Unit 2.1: Understanding Parallelism

Video lesson 3: Speed–up and efficiency

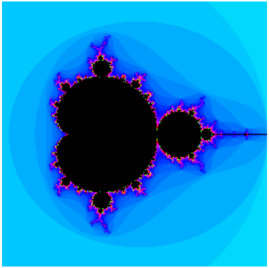
Eduard Ayguadé, Josep Ramon Herrero,
Daniel Jiménez and Gladys Utrera

Barcelona Supercomputing Center
Universitat Politècnica de Catalunya

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

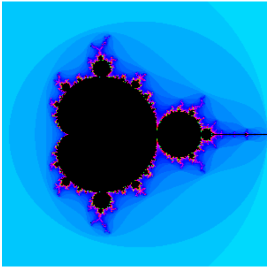
Barcelona Supercomputing Center Centro Nacional de Supercomputación

Motivation: Mandelbrot set



- The Mandelbrot set is the set of complex numbers p in a delimited two-dimensional space for which the sequence $z_{n+1} = z_n^2 + p$ (starting with $z_0 = 0$):
$$p,\ p^2 + p,\ (p^2 + p)^2 + p,\ ((p^2 + p)^2 + p)^2 + p,\ \ldots$$
fulfils $|z_\infty| < 2$

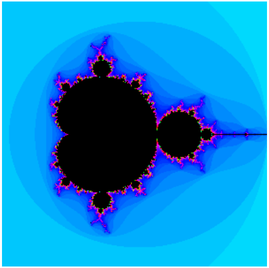
Motivation: Mandelbrot set



```
n = 0; z.real = z.imag = 0;
do  {
    temp = z.real*z.real - z.imag*z.imag + p.real;
    z.imag = 2*z.real*z.imag + p.imag;
    z.real = temp;
    norm_sq = z.real*z.real + z.imag*z.imag;
} while (norm_sq < (2*2) && ++n < max);
```

limiting the exploration of the sequence up to a maximum number of steps ($n < max$)

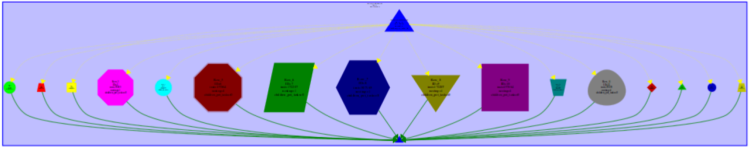
Motivation: Mandelbrot set



- The plot of the Mandelbrot set is created by coloring each point p in the complex plane according to the number of steps n (dark in the plot above if ($n = max$)

- Assume a task corresponds with the computation of the previous recurrence for a set of consecutive rows of the two-dimensional space
- Embarrassingly parallel decomposition of the problem in tasks

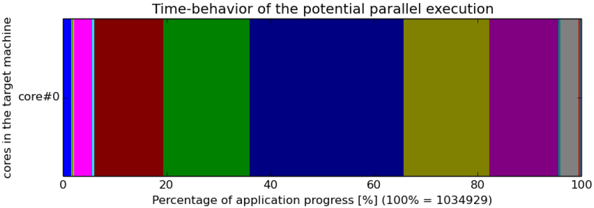
...



... but heavily unbalanced in terms of computational load

Motivation: Mandelbrot set with Tareador

- If we execute the tasks generated in a machine with a single processor ...



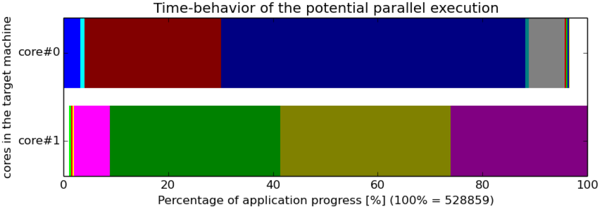
the execution time is $T_1 = 1034929$, which we will take as reference time for the sequential execution

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Barcelona Supercomputing Center Centro Nacional de Supercomputación

Motivation: Mandelbrot set with Tareador

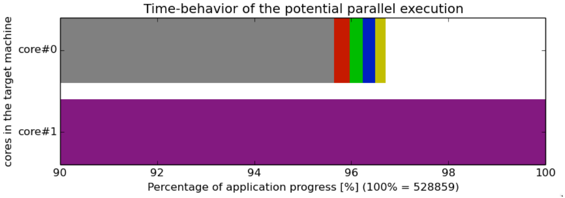
- What if we execute with 2 processors?



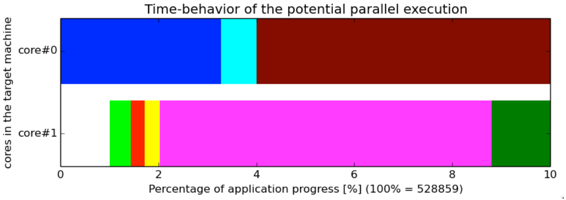
resulting in an execution time of $T_2 = 528859$ time units, 1.95 times faster than the sequential execution

Motivation: Mandelbrot set with Tareador

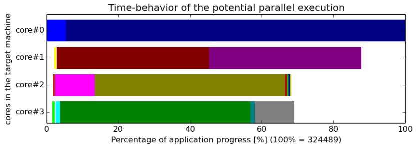
- Load unbalance



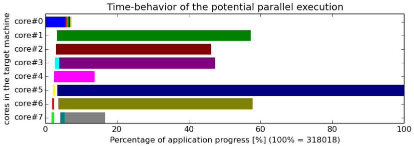
- Task creation overhead



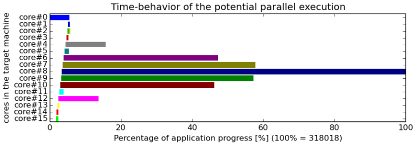
Motivation: Mandelbrot set with Tareador



$T_4 = 324489$, 3.18 times faster



$T_8 = 318018$, 3.25 times faster

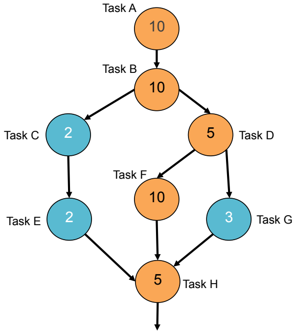


$T_{16} = 318018$, 3.25 times faster

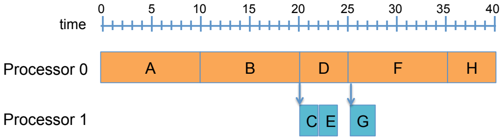
UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Barcelona Supercomputing Center Centro Nacional de Supercomputación

Execution time bounds on P processors



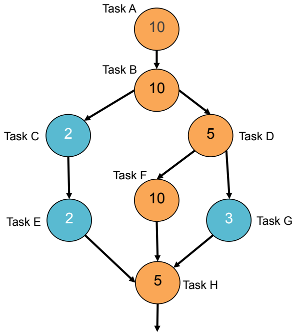
- T_p = execution time on P processors
- **Task scheduling**: how are tasks assigned to processors? For example:



- Lower bounds
  - $T_p \geq T_1/P$
  - $T_p \geq T_\infty$

Speed–up

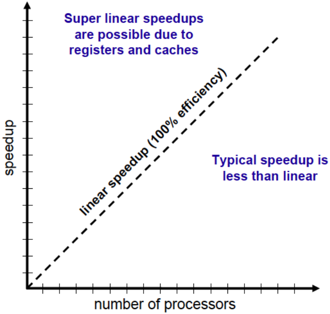
Speedup S_p: relative reduction of the sequential execution time when using P processors



- $S_p = T_1 \div T_p$
- In this example:
  $T_2 = 40, \ S_2 = 47/40 = 1.175$

Scalability and efficiency

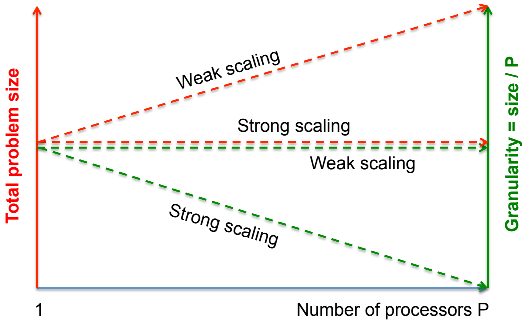
- Scalability: how the speed-up evolves when the number of processors is increased
- Efficiency: $E_p = S_p \div P$



Strong vs. weak scalability

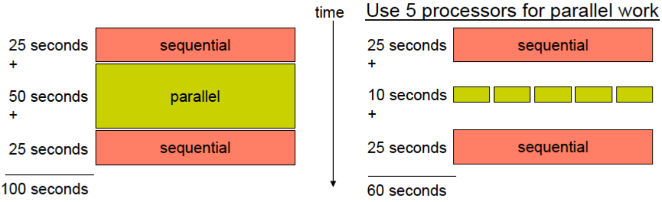
Two usual scenarios to evaluate the scalability of one application:

- Increase the number of processors P with constant problem size (strong scaling $\rightarrow$ reduce the execution time)
- Increase the number of processors P with problem size proportional to P (weak scaling $\rightarrow$ solve larger problem)



Amdahl's law

Performance improvement is limited by the fraction of time the program does not run in fully parallel mode



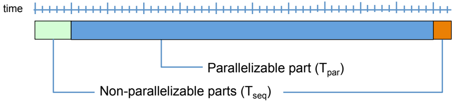
- Parallel part is 5 times faster: $Speedup_{parallel\_part} = 50/10 = 5$
- Parallel version is just 1.67 times faster: $S_p = 100/60 = 1.67$, $E_p = 1.67/5 = 0.33$

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Barcelona Supercomputing Center Centro Nacional de Supercomputación

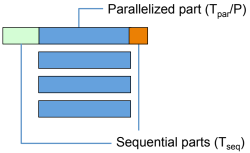
Amdahl's law

Assume the following simplified case, where the parallel fraction φ is the fraction, of total execution time, the program can be parallelized



$$T_1 = T_{seq} + T_{par}$$
$$\varphi = T_{par}/T_1$$
$$T_{seq} = (1-\varphi) \times T_1$$
$$T_{par} = \varphi \times T_1$$

$$T_1 = (1-\varphi) \times T_1 + \varphi \times T_1$$



$$T_P = T_{seq} + T_{par}/P$$
$$T_P = (1-\varphi) \times T_1 + (\varphi \times T_1/P)$$

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Barcelona Supercomputing Center Centro Nacional de Supercomputación

Amdahl's law

From where we can compute the speed–up S_P that can be achieved as

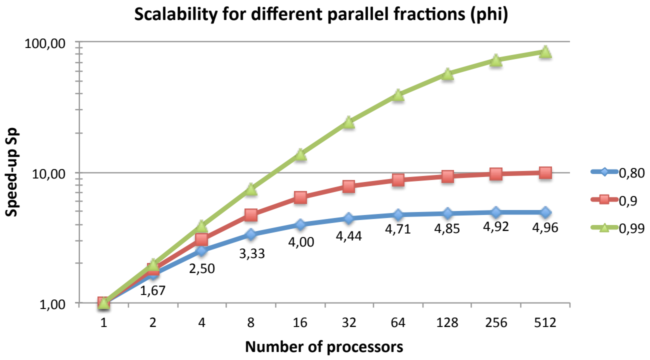
$$S_p = \frac{T_1}{T_p} = \frac{T_1}{(1-\varphi) \times T_1 + (\varphi \times T_1/P)}$$

$$S_p = \frac{1}{((1-\varphi) + \varphi/P)}$$

Two particular cases:

$$\varphi = 0 \rightarrow S_p = 1$$
$$\varphi = 1 \rightarrow S_p = P$$

Amdahl's law



When $P \to \infty$ the expression of the speed–up becomes

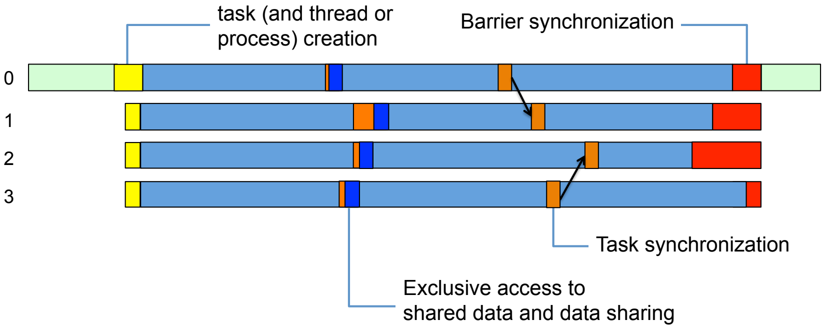
$$S_{p\to\infty} = \frac{1}{(1-\varphi)}$$

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

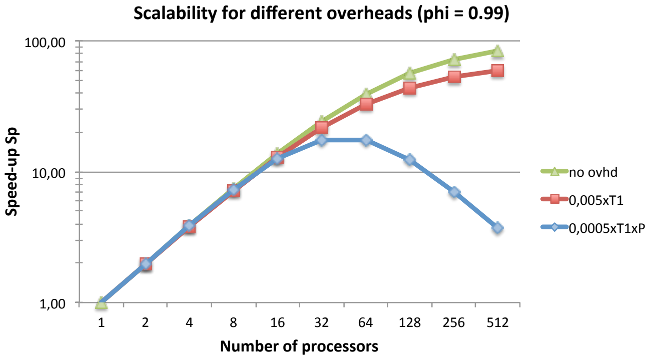
Sources of overhead

Parallel computing is not for free, we should account overheads (i.e. any cost that gets added to a sequential computation so as to enable it to run in parallel)



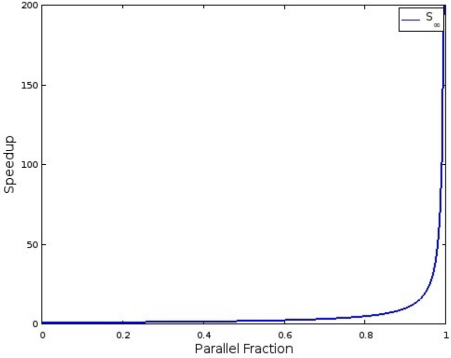
Amdahl's law (with constant and linear overheads)

$$T_p = (1 - \varphi) \times T_1 + \varphi \times T_1/p + overhead(p)$$



Conclusions of Amdahl's Law

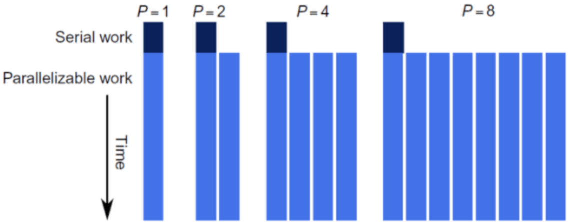
Amdahl's Law can be overly pessimistic:



- Parallel processing might not be worthwhile if there is a large amount of inherently sequential code.

However, often in practice . . .

- The goal of applying parallelism is to increase the accuracy of the solution that can be computed in a fixed amount of time.
  $\rightarrow$ Treat time as constant and let problem size increase with P.
- The serial part grows slowly or remains fixed
  $\rightarrow$ It's proportion gets reduced as the problem size increases.



- Speedup grows as workers are added and the problem size is increased. (Weak Scaling).

Unit 2.1: Understanding Parallelism

Video lesson 3: Speed–up and efficiency

Eduard Ayguadé, Josep Ramon Herrero,
Daniel Jiménez and Gladys Utrera

Barcelona Supercomputing Center
Universitat Politècnica de Catalunya