Weak and Strong Scalability - A1

Marco Franzon

November, 2018

Abstract

In this exercise the aim is to investigate the effect of applying parallelism to an algorithm designed to calculate the value of π using a Monte Carlo approach. The algorithm is ran both in serial and in parallel and the weak and strong scalability is investigated.

1 Introduction

Scalability is defined as the capability of a system, network or process to handle a growing amount of work. In the field of high performance computing, we use two types of scalability to define the performance of a system on an application:

- Strong scalability refers how the performance of the system increases by increasing the number of processors and keeping the problem size fixed.
- Weak scalability refers how the performance of the system remains constant by increasing both the number of processors and the problem size.

2 Strong scalability

Strong scalability refers to the behavior the time solution of the algorithm takes when the problem size is kept constant but the number of processing cores is increased. The best way to measure this scalability is to measure the speedup due to the increasing number of processors used, *Speedup* defined as:

$$Speedup = \frac{S}{P} = \frac{t_{serial}}{t_{parallel}} \tag{1}$$

where $t_{parallel}$ and t_{serial} are the walltime of the parallel and serial algorithm respectively.

If the amount of work is perfectly divided between all the processors, the problem should be perfectly scalable and the speed up should be a straight line, parallel to the bisector with to respect to the number of processors.

Figure 1 shows the serial to parallel runtime ratio (speedup) as a function of the number of cores used. In Figure 2 is shown the efficiency of strong scaling.

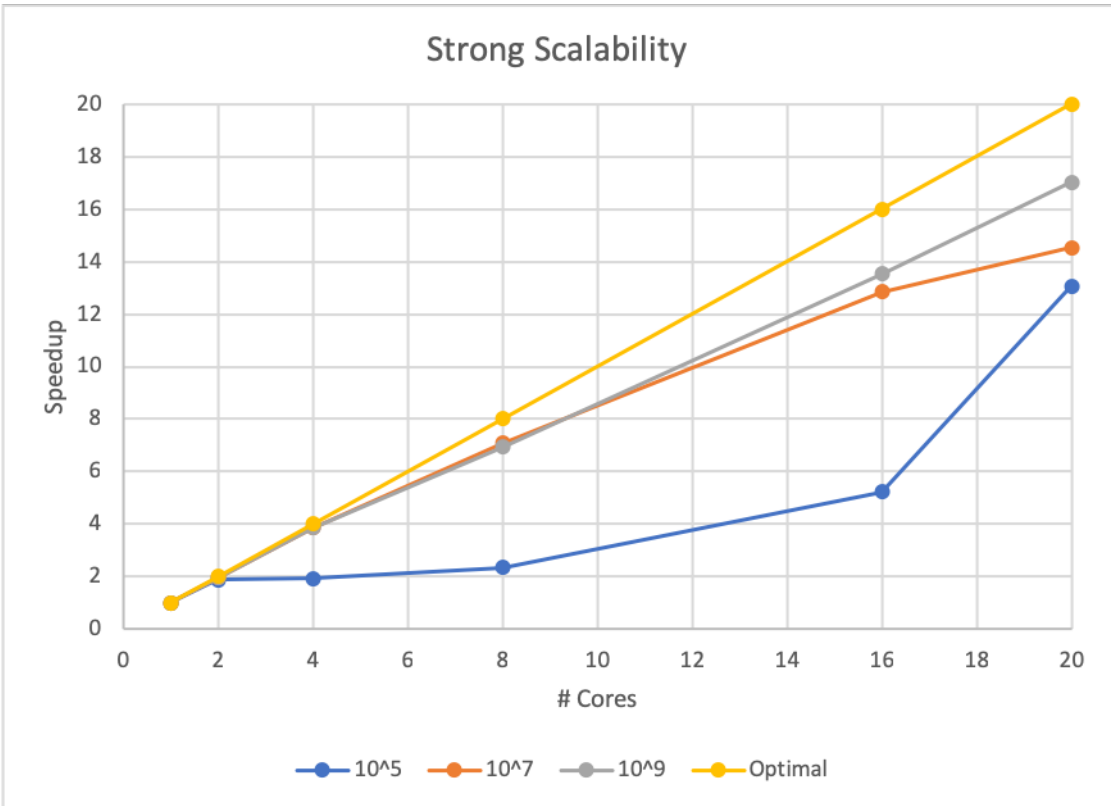

Figure 1: Strong Scalability: Speedup to number of cores

3 Weak scalability

To test the weak scaling we increased the amount of montecarlo steps accordly to the number of processor used. In the ideal case the time elapsed during the process should remain constant. The algorithm has been run for 1, 2, 4, 8, 16, 20 cores and by definition of weak scalability N has been proportionally varied from N = 1 $\times 10^i$ to N = 20 $\times 10^i$ where i = 5,7,9. Figure 3 shows the efficiency of weak scalability as function of numbers of cores.

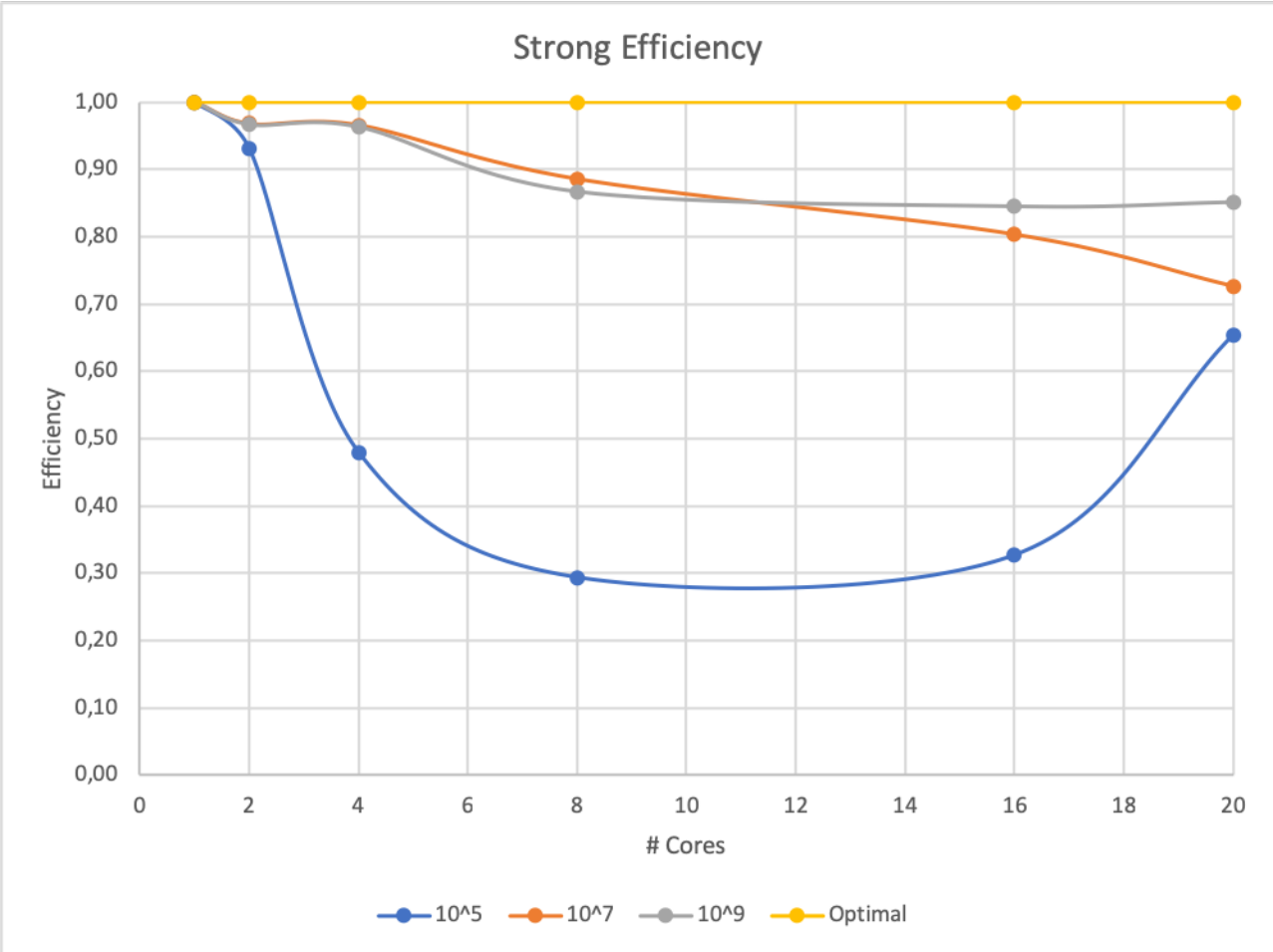

Figure 2: Efficiency Strong Scalability: Efficiency to number of cores

4 Discussion and Conclusions

Theoretically, perfect strong scalability would yield to a straight line of slope 1 in Figure 1. The performance is reduced because of the time and resources used to make the different cores communicate between them. This could be appreciate in Figure 2. A similar effect shows up in Figure 3. Theoretically, in an embarrassingly parallel algorithm if the workload is evenly distributed between all cores and there is no time loss due to communication between them, then the serial to parallel ratio would remain constants. Since the code is itself not embarrassingly parallel and there exists latency and transfer time the ratio decreases to approximately 0.8 as the communication between cores increases as their number increases.

Figure 3: Efficiency Weak Scalability: Efficiency to number of cores