Giulia Franco
matricola SM3500370
Year 2018/2019
Exercise 1, High Performance Computing course.

Weak and Strong Scalability.

The assay consists in compute a code in serial and parallel and checking the computational time. The code compute Pi with a Hit-or-Miss algorithm. With the serial computation we have an estimated Pi=3.13460 for N=100000 iterations.
The next step is running the code in parallel (on Ulysses). We may first start studying strong scalability. This means that the number of iteration N is constant and kept on 100000, and the number of nodes is a variable (2,4,8,16,32). As reported in the following graphic, the computational time decrease as the number of nodes increase. This is expected: the execution time, measured in seconds, to solve the problem for a fixed size decrease as the computation is splitted between an increasing number of nodes.

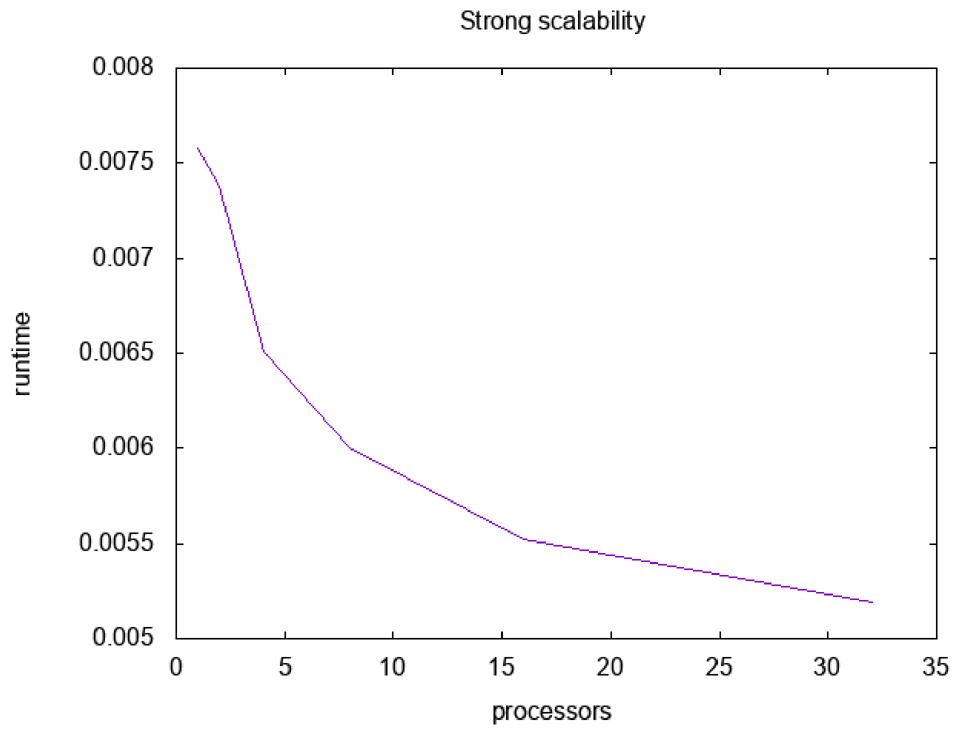

Figura 1: Strong Scalability for N=100000.

It's also possible to compute weak scalability by increasing the number of nodes proportionally to the problem size, so the time is expected to be constant. The idea is to do more tasks of fixed size in the same length of *wall time*, rather than a fixed workload in less time. The result is reported in the following graphic.

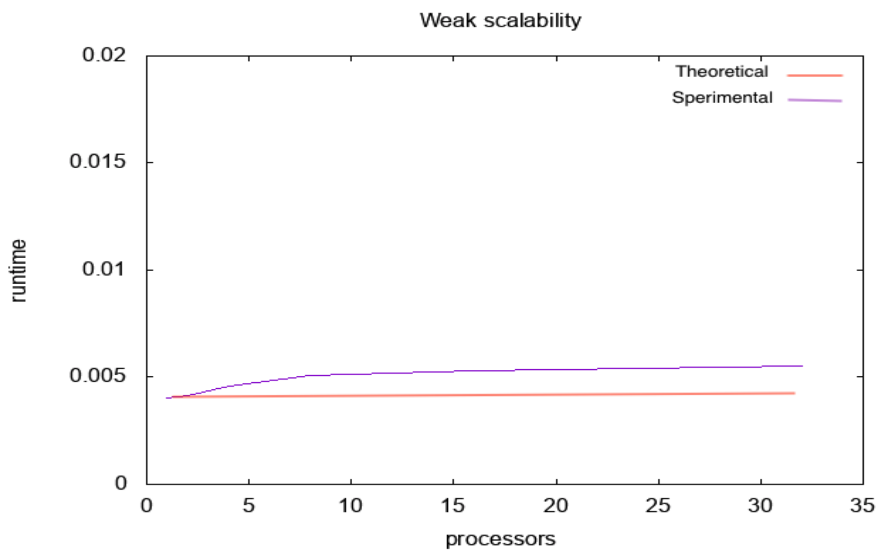

Figura 2: Weak Scalability.

As Fig.2 shows, the experimental data (purple line) don't really respect the theoretical expectation (red line). This might be caused by the small size of the problem (major N=320000 for 32 parallel process). Since the error is acceptable, this data are acceptable (the computed error is under 5 percent confidence level.)
The Speedup S is also computable, and it's defined as:

$$S(p) = \frac{T_1(N)}{T_p(N)} \tag{1}$$

for p=(1,2,4,8,16,32) parallel processes and fixed problem size N=100000. The result can be observed in Fig.3.

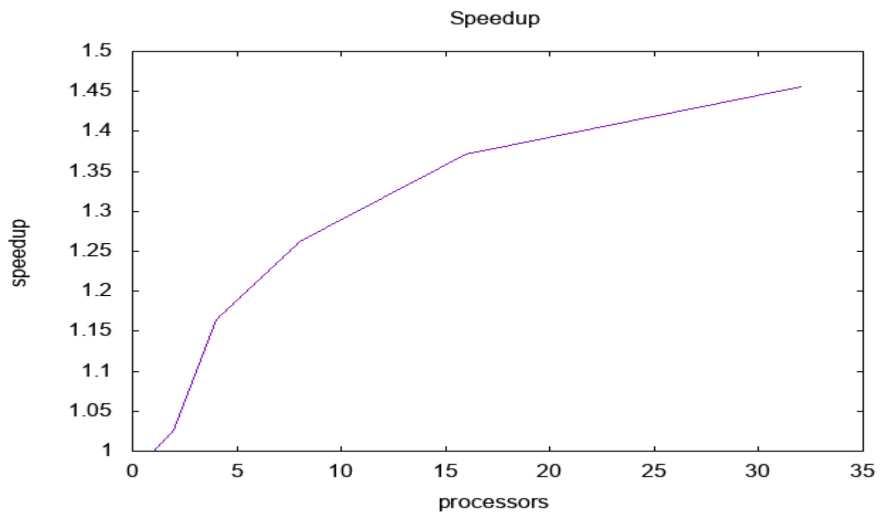

Figura 3: Speedup.

Fig. 3 shows that Speedup, even if clearly not linear scaling, is not yet asymptotic to a constant value as expected: this can't be achieved due to Amdahl's Law which states that there's no speedup for serial parts of the code. For large N, the parallel speedup doesn't asymptote to N, but to a constant $1/a$, where a is the serial fraction of the work.
Since the perfect speedup isn't achievable due to the serial part of the code, it's reasonable to study how far the results are from the form $S(p) = p$. This mean computing the efficienty, which is defined as:

$$Efficienty = S(p)/p \tag{2}$$

for p processors.
In particular, while for strong scalability the efficienty is defined ad in (2), for weak scalability is computed as the Speedup in (1).

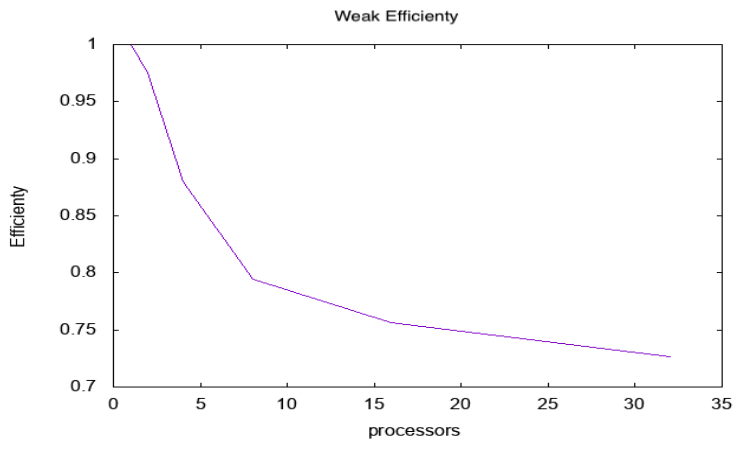

Figura 4: Weak Efficienty.

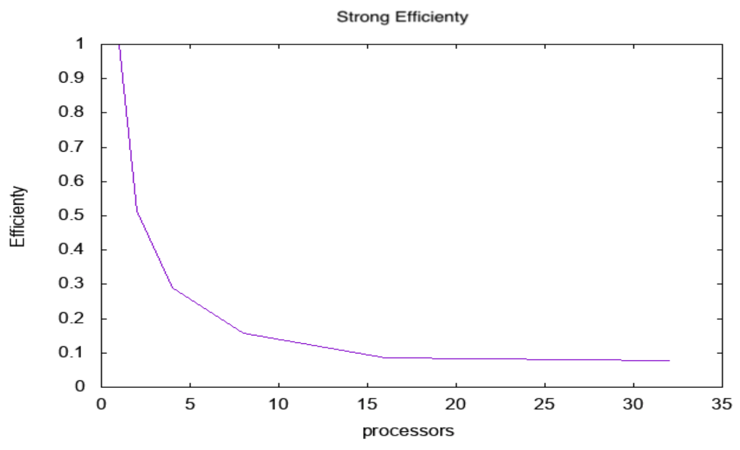

Figura 5: Strong Efficienty.

The formers graphs show that efficienty takes higher values, on average, using weak scalability model (Fig. 4) than the strong one (Fig. 5). So it's possible to conclude from the analysis and the Gustafson's Law, which states that the *scaled speedup* is linear in the problem size, that the weak scalability is the one that suits better the problem.