1

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

2

Parallelo vs distribuito

Calcolatore parallelo

- reti veloci
- risorse limitate
- risorse dedicate e omogenee
- applicazione gestisce le risorse
- costo hardware notevole
- overhead sw sistema < 5%
- presenza di vincoli temporali
- es. Roadrunner
  - 16000 processori
  - 1 Pflops

Ambiente per il C.D.

- reti lente
- risorse potenzialmente illimitate
- risorse condivise e disomogenee
- ambiente sw.gestisce le risorse
- costo hardware trascurabile
- overhead sw sistema > 20%
- assenza di vincoli temporali
- es. SETI@home
  - 5 milioni processori
  - 100 Tflops

Molte differenze ! ! !

3

Domanda

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

Quelle che riducono efficientemente il tempo di esecuzione !!

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

4

Impatto sull'efficienza

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

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

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

Efficienza vs OC

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

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: $E_p=0.8 \rightarrow OC_p < 0.25$

Poiche' in un ambiente distribuito

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

è maggiore di 8000

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

In generale, se si vuole $E_p=0.8$

IBM BG

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

beowulf

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

Ambiente C.D.

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

Quindi

IBM BG	cluster	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 tra i nodi
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

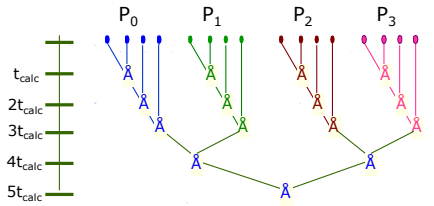
Per un Calcolatore Parallelo abbiamo assunto che:

- 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

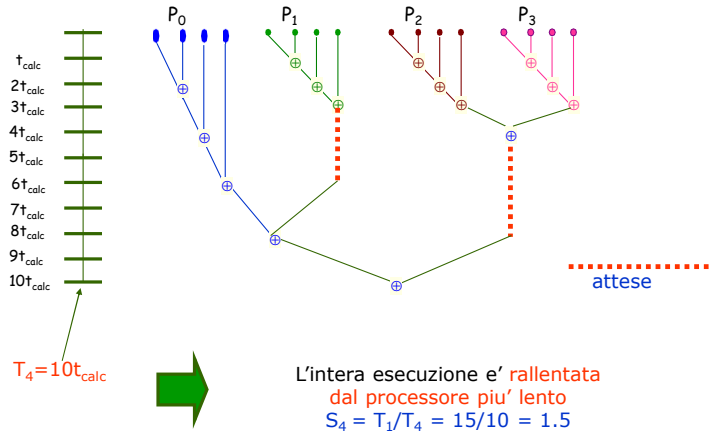
t_{calc} non e' lo stesso in tutti i processori

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.



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

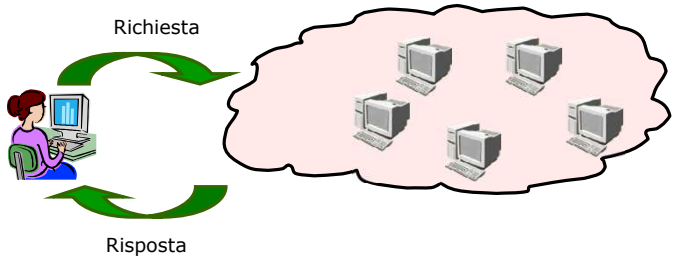
Domanda:

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.



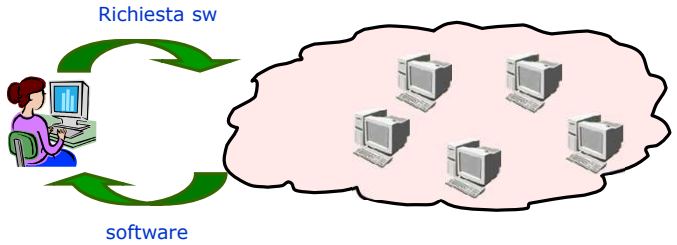
In un ambiente per il calcolo distribuito l'utente dialoga con un ambiente software richiedendogli un servizio (modello client-server)

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

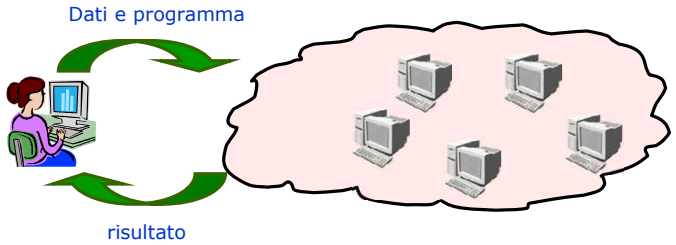
Esempio 1: server software



- 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)

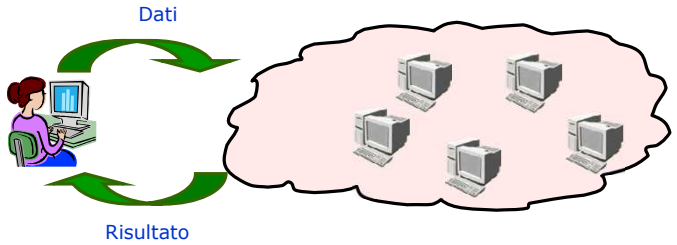
Esempio 2: server hardware



- 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



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

(remote computing)

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)

MA 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

Ma, le ipotesi di

- Omogeneita'
- Dedicazione al calcolo

non possono essere fatte in un ambiente di calcolo distribuito

Prodotto a blocchi di due matrici

A_{00}	A_{01}	A_{02}	A_{03}
A_{10}	A_{11}	A_{12}	A_{13}
A_{20}	A_{21}	A_{22}	A_{23}
A_{30}	A_{31}	A_{32}	A_{33}

B_{00}	B_{01}
B_{10}	B_{11}
B_{20}	B_{21}
B_{30}	B_{31}

C_{00}	C_{01}
C_{10}	C_{11}
C_{20}	C_{21}
C_{30}	C_{31}

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

Come eseguirlo in un ambiente di C.D. ?

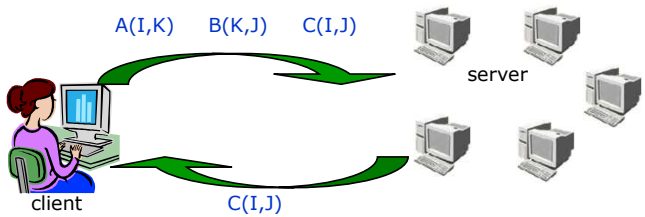
osservazione

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

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

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

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

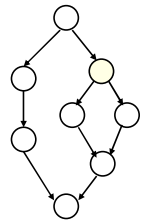
Qual e' la migliore versione?

Quella che minimizza il numero di sincronizzazioni

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

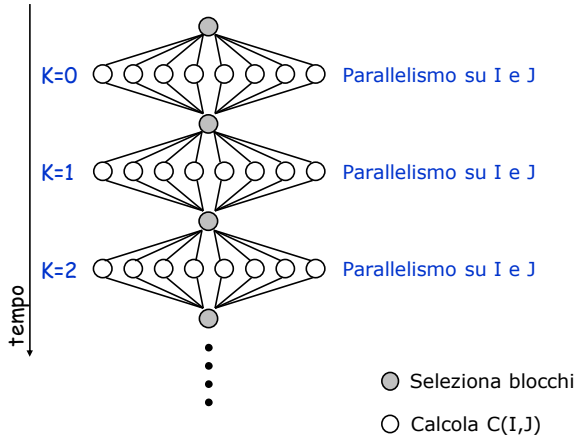
Analisi delle tre versioni

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

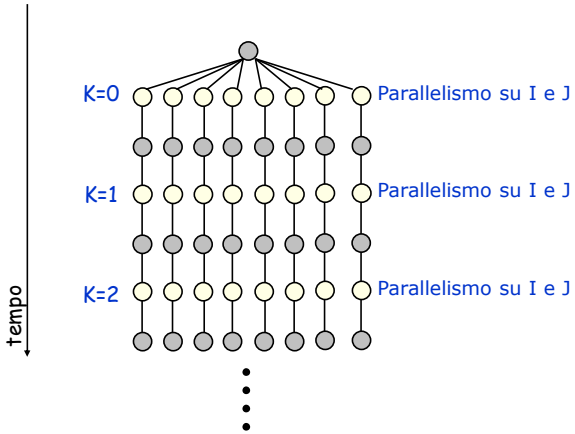


Grafo = insieme di nodi e archi
Aciclico = assenza di cicli nel grafo
Diretto = gli archi hanno un solo verso

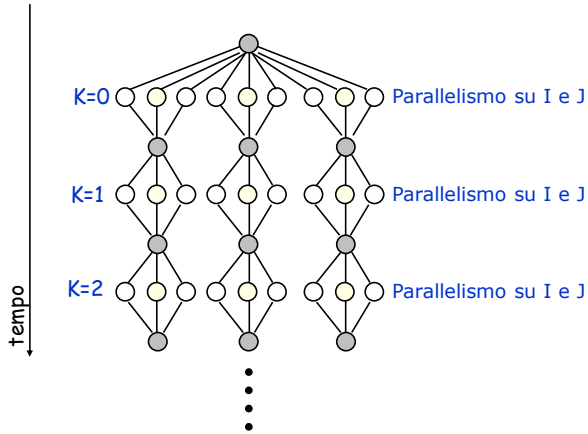
versione (K,I,J)



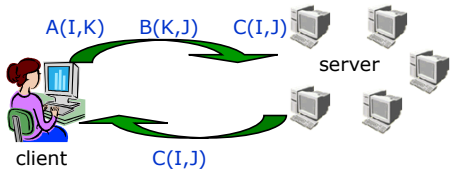
versione (I,J,K)



versione (I,K,J)



Tempo di ogni task



Sia t_{ijk}
Il tempo per

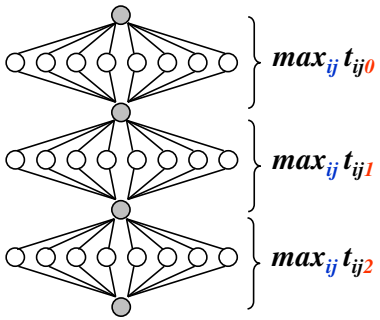
- inviare A(I,K) B(K,J) C(I,J)
- calcolare C(I,J)=C(I,J)+A(I,K)B(K,J)
- ricevere C(I,J)

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?

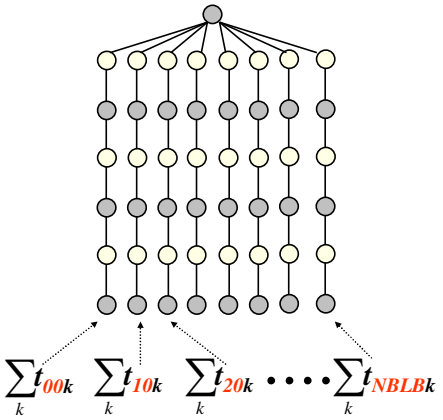


Versione
(K,I,J)

$$\text{tempo totale} = T^{(K,I,J)}$$

$$= \sum_k \max_{i,j} t_{ijk}$$

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



Versione
(I,J,K)

$$\text{tempo totale} = T^{(I,J,K)}$$

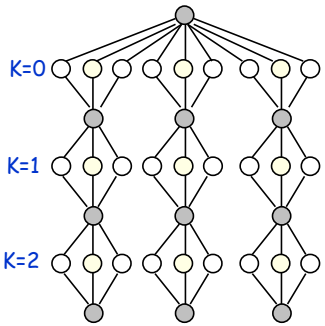
$$= \max_{i,j} \sum_k t_{ijk}$$

Qual e' il tempo di esecuzione delle 3 versioni?

Versione (I,K,J)

$$\text{tempo totale} = T^{(I,K,J)}$$

$$= \max_i \sum_k \max_j t_{ijk}$$



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

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:

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

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

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

Sfruttando tale proprieta' si ha:

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

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

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

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