Avaliação de Desempenho

ELC139 - Programação Paralela

João Vicente Ferreira Lima (UFSM)

Universidade Federal de Santa Maria
jvlima@inf.ufsm.br
http://www.inf.ufsm.br/~jvlima

2023/1

Outline

Outline

Speedup

- Número de processadores/cores $= p$
- Tempo de execução sequencial $= T_{serial}$
- Tempo de execução paralelo em p processadores $= T_p$
- Speedup $= S$

$$S = \frac{T_{serial}}{T_p}$$

Se $T_p = \frac{T_{serial}}{p} \rightarrow$ **speedup linear**

Eficiência

$$E = \frac{S}{p} = \frac{\frac{T_{serial}}{T_p}}{p} = \frac{T_{serial}}{p * T_p}$$

Table 2.4 Speedups and Efficiencies of a Parallel Program

p	1	2	4	8	16
S	1.0	1.9	3.6	6.5	10.8
E = S/p	1.0	0.95	0.90	0.81	0.68

Introduction to Parallel Computing, Grama et al, 2003.

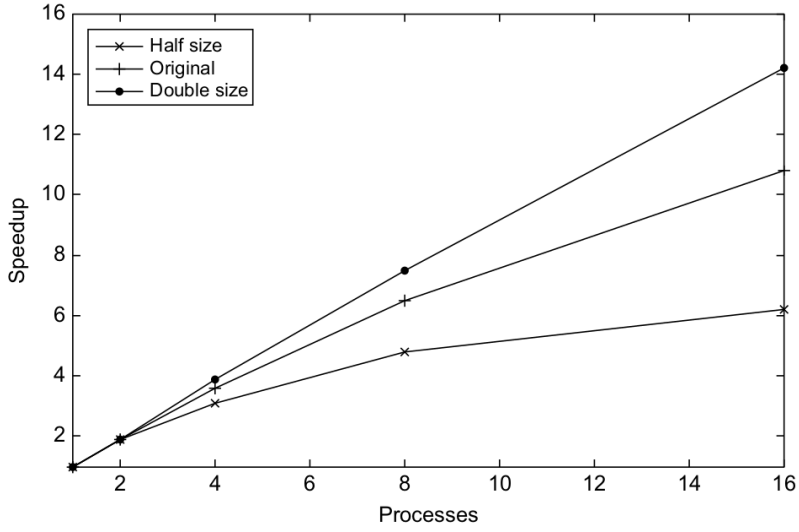
Eficiência

Table 2.5 Speedups and Efficiencies of a Parallel Program on Different Problem Sizes

	p	1	2	4	8	16
Half	S	1.0	1.9	3.1	4.8	6.2
	E	1.0	0.95	0.78	0.60	0.39
Original	S	1.0	1.9	3.6	6.5	10.8
	E	1.0	0.95	0.90	0.81	0.68
Double	S	1.0	1.9	3.9	7.5	14.2
	E	1.0	0.95	0.98	0.94	0.89

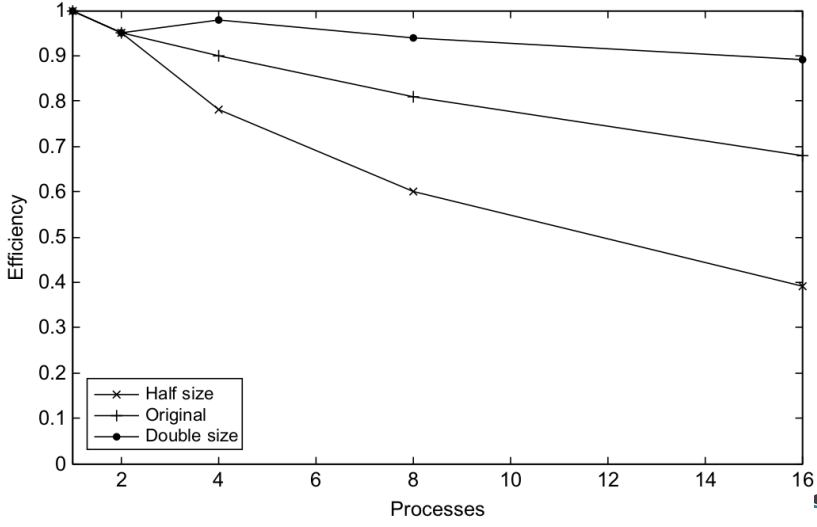
Introduction to Parallel Computing, Grama et al, 2003.

Speedup



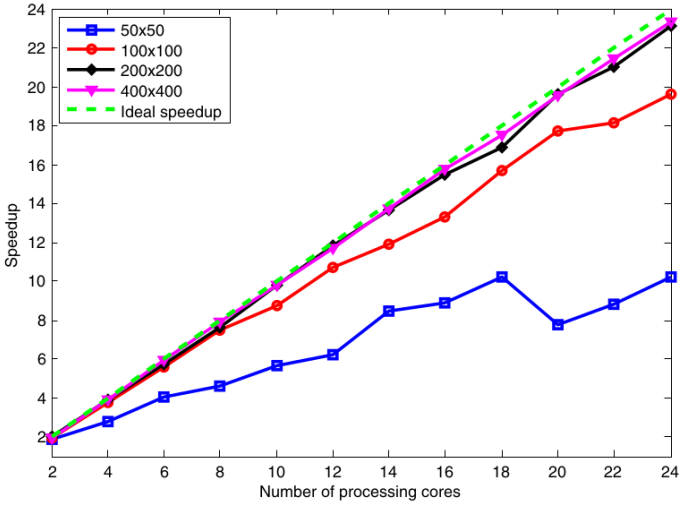
Introduction to Parallel Computing, Grama et al, 2003.

Eficiência



Introduction to Parallel Computing, Grama et al, 2003.

Speedup



On the parallel efficiency and scalability of the correntropy coefficient for image analysis. https://doi.org/10.1186/s13173-014-0018-4

Sobrecusto

- Speedup/eficiência aumenta com o tamanho do problema
- Muitos programas paralelos adicionam um sobrecusto (*overhead*) de dividir o trabalho

$$T_p = \frac{T_{serial}}{p} + T_{overhead}$$

- Uma das formas de medir o sobrecusto paralelo é medindo o tempo do programa paralelo com um processador T_1

$$T_{overhead} = \frac{T_1}{T_{serial}}$$

Speedup

Tempo sequencial T_{serial}

- Qual implementação sequencial?
- Versão base da paralela ou outra versão?

- Exemplo: shell sort paralelo
  - Shell sort sequencial?
  - T_1?
  - Quicksort?
  - Mesmo processador ou outro mais rápido?

Outline

Lei de Amdahl

- Observação feita por Gene Amdahl em 1960, mais tarde **Lei de Amdahl**
- O speedup máximo é limitado independentemente do número de processadores, a menos que a totalidade do programa seja paralelo
  - Speedup máximo de $\frac{1}{r}$ onde r é a fração não paralela

Exemplo

- 90% do programa é paralelo e $T_{serial} = 20$. Então:

$$T_p = 0.9 * \frac{T_{serial}}{p} + 0.1 * T_{serial} = 18/p + 2$$

$$S = \frac{T_{serial}}{0.9 * T_{serial} + 0.1 * T_{serial}} = \frac{20}{18/p + 2}$$

Se $p \to \infty$ então $S <= \frac{T_{serial}}{0.1 * T_{serial}} = \frac{20}{10} = 10$

Lei de Gustafson

Há diversas razões para não se preocupar com a lei de Amdahl. Quanto maior o problema menor a parte sequencial na maioria dos problemas.

- Muitos problemas apresentam speedup significativos em sistemas distribuídos.
- Speedup pequeno não é problema, sobretudo quando o esfoço é pequeno.

Escalabilidade

O problema é escalável se a eficiência E permaneçe constante quando aumentamos o tamanho do problema e o número de processadores.
Se é **escalável** podemos tratar problemas maiores.

- **Escalabilidade forte** – a eficiência é mantida se p cresce sem aumentar o tamanho do problema.
- **Escalabilidade fraca** – a eficiência é mantida se p cresce e também aumenta o tamanho do problema.

Outline

Medindo tempo

- Como medir o tempo ?
  - Tempo de processamento
  - Tempo de E/s
  - Tempo de comunicação
  - Tempo de barreira (*wall clock time*)
- Não estamos interessados no tempo entre começo e fim do programa no geral.
  - Em um bubble sort, estamos interessados no tempo para ordenar as chaves.
  - Não queremos o tempo para ler ou imprimir as chaves.
- Queremos o tempo entre quando o primeiro processo começou a execução do código até quando o último processo terminou a execução.

Medindo tempo

- Função hipotética que retorna o tempo em segundos desde um tempo fixo.
  - UNIX é 01/01/1970
- As APIs tem uma função de tempo.
  - `MPI_Wtime`
  - `omp_get_wtime`

```
1 double start, finish;
2
3 start = Get_current_time();
4
5 /* Code that we want to time */
6
7 finish = Get_current_time();
8 printf("The elapsed time = %e seconds\n", finish-start);
```

Medindo tempo

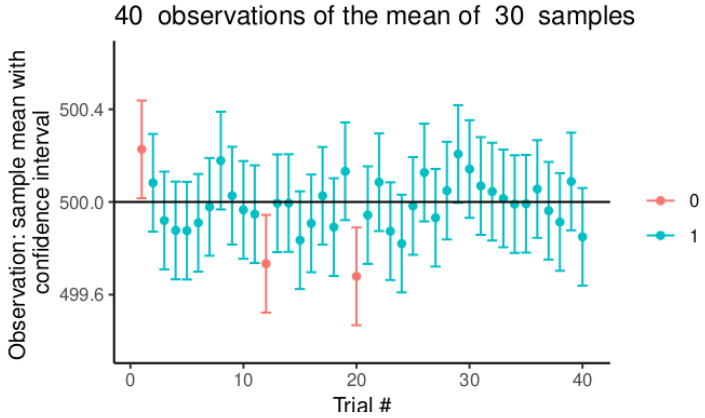
- A sincronização pode ser garantida por barreiras.

```
1  double global_elapsed ;
2  double my_start , my_finish , my_elapsed;
3  /* Synchronize all processes/threads */
4  Barrier ();
5  my_start = Get_current_time ();
6  /* Code that we want to time */
7  my_finish = Get_current_time ();
8  my_elapsed = my_finish - my_start ;
9  /* Find the max across all processes/threads */
10 global_elapsed = Global_max ( my_elapsed );
11 if ( my_rank == 0)
12   printf ("The elapsed time = %e seconds\n", global_elapsed
```

Medindo tempo

Como apresentar os tempos?

- Média – μ
- Mediana
- Desvio padrão – σ
- Intervalo de confiança



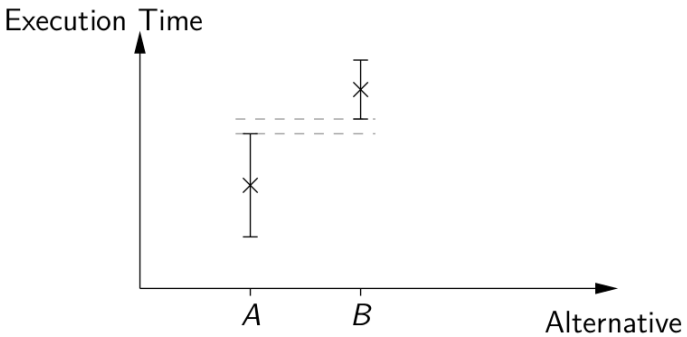
Intervalo de confiança

Chance de 95% da média real estar dentro de $2\frac{\sigma}{\sqrt{n}}$ da média da amostra.

Medindo tempo

- Assumindo que temos duas alternativas A e B
- Temos as médias μ_A e μ_B e intervalos de confiança
- Os dois intervalos de confiança com 95% não sobrepõem

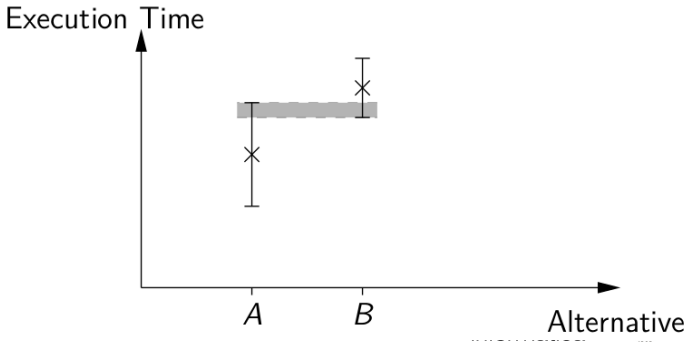
Então $\mu_A < \mu_B$ com 95% de confiança.



Medindo tempo

- Assumindo que temos duas alternativas A e B
- Temos as médias μ_A e μ_B e intervalos de confiança
- Os dois intervalos de confiança com 95% se sobrepõem.

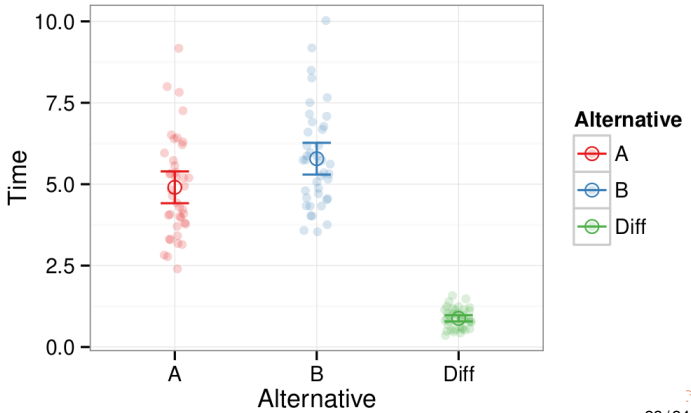
Nada podemos concluir!!



Medindo tempo

- Assumindo que temos duas alternativas A e B
- Temos as médias μ_A e μ_B e intervalos de confiança
- Os dois intervalos de confiança com 95% se sobrepõem.

Nada podemos concluir!!



https://joao-ufsm.github.io/par2023a/