Computing weak and strong scalability of an HPC application

Francesco Cicala

Assignment 1

Abstract

In the following report we describe the results of a study of scalability on a given script. It performed an evaluation of the pi number by means of a Monte Carlo method: points are incrementally added with uniform distribution on a square, and it is measured in which proportion they fall inside the circle inscribed within the square. We show how the simulation performs under weak and strong scaling conditions by computing the walltime, the speed-up and the efficiency for a variable number of processors and points extractions.

1. Introduction

This analysis aims to reproduce the conditions for weak and strong scaling by the use of OpenMP, which is a set of compiler directives and it represents the shared-memory programming standard. The scalability of a system refers to its ability to cope and perform well when its resources or the workload are increased, i.e. how well a task can be parallelized on a given system, according to a scalability metric.

We make the following simple (and quite strong) assumption, that the overall problem size can be written as

$$s + p = 1$$

where s is the nonparallelizable part and p is the perfectly parallelizable fraction of the amount of work.

1.1. Strong scaling

Since the normalized time for a fixed size problem to be solved is

$$T_f^s = s + p$$

solving the problem on N processors will require a runtime of

$$T_f^p = s + \frac{p}{N}$$

This setting defines what is called strong scaling.

1.2. Weak scaling

If our goal is to scale the problem size with the increasing number of processors, under the assumption that the serial fraction s is a constant, the serial runtime for a scaled problem is:

$$T_v^s = s + pN$$

and the parallel runtime is

$$T_v^p = s + p$$

1.3. Scalability metrics

A common scalability metric is the speed-up, which is defined as the quotient of parallel and serial performance for fixed problem size. For our purposes, the computing time is an adequate performance value. Therefore, we define the speedup as:

$$S_f = \frac{T_f^p}{T_f^s} = \frac{1}{s + \frac{1-s}{N}}$$

This result is also called "Amdahl's Law".

The last scalability metric we consider is directly derived from the previous one. It is called efficiency and it is expressed as

$$\epsilon = \frac{\text{speedup}}{N}$$

2. The experiment settings

All the executions have been performed on the SISSA Ulysses cluster. Either for weak and strong scalability, we have repeated the simulation over a variable number of processors going from 1 to 20. Moreover, the weak and strong scaling experiments have been repeated for three different problem sizes:

1. The weak scaling tests have been run for $N = 10^5$, 10^6, 10^7
2. The strong scaling tests have been run for $N = 10^6$, 10^7, 10^8

We compiled the mpi.c script with the mpicc instruction, and then we ran it by means of the mpirun command. For each execution, both for the strong and weak scaling tests, we saved the walltime, computed inside the .c script. This choice is motivated by the fact that we acted within the assumption that, as introduced in the previous section, the serial workload remains constant throughout the experiment. Therefore, we assumed that the walltime was an adequate choice in this perspective.

3. Results

Below, the quantitative results of the experiment are shown.

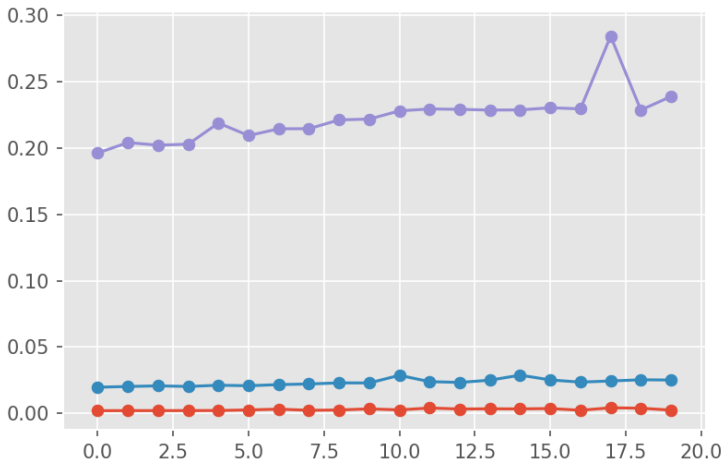

Figure 1: Weak scalability: time vs processors number

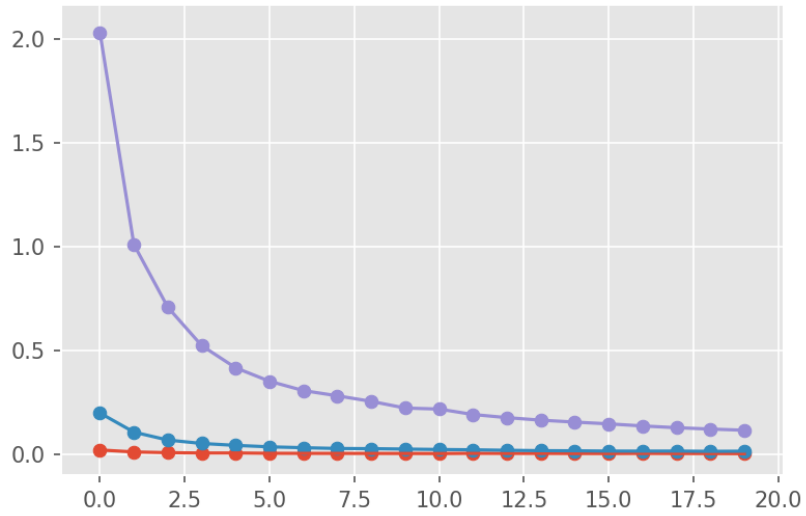

Figure 2: Strong scalability: time vs processors number

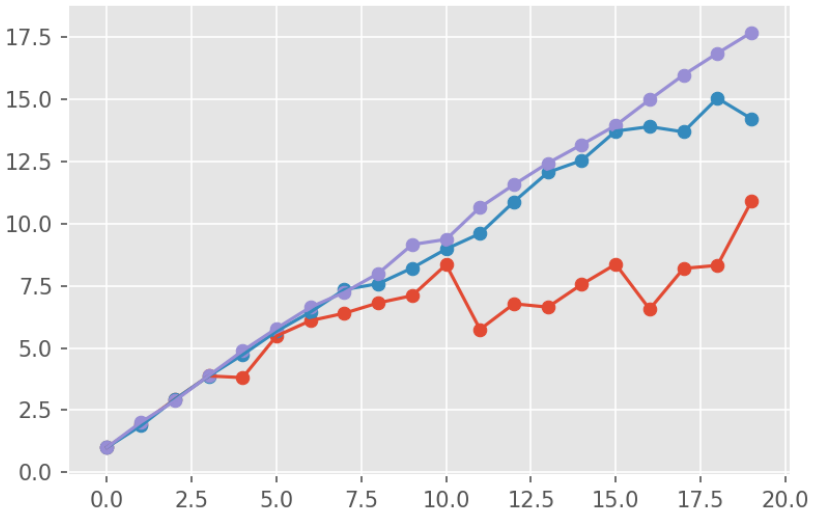

Figure 3: Strong scalability: speed-up vs processors number

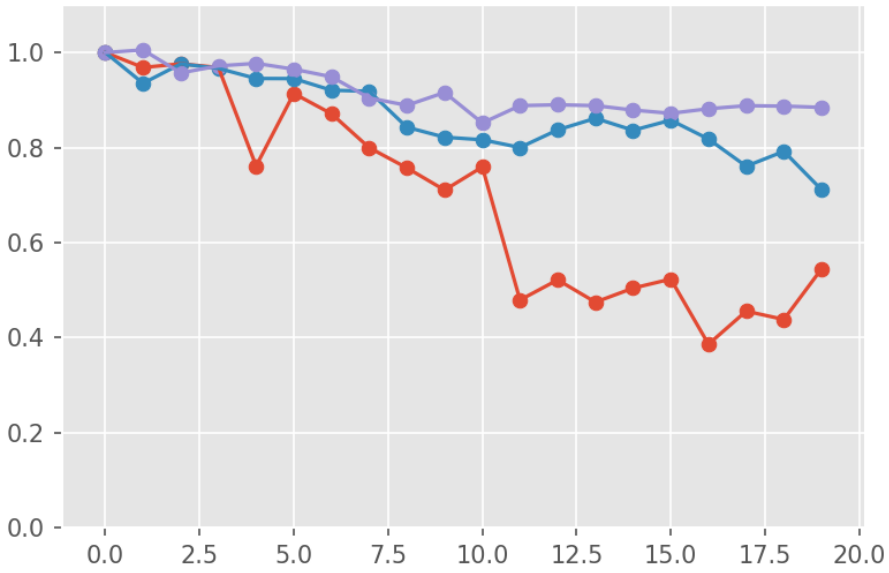

Figure 4: Strong scalability: efficiency vs processors number

4. Conclusions

The results do agree with good approximation with the predictions over the assumptions we described in the introduction. We can infer that, within the range given by the number of processors we used and the problem sizes we tested, the problem scales well both in weak and strong settings. For what concern the latter, we observe that the relative fluctuations seems to become less relevant by increasing the size of the problem.

References

Hager, G., Wellein, G. (2011). Introduction to High Performance Computing for Scientists and Engineers. CRC Press