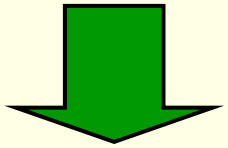
PROBLEMA:
prodotto di matrici

$$C = A \cdot B$$

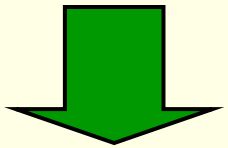
in un ambiente di calcolo distribuito

Possibile algoritmo parallelo

suddivisione del problema



Suddivisione delle matrici a blocchi



Algoritmi a blocchi

Possibile algoritmo parallelo

Decomposizione delle matrici
A, B e C a blocchi di righe e di colonne
(decomposizione ciclica a blocchi)

esempio

$MB=4$

$NB=4$

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}

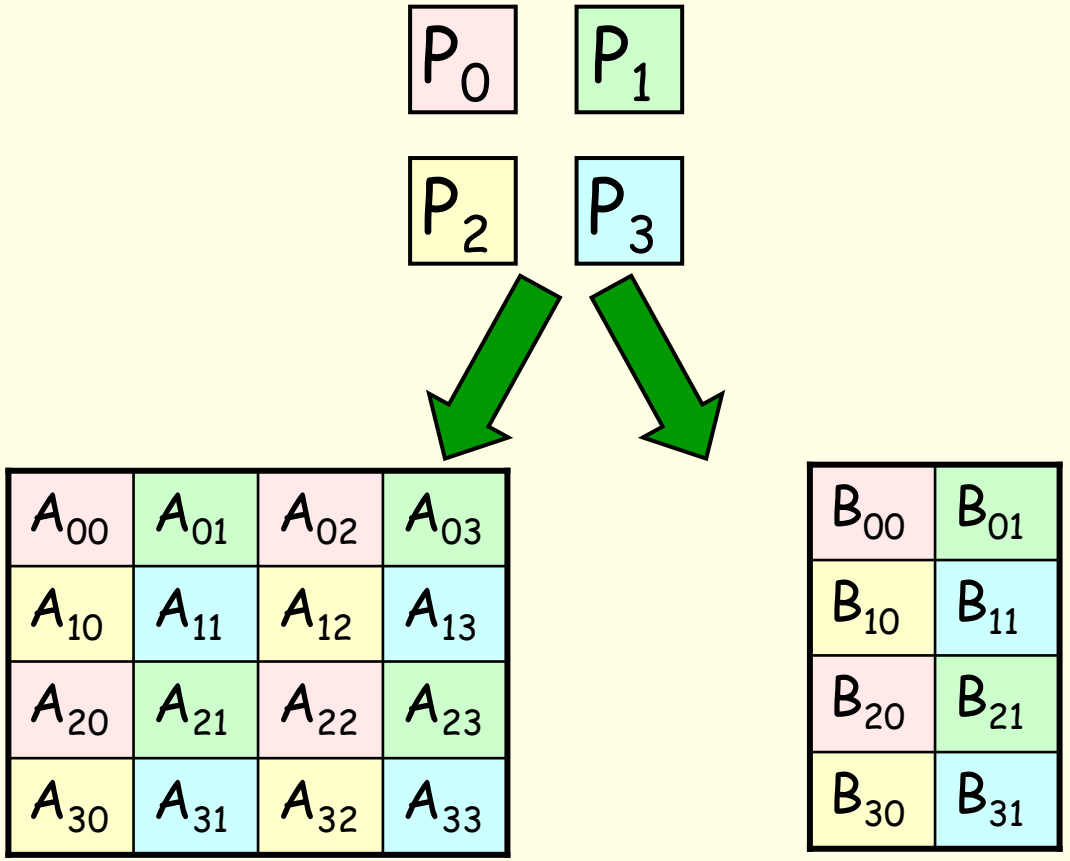
$LB=2$

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

NB, MB e LB = numero di blocchi

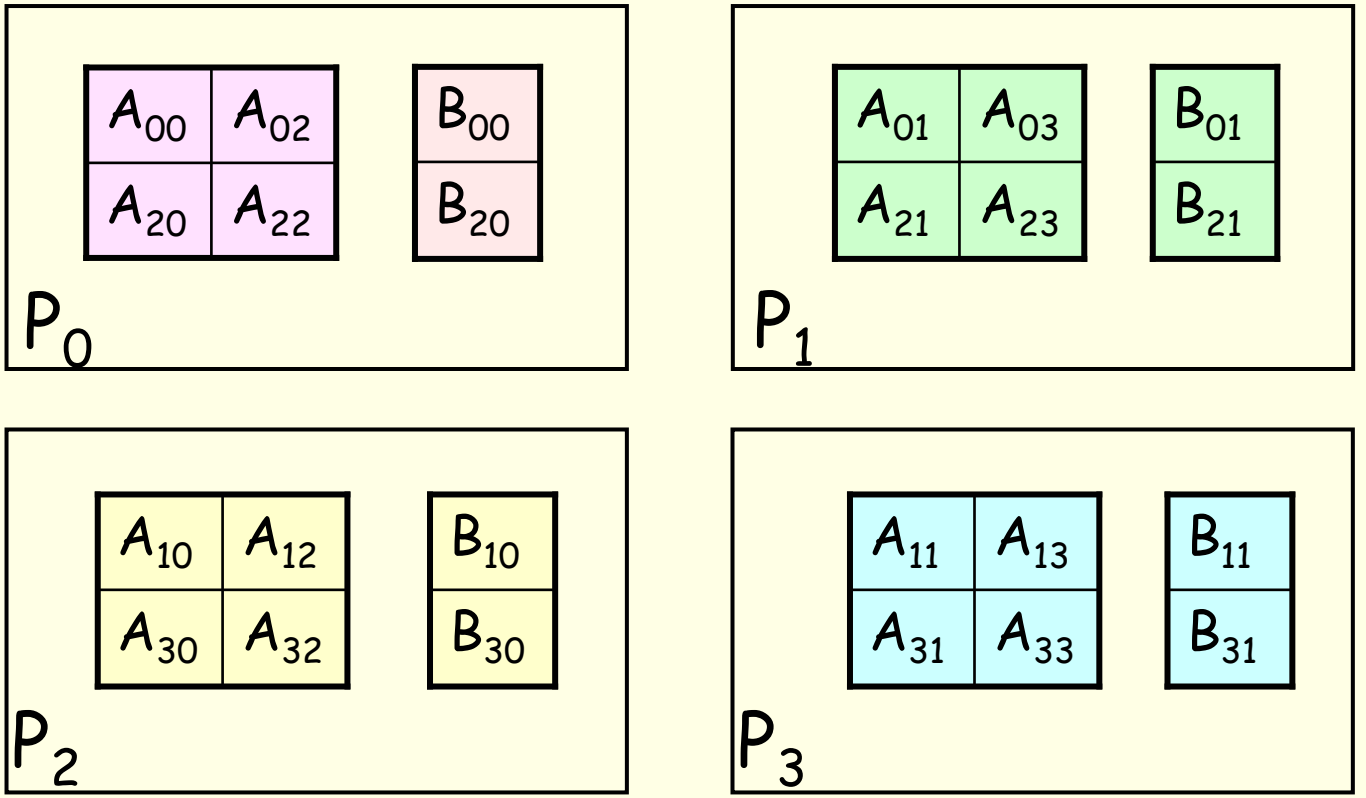
Distribuzione delle matrici

Distribuiamo le matrici su una griglia di 2x2 processori

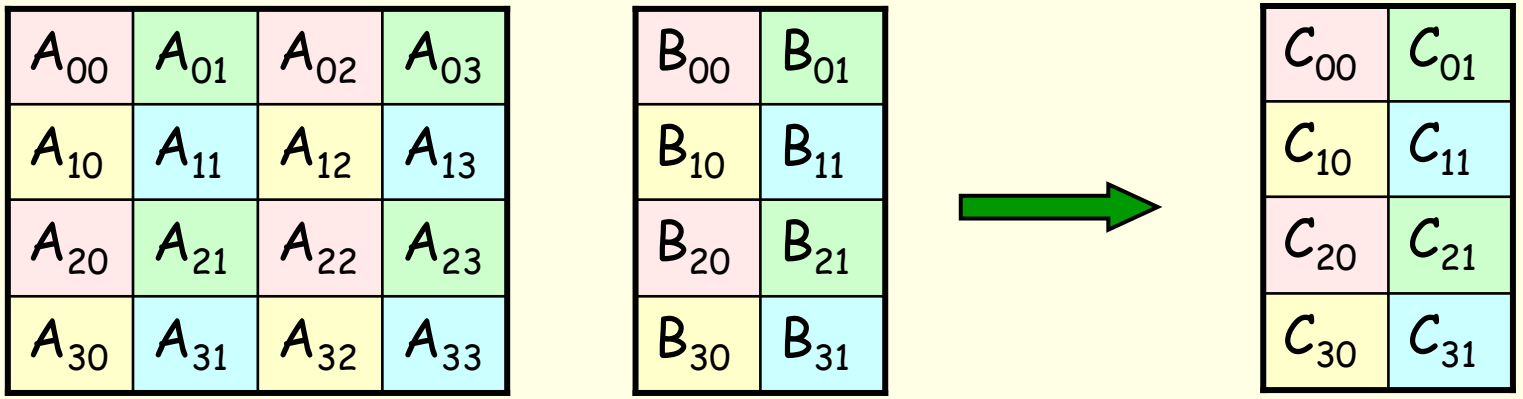


Distribuzione ciclica a blocchi

Quali blocchi hanno i processori?



Prodotto a blocchi



$P_0 \rightarrow C_{00} = A_{00}B_{00} + A_{01}B_{10} + A_{02}B_{20} + A_{03}B_{30}$

$P_1 \rightarrow C_{01} = A_{00}B_{01} + A_{01}B_{11} + A_{02}B_{21} + A_{03}B_{31}$

$P_2 \rightarrow C_{10} = A_{10}B_{00} + A_{11}B_{10} + A_{12}B_{20} + A_{13}B_{30}$

$P_3 \rightarrow C_{11} = A_{10}B_{01} + A_{11}B_{11} + A_{12}B_{21} + A_{13}B_{31}$

$P_0 \rightarrow C_{20} = A_{20}B_{00} + A_{21}B_{10} + A_{22}B_{20} + A_{23}B_{30}$

$P_1 \rightarrow C_{21} = A_{20}B_{01} + A_{21}B_{11} + A_{22}B_{21} + A_{23}B_{31}$

$P_2 \rightarrow C_{30} = A_{30}B_{00} + A_{31}B_{10} + A_{32}B_{20} + A_{33}B_{30}$

$P_3 \rightarrow C_{31} = A_{30}B_{01} + A_{31}B_{11} + A_{32}B_{21} + A_{33}B_{31}$

Osservazione

$P_0 \rightarrow C_{00} = A_{00}B_{00} + A_{01}B_{10} + A_{02}B_{20} + A_{03}B_{30}$

$P_1 \rightarrow C_{01} = A_{00}B_{01} + A_{01}B_{11} + A_{02}B_{21} + A_{03}B_{31}$

$P_2 \rightarrow C_{10} = A_{10}B_{00} + A_{11}B_{10} + A_{12}B_{20} + A_{13}B_{30}$

$P_3 \rightarrow C_{11} = A_{10}B_{01} + A_{11}B_{11} + A_{12}B_{21} + A_{13}B_{31}$

$P_0 \rightarrow C_{20} = A_{20}B_{00} + A_{21}B_{10} + A_{22}B_{20} + A_{23}B_{30}$

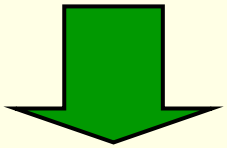
$P_1 \rightarrow C_{21} = A_{20}B_{01} + A_{21}B_{11} + A_{22}B_{21} + A_{23}B_{31}$

$P_2 \rightarrow C_{30} = A_{30}B_{00} + A_{31}B_{10} + A_{32}B_{20} + A_{33}B_{30}$

$P_3 \rightarrow C_{31} = A_{30}B_{01} + A_{31}B_{11} + A_{32}B_{21} + A_{33}B_{31}$

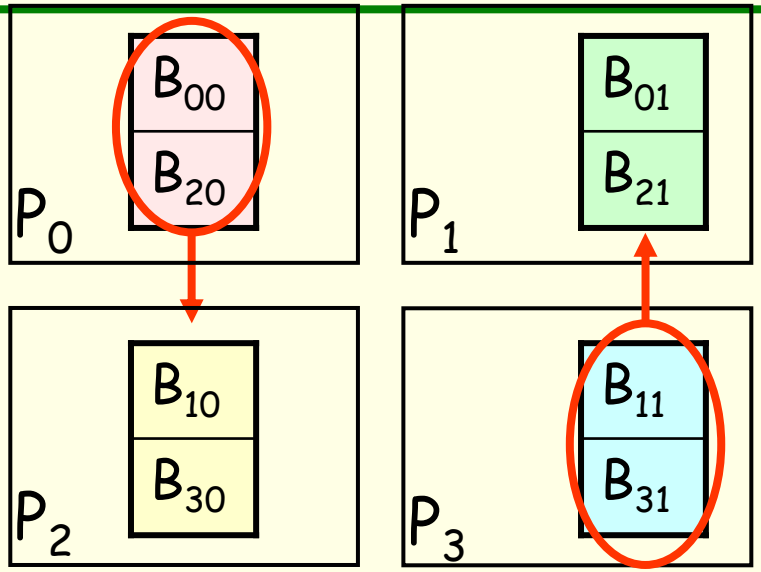
Rossi = Dati disponibili

Blu = Dati non disponibili



P_1 e P_2 non possono iniziare il calcolo

I passo: comunicazione dei blocchi di B



Broadcast di B_{00}, B_{20} B_{11}, B_{31} sulla colonna di processori

Dati acquisiti

$P_0 \rightarrow C_{00} = A_{00}B_{00} + A_{01}B_{10} + A_{02}B_{20} + A_{03}B_{30}$

$P_1 \rightarrow C_{01} = A_{00}B_{01} + A_{01}B_{11} + A_{02}B_{21} + A_{03}B_{31}$

$P_2 \rightarrow C_{10} = A_{10}B_{00} + A_{11}B_{10} + A_{12}B_{20} + A_{13}B_{30}$

$P_3 \rightarrow C_{11} = A_{10}B_{01} + A_{11}B_{11} + A_{12}B_{21} + A_{13}B_{31}$

$P_0 \rightarrow C_{20} = A_{20}B_{00} + A_{21}B_{10} + A_{22}B_{20} + A_{23}B_{30}$

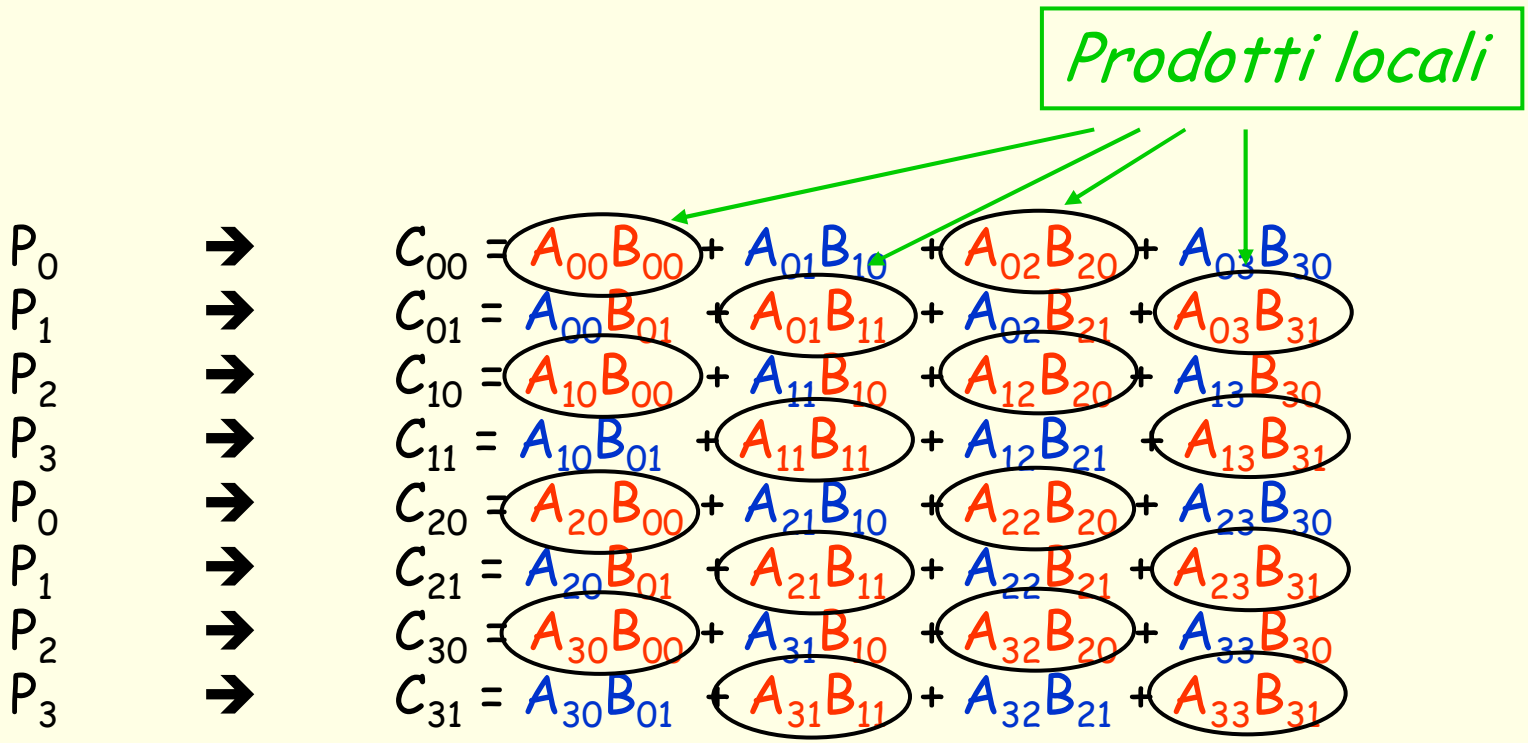
$P_1 \rightarrow C_{21} = A_{20}B_{01} + A_{21}B_{11} + A_{22}B_{21} + A_{23}B_{31}$

$P_2 \rightarrow C_{30} = A_{30}B_{00} + A_{31}B_{10} + A_{32}B_{20} + A_{33}B_{30}$

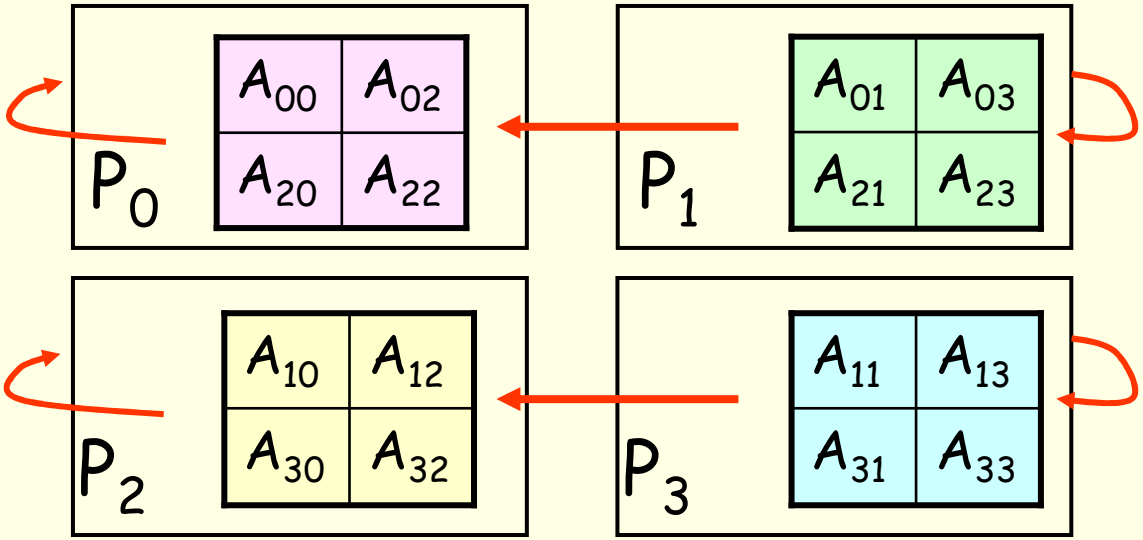
$P_3 \rightarrow C_{31} = A_{30}B_{01} + A_{31}B_{11} + A_{32}B_{21} + A_{33}B_{31}$

I passo: calcolo

Prodotto a blocchi localmente in ogni processore



I passo: comunicazione dei blocchi di A



Comunicazione a sinistra dei blocchi di A

Dati acquisiti

$P_0 \rightarrow C_{00} = A_{00}B_{00} + A_{01}B_{10} + A_{02}B_{20} + A_{03}B_{30}$

$P_1 \rightarrow C_{01} = A_{00}B_{01} + A_{01}B_{11} + A_{02}B_{21} + A_{03}B_{31}$

$P_2 \rightarrow C_{10} = A_{10}B_{00} + A_{11}B_{10} + A_{12}B_{20} + A_{13}B_{30}$

$P_3 \rightarrow C_{11} = A_{10}B_{01} + A_{11}B_{11} + A_{12}B_{21} + A_{13}B_{31}$

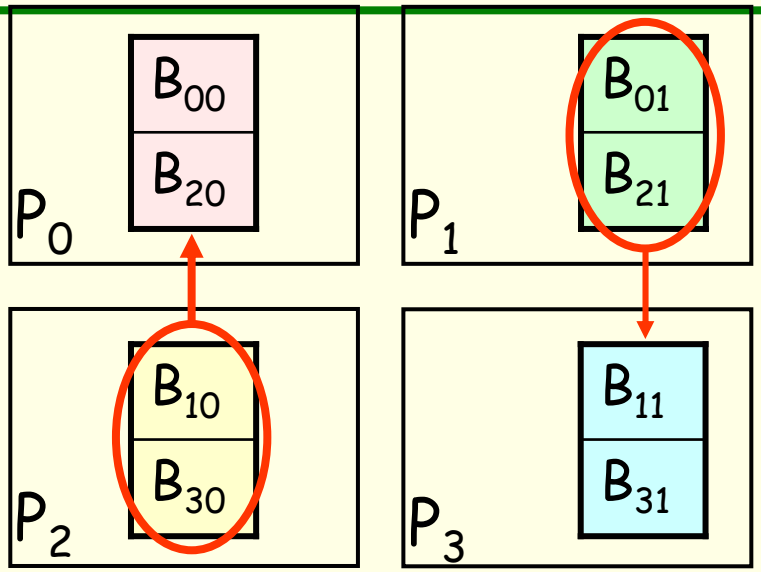
$P_0 \rightarrow C_{20} = A_{20}B_{00} + A_{21}B_{10} + A_{22}B_{20} + A_{23}B_{30}$

$P_1 \rightarrow C_{21} = A_{20}B_{01} + A_{21}B_{11} + A_{22}B_{21} + A_{23}B_{31}$

$P_2 \rightarrow C_{30} = A_{30}B_{00} + A_{31}B_{10} + A_{32}B_{20} + A_{33}B_{30}$

$P_3 \rightarrow C_{31} = A_{30}B_{01} + A_{31}B_{11} + A_{32}B_{21} + A_{33}B_{31}$

II passo: comunicazione dei blocchi di B



Broadcast di B_{10}, B_{30} B_{01}, B_{21} sulla colonna di processori

Dati acquisiti

P_0 → $C_{00} = A_{00}B_{00} + A_{01}B_{10} + A_{02}B_{20} + A_{03}B_{30}$

P_1 → $C_{01} = A_{00}B_{01} + A_{01}B_{11} + A_{02}B_{21} + A_{03}B_{31}$

P_2 → $C_{10} = A_{10}B_{00} + A_{11}B_{10} + A_{12}B_{20} + A_{13}B_{30}$

P_3 → $C_{11} = A_{10}B_{01} + A_{11}B_{11} + A_{12}B_{21} + A_{13}B_{31}$

P_0 → $C_{20} = A_{20}B_{00} + A_{21}B_{10} + A_{22}B_{20} + A_{23}B_{30}$

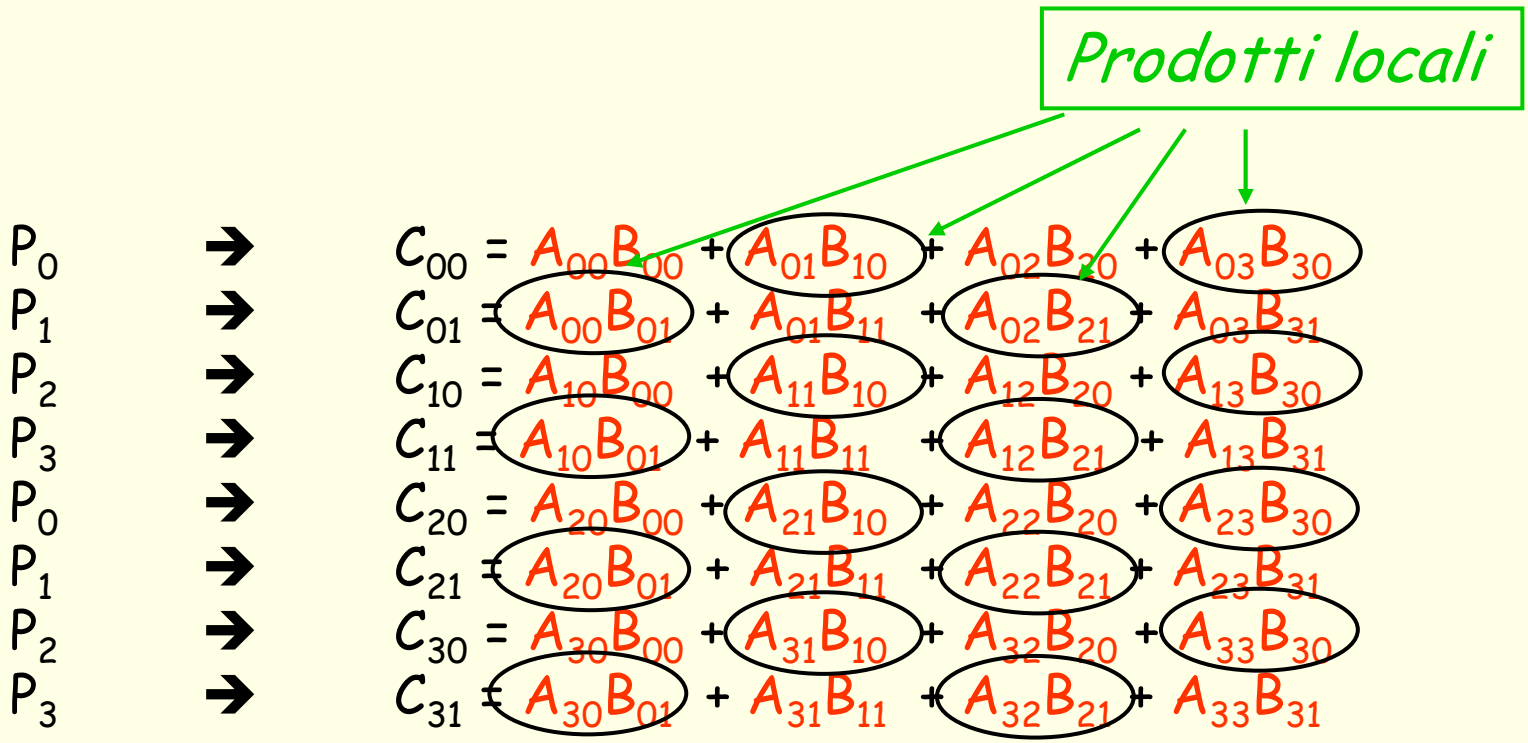
P_1 → $C_{21} = A_{20}B_{01} + A_{21}B_{11} + A_{22}B_{21} + A_{23}B_{31}$

P_2 → $C_{30} = A_{30}B_{00} + A_{31}B_{10} + A_{32}B_{20} + A_{33}B_{30}$

P_3 → $C_{31} = A_{30}B_{01} + A_{31}B_{11} + A_{32}B_{21} + A_{33}B_{31}$

II passo: calcolo

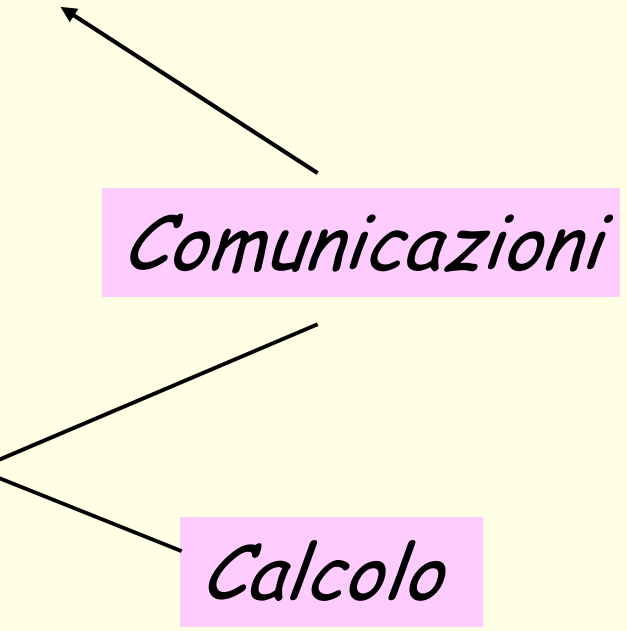
Prodotto a blocchi localmente in ogni processore



In generale:

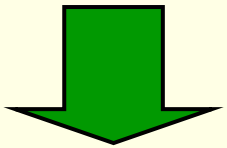
- MB, NB e LB numero di blocchi delle matrici
- griglia di P x Q processori

```
do K1= 0, Q-1
   broadcast MB/Q blocchi of B su una colonna di processori
   do K2= 0, MB/Q-1
      K= k1+ K2*Q
      do I=0, NB-1 in parallel
         KP= MOD(K+ MOD(I,Q),MB)
         do J=0, LB-1 in parallel
            C(I,J) = C(I,J) + A(I,KP)B(KP,J)
         enddo
      enddo
   enddo
   comunica blocchi di  A a sinistra
enddo
```



Il precedente algoritmo

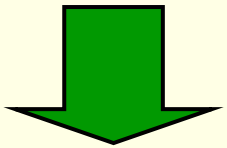
- massimizza la granularita' del calcolo
- bilancia il carico tra i processori
- ha una efficiente strategia di comunicazione



E' alla base della routine
PDGEMM di ScaLAPACK

Osservazione

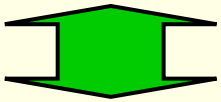
Il precedente algoritmo richiede una stretta sincronizzazione dei processori



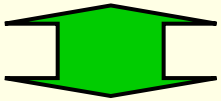
Algoritmo sistolico

In definitiva

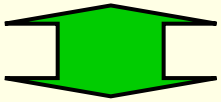
Buona efficienza parallela



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



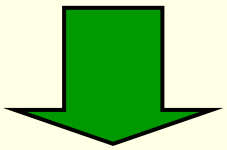
Ambiente omogeneo e dedicato

Osservazione

Le ipotesi di

- Omogeneita'
- Dedicazione al calcolo

non possono essere fatte in un ambiente di calcolo distribuito

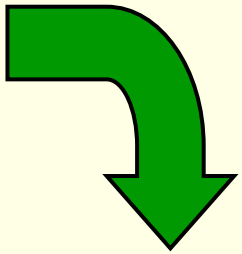


Rivisitazione dell'algoritmo

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

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}

$$C(I,J) = \sum_{K=0}^{MB-1} A(I,K)B(K,J)$$

$$I = 0,..,NB-1$$

$$J = 0,..,LB-1$$

Come eseguirlo in un ambiente di C.D. ?

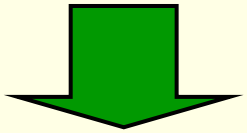
osservazione

$$C(I,J) = \sum_{K=0}^{MB-1} A(I,K)B(K,J)$$

$$I = 0,..,NB-1$$

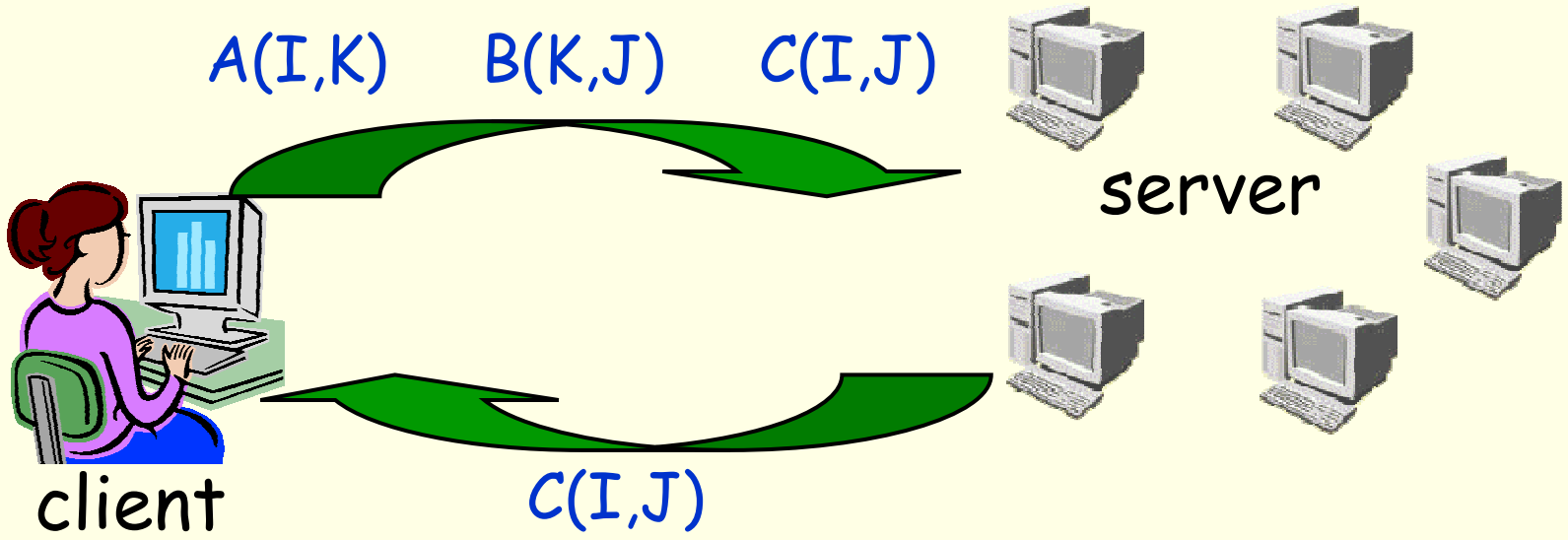
$$J = 0,..,LB-1$$

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)

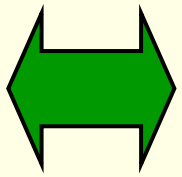
```
for K = 0,MB-1
        for I = 0,NB-1 (in parallelo)
                for J = 0,LB-1 (in parallelo)
                        C(I,J)=C(I,J)+A(I,K)B(K,J)
                endfor
        endfor
endfor
```

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
```

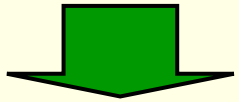


```
for J = 0,LB-1  (in parallelo)
  for I = 0,NB-1  (in parallelo)
    for K = 0,MB-1
      C(I,J)=C(I,J)+A(I,K)B(K,J)
    endfor
  endfor
endfor
```

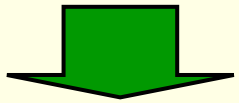
Osservazione 2

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

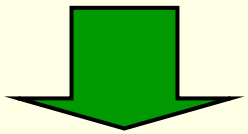
calcola in parallelo su I e J $C(I,J) = C(I,J)+A(I,0)B(0,J)$

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

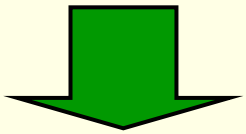
calcola $C(I,J) = C(I,J)+A(I,0)B(0,J)$

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)

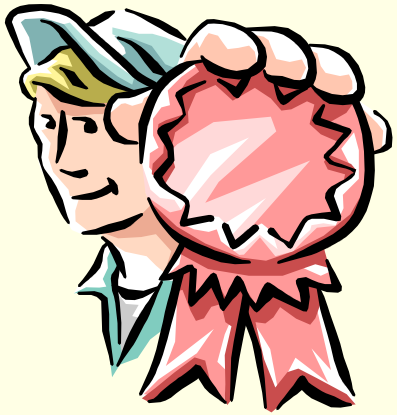
K=2

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



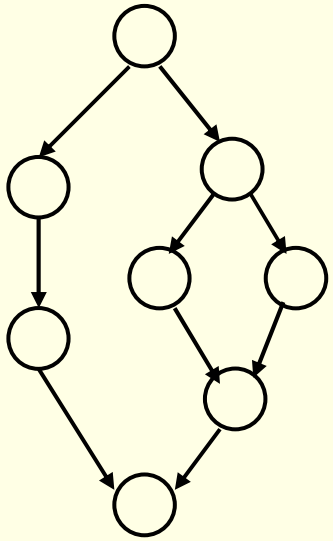
Versione migliore

=

Versione (I,J,K) !!

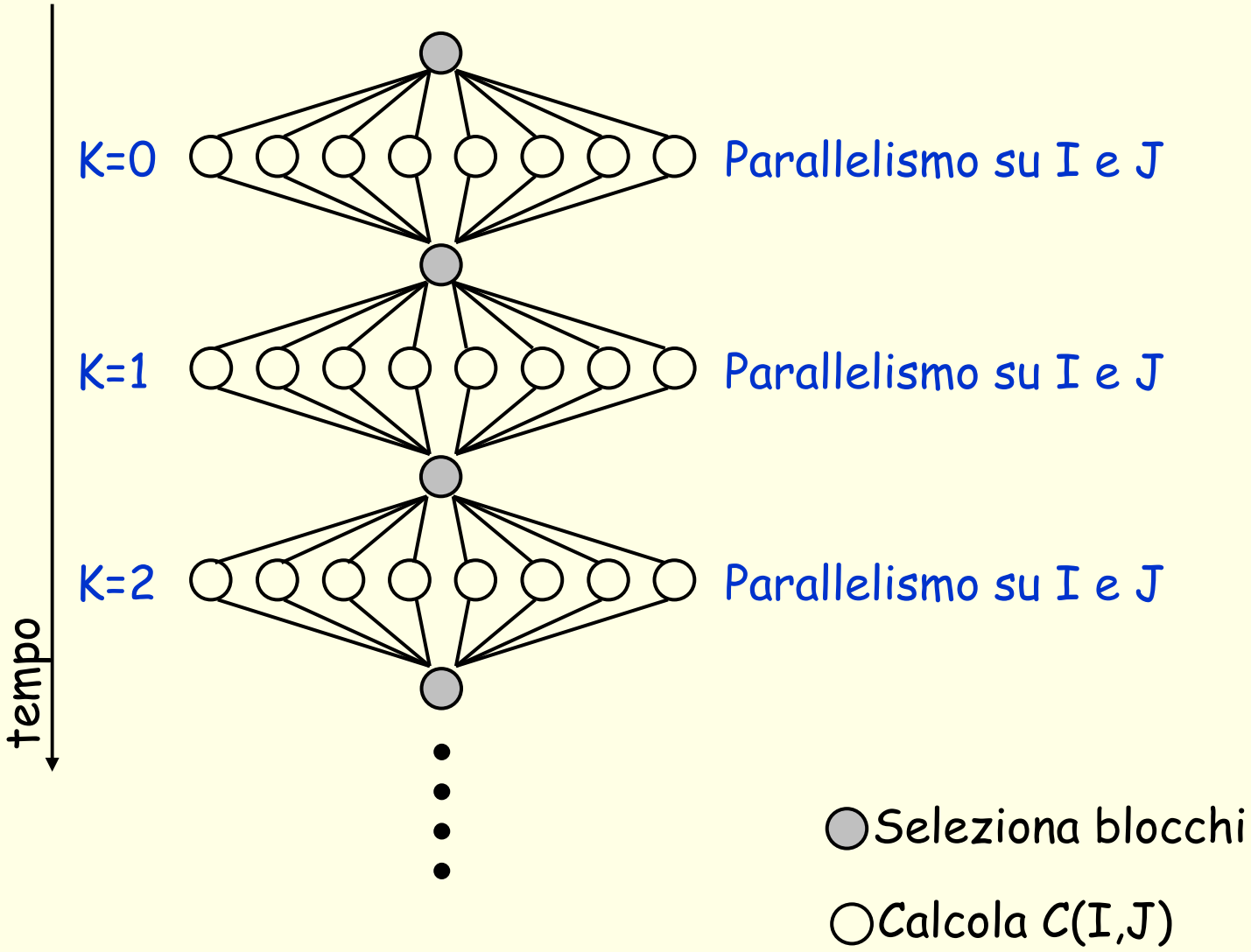
Analisi delle tre versioni

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

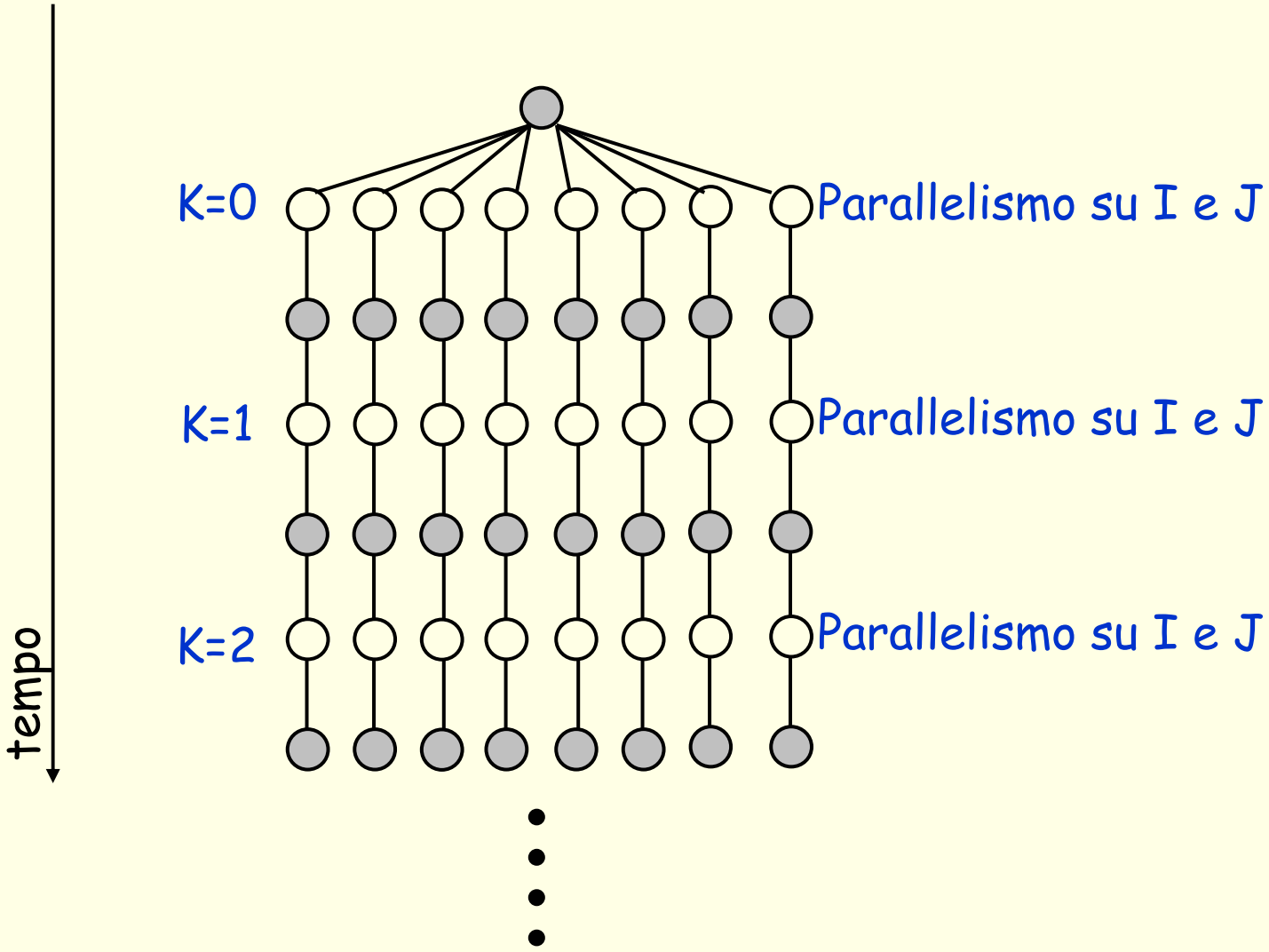


Grafo = insieme di nodi e archi
Aciclico = assenza di cicli nel grafo
Diretto = gli archi hanno un solo verso

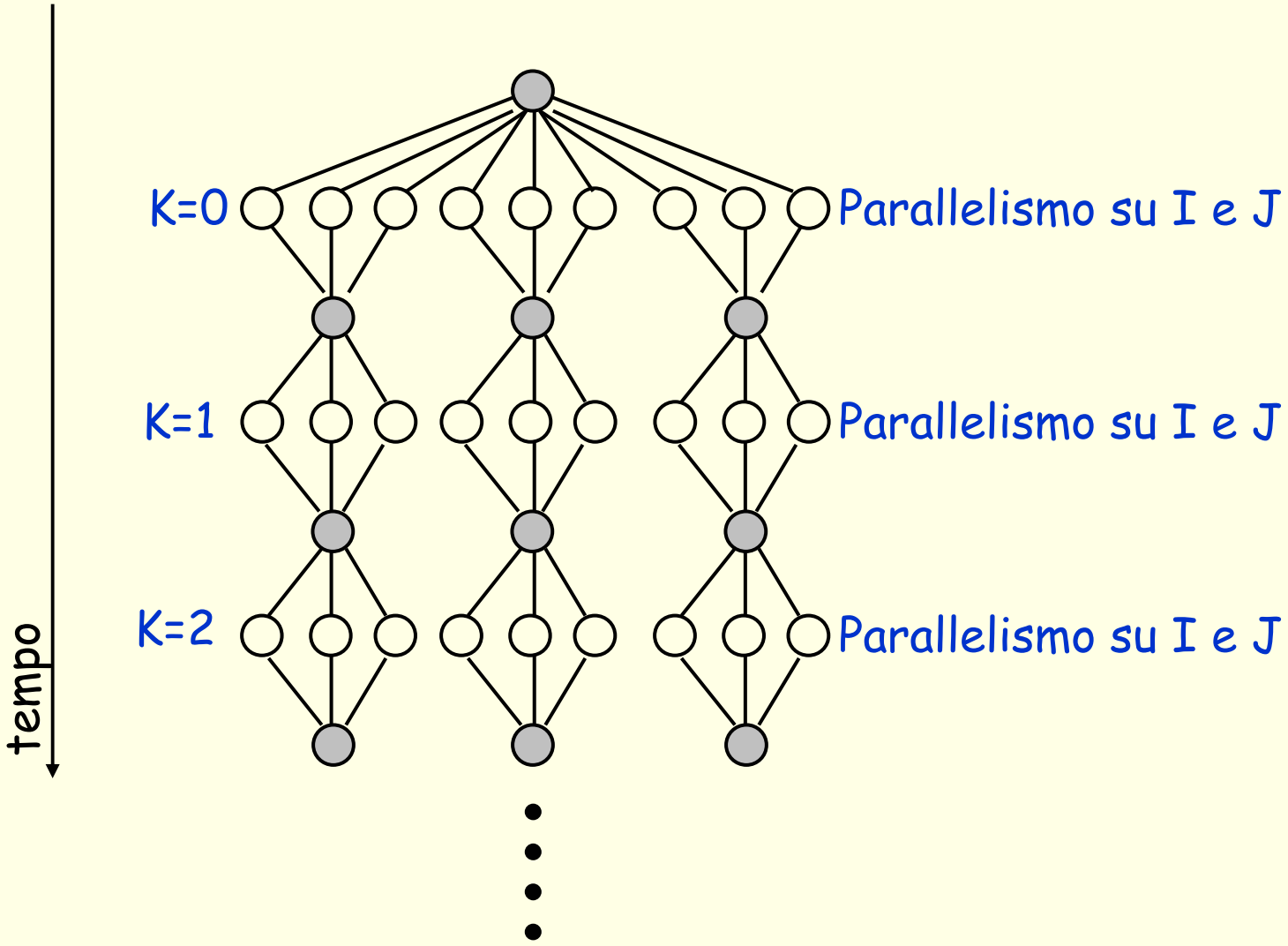
versione (K,I,J)



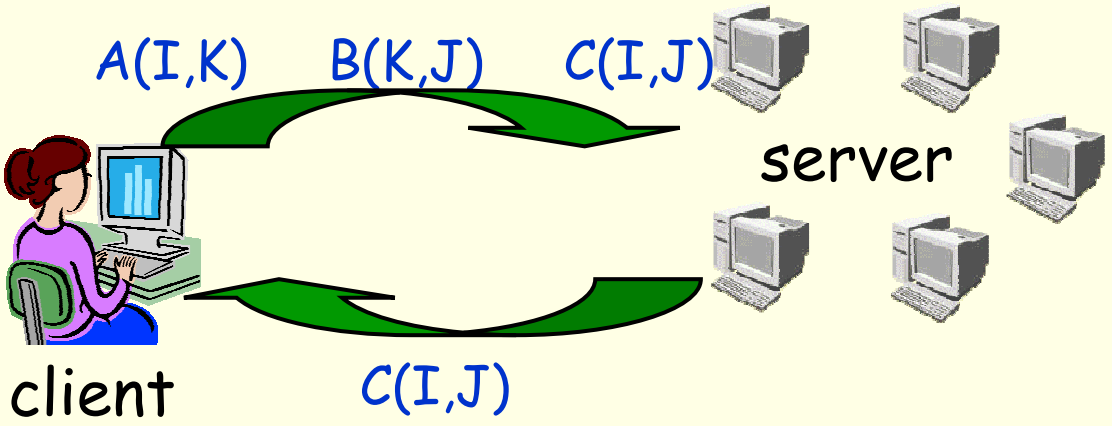
versione (I,J,K)



versione (I,K,J)



Tempo di ogni task



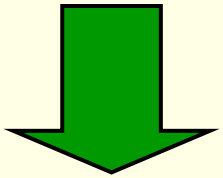
Sia t_{ijk}

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)

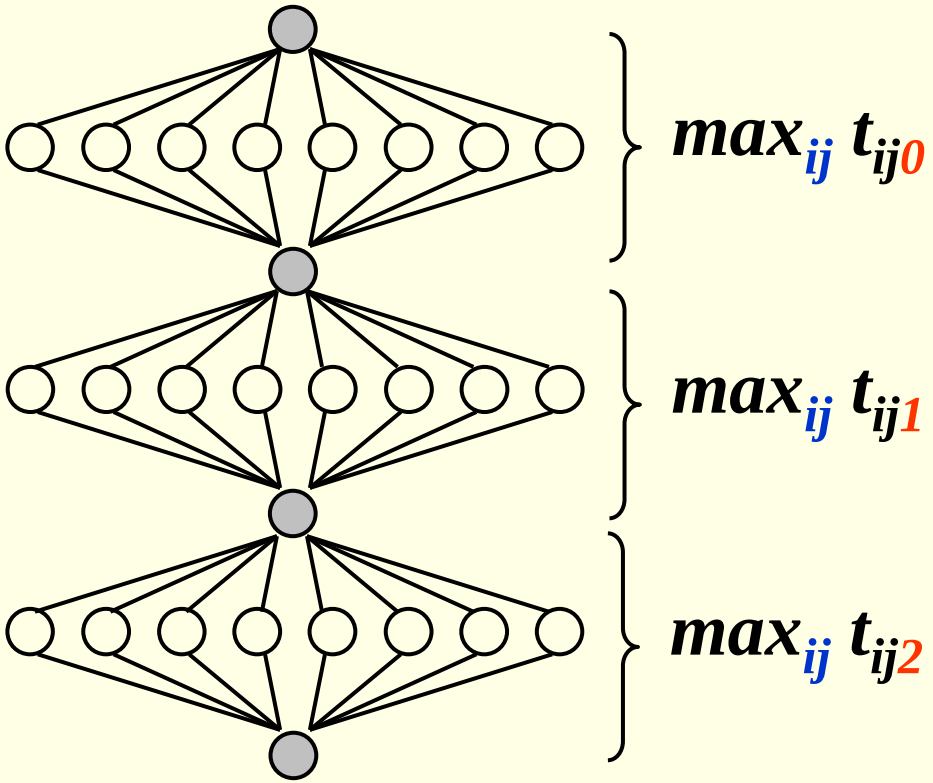
In un ambiente di calcolo distribuito

- 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?

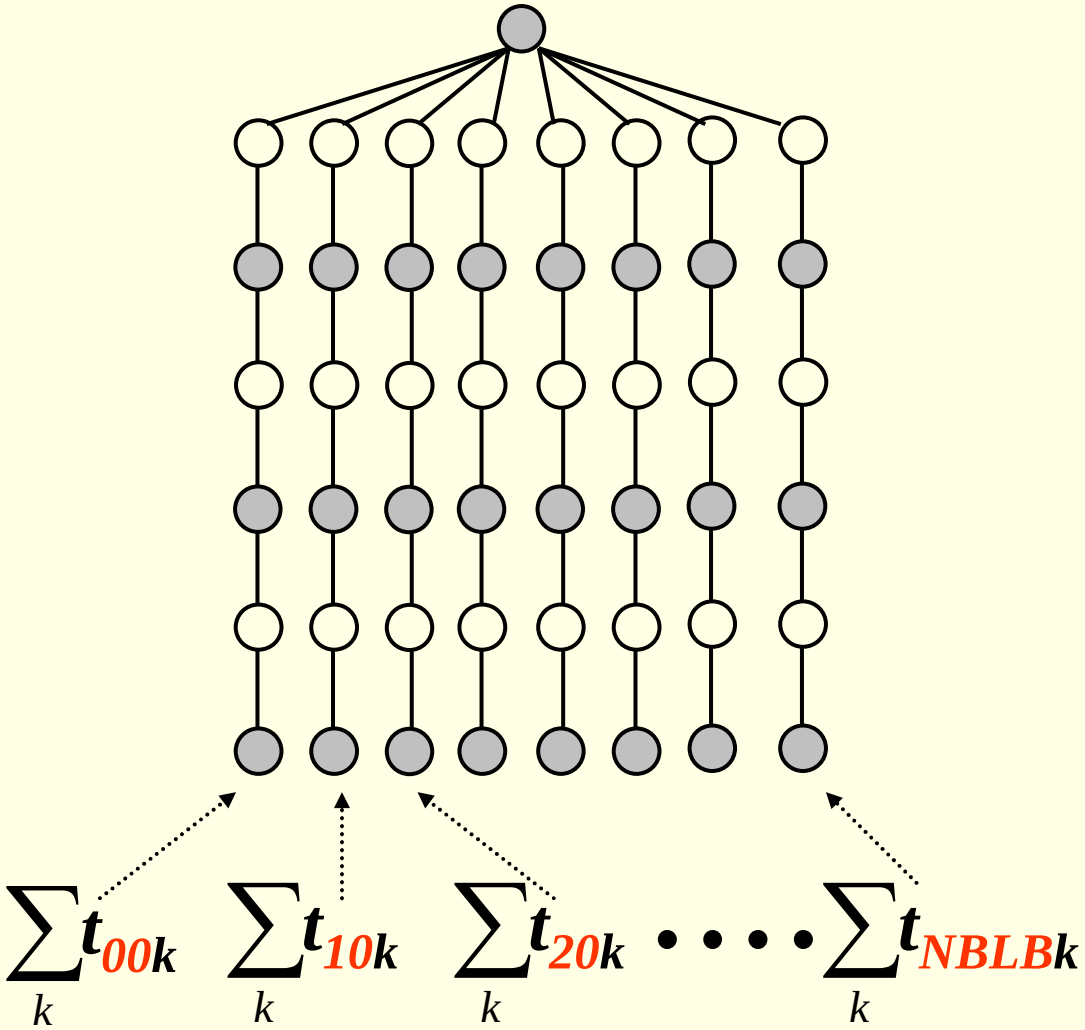


Versione (K,I,J)

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

$$= \sum_{k} \max_{i,j}\ t_{ijk}$$

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



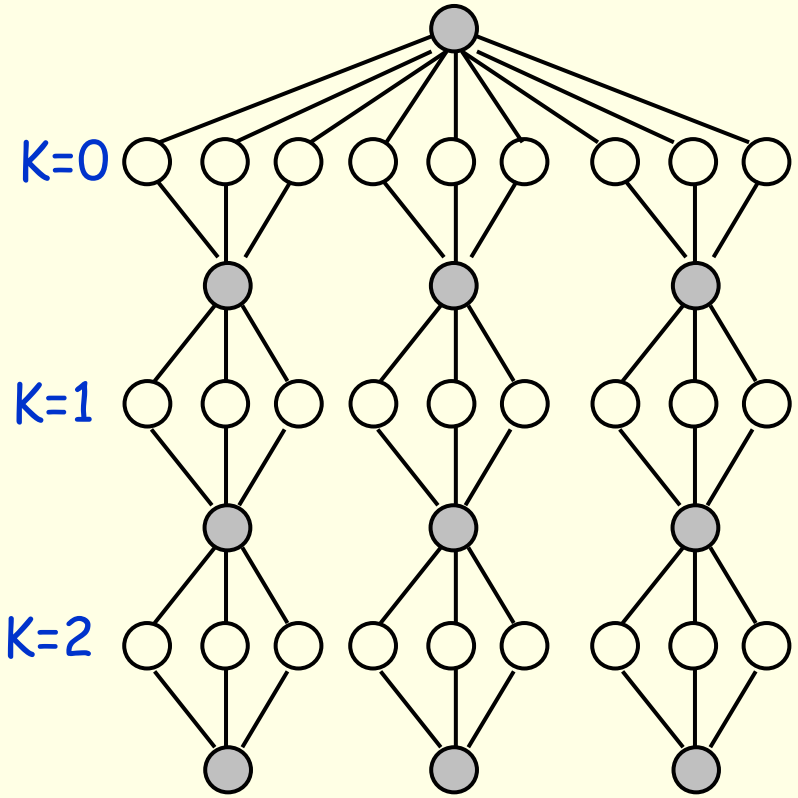
Versione
(I,J,K)

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

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

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

Versione
(I,K,J)



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

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

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

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:

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

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

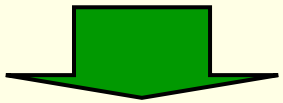
(il massimo della somma e' minore della somma dei massimi)

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)