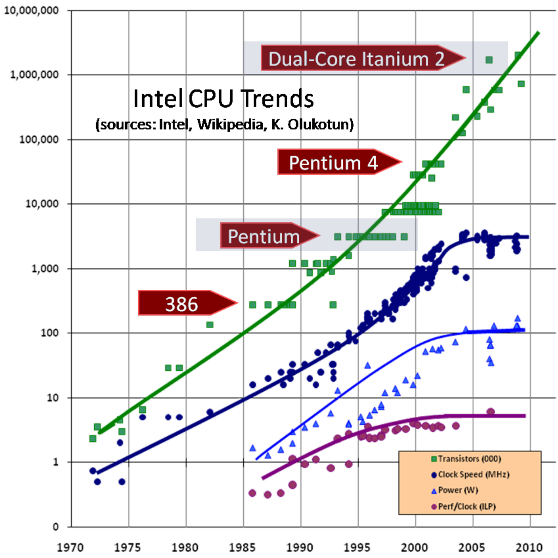
Some Parallel Computing Concepts

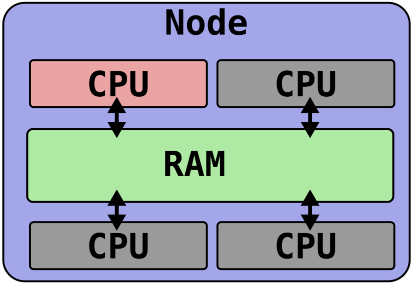
jean-luc.falcone@unige.ch

July, 2018

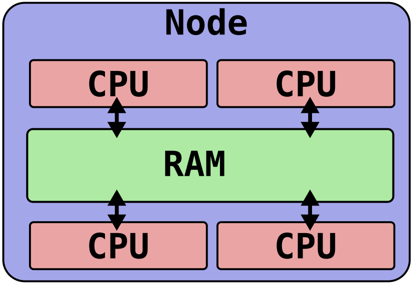
Free Lunch is over



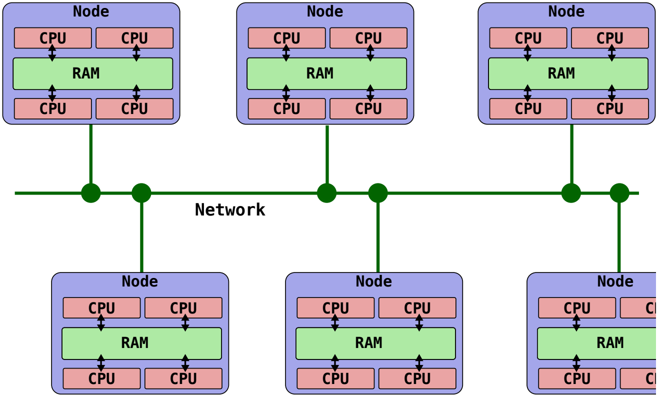
Single core



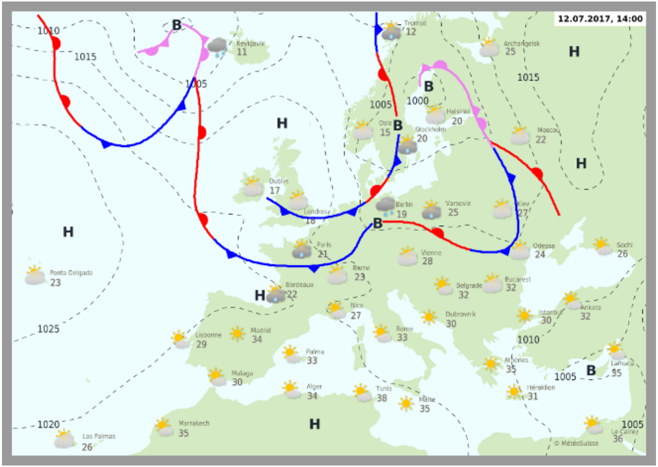
Shared memory (multithread, openmp)



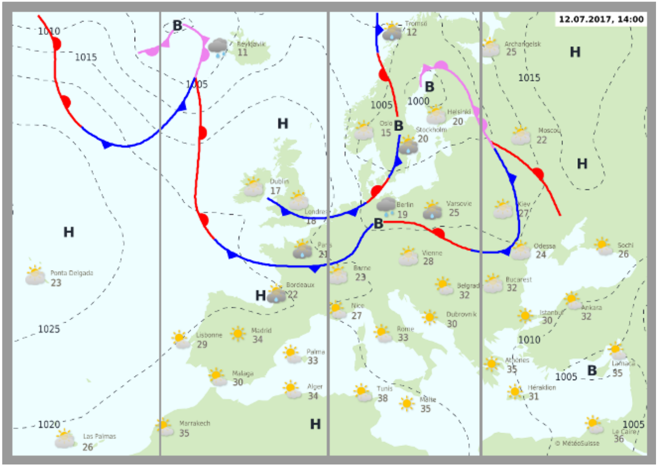
Distributed memory (cluster, mpi)



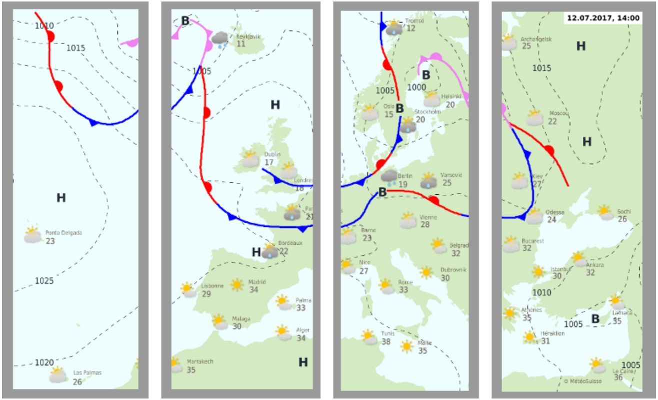
Sequential


12.07.2017, 14:00
1010
1015
1005
1000
1025
1020
Reykjavik 11
Tromsø 12
Archangelsk 25
Helsinki 20
Oslo 15
Stockholm 20
Moscou 22
Dublin 17
Londres 18
Berlin 19
Varsovie 25
Kiev 27
Paris 21
Vienne 28
Odessa 24
Sochi 26
Berne 23
Bordeaux 22
Belgrade 32
Bucarest 32
Ponta Delgada 23
Nice 27
Dubrovnik 30
Istanbul 30
Ankara 32
Lisbonne 29
Madrid 34
Rome 33
Palma 33
Athènes 35
Malaga 30
Alger 34
Tunis 38
Malte 35
Héraklion 31
Larnaca 35
Marrakech 35
Las Palmas 26
Le Caire 36
© MétéoSuisse

Parallel: Shared Memory



Parallel: Distributed Memory



Single Worker vs Multiple Worker

- Suppose 1 worker can dig a hole in 2 hours
- How much does it take for 2 workers to dig the same hole ?
- Can 60 workers dig the same hole in 2 minutes ?

The Speed-Up

We define the speed-up with p workers as:

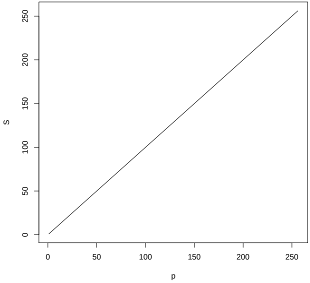
$$S(p) = \frac{T_1}{T_p} \tag{1}$$

where T_i is the time required for i workers.

Ideal Speed-Up

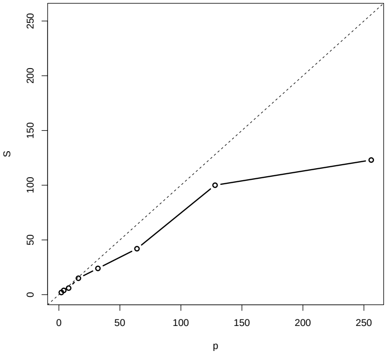
Ideally, p workers should be able to complete a task p times faster than one single worker:

$$T_p = \frac{T_1}{p} \quad \rightarrow \quad S(p) = \frac{T_1}{T_p} = p \tag{2}$$



Measured Speed Up

But of course the reality can be far from ideal. . .



Absolute vs. Relative Speed-Up

Attention

When applied to computer programs. We will take the best sequential implementation (T_s) instead of a parallel implementation with just one processor (T_1):

$$S(p) = \frac{T_s}{T_p} < \frac{T_1}{T_p} \tag{3}$$

Better speed-up: larger domain

Embarrassingly parallel

- Suppose 1 worker can dig a hole in 2 hours
- Then, 60 workers can dig 60 holes in 2 hours
- In average, that's 1 hole per 2 minutes !

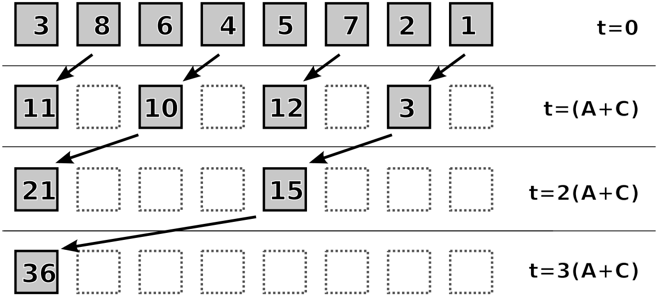
Definition

Suppose we want to sum an large array of n numbers in parallel. Sequentially, it will take:

$$T_s = (n-1) \cdot A \tag{4}$$

where n is the array size and a the constant duration of a single addition. (approximation).

First Parallel implementation (1)



First Parallel implementation (2)

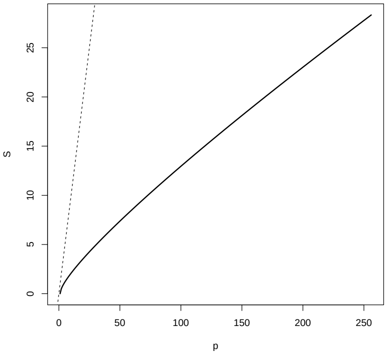
Suppose we have 1 number per processor ($p = n$), then the parallel time will be:

$$T_p = (A + C) \cdot \log_2(n) \tag{5}$$

where $A + C$ is sum of the duration of a single addition, plus a single communication time.

First Parallel Implementation Speed-Up

$$S(p) = \frac{T_s}{T_p} = \frac{(p-1)A}{(A+C)\cdot \log_2(p)} = \frac{p-1}{\left(1+\frac{C}{A}\right)\log_2 p} \qquad (6)$$



Better parallel implementation

Suppose now we have less processors than numbers to sum ($p << n$), then the parallel time will be:

$$T_p = \left(\frac{n}{p} - 1\right) A + (C + A)\log_2(p) \tag{7}$$

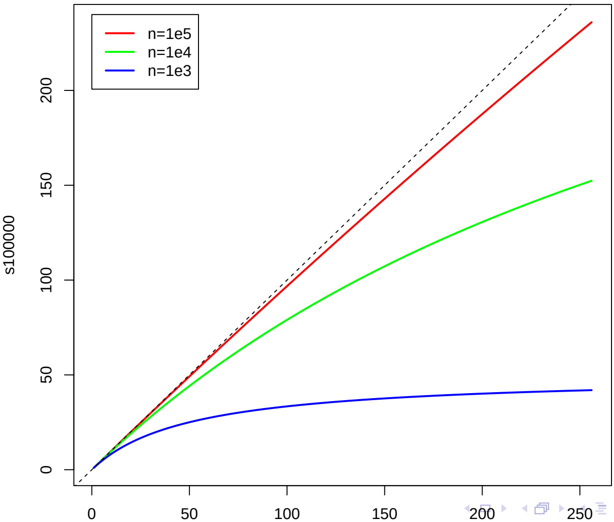
Better implementation Speed-up (1)

$$S(p) = \frac{p}{\frac{n-p}{n-1} + \left(1 + \frac{C}{A}\right) \frac{p}{n-1} \log_2(p)} \tag{8}$$

Since n is very large $\frac{n-p}{n-1} \approx 1$ and $\frac{p}{n-1} \approx \frac{p}{n}$. Therefore:

$$S(p) \approx \frac{p}{1 + \left(1 + \frac{C}{A}\right) \frac{p}{n} \log_2(p)} \tag{9}$$

Better implementation Speed-up (2)



Problem definition

A given parallel problem has sequential parts. For instance:

- A single processor reads a configuration file and makes data available to other processors.
- A single processor must collect all data to write results on the disk.
- Sequential post/pre processing

Amdahl's Law derivation (1)

Let W be the total amount of work and R the "computing power" of a single processor. The sequential time will be:

$$T_s = \frac{W}{R} \tag{10}$$

Let α be the fraction of the parallel implementation running sequentially:

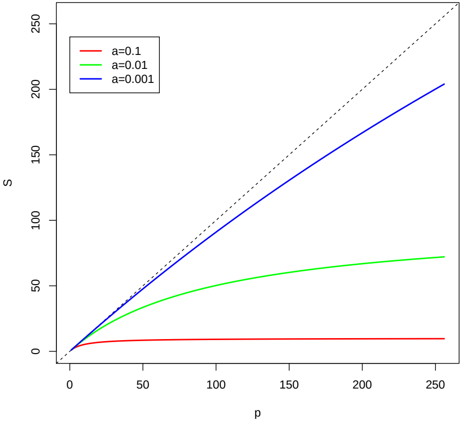
$$T_p = \frac{\alpha W}{R} + \frac{(1-\alpha)W}{pR} \tag{11}$$

Amdahl's Law derivation (2)

Therefore the speed-up is:

$$S(p) = \frac{T_s}{T_p} = \frac{\frac{W}{R}}{\frac{\alpha W}{R} + \frac{(1-\alpha)W}{pR}} = \frac{1}{\alpha + \frac{(1-\alpha)}{p}} \leq \frac{1}{\alpha} \qquad (12)$$

Amdahl's Law



Problem definition

Suppose now that the amount of work varies with the number of available processors. Remember that $T = \frac{W}{R}$:

$$W_p = \beta T_p R + p(1 - \beta) T_p R \tag{13}$$

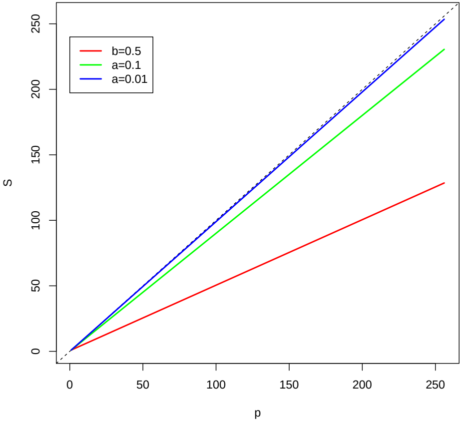
where βT_p is the amount of time spent in the sequential part of the code.

Law derivation

Since $T_s = \frac{W}{R}$ the speed-up is now:

$$S(p) = \frac{T_s}{T_p} = \beta + p(1-\beta) = O(p) \tag{14}$$

Gustafson's Law



Efficiency

We define the parallel efficiency with p workers as:

$$E(p) = S(p)/p = \frac{T_1}{pT_p} \tag{15}$$

Efficiency: Amdahl's Law