Performance Metrics

Ricardo Rocha and Fernando Silva

Computer Science Department

Faculty of Sciences

University of Porto

Parallel Computing 2015/2016

Performance

Two key goals to be achieved with the design of parallel applications are:

- **Performance** – the capacity to reduce the time needed to solve a problem as the computing resources increase
- **Scalability** – the capacity to increase performance as the size of the problem increases

The main factors limiting the performance and the scalability of an application can be divided into:

- **Architectural limitations**
- **Algorithmic limitations**

Factors Limiting Performance

Architectural limitations:

- **Latency and bandwidth**
- **Data coherency**
- **Memory capacity**

Algorithmic limitations:

- **Missing parallelism** (sequential code)
- **Communication frequency**
- **Synchronization frequency**
- **Poor scheduling** (task granularity/load balancing)

Performance Metrics

There are 2 distinct classes of performance metrics:

- **Performance metrics for processors/cores** – assess the performance of a processing unit, normally done by measuring the speed or the number of operations that it does in a certain period of time
- **Performance metrics for parallel applications** – assess the performance of a parallel application, normally done by comparing the execution time with multiple processing units against the execution time with just one processing unit

Here, we are mostly interested in metrics that measure the performance of parallel applications.

Performance Metrics for Processors/Cores

Some of the best known metrics are:

- ***MIPS*** – *Millions of Instructions Per Second*
- ***MFLOPS*** – Millions of *FLoating point Operations Per Second*
- ***SPECint*** – SPEC (*Standard Performance Evaluation Corporation*) *benchmarks that evaluate processor performance on integer arithmetic* (first release in 1992)
- ***SPECfp*** – SPEC benchmarks that evaluate processor performance on floating point operations (first release in 1989)
- ***Whetstone*** – synthetic benchmarks to assess processor performance on floating point operations (first release in 1972)
- ***Dhrystone*** – synthetic benchmarks to assess processor performance on integer arithmetic (first release in 1984)

Performance Metrics for Parallel Applications

Some of the best known metrics are:

- ***Speedup***
- **Efficiency**
- **Redundancy**
- **Utilization**

There also some laws/metrics that try to explain and assert the potential performance of a parallel application. The best known are:

- **Amdahl's law**
- **Gustafson-Barsis' law**
- **Karp-Flatt metric**
- **Isoefficiency metric**

Speedup

Speedup is a measure of performance. It measures the ratio between the sequential execution time and the parallel execution time.

$$S(p) = \frac{T(1)}{T(p)}$$

T(1) is the execution time with one processing unit

T(p) is the execution time with ***p*** processing units

	1 CPU	2 CPUs	4 CPUs	8 CPUs	16 CPUs
T(p)	1000	520	280	160	100
S(p)	1	1.92	3.57	6.25	10.00

Efficiency

Efficiency is a measure of the usage of the computational capacity. It measures the ratio between performance and the number of resources available to achieve that performance.

$$E(p) = \frac{S(p)}{p} = \frac{T(1)}{p \times T(p)}$$

S(p) is the speedup for ***p*** processing units

	1 CPU	2 CPUs	4 CPUs	8 CPUs	16 CPUs
S(p)	1	1.92	3.57	6.25	10.00
E(p)	1	0.96	0.89	0.78	0.63

Redundancy

Redundancy measures the increase in the required computation when using more processing units. It measures the ratio between the number of operations performed by the parallel execution and by the sequential execution.

$$R(p) = \frac{O(p)}{O(1)}$$

O(1) is the total number of operations performed by one processing unit

O(p) is the total number of operations performed by ***p*** processing units

	1 CPU	2 CPUs	4 CPUs	8 CPUs	16 CPUs
O(p)	10000	10250	11000	12250	15000
R(p)	1	1.03	1.10	1.23	1.50

Utilization

Utilization is a measure of the good use of the computational capacity. It measures the ratio between the computational capacity used during parallel execution and the capacity that was available.

$$U(p) = R(p) \times E(p)$$

	1 CPU	2 CPUs	4 CPUs	8 CPUs	16 CPUs
R(p)	1	1.03	1.10	1.23	1.50
E(p)	1	0.96	0.89	0.78	0.63
U(p)	1	0.99	0.98	0.96	0.95

Amdahl's Law

We can divide the computations performed by a parallel application in three major classes:

- ***C(seq)*** – computations that can be done only sequencially
- ***C(par)*** – computations that can be done in parallel
- ***C(com)*** – computations related to parallel communication and synchronization

Using these classes, the speedup of an application can be defined as:

$$S(p) = \frac{T(1)}{T(p)} = \frac{C(seq) + C(par)}{C(seq) + \frac{C(par)}{p} + C(com)}$$

Amdahl's Law

Since ***C(com) ≥ 0*** then:

$$S(p) \leq \frac{C(seq)+C(par)}{C(seq)+\frac{C(par)}{p}}$$

Let ***f*** be the **fraction of the computation that can be realized only sequentially**:

$$f = \frac{C(seq)}{C(seq)+C(par)} \quad \text{and} \quad S(p) \leq \frac{\frac{C(seq)}{f}}{C(seq)+\frac{C(seq)\times\left(\frac{1}{f}-1\right)}{p}}$$

Amdahl's Law

Simplifying:

$$S(p) \leq \frac{\frac{C(seq)}{f}}{C(seq) + \frac{C(seq) \times \left(\frac{1}{f} - 1\right)}{p}}$$

$$\Rightarrow \quad S(p) \leq \frac{\frac{1}{f}}{1 + \frac{\frac{1}{f} - 1}{p}} \quad \Rightarrow \quad S(p) \leq \frac{1}{f + \frac{1-f}{p}}$$

Amdahl's Law

Let $0 \leq f \leq 1$ be the fraction of the computation that can be realized only sequentially then Amdahl's law tells us that the maximum speedup that a parallel application can achieve with p processing units is:

$$S(p) \leq \frac{1}{f + \frac{1-f}{p}}$$

Amdahl's law can also be used to determine the **limit of maximum speedup that a parallel application can achieve** regardless of the number of processing units available.

Amdahl's Law

Suppose one wants to determine if it is advantageous to develop a parallel version of a certain application. Through experimentation, it was verified that 90% of the execution time is spent in procedures that can be parallelizable. What is the maximum speedup that can be achieved with a parallel version of the application executing on 8 processing units?

$$S(p) \leq \frac{1}{0.1 + \frac{1-0.1}{8}} \approx 4{,}71$$

And the limit of maximum speedup that can be achieved?

$$\lim_{p \to \infty} \frac{1}{0.1 + \frac{1-0.1}{p}} = 10$$

Limitations of Amdahl's Law

Amdahl's law **ignores the costs with communication/synchronization operations** associated with the parallel version of the problem. For that reason, it can result in predictions not very realistic for certain problems.

Consider a parallel application with complexity $O(n^2)$ (where n is the size of the problem) whose execution pattern is the following:

- Execution time of the sequential part (input and output of data): $18000 + n$
- Execution time of the parallel part: $\frac{n^2}{100}$
- Total communication/synchronization points per processing unit: $\lceil \log n \rceil$
- Execution time due to communication/synchronization:

$$n \times \lceil \log p \rceil + \frac{n}{10}$$

Limitations of Amdahl's Law

What is the maximum speedup that can be achieved?

- Using Amdahl's law:

$$f = \frac{18000 + n}{18000 + n + \frac{n^2}{100}} \quad \text{and} \quad S(p) \leq \frac{18000 + n + \frac{n^2}{100}}{18000 + n + \frac{n^2}{p \times 100}}$$

- Using the speedup measure:

$$S(p) = \frac{18000 + n + \frac{n^2}{100}}{18000 + n + \frac{n^2}{p \times 100} + \lceil \log n \rceil \times \left(n \times \lceil \log p \rceil + \frac{n}{10} \right)}$$

Limitations of Amdahl's Law

		1 CPU	2 CPUs	4 CPUs	8 CPUs	16 CPUs
Amdahl's law	$n = 10000$	1	1.95	3.70	6.72	11.36
	$n = 20000$	1	1.98	3.89	7.51	14.02
	$n = 30000$	1	1.99	3.94	7.71	14.82
Speedup measure	$n = 10000$	1	1.61	2.11	2.22	2.57
	$n = 20000$	1	1.87	3.21	4.71	6.64
	$n = 30000$	1	1.93	3.55	5.89	9.29

Gustafson-Barsis' Law

Consider again the speedup measure defined previously:

$$S(p) \leq \frac{C(seq) + C(par)}{C(seq) + \frac{C(par)}{p}}$$

Let ***f*** be the **fraction of the parallel computation spent executing sequential computations** then ***(1 - f)*** is the **fraction of the time spent in the parallel part**:

$$f = \frac{C(seq)}{C(seq) + \frac{C(par)}{p}} \quad \text{and} \quad (1 - f) = \frac{\frac{C(par)}{p}}{C(seq) + \frac{C(par)}{p}}$$

Gustafson-Barsis' Law

Then:

$$C(seq) = f \times \left(C(seq) + \frac{C(par)}{p} \right)$$

$$C(par) = p \times (1 - f) \times \left(C(seq) + \frac{C(par)}{p} \right)$$

Simplifying:

$$S(p) \leq \frac{(f + p \times (1 - f)) \times \left(C(seq) + \frac{C(par)}{p} \right)}{C(seq) + \frac{C(par)}{p}}$$

$$\Rightarrow \quad S(p) \leq f + p \times (1 - f) \qquad \Rightarrow \qquad S(p) \leq p + f \times (1 - p)$$

Gustafson-Barsis' Law

Let $0 \leq f \leq 1$ be the fraction of parallel computation spent executing sequential computations then Gustafson-Barsis' law tells us that the maximum speedup that a parallel application can achieve with p processing units is:

$$S(p) \leq p + f \times (1 - p)$$

While Amdahl's law **starts from the sequential execution time** to estimate the maximum speedup that can be achieved with multiple processing units, Gustafson-Barsis' law does the opposite, i.e., it **starts from the parallel execution time** to estimate the maximum speedup in comparison with the sequential execution.

Gustafson-Barsis' Law

Consider that a certain application executes in 220 seconds in 64 processing units. What is the maximum speedup of the application knowing that, by experimentation, 5% of the execution time is spent on sequential computations.

$$S(p) \leq 64 + (0.05) \times (1 - 64)$$

$$S(p) \leq 64 - 3.15$$

$$S(p) \leq 60.85$$

Gustafson-Barsis' Law

Consider that a certain company wants to buy a supercomputer with 16384 processors to achieve a speedup of 15000 in an important and fundamental problem. What is the maximum fraction of the parallel execution that can be spent in sequential computations to achieve the expected speedup?

$$15000 \leq 16384 + f \times (1 - 16384)$$

$$f \times 16383 \leq 1384$$

$$f \leq 0.084$$

Limitations of Gustafson-Barsis' Law

By using the parallel execution time as the starting point, instead of the sequential execution time, the Gustafson-Barsis law assumes that the execution time with one processing unit is, in the worst case, ***p*** times slower than the execution with ***p*** processing units.

This may not be true if the available **memory for the execution with one processing unit is insufficient** when compared to the computation with ***p*** processing units. For this reason, the estimated speedup by the Gustafson-Barsis law is often designated as ***scaled speedup***.

Karp-Flatt Metric

Let us consider again the definitions of sequential and parallel execution time:

$$T(1) = C(seq) + C(par)$$

$$T(p) = C(seq) + \frac{C(par)}{p} + C(com)$$

Let ***e*** be the **experimentally determined sequential fraction of a parallel computation**:

$$e = \frac{C(seq)}{T(1)}$$

Karp-Flatt Metric

Then:

$$C(seq) = e \times T(1)$$

$$C(par) = (1-e) \times T(1)$$

If one considers that ***C(com)*** is negligible then:

$$T(p) = e \times T(1) + \frac{(1-e) \times T(1)}{p}$$

On the other hand:

$$S(p) = \frac{T(1)}{T(p)}$$

$$\Rightarrow \quad T(1) = S(p) \times T(p)$$

Karp-Flatt Metric

Simplifying:

$$T(p) = e \times S(p) \times T(p) + \frac{(1-e) \times S(p) \times T(p)}{p}$$

$$\Rightarrow \quad 1 = e \times S(p) + \frac{(1-e) \times S(p)}{p}$$

$$\Rightarrow \quad \frac{1}{S(p)} = e + \frac{(1-e)}{p} \qquad \Rightarrow \qquad \frac{1}{S(p)} = e + \frac{1}{p} - \frac{e}{p}$$

$$\Rightarrow \quad \frac{1}{S(p)} = e \times \left(1 - \frac{1}{p}\right) + \frac{1}{p} \qquad \Rightarrow \qquad e = \frac{\frac{1}{S(p)} - \frac{1}{p}}{1 - \frac{1}{p}}$$

Karp-Flatt Metric

Let ***S(p)*** be the speedup of a parallel application with ***p > 1*** processing units then the Karp-Flatt metric tells us that the experimentally determined sequential fraction is:

$$e = \frac{\frac{1}{S(p)} - \frac{1}{p}}{1 - \frac{1}{p}}$$

The Karp-Flatt metric is interesting because by neglecting the costs with communication/synchronization operations associated with parallelism, allows us **to determine, a posteriori, the relevance of the *C(com)* component** in the eventual decrease of the application's efficiency.

Karp-Flatt Metric

By definition, the experimentally determined sequential fraction is a constant value that does not depend on the number of processing units.

$$e = \frac{C(seq)}{T(1)}$$

On the other hand, the Karp-Flatt metric is a function of the number of processing units.

$$e = \frac{\frac{1}{S(p)} - \frac{1}{p}}{1 - \frac{1}{p}}$$

Karp-Flatt Metric

Considering that the efficiency of an application is a decreasing function on the number of processing units, the Karp-Flatt metric allows us to determine the relevance of ***C(com)*** component in that decrease.

- If the **values of *e* are constant** as we increase the number of processing units then that means that the ***C(com)*** component is also constant, i.e., **the efficiency decrease is due to the scarce parallelism available in the application**
- If the **values of *e* increase** as we increase the number of processing units then that means that the ***C(com)*** component is also increasing, i.e., **the efficienct decrease is due to the excessive costs associated with the parallel computation** (initialization, communication and/or synchronization costs)

The Karp-Flatt metric allows us to detect sources of inefficiency not considered by the model which assumes that ***p*** processing units execute the parallel part ***p*** times faster than executing with just one unit.

Karp-Flatt Metric

Consider the speedups obtained by a parallel application:

	2 CPUs	3 CPUs	4 CPUs	5 CPUs	6 CPUs	7 CPUs	8 CPUs
S(p)	1.82	2.50	3.08	3.57	4.00	4.38	4.71
e	0.099	0.100	0.100	0.100	0.100	0.100	0.100

What is the main reason for the application to just achieve a speedup of 4.71 with 8 processors?

- Given that ***e*** not increases with the number of processors, the main reason for the small speedup is the scarce parallelism avaiable in the application

Karp-Flatt Metric

Consider the speedups obtained by a parallel application:

	2 CPUs	3 CPUs	4 CPUs	5 CPUs	6 CPUs	7 CPUs	8 CPUs
S(p)	1.87	2.61	3.23	3.73	4.14	4.46	4.71
e	0.070	0.075	0.079	0.085	0.090	0.095	0.100

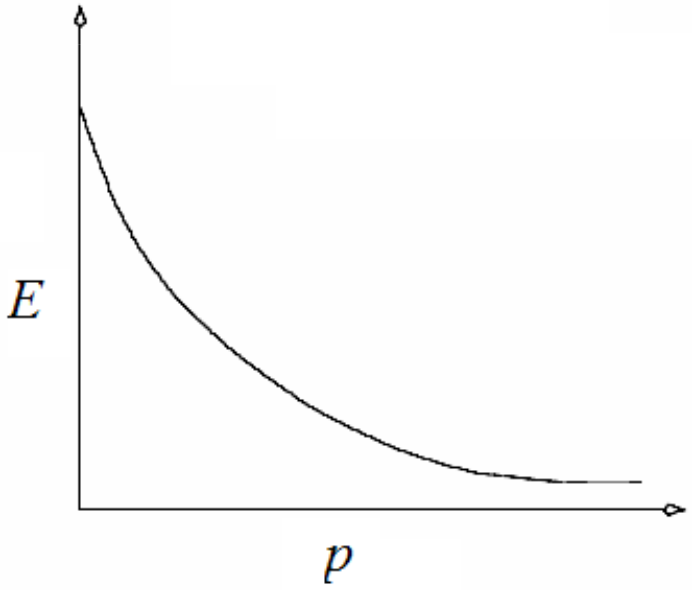
What is the main reason for the application to just achieve a speedup of 4.71 with 8 processors?

- Given that ***e*** increases with the number of processors, the main reason for the small speedup are the excessive costs associated to the parallel computation

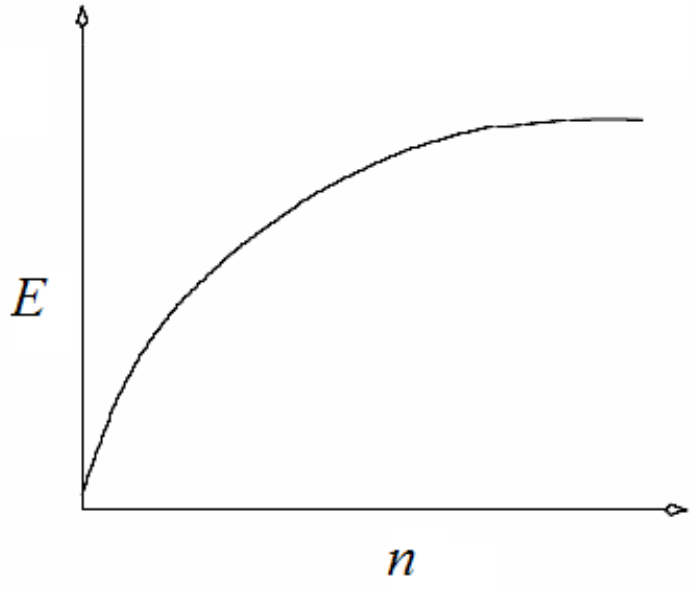
Efficiency and Scalability

From the previous results, we can conclude that the efficiency of an application is:

- A **decreasing function on the number of processing units**
- Typically, an **increasing function on the size of the probem**



Problem size fixed (n)



Number of processing units fixed (p)

Efficiency and Scalability

An application is said to be **scalable** when it shows capacity to maintain the **same efficiency as the number of processing units and the size of the problem are increased proportionally**.

		1 CPU	2 CPUs	4 CPUs	8 CPUs	16 CPUs
Efficiency	$n = 10000$	1	0.81	0.53	0.28	0.16
	$n = 20000$	1	0.94	0.80	0.59	0.42
	$n = 30000$	1	0.96	0.89	0.74	0.58

The scalability of an application reflects its **capacity of making use of more computational resources effectively**.

Isoefficiency Metric

Typically, the efficiency of an application is an increasing function on the size of the problem since the complexity of communication is, usually, smaller than the complexity of computation, i.e., **to maintain the same level of efficiency as we increase the number of processing units one needs to increase the size of the problem**. The isoefficiency metric formalizes this idea.

Let us consider again the definition of speedup:

$$S(p) = \frac{C(seq) + C(par)}{C(seq) + \frac{C(par)}{p} + C(com)} = \frac{p \times \left(C(seq) + C(par)\right)}{p \times C(seq) + C(par) + p \times C(com)}$$

$$= \frac{p \times \left(C(seq) + C(par)\right)}{C(seq) + C(par) + (p-1) \times C(seq) + p \times C(com)}$$

Isoefficiency Metric

Let $T_o(p)$ be the **execution time spent by p processing units on the parallel algorithm performing computations not done in the sequential algorithm**:

$$T_0(p) = (p-1) \times C(seq) + p \times C(com)$$

Simplifying:

$$S(p) = \frac{p \times \left(C(seq) + C(par)\right)}{C(seq) + C(par) + T_0(p)}$$

$$E(p) = \frac{C(seq) + C(par)}{C(seq) + C(par) + T_0(p)} = \frac{1}{1 + \frac{T_0(p)}{C(seq) + C(par)}} = \frac{1}{1 + \frac{T_0(p)}{T(1)}}$$

Isoefficiency Metric

Then:

$$E(p) = \frac{1}{1 + \frac{T_0(p)}{T(1)}}$$

$$\Rightarrow \quad \frac{T_0(p)}{T(1)} = \frac{1 - E(p)}{E(p)} \qquad \Rightarrow \qquad T(1) = \frac{E(p)}{1 - E(p)} \times T_0(p)$$

If one wants to **maintain the same level of efficiency** as we increase the number of processing units, then:

$$\frac{E(p)}{1 - E(p)} = c \qquad \text{and} \qquad T(1) \geq c \times T_0(p)$$

Isoefficiency Metric

Let ***E(p)*** be the efficiency of a parallel application with ***p*** processing units then the isoefficiency metric tells us that, to maintain the same level of efficiency as we increase the number of processing units, the size of the problem must be increased so that the following inequality is satisfied:

$$T(1) \geq c \times T_0(p)$$

$$with \quad c = \frac{E(p)}{1 - E(p)} \quad \text{and} \quad T_0(p) = (p-1) \times C(seq) + p \times C(com)$$

The applicability of the isoefficiency metric **may depend on the available memory**, since the maximum size of the problem that can be solved is limited by that quantity.

Isoefficiency Metric

Suppose that the isoefficiency metric for a problem of size ***n*** is given as a function on the number of processing units ***p***:

$$n \geq f(p)$$

If ***M(n)*** designates the quantity of required memory to solve a problem of size ***n*** then:

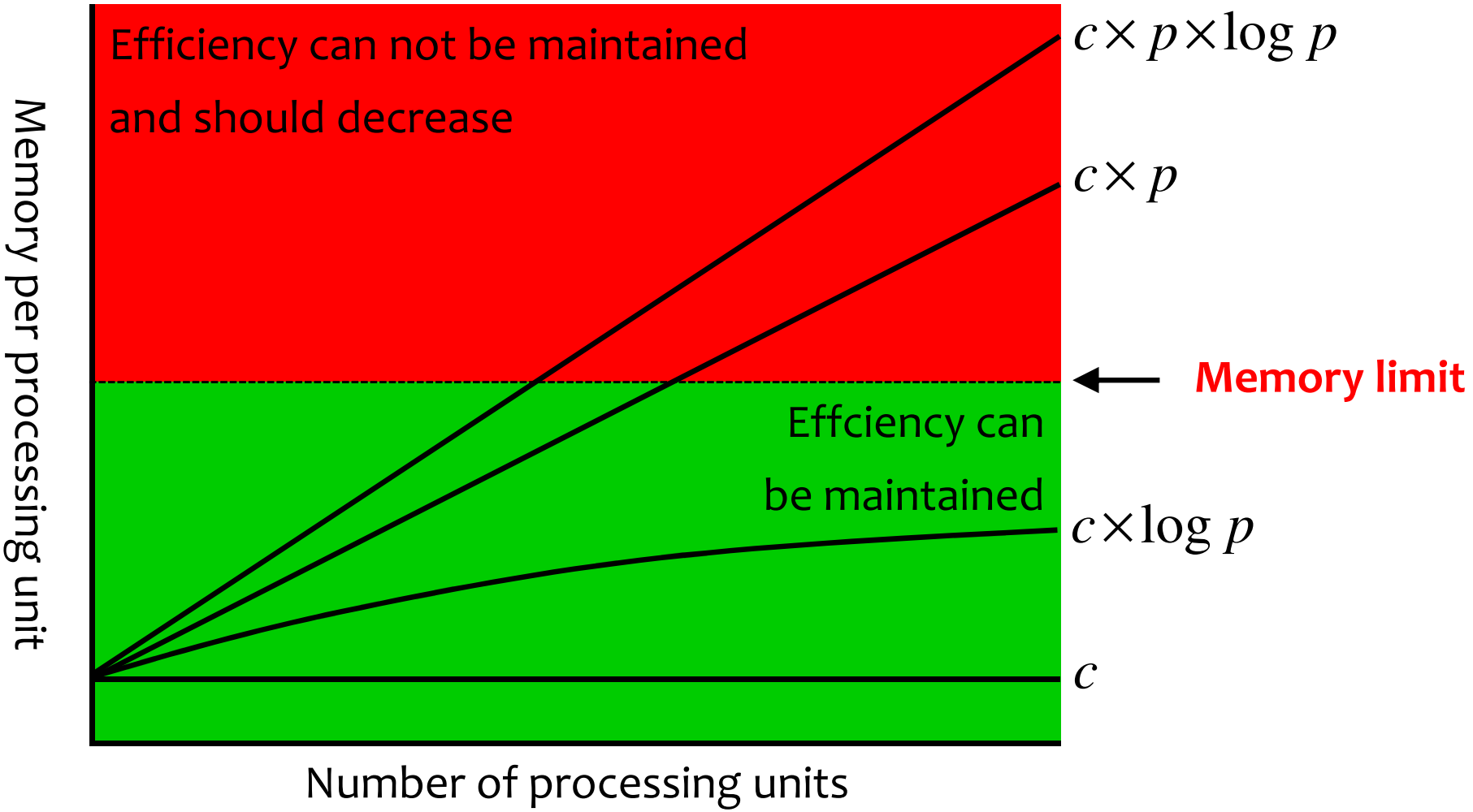
$$M(n) \geq M\big(f(p)\big)$$

That is, to maintain the same level of efficiency, the quantity of memory required per processing unit is:

$$\frac{M(n)}{p} \geq \frac{M\big(f(p)\big)}{p}$$

Isoefficiency Metric

To maintain the same level of efficiency, as we increase the number of processing units, there is a **memory limit on the size of the problem**.



Isoefficiency Metric

Consider that the sequential version of a certain application has complexity $O(n^3)$ and that the execution time spent by each of the p processing units of the parallel version in communication and synchronization operations is $O(n^2 \log p)$. If the amount of memory necessary to represent a problem of size n is n^2, what is the scalability of the application in terms of memory?

$$n^3 \geq c \times p \times n^2 \times \log p$$

$$n \geq c \times p \times \log p$$

$$M(n) = n^2 \quad \Rightarrow \quad \frac{M(c \times p \times \log p)}{p} = \frac{c^2 \times p^2 \times \log^2 p}{p} = c^2 \times p \times \log^2 p$$

Thus, the scalability of the application is low.

Superlinear Speedup

We say that the speedup is superlinear when the ratio between the sequential execution time and the parallel execution time with ***p*** processing units is greater than ***p***.

$$\frac{T(1)}{T(p)} \geq p$$

Some factors that may make the speedup superlinear are:

- Almost inexistent initialization, comunication and/or synchronization costs
- Tolerancy to communication latency
- Increased memory capacity (the problem may start to fit all in memory)
- Subdivision of the problem (smaller tasks may generate less cache misses)
- Computation randomness in optimization problems or with multiple solutions