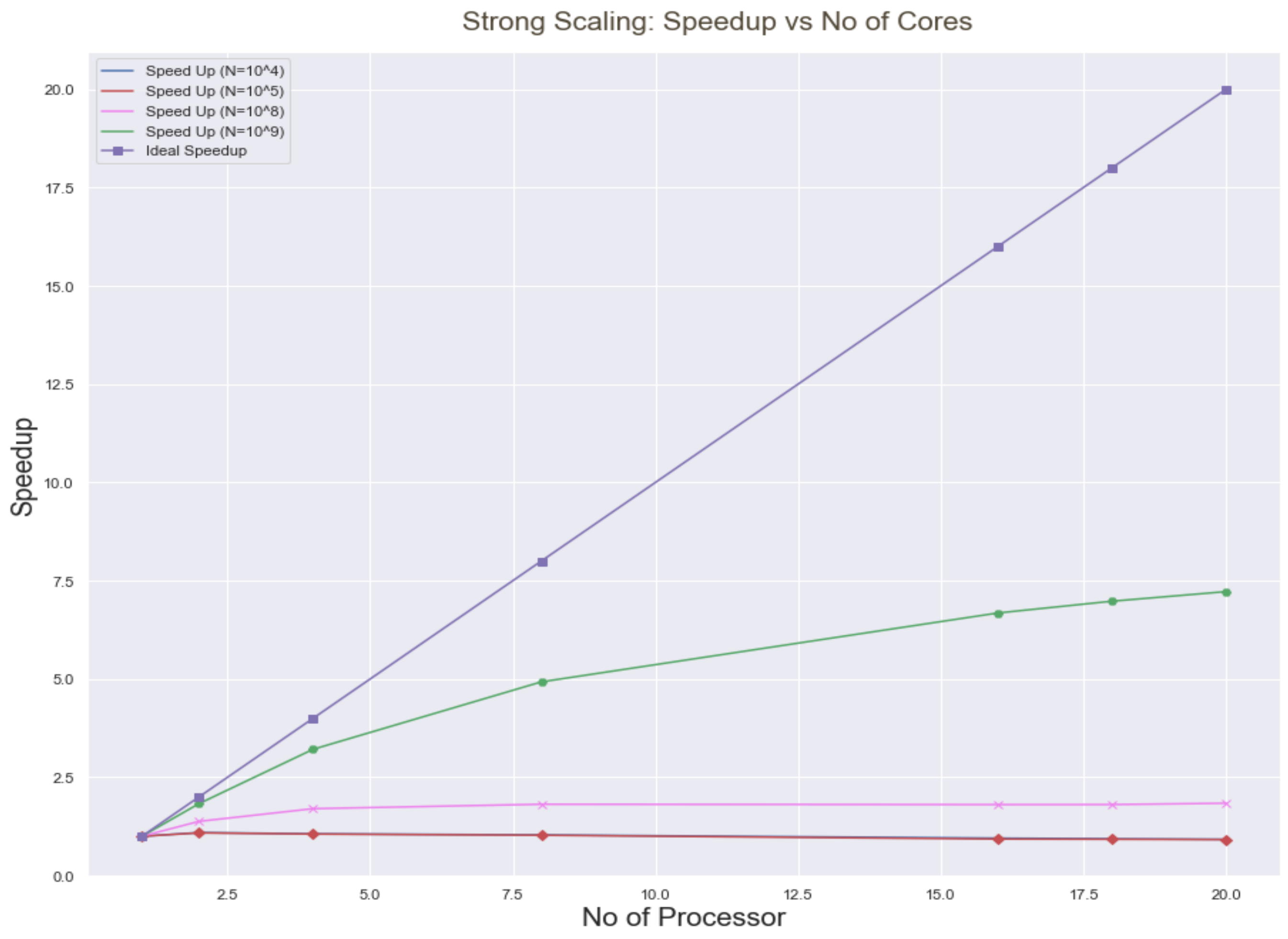


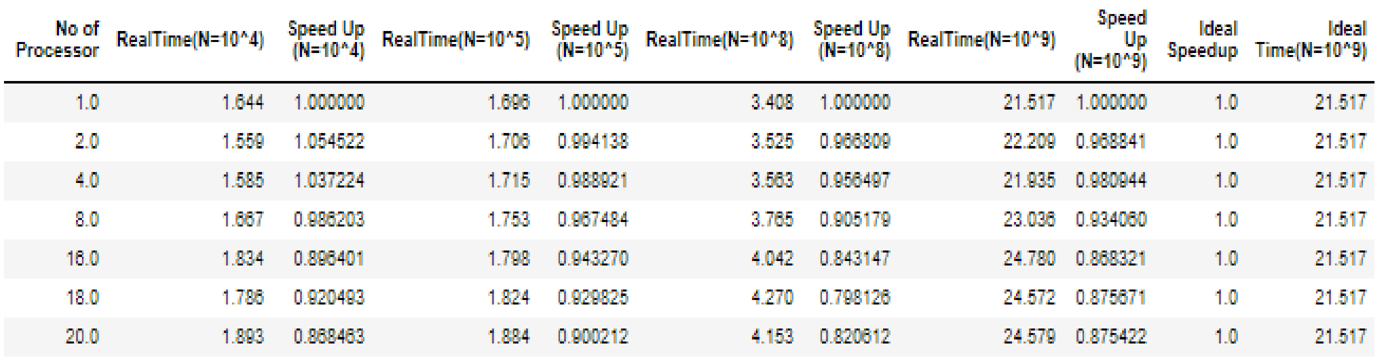
Fig1: Speedup measured against different problem sizes with range of processors.

Speedup was calculated based on real time (included overheads) to see how the performance of our application changed while there were larger core counts and in other words to estimate the number of processors appropriate for a particular problem size.

If we see in the figure, the green line is for problem size N= 10^9, for ***20*** *processors we have speed up only at* ***7.2*** *which indicates that the parallel efficiency is just at around* ***36%.*** *So we can say that we are not going to gain more efficiency if we run the problem with more than 20 processors. The tail off seen in the figure is caused by overheads and this graph holds true to Amdahl's law which states that the performance improvement to be gained by parallelisation is limited by the serial part of the code.*

Weak Scaling:

During the weak scaling test unlike in the case of strong scaling, we increased the problem size in proportion to the number of cores. The driving questions for this experiment were; whether the time will remain constant as we divide the task per core equally or will the time increases due to communication overheads? In this case our ideal time should remain constant for each problem size and the ideal speedup measure remains constant.



No of Processor	RealTime(N=10^4)	Speed Up (N=10^4)	RealTime(N=10^5)	Speed Up (N=10^5)	RealTime(N=10^8)	Speed Up (N=10^8)	RealTime(N=10^9)	Speed Up (N=10^9)	Ideal Speedup	Ideal Time(N=10^9)
1.0	1.644	1.000000	1.696	1.000000	3.408	1.000000	21.517	1.000000	1.0	21.517
2.0	1.559	1.054522	1.706	0.994138	3.525	0.966809	22.209	0.968841	1.0	21.517
4.0	1.585	1.037224	1.715	0.988921	3.563	0.956497	21.935	0.980944	1.0	21.517
8.0	1.667	0.986203	1.753	0.967484	3.765	0.905179	23.036	0.934060	1.0	21.517
16.0	1.834	0.896401	1.798	0.943270	4.042	0.843147	24.780	0.868321	1.0	21.517
18.0	1.786	0.920493	1.824	0.929825	4.270	0.798126	24.572	0.875671	1.0	21.517
20.0	1.893	0.868463	1.884	0.900212	4.153	0.820612	24.579	0.875422	1.0	21.517

Table 2: Weak Scaling Test data with measured time and speed up.

The weak scaling figure 2 below depicts that as the number of processors were increased, the speed up deviated from the ideal speed up scenario and this can be accounted by the communication overheads. However as the problem size was increased with proportion to number of processors, the speed up seemed to stabilize at certain value of speed up rather than dipping towards the axis which is contrary to Amdahl's law and this phenomena is captured by Gustafson's law. As we look at the wall time for each problem size, the time taken to complete the task has gradually increased rather than staying constant. The parallel efficiency can be calculated for the task size 10^9 to be at 0.875/20 =4.35%.

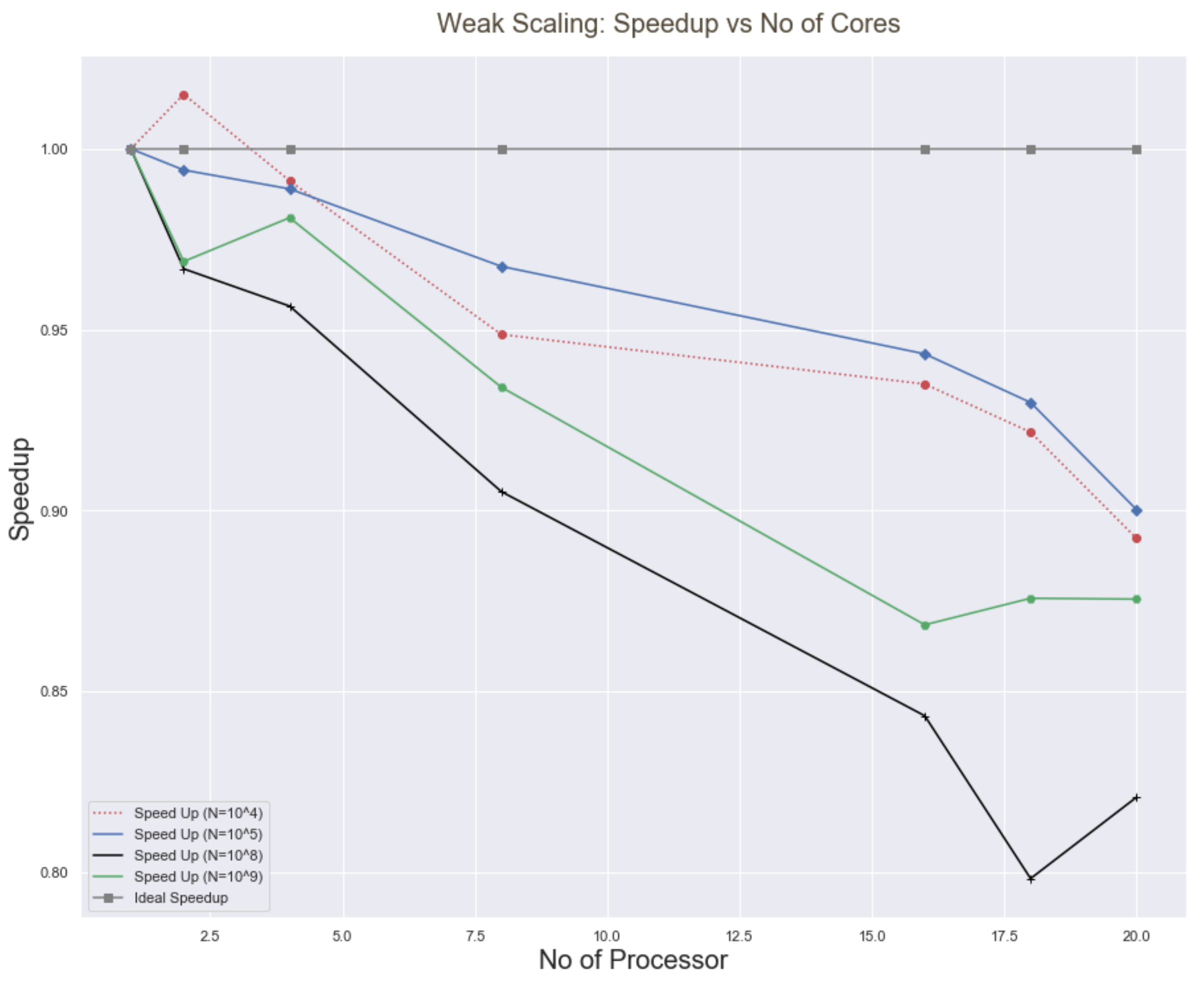


Fig 2: Weak scaling speed up measured with different problem sizes

Summary:

Deviations from the expected ideal speed up as seen in this exercise is normal in parallel computation therefore the number of processors chosen to run a task should be carefully considered to avoid the wastefulness of computing resources. While comparing the speedup between weak and strong scaling by picking up the problem size of N=10^9 for the given code, the strong scaling has better parallel efficiency i.e. 36% in comparison to the parallel efficiency of weak scaling which is at 4.35%.