# Valutazione dell'efficienza di algoritmi e software in ambiente parallelo

parte 2-0

PARALLEL AND DISTRIBUTED COMPUTING

PROF. G. LACCETTI

A.A. 2022/2023

1

## Domanda

2

E' possibile ottenere
speed-up (scalati) prossimi allo
speed-up ideale?



2

# T(p) si può decomporre in 2 parti:

3

una parte relativa alle operazioni che sono eseguite esclusivamente in sequenziale

una parte relativa alle operazioni che possono essere eseguite concorrentemente

$T_s$ $T'_c$

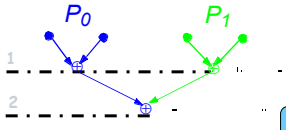
$T(1) = T_s + T'_c$



3

# Somma di $n$ =4 su p=2, $T(1)= (n-1)t_{calc} = 3t_{calc}$

4




2 addizioni eseguite concorrentemente ($T_c$) da 2 processori (p) al passo 1

1 addizione eseguita in sequenziale ($T_s$) da 1 processore al passo 2

$T(1) = T_s + T_c$

$$T(1) = (1+2)t_{calc} = 3t_{calc}$$

$T_s = 1\ t_{calc}$

Operazione che deve essere eseguita in sequenziale

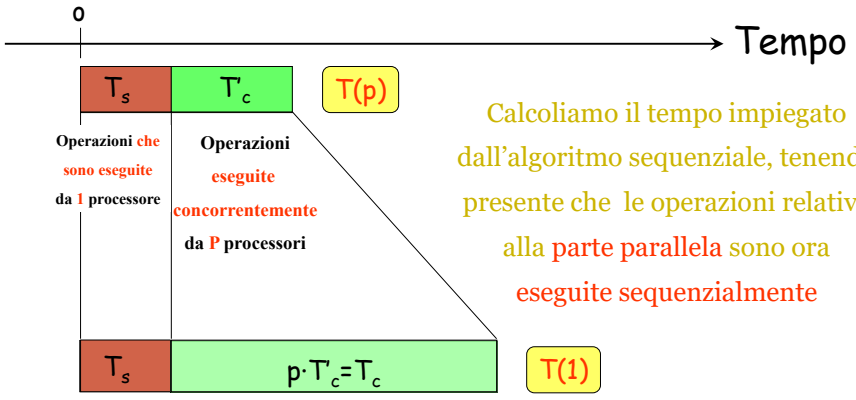
$T_c = 2\ t_{calc}$

Operazioni che possono essere eseguite in parallelo



4

$$T(p) = T_s + T_c'$$




Calcoliamo il tempo impiegato dall'algoritmo sequenziale, tenendo presente che le operazioni relative alla parte parallela sono ora eseguite sequenzialmente

$$T(1) = T_s + p \cdot T_c'$$



5

# Lo speed up...



Avendo :

$$\begin{cases} T(1) = T_s + p \cdot T_c' \\ T(p) = T_s + T_c' \end{cases}$$

Si assume T(p) come tempo unitario

$$T(p) = T_s + T_c' = 1 \longrightarrow T_c' = 1 - T_s$$

$$T(1) = T_s + p\left(1 - T_s\right)$$



6

# Lo speed up...

7

Posto $\mathbf{T_s = \alpha'}$

(frazione di T(p) per la componente sequenziale)

$$SS(p) = \frac{T(1)}{T(p)} = \frac{T_s + p(1 - T_s)}{1} = \alpha' + p(1 - \alpha') = p + (1 - p)\alpha'$$

Modello di SPEED UP SCALATO

**LEGGE DI GUSTAFSON**

(Reinterpretazione della legge di Ware)



7

# Lo speed up...

8

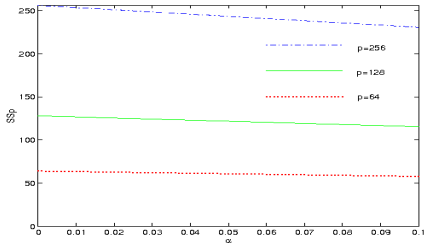
$$SS(p) = \alpha' + p(1 - \alpha') = p + (1 - p)\alpha'$$

$\alpha' = 0.10$

p = 10 ➡ SS(p) = 9.10

p = 64 ➡ SS(p) = 57.60

p = 100 ➡ SS(p) = 90.1




Si possono ottenere

Speed up prossimi allo speed-up Ideale!



8

# Quindi

9

Nell'algoritmo della somma

abbiamo calcolato speedup ed efficienza scalata

aumentando la complessità di tempo del problema

in maniera proporzionale al numero di processori

ovvero

Nel passare da $p_0$ a $p_1$ processori con $p_1 = k\ p_0$,

Siamo passati da $T(p_0, n_0)$ a $T(p_1, n_1)$ con $n_1 = k\ n_0$



9

# Quindi

10

$$T(1,n_1) = \frac{p_1}{p_0} T(1,n_0) \quad \Leftarrow \quad T(1,n_1) = I(p_0,p_1,n_0)$$

k

la funzione $I$ esprime la legge secondo cui deve aumentare la *complessità di tempo* dell'algoritmo sequenziale con dimensione $n_1$ su $p_1$ processori a partire da $p_0$ e da $T(1, n_0)$ affinché l'efficienza scalata non degradi.

I è detta ISOEFFICIENZA



10

## Domanda

11

Come si calcola l'isoefficienza
per un qualsiasi algoritmo ?



11

## Domanda

12

Nel passare da $p_0$ a $p_1$ processori con $p_1 > p_0$,
come deve aumentare (*scalare*)
la complessità del problema
affinché l'efficienza sia costante ?



12

## Come calcolare l'isoefficienza?

13

$$E(p_0, n_0) = \frac{S(p_0, n_0)}{p_0} = \frac{1}{\frac{O_h(n_0, p_0)}{T(1, n_0)} + 1} = E(p_1, n_1) = \frac{S(p_1, n_1)}{p_1} = \frac{1}{\frac{O_h(n_1, p_1)}{T(1, n_1)} + 1}$$

$$\frac{O_h(n_0, p_0)}{T(1, n_0)} = \frac{O_h(n_1, p_1)}{T(1, n_1)}$$

$$T(1, n_1) = T(1, n_0) \frac{O_h(n_1, p_1)}{O_h(n_0, p_0)}$$

ISOEFFICIENZA



13

## Come calcolare l'isoefficienza della somma?

14

$O_h(p) = p \log_2 p$

$T(1,n) = n$

$$n_1 = n_0 \frac{p_1 \log_2 p_1}{p_0 \log_2 p_0} = I(n_0, p_0, p_1)$$

ISOEFFICIENZA

Nel passare da $p_0$ a $p_1$ con $p_1 > p_0$, la complessità di tempo del problema deve aumentare, rispetto alla dimensione iniziale $n_0$, del fattore

$k = (p_1 \log_2 p_1)/ p_0 \log_2 p_0$



14

## In generale



Un algoritmo si dice scalabile se l'efficienza rimane costante (o non degrada) al crescere del numero dei processori e della dimensione del problema



15

## Quello per la somma è un algoritmo scalabile?



$$\frac{n_1}{n_0} = \frac{p_1 \log_2 p_1}{p_0 \log_2 p_0}$$

k

$p_0 = 4$ $n_0 = 64$

$p_1 = 8$ $n_1 = 3 \times n_0 = 192$

$p_1 = 16$ $n_1 = 8 \times n_0 = 512$

k = (8 x 3) / (4 x 2) = 3

k = (16 x 4) / (4 x 2) = 8



16

## Somma di n numeri, p processori: isoefficienza



| n \ p | 1 | 4 | 8 | 16 | 32 |
|---|---|---|---|---|---|
| 64 | 1.0 | .80 | .57 | .33 | .17 |
| 192 | 1.0 | .92 | .80 | .60 | .38 |
| 512 | 1.0 | .97 | .91 | .80 | .62 |

L'efficienza rimane costante al crescere di p e di n

La somma è scalabile!



17

## Speed up ed efficienza scalata



$$SS = \frac{T(p_0,n_0)}{T(p_1,n_1)}$$

Speed-up scalato

$$ES = \frac{SS}{p}$$

Efficienza scalata



18

## Problema

19

Se si esegue l'algoritmo su un calcolatore MIMD a memoria distribuita, il tempo di esecuzione dipende solo dal numero di operazioni eseguite in differenti passi temporali?

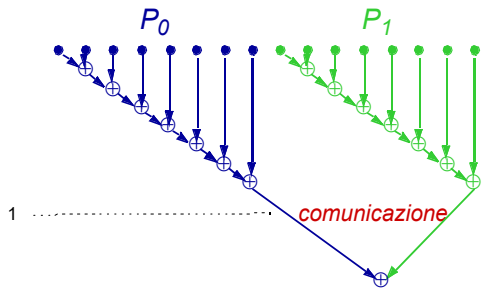


19

## Esempio: calcolo della somma di n=16 numeri

20

ALGORITMO PARALLELO
p=2




$t_{com}$= tempo per comunicare un dato tra due processori

**L'algoritmo richiede la comunicazione di 1 dato tra 2 processori**

$$T(2)=8\ t_{calc} + t_{com}$$

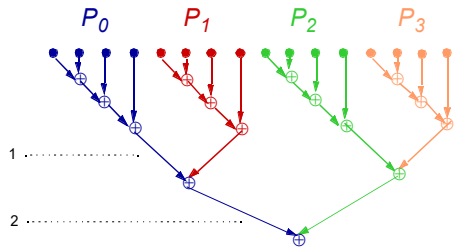


20

## Esempio: calcolo della somma di n=16 numeri



ALGORITMO PARALLELO
p=4




$T(4)=5\ t_{calc}+2t_{com}$

$t_{com}$= tempo per comunicare un dato tra due processori

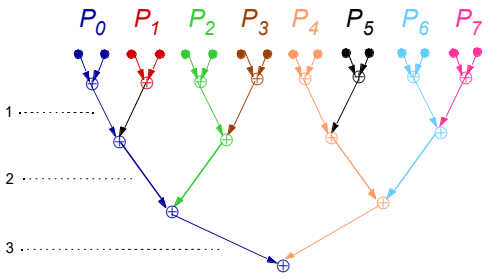


21

## Esempio: calcolo della somma di n=16 numeri



ALGORITMO PARALLELO
p=8




$T(8)=4\ t_{calc}+3t_{com}$

$t_{com}$= tempo per comunicare un dato tra due processori



22

## In sintesi…

23

| p | T(p) |
|---|---|
| 1 | $15t_{calc}$ |
| 2 | $8t_{calc} + 1\ t_{com}$ |
| 4 | $5t_{calc} + 2\ t_{com}$ |
| 8 | $4t_{calc} + 3\ t_{com}$ |
| $2^k$ | $?\ t_{calc} + ?\ t_{com}$ |

?

In generale quanto vale T(p) considerando le comunicazioni?





## In generale: calcolo di T(p) considerando le comunicazioni

24

ALGORITMO PARALLELO della somma di *n* numeri su $p=2^k$ processori

$p=1 \quad T(1)=15\ t_{calc}$

$p=2 \quad T(2)=8\ t_{calc}= (7+1)\ t_{calc} + 1\ t_{com}$

$p=4 \quad T(4)=5\ t_{calc}= (3+2)\ t_{calc} + 2\ t_{com}$

$p=8 \quad T(8)=4\ t_{calc}= (1+3)\ t_{calc} + 3\ t_{com}$

***n = 16***

.....

$p=2^k \quad T(p)= (\frac{n}{p} -1 +\log_2 p)\ t_{calc} + (\log_2 p)\ t_{com}$

$t_{calc}$= tempo di esecuzione di 1 addizione

$t_{com}$= tempo di 1 comunicazione

## In generale

25

Il tempo di esecuzione di un algoritmo parallelo, distribuito su $p$ processori, comprende 3 componenti:

- $\mathbf{T_s}$ tempo per eseguire la parte seriale
- $\mathbf{T_c/p}$ tempo per eseguire la parte parallela
- $T_o$ costo di comunicazione con $\mathbf{T_o(p) \geq 0,\ p > 1}$

$$T(p) = T_s + \frac{T_c}{p} + T_0(p)$$



25

## Overhead di comunicazione unitario

26

$t_{com}$ = tempo di comunicazione di 1 dato

$t_{calc}$ = tempo di esecuzione di 1 operazione fl.p.

definiamo

Overhead di Comunicazione Unitario

il rapporto:

$$OC = \frac{t_{com}}{t_{calc}}$$



26

## Esempio: calcolo della somma di n=16 numeri



supponiamo
T(1)=15tcalc

$$oc = \frac{t_{com}}{t_{calc}} = 2$$

Trascurando la comunicazione

| p | $T_p$ | $S_p$ | $E_p$ |
|---|---|---|---|
| 2 | $8t_{calc}$ | 1.88 | 0.94 |
| 4 | $5t_{calc}$ | 3.00 | 0.75 |
| 8 | $4t_{calc}$ | 3.75 | 0.47 |

$p=2^k \quad T(p)= (\frac{n}{p} -1 +\log_2 p)\ t_{calc}$

Considerando la comunicazione

comunicazioni

| p | $T_p$ | $S_p$ | $E_p$ |
|---|---|---|---|
| 2 | $(8+2\cdot1)t_{calc}$ | 1.50 | 0.75 |
| 4 | $(5+2\cdot2)t_{calc}$ | 1.67 | 0.42 |
| 8 | $(4+2\cdot3)t_{calc}$ | 1.88 | 0.24 |

$p=2^k \quad T(p)=(\frac{n}{p} -1 +\log_2 p)\ t_{calc} + (\log_2 p)\ t_{com}$





## Overhead di comunicazione



$t_{com}$ = tempo di comunicazione di 1 dato
$t_{calc}$ = tempo di esecuzione di 1 operazione fl.p.

definiamo

Overhead di Comunicazione Totale

il rapporto:

$$OC_p = \frac{T_{com}(p)}{T_{calc}(p)}$$

## Overhead di comunicazione

29

$t_{com}$ = tempo di comunicazione di 1 dato

$t_{calc}$ = tempo di esecuzione di 1 operazione fl.p.

definiamo

**Overhead di Comunicazione Totale**

il rapporto:

$$OC_p = \frac{T_{com}(p)}{T_{calc}(p)}$$

Fornisce una misura del "peso" della comunicazione sul tempo di esecuzione dell' algoritmo





## Esempio: calcolo della somma di n=16 numeri

30

$$oc = \frac{t_{com}}{t_{calc}} = 2$$

Posto ad esempio:

| p | $T_{calc}(p)$ | $T_{com}(p)$ | $OC_p$ |
|---|---|---|---|
| 2 | $8t_{calc}$ | $1t_{com}$ | 0.25 |
| 4 | $5t_{calc}$ | $2t_{com}$ | 0.80 |
| 8 | $4t_{calc}$ | $3t_{com}$ | 1.50 |

Su 8 processori il tempo di comunicazione pesa di più rispetto al tempo di esecuzione