Dalla precedente lezione

Calcolatore parallelo

Principale obiettivo:
Riduzione efficiente dei tempi di esecuzione

Sistema ad arch. distribuita

Principale obiettivo:
Aggregazione efficiente di risorse esistenti

Differenti obiettivi

Parallelo vs distribuito

Calcolatore parallelo	Ambiente per il C.D.
• reti veloci	• reti lente
• risorse limitate	• risorse potenzialmente illimitate
• risorse dedicate e omogenee	• risorse condivise e disomogenee
• applicazione gestisce le risorse	• ambiente sw.gestisce le risorse
• costo hardware notevole	• costo hardware trascurabile
• overhead sw sistema < 5%	• overhead sw sistema > 20%
• presenza di vincoli temporali	• assenza di vincoli temporali
• es. Roadrunner	• es. SETI@home
• 16000 processori	• 5 milioni processori
• 1 Pflops	• 100 Tflops

Molte differenze ! ! !

Domanda

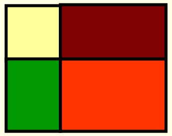
Quali sono le applicazioni "predisposte" al calcolo parallelo (o *naturalmente parallele*) ?

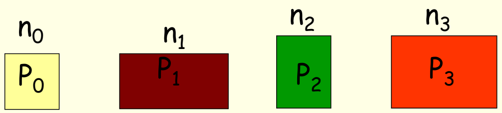
Quelle che riducono efficientemente il tempo di esecuzione !!

Che vuol dire " naturalmente parallela?"

Supponiamo che il problema si possa decomporre in
4 sottoproblemi indipendenti ma
di complessità diversa …

N
Numero di operazioni
del problema completo





n_i = numero di operazioni del sottoproblema i-mo

Assumiamo che

- i processori siano omogenei
- abbiano lo stesso carico di lavoro

(ipotesi non restrittive per un calcolatore parallelo)

Detto

t_i il tempo di esecuzione del i-mo problema
sul processore P_i

$t_i = K\, n_i$ (K costante per tutti i processori)

Inoltre, …

detto T_1 il tempo di esecuzione del problema
su 1 processore $T_1 = K N$ cioe' $N = T_1/K$

Supponiamo ora che:

$n_0 = 0.2\,N$ $n_1 = 0.1\,N$ $n_2 = 0.2\,N$ $n_3 = 0.5\,N$

$t_0 = K\, n_0 = 0.2\, K N = 0.2\, T_1$

analogamente $t_1 = 0.1\, T_1$ $t_2 = 0.2\, T_1$ $t_3 = 0.5\, T_1$

$T_4 = \max(t_i) = t_3 = 0.5\, T_1$

quindi

$$S_4 = \frac{T_1}{T_4} = \frac{T_1}{0.5\,T_1} = 2$$

Anche se i sottoproblemi sono indipendenti
lo speed up sui 4 proc. è estremamente basso !

riduzione non efficiente del tempo
(Problema non predisposto al calcolo parallelo)

Se invece

Si riesce a decomporre il problema
in 4 sottoproblemi equivalenti

n_0	n_1
n_2	n_3

se $n_0 = n_1 = n_2 = n_3$

$T_4 = t_0 = T_1/4$

ovvero

$$S_4 = \frac{T_1}{T_4} = \frac{T_1}{T_1/4} = 4$$

riduzione efficiente del tempo

…in questo caso l'applicazione è
naturalmente parallela

In definitiva…

Un'applicazione è naturalmente parallela se

$$\boxed{S_p \approx p \quad \Leftrightarrow \quad T_p \approx \frac{T_1}{p}}$$

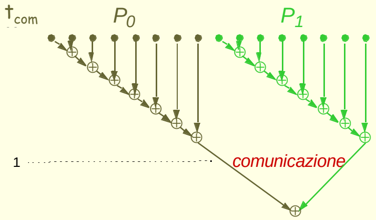
Domanda

Quali sono le applicazioni
"predisposte"
al calcolo distribuito

Quelle che aggregano efficientemente le
risorse di calcolo !!

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

ALGORITMO PARALLELO
p=2



L'algoritmo richiede 8 addizioni e 1 comunicazione

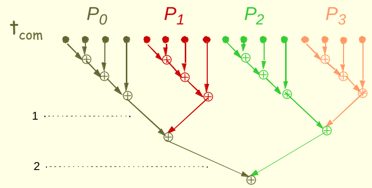
$$T_2=8\ t_{calc}+1t_{com}$$

t_{calc} = tempo di esecuzione unitario (di 1 op. fl.p.)

t_{com} = tempo di comunicazione unitario (di 1 dato fl.p. = 8 bytes)

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

ALGORITMO PARALLELO
p=4

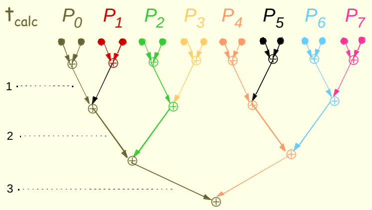


L'algoritmo richiede 5 addizioni e 2 comunicazioni

$$T_4=5\ t_{calc}+2t_{com}$$

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

ALGORITMO PARALLELO
p=8



L'algoritmo richiede 4 addizioni e 3 comunicazioni

$$T_8=4\ t_{calc}+3t_{com}$$

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

$T_1=15t_{calc}$

Supponendo $\frac{t_{com}}{t_{calc}}=0$ (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

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

p	T_p	S_p	E_p
2	$10t_{calc}$	1.50	0.75
4	$9t_{calc}$	1.67	0.42
8	$10t_{calc}$	1.50	0.19

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

Supponendo

$$\frac{t_{com}}{t_{calc}} = 4$$

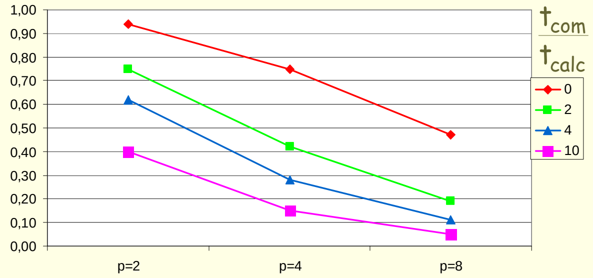
p	T_p	S_p	E_p
2	$12t_{calc}$	1.25	0.62
4	$13t_{calc}$	1.15	0.28
8	$16t_{calc}$	0.93	0.11

Supponendo

$$\frac{t_{com}}{t_{calc}} = 10$$

p	T_p	S_p	E_p
2	$18t_{calc}$	0.8	0.4
4	$25t_{calc}$	0.6	0.15
8	$34t_{calc}$	0.4	0.05

Efficienze



le prestazioni dell' algoritmo

possono cambiare notevolmente!

In generale...

Qual è l'impatto della comunicazione
sull'efficienza di
un algoritmo in ambiente parallelo/distribuito?

Impatto sull'efficienza

$$E_P = \frac{T_1}{PT_P} = \frac{T_1}{P(T_{comm} + T_{calc})}$$

Poiche' $T_{calc} \geq \frac{T_1}{P}$ $\Rightarrow$ $E_P \leq \frac{T_{calc}}{T_{comm} + T_{calc}}$

Cioe': $E_P \leq \frac{1}{1 + \frac{T_{comm}}{T_{calc}}} = \frac{1}{1 + OC_P}$

In GENERALE

Per l'overhead totale di comunicazione risulta …

$$OC_p = \frac{T_p^{com}}{T_p^{calc}} = \frac{t_{com}}{t_{calc}} \times \frac{N_{com}}{N_{calc}}$$

Dipendenza dall'ambiente (hw/sw) di calcolo

Dipendenza dall'algoritmo

Quanto vale t_{comm}/t_{calc} ?

IBM BlueGene
Bandw. = 2.8 GB/sec → tcomm = 1.4×10^{-9} sec
P.P. = 2.4 Gflops per proc. → tcalc = 0.41×10^{-9} sec
tcomm/tcalc = 3.4

Beowulf
Bandw. = 0.15 GB/sec → tcomm = 26×10^{-9} sec
P.P. = 2 Gflops per proc. → tcalc = 0.5×10^{-9} sec
tcomm/tcalc = 53.3

Ambiente C.D.
Bandw. = 0.001 GB/sec → tcomm = 4000×10^{-9} sec
P.P. = 2 Gflops per proc. → tcalc = 0.5×10^{-9} sec
tcomm/tcalc = 8000

Quanto vale N_{comm}/N_{calc} ?

Per l'algoritmo della somma:

P=2 ⇨ 1/8 = 0.12
P=4 ⇨ 2/5 = 0.4
P=8 ⇨ 3 /4 = 0.75

Quanto vale OC_p ?

IBM BlueGene
P=2 → OC_p = 3.4 x 0.12 = 0.41
P=4 → OC_p = 3.4 x 0.4 = 1.36
P=8 → OC_p = 3.4 x 0.75 = 2.55

Beowulf
P=2 → OC_p = 53.3 x 0.12 = 6.4
P=4 → OC_p = 53.3 x 0.4 = 21.3
P=8 → OC_p = 53.3 x 0.75 = 40

Ambiente C.D.
P=2 → OC_p = 8000 x 0.12 = 960
P=4 → OC_p = 8000 x 0.4 = 3200
P=8 → OC_p = 8000 x 0.75 = 6000

Overhead basso

Overhead alto

Quindi…

da $E_P < \frac{1}{1+OC_P}$

	E_P
OC=1	< 0.5
OC=10	< 0.1
OC=100	< 0.01
OC=1000	< 0.001

Somma su IBM BG

Somma su beowulf

Somma in ambiente C.D.

Impatto devastante!!

Aggregazione non efficiente delle risorse

VICEVERSA

Che OC_P si puo' tollerare se si vuole una data efficienza?

Efficienza vs OC

Da $E_P < \frac{1}{1+OC_P} \Rightarrow OC_P < \frac{1-E_P}{E_P}$

	OC
E_P=0.95	< 0.05
E_P=0.9	< 0.11
E_P=0.8	< 0.25
E_P=0.5	< 1

esempio: E_P=0.8 → OC_P < 0.25

Poiche' in un ambiente distribuito

$$OC_p = \frac{T_p^{com}}{T_p^{calc}} = \boxed{\frac{t_{com}}{t_{calc}}} \times \frac{N_{com}}{N_{calc}}$$

è maggiore di 8000

$$\frac{N_{com}}{N_{calc}} \approx 10^{-4}$$

In generale, se si vuole E_p=0.8

IBM BG

$\frac{t_{com}}{t_{calc}} = 3.4$ ⇒ $\frac{N_{com}}{N_{calc}} \approx 10^{-1}$

beowulf

$\frac{t_{com}}{t_{calc}} = 53.5$ ⇒ $\frac{N_{com}}{N_{calc}} \approx 10^{-2}$

Ambiente C.D.

$\frac{t_{com}}{t_{calc}} = 8000$ ⇒ $\frac{N_{com}}{N_{calc}} \approx 10^{-4}$

Quindi

IBM BG	beowulf	Ambiente C.D.
$\frac{N_{com}}{N_{calc}} \approx 10^{-1}$	$\frac{N_{com}}{N_{calc}} \approx 10^{-2}$	$\frac{N_{com}}{N_{calc}} \approx 10^{-4}$
⇓	⇓	⇓
Circa 1 comunicazione ogni 10 operazioni f.p.	Circa 1 comunicazione ogni 100 operazioni f.p.	Circa 1 comunicazione ogni 10000 operazioni f.p.

Numero di comunicazioni trascurabili!!!

In definitiva…

Quali sono le applicazioni “predisposte” al calcolo distribuito ?

⇓

quelle per cui il numero di comunicazioni è praticamente trascurabile rispetto al numero di operazioni

Osservazione

Supponiamo che

t_{comm}=0 (comunicazione gratis !!)

Possiamo risolvere un problema con poche comunicazioni in un ambiente di calcolo distribuito ed avere una efficienza elevata?

Osservazione

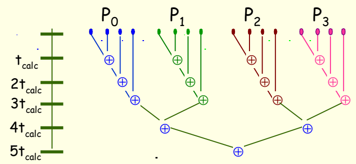
Per un Calcolatore Parallelo abbiamo assunto che:

- i processori sono omogenei
- hanno lo stesso carico di lavoro

In un ambiente per il Calcolo Distribuito
tali ipotesi non possono essere fatte

Riprendiamo l'esempio della somma

Esempio: P=4



Assumendo $t_{comm}=0$
per un calcolatore parallelo
si ha $T_4=5\ t_{calc}$

MA

Cio' vale solo se si assume che il tempo per una somma e' lo stesso per tutti i processori!!!!

In un ambiente per il C.D.

Processori eterogenei

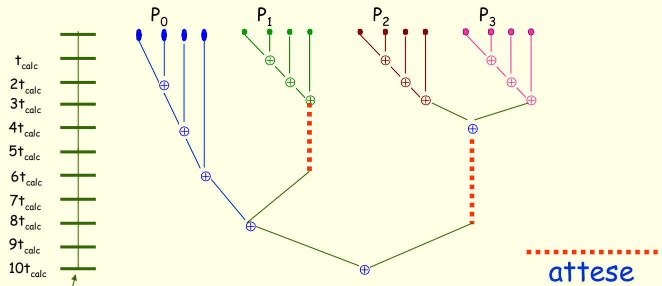
Risorse non dedicate

t_{calc} non e' lo stesso in tutti i processori

Assumiamo per esempio

$t^{(0)}_{calc} \sim 2\ t^{(i)}_{calc}$ (tempo per la somma su P_0 circa doppio rispetto agli altri)

Esecuzione in ambiente pr C.D.



$T_4=10t_{calc}$

L'intera esecuzione e' rallentata dal processore piu' lento
$S_4 = T_1/T_4 = 15/10 = 1.5$

In definitiva

Le comunicazioni sono
punti di sincronizzazione
negli algoritmi

In un ambiente distribuito
(dove le risorse non sono dedicate)
la presenza di comunicazioni tra i
processori rendono molto bassa
l'efficienza anche se si assume
$t_{comm} = 0$!!!

Quindi ...

Un'applicazione predisposta al
Calcolo Distribuito e'
un'applicazione in cui e'
completamente assente
la comunicazione tra i processori

Aggregazione efficiente di risorse
geograficamente e amministrativamente
distribuite

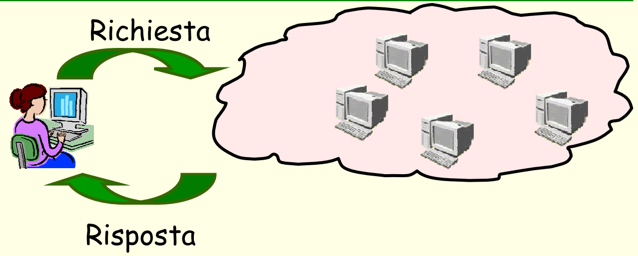
Domanda:

Calcolatore parallelo
- risorse omogene e dedicate
- reti di connessione veloci / memorie condivise

Sviluppo di applicazioni mediante
Message passing

Quale modello di programmazione
usare per sviluppare applicazioni distribuite?

Modello di programmazione per il C.D.



In un ambiente per il calcolo distribuito
l'utente dialoga con un ambiente software
richiedendogli un servizio
(modello client-server)

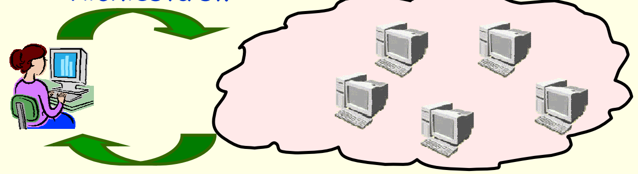
Modello client - server

L'ambiente software puo' fornire:
- servizi software (server software)
- servizi hardware (server hardware)
- entrambi i servizi hw e sw

All'interno del modello client-server si possono distinguere differenti modalita' di esecuzione

Esempio 1: server software

Richiesta sw



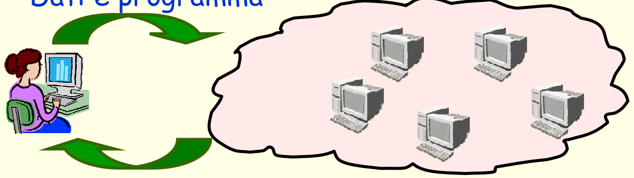
software

- il client richiede al server un programma
- il server fornisce al client il software richiesto
- il client esegue il calcolo su dati in suo possesso

(Code shipping)

Esempio 2: server hardware

Dati e programma



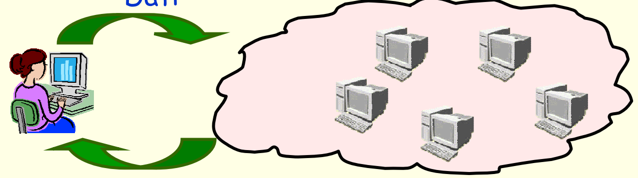
risultato

- il client fornisce al server dati e programma
- il server esegue il calcolo
- il server ritorna il risultato al client

(Proxy computing)

Esempio 3: server sw e hw

Dati



Risultato

- il client manda al server i dati
- il server elabora i dati localmente
- il server manda il risultato al client

(remote computing)

PROBLEMA:
prodotto di matrici

$$C = A \cdot B$$

in un ambiente di calcolo distribuito

Possibile algoritmo parallelo

suddivisione del problema

⇓

Suddivisione delle matrici a blocchi

⇓

Algoritmi a blocchi (es. SUMMA)

Osservazione

SUMMA richiede una
stretta sincronizzazione dei processori

⇓

Algoritmo sistolico

In definitiva

Buona efficienza parallela

⇕

Tutti i processori raggiungono i punti di sincronizzazione in circa lo stesso tempo

⇕

tempo per il calcolo di
C(I,J)=C(I,J)+A(I,K)B(K,J)
circa uguale in tutti i processori

⇕

Ambiente omogeneo e dedicato

Osservazione

Le ipotesi di

- Omogeneita'
- Dedicazione al calcolo

non possono essere fatte in un ambiente di calcolo distribuito

Rivisitazione dell'algoritmo

Obiettivo del C.D.

Aggregare efficientemente risorse computazionali per il calcolo dei vari $C(I,J)+A(I,K)B(K,J)$

Che significa?

Eliminare le sincronizzazioni tra i task paralleli

Prodotto a blocchi di due matrici

$$\begin{array}{|c|c|c|c|}\hline A_{00} & A_{01} & A_{02} & A_{03} \\ \hline A_{10} & A_{11} & A_{12} & A_{13} \\ \hline A_{20} & A_{21} & A_{22} & A_{23} \\ \hline A_{30} & A_{31} & A_{32} & A_{33} \\ \hline \end{array} \quad \begin{array}{|c|c|}\hline B_{00} & B_{01} \\ \hline B_{10} & B_{11} \\ \hline B_{20} & B_{21} \\ \hline B_{30} & B_{31} \\ \hline \end{array} \quad \longrightarrow \quad \begin{array}{|c|c|}\hline C_{00} & C_{01} \\ \hline C_{10} & C_{11} \\ \hline C_{20} & C_{21} \\ \hline C_{30} & C_{31} \\ \hline \end{array}$$

$$C(I,J) = \sum_{K=0}^{MB-1} A(I,K)B(K,J) \qquad \begin{array}{l} I = 0,..,NB-1 \\ J = 0,..,LB-1 \end{array}$$

Come eseguirlo in un ambiente di C.D. ?

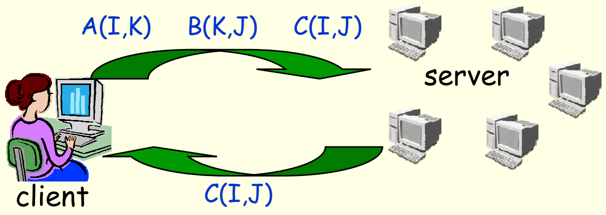
osservazione

$$C(I,J) = \sum_{K=0}^{MB-1} A(I,K)B(K,J) \qquad \begin{array}{l} I = 0,..,NB-1 \\ J = 0,..,LB-1 \end{array}$$

Ogni $C(I,J)$ puo' essere calcolato indipendentemente dagli altri

Gli unici parallelismi possibili sono sugli indici I e J (non sull'indice K)

in un ambiente di C.D.



- il client invia A(I,K) B(K,J) C(I,J)
- un server calcola C(I,J)=C(I,J)+A(I,K)B(K,J)
- il server invia il risultato C(I,J) al client

Problema

Con che ordine inviare i blocchi?

Prodotto a blocchi versione (I,J,K)

```
for I = 0,NB-1  (in parallelo)
        for J = 0,LB-1  (in parallelo)
                for K = 0,MB-1
                        C(I,J)=C(I,J)+A(I,K)B(K,J)
                endfor
        endfor
endfor
```

Prodotto a blocchi versione (I,K,J)

```
for I = 0,NB-1  (in parallelo)
        for K = 0,MB-1
                for J = 0,LB-1 (in parallelo)
                        C(I,J)=C(I,J)+A(I,K)B(K,J)
                endfor
        endfor
endfor
```

Prodotto a blocchi versione (K,I,J)

```
for K = 0,MB-1
     for I = 0,NB-1 (in parallelo)
          for J = 0,LB-1 (in parallelo)
               C(I,J)=C(I,J)+A(I,K)B(K,J)
          endfor
     endfor
endfor
```

Osservazione 1

Le altre versioni ottenute invertendo l'ordine degli indici I e J sono equivalenti alle precedenti tre

Esempio:

```
for I = 0,NB-1  (in parallelo)
  for J = 0,LB-1  (in parallelo)
    for K = 0,MB-1
      C(I,J)=C(I,J)+A(I,K)B(K,J)
    endfor
  endfor
endfor
```

```
for J = 0,LB-1  (in parallelo)
  for I = 0,NB-1  (in parallelo)
    for K = 0,MB-1
      C(I,J)=C(I,J)+A(I,K)B(K,J)
    endfor
  endfor
endfor
```

Osservazione 2

Che dimensione devono avere i blocchi A(I,K), B(K,J) e C(I,J) ?

In un ambiente di C.D. non sono note le caratteristiche delle risorse computazionali

Non e' possibile determinare una ripartizione uniforme del carico di lavoro prima dell'esecuzione

Possiamo supporre i blocchi quadrati di uguale dimensione

problemi

Che differenza c'e' tra le tre versioni?

Qual'e' la migliore per un ambiente di calcolo distribuito?

Versione (K,I,J)   (K esterno)

K=0

calcola in parallelo su I e J   C(I,J) = C(I,J)+A(I,0)B(0,J)

K=1

calcola in parallelo su I e J   C(I,J) = C(I,J)+A(I,1)B(1,J)

K=2

calcola in parallelo su I e J   C(I,J) = C(I,J)+A(I,2)B(2,J)

Sincronizzazione tra 2 successivi valori di K
tra tutti i task paralleli su I e J

Versione (I,J,K)   (K interno)

In parallelo su I e J esegui

K=0

calcola C(I,J) = C(I,J)+A(I,0)B(0,J)

K=1

calcola C(I,J) = C(I,J)+A(I,1)B(1,J)

K=2

calcola C(I,J) = C(I,J)+A(I,2)B(2,J)

Sincronizzazione tra 2 successivi valori di K
solo per una fissata coppia I e J

Versione (I,K,J)   (K in mezzo)

In parallelo su I esegui

K=0

calcola in parallelo su J C(I,J) = C(I,J)+A(I,0)B(0,J)

K=1

calcola in parallelo su J C(I,J) = C(I,J)+A(I,1)B(1,J)

K=2

calcola in parallelo su J C(I,J) = C(I,J)+A(I,2)B(2,J)

Sincronizzazione tra 2 successivi valori di K
tra tutti i task paralleli J

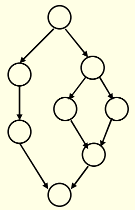
Qual'e' la migliore versione?

Quella che minimizza il numero di sincronizzazioni

Versione migliore
=
Versione (I,J,K) !!

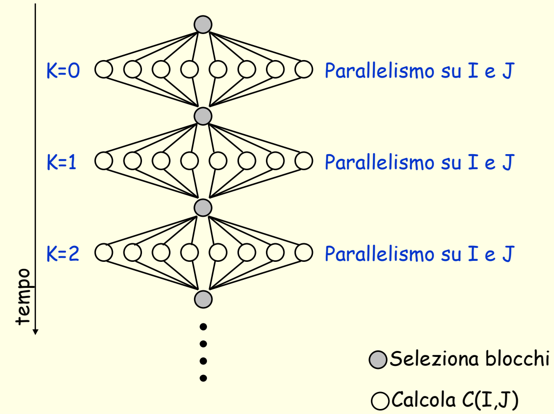
Analisi delle tre versioni

Una analisi precisa dei precedenti algoritmi
puo' essere effettuata mediante i
Grafi Aciclici Diretti
dove i nodi sono i task
e gli archi rappresentano le dipendenze

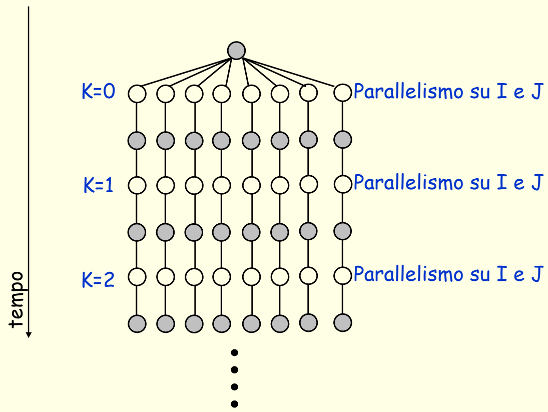


Grafo = insieme di nodi e archi
Aciclico = assenza di cicli nel grafo
Diretto = gli archi hanno un solo verso

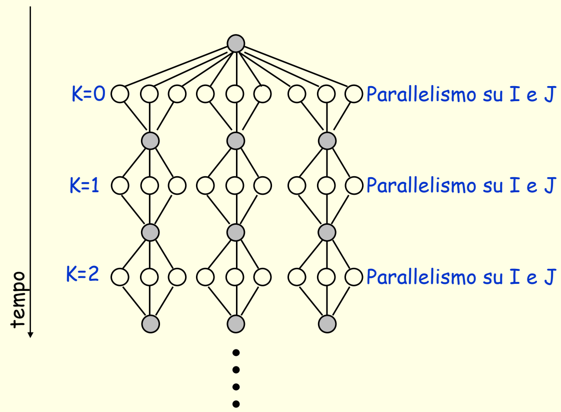
versione (K,I,J)



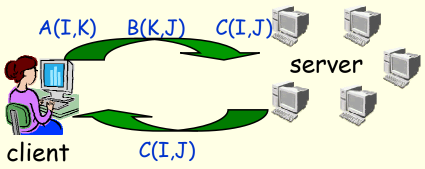
versione (I,J,K)



versione (I,K,J)



Tempo di ogni task



Sia t_{ijk}

Il tempo per

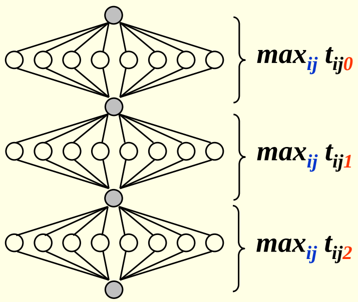
- inviare A(I,K) B(K,J) C(I,J)
- calcolare C(I,J)=C(I,J)+A(I,K)B(K,J)
- ricevere C(I,J)

In un ambiente di calcolo distribuito

- risorse non omogene
- risorse non dedicate

t_{ijk} diverso per ogni valore degli indici I, J e K (anche se i blocchi sono tutti uguali)

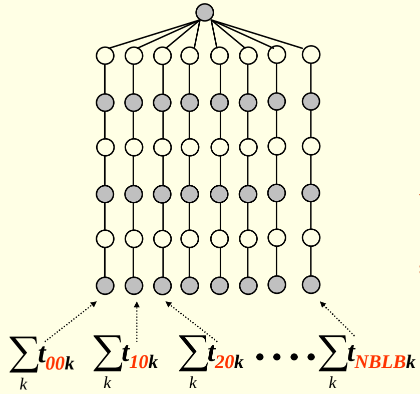
Qual'e' il tempo di esecuzione delle 3 versioni?



Versione (K,I,J)

$$\text{tempo totale} = T^{(K,I,J)} = \sum_k \max_{i,j} t_{ijk}$$

Qual'e' il tempo di esecuzione delle 3 versioni?



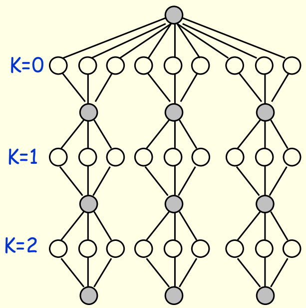
Versione (I,J,K)

$$\text{tempo totale} = T^{(I,J,K)} = \max_{i,j} \sum_k t_{ijk}$$

Qual'e' il tempo di esecuzione delle 3 versioni?

Versione (I,K,J)

$$\text{tempo totale} = T^{(I,K,J)} = \max_i \sum_k \max_j t_{ijk}$$



$$\sum_k \max_j t_{0jk} \;\cdots\; \sum_k \max_j t_{NBjk}$$

Riassumendo …

$$T^{(I,J,K)} = \max_{i,j} \sum_k t_{ijk}$$

$$T^{(I,K,J)} = \max_i \sum_k \max_j t_{ijk}$$

$$T^{(K,I,J)} = \sum_k \max_{i,j} t_{ijk}$$

Qual'e' il valore minimo?

In generale:

Siano $a_{pq} > 0$ gli elementi di un insieme indicizzati da p e q

Si dimostra che:

$$\max_p \sum_q a_{pq} \leq \max_p \sum_q \max_r a_{rq} = \sum_q \max_r a_{rq} = \sum_q \max_p a_{pq}$$

(il massimo della somma e' minore della somma dei massimi)

Sfruttando tale proprieta' si ha:

$$T^{(I,J,K)} = \max_{i,j} \sum_k t_{ijk} = \max_i \max_j \sum_k t_{ijk} \leq$$

$$\max_i \sum_k \max_j t_{ijk} = T^{(I,K,J)} \leq$$

$$\sum_k \max_i \max_j t_{ijk} = \sum_k \max_{i,j} t_{ijk} = T^{(K,I,J)}$$

La versione piu' adatta ad un ambiente per il calcolo distribuito e' la versione (I,J,K)