



# Parallel and Distributed Computing

a.a. 2023-2024

---

## Somma di $N$ numeri

Su architettura MIMD-DM

Prof. Giuliano Laccetti

Materiale tratto dal testo  
A. Murli – Lezioni di Calcolo Parallelo, Liguori  
e  
da Appunti e Lezioni tenute dal prof. Murli in vari corsi in a.a. precedenti

27/09/2023

1



27/09/2023


2



---

27/09/2023 Parallel and Distributed Computing - a.a. 2022/2023 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory 3

3



# Parallel and Distributed Computing

a.a. 2023-2024

---

## Somma di N numeri

Su architettura MIMD-DM

Prof. Giuliano Laccetti

27/09/2023 Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

4



## PROBLEMA:

Calcolo della somma di  $N$   
numeri

$$a_0 + a_1 + \dots + a_{N-1}$$

su un calcolatore parallelo  
tipo **MIMD** con  $p$  processori  
**A MEMORIA DISTRIBUITA**



## Somma di $N$ numeri

Su un calcolatore monoprocesore la somma è  
calcolata *eseguendo le  $N-1$  addizioni una per volta*  
secondo un ordine prestabilito

sumtot :=  $a_0$

sumtot := sumtot +  $a_1$

sumtot := sumtot +  $a_2$

.....

sumtot := sumtot +  $a_{N-1}$

## Somma di N numeri

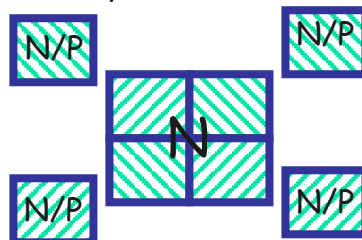
Su un calcolatore monoprocesore la somma è calcolata eseguendo le  $N-1$  addizioni una per volta secondo un ordine prestabilito

```
begin
  sumtot := a0;
  for i=1 to N-1 do
    sumtot := sumtot + ai ;
  endfor
end
```

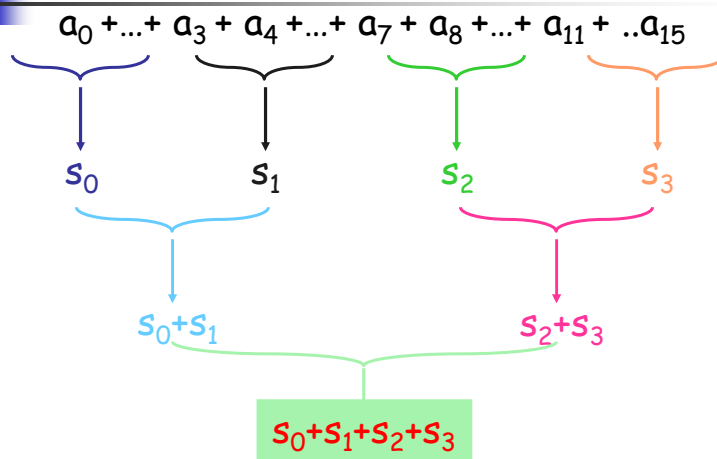
Qual è  
l'ALGORITMO PARALLELO?

## CALCOLO PARALLELO

Decomporre un problema di dimensione  $N$  in  $P$  sottoproblemi di dimensione  $N/P$  e risolverli contemporaneamente su più calcolatori



## Esempio: $N=16, p=4$



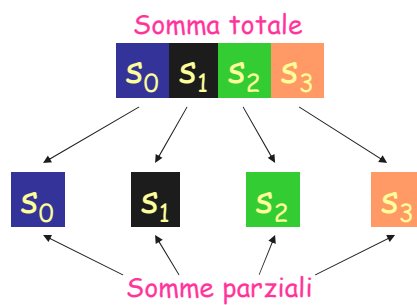
27/09/2023

Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

9

## IDEA

Suddividere la somma in somme parziali ed assegnare ciascuna somma parziale ad un processore



27/09/2023

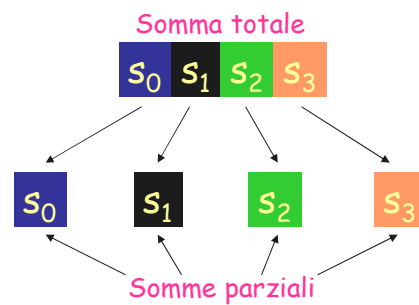
Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

10



## IDEA

Le somme parziali devono poi essere **combinare** in modo opportuno per ottenere la somma totale



27/09/2023

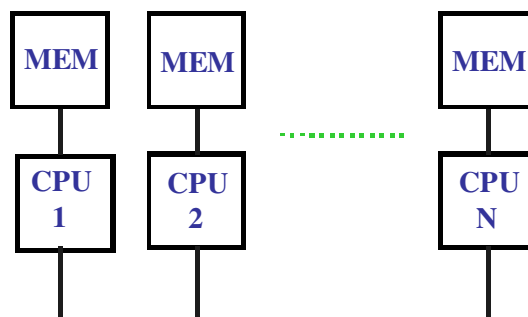
Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

11

11



## Schema Calcolatori MIMD a memoria distributa (distributed-memory)



27/09/2023

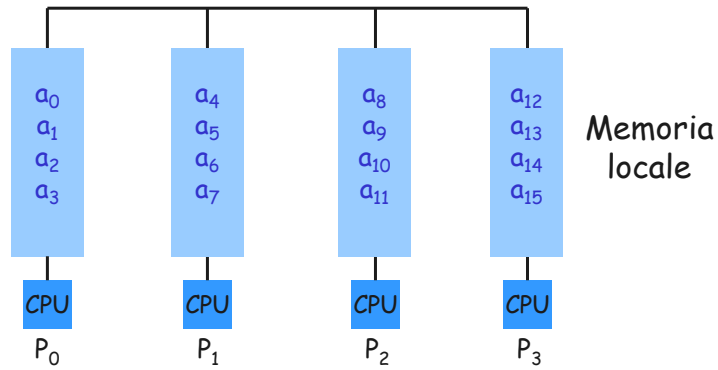
Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

12

12

## Somma - MIMD Distributed Memory

Esempio:  $N=16$ ,  $p=4$



Distribuzione degli addendi fra i processori.

27/09/2023

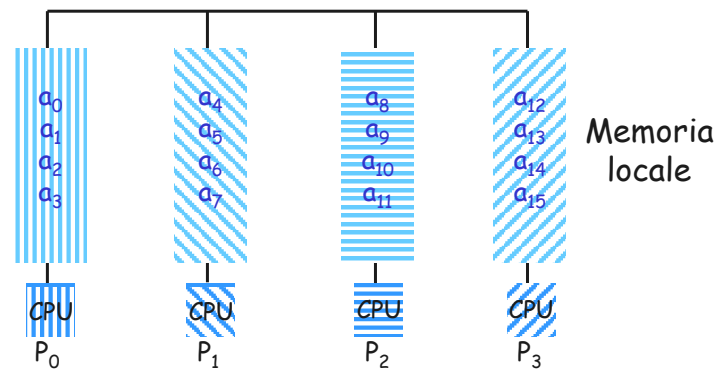
Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

13

13

## Somma - MIMD Distributed Memory

Esempio:  $N=16$ ,  $p=4$



Concorrentemente tutti i processori  
calcolano una somma parziale

27/09/2023

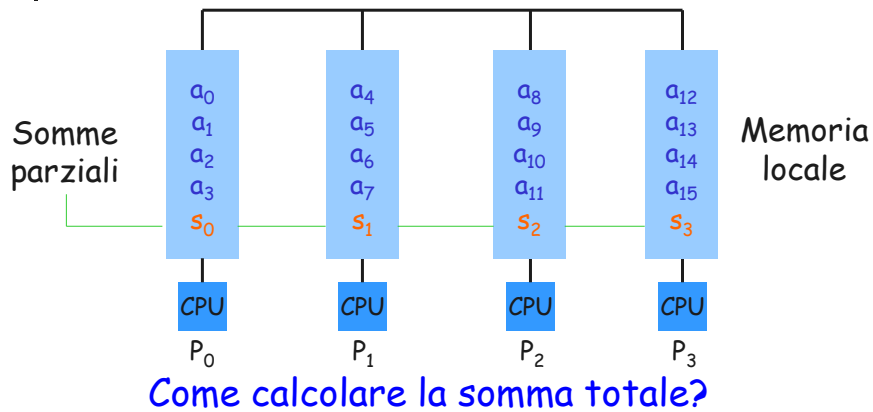
Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

14

14

## Somma - MIMD Distributed Memory

Esempio:  $N=16, p=4$



27/09/2023

Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

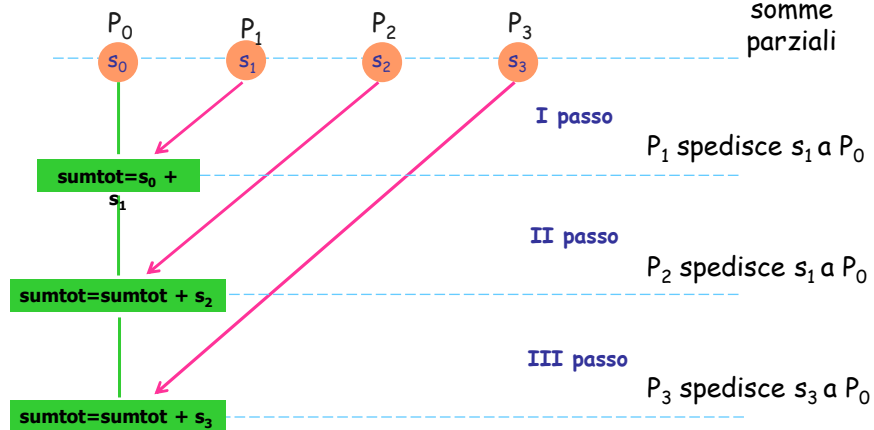
15

15

### I strategia

$p=4$

Calcolo  
somme  
parziali



29/09/2022

Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

16

16



## I strategia

Ogni processore

- calcola la propria **somma parziale**

Ad ogni passo

- ciascun processore invia tale valore ad un **unico processore prestabilito**

Tale processore **contiene la somma totale.**

**Operazioni concorrenti**

27/09/2023

Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

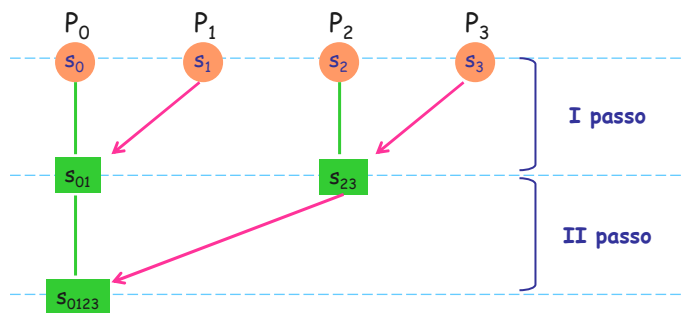
17

17

## II strategia

$p=4$

Calcolo  
somme  
parziali



Il numero di "passi" è 2

27/09/2023

Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

18

18

## II strategia

Ogni processore

- calcola la propria **somma parziale**.

Ad ogni passo,

- coppie distinte di processori **comunicano contemporaneamente**
- in ogni coppia, un processore **invia all'altro la propria somma parziale che provvede all'aggiornamento della somma**

Il risultato è in un unico processore **prestabilito**

**Operazioni concorrenti**

27/09/2023

Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

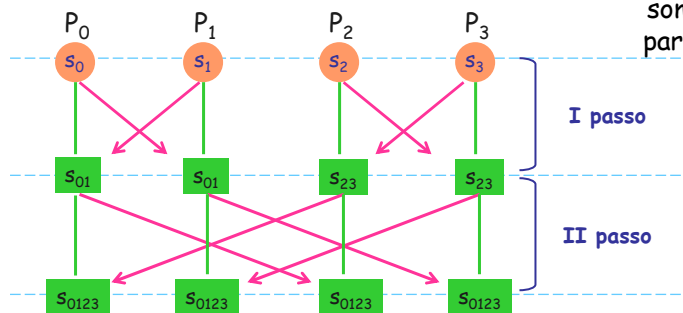
19

19

## III strategia

$p=4$

Calcolo  
somme  
parziali



Il numero di "passi" è 2

27/09/2023

Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

20

20

### III strategia

Ogni processore

- calcola la propria **somma parziale** .

Ad ogni passo

- coppie distinte di processori  
comunicano contemporaneamente:
  - in ogni coppia i processori  
si scambiano le proprie **somme parziali**

Il risultato è in **tutti** i processori

Operazioni concorrenti

27/09/2023

Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

21

21

### I, II, III strategia

Ogni processore calcola la sua **somma parziale**  
ed **invia** tale valore agli altri processori in  
modo da ottenere la somma totale

PARALLELISMO



COMUNICAZIONE  
TRA I PROCESSORI

27/09/2023

Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

22

22

## Algoritmo per la somma di $N=kp$ numeri su MIMD-Distributed Memory

### I Strategia

```
begin
  forall  $P_i$  ,  $0 \leq i \leq p-1$  do
     $sum_i := 0$ 
     $h := i * (n/p)$ 
    for  $j = h$  to  $h+(n/p)-1$  do
       $sum_i := sum_i + a_j$ 
    endfor
    if  $P_0$  then
      for  $k = 1$  to  $p-1$  do
        recv( $sum_k$  ,  $P_k$  )
         $sumtot := sumtot + sum_k$ 
      endfor
    else if  $P_i$  then
      send( $sum_i$  ,  $P_0$ )
    endif
  endforall
end
```

27/09/2023

23

23

## Uno strumento software per lo sviluppo di algoritmi in ambiente di calcolo MIMD-Distributed Memory

**Message Passing Interface**  
**MPI**

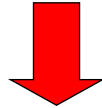
27/09/2023

Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

24

24

... ogni processore può conoscere i dati nella memoria di un altro processore o far conoscere i propri, attraverso il trasferimento di dati.



*Message Passing* :  
modello per la progettazione  
di software in  
ambiente di Calcolo Parallelo.

La necessità di rendere portabile  
il modello *Message Passing*  
ha condotto alla definizione  
e all'implementazione  
di un ambiente standard.



**Message Passing Interface**  
**MPI**

## Problema



Valutare l'efficienza di un algoritmo  
in ambiente di calcolo parallelo



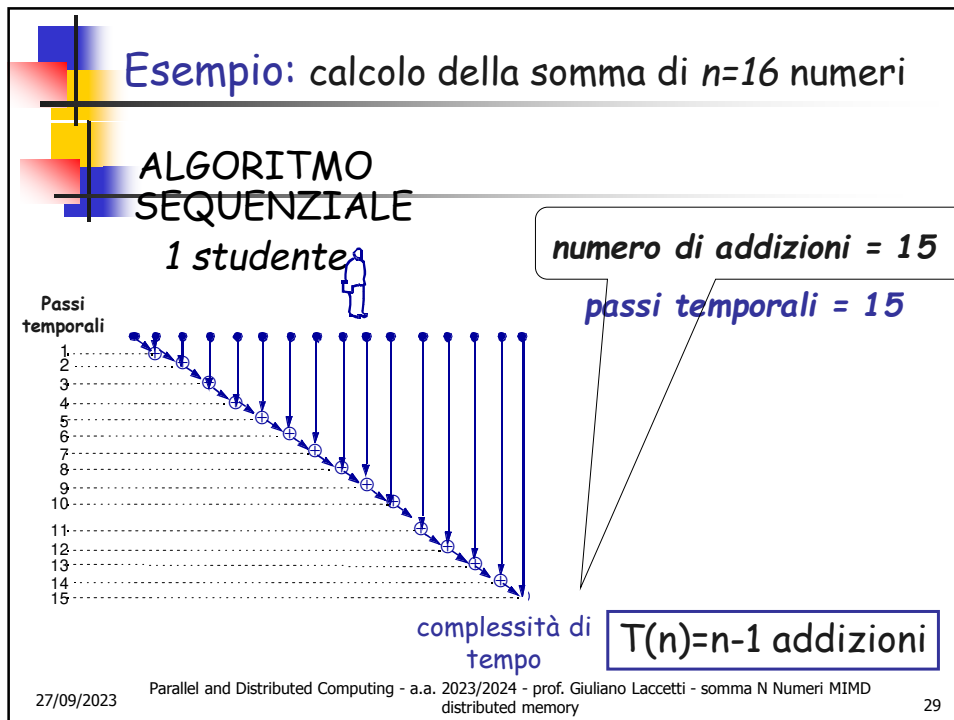
Cosa si intende per "EFFICIENZA" ?

## Efficienza di un algoritmo sequenziale



- COMPLESSITA' di TEMPO  $T(n)$   
Numero di operazioni eseguite dall'algoritmo

- COMPLESSITA' di SPAZIO  $S(n)$   
Numero di variabili utilizzate dall'algoritmo



29

In un algoritmo sequenziale

Il numero complessivo di operazioni  
determina *anche*  
il numero dei passi temporali  
(Il tempo di esecuzione)

27/09/2023 Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory 30

30

In un software sequenziale



L'efficienza del software  
dipende dal  
tempo di esecuzione  
delle  $T(n)$  operazioni fl.p.

Domanda



Cosa si intende  
per efficienza di un algoritmo  
in ambiente parallelo?



## Domanda



Si può ancora legare il tempo di esecuzione  
al numero delle operazioni che costituiscono  
l'algoritmo ?

33

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



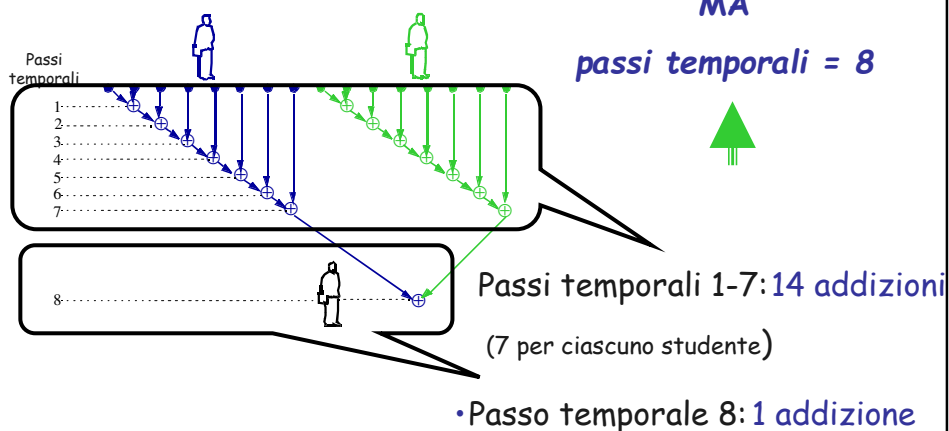
### ALGORITMO PARALLELO

2 studenti

numero di addizioni = 15

MA

passi temporali = 8



34

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

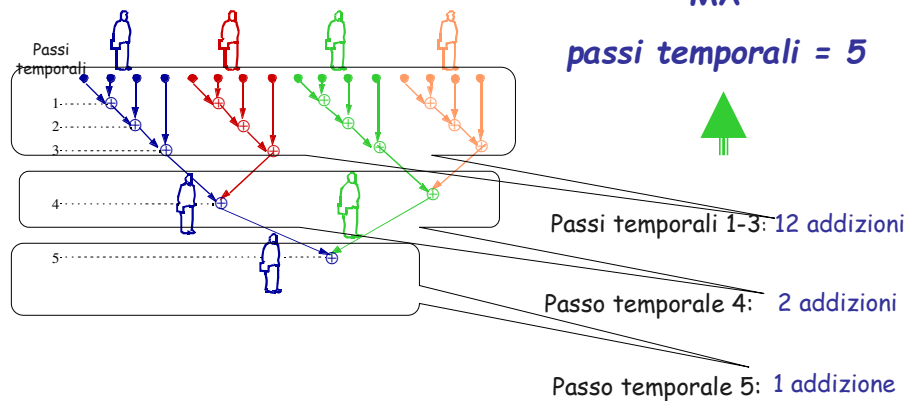
## ALGORITMO PARALLELO

4 studenti

numero di addizioni = 15

MA

passi temporali = 5



27/09/2023

Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

35

35

Nell'algoritmo parallelo della somma

Il numero delle operazioni  
**non è legato**  
al numero dei passi temporali

27/09/2023

Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

36

36

## Infatti ...



Un calcolatore parallelo è in grado di eseguire più operazioni

**concorrentemente**

(allo stesso passo temporale)



Il tempo di esecuzione **non è proporzionale** alla complessità di tempo  
(ovvero **non dipende soltanto** dal numero di operazioni fl. p. effettuate)



La complessità di tempo **non è adatta** a misurare  
l'efficienza di un algoritmo parallelo

## ... e allora



Che cosa si intende per  
efficienza  
di un algoritmo  
in ambiente parallelo?

... e allora



Che cosa si intende per  
efficienza  
di un algoritmo  
in ambiente parallelo?

Da questo momento ci riferiremo a una macchina che sia in grado di eseguire l'operatore



27/09/2023

Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD  
distributed memory

39

39

Esempio:



Se  $p$  indica il numero di processori,  $p=1, 2, 3, \dots$

$T(p)$  = tempo di esecuzione su  $p$  processori

$p = 2$



Ci aspettiamo che  $T(1)$  sia il doppio di  $T(2)$ :

$$\frac{T(1)}{T(2)} = 2$$

27/09/2023

Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD  
distributed memory

40

40

## Esempio:

Se  $p$  indica il numero di processori,  $p=1, 2, 3, \dots$

$T(p)$  = tempo di esecuzione su  $p$  processori

$$p = 4$$



Ci aspettiamo che  $T(1)$  sia il quadruplo di  $T(4)$ :

$$\frac{T(1)}{T(4)} = 4$$

## In generale

Con  $p$  processori ci aspettiamo che

$T(1)$  sia  $p$  volte  $T(p)$

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

ovvero

ci aspettiamo di ridurre  $p$   
volte il tempo di esecuzione

## IDEA

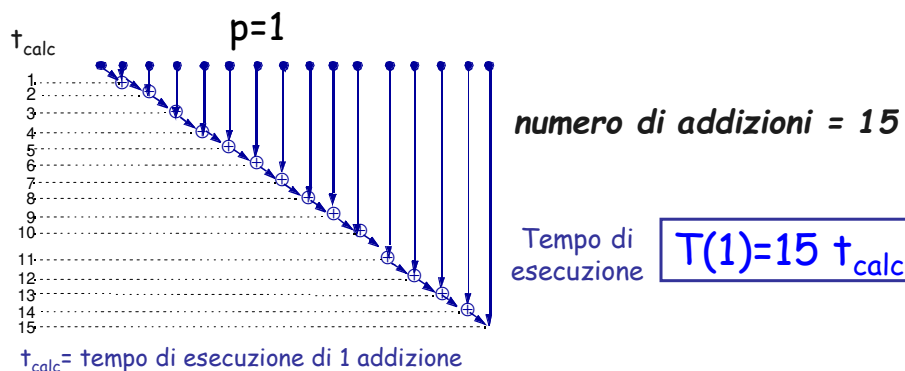


Misuriamo di quanto si riduce il tempo di  
esecuzione su  $p$  processori  
rispetto al tempo di esecuzione  
su  $1$  processore...

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



### ALGORITMO SEQUENZIALE

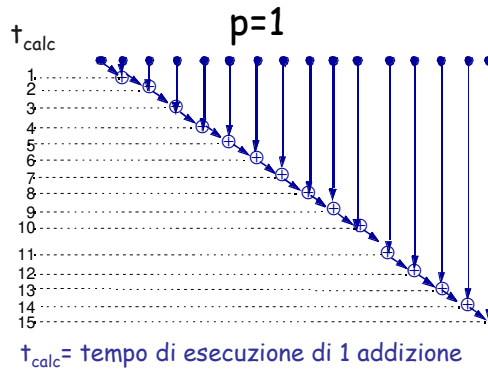


## Esempio: calcolo della somma di $n=16$ numeri



### ALGORITMO SEQUENZIALE

Diciamo che un operatore DIPENDE da un altro se la sua esecuzione è subordinata all'esecuzione dell'altro.



numero di addizioni = 15

Tempo di esecuzione

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

27/09/2023

Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

45

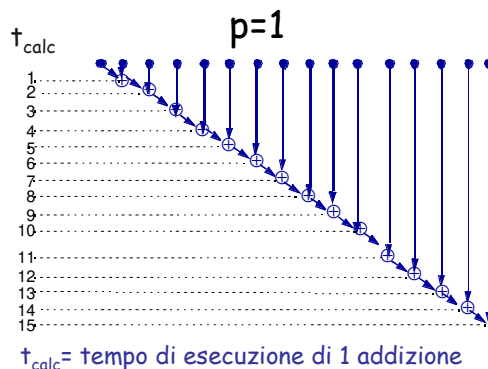
45

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



### ALGORITMO SEQUENZIALE

In questo caso, dopo il primo, ogni operatore dipende da tutti i precedenti.



numero di addizioni = 15

Tempo di esecuzione

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

27/09/2023

Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

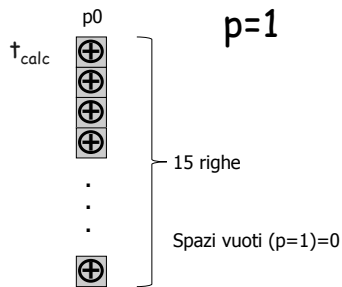
46

46

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



### ALGORITMO SEQUENZIALE



In questo caso, dopo il primo, ogni operatore dipende da tutti i precedenti. Disegniamo uno sotto l'altro gli operatori dipendenti

**numero di addizioni = 15**

**$righe(1) = 15$**

Tempo di esecuzione

$$T(1) = 15 t_{calc}$$

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

**$spazi\_vuoti(1) = 0$**

27/09/2023

Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

47

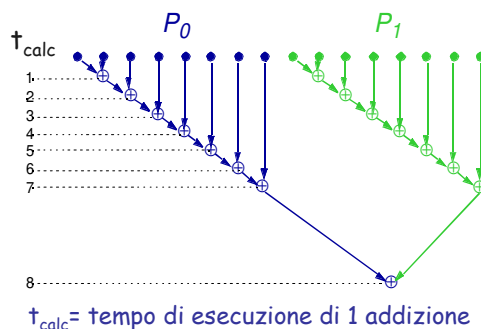
47

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



### ALGORITMO PARALLELO

$p=2$



In questo caso, per 7 passi due operatori possono essere eseguiti contemporaneamente: sono INDIPENDENTI tra loro

**numero di addizioni = 15**

$$T(2) = 8 t_{calc}$$

27/09/2023

Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

48

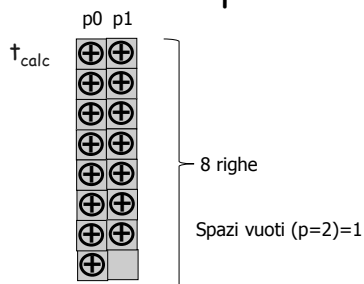
48



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

### ALGORITMO PARALLELO

$p=2$



Disegniamo uno a fianco all'altro gli operatori che sono **INDIPENDENTI** tra loro, cioè che possono essere eseguiti **concorrentemente**

**numero di addizioni = 15**

**$righe(2) = 8$**

$$T(2) = 8 t_{calc}$$

**$spazi\_vuoti(2) = 1$**

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

27/09/2023

Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

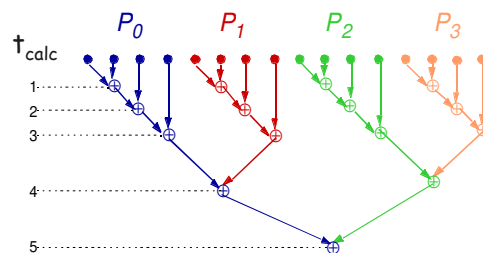
49

49

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

### ALGORITMO PARALLELO

$p=4$



**numero di addizioni = 15**

$$T(4) = 5 t_{calc}$$

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

27/09/2023

Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

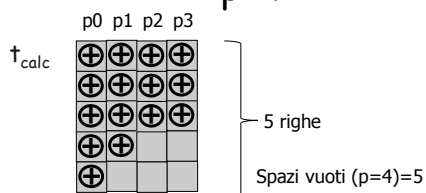
50

50

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

## ALGORITMO PARALLELO

$p=4$



**numero di addizioni = 15**

$$righe(4) = 5$$

$$T(4) = 5 t_{calc}$$

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

$$spazi\_vuoti(4) = 5$$

27/09/2023

Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

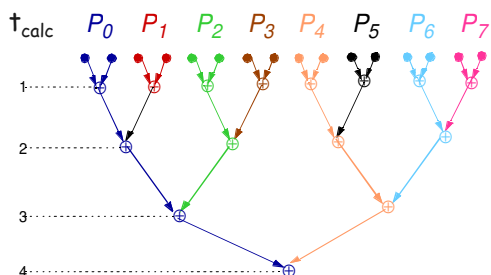
51

51

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

## ALGORITMO PARALLELO

$p=8$



**numero di addizioni = 15**

$$T(8) = 4 t_{calc}$$

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

27/09/2023

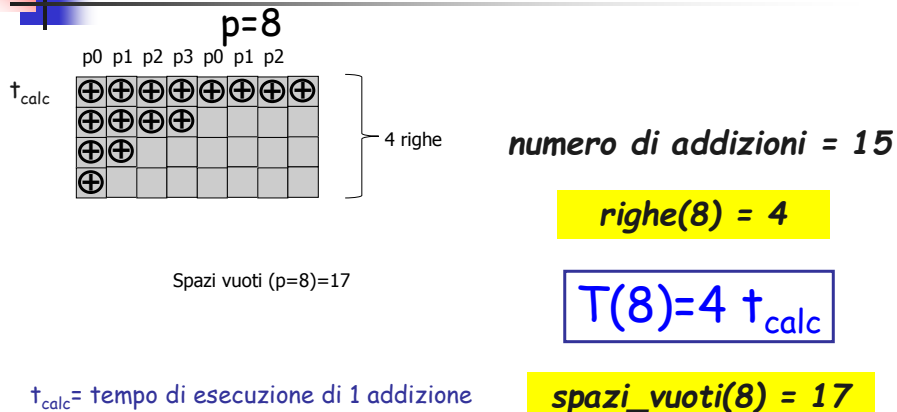
Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

52

52

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

### ALGORITMO PARALLELO



27/09/2023

Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

53

53

## In sintesi...

In generale possiamo dire che per qualsiasi algoritmo parallelo descritto attraverso una tabella di operatori come quelle viste

$$T(p) = righe(p) t_{calc}$$

$p$	$T_p$
1	$15 t_{calc}$
2	$8 t_{calc}$
4	$5 t_{calc}$
8	$4 t_{calc}$
$2^k$	$? t_{calc}$

27/09/2023

Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

54

54

## In sintesi...

In generale possiamo dire che per qualsiasi algoritmo parallelo descritto attraverso una tabella di operatori come quelle viste

$$T(p) = \text{righe}(p) \cdot t_{\text{calc}}$$

p	$T_p$
1	$15 t_{\text{calc}}$
2	$8 t_{\text{calc}}$
4	$5 t_{\text{calc}}$
8	$4 t_{\text{calc}}$
$2^k$	$? t_{\text{calc}}$



In generale quanto vale  $T(p)$  per l'algoritmo della somma?

## In generale: calcolo di $T(p)$

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

$$\left. \begin{array}{l} p=1 \quad T(1)=15 t_{\text{calc}} \\ p=2 \quad T(2)=8 t_{\text{calc}} = (7+1) t_{\text{calc}} \\ p=4 \quad T(4)=5 t_{\text{calc}} = (3+2) t_{\text{calc}} \\ p=8 \quad T(8)=4 t_{\text{calc}} = (1+3) t_{\text{calc}} \end{array} \right\} n = 16$$

$$T(p) = \left( \frac{n}{p} - 1 + \log_2 p \right) t_{\text{calc}}$$

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

Domanda...



p	$T_p$
1	$15t_{calc}$
2	$8t_{calc}$
4	$5t_{calc}$
8	$4t_{calc}$

Qual è l'algoritmo che impiega meno tempo?



Quanto è più veloce di quello sequenziale?

27/09/2023

Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

57

57

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



p	$T(p)$	$\frac{T(1)}{T(p)}$
1	$15t_{calc}$	1.00
2	$8t_{calc}$	1.88
4	$5t_{calc}$	3.00
8	$4t_{calc}$	3.75

Maggiore riduzione del tempo ovvero maggiore aumento della velocità

L'algoritmo su 8 processori è il più veloce  
E' più veloce di 3.75 volte di quello su 1 processore

27/09/2023

Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

58

58

## Speed-up



Si definisce il rapporto  $T(1)$  su  $T(p)$

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

Lo speed up misura la riduzione del tempo di esecuzione rispetto all'algoritmo su 1 processore

$$S(p) < p$$

$$\left( \begin{array}{l} \text{SPEEDUP IDEALE} \\ S^{\text{ideale}}(p) = p \end{array} \right)$$

## Speed-up



Si definisce il rapporto  $T(1)$  su  $T(p)$

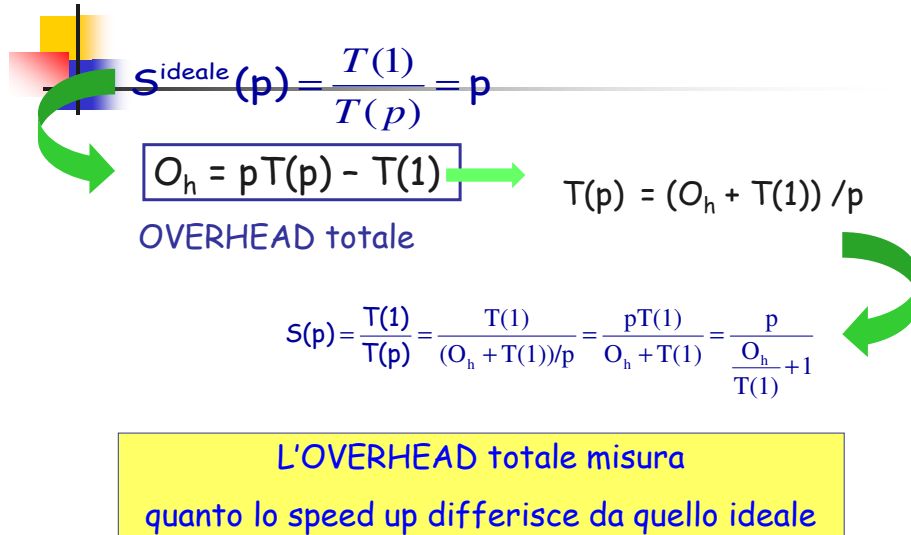
$$S(p) = \frac{\text{righe}(1)}{\text{righe}(p)}$$

Lo speed up misura la riduzione del tempo di esecuzione rispetto all'algoritmo su 1 processore

$$S(p) < p$$

$$\left( \begin{array}{l} \text{SPEEDUP IDEALE} \\ S^{\text{ideale}}(p) = p \end{array} \right)$$

## Osservazione



$$s^{ideale}(p) = \frac{T(1)}{T(p)} = p$$

$$O_h = pT(p) - T(1)$$

OVERHEAD totale

$$T(p) = (O_h + T(1)) / p$$

$$s(p) = \frac{T(1)}{T(p)} = \frac{T(1)}{(O_h + T(1))/p} = \frac{pT(1)}{O_h + T(1)} = \frac{p}{\frac{O_h}{T(1)} + 1}$$

L'OVERHEAD totale misura  
quanto lo speed up differisce da quello ideale

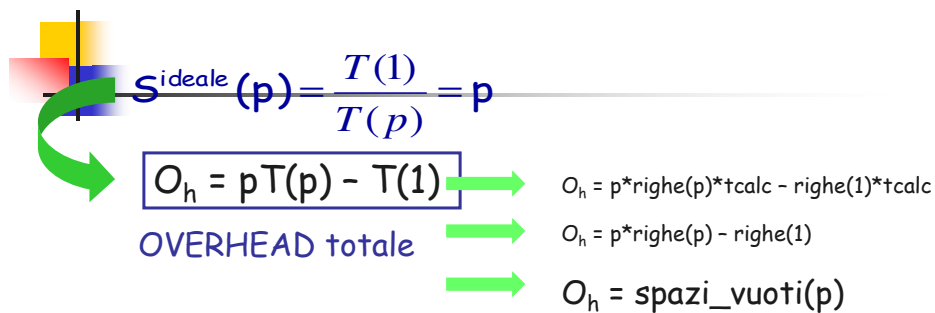
27/09/2023

Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

61

61

## Osservazione



$$s^{ideale}(p) = \frac{T(1)}{T(p)} = p$$

$$O_h = pT(p) - T(1)$$

OVERHEAD totale

$$O_h = p \cdot \text{righe}(p) \cdot t_{\text{calc}} - \text{righe}(1) \cdot t_{\text{calc}}$$

$$O_h = p \cdot \text{righe}(p) - \text{righe}(1)$$

$$O_h = \text{spazi\_vuoti}(p)$$

L'OVERHEAD totale misura  
quanto lo speed up differisce da quello ideale

27/09/2023

Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD distributed memory

62

62

## Osservazione

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

$$O_h = pT(p) - T(1)$$

OVERHEAD totale

$$O_h = p \cdot \text{righe}(p) \cdot t_{calc} - \text{righe}(1) \cdot t_{calc}$$

$$O_h = p \cdot \text{righe}(p) - \text{righe}(1)$$

$$O_h = \text{spazi\_vuoti}(p)$$

La rappresentazione degli operatori in tabella dà immediate informazioni sul tempo d'esecuzione e sull'overhead totale dell'algoritmo!

L'OVERHEAD totale misura  
quanto lo speed up differisce da quello ideale

## L'overhead nell'algoritmo parallelo della somma

$$T(1) = n-1$$

$$T(p) = n/p - 1 + \log_2 p$$

$$O_h = p T(p) - T(1) = p (n/p + \log_2 p) - (n-1) =$$

$$= n + p \log_2 p - n + 1 = O(p \log_2 p)$$

p	$O_h$
2	2
4	8
8	24
$2^k$	$p \log_2 p$

Al crescere di p  
l'overhead aumenta!



Quindi

Se si vuole calcolare la somma di 16 numeri  
nel minor tempo possibile


l'algoritmo su 8 processori è da preferire

Infatti, aumentando il numero di processori  
si riduce

il tempo impiegato per eseguire le operazioni  
richieste

MA....

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



p	Speed-up ottenuto	Speed-up ideale
2	1.88	2
4	3.00	4
8	3.75	8

Lo speed-up su 8 processori è il maggiore

**MA**

Lo speed-up su 2 processori è "il più vicino"  
allo speed-up ideale...

Cioè



Ho utilizzato 8 processori per  
avere un incremento  
di appena 4 volte

In altre parole



... lo speed up non basta a  
fornire informazioni sull'efficienza  
dell'algoritmo parallelo!

... e allora ?

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



... se si rapporta lo speed-up al numero di  
processori...


p	$S(p)$	$\frac{S(p)}{p}$
2	1.88	0.94
4	3.00	0.75
8	3.75	0.47

Rapporto più grande



maggiore sfruttamento dei  
processori per  $p=2$

In altre parole



~~L'utilizzo di un maggior numero di processori NON~~  
è sempre una garanzia di sviluppo di algoritmi  
paralleli "efficaci"

OVVERO

Di algoritmi che sfruttano  
tutte le risorse della macchina parallela !



Come misurare se e quanto  
è stata sfruttato il calcolatore parallelo ?

# Efficienza

Si definisce Efficienza  $E(p)$  il rapporto  
 $S(p)$  diviso  $p$

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

che misura quanto l'algoritmo sfrutta il parallelismo  
del calcolatore

## EFFICIENZA IDEALE

$$E^{\text{ideale}}(p) = \frac{S^{\text{ideale}}(p)}{p} = 1$$

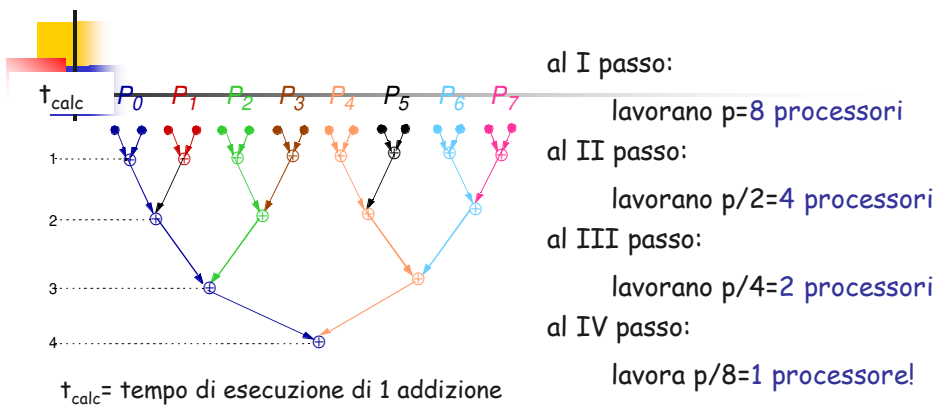
27/09/2023

Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD  
distributed memory

73

73

## Infatti, per la somma con $p=8$



Ad ogni passo si dimezza il numero  
di processori attivi

27/09/2023

Parallel and Distributed Computing - a.a. 2023/2024 - prof. Giuliano Laccetti - somma N Numeri MIMD  
distributed memory

74

74



Fine Lezione