Calcolo Parallelo e Distribuito

I parametri di valutazione di un algoritmo parallelo:
isoefficienza

Docente: Prof. L. Marcellino

Tutor: Prof. P. De Luca

Domanda

Quando è possibile

ottenere

speed-up prossimi allo

speed-up ideale ?

In generale MIMD-SM

La complessità computazionale di un algoritmo parallelo, che lavora con *p* unità processanti, comprende 2 componenti:

- $\mathbf{T_s}$ frazione delle operazioni per eseguire la parte seriale
- $\mathbf{T_c/p}$ frazione delle operazioni per eseguire la parte parallela

Legge di Ware-Amdhal

$T_s = \alpha \; , \; T_c = 1 - \alpha$

$T_p = T_s + T_c / p \Rightarrow T_p = \alpha + (1 - \alpha) / p$

$$S_p = \frac{T_1}{T_p} = \frac{1}{\alpha + (1-\alpha)/p}$$

Analizziamo questa caratterizzazione dello speed-up

l'andamento dello speed-up, caratterizzato dalla legge di W-A all'*aumentare del numero p delle CPU*, ovvero per p→∞

$$S_p = \frac{1}{\alpha + (1-\alpha)/p} \xrightarrow[p \to \infty]{} \frac{1}{\alpha}$$

$$S_p \to \frac{1}{\alpha}$$

Analizziamo questa caratterizzazione dello speed-up

$$S_p = \frac{1}{\underbrace{\boxed{\alpha}}_{0} + \frac{(1-\underbrace{\boxed{\alpha}}_{0})}{p}} \xrightarrow[\alpha \to 0]{} p$$

$S_p \to p$

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

Per incrementare le performance

Individuare e fissare il miglior numero p

di CPU

e

aumentare la dimensione del problema

Cosa abbiamo imparato dalla legge di W-A per l'algoritmo parallelo

della somma?

1.

Fissato il size n del problema e aumentando il numero p di CPU

...esiste p_{max} miglior numero di processori per risolvere il problema con l'algoritmo in esame

superato questo valore le prestazioni peggiorano!!!

2.

Fissato il numero p delle CPU e aumentando il size n del problema

...esiste n_{max} la massima dimensione che la macchina può memorizzare/elaborare

superato questo valore potrei avere problemi con la memoria hardware!!!

Domanda

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

- somma di due vettori di lunghezza N
- somma di N numeri

Domanda

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

- somma di due vettori di lunghezza N
- somma di N numeri

Esempio: somma di n numeri in parallelo **(Ist)**

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	$(1-\alpha)/p$	$S_p(n)$	$E_p(n)$
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 tende a zero!

Esempio: somma di n numeri in parallelo

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* è "costante" !

Esempio: somma di n numeri in parallelo

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	$(1-\alpha)/p$	$S_p(n)$	$E_p(n)$
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!

Esempio: somma di n numeri in parallelo

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	$(1-\alpha)/p$	$S_p(n)$	$E_p(n)$
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,

con un rapporto fissato,

le prestazioni

dell'algoritmo parallelo

non degradano molto velocemente

Ho fissato $n/p=4$ e se la regola

che deve legare questi due valori

non fosse questa?

Qual è il valore giusto per n/p?

Aumentando n e p in rapporto **costante**

deve essere

$$\boxed{E_p(n) = cost}$$

Quale deve essere il rapporto costante tra n e p???

Devo trovare **I=n/p**, tale che l'efficienza resti costante

Allora… I non può essere una costante, deve essere una funzione!

$$I = I(n_0, p_0, p_1)$$

la funzione I è detta ISOEFFICIENZA

esprime

la legge secondo cui si deve

scegliere la nuova dimensione n_1

affinché si possa verificare che l'efficienza

resti costante!

Domanda

Come si calcola l'isoefficienza

per un qualsiasi algoritmo ?

OVVERO

Nel passare da p_0 a p_1 processori con $p_1 > p_0$,

come deve aumentare (*scalare*)

la dimensione del problema da risolvere

affinché l'efficienza scalata sia costante ?

Overhead totale O_h

$$\boxed{O_h = pT_p - T_1} \longrightarrow T_p = (O_h + T_1)/p$$

OVERHEAD totale

$$S_p = \frac{p}{\frac{O_h}{T_1} + 1}$$

Come calcolare l'isoefficienza?

Imporre:

$$E_{p_0}(n_o) = E_{p_1}(n_1)$$

$$S_p = \frac{p}{\frac{O_h}{T_1} + 1}$$

Come calcolare l'isoefficienza?

$$E_{p_0}(n_o) = \frac{S_{p_o}(n_o)}{p_0} \quad = \quad E_{p_1}(n_1) = \frac{S_{p_1}(n_1)}{p_1}$$

$$\boxed{S_p = \frac{p}{\frac{O_h}{T_1} + 1}}$$

Come calcolare l'isoefficienza?

$$E_{p_0}(n_o) = \frac{S_{p_o}(n_o)}{p_0} = \frac{p_o}{p_o(\frac{O_h}{T_1(n_0)}+1)} = \frac{1}{\frac{O_h}{T_1(n_0)}+1} = E_{p_1}(n_1) = \frac{S_{p_1}(n_1)}{p_1} = \frac{p_1}{p_1(\frac{O_h}{T_1(n_1)}+1)} = \frac{1}{\frac{O_h}{T_1(n_1)}+1}$$

$$S_p = \frac{p}{\frac{O_h}{T_1}+1}$$

$$\frac{O_h(n_0,p_0)}{T_1(n_0)} = \frac{O_h(n_1,p_1)}{T_1(n_1)}$$

$$T_1(n_1) = \boxed{\frac{O_h(n_1,p_1)}{O_h(n_0,p_0)}}\, T_1(n_0)$$

I

ISOEFFICIENZA

Calcoliamo l'isoefficienza nella somma IIst

$O_h(n,p) = p \log_2 p$

$T_1(n) = n-1$

$$T_1(n_1) = \frac{O_h(n_1,p_1)}{O_h(n_0,p_0)} T_1(n_0)$$

$$\boxed{n_1 - 1 = \frac{p_1 \log_2 p_1}{p_0 \log_2 p_0} (n_0 - 1)}$$

ISOEFFICIENZA

Nel passare da p_0 a p_1 con $p_1 > p_0$, la dimensione del problema deve aumentare, rispetto alla dimensione iniziale n_0, del fattore

$I = (p_1 \log_2 p_1)/(p_0 \log_2 p_0)$

In generale…

Un algoritmo si dice scalabile se

l'efficienza rimane costante

al crescere del numero dei processori e

della dimensione del problema,

in un rapporto costante pari a:

$$I = \frac{O_h(n_1, p_1)}{O_h(n_0, p_0)}$$

La somma è un algoritmo scalabile?

I

$$\frac{n_1}{n_0} = \frac{p_1 \log_2 p_1}{p_0 \log_2 p_0}$$

$p_0 = 4$ $\quad n_0 = 64$

$p_1 = 8$ $\quad n_1 = ?\ 3 \times n_0 = 192$

$p_2 = 16$ $\quad n_2 = ?\ 8 \times n_0 = 512$

$I = (8 \times 3) / (4 \times 2) = 3$

$I = (16 \times 4) / (4 \times 2) = 8$

La somma è un algoritmo scalabile?

I

$$\frac{n_1}{n_0} = \frac{p_1 \log_2 p_1}{p_0 \log_2 p_0}$$

$p_0 = 4$ $n_0 = 64$ $T_4(64) = 17$

$p_1 = 8$ $n_1 = 192$ $T_8(192) = 26$

$p_2 = 16$ $n_2 = 512$ $T_{16}(512) = 35$

$$\boxed{T_p(n) = \left(\frac{n}{p} - 1 + \log_2 p\right) t_{calc}}$$

somma di n numeri con p processori: isoefficienza

Calcoliamo l'efficienza secondo i T_p calcolati

Efficienza

n \ p	1	4	8	16
64	1.0	0.91	0.57	0.33
192	1.0	0.97	0.91	0.79
512	1.0	0.97	0.96	0.91

L'efficienza rimane costante

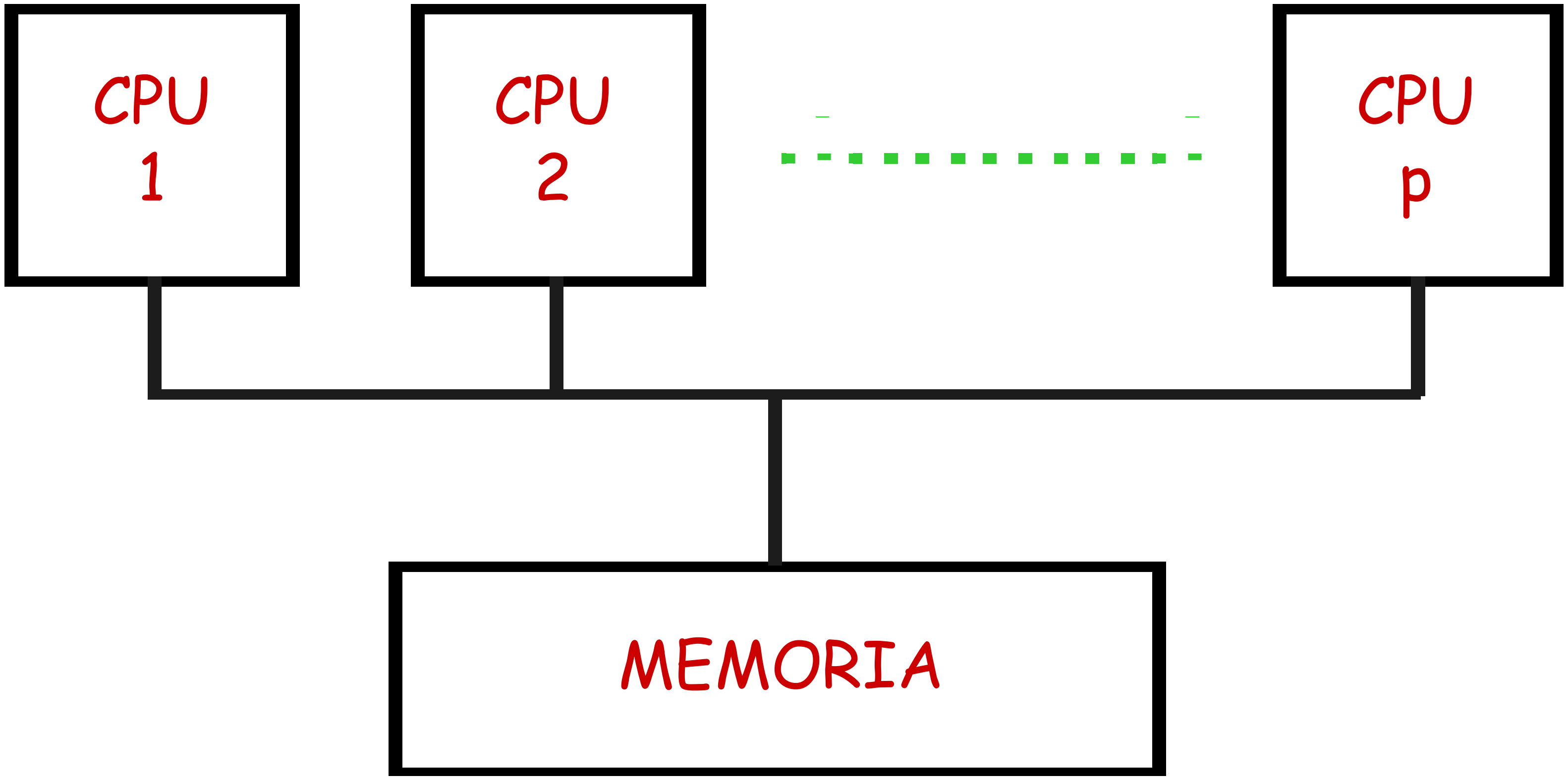
al crescere di p e di n con la legge di isoefficienza

La somma è scalabile!

incominciamo a vedere
come effettuare la stima
di queste metriche
e
come usare queste informazioni nella
valutazione delle strategie di
parallelizzazione

Calcoliamo S_p, E_p, Oh per i nuclei computazionali che stiamo studiando e implementando

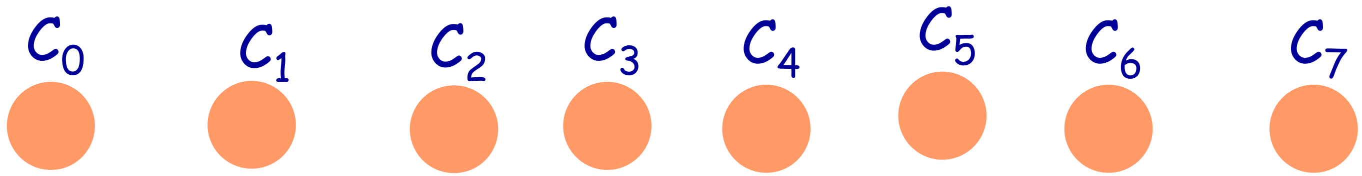
Calcolatori MIMD a memoria condivisa (shared-memory)



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

algoritmo parallelo per
la somma di due vettori di
dimensione N

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



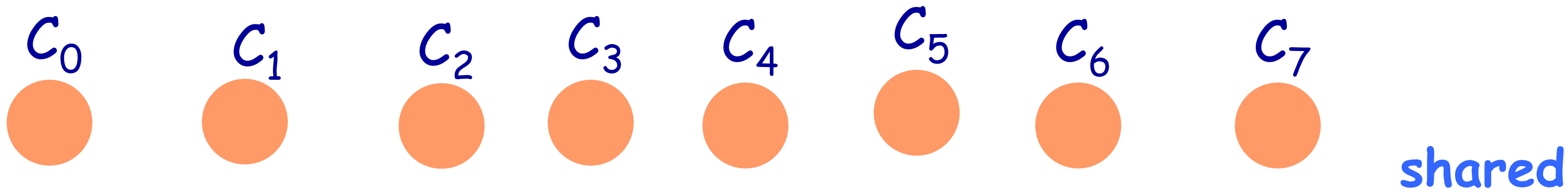
shared

Calcolo somme vettori parziali

N/p somme

N/p

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



In sequenziale

$$T_1(N) = N$$

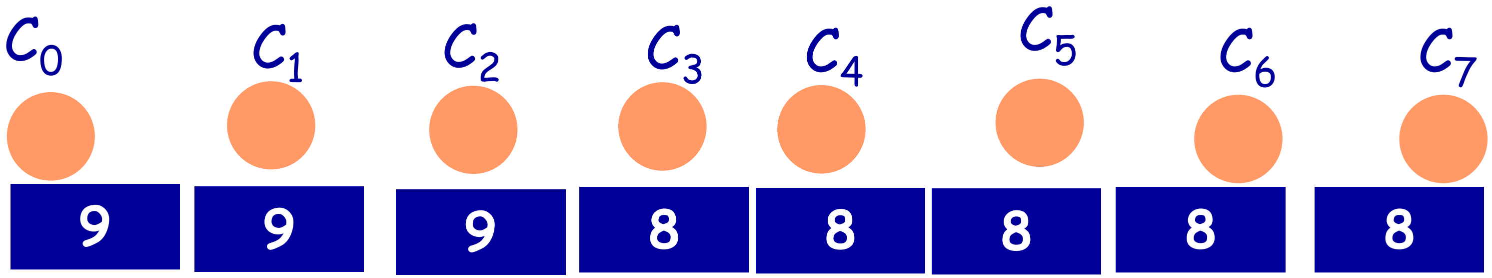
$$S_p = T_1(N)/T_p(N) =$$

$$= N/(N/p) = p$$

$$Oh = p\,T_p(N) - T_1(N) = p(N/p) - N = 0$$

$$E_p = S_p/p = p/p = 1$$

$p=8, \quad N=67$



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

Alcuni core faranno 1 somma in più:

Es: N=67, p=8

$T_p(N) =$

parte
intera

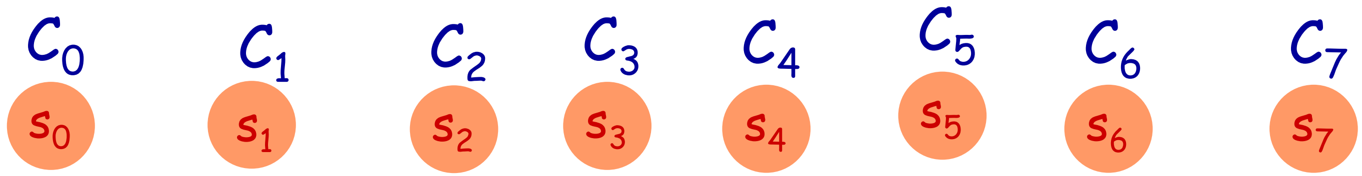
$= \boxed{[(\text{int}(N/p)} + 1)$

Come se tutti i core avessero 8+1 = 9 elementi!

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

algoritmo parallelo della
somma
di N numeri

I strategia p=8, N



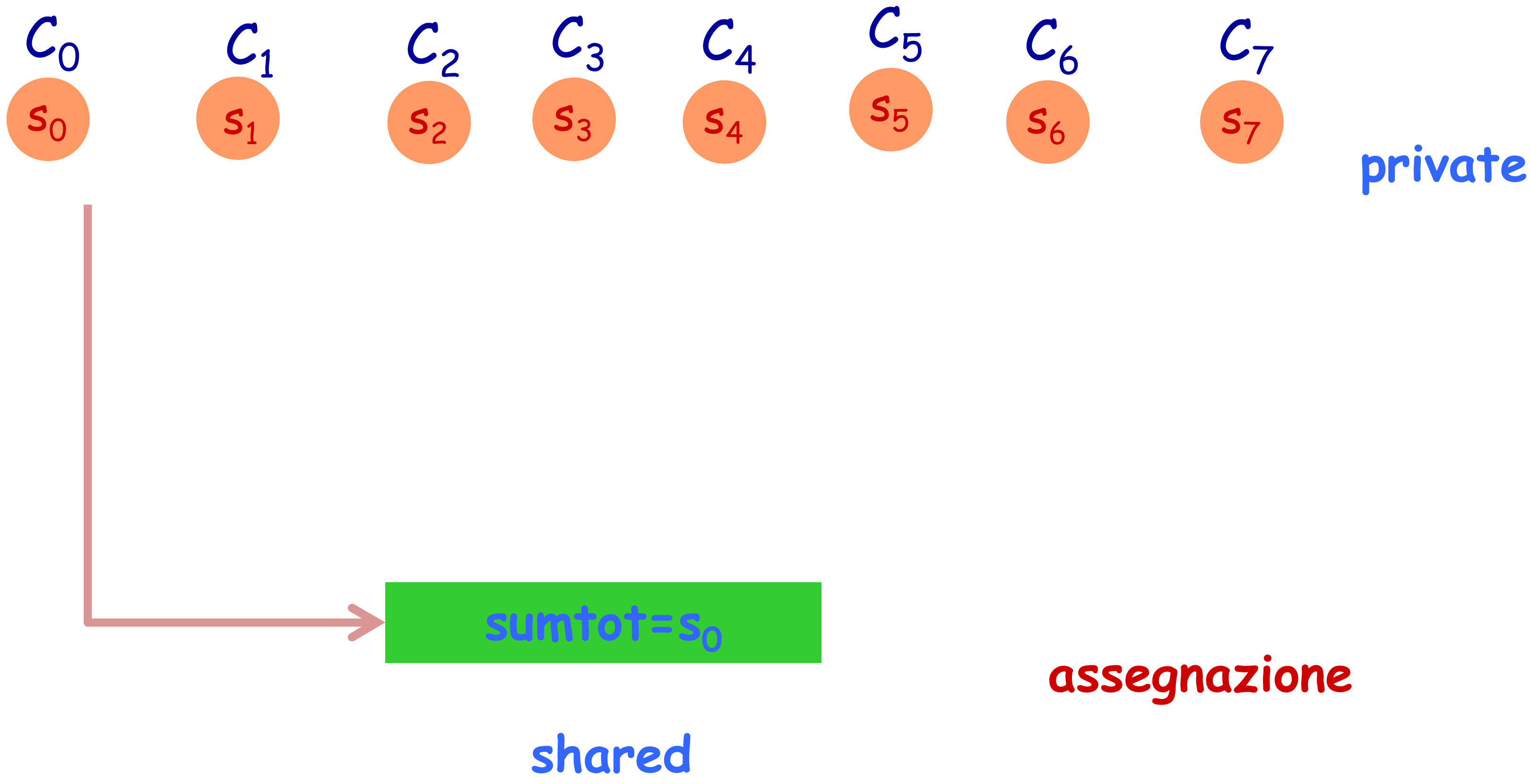
private

Calcolo somme parziali

N/p -1 somme

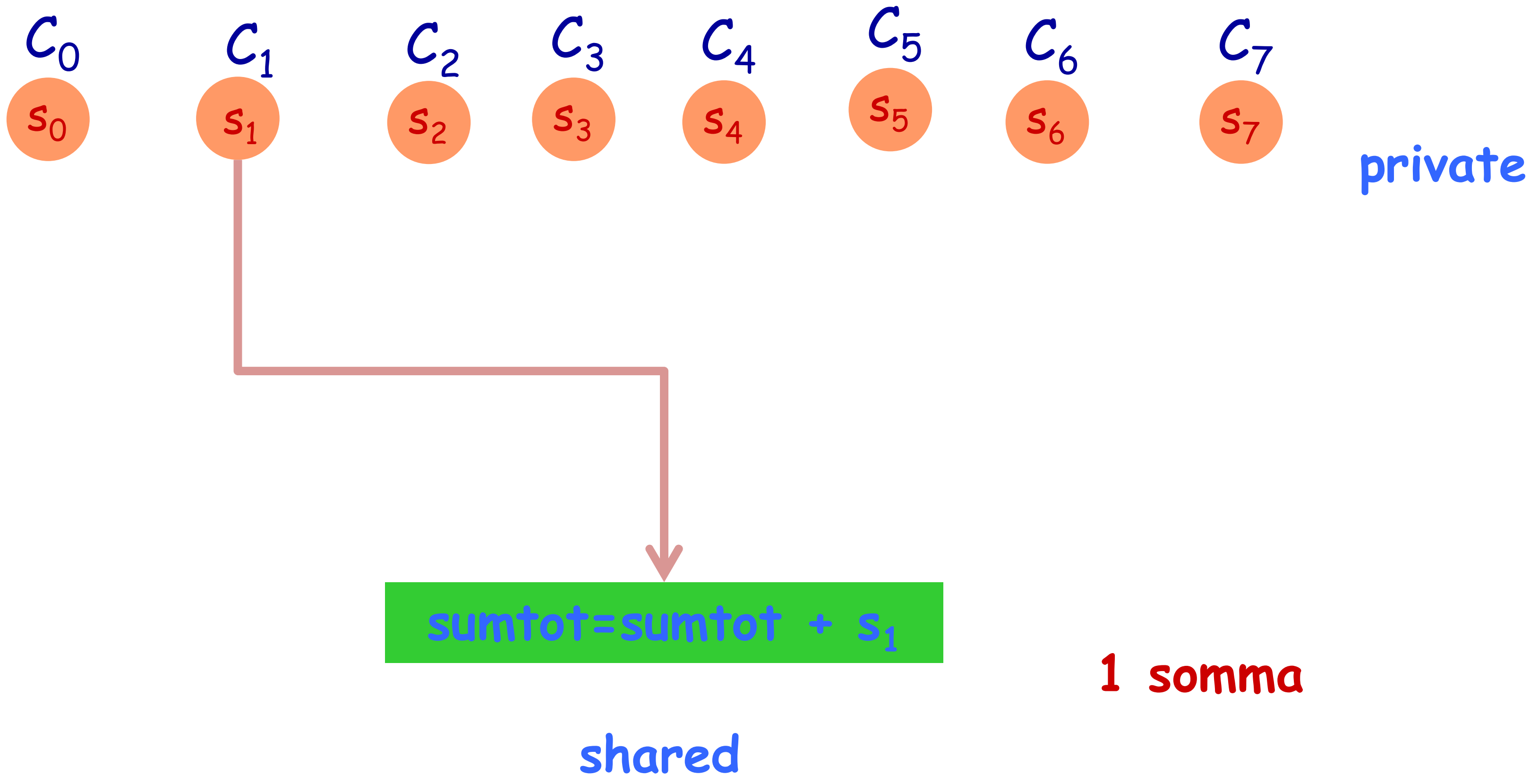
(N/p -1)

I strategia p=8, N



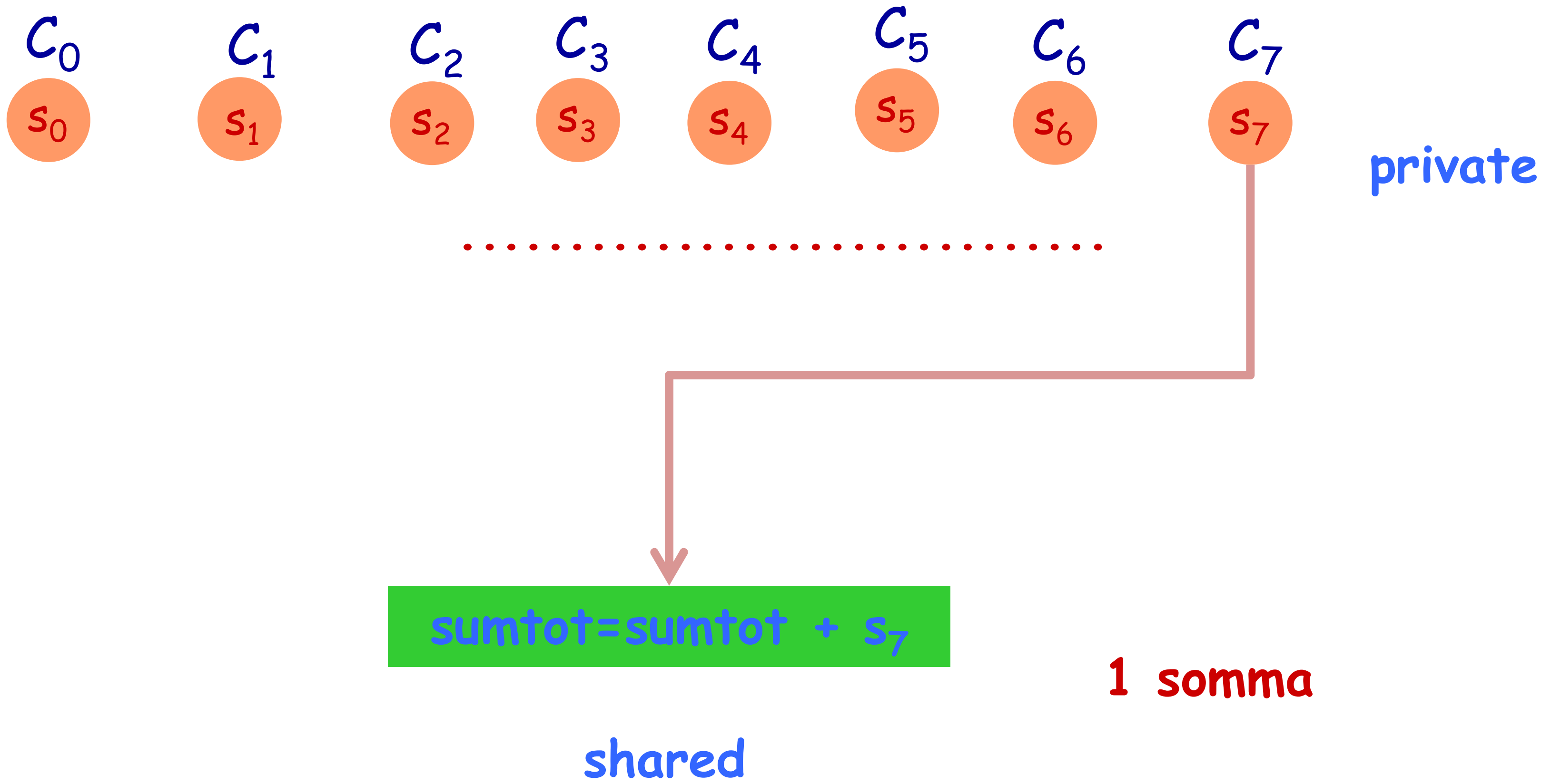
somma N numeri

I strategia p=8, N



somma N numeri

I strategia p=8, N



somma N numeri

I strategia $\quad p=8, \quad N$

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

private

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

+

somma totale — p-1 somme → (p -1)

(N/p -1) + (p -1)

somma N numeri

I strategia $\quad p=8, \quad N$

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

private

In sequenziale

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

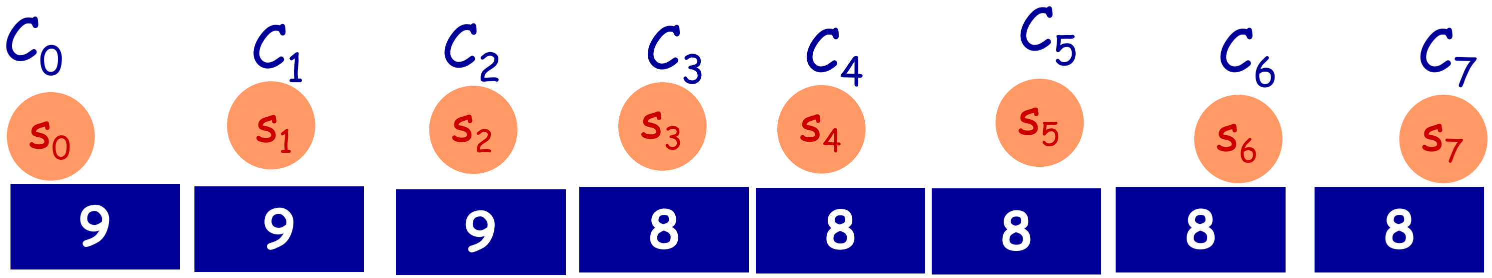
$$S_p = T_1(N)/T_p(N) =$$

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

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

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

I strategia $p=8, \quad N=67$



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

Alcuni core faranno 1 somma in più nella fase puramente parallela:

Es: N=67, p=8

parte intera

$T_p(N) =$

$= [(\mathrm{int}(N/p)) + (p-1)]$

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

II strategia p=8, N

C_0 C_1 C_2 C_3 C_4 C_5 C_6 C_7

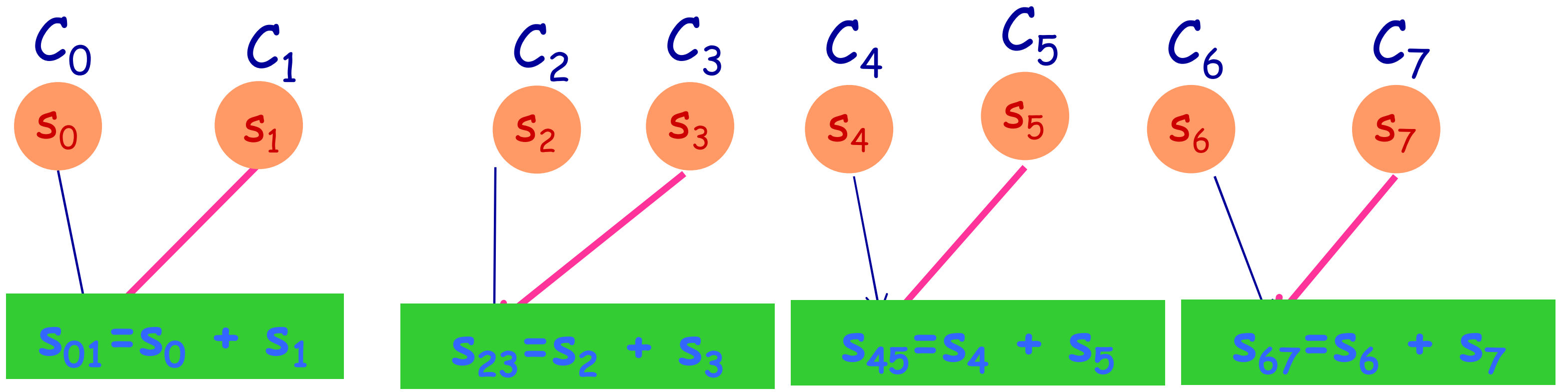
s_0 s_1 s_2 s_3 s_4 s_5 s_6 s_7

shared

Calcolo somme parziali

N/p -1 somme → **(N/p -1)**

II strategia p=8, N

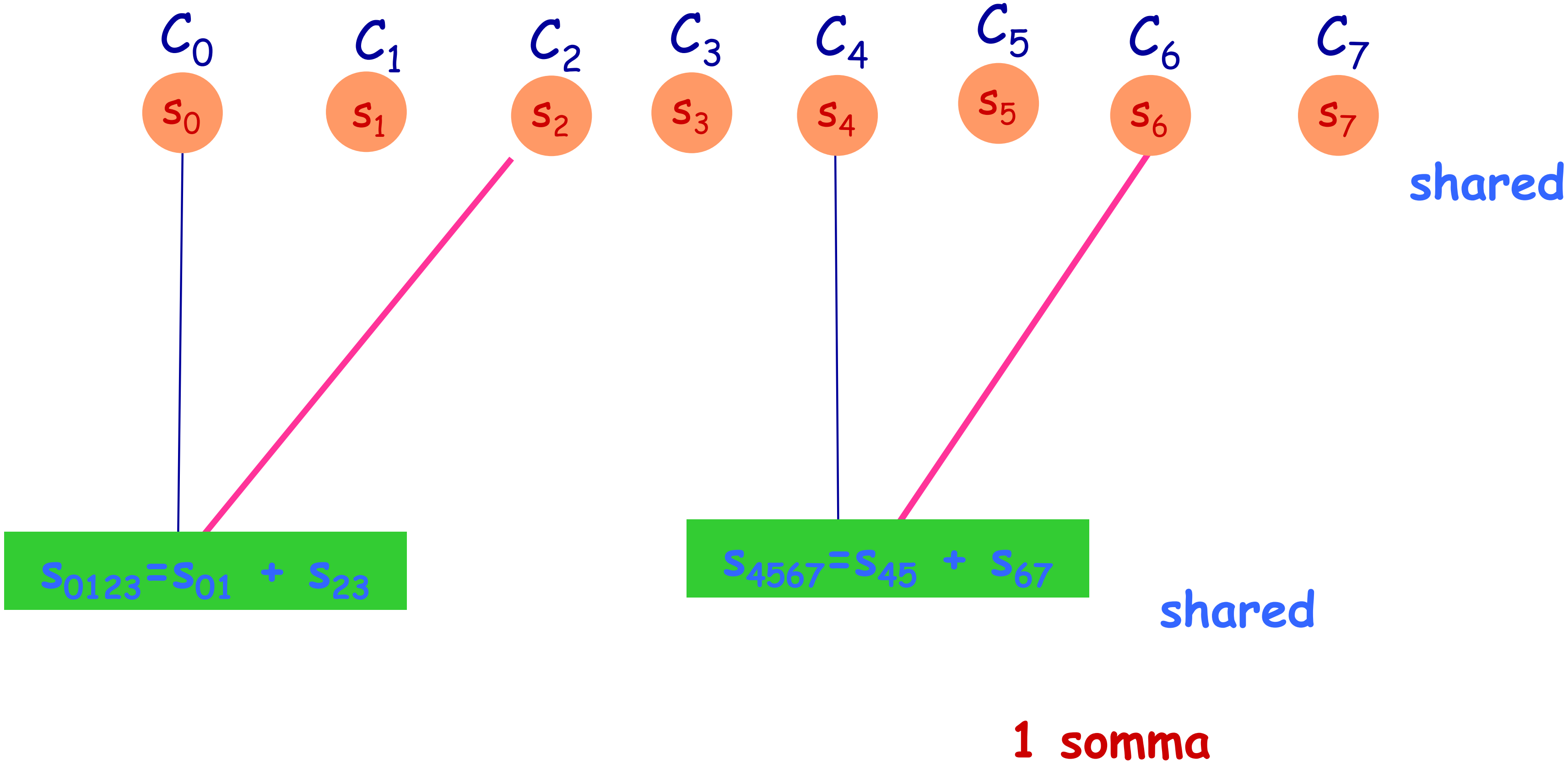


shared

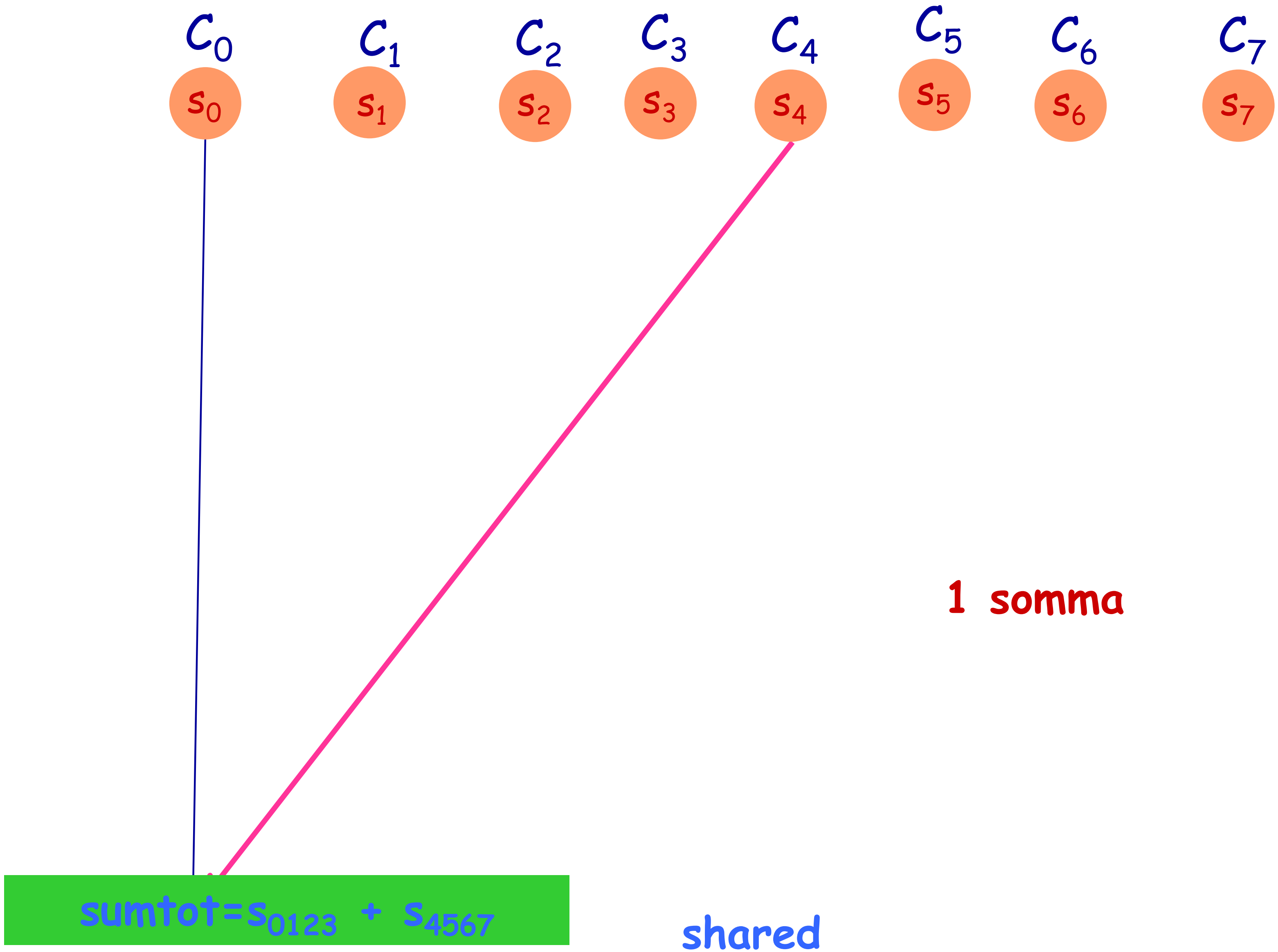
shared

1 somma

II strategia p=8, N



II strategia p=8, N



shared

1 somma

shared

II strategia $p=8$, N

C_0 s_0 C_1 s_1 C_2 s_2 C_3 s_3 C_4 s_4 C_5 s_5 C_6 s_6 C_7 s_7

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

\+

somma totale — log(p) somme → log(p)

(N/p -1) + log(p)

II 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

In sequenziale

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

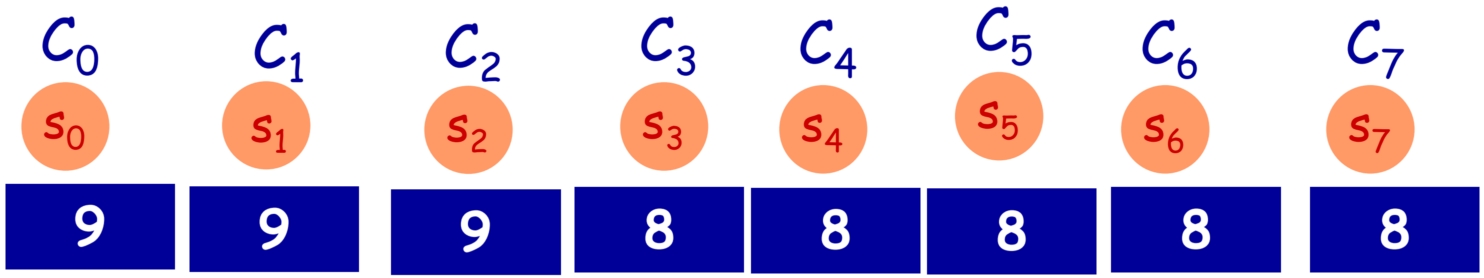
$$S_p = T_1(N)/T_p(N) =$$

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

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

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

II strategia $p=8$, N



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

Alcuni core faranno 1 somma in più nella fase puramente parallela:

Es: $N=67$, $p=8$

$T_p(N)$ =

$= [\text{int}(N/p)+ \log(p)]$