Calcolo Parallelo e Distribuito

Il prodotto scalare in ambiente multicore
strategie di parallelizzazione

Docente: Prof. L. Marcellino
Tutor: Prof. P. De Luca

Previously on… CPD

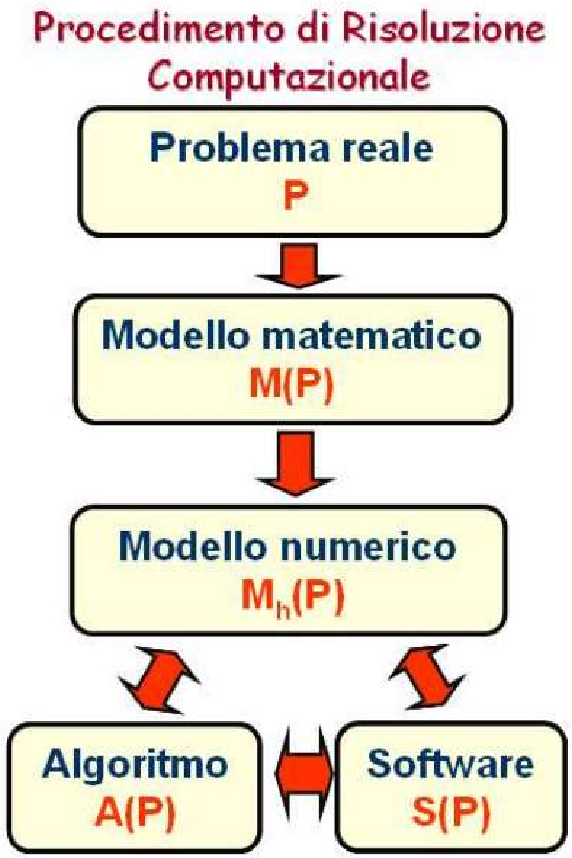
Previously on… CPD

- **Classificazione di Flynn -** MIMD (SM) multicore
- **I° nucleo computazionale: somma tra due vettori di lunghezza N – decomposizione del dominio (algoritmi full parallel)**
- **II° nucleo computazionale: somma di N numeri – decomposizione del dominio – collezione dei risultati (1-2 strategia)**
- **Parametri di valutazione algoritmo parallelo:**
  - **Speedup, efficienza, overhead, legge di Ware-Amdahl, isoefficienza, scalabilità**
- **Implementazione in ambiente multicore (usando openMP) dei nuclei computazionali:**

somma vettori – somma N numeri (1-2 strat) - calcolo PiGreco

Modellizzazione di problemi su larga scala

- Ricerca su internet
- Trasporto
- Pubblicità e Marketing
- Servizi bancari e finanziari
- Media e intrattenimento
- Meteorologia
- Assistenza sanitaria
- Sicurezza informatica
- Formazione



Usiamo quello che abbiamo studiato (I-II nucleo computazionale) per parallelizzare il

PRODOTTO SCALARE

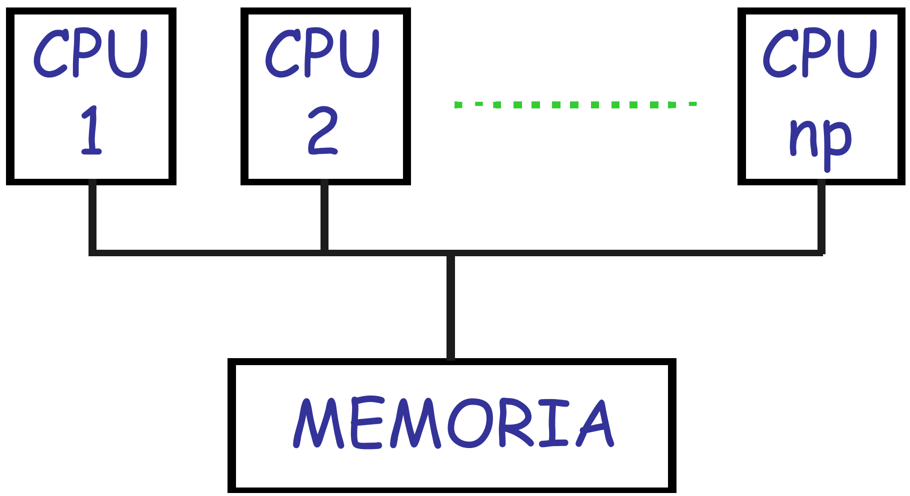
Input: $a = (a_0, a_1, a_2, \ldots\ldots, a_{N-1})$, $b = (b_0, b_1, b_2, \ldots\ldots, b_{N-1})$

Output: $c = a_0 \times b_0 + a_1 \times b_1 + a_2 \times b_2 + a_{N-1} \times b_{N-1}$

su un calcolatore parallelo

tipo MIMD

A MEMORIA CONDIVISA



Algoritmo: prodotto scalare

Su un calcolatore monoprocessore il prodotto scalare è calcolato eseguendo:

1. *N moltiplicazioni* una per volta (prodotto puntuale tra due vettori)

$$c_0 := a_0 \times b_0$$
$$c_1 := a_1 \times b_1$$
...
...
$$c_{N-1} := a_{N-1} \times b_{N-1}$$

È uguale alla somma tra vettori. Devo solo sostituire + con ×

2. *N-1 addizioni* una per volta (somma tra gli elementi di un vettore)

$$c := c_0$$
$$c := c + c_1$$
$$c := c + c_2$$
...
...
$$c := c + c_{N-1}$$

È uguale alla somma tra gli elementi di un vettore!

Prodotto scalare tra due vettori di dimensione N - algoritmo sequenziale

```
begin
     c := 0;
     for i=0 to N-1 do
          c := c + (a_i × b_i);
     endfor
end
```

Qual è

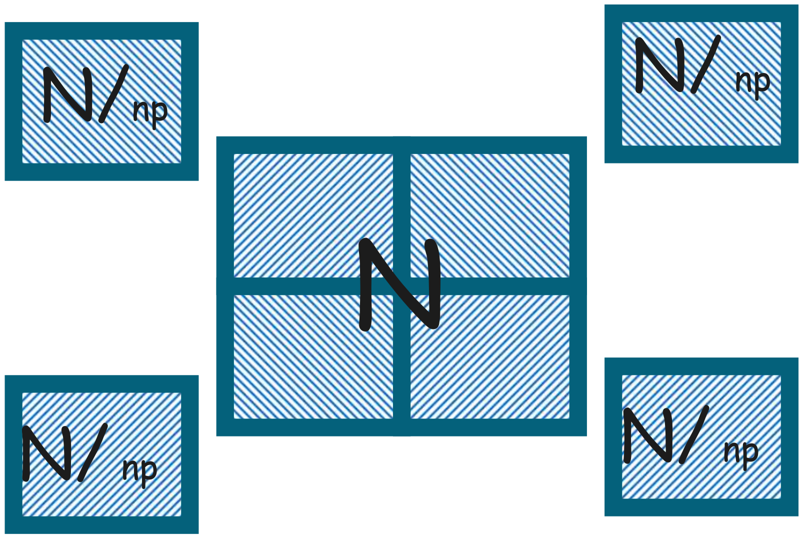
l'*ALGORITMO PARALLELO*?

Il parallelismo delle architetture MIMD

prodotto scalare

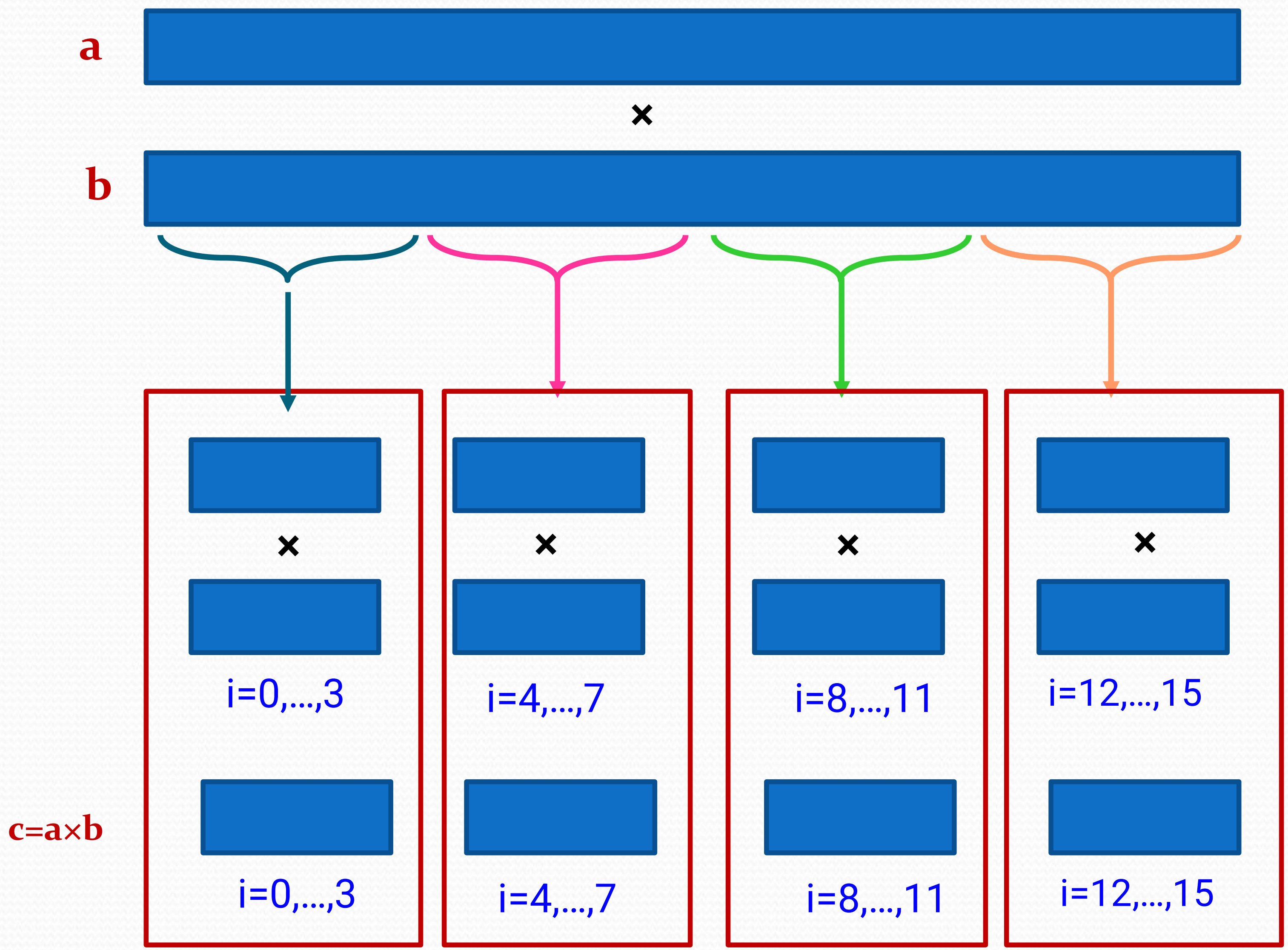
Se ho a disposizione np unità processanti, come posso procedere sfruttando il concetto di calcolo parallelo?

Decomporre un problema di dimensione N in **np** sottoproblemi di dimensione N/**np** e risolverli contemporaneamente usando **np** CPU



Strategia di parallelizzazione
prodotto scalare

Esempio: N=16, np=4 – fase1



Strategia di parallelizzazione
prodotto scalare

Esempio: N=16, np=4 – fase 2

$$\underbrace{\underbrace{c_0+\dots+c_3}_{s_0} \quad \underbrace{c_4+\dots+c_7}_{s_1} \quad \underbrace{c_8+\dots+c_{11}}_{s_2} \quad \underbrace{c_{12}+\dots+c_{15}}_{s_3}}_{s_0+s_1+s_2+s_3}$$

I° - II°
strategia
della somma

Calcolo di **speedup**, **overhead** ed **efficienza** (**def classica**)

algoritmo parallelo per il prodotto scalare

dim[a]=dim[b]=N

Complessità computazionale sequenziale

N prodotti + (N-1) somme

somme ~ prodotti

$$T_1(N) = 2N - 1$$

p=8, dim[a]=dim[b]=N

C_0 C_1 C_2 C_3 C_4 C_5 C_6 C_7

shared

Calcolo prodotti locali — N/p prodotti → **N/p**

+

Calcolo somme parziali — N/p -1 somme → **(N/p -1)**

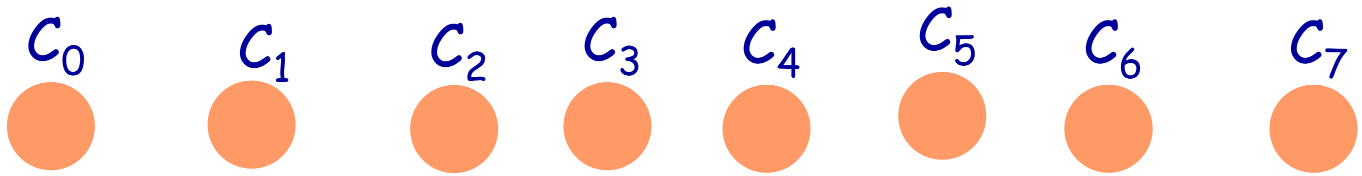
2 N/p - 1

… a questo punto devo decidere come voglio collezionare le somme parziali!

prodotto scalare

collezione 1 strategia

p=8, N



private

In sequenziale

$$T_1(N) = 2\,N - 1$$

$$S_p = T_1(N)/T_p(N) =$$

$$= [2\,N-1] \,/[(2\,N/p\,-1) + (p\,-1)\ \ ]$$

$$Oh = p\,T_p(N) - T_1(N) = p[(2\,N/p\,-1) + (p\,-1)\,] - (2N-1)$$

$$E_p = S_p\,/p = [2\,N-1]\,/p\,[(2\,N/p\,-1) + (p\,-1)\,]$$

prodotto scalare

collezione 2 strategia

p=8, N

c_0 c_1 c_2 c_3 c_4 c_5 c_6 c_7

s_0 s_1 s_2 s_3 s_4 s_5 s_6 s_7

shared

In sequenziale

$$T_1(N) = 2\ N - 1$$

$$S_p = T_1(N)/T_p(N) =$$

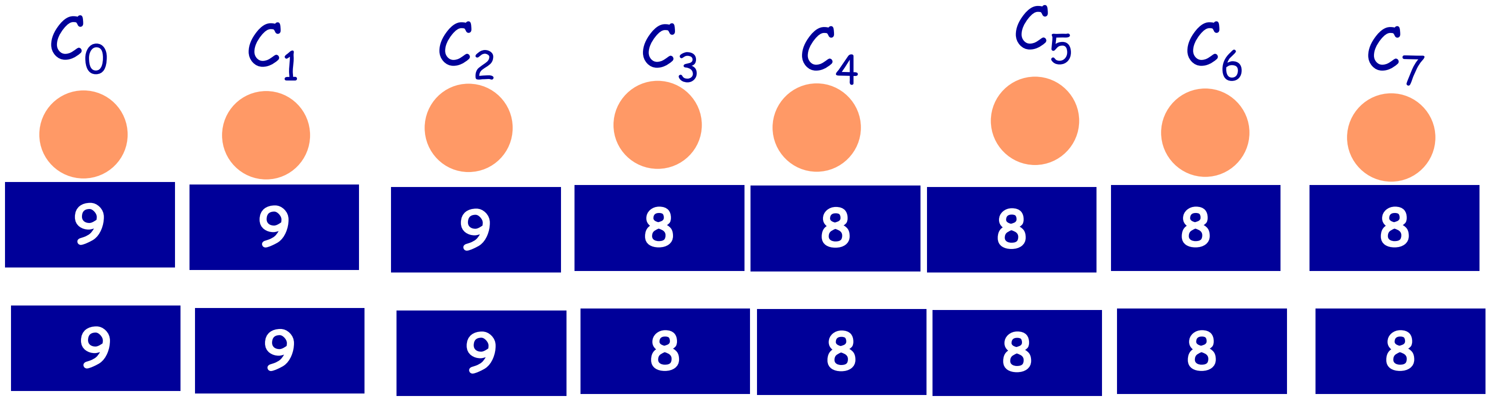
$$= [2\ N-1] /[(2\ N/p\ -1) + \log(p)\ ]$$

$$Oh = p\ T_p(N) - T_1(N) = p[(2\ N/p\ -1) + \log(p)\ ] - (2N-1)$$

$$E_p = S_p /p = [2\ N-1] /p\ [(2\ N/p\ -1) + \log(p)\ ]$$

p=8, N=67

Cosa succede se N non è esattamente divisibile per p???



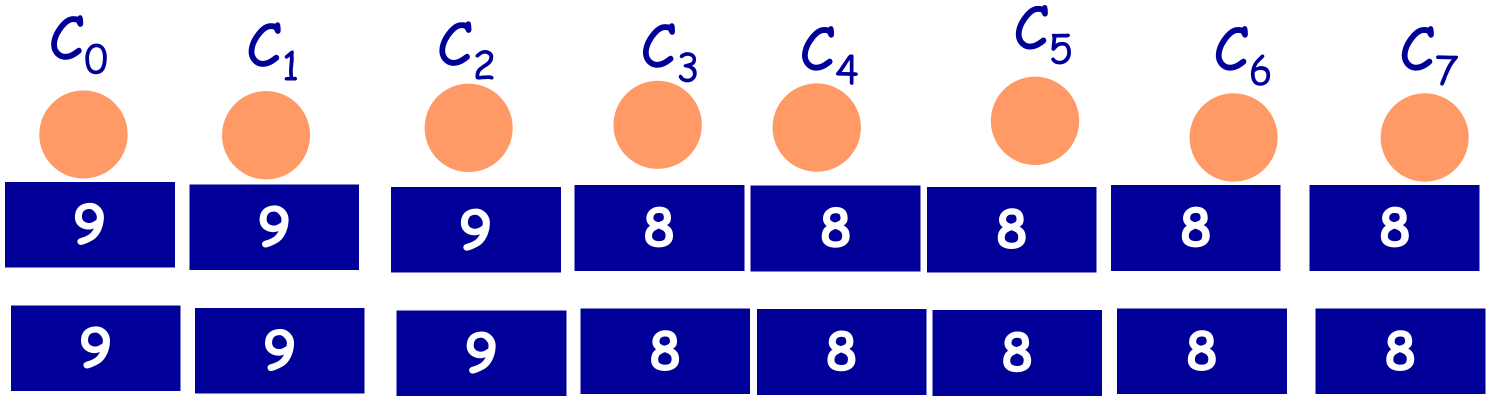
Calcolo prodotti locali

int(N/p)+1 prodotti

Come se tutti i processori avessero due vettori da 9 elementi tra cui fare il prodotto puntuale!

p=8, N=67

Cosa succede se N non è esattamente divisibile per p???



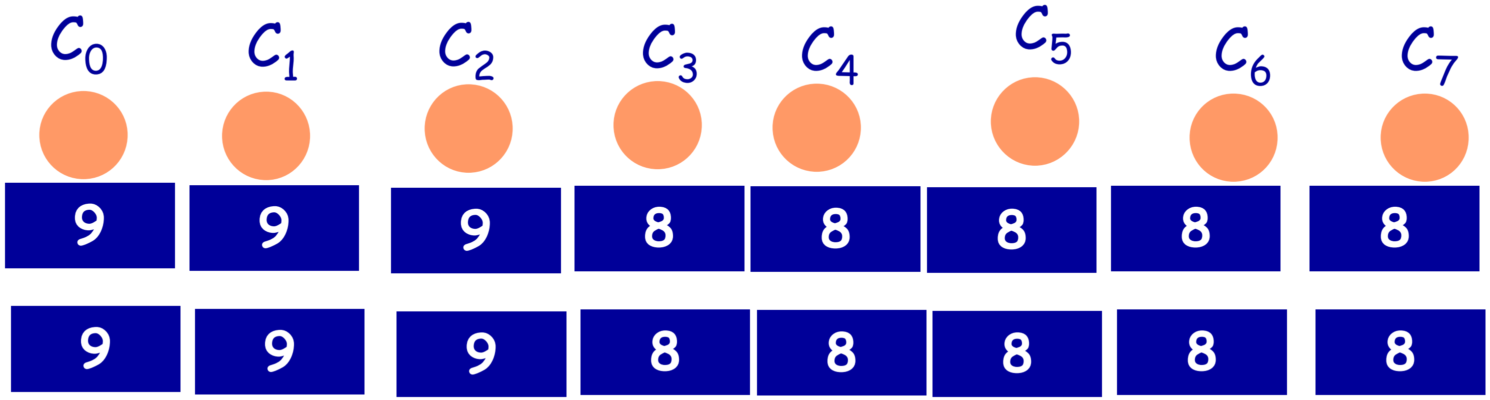
Calcolo somme parziali locali

int(N/p) somme

Come se tutti i processori avessero 9 elementi da somma tra loro!

p=8, N=67

Cosa succede se N non è esattamente divisibile per p???



Fase Locale

int(N/p)+1 prodotti + int(N/p) somme

2 int(N/p) + 1

La fase di collezione dei risultati, ovviamente resta invariata…

ma attenzione!

p, N

Cosa succede se N non è esattamente divisibile per p???

In sequenziale

$$T_1(N) = 2\,N - 1$$

$T_p(N) = 2\ \mathrm{int}(N/p) + 1 + p-1$ 1 strategia

$T_p(N) = 2\ \mathrm{int}(N/p) + 1 + \log(p)$ 2 strategia

isoefficienza

Definizione

L'ISOEFFICIENZA

è una funzione di tre variabili p_0 , p_1 , n_0

e definisce la costante che lega la nuova dimensione del problema da scegliere n_1 per valutare la **scalabilità** di un algortimo

$$I(p_0 , p_1 , n_0) = O_h(p_1 , n_1)/O_h(p_0 , n_0)$$

Calcolo isoefficienza

prodotto scalare di 2 vettori di dimensione N

$O_h(p, N) = T_1(N) - p\ T_p\ (N) =$

I strategia

$= p[(2\ N/p - 1) + (p - 1)\ ] - (2N-1) =$

$= 2pN/p - p + p^2 - p - 2N + 1 =$

$= 2N + p^2 - 2N + 1 - 2p = p^2 - 2p + 1$

$\Rightarrow$ $I = (p_1^2 - 2\ p_1 +1)/(p_0^2 - 2\ p_0 +1)$

È sempre la 1 strategia di collezione della somma

Calcolo isoefficienza

prodotto scalare di 2 vettori di dimensione N

II strategia

$O_h(p, N) = T_1(N) - p\ \ T_p\ (N) =$

$= p[(2\ N/p\ -1) + \log(p)\ ] - (2N-1) =$

$= 2pN/p - p + p\ \log(p) - 2N + 1 =$

$= 2N - p + p\ \log(p) - 2N + 1 = p\ \log(p) - p + 1$

$\Rightarrow$ $I = (p_1 \log(p_1) - p_1 + 1)/(p_0 \log(p_0) - p + 1)$

È sempre la 2 strategia di collezione della somma

Caratterizziamo ora lo **speedup** **usando la legge di Ware-Amdahl**

generalizzata

$$\boxed{S_p = \frac{1}{\alpha_p + \sum_{k=2}^{p-1} \frac{\alpha_k}{k} + \alpha_1}}$$

p=8, dim[a]=dim[b]=32, nloc=4

C_0 C_1 C_2 C_3 C_4 C_5 C_6 C_7

In sequenziale 2N-1=63 operazioni

1 fase (tutta parallela)

nloc = 4 prodotti
+
nloc-1 = 3 somme

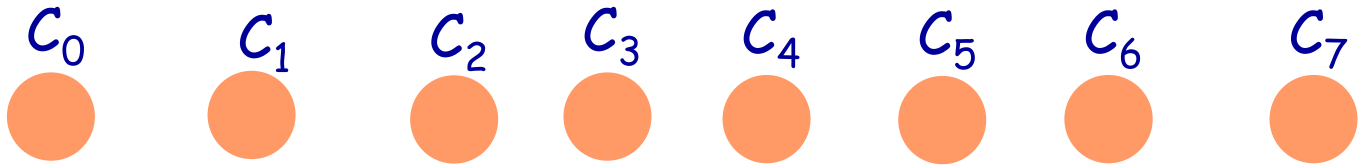
7 operazioni fatte contemporaneamente da 8 core

7x8=56 delle 63 operazioni

$$\alpha_8 = 56/63$$

... a questo punto devo decidere come voglio collezionare le somme parziali!

p=8, N=32, nloc=4



In sequenziale

In sequenziale 2N-1=63 operazioni

$$\alpha_8 = 56/63$$

I strategia

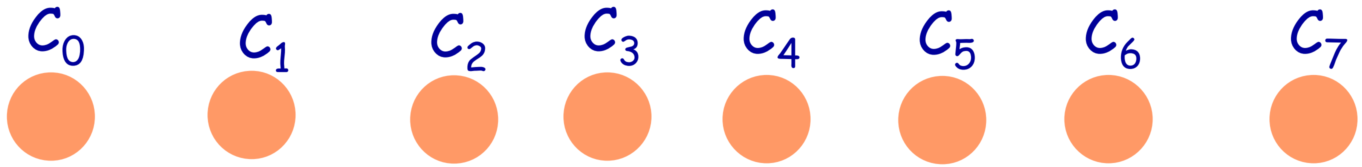
$$\alpha_7 = \alpha_6 = \alpha_5 = \alpha_4 = \alpha_3 = \alpha_2 = 0$$

7 somme fatte da 1 solo core

$$\boxed{\alpha_1 = (7 \cdot 1)/63}$$

È sempre la 1 strategia di collezione della somma

$p=8, \quad N=32, nloc=4$



In sequenziale

In sequenziale 2N-1=63 operazioni

$$\alpha_8 = 56/63$$

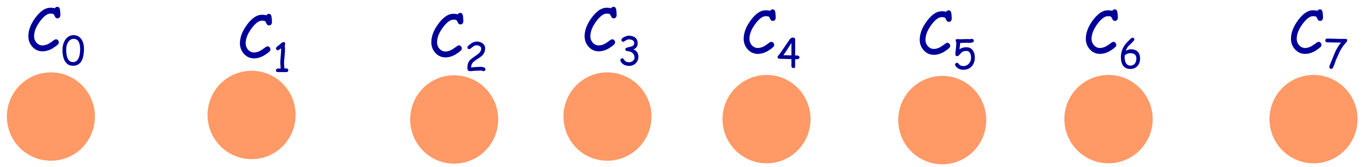
II strategia

$$\alpha_7 = \alpha_6 = \alpha_5 = 0$$

1 somma fatta da 4 core

$$\boxed{\alpha_4 = (1 \cdot 4)/63}$$

p=8, N=32, nloc=4



In sequenziale

In sequenziale 2N-1=63 operazioni

$$\alpha_8 = 56/63$$

$$\alpha_7 = \alpha_6 = \alpha_5 = 0$$

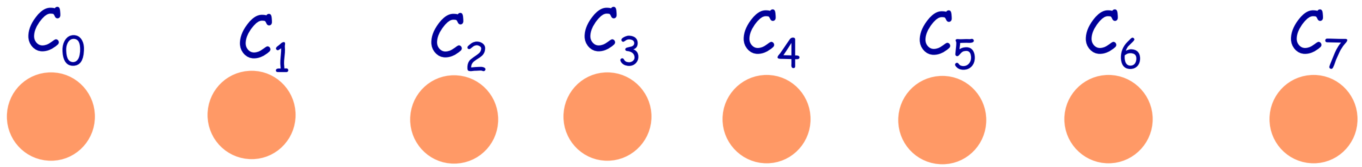
II strategia

$$\alpha_3 = 0$$

1 somma fatta da 2 core

$$\boxed{\alpha_2 = (1 \cdot 2)/63}$$

p=8, N=32, nloc=4



In sequenziale

In sequenziale 2N-1=63 operazioni

II strategia

$$\alpha_8 = 56/63$$

$$\alpha_7 = \alpha_6 = \alpha_5 = 0$$

$$\alpha_3 = 0$$

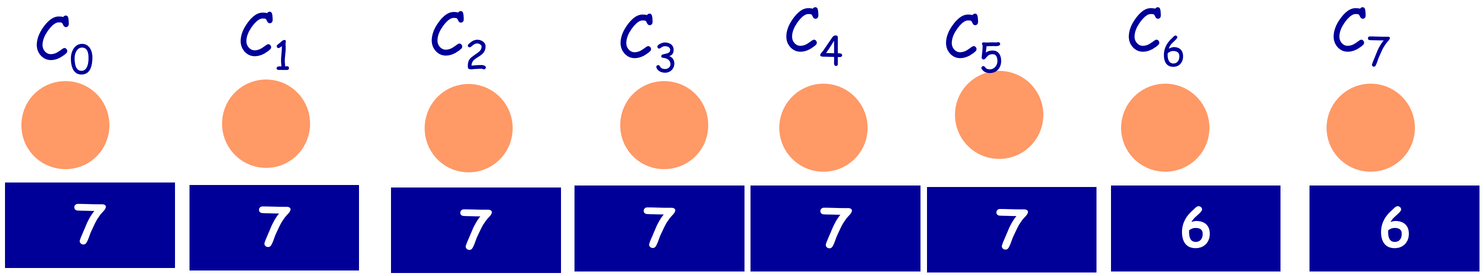
$$\alpha_2 = (1 \cdot 2)/63$$

1 somma fatta da 1 core

$$\boxed{\alpha_1 = (1 \cdot 1)/63}$$

È sempre la 2 strategia di collezione della somma

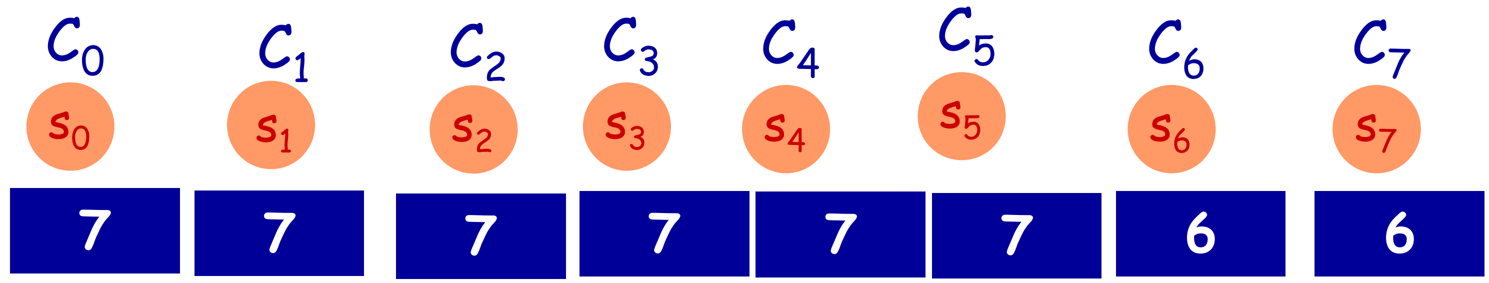
Cosa succede se N non è esattamente divisibile per p???



$p=8$, $N=54$

Quale che sia la strategia per la collezione delle somme parziali, quello che cambia è solo α_8

p=8, N=54, nloc=6, r=6



In sequenziale

2 N - 1 = 107 operazioni

calcolo locale

6 core - 2nloc-1=13 operazioni
2 core - 2nloc-1=11 operazioni

13 operazioni fatte contemporaneamente da 6 processori/core

+

11 operazioni fatte contemporaneamente da 2 processori/core

$$\alpha_8 = (13\times 6+11\times 2)/107=(78+22)/107=100/107$$