Valutazione dell'efficienza di algoritmi e software in ambiente parallelo

parte 1

PARALLEL AND DISTRIBUTED COMPUTING

PROF. G. LACCETTI

A.A. 2023/2024

1

Problema

Valutare l'efficienza di un algoritmo

in ambiente di calcolo parallelo

Cosa si intende per "EFFICIENZA" ?

2

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

3

In un algoritmo sequenziale

Il numero complessivo di operazioni

determina anche

il numero dei passi temporali

(Il tempo di esecuzione)

4

In un software sequenziale

5

L'efficienza del software dipende dal

tempo di esecuzione

delle T(n) operazioni fl.p.

5

Domanda

6

Cosa si intende

per efficienza di un algoritmo

in ambiente parallelo?

6

Nell'algoritmo parallelo della somma…

7

Il numero delle operazioni

non è legato

al numero dei passi temporali

7

Infatti…

8

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

8

In generale

9

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

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

9

Esempio: Somma di n=16 numeri

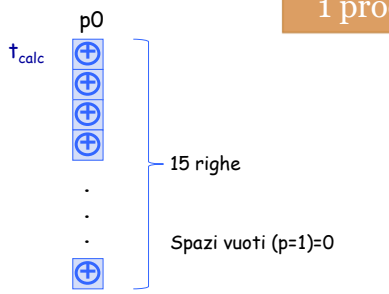
10

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

Disegniamo uno sotto l'altro gli operatori dipendenti

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

1 processore



righe(1) = 15

Tempo di esecuzione $T(1)=15\ t_{calc}$

spazi_vuoti(1) = 0

t_{calc}= tempo di esecuzione di 1 addizione

10

Esempio: Somma di n=16 numeri

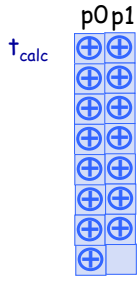
11

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

Disegniamo uno sotto l'altro gli operatori dipendenti

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

2 processori



righe(2) = 8

Tempo di esecuzione $T(2)=8\ t_{calc}$

spazi_vuoti(2) = 1

t_{calc}= tempo di esecuzione di 1 addizione

11

Esempio: Somma di n=16 numeri

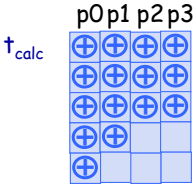
12

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

Disegniamo uno sotto l'altro gli operatori dipendenti

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

4 processori



righe(4) = 5

Tempo di esecuzione $T(2)=5\ t_{calc}$

spazi_vuoti(4) = 5

t_{calc}= tempo di esecuzione di 1 addizione

12

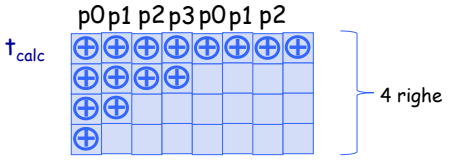
Esempio: Somma di n=16 numeri

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

Disegniamo uno sotto l'altro gli operatori dipendenti

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

8 processori



Spazi vuoti (p=8)=17

righe(8) = 4

Tempo di esecuzione $T(8)=4\ t_{calc}$

spazi_vuoti(8) = 17

t_{calc}= tempo di esecuzione di 1 addizione

13

Speed Up

Si definisce SPEED UP il rapporto

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

misura

la riduzione del tempo di esecuzione rispetto all'algoritmo su 1 processore

E in generale è

$$S(p) < p = S^{ideale}(p)$$

14

Speed Up

15

Si definisce SPEED UP il rapporto

Possiamo scrivere anche

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

misura

la riduzione del tempo di esecuzione rispetto all'algoritmo su 1 processore

E in generale è

$$S(p) < p = S^{ideale}(p)$$

15

Speed Up

16

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

OVERHEAD totale

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

L'OVERHEAD totale misura

quanto lo speed up differisce da quello ideale

16

Speed Up

17

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

OVERHEAD totale

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

- O_h (p)= p*righe(p)*tcalc – righe(1)*tcalc
- O_h (p)= p*righe(p) – righe(1)
- O_h (p)= spazi_vuoti(p)

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

17

Speed Up

18

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

OVERHEAD totale

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

- O_h (p)= p*righe(p)*tcalc – righe(1)*tcalc
- O_h (p)= p*righe(p) – righe(1)
- O_h (p)= spazi_vuoti(p)

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

18

Speed Up

19

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

OVERHEAD totale

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

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

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

19

Quindi

20

Se si vuole calcolare la somma di 16 numeri

nel minor tempo possibile

l'algoritmo su 8 processori è da preferire

Aumentando il numero di processori

si riduce

il tempo impiegato per eseguire le operazioni richieste

20

Esempio: Somma di n=16 numeri

21

p	S(p)	$S^{Ideale}(p)$
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...

21

Esempio: Somma di n=16 numeri

22

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

Lo speed-up su 8 processori è il maggiore

MA

maggiore sfruttamento dei processori per p=2

22

In altre parole…

23

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 stato "sfruttato" il calcolatore parallelo ?

Efficienza

24

Si definisce EFFICIENZA il rapporto

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

misura

quanto l'algoritmo sfrutta il parallelismo del calcolatore

E in generale è

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

Efficienza ed Overhead totale

In generale, ricordando le definizioni

$$\begin{cases} E(p) = \dfrac{S(p)}{p} \\ O_h = pT(p) - T(1) \end{cases}$$

$$E(p) = \frac{S(p)}{p} = \frac{T(1)}{pT(p)} = \frac{T(1)}{O_h + T(1)} = \frac{1}{\frac{O_h}{T(1)} + 1}$$

25

Domanda

E' possibile ottenere

speed-up prossimi allo speed-up ideale ?

26

T(1) si può decomporre in 2 parti:

27

una parte relativa alle operazioni che sono eseguite esclusivamente in sequenziale

una parte relativa alle operazioni che possono essere eseguite concorrentemente

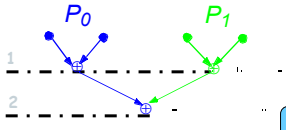
T_s T_c

$T(1) = T_s + T_c$

27

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

28



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

$T(1) = T_s + T_c$

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

$T_s = 1\ t_{calc}$

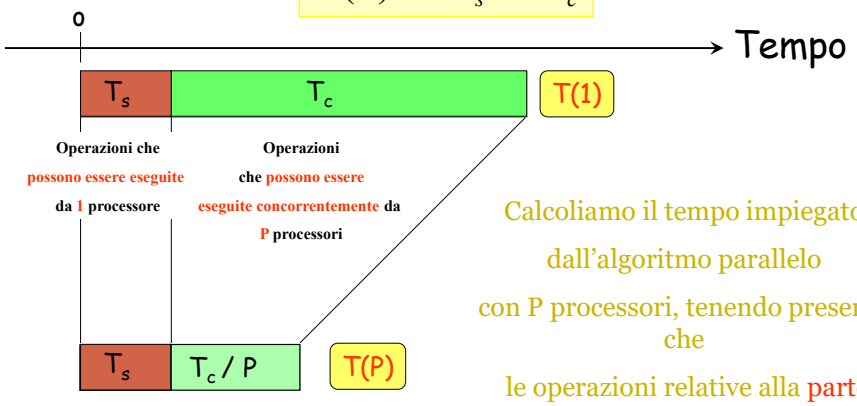
Operazione che deve essere eseguita in sequenziale

$T_c = 2\ t_{calc}$

Operazioni che possono essere eseguite in parallelo

28

$$T(1) = T_s + T_c$$



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

Calcoliamo il tempo impiegato

dall'algoritmo parallelo

con P processori, tenendo presente che

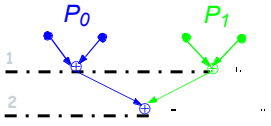
le operazioni relative alla parte parallela

sono ora seguite concorrentemente

dai P processori

29

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



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

$T(1) = T_s + T_c/p$

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

$T_s = 1\ t_{calc}$

Operazione che deve essere eseguita in sequenziale

$T_c /p = 2/2 = 1\ t_{calc}$

Operazioni che possono essere eseguite in parallelo

30

Calcoliamo lo speed-up:

Somma di n =4 su p=2,

$T(1) = 3\ t_{calc}$

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

$$S(2) = \frac{T(1)}{T(p)} = \frac{3t_{calc}}{2t_{calc}} = 1{,}5 < 2 = S^{ideale}(p)$$

Nel calcolare lo speed up...

... abbiamo calcolato di

quanto si riduce il tempo impiegato

da un solo processore se

le operazioni concorrenti

sono eseguite da p processori

In generale

33

Il tempo di esecuzione di un algoritmo parallelo, distribuito su p processori, comprende 2 componenti:

- Ts tempo per eseguire la parte seriale
- Tc/p tempo per eseguire la parte parallela

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

33

Overhead totale O_h

34

$O_h = pT(p) - T(1)$ dove $T(p) = T_s + \frac{T_c}{p}$

posto $T(1)=1$, $T_s=\alpha$, $T_c=1-\alpha$

$$O_h = (p-1)\alpha$$

Con **α** parte seriale
cioè la parte non parallelizzabile
dell'algoritmo

Osservazione:

O_h dipende da p e da α

34

Quindi l'efficienza...

35

$$\begin{cases} E(p) = \dfrac{S(p)}{p} = \dfrac{1}{O_h + 1} \\ O_h = (p-1)\alpha \end{cases}$$

$$E(p) = \frac{1}{O_h + 1} = \frac{1}{(p-1)\alpha + 1}$$

35

...e lo Speed Up

36

$$E(p) = \frac{1}{(p-1)\alpha + 1}$$

$$S(p) = p \cdot E(p) = \frac{1}{\alpha + \dfrac{1-\alpha}{p}}$$

Legge di Ware (Amdahl)

36

...e lo Speed Up

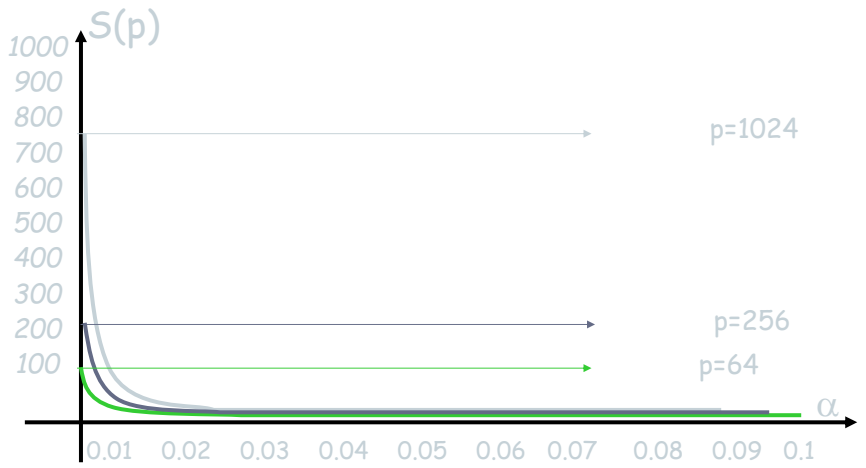
Analizziamo l'andamento dello speed-up all'aumentare del numero p di processori

$$S(p) = \frac{1}{\alpha + \frac{(1-\alpha)}{p}} \longrightarrow \frac{1}{\alpha}$$

Il valore asintotico di S(p) è $\frac{1}{\alpha}$

37

Andamento speed up secondo la legge di Amdahl



Una "piccola" parte sequenziale può degradare fortemente lo speed-up, quando il numero di processori è sufficientemente elevato!

38

Esempio 1 (n fissato e p variabile)

39

Applichiamo la legge di Amdahl con n =32 e p =2, 4, 8, 16

α_1 = frazione di T_1 eseguita da un solo processore (sequenziale)

α_p = frazione eseguita con parallelismo *totale* p

Al crescere del numero p di processori…

p	α_1	α_p	S(p)	E(p)
2	0,032	0,968	1,9	0,95
4	0,032	0,903	3,4	0,85
8	0,032	0,775	5,1	0,6
16	0,032	0,516	6,2	0,3

Speed up ed efficienza degradano perché la parte parallela diminuisce!

39

In sintesi

40

Se la dimensione n del problema è fissata,
al crescere del numero p di processori,
non solo non si riescono ad ottenere
speed up vicini a quello ideale

MA

Le prestazioni peggiorano!

(non conviene utilizzare un maggior numero di processori!!)

40

Esempio 2 (n aumenta e p fissato)

41

Applichiamo la legge di Amdahl con *p=2* e *n* = 8, 16, 32, 64

α_1 = frazione di T_1 eseguita da un solo processore (sequenziale)

α_p = frazione eseguita con parallelismo *totale* p

Al crescere della dimensione n…

n	α	1-α	S(2,*n*)	E(2,*n*)
8	0,14	0,86	1,75	0,875
16	0,06	0,94	1,8	0,9
32	0,03	0,97	1,9	0,96
64	0,01	0,99	1,96	0,99

La parte *sequenziale*

tende a zero!

41

Esempio 2 (n aumenta e p fissato)

42

Applichiamo la legge di Amdahl con *p=2* e *n* = 8, 16, 32, 64

α_1 = frazione di T_1 eseguita da un solo processore (sequenziale)

α_p = frazione eseguita con parallelismo *totale* p

Al crescere della dimensione n…

n	α	1-α	S(2,*n*)	E(2,*n*)
8	0,14	0,86	1,75	0,875
16	0,06	0,94	1,8	0,9
32	0,03	0,97	1,9	0,96
64	0,01	0,99	1,96	0,99

La parte *parallela* cresce, *tende a 1*

42

Esempio 2 (n aumenta e p fissato)

Applichiamo la legge di Amdahl con *p=2* e *n* = 8, 16, 32, 64

α_1 = frazione di T_1 eseguita da un solo processore (sequenziale)

α_p = frazione eseguita con parallelismo *totale* p

Al crescere della dimensione n…

n	α	1-α	S(2,*n*)	E(2,*n*)
8	0,14	0,86	1,75	0,875
16	0,06	0,94	1,8	0,9
32	0,03	0,97	1,9	0,96
64	0,01	0,99	1,96	0,99

Speed up ed efficienza sono "costanti"!

Infatti

Se p è fissato, al crescere della dimensione *n* del problema…

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

$\alpha \to 0$ $\quad$ $(1-\alpha) \to 0$

Il valore asintotico di

S(p) è p

$(S^{ideale}(p) = p)$

In sintesi

45

Fissando il numero p di processori e aumentando la dimensione del problema si possono ottenere speed up prossimi a quello ideale

MA

Non è possibile aumentare in maniera indefinita la dimensione *n* del problema: le risorse (hardware) sono limitate!

45

Quindi

46

Secondo la legge di Ware…

Aumentando il numero p di processori e mantenendo fissata la dimensione *n* del problema si riesce ad utilizzare in maniera efficiente l'ambiente di calcolo parallelo, se $p \leq p_0$

Aumentando la dimensione *n* del problema e mantenendo fisso il numero p di processori le prestazioni dell'algoritmo parallelo non degradano se $n >= n_0$

46

Domanda

47

Calcolando: $S(p) = \dfrac{1}{\alpha + \dfrac{(1-\alpha)}{p}}$

Cosa succede se aumentiamo
il numero p di processori e
la dimensione *n* del problema ?

47

Esempio: Somma di n numeri

48

Applichiamo la legge di Amdhal con p= 2, 4, 8, 16

Consideriamo quindi n= 8 , 16, 32, 64

$\frac{n}{p} = 4$ è costante

n	p	α_1	α_p	S(p,*n*)	E(p,*n*)
8	2	0,14	0,86	1,75	0,875
16	4	0,06	0,8	3	0,75
32	8	0,03	0,77	5,1	0,64
64	16	0,01	0,76	9	0,56

La frazione di operazioni eseguite *in sequenziale* decresce!

48

Esempio: Somma di n numeri

49

Applichiamo la legge di Amdhal con p= 2, 4, 8, 16

Consideriamo quindi n= 8 , 16, 32, 64

$\frac{n}{p} = 4$ è costante

n	p	α_1	α_p	S(p,*n*)	E(p,*n*)
8	2	0,14	0,86	1,75	0,875
16	4	0,06	0,8	3	0,75
32	8	0,03	0,77	5,1	0,64
64	16	0,01	0,76	9	0,56

La frazione di operazioni eseguite ***in parallelo su tutti i p processori*** rimane quasi costante!

49

Esempio: Somma di n numeri

50

Applichiamo la legge di Amdhal con p= 2, 4, 8, 16

Consideriamo quindi n= 8 , 16, 32, 64

$\frac{n}{p} = 4$ è costante

n	p	α_1	α_p	S(p,*n*)	E(p,*n*)
8	2	0,14	0,86	1,75	0,875
16	4	0,06	0,8	3	0,75
32	8	0,03	0,77	5,1	0,64
64	16	0,01	0,76	9	0,56

Lo speed up AUMENTA!

50

Esempio: Somma di n numeri

Applichiamo la legge di Amdhal con p= 2, 4, 8, 16

Consideriamo quindi n= 8 , 16, 32, 64

$\frac{n}{p} = 4$ è costante

n	p	α_1	α_p	S(p,*n*)	E(p,*n*)
8	2	0,14	0,86	1,75	0,875
16	4	0,06	0,8	3	0,75
32	8	0,03	0,77	5,1	0,64
64	16	0,01	0,76	9	0,56

L'efficienza è "quasi costante"

Quindi

Aumentando sia **n** che **p**,
le prestazioni
dell'algoritmo parallelo
non degradano

Ma
aumentando sia **n** che **p**
cosa ci aspettiamo dall'algoritmo parallelo?

Quindi

Se p=2 ci si aspetta di calcolare nel tempo T(1,n) la somma di $2n$ numeri

$$T(1,n) = T(2,2n)$$

Se p=4 ci si aspetta di calcolare nel tempo T(1,n) la somma di $4n$ numeri

$$T(1,n) = T(4,4n)$$

In generale:

$$T(1,n) = T(2,2n) = T(4,4n) = \ldots = T(p,pn)$$

In generale per la somma…

$$T(1,n) = T(2,2n) = T(4,4n) = \ldots = T(p,pn)$$

Pertanto…

$$\frac{T(1,n)}{T(p,pn)} = 1 \iff p \cdot \frac{T(1,n)}{T(p,pn)} = p \iff \frac{T(1,pn)}{T(p,pn)} = p = S_p^{ideale}$$

..se si assume

$$pT(1,n) = T(1,pn)$$

Ovvero se si assume T(1,pn) uguale al tempo che si ottiene moltiplicando per p il tempo per risolvere su 1 processore il problema di dimensione n…

In generale per la somma…

$$SS(p,n) = \frac{T(1,pn)}{T(p,pn)} = \frac{pT(1,n)}{T(p,np)}$$

SPEEDUP SCALATO

55

In generale per la somma…

dividendo SS(p,n) per il numero di processori

$$ES(p,n) = \frac{SS(p,n)}{p} = \frac{T(1,n)}{T(p,pn)}$$

EFFICIENZA SCALATA

56