Por que usar programação paralela?

Por que precisamos de programação paralela?

https://computing.llnl.gov/tutorials/parallel_comp/

(resposta 1)

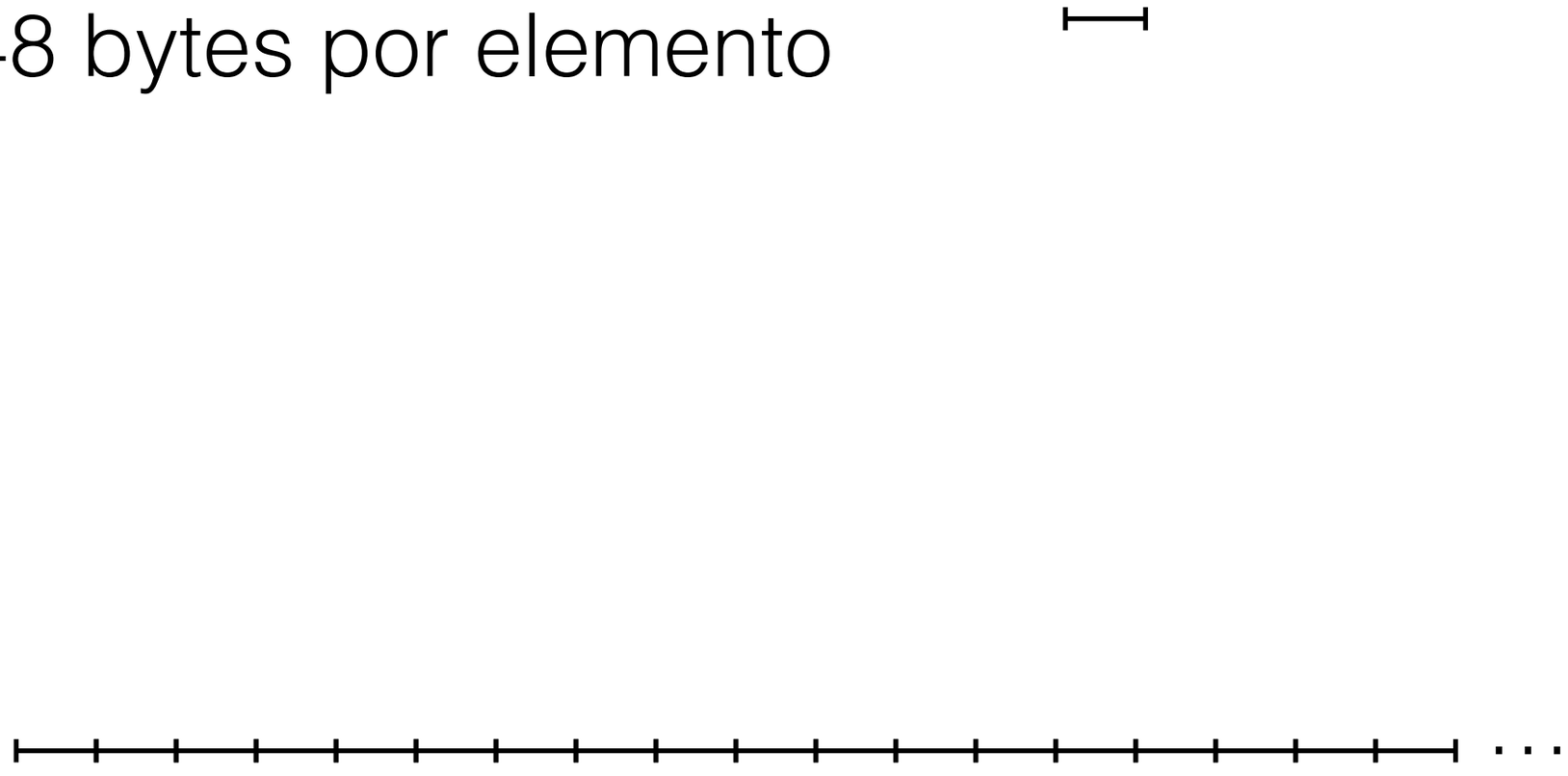
GRANDES problemas
(uma questão de *n*-dimensões)

1D: 48 bytes por elemento

(resposta 1)

GRANDES problemas
(uma questão de *n*-dimensões)

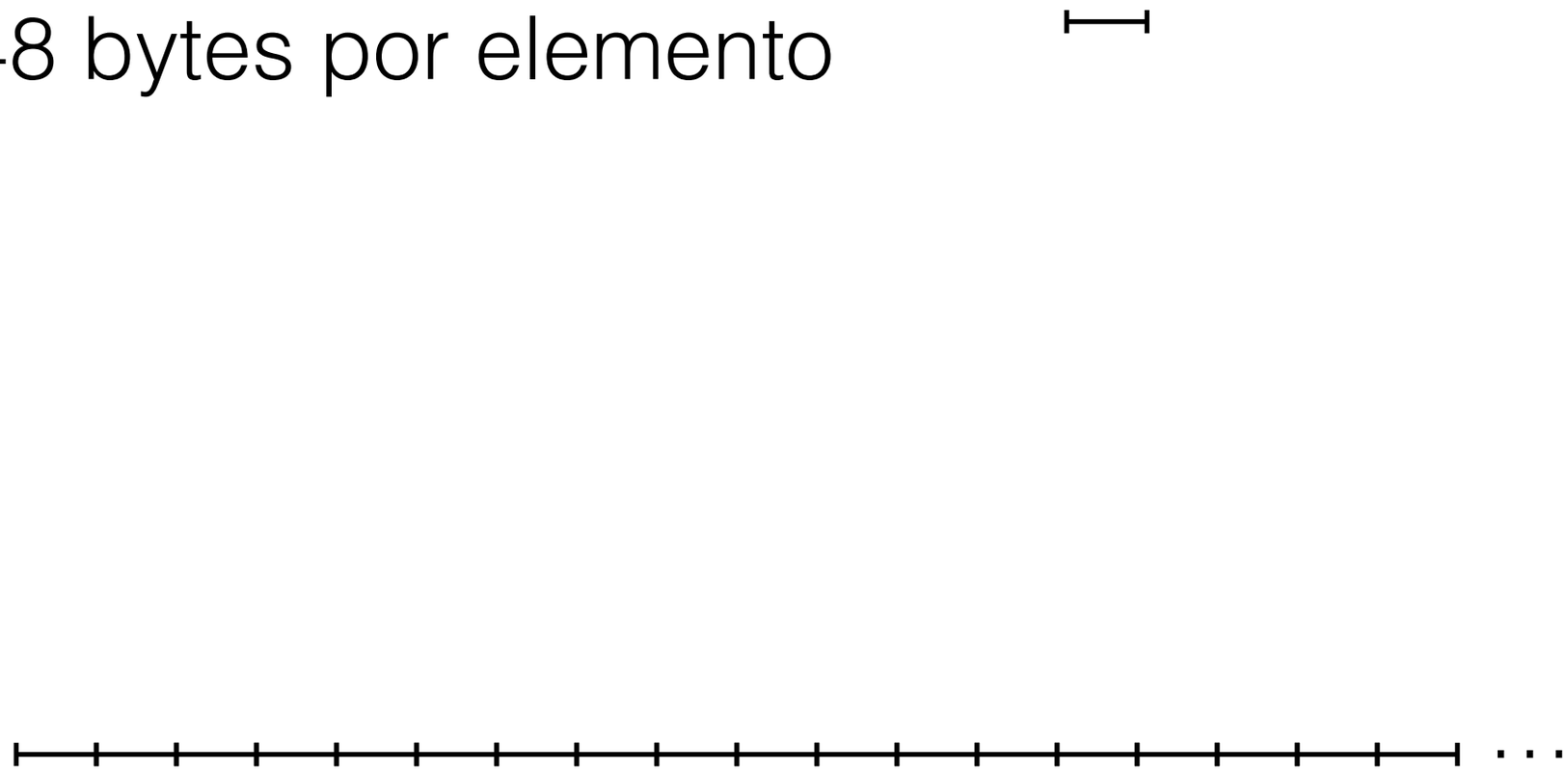
1D: 48 bytes por elemento



(resposta 1)

GRANDES problemas
(uma questão de *n*-dimensões)

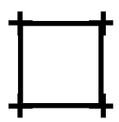
1D: 48 bytes por elemento



1000 elementos -> 48 kB

(resposta 1)

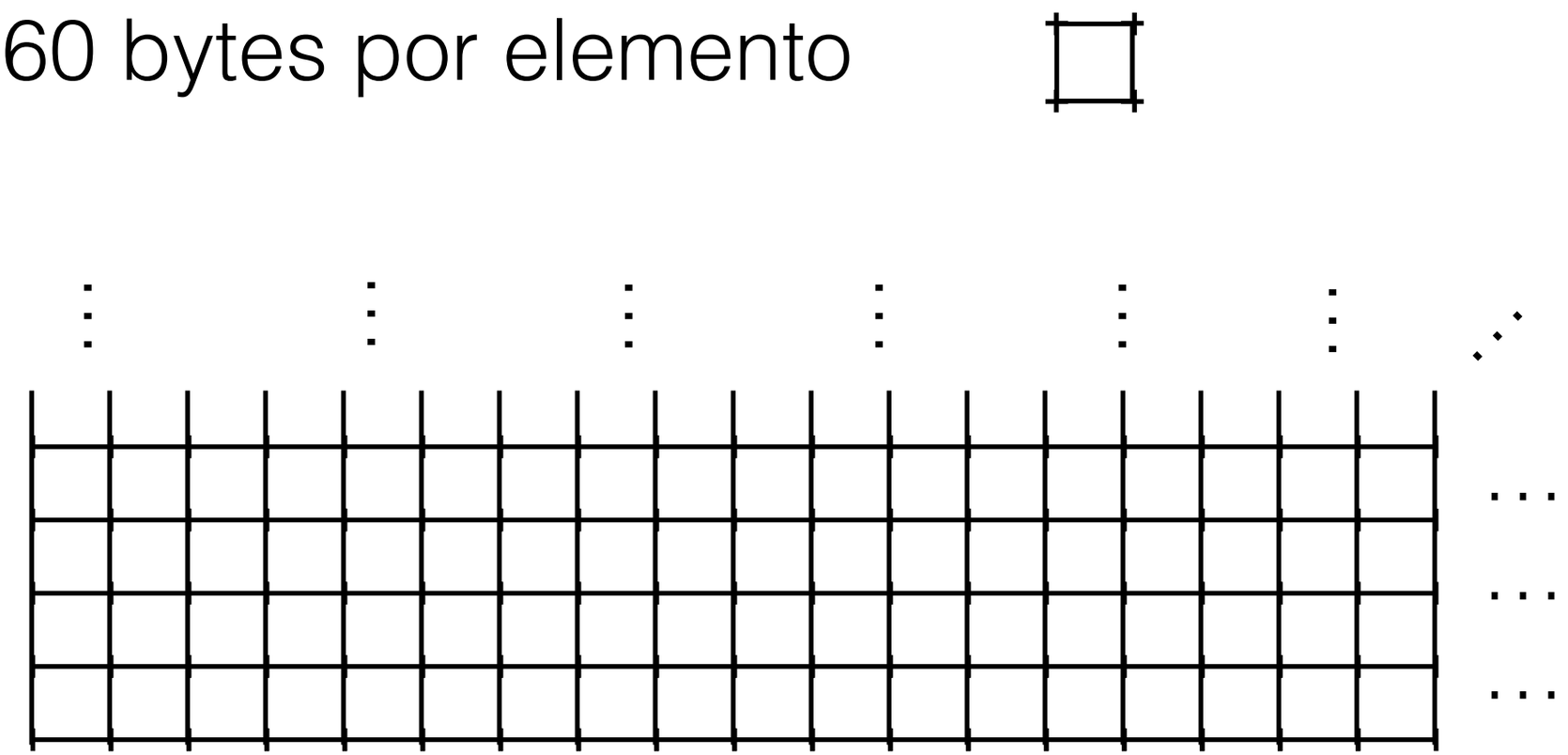
GRANDES problemas
(uma questão de *n*-dimensões)

2D: 160 bytes por elemento 

(resposta 1)

GRANDES problemas
(uma questão de *n*-dimensões)

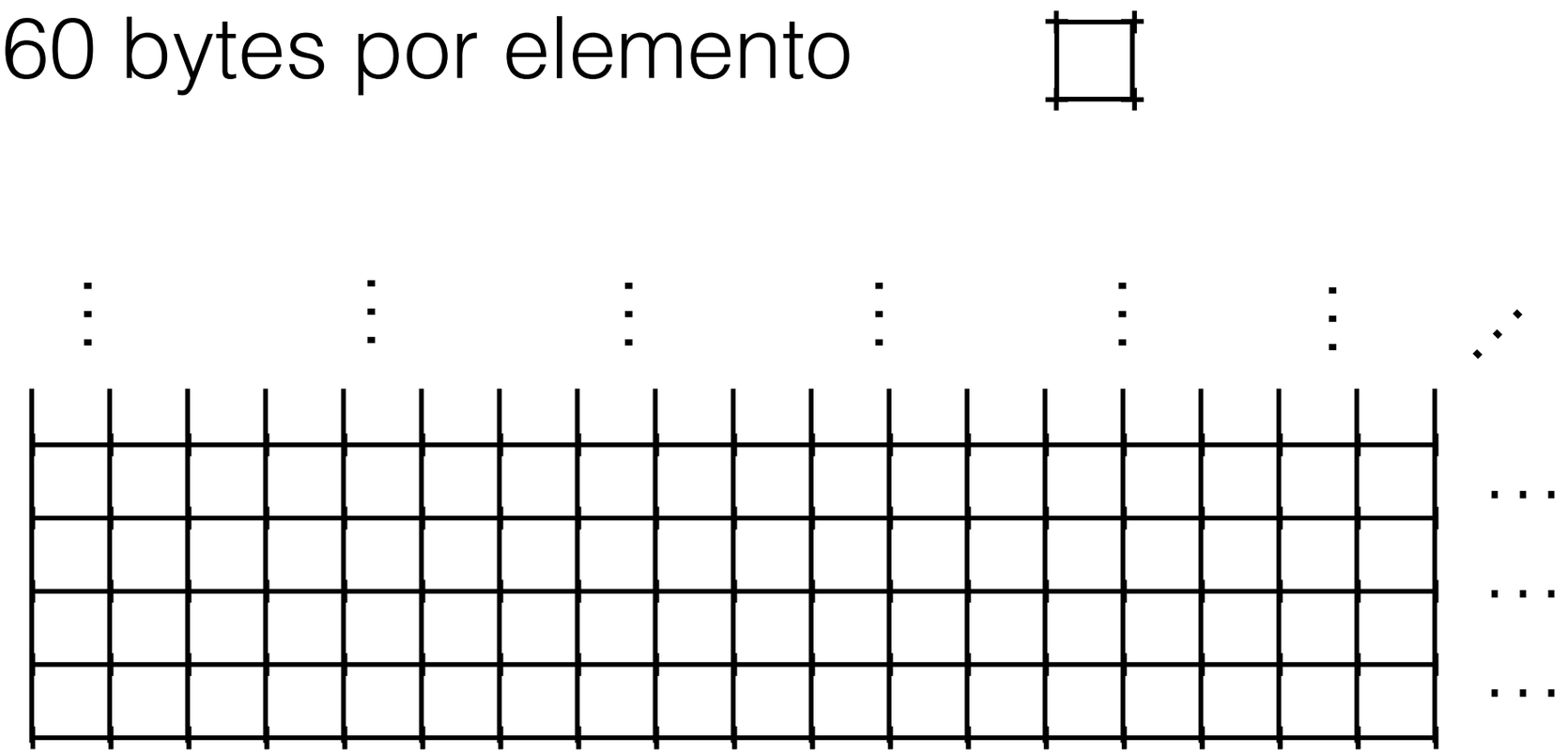
2D: 160 bytes por elemento



(resposta 1)

GRANDES problemas
(uma questão de *n*-dimensões)

2D: 160 bytes por elemento

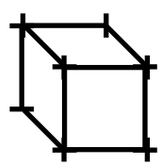


1000x1000 elementos -> 160 MB

(resposta 1)

GRANDES problemas
(uma questão de *n*-dimensões)

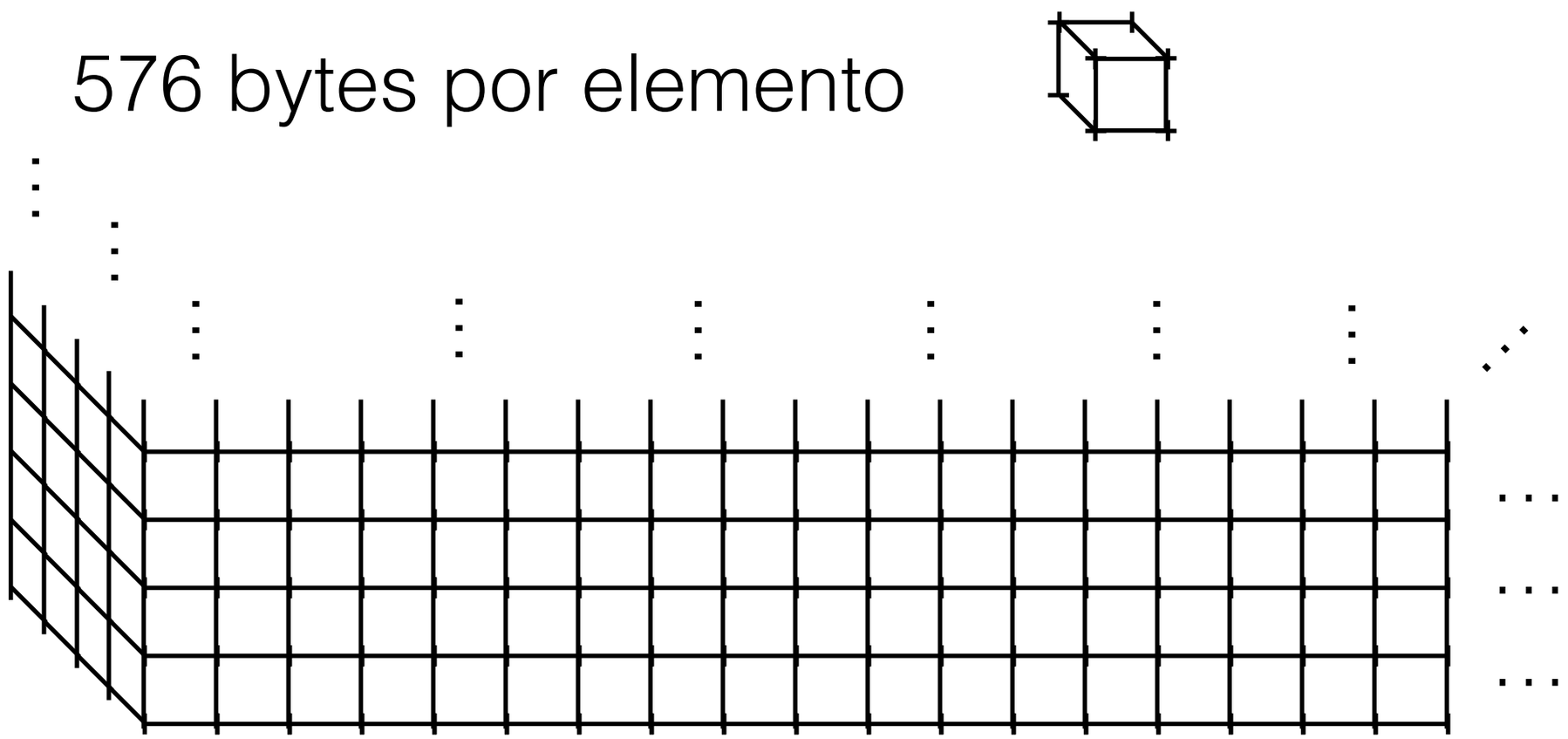
3D: 576 bytes por elemento



(resposta 1)

GRANDES problemas
(uma questão de *n*-dimensões)

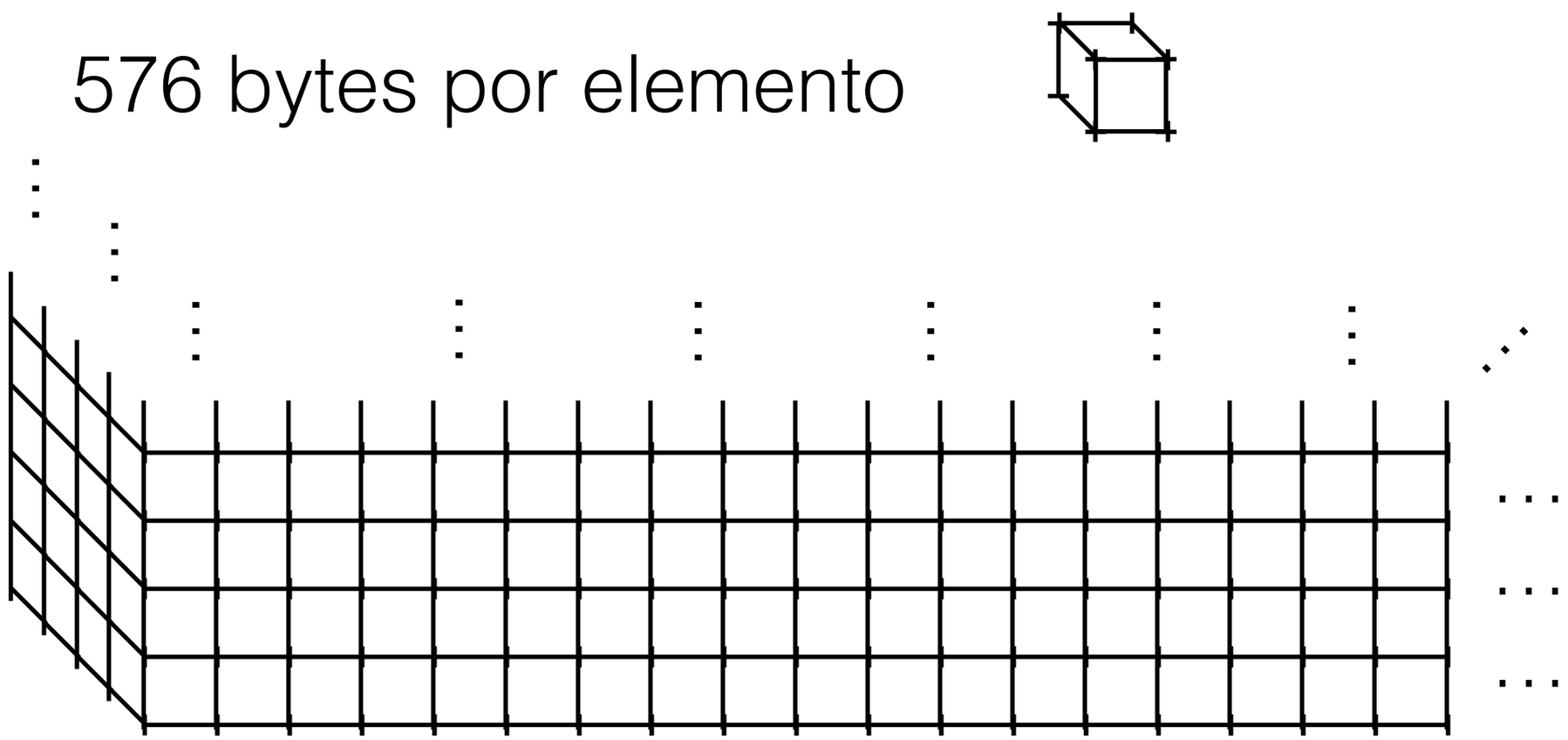
3D: 576 bytes por elemento



(resposta 1)

GRANDES problemas
(uma questão de *n*-dimensões)

3D: 576 bytes por elemento



1000x1000x1000 elementos -> 576 GB

(resposta 1)

GRANDES problemas

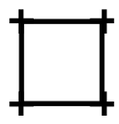
(uma questão de *n*-dimensões)

(resposta 1)

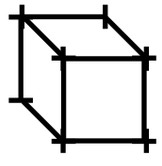
GRANDES problemas
(uma questão de *n*-dimensões)

1D: 48 bytes por elemento 

1000 elementos -> 48 kB

2D: 160 bytes por elemento 

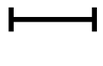
1000x1000 elementos -> 160 MB

3D: 576 bytes por elemento 

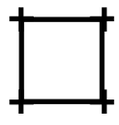
1000x1000x1000 elementos -> 576 GB

(resposta 1)

GRANDES problemas
(uma questão de *n*-dimensões)

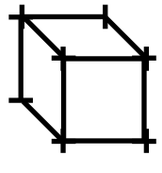
1D: 48 bytes por elemento 

1000 elementos -> 48 kB

2D: 160 bytes por elemento 

~3333x

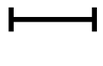
1000x1000 elementos -> 160 MB

3D: 576 bytes por elemento 

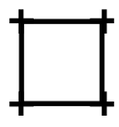
1000x1000x1000 elementos -> 576 GB

(resposta 1)

GRANDES problemas
(uma questão de *n*-dimensões)

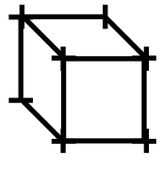
1D: 48 bytes por elemento 

1000 elementos -> 48 kB

2D: 160 bytes por elemento 

~3333x

1000x1000 elementos -> 160 MB

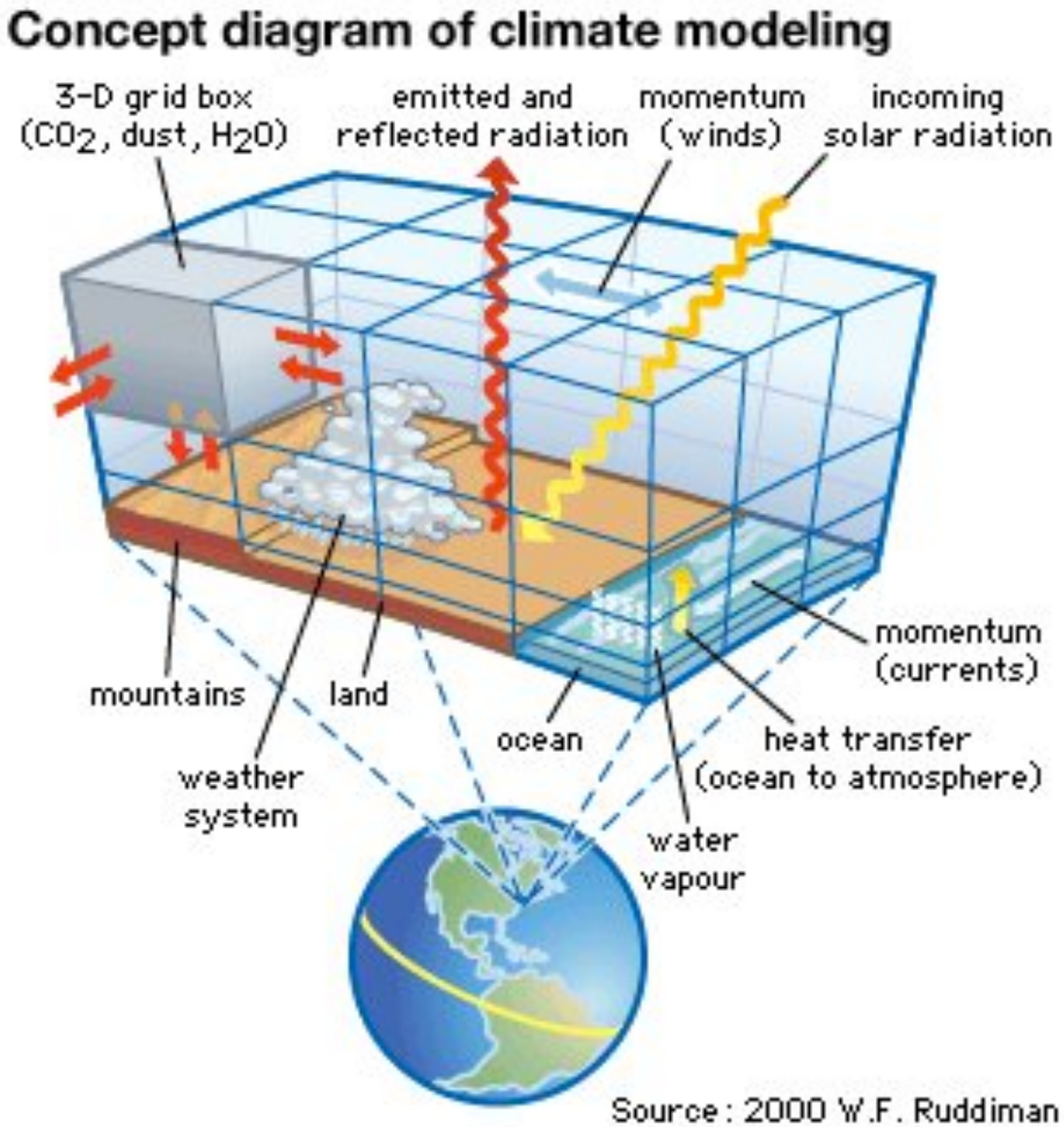
3D: 576 bytes por elemento 

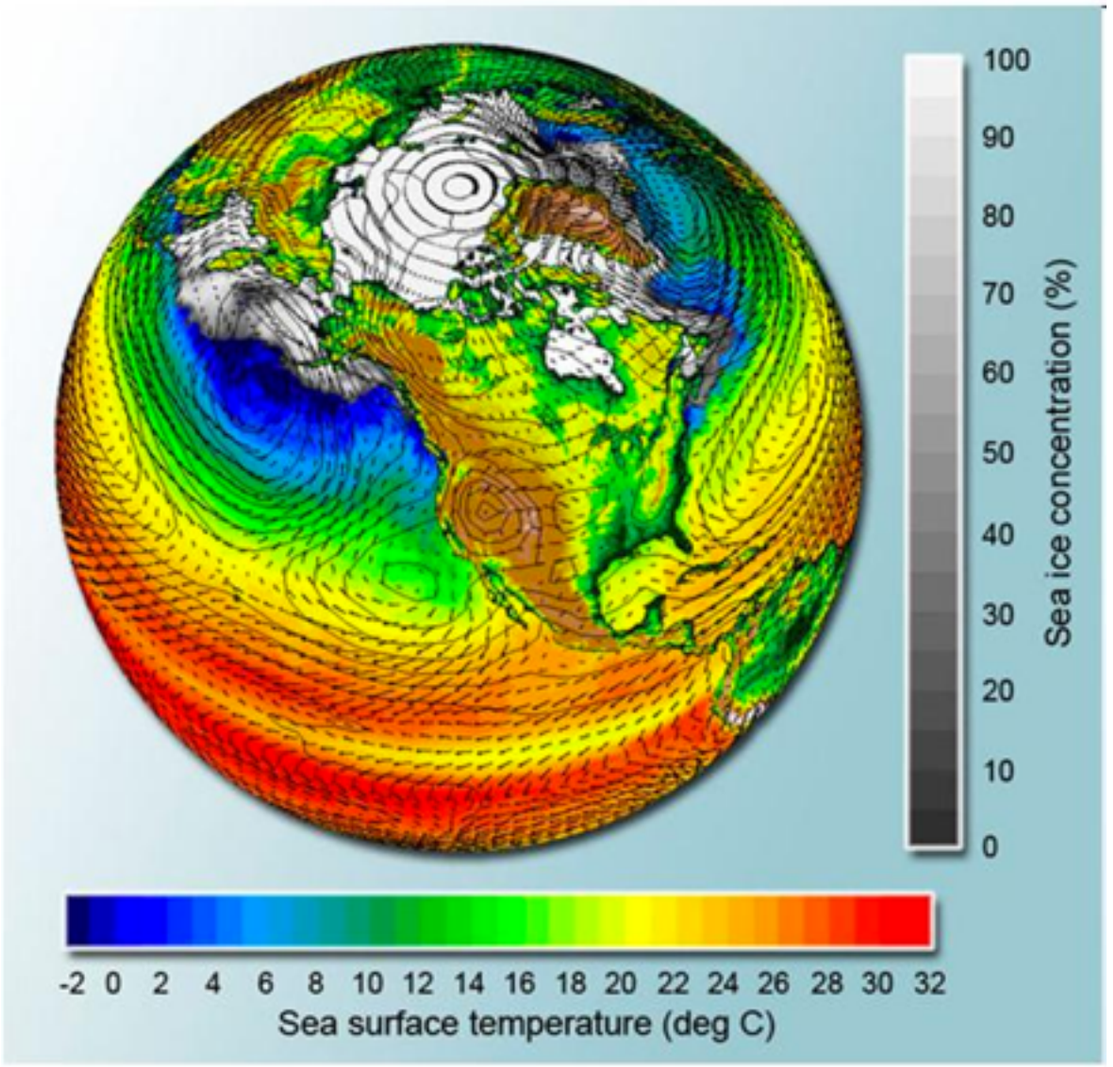
3600x

1000x1000x1000 elementos -> 576 GB

(resposta 2)

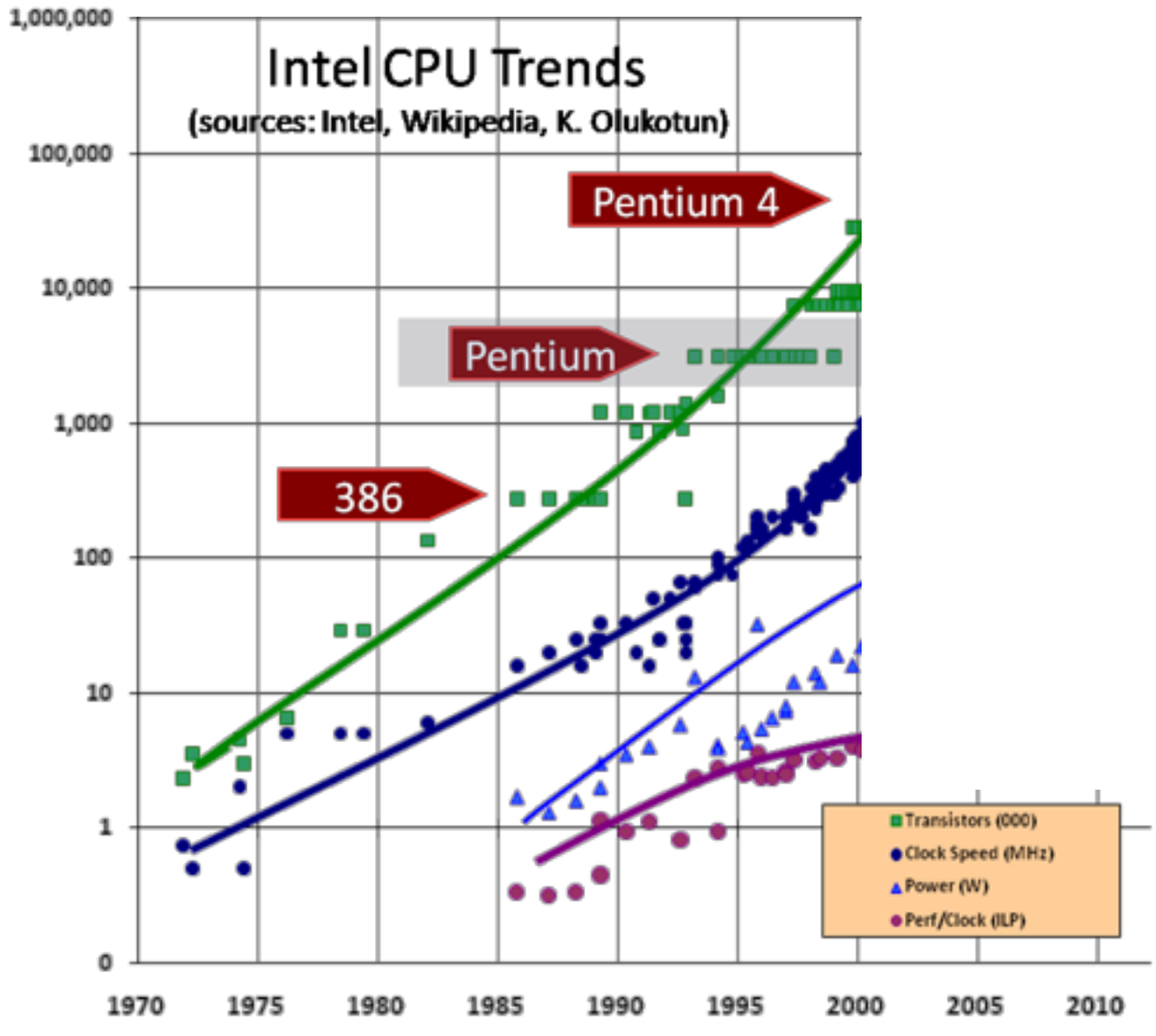
TEMPO de execução





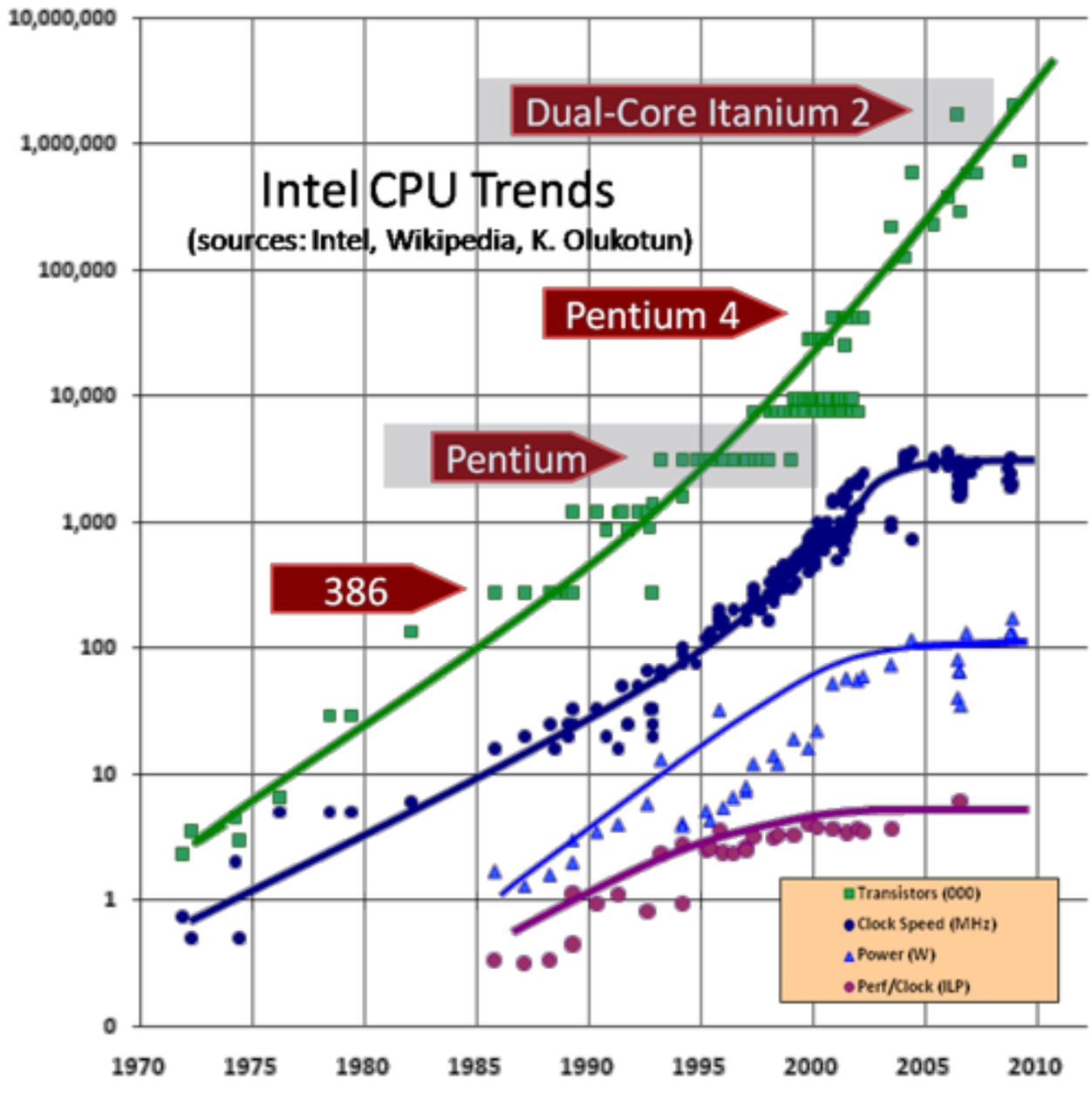
http://www.drroyspencer.com/2009/07/how-do-climate-models-work/

Performance dos computadores através do tempo



http://www.gotw.ca/publications/concurrency-ddj.htm

Performance dos computadores através do tempo



http://www.gotw.ca/publications/concurrency-ddj.htm

Trabalho em equipe

Primórdios da computação paralela
(o computador humano)

NACA High Speed Flight Station
"Computer Room"

http://en.wikipedia.org/wiki/Human_computer

Primórdios da computação paralela
(o computador humano)

A COMPUTER WANTED.

WASHINGTON, May 1.—A civil service examination will be held May 18 in Washington, and, if necessary, in other cities, to secure eligibles for the position of computer in the Nautical Almanac Office, where two vacancies exist—one at $1,000, the other at $1,400..

The examination will include the subjects of algebra, geometry, trigonometry, and astronomy. Application blanks may be obtained of the United States Civil Service Commission.

The New York Times

NACA High Speed Flight Station
"Computer Room"

http://en.wikipedia.org/wiki/Human_computer

Primórdios da computação paralela
(o computador humano)

"The human computer is supposed to be following fixed rules; he has no authority to deviate from them in any detail." (Turing, 1950)

Gravidez

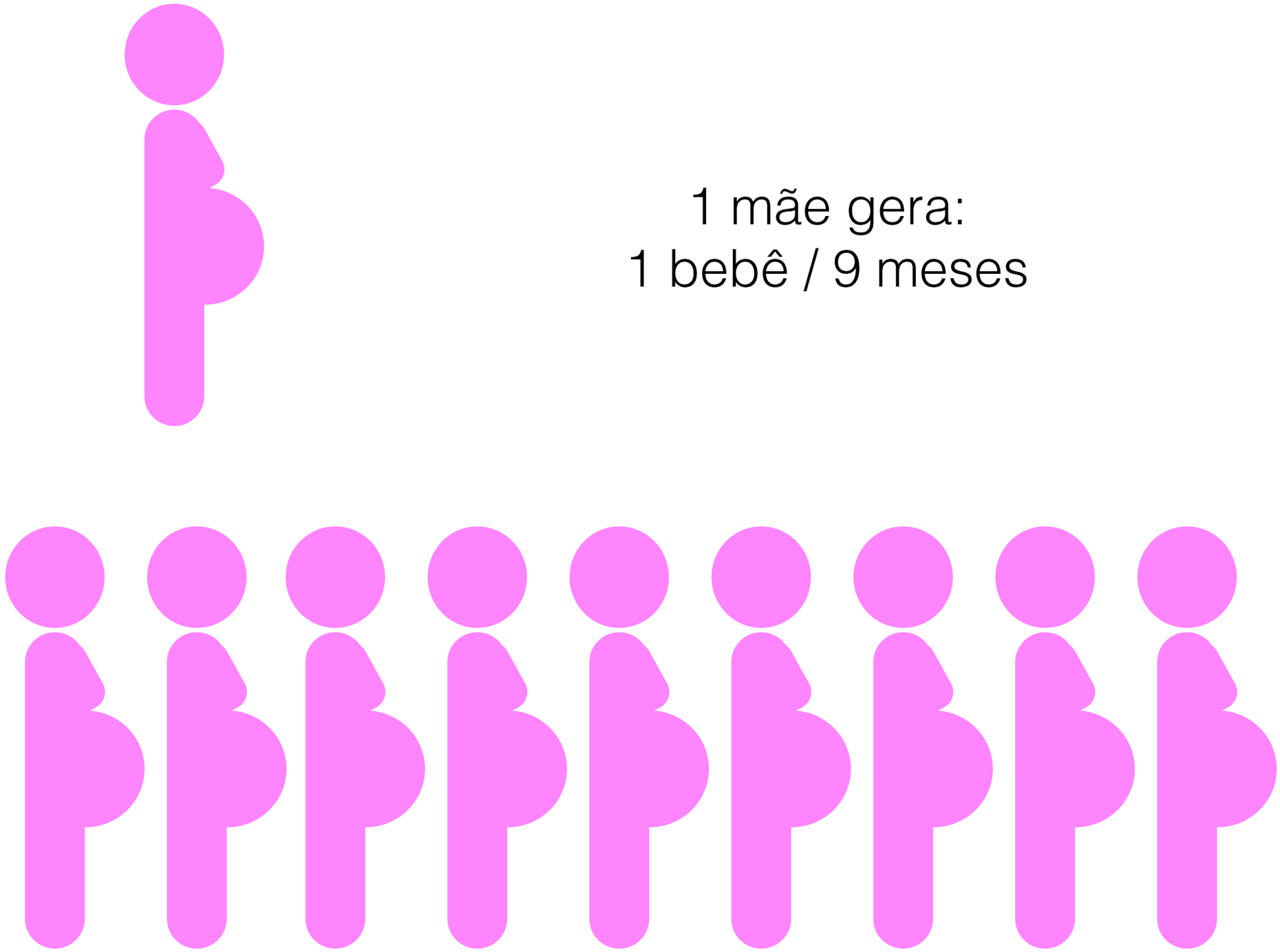
Gravidez

Gravidez

1 mãe gera:
1 bebê / 9 meses

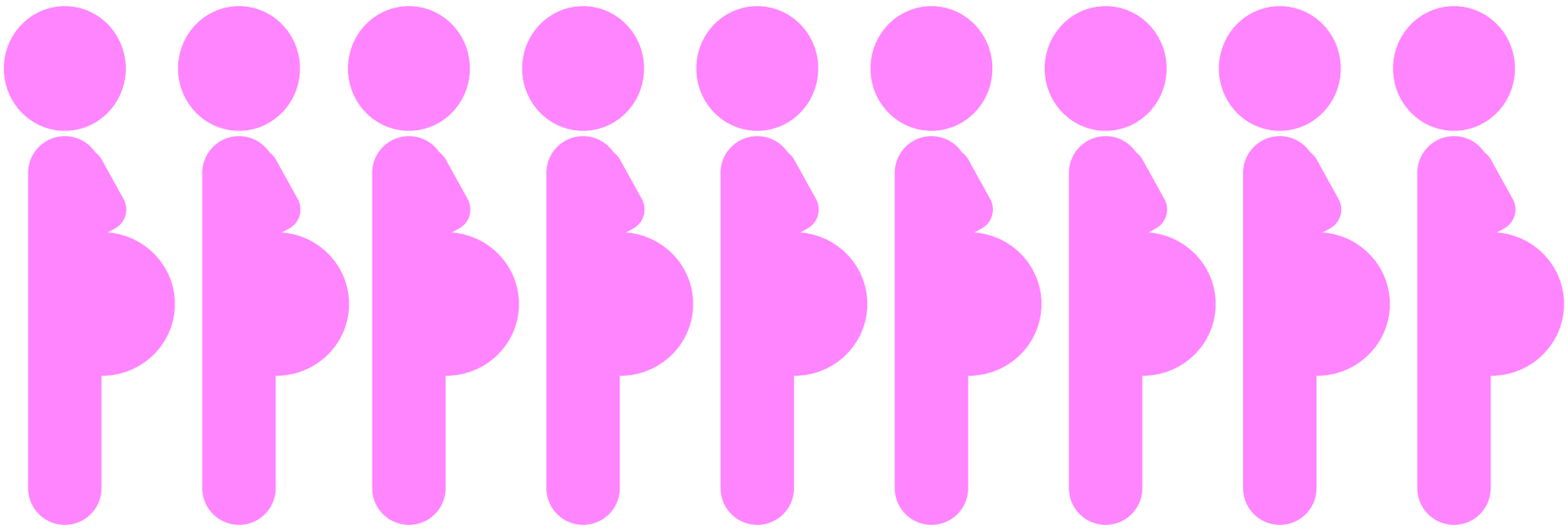
Gravidez

1 mãe gera:
1 bebê / 9 meses



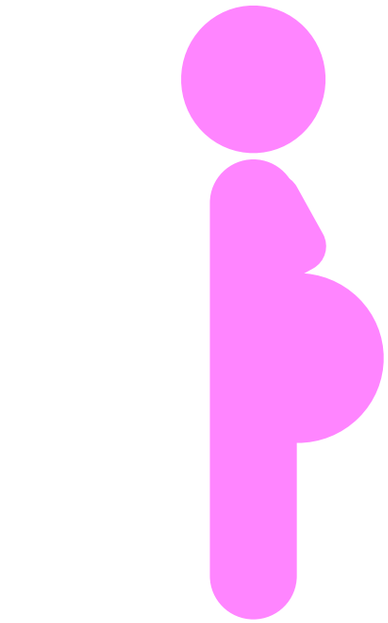
Gravidez

1 mãe gera:
1 bebê / 9 meses

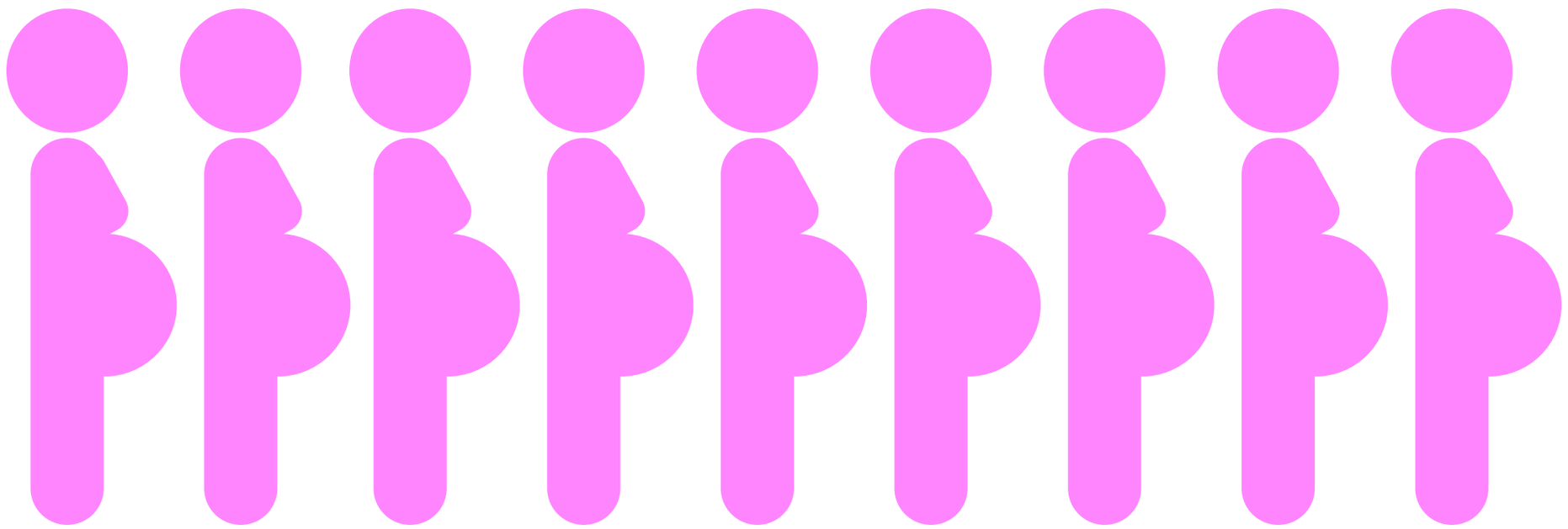


9 mães geram:

Gravidez



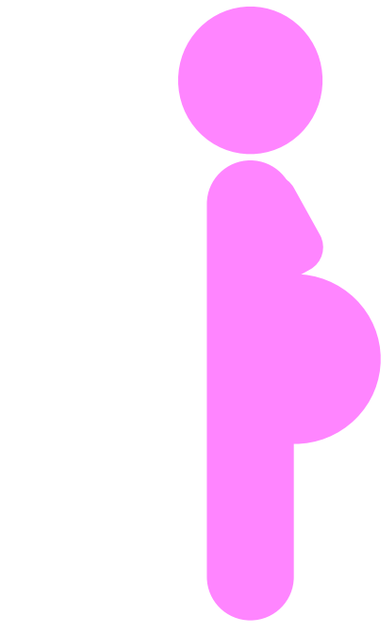
1 mãe gera:
1 bebê / 9 meses



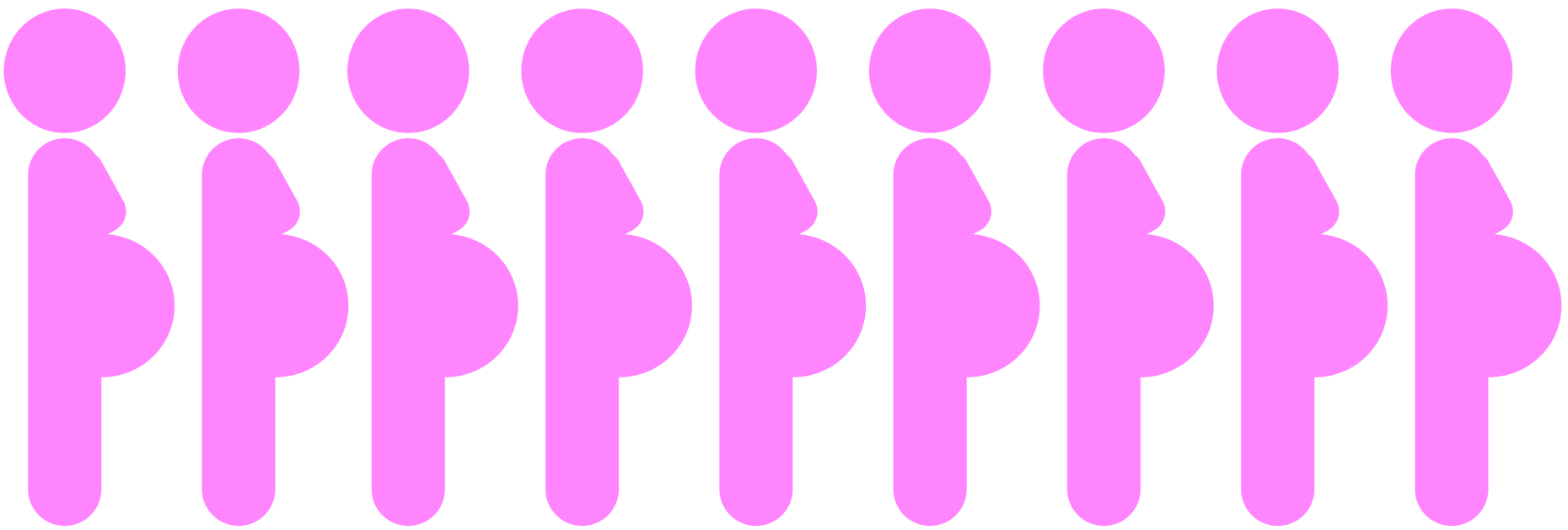
9 mães geram:

1 bebê / 1 mês?!

Gravidez



1 mãe gera:
1 bebê / 9 meses

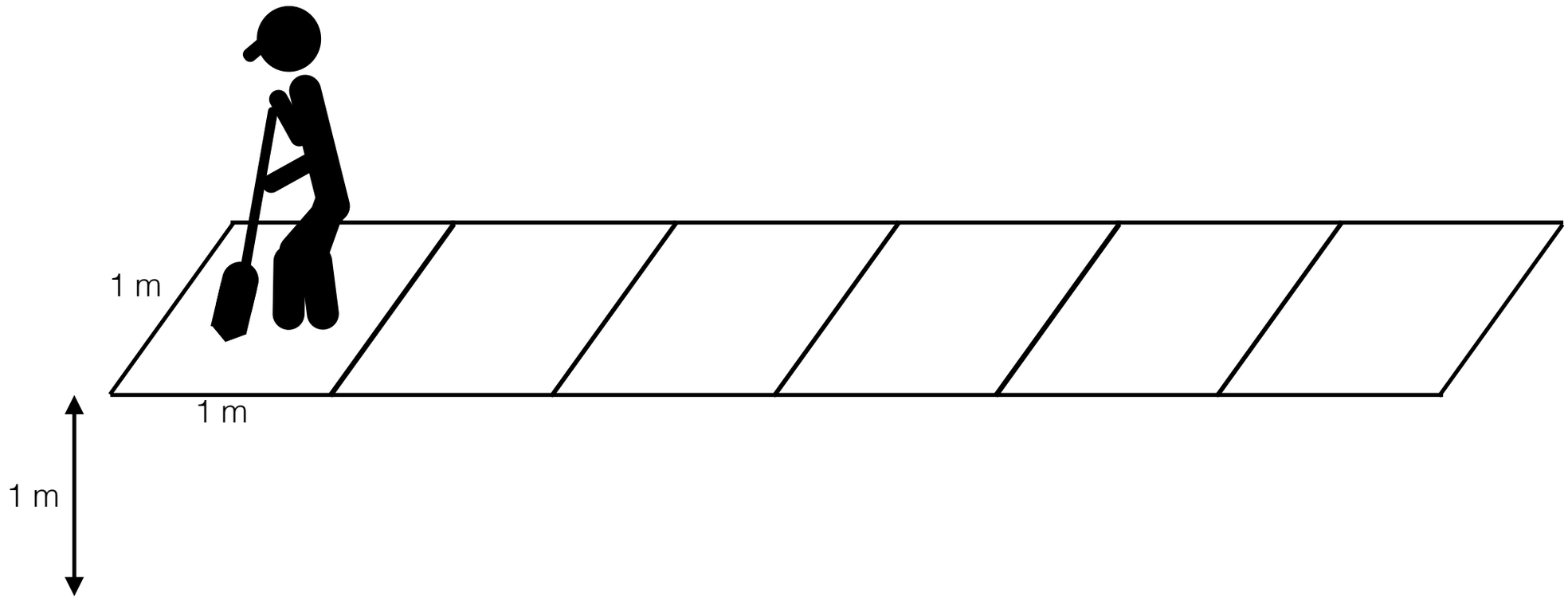


9 mães geram:

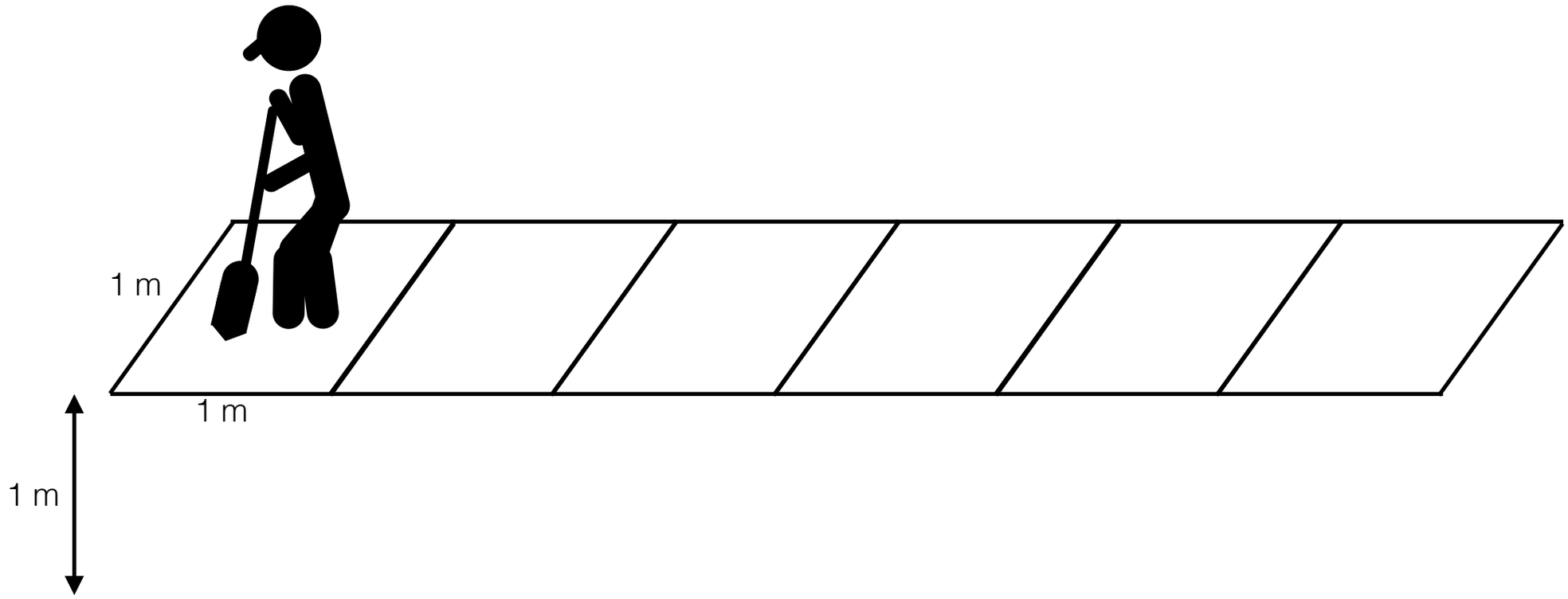
~~1 bebê / 1 mês?!~~

9 bebês / 9 meses

Cavar uma vala

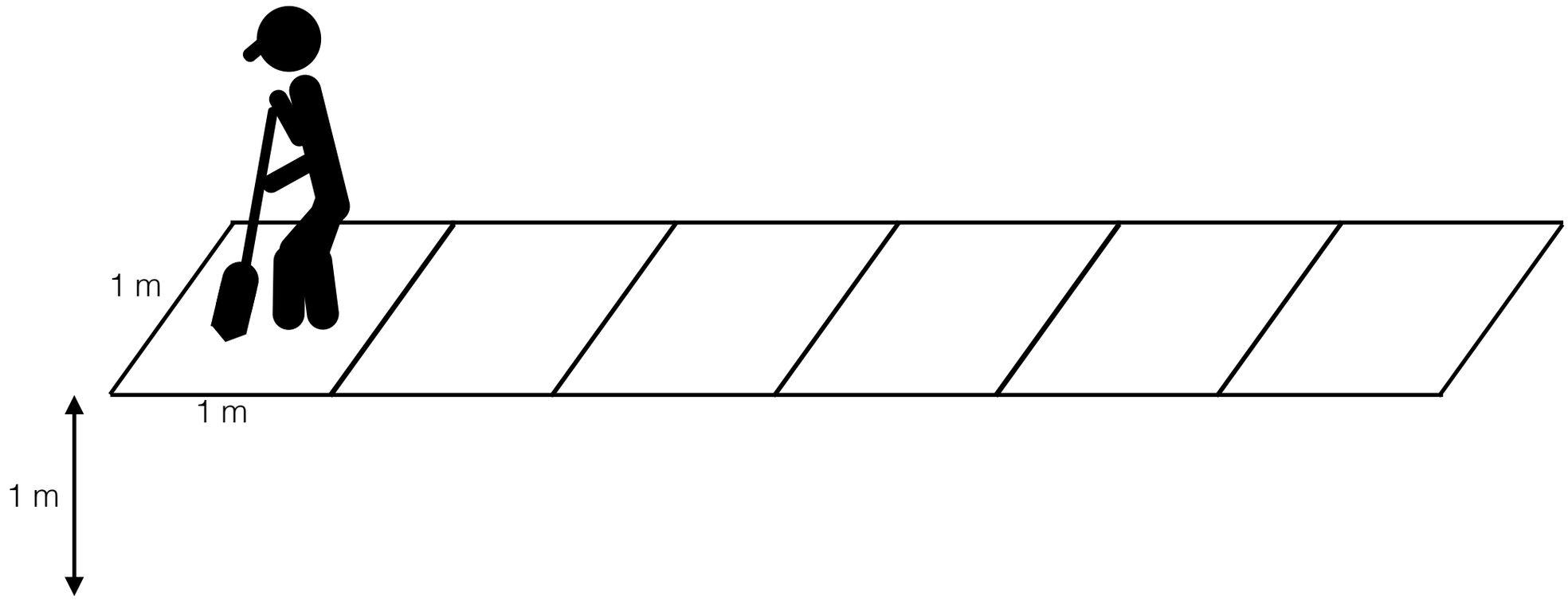


Cavar uma vala



1 trabalhador leva 1 hora para cavar 1 m^3

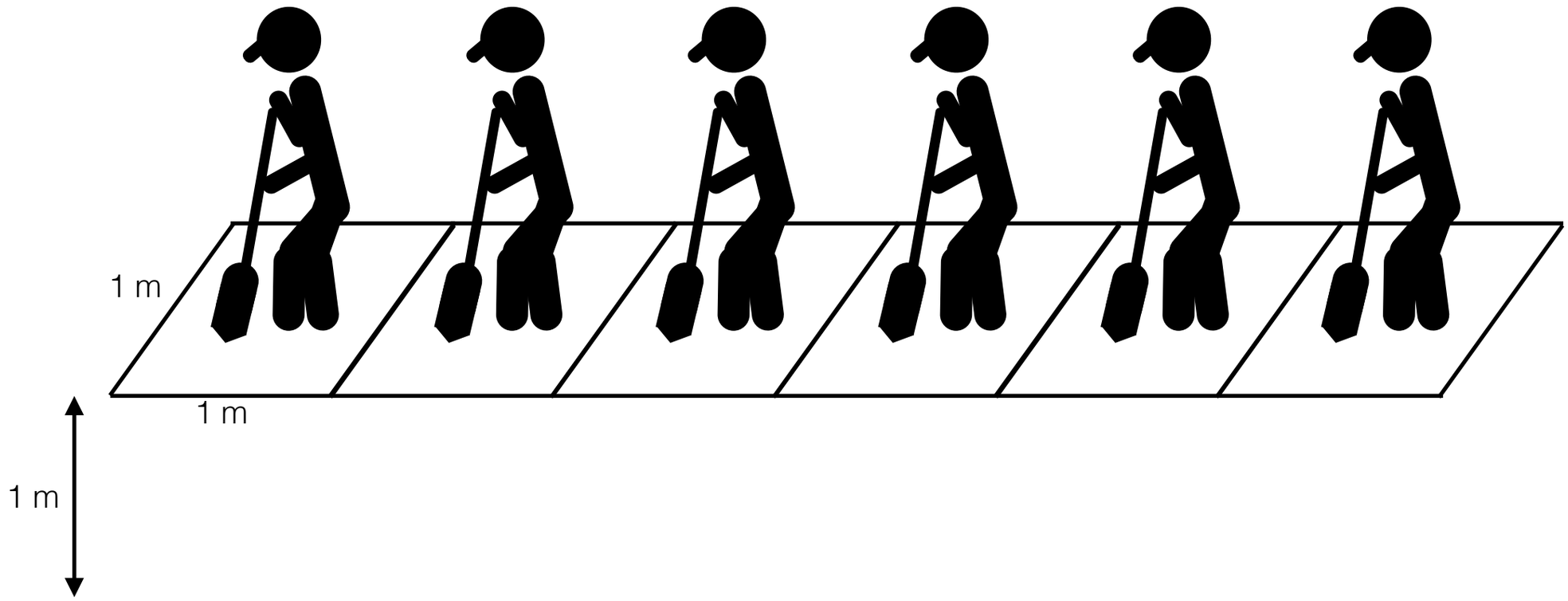
Cavar uma vala



1 trabalhador leva 1 hora para cavar 1 m^3

1 trabalhador leva 6 horas para cavar 6 m^3 (a vala toda)

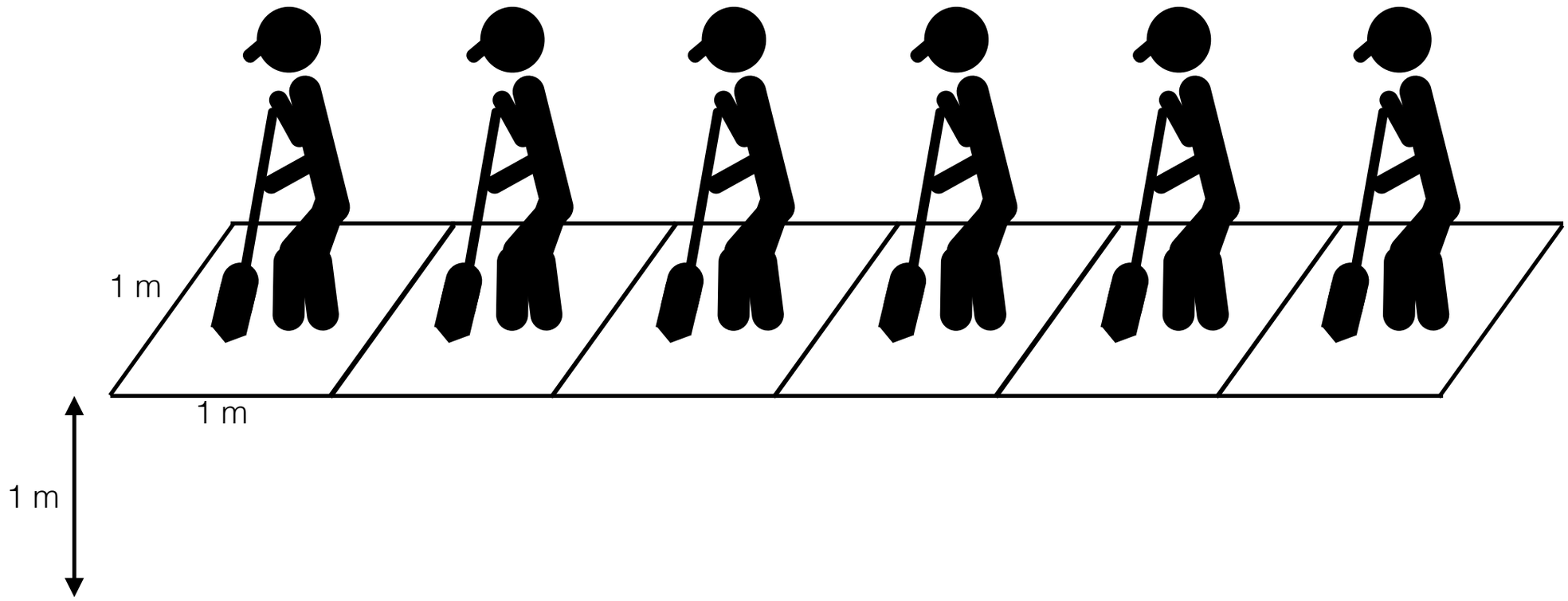
Cavar uma vala



1 trabalhador leva 1 hora para cavar 1 m^3

1 trabalhador leva 6 horas para cavar 6 m^3 (a vala toda)

Cavar uma vala

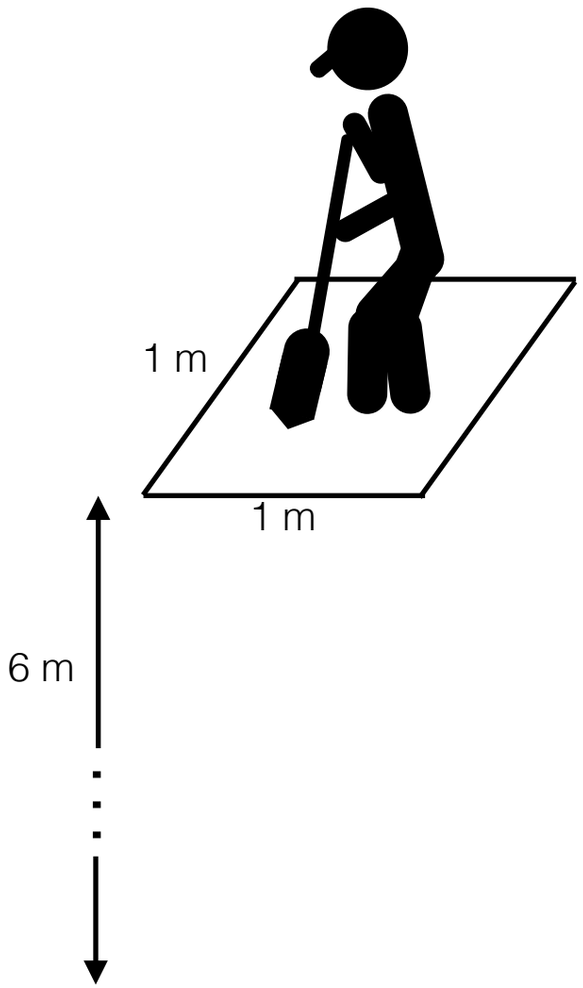


1 trabalhador leva 1 hora para cavar 1 m^3

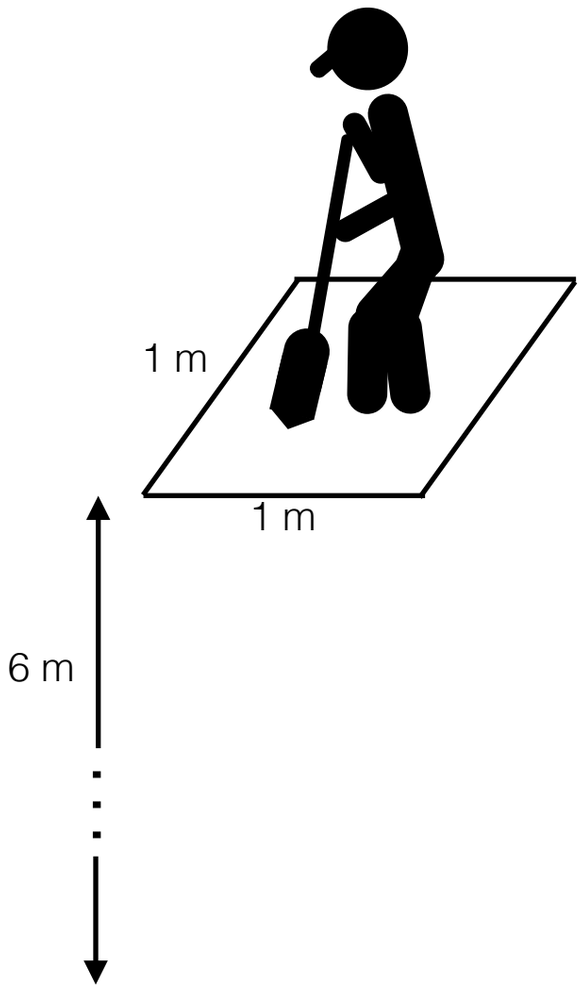
1 trabalhador leva 6 horas para cavar 6 m^3 (a vala toda)

6 trabalhadores levam 1 hora para cavar 6 m^3 (a vala toda)

Cavar um poço

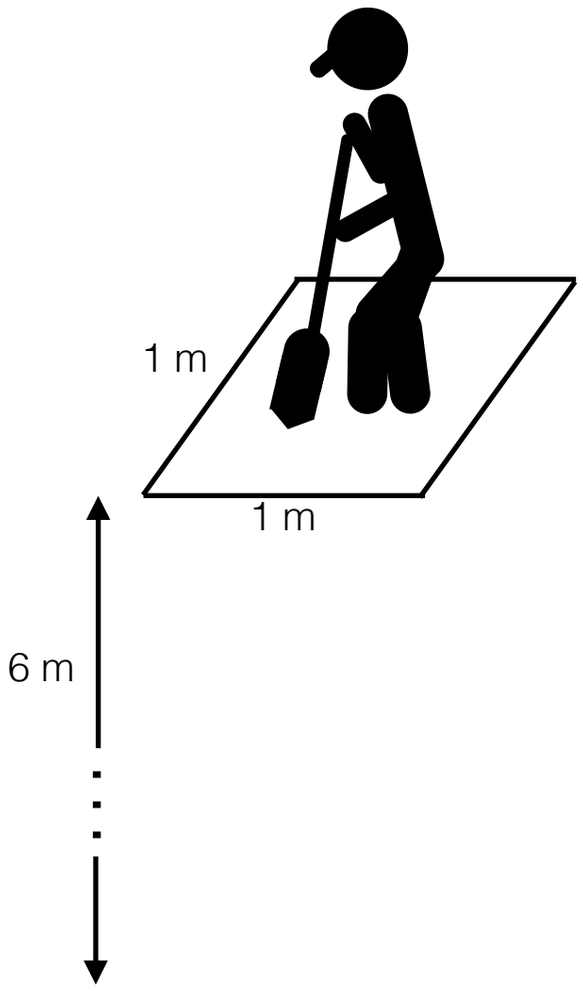


Cavar um poço



1 trabalhador leva 1 hora para cavar 1 m^3

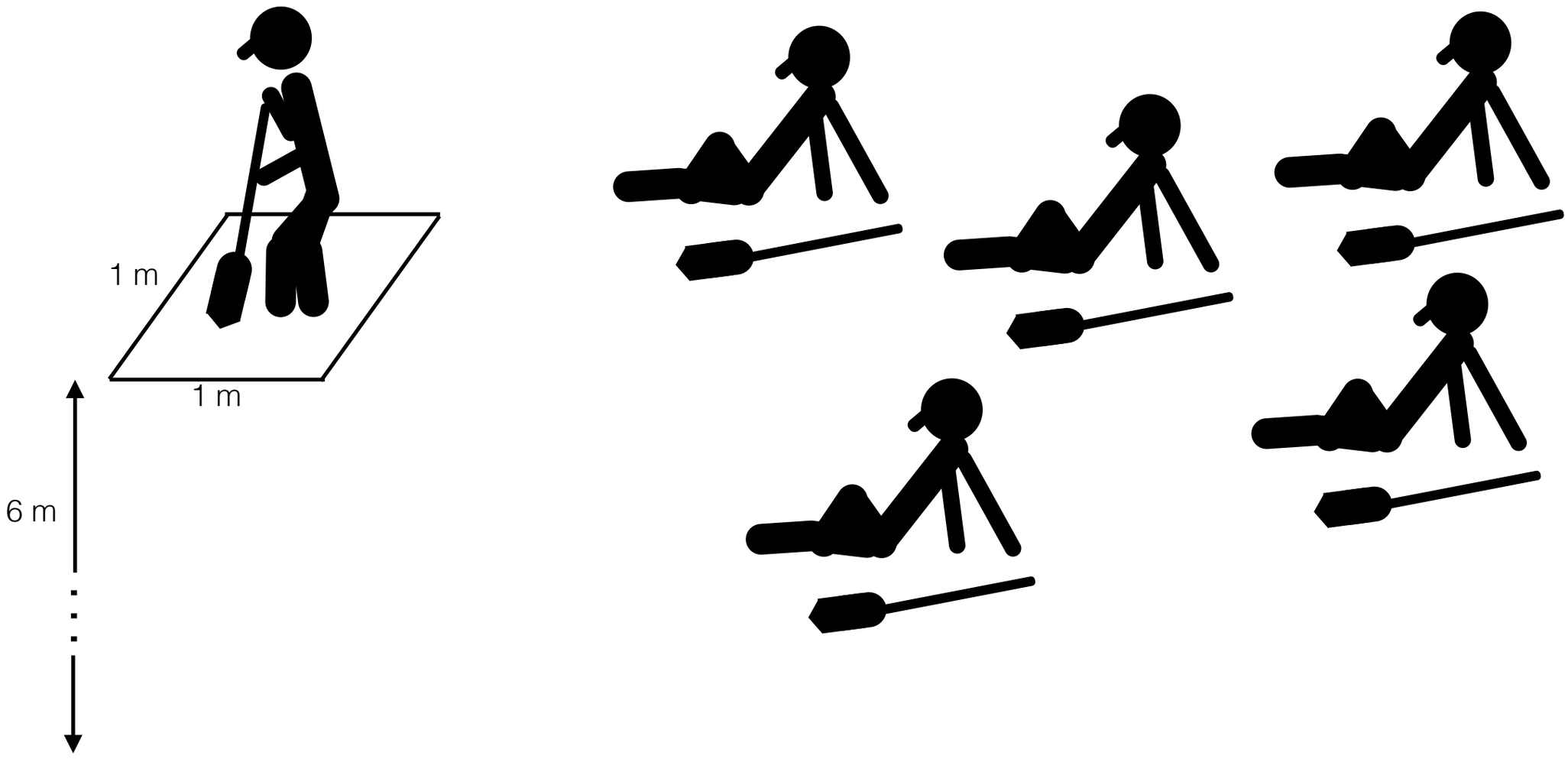
Cavar um poço



1 trabalhador leva 1 hora para cavar 1 m^3

1 trabalhador leva 6 horas para cavar 6 m^3 (o poço todo)

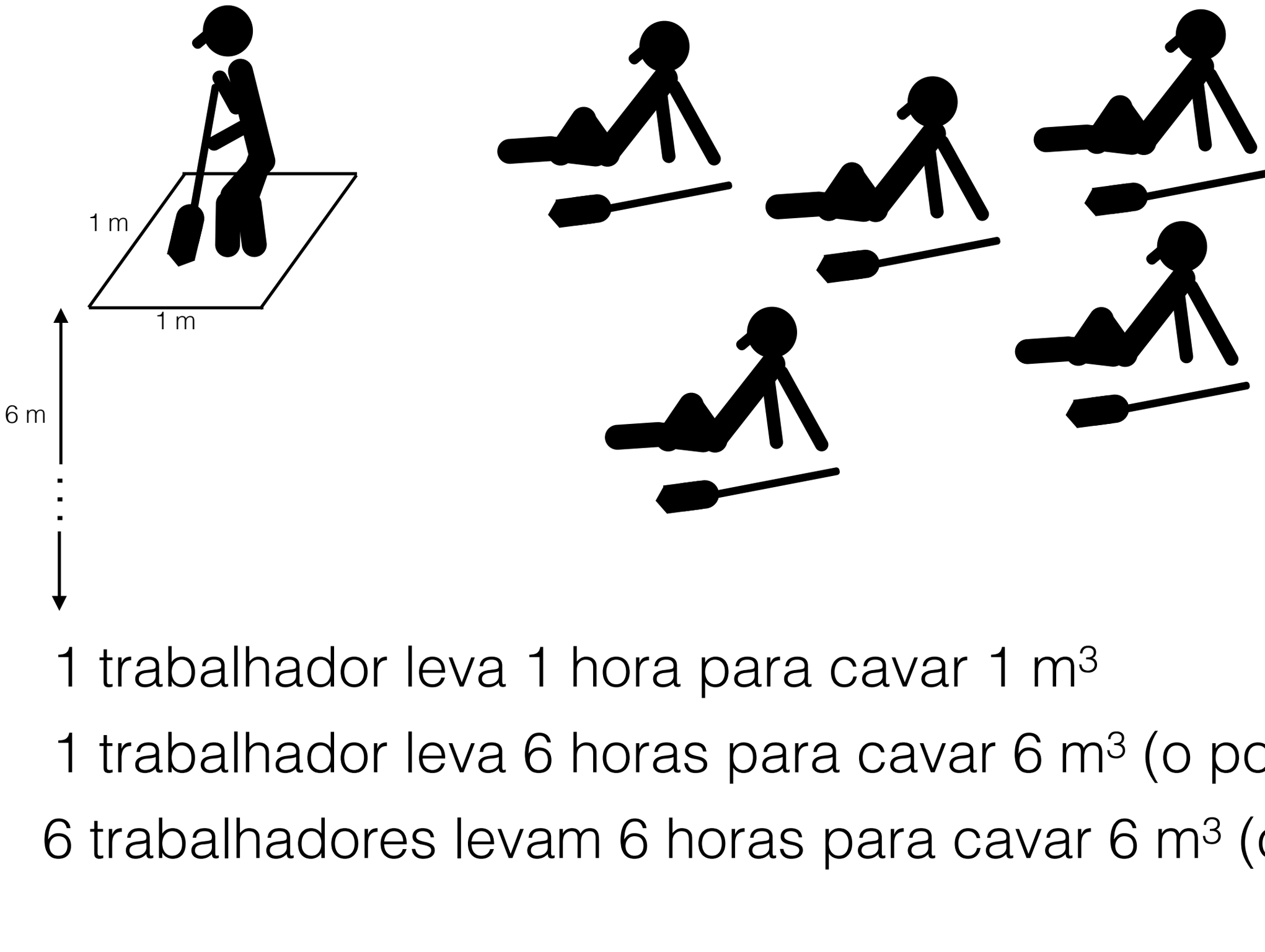
Cavar um poço



1 trabalhador leva 1 hora para cavar 1 m^3

1 trabalhador leva 6 horas para cavar 6 m^3 (o poço todo)

Cavar um poço



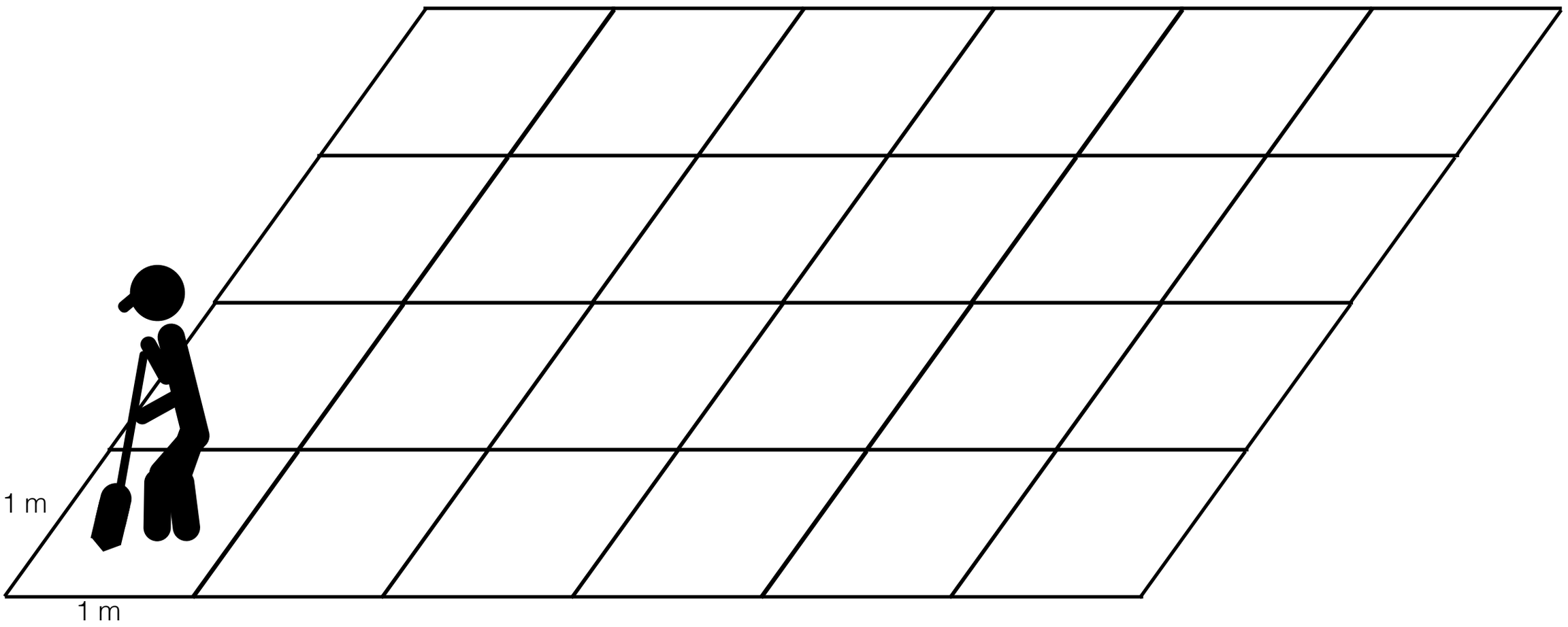
1 trabalhador leva 1 hora para cavar 1 m^3

1 trabalhador leva 6 horas para cavar 6 m^3 (o poço todo)

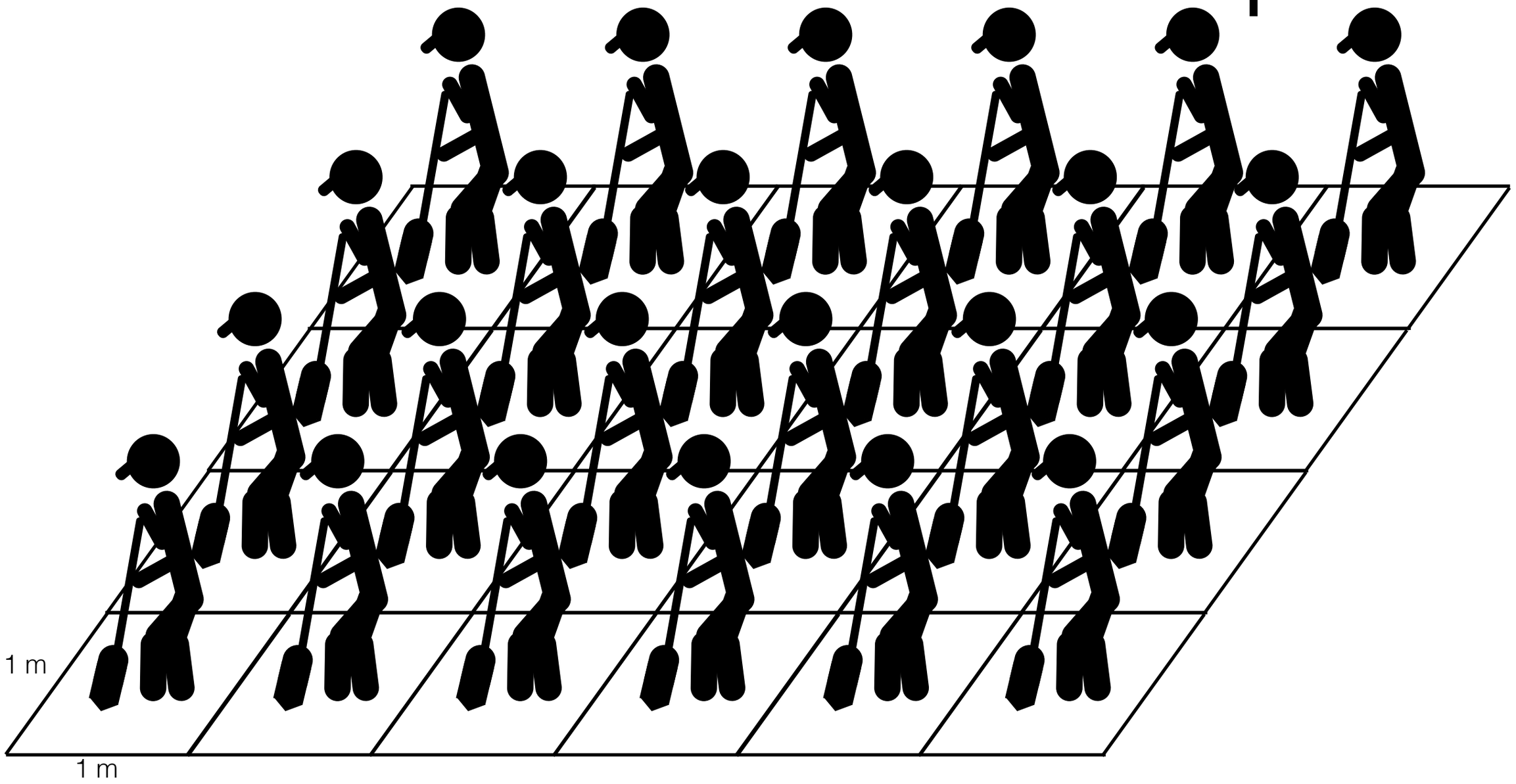
6 trabalhadores levam 6 horas para cavar 6 m^3 (o poço todo)


Gerente de Recursos Humanos
Gerente de Marketing
Gerente de Logistica
Gerente de Segurança
Gerente de Recursos Humanos
Gerente de Comunicação
Gerente do Projeto
Supervisor Interno
Gerente de Produção
Gerente de Desenvolvimento de Produtos
Zé

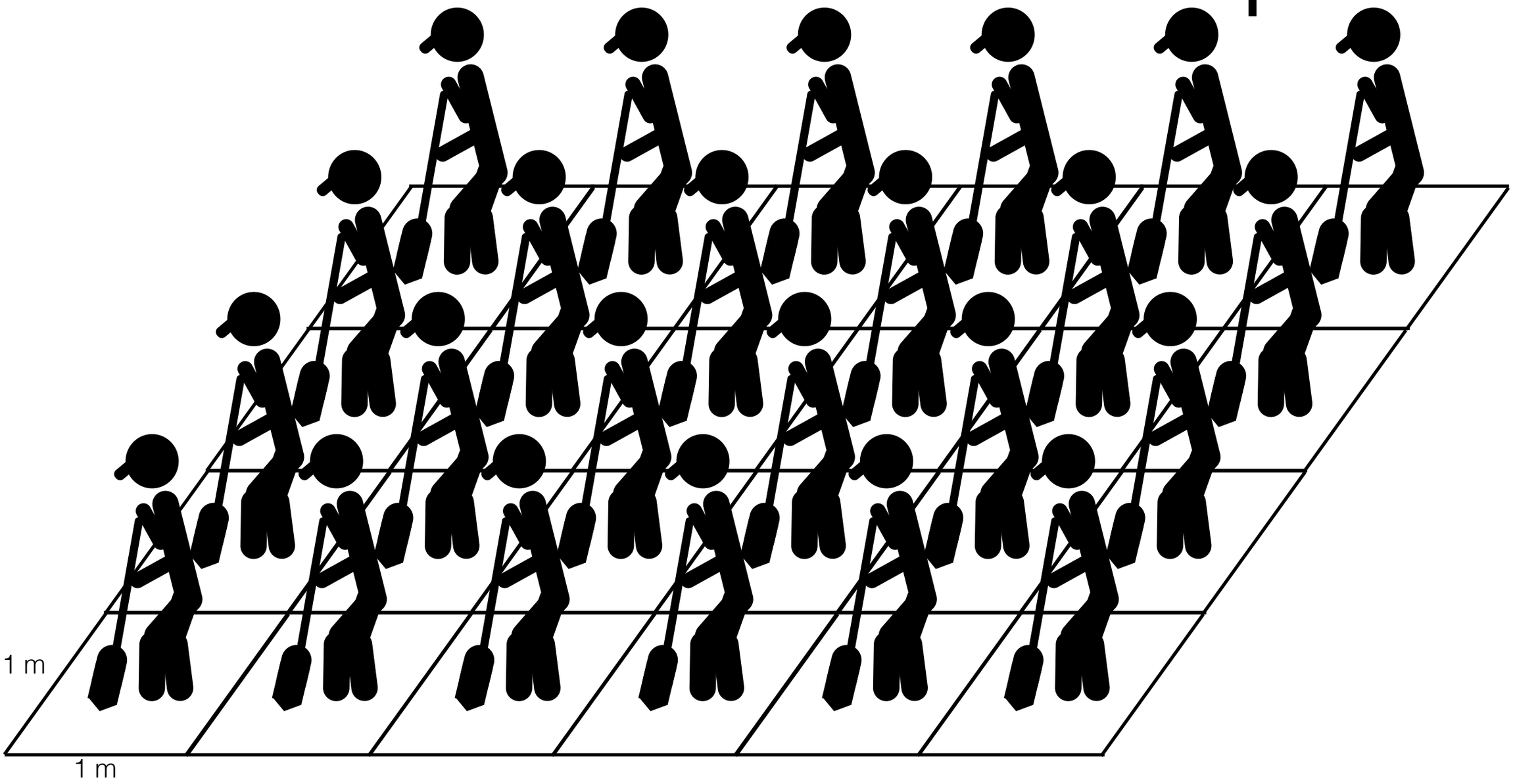
Cavar uma piscina



Cavar uma piscina

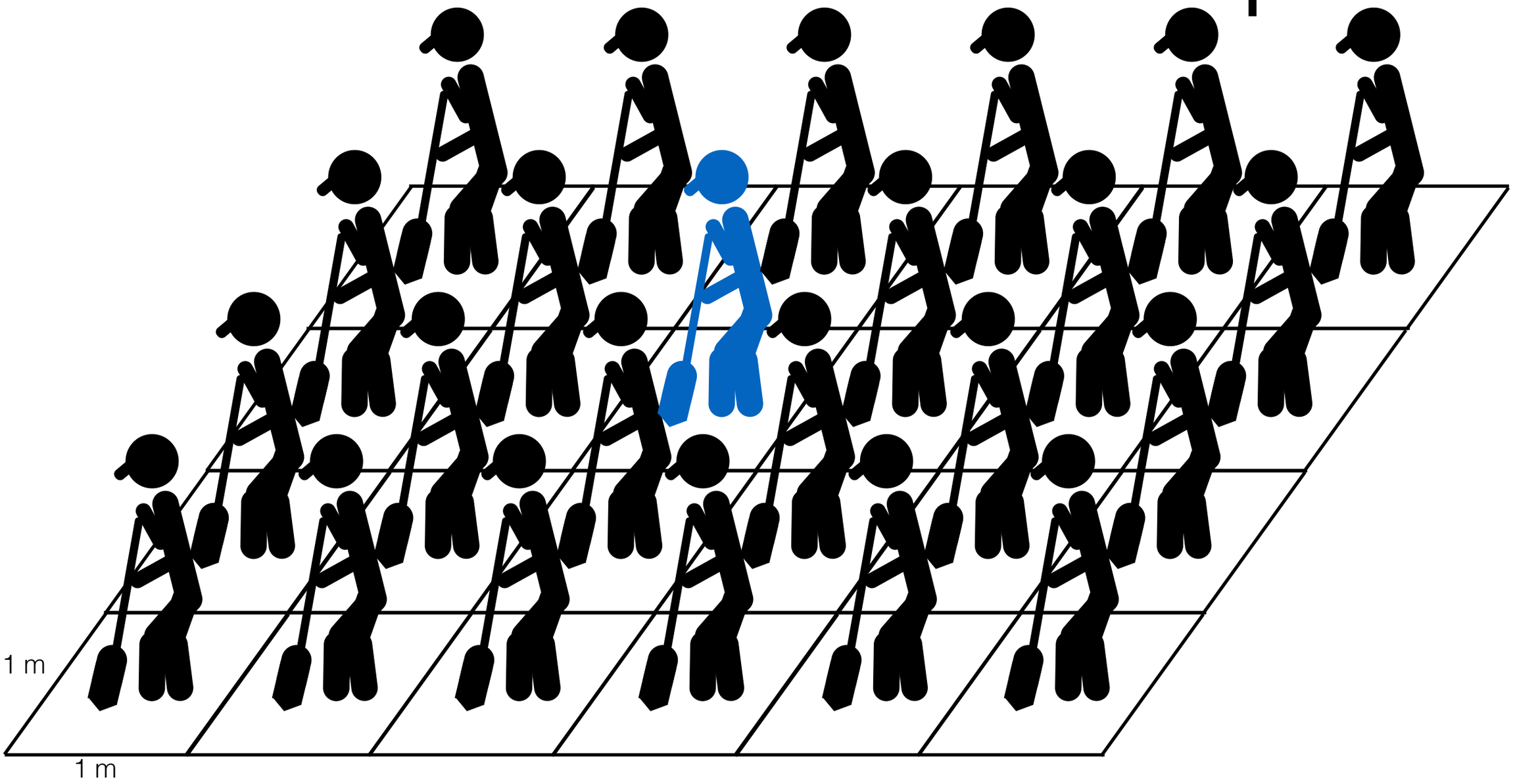


Cavar uma piscina



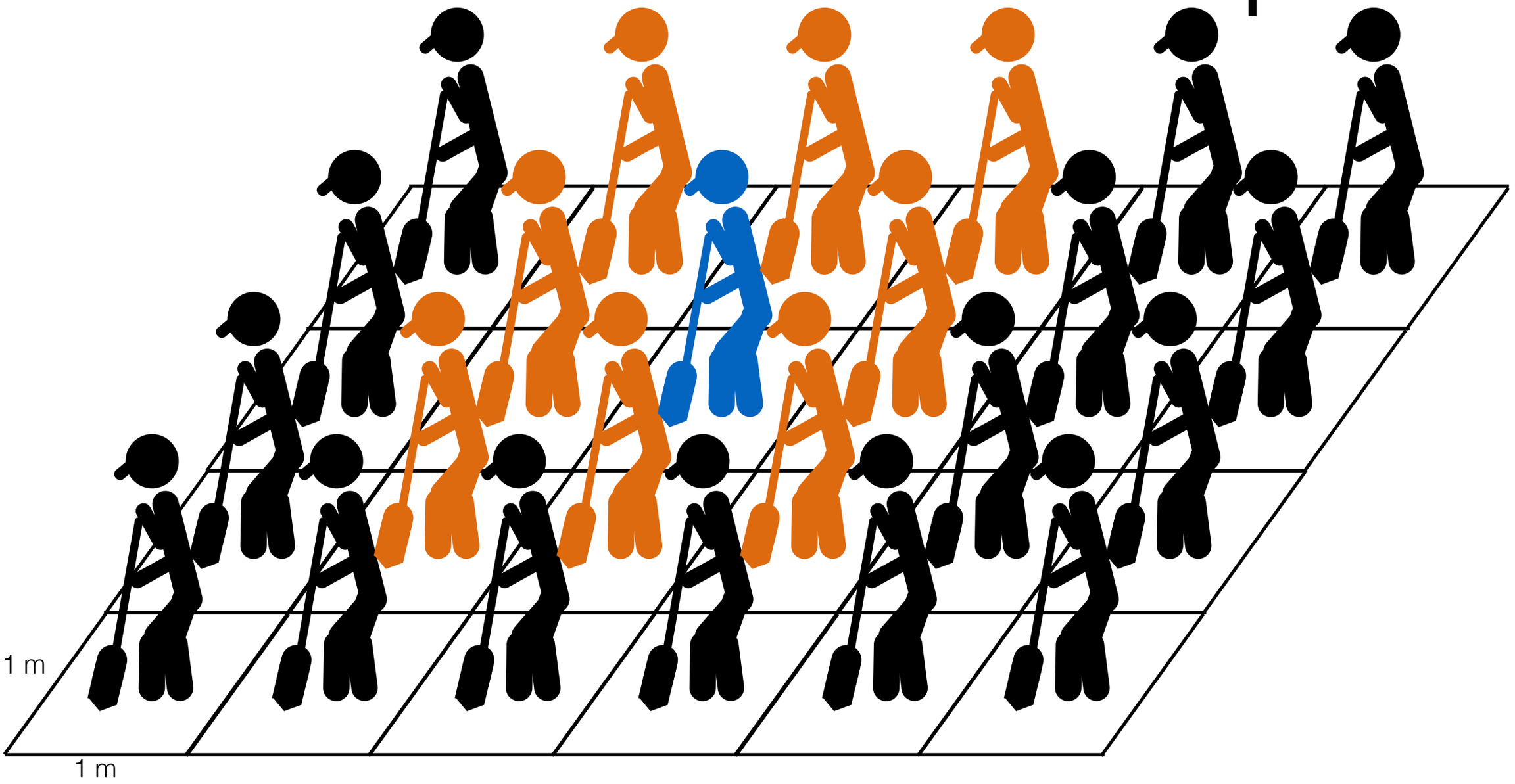
o tempo necessário para cavar toda a piscina vai depender do modo como os trabalhadores jogam a terra cavada nas células vizinhas.

Cavar uma piscina



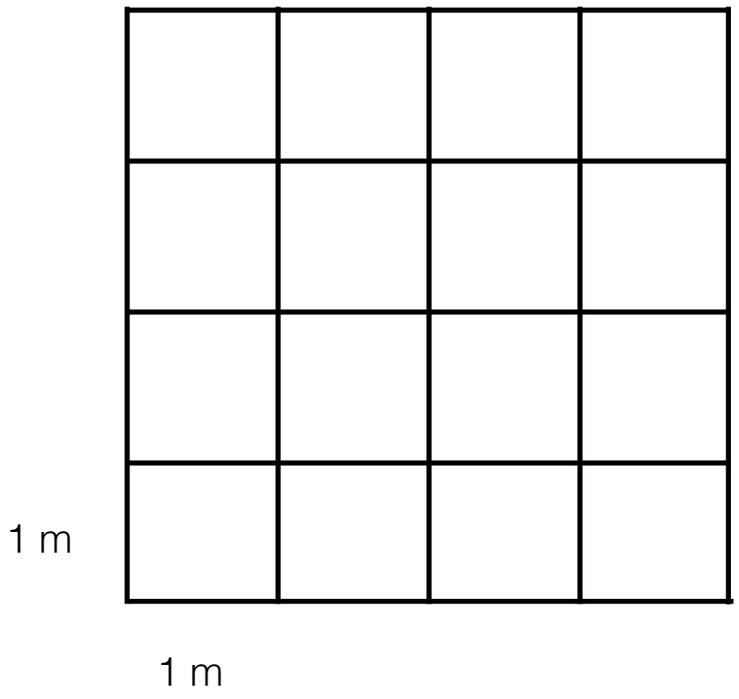
o tempo necessário para cavar toda a piscina vai depender do modo como os trabalhadores jogam a terra cavada nas células vizinhas.

Cavar uma piscina



o tempo necessário para cavar toda a piscina vai depender do modo como os trabalhadores jogam a terra cavada nas células vizinhas.

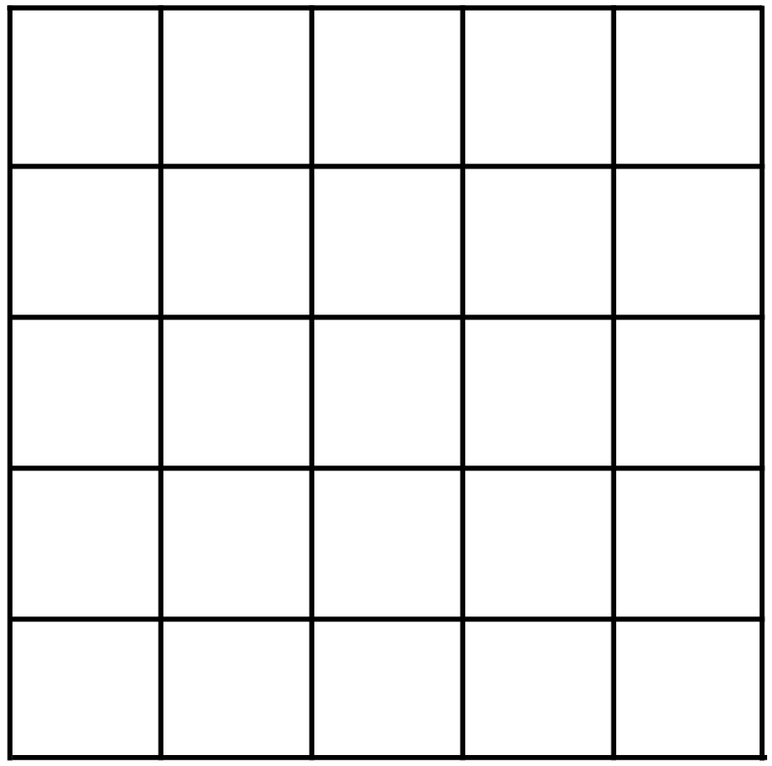
Qual é o tempo mínimo necessário para cavar toda a piscina com 1 m de profundidade?



*4x4 m^2

*16 trabalhadores

*1 trabalhador
cava 1 m^3/h



*5x5 m^2

*25 trabalhadores

*1 trabalhador
cava 1 m^3/h

Tipos básicos de programas paralelos

- Paralelismo de tarefas: *as tarefas são divididas entre os núcleos para resolver um certo problema.*

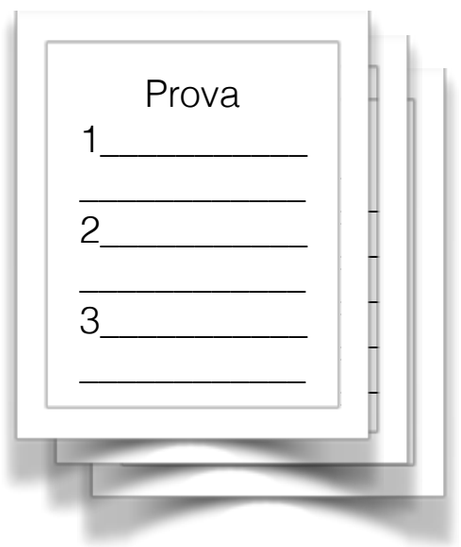
- Paralelismo de dados: *os dados são divididos entre os núcleos para resolver um certo problema.*

Paralelismo de Tarefas x Paralelismo de Dados

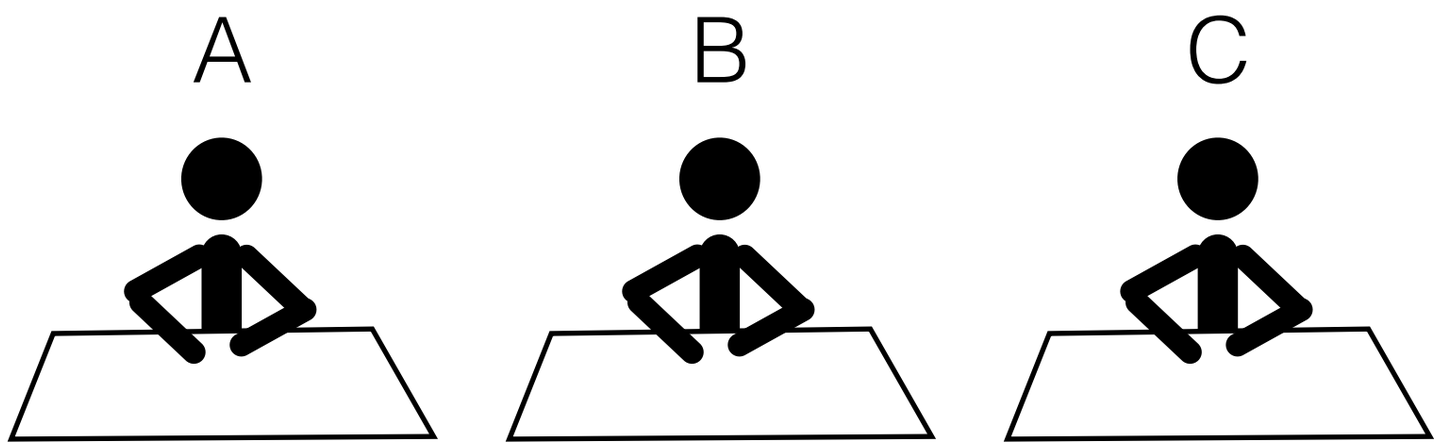
Paralelismo de Tarefas x Paralelismo de Dados

90 provas

Paralelismo de Tarefas x Paralelismo de Dados



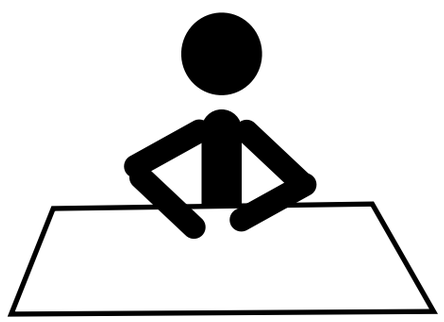
90 provas



Paralelismo de Tarefas x Paralelismo de Dados

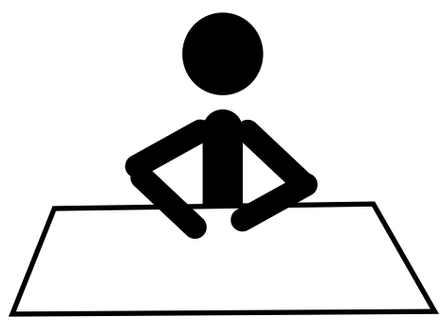
90 provas

A



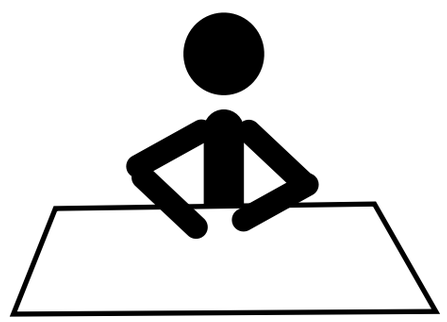
30 provas
3 questões

B



30 provas
3 questões

C

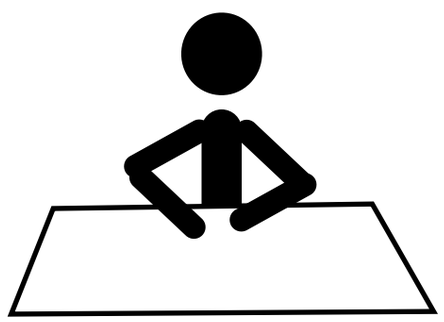


30 provas
3 questões

Paralelismo de Tarefas x Paralelismo de Dados

90 provas

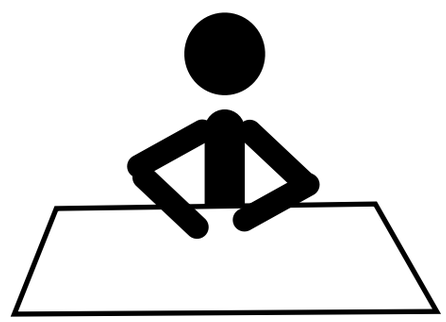
A



30 provas
3 questões

90 provas
questão nº1

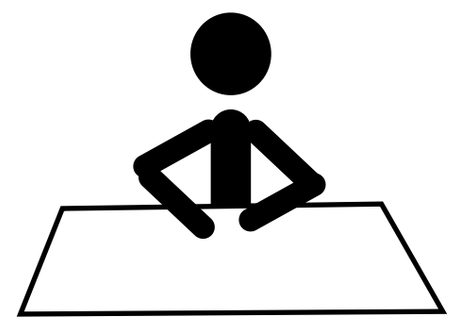
B



30 provas
3 questões

90 provas
questão nº2

C



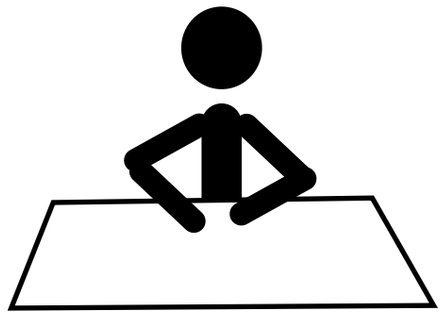
30 provas
3 questões

90 provas
questão nº3

Paralelismo de Tarefas x Paralelismo de Dados

90 provas

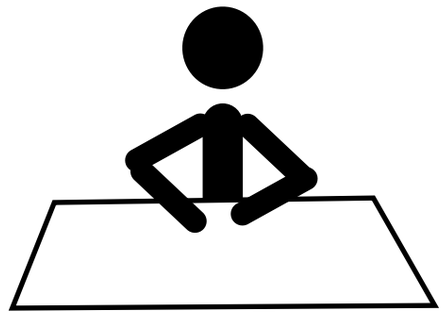
A



30 provas
3 questões

90 provas
questão nº1

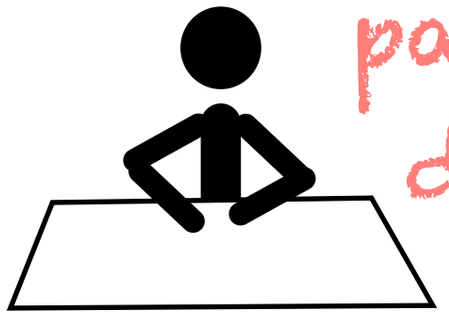
B



30 provas
3 questões

90 provas
questão nº2

C



30 provas
3 questões

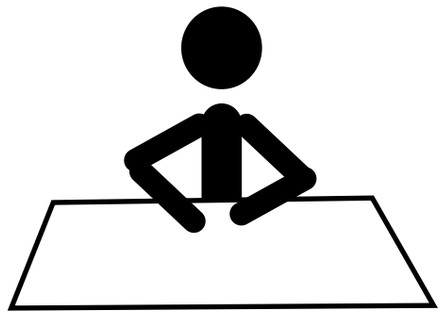
90 provas
questão nº3

paralelismo de dados

Paralelismo de Tarefas x Paralelismo de Dados

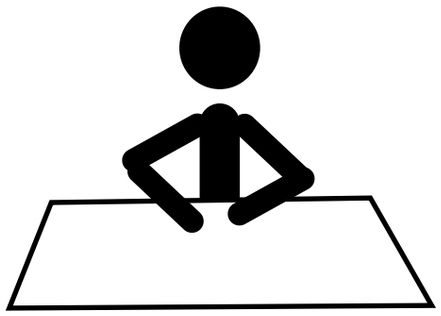
90 provas

A



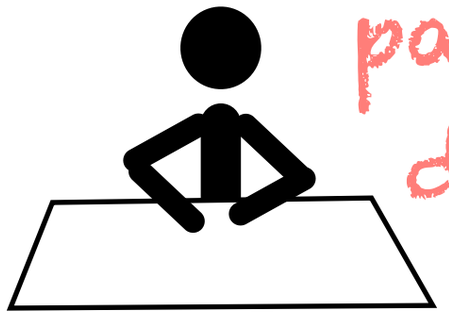
30 provas
3 questões

B



30 provas
3 questões

C



30 provas
3 questões

paralelismo de dados

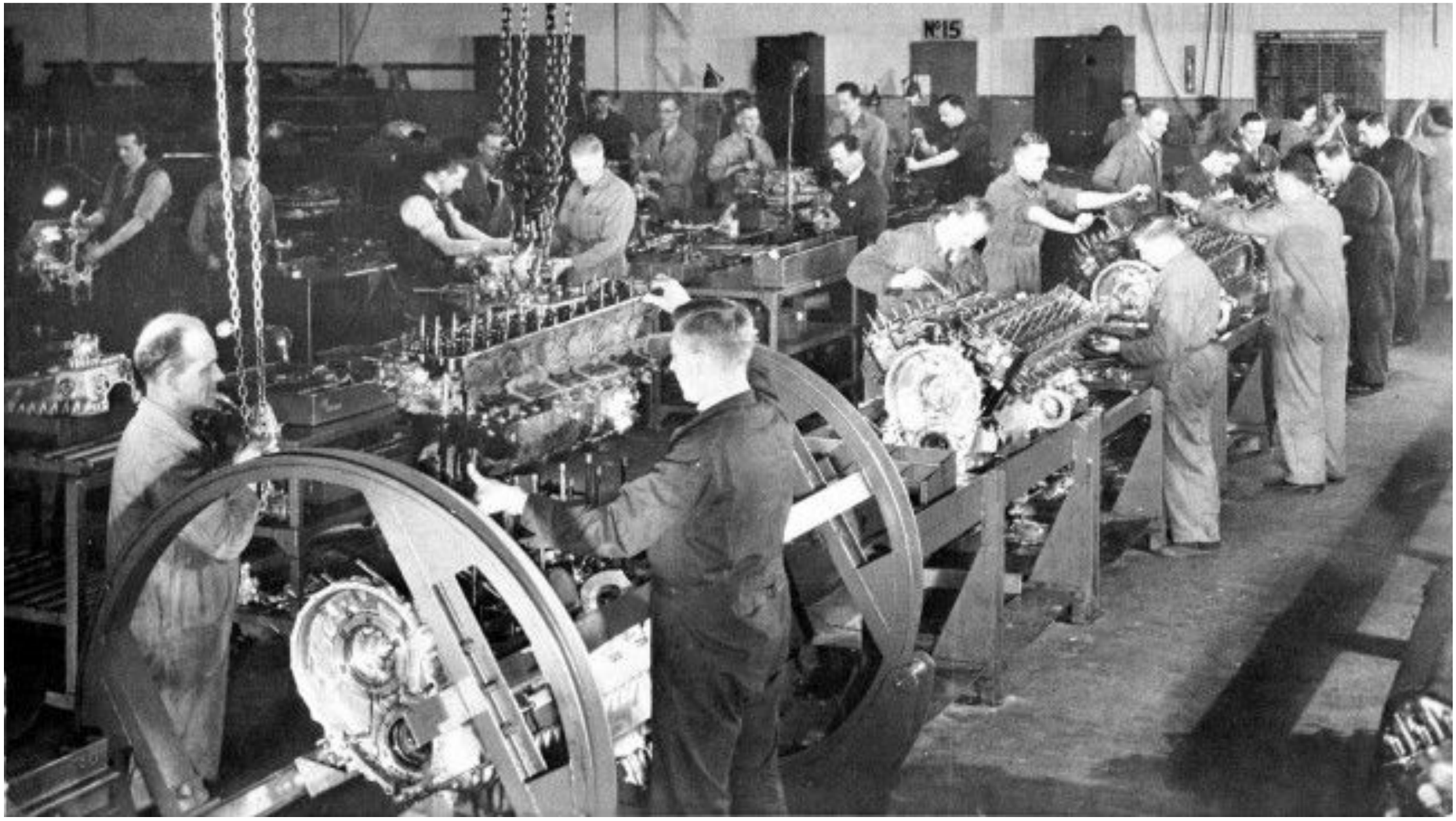
90 provas
questão nº1

90 provas
questão nº2

90 provas
questão nº3

paralelismo de tarefas

Fordismo



Como paralelizar?

```
//fragmento de código
int f=0,i;

for (i=0;i<1000;i++){
     f += i;
}
```

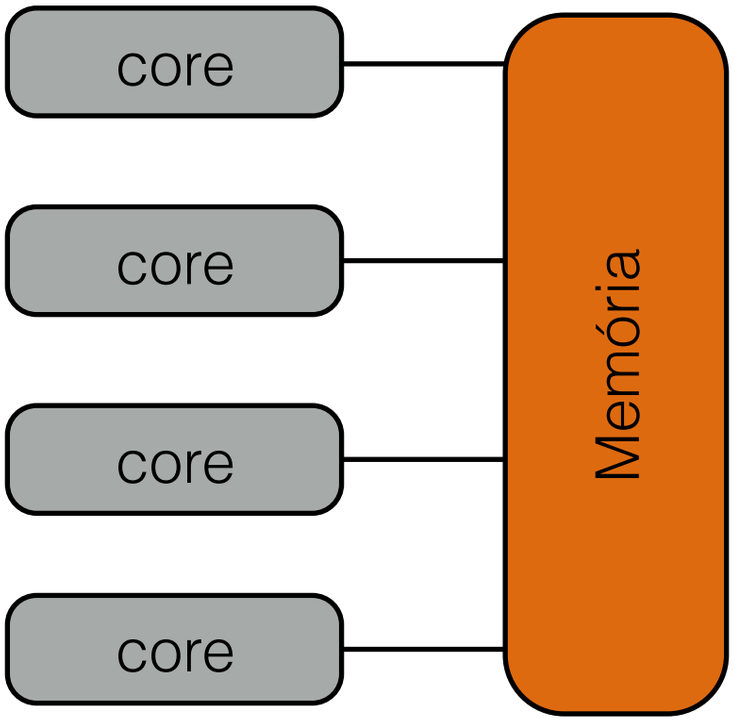
Como paralelizar?

```
//fragmento de código
int k[1000],i;


k[0]=1;
k[1]=1;

for (i=2;i<1000;i++){
     k[i]=k[i-1]+k[i-2];
}
```

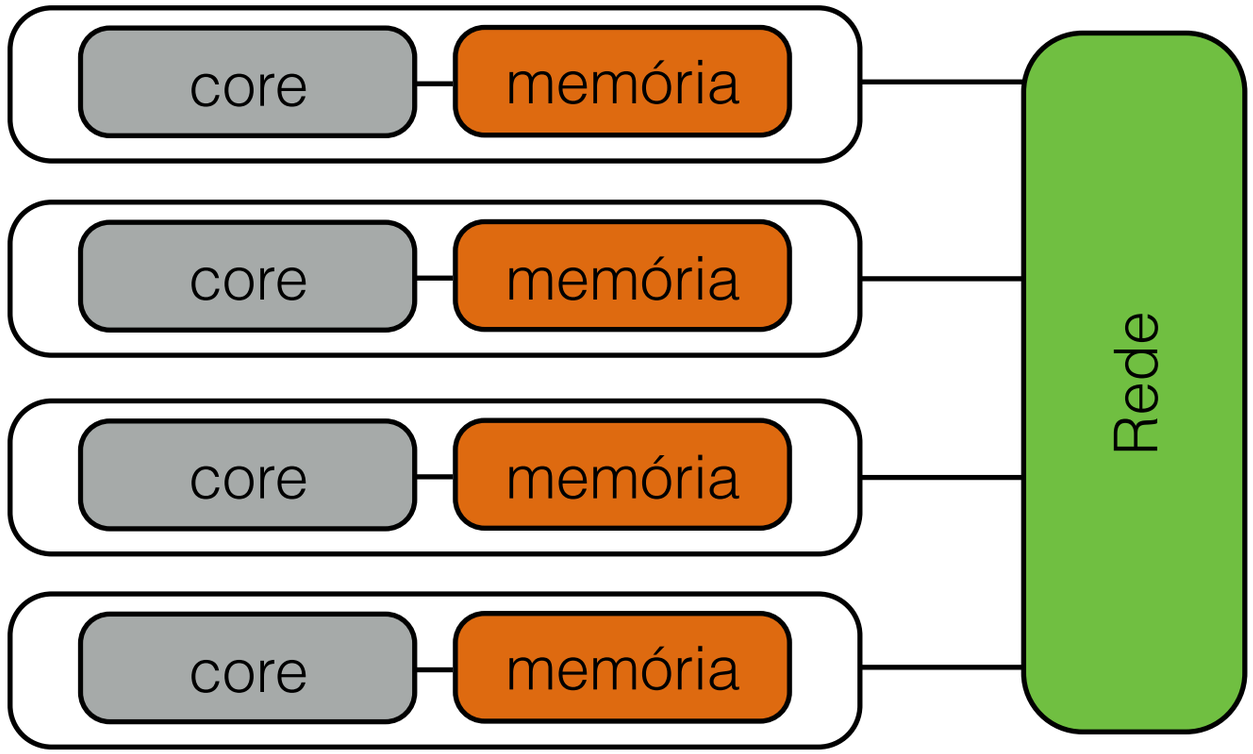
Como paralelizar?

```
//fragmento de código

float ka[1000], kb[1000];
int i,t;

// ka inicializado por aqui!!!

for (t=0;t<100;t++){
    for (i=1;i<999;i++){
        kb[i]=ka[i-1] - 2*ka[i] + ka[i+1];
    }
    for (i=1;i<999;i++){
        ka[i]=kb[i];
    }
}
```

Tipos de paralelismo

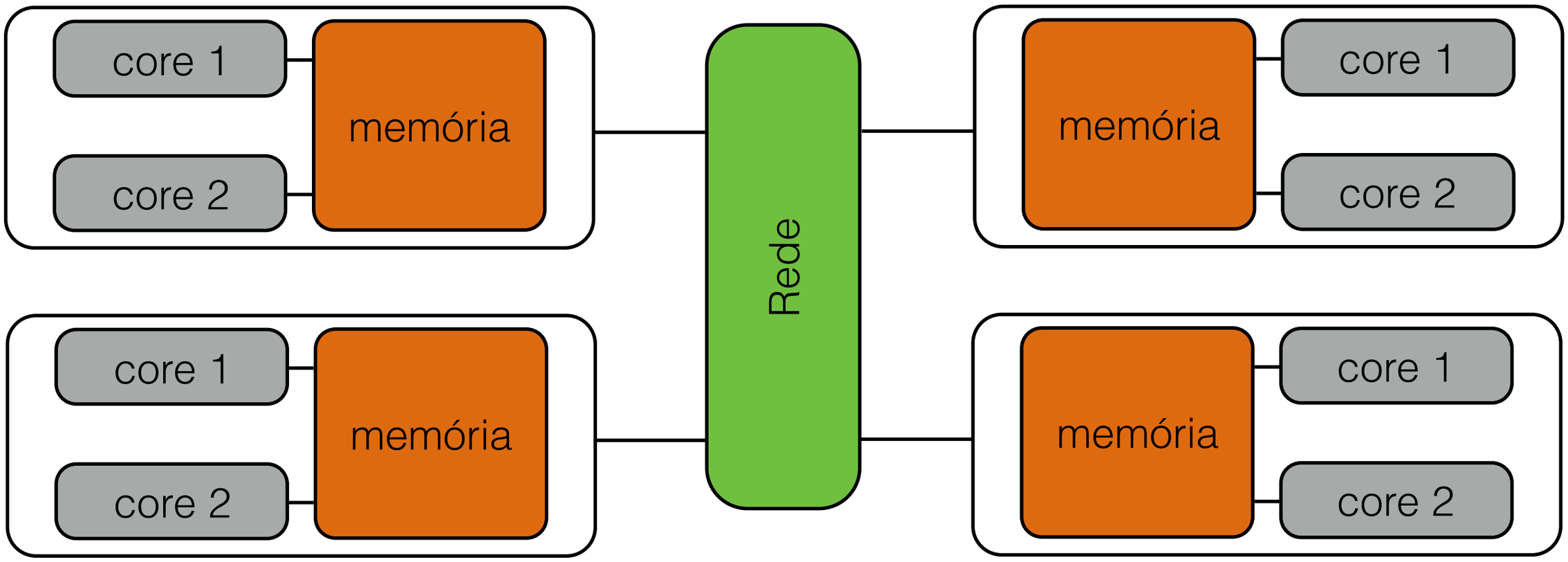


Memória compartilhada
(OpenMP)

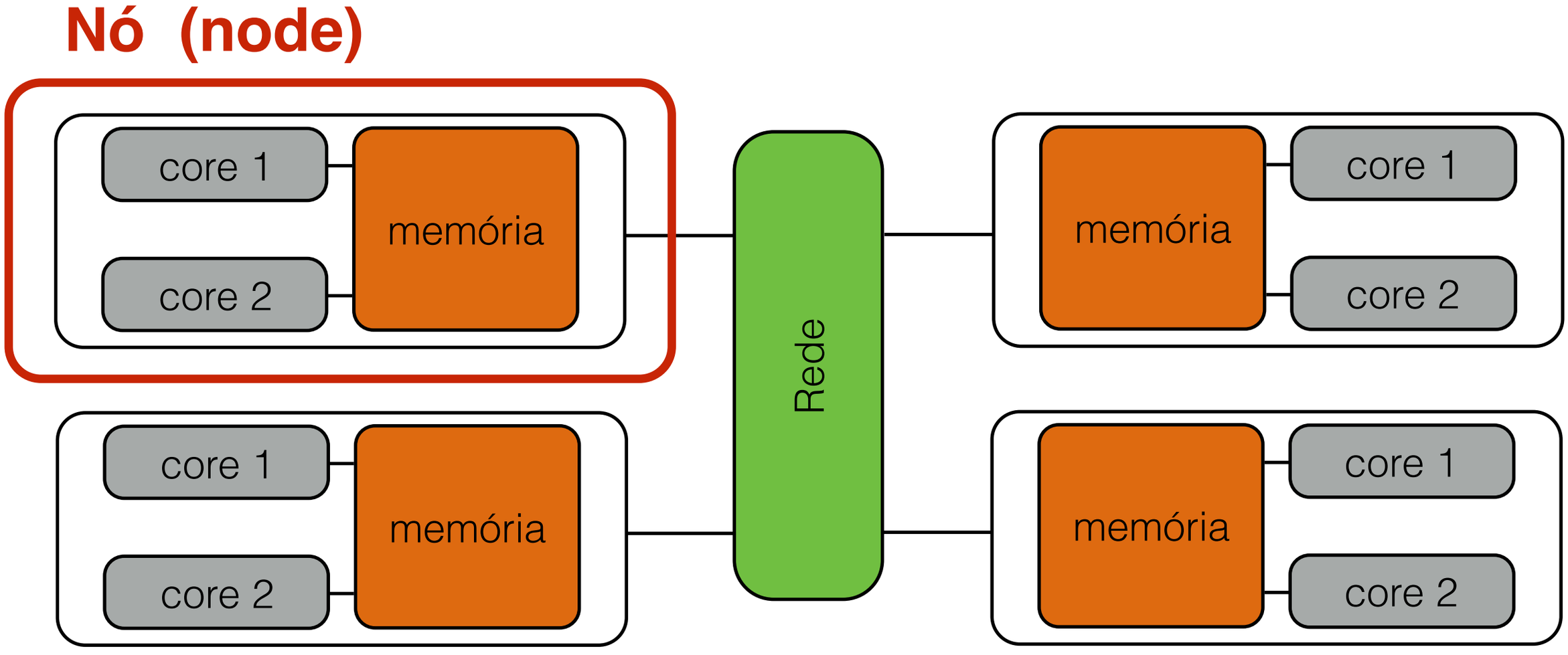


Memória distribuída
(MPI)

Tipos de paralelismo



Tipos de paralelismo



Exemplo

Exemplo

- Suponha que um programa execute 10^{12} instruções para resolver um problema particular. Suponha que rodando em um único processador o programa leve 10^6 s (cerca de 11.6 dias). Então, na média o programa executa 10^6 instruções por segundo.

Exemplo

- Suponha que um programa execute 10^{12} instruções para resolver um problema particular. Suponha que rodando em um único processador o programa leve 10^6 s (cerca de 11.6 dias). Então, na média o programa executa 10^6 instruções por segundo.

- Suponha que em um programa paralelizado em um sistema de memória distribuída em p processadores, cada processador vai fazer $10^{12}/p$ operações e deve mandar $10^9(p\text{-}1)$ mensagens.

Exemplo

Exemplo

- Suponha que leve 10^{-9} segundos para enviar uma mensagem. Quanto tempo vai levar para rodar o programa em 1000 processadores?

Exemplo

- Suponha que leve 10^{-9} segundos para enviar uma mensagem. Quanto tempo vai levar para rodar o programa em 1000 processadores?

- Suponha que leve 10^{-3} segundos para enviar uma mensagem. Quanto tempo vai levar para rodar o programa em 1000 processadores?

Minimize as mensagens!!!

Talk Less. Work More.

Speedup e Eficiência

Speedup

Speedup

$$S = \frac{T_{sequencial}}{T_{paralelo}}$$

Speedup

$$S = \frac{T_{sequencial}}{T_{paralelo}}$$

$T_{sequencial}$

Speedup

$$S = \frac{T_{sequencial}}{T_{paralelo}}$$

$T_{sequencial}$ Tempo de execução do programa sequencial.

Speedup

$$S = \frac{T_{sequencial}}{T_{paralelo}}$$

$T_{sequencial}$ Tempo de execução do programa sequencial.

$T_{paralelo}$

Speedup

$$S = \frac{T_{sequencial}}{T_{paralelo}}$$

$T_{sequencial}$ Tempo de execução do programa sequencial.

$T_{paralelo}$ Tempo de execução do programa paralelo.

Speedup

$$S = \frac{T_{sequencial}}{T_{paralelo}}$$

$T_{sequencial}$ Tempo de execução do programa sequencial.

$T_{paralelo}$ Tempo de execução do programa paralelo.

Para p processos, o valor mínimo possível do tempo de execução em paralelo é $T_{paralelo} = T_{sequencial}/p$.

Speedup

$$S = \frac{T_{sequencial}}{T_{paralelo}}$$

$T_{sequencial}$ Tempo de execução do programa sequencial.

$T_{paralelo}$ Tempo de execução do programa paralelo.

Para p processos, o valor mínimo possível do tempo de execução em paralelo é $T_{paralelo} = T_{sequencial}/p$.

Assim

Speedup

$$S = \frac{T_{sequencial}}{T_{paralelo}}$$

$T_{sequencial}$ Tempo de execução do programa sequencial.

$T_{paralelo}$ Tempo de execução do programa paralelo.

Para p processos, o valor mínimo possível do tempo de execução em paralelo é $T_{paralelo} = T_{sequencial}/p$.

Assim

$$S_{max} = \frac{T_{sequencial}}{T_{paralelo}} = p$$

Speedup

Speedup

Porém, na vida real, o speedup S torna-se uma fração cada vez menor de $S_{max} = p$.

Speedup

Porém, na vida real, o speedup S torna-se uma fração cada vez menor de $S_{max} = p$.

Uma outra forma de dizer isso é que a razão S/p será cada vez menor quando p cresce.

Speedup

Porém, na vida real, o speedup S torna-se uma fração cada vez menor de $S_{max} = p$.

Uma outra forma de dizer isso é que a razão S/p será cada vez menor quando p cresce.

Essa razão é comumente chamada de eficiência de programas paralelos. Substituindo na fórmula temos:

Speedup

Porém, na vida real, o speedup S torna-se uma fração cada vez menor de $S_{max} = p$.

Uma outra forma de dizer isso é que a razão S/p será cada vez menor quando p cresce.

Essa razão é comumente chamada de eficiência de programas paralelos. Substituindo na fórmula temos:

$$E = \frac{S}{p}$$

Speedup

Porém, na vida real, o speedup S torna-se uma fração cada vez menor de $S_{max} = p$.

Uma outra forma de dizer isso é que a razão S/p será cada vez menor quando p cresce.

Essa razão é comumente chamada de eficiência de programas paralelos. Substituindo na fórmula temos:

$$E = \frac{S}{p} = \frac{\frac{T_{sequencial}}{T_{paralelo}}}{p}$$

Speedup

Porém, na vida real, o speedup S torna-se uma fração cada vez menor de $S_{max} = p$.

Uma outra forma de dizer isso é que a razão S/p será cada vez menor quando p cresce.

Essa razão é comumente chamada de eficiência de programas paralelos. Substituindo na fórmula temos:

$$E = \frac{S}{p} = \frac{\frac{T_{sequencial}}{T_{paralelo}}}{p} = \frac{T_{sequencial}}{p \cdot T_{paralelo}}$$

Exemplo:
problema de *n*-corpos

Exemplo: problema de *n*-corpos

	número de proc.		
100 corpos	1	2	4

Exemplo: problema de *n*-corpos

	número de proc.		
100 corpos	1	2	4
Tempo (s)	14,10	7,50	4,83

Exemplo: problema de *n*-corpos

100 corpos	número de proc. 1	2	4
Tempo (s)	14,10	7,50	4,83
S	1,00	1,88	2,92

Exemplo: problema de *n*-corpos

100 corpos	número de proc. 1	2	4
Tempo (s)	14,10	7,50	4,83
S	1,00	1,88	2,92
E	1,00	0,94	0,73

Exemplo: problema de *n*-corpos

100 corpos	número de proc. 1	2	4
Tempo (s)	14,10	7,50	4,83
S	1,00	1,88	2,92
E	1,00	0,94	0,73

Exemplo:
problema de *n*-corpos

Exemplo: problema de *n*-corpos

	número de proc.		
200 corpos	1	2	4

Exemplo: problema de *n*-corpos

200 corpos	número de proc. 1	2	4
Tempo (s)	55,19	28,69	18,09

Exemplo: problema de *n*-corpos

200 corpos	número de proc. 1	2	4
Tempo (s)	55,19	28,69	18,09
S	1,00	1,92	3,05

Exemplo: problema de *n*-corpos

200 corpos	número de proc. 1	2	4
Tempo (s)	55,19	28,69	18,09
S	1,00	1,92	3,05
E	1,00	0,96	0,76

Exemplo: problema de *n*-corpos

200 corpos	número de proc. 1	2	4
Tempo (s)	55,19	28,69	18,09
S	1,00	1,92	3,05
E	1,00	0,96	0,76

Exemplo:
problema de *n*-corpos

Exemplo: problema de *n*-corpos

	número de proc.		
400 corpos	1	2	4

Exemplo: problema de *n*-corpos

400 corpos	número de proc. 1	2	4
Tempo (s)	222,71	114,73	64,59

Exemplo: problema de *n*-corpos

400 corpos	número de proc. 1	2	4
Tempo (s)	222,71	114,73	64,59
S	1,00	1,94	3,45

Exemplo: problema de *n*-corpos

400 corpos	número de proc. 1	2	4
Tempo (s)	222,71	114,73	64,59
S	1,00	1,94	3,45
E	1,00	0,97	0,86

Exemplo: problema de *n*-corpos

400 corpos	número de proc. 1	2	4
Tempo (s)	222,71	114,73	64,59
S	1,00	1,94	3,45
E	1,00	0,97	0,86

Exemplo: problema de *n*-corpos

Speedup

Eficiência

Exemplo: problema de *n*-corpos

Speedup Eficiência

	número de proc.		
corpos	1	2	4

Exemplo: problema de *n*-corpos

Speedup Eficiência

	número de proc.		
corpos	1	2	4
100	1,00	1,88	2,92

Exemplo: problema de *n*-corpos

Speedup

	número de proc.		
corpos	1	2	4
100	1,00	1,88	2,92
200	1,00	1,92	3,05

Eficiência

Exemplo: problema de *n*-corpos

Speedup

Eficiência

	número de proc.		
corpos	1	2	4
100	1,00	1,88	2,92
200	1,00	1,92	3,05
400	1,00	1,94	3,45

Exemplo: problema de *n*-corpos

Speedup

	número de proc.		
corpos	1	2	4
100	1,00	1,88	2,92
200	1,00	1,92	3,05
400	1,00	1,94	3,45

Eficiência

	número de proc.		

Exemplo: problema de *n*-corpos

Speedup

	número de proc.		
corpos	1	2	4
100	1,00	1,88	2,92
200	1,00	1,92	3,05
400	1,00	1,94	3,45

Eficiência

	número de proc.		
corpos	1	2	4

Exemplo: problema de *n*-corpos

Speedup

	número de proc.		
corpos	1	2	4
100	1,00	1,88	2,92
200	1,00	1,92	3,05
400	1,00	1,94	3,45

Eficiência

	número de proc.		
corpos	1	2	4
100	1,00	0,94	0,73

Exemplo: problema de *n*-corpos

Speedup

	número de proc.		
corpos	1	2	4
100	1,00	1,88	2,92
200	1,00	1,92	3,05
400	1,00	1,94	3,45

Eficiência

	número de proc.		
corpos	1	2	4
100	1,00	0,94	0,73
200	1,00	0,96	0,76

Exemplo: problema de *n*-corpos

Speedup

	número de proc.		
corpos	1	2	4
100	1,00	1,88	2,92
200	1,00	1,92	3,05
400	1,00	1,94	3,45

Eficiência

	número de proc.		
corpos	1	2	4
100	1,00	0,94	0,73
200	1,00	0,96	0,76
400	1,00	0,97	0,86

Exemplo: problema de *n*-corpos

Speedup

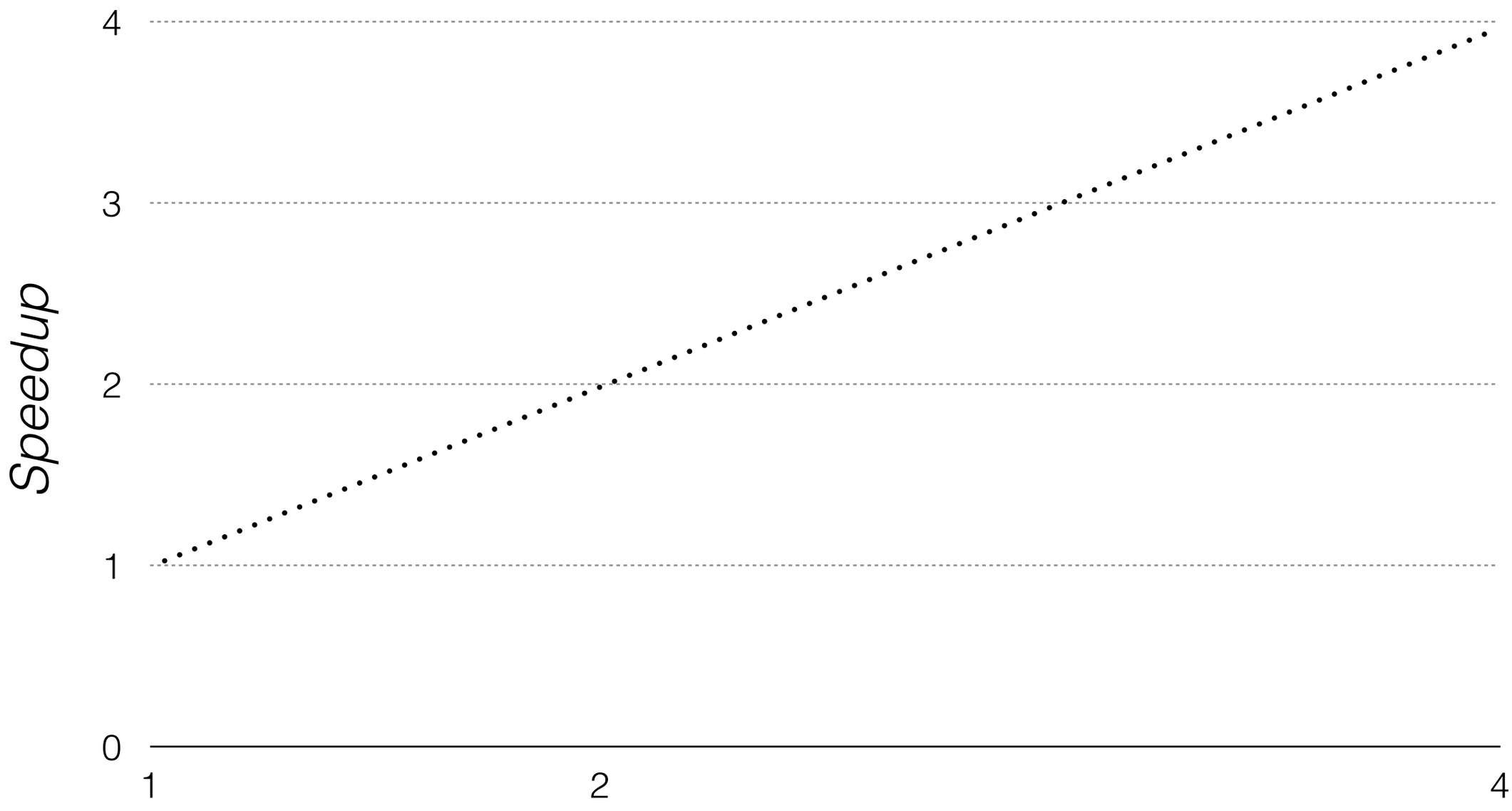
	número de proc.		
corpos	1	2	4
100	1,00	1,88	2,92
200	1,00	1,92	3,05
400	1,00	1,94	3,45

Eficiência

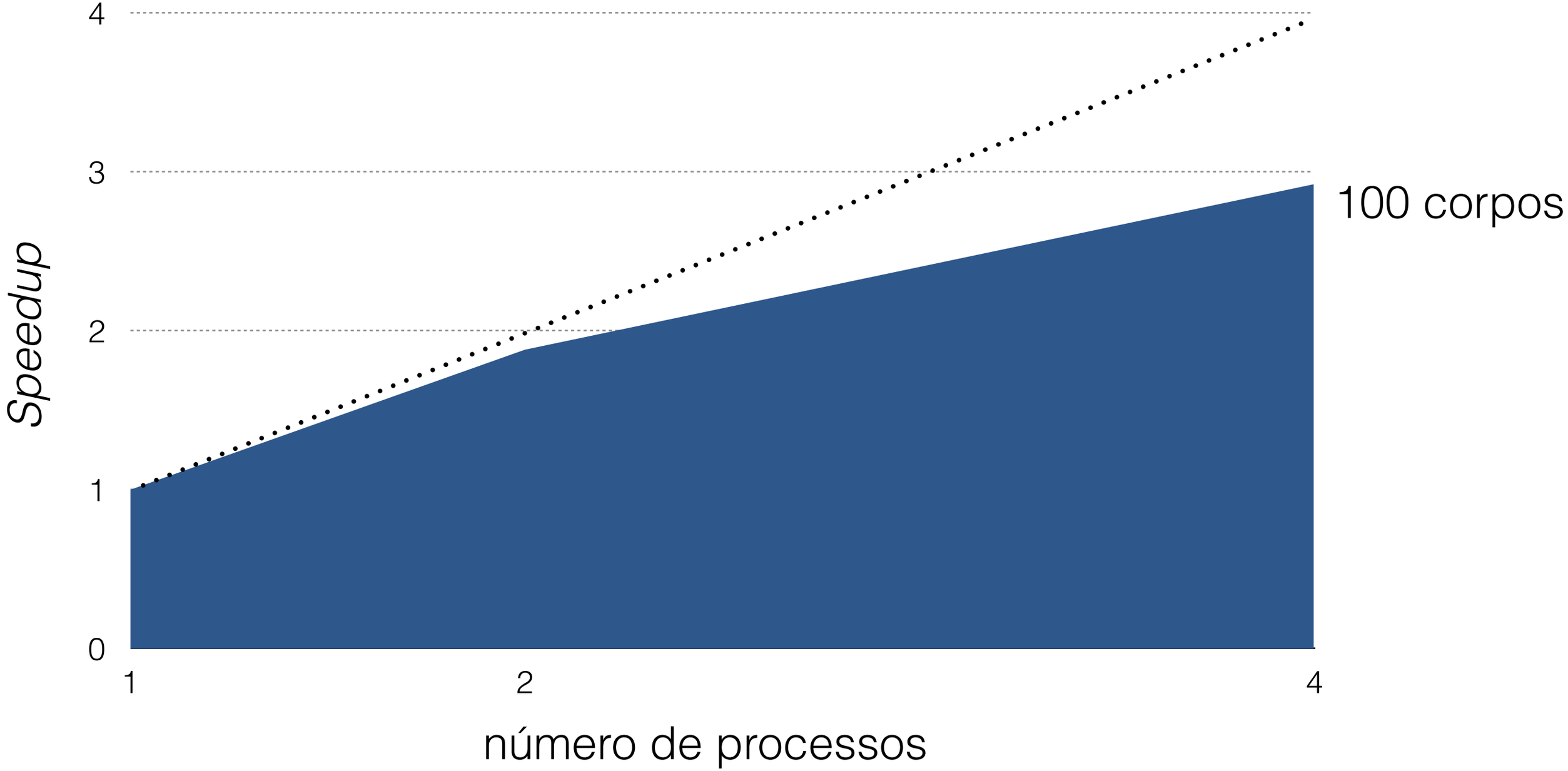
	número de proc.		
corpos	1	2	4
100	1,00	0,94	0,73
200	1,00	0,96	0,76
400	1,00	0,97	0,86

Exemplo:
problema de *n*-corpos

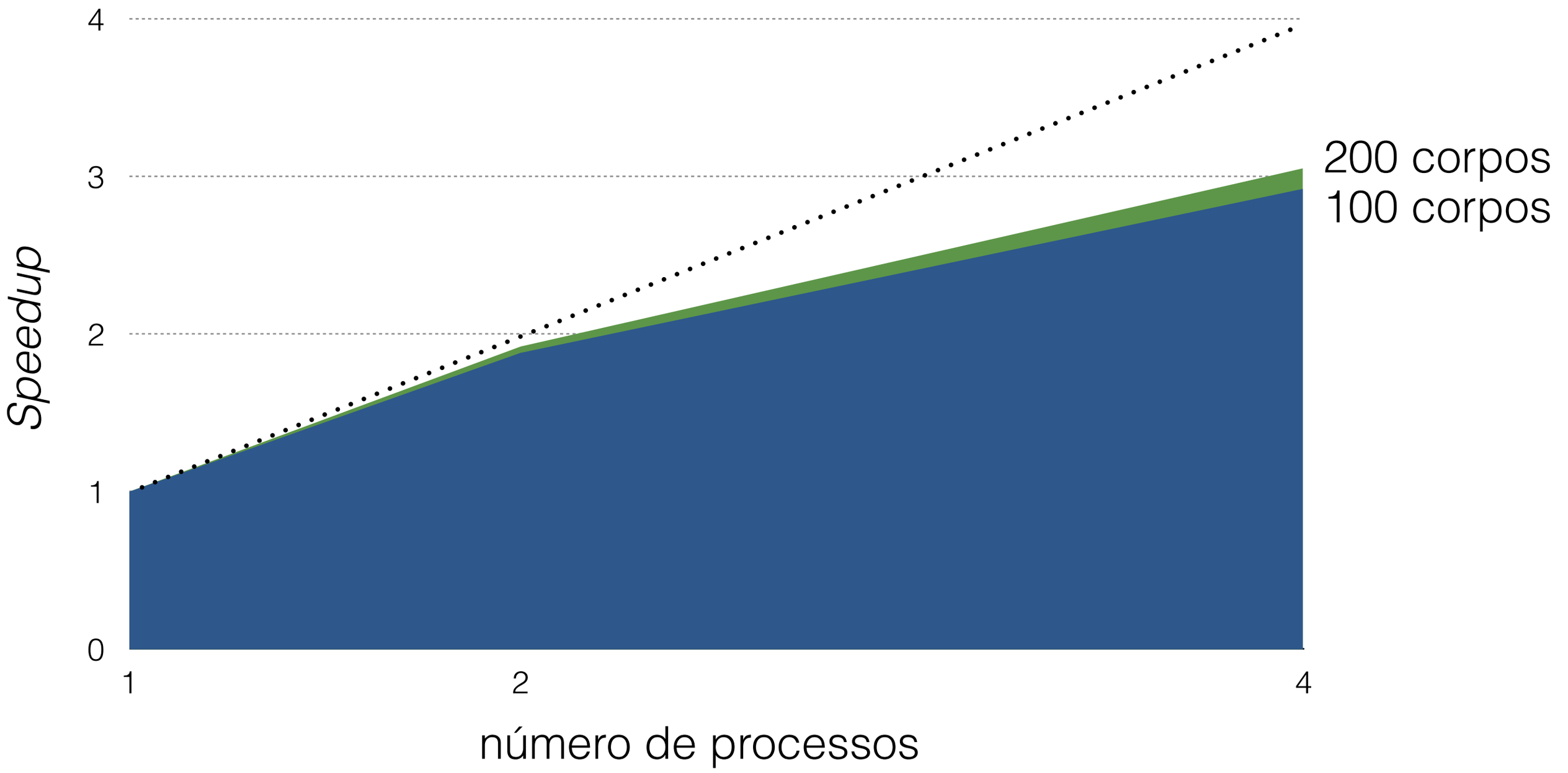
Exemplo: problema de *n*-corpos



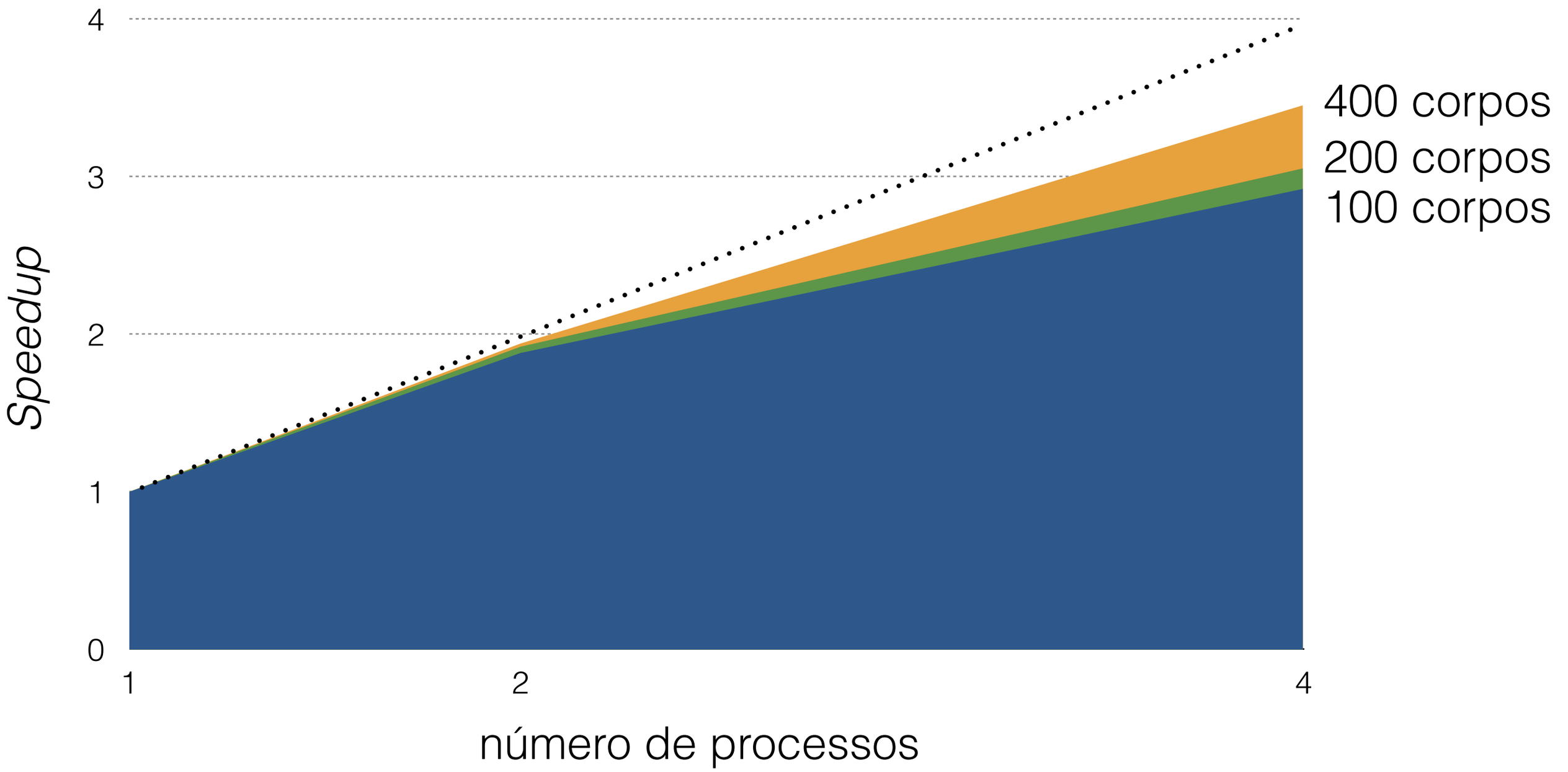
Exemplo: problema de *n*-corpos



Exemplo:
problema de *n*-corpos



Exemplo: problema de *n*-corpos



Lei de Amdahl

Gene Amdahl
(1922 - 2015)

Lei de Amdahl

Imagine um programa que leva $T_{seq} = 20\ \mathrm{s}$ para executar sequencialmente. Imagine que apenas 90% do código pode ser paralelizado.

Gene Amdahl
(1922 - 2015)

Lei de Amdahl

Imagine um programa que leva $T_{seq} = 20\ \mathrm{s}$ para executar sequencialmente. Imagine que apenas 90% do código pode ser paralelizado.

$$T_{paralelo} = 0.9 \times T_{seq}/p + 0.1 \times T_{seq}$$

Gene Amdahl
(1922 - 2015)

Lei de Amdahl

Imagine um programa que leva $T_{seq} = 20$ s para executar sequencialmente. Imagine que apenas 90% do código pode ser paralelizado.

$$T_{paralelo} = 0.9 \times T_{seq}/p + 0.1 \times T_{seq} = (18/p + 2) \text{ s}$$

Gene Amdahl
(1922 - 2015)

Lei de Amdahl

Imagine um programa que leva $T_{seq} = 20$ s para executar sequencialmente. Imagine que apenas 90% do código pode ser paralelizado.

$$T_{paralelo} = 0.9 \times T_{seq}/p + 0.1 \times T_{seq} = (18/p + 2) \text{ s}$$

$$S = \frac{T_{seq}}{T_{paralelo}}$$

Gene Amdahl
(1922 - 2015)

Lei de Amdahl

Imagine um programa que leva $T_{seq} = 20$ s para executar sequencialmente. Imagine que apenas 90% do código pode ser paralelizado.

$$T_{paralelo} = 0.9 \times T_{seq}/p + 0.1 \times T_{seq} = (18/p + 2) \text{ s}$$

$$S = \frac{T_{seq}}{T_{paralelo}} = \frac{20}{18/p + 2}$$

Gene Amdahl
(1922 - 2015)

Lei de Amdahl

Imagine um programa que leva $T_{seq} = 20$ s para executar sequencialmente. Imagine que apenas 90% do código pode ser paralelizado.

$$T_{paralelo} = 0.9 \times T_{seq}/p + 0.1 \times T_{seq} = (18/p + 2) \text{ s}$$

$$S = \frac{T_{seq}}{T_{paralelo}} = \frac{20}{18/p + 2} \leq \frac{20}{2} = 10$$

Gene Amdahl
(1922 - 2015)

Lei de Amdahl

Imagine um programa que leva $T_{seq} = 20$ s para executar sequencialmente. Imagine que apenas 90% do código pode ser paralelizado.

$$T_{paralelo} = 0.9 \times T_{seq}/p + 0.1 \times T_{seq} = (18/p + 2) \text{ s}$$

$$S = \frac{T_{seq}}{T_{paralelo}} = \frac{20}{18/p + 2} \leq \frac{20}{2} = 10$$

Generalizando, se uma fração r do código sequencial não pode ser paralelizado, o máximo speedup será $1/r$.

Gene Amdahl
(1922 - 2015)

Escalabilidade

Escalabilidade

Um programa é escalável se o tamanho do problema pode ser aumentado de forma que a eficiência não diminua conforme o número de processos aumenta.

Escalabilidade

Um programa é escalável se o tamanho do problema pode ser aumentado de forma que a eficiência não diminua conforme o número de processos aumenta.

Exemplo:

Escalabilidade

Um programa é escalável se o tamanho do problema pode ser aumentado de forma que a eficiência não diminua conforme o número de processos aumenta.

Exemplo:

$$T_{seq} = n$$

Escalabilidade

Um programa é escalável se o tamanho do problema pode ser aumentado de forma que a eficiência não diminua conforme o número de processos aumenta.

Exemplo:

$$T_{seq} = n \qquad T_{paralelo} = n/p + 1$$

Escalabilidade

Um programa é escalável se o tamanho do problema pode ser aumentado de forma que a eficiência não diminua conforme o número de processos aumenta.

Exemplo:

$$T_{seq} = n \qquad T_{paralelo} = n/p + 1$$

$$E = \frac{n}{p(n/p+1)}$$

Escalabilidade

Um programa é escalável se o tamanho do problema pode ser aumentado de forma que a eficiência não diminua conforme o número de processos aumenta.

Exemplo:

$$T_{seq} = n \qquad T_{paralelo} = n/p + 1$$

$$E = \frac{n}{p(n/p+1)} = \frac{n}{n+p}$$

Escalabilidade

Um programa é escalável se o tamanho do problema pode ser aumentado de forma que a eficiência não diminua conforme o número de processos aumenta.

Exemplo:

$$T_{seq} = n \qquad T_{paralelo} = n/p + 1$$

$$E = \frac{n}{p(n/p+1)} = \frac{n}{n+p}$$

Aumentar o número de processos:

Escalabilidade

Um programa é escalável se o tamanho do problema pode ser aumentado de forma que a eficiência não diminua conforme o número de processos aumenta.

Exemplo:

$$T_{seq} = n \qquad T_{paralelo} = n/p + 1$$

$$E = \frac{n}{p(n/p+1)} = \frac{n}{n+p}$$

Aumentar o número de processos: $p \to kp$

Escalabilidade

Um programa é escalável se o tamanho do problema pode ser aumentado de forma que a eficiência não diminua conforme o número de processos aumenta.

Exemplo:

$$T_{seq} = n \qquad T_{paralelo} = n/p + 1$$

$$E = \frac{n}{p(n/p+1)} = \frac{n}{n+p}$$

Aumentar o número de processos: $p \to kp$

Aumentar o tamanho do problema tal que:

Escalabilidade

Um programa é escalável se o tamanho do problema pode ser aumentado de forma que a eficiência não diminua conforme o número de processos aumenta.

Exemplo:

$$T_{seq} = n \qquad T_{paralelo} = n/p + 1$$

$$E = \frac{n}{p(n/p+1)} = \frac{n}{n+p}$$

Aumentar o número de processos: $p \to kp$

Aumentar o tamanho do problema tal que: $n \to xn$

Escalabilidade

Um programa é escalável se o tamanho do problema pode ser aumentado de forma que a eficiência não diminua conforme o número de processos aumenta.

Exemplo:

$$T_{seq} = n \qquad T_{paralelo} = n/p + 1$$

$$E = \frac{n}{p(n/p+1)} = \frac{n}{n+p}$$

Aumentar o número de processos: $p \to kp$

Aumentar o tamanho do problema tal que: $n \to xn$

$$E^* = \frac{xn}{xn+kp}$$

Escalabilidade

Um programa é escalável se o tamanho do problema pode ser aumentado de forma que a eficiência não diminua conforme o número de processos aumenta.

Exemplo:

$$T_{seq} = n \qquad T_{paralelo} = n/p + 1$$

$$E = \frac{n}{p(n/p+1)} = \frac{n}{n+p}$$

Aumentar o número de processos: $p \to kp$

Aumentar o tamanho do problema tal que: $n \to xn$

$$E^* = \frac{xn}{xn+kp} \qquad \text{Se } x = k$$

Escalabilidade

Um programa é escalável se o tamanho do problema pode ser aumentado de forma que a eficiência não diminua conforme o número de processos aumenta.

Exemplo:

$$T_{seq} = n \qquad T_{paralelo} = n/p + 1$$

$$E = \frac{n}{p(n/p+1)} = \frac{n}{n+p}$$

Aumentar o número de processos: $p \to kp$

Aumentar o tamanho do problema tal que: $n \to xn$

$$E^* = \frac{xn}{xn+kp} \qquad \text{Se } x = k \qquad E^* = \frac{n}{n+p}$$

Escalabilidade

Um programa é escalável se o tamanho do problema pode ser aumentado de forma que a eficiência não diminua conforme o número de processos aumenta.

Exemplo:

$$T_{seq} = n \qquad T_{paralelo} = n/p + 1$$

$$E = \frac{n}{p(n/p+1)} = \frac{n}{n+p}$$

Aumentar o número de processos: $p \to kp$

Aumentar o tamanho do problema tal que: $n \to xn$

$$E^* = \frac{xn}{xn+kp} \qquad \text{Se } x = k \qquad E^* = \frac{n}{n+p} = E$$

Escrevendo programas paralelos

Escrevendo programas paralelos

Para um “bom” programa paralelo, você deve:

Escrevendo programas paralelos

Para um “bom” programa paralelo, você deve:

- distribuir o trabalho mais ou menos igualmente entre os processos (balanceamento de carga).

Escrevendo programas paralelos

Para um “bom” programa paralelo, você deve:

- distribuir o trabalho mais ou menos igualmente entre os processos (balanceamento de carga).
- minimizar a comunicação.

Escrevendo programas paralelos

Para um “bom” programa paralelo, você deve:

- distribuir o trabalho mais ou menos igualmente entre os processos (balanceamento de carga).

- minimizar a comunicação.

(porém não há um método automático para paralelizar um programa. Se existisse, a gente não precisaria dar esta parte do curso…)

Metodologia de Foster

Ian T. Foster
(1959 -)

Metodologia de Foster

4 passos:

Ian T. Foster
(1959 -)

Metodologia de Foster

Ian T. Foster
(1959 -)

4 passos:

- Particionamento: Dividir a computação em tarefas menores, com o objetivo de identificar tarefas que possam ser paralelizadas.

Metodologia de Foster

Ian T. Foster
(1959 -)

4 passos:

- Particionamento: Dividir a computação em tarefas menores, com o objetivo de identificar tarefas que possam ser paralelizadas.

- Comunicação: Determina que comunicação precisa ser feita entre as tarefas identificadas.

Metodologia de Foster

Ian T. Foster
(1959 -)

4 passos:

- Particionamento: Dividir a computação em tarefas menores, com o objetivo de identificar tarefas que possam ser paralelizadas.

- Comunicação: Determina que comunicação precisa ser feita entre as tarefas identificadas.

- Aglomeração: Combina as tarefas e as comunicações em tarefas aglomeradas.

Metodologia de Foster

Ian T. Foster
(1959 -)

4 passos:

- Particionamento: Dividir a computação em tarefas menores, com o objetivo de identificar tarefas que possam ser paralelizadas.

- Comunicação: Determina que comunicação precisa ser feita entre as tarefas identificadas.

- Aglomeração: Combina as tarefas e as comunicações em tarefas aglomeradas.

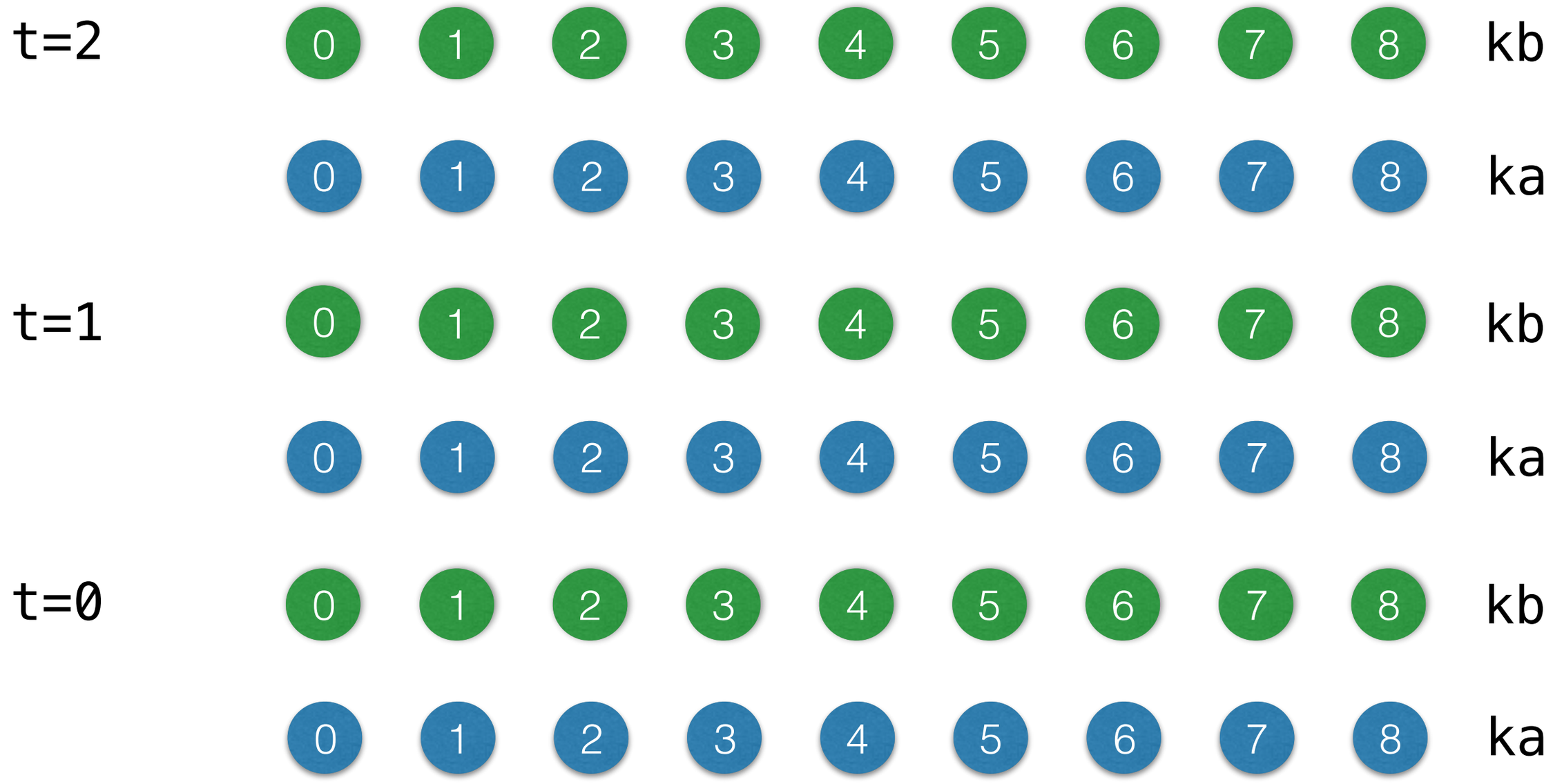
- Mapeamento: Atribui as tarefas aglomeradas a diferentes processos.

Exemplo

```
//fragmento de código

float ka[9], kb[9];
int i,t;

// ka inicializado por aqui!!!

for (t=0;t<3;t++){
    for (i=1;i<8;i++){
        kb[i]=ka[i-1] - 2*ka[i] + ka[i+1];
    }
    for (i=1;i<8;i++){
        ka[i]=kb[i];
    }
}
```

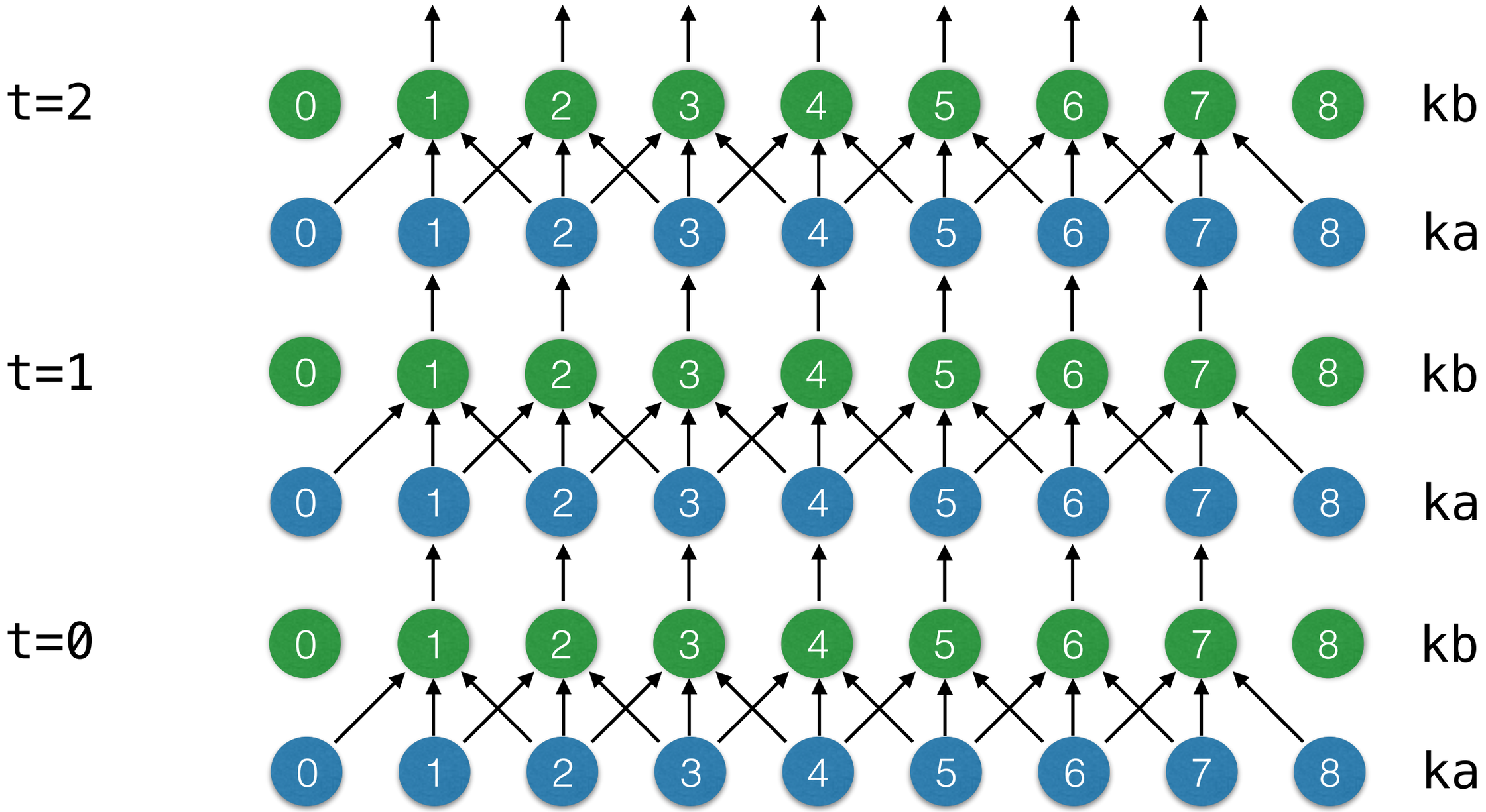
Exemplo

Particionamento



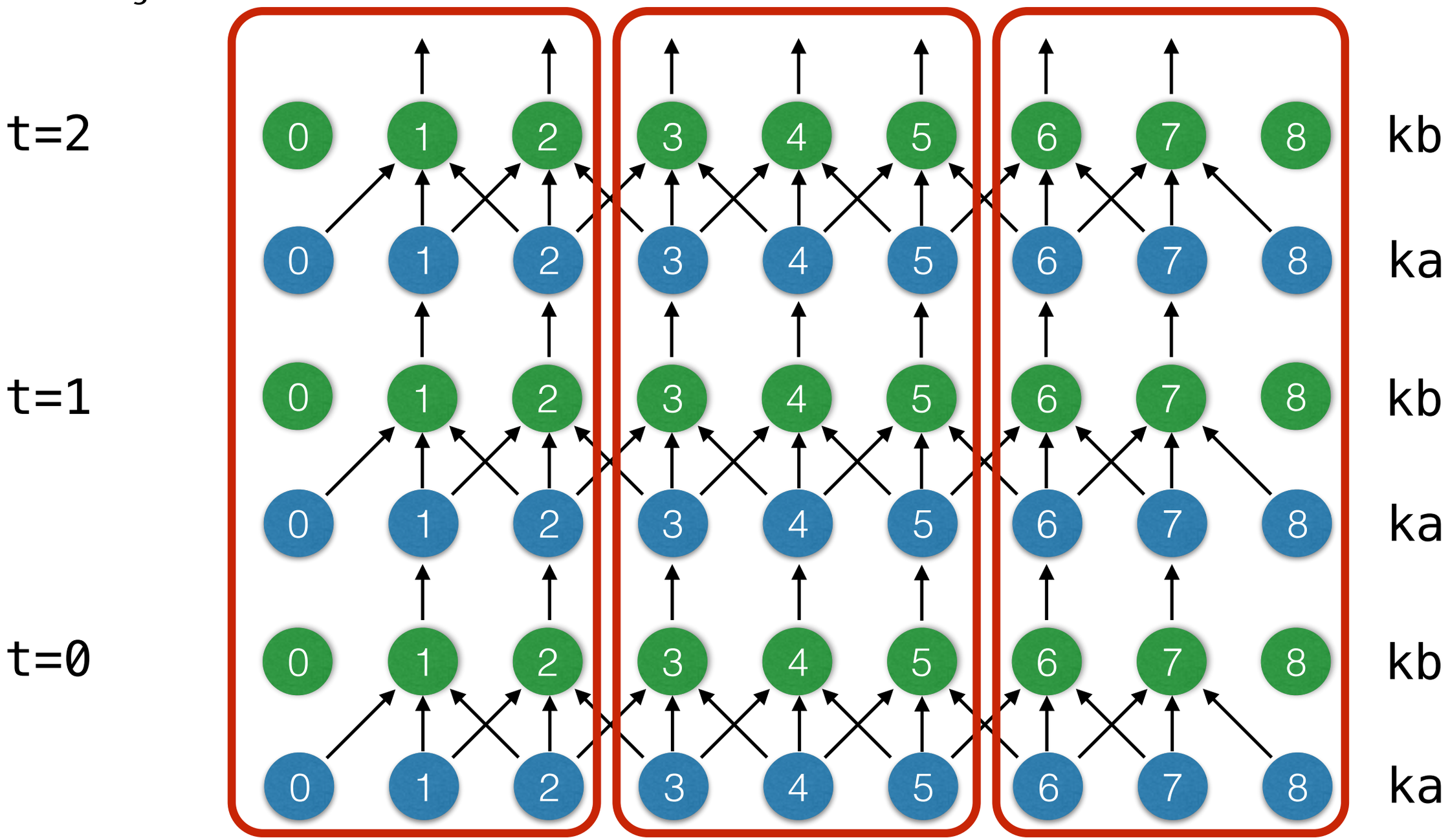
Exemplo

Comunicação



Exemplo

Aglomeração



Exemplo

Mapeamento