SPEED-UP $$S(n, p(n)) = \frac{T(n,1)}{T(n, p(n))}$$

Desideriamo $S(n, p(n)) \to \infty$ ma
come cresce $p(n)$?

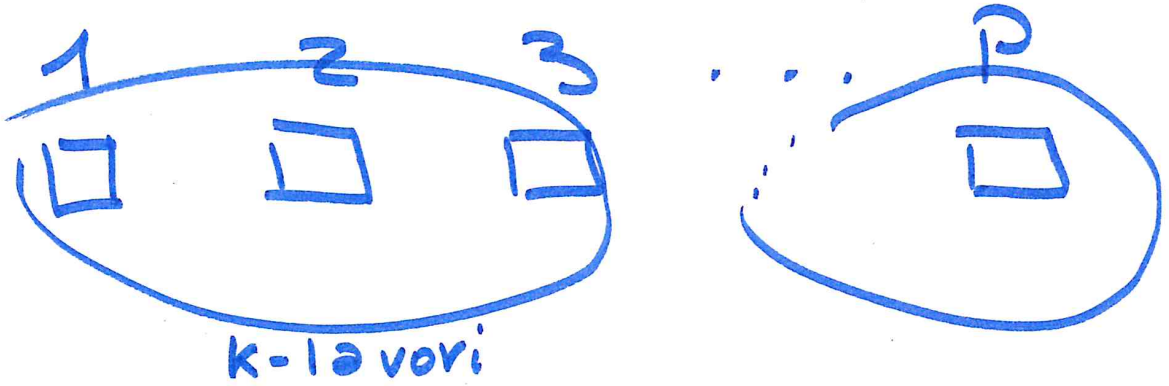
EFFICIENZA $$E(n, p(n)) = \frac{T(n,1)}{p(n) \cdot T(n, p(n))}$$

$$0 \leq E(n, p(n)) \leq 1$$

Se $E \to 0$ dato che $T(n, p(n)) = \sigma(T(n,1))$
è $p(n)$ a crescere troppo

⇒ Principio di Wyllie

Algoritmo parallelo

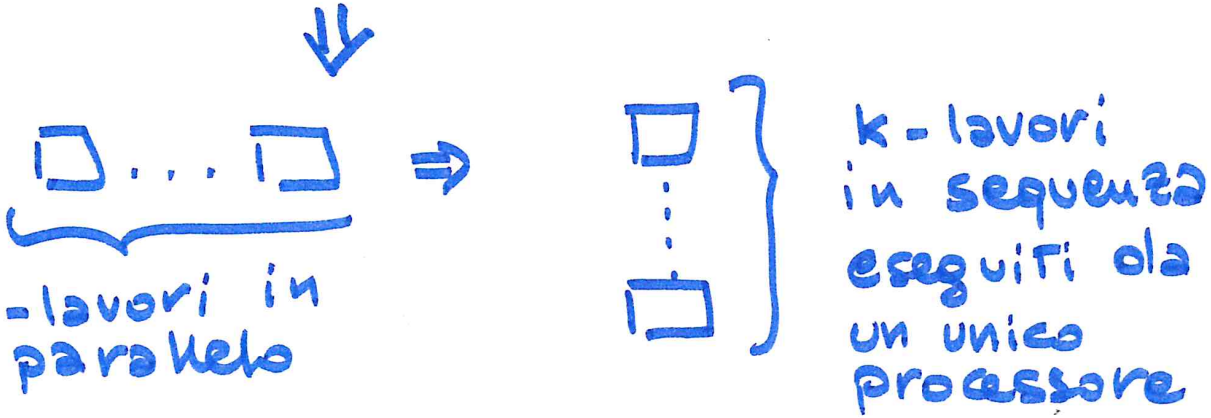


Tempo $T(n, P)$

i-esimo passo $\rightarrow t_i(n)$

$T(n, P/k) = ?$

Utilizzo adesso P/k processori



$$T(n, P/k) \leq \sum_{i=1} k\, t_i(n) = k \sum_i t_i(n)$$

$$= k\, T(n, P)$$

Quanto vale $E(n, p/k) = ?$

$$E(n, p/k) = \frac{T(n,1)}{p/k \; T(n, p/k)} \geq \frac{T(n,1)}{\frac{p}{\cancel{k}} \cdot \cancel{k} \, T(n,p)} = E(n,p)$$

La formula mostra che diminuendo i processori migliora il parametro efficienza (cioè rende vera la ricetta di Wyllie)
Vediamo se $k \to p$

$$1 = E(n,1) = E(n, p/p) \geq E(n, p/k) \geq E(n,p)$$

<u>Attenzione</u> però a mantenere $T(n, p/k) = \Theta(T(n,1))$
(perché $E(n,1) = 1$, ma $T(n, p=1) = T(n,1)$ cioè sequenziale)

SOMMATORIA

Motivazioni

- Tecnica: scomposizione del problema in sottoproblemi e fusione dei risultati
- Schema: per la soluzione di altre operazioni associative
- modulo: sottoprdolema di altri problemi

SOMMATORIA

INPUT: $M[1], M[2], \ldots, M[n]$

OUTPUT: $M[n] = \sum_{i=1}^{n} M[i]$

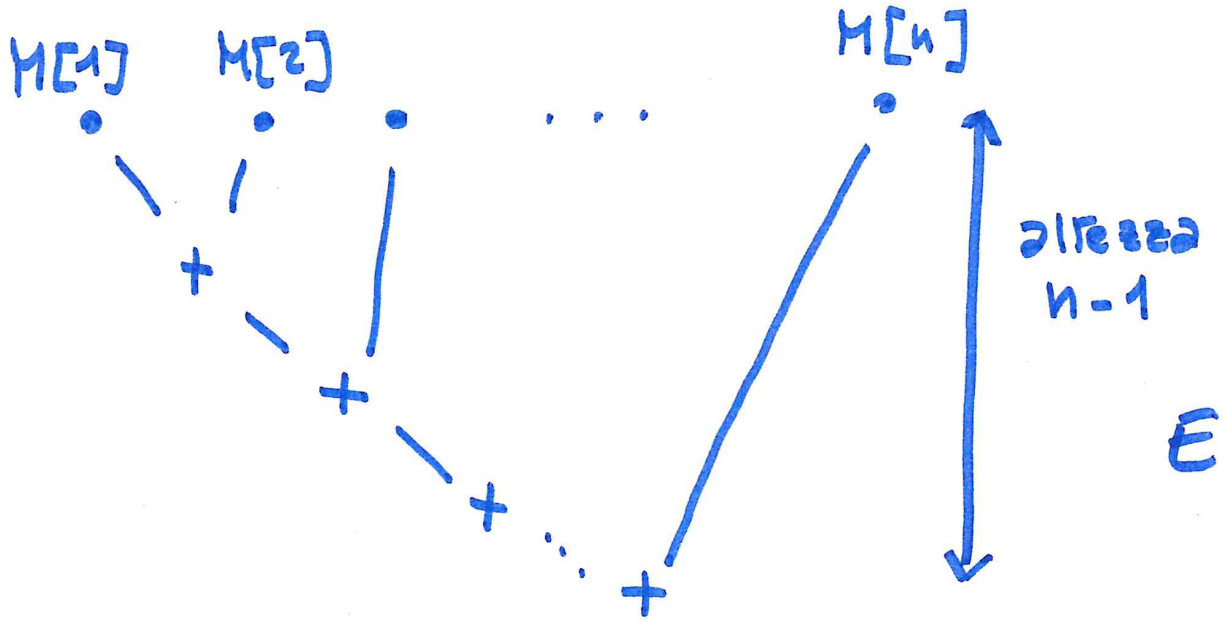
Algoritmo sequenziale

```
for i = 1 to n-1 do
    M[n] = M[n] + M[i]
```

M[n] = M[n] + M[1]
" M[n] + M[2]
⋮
M[n] = M[n] + M[n-1]

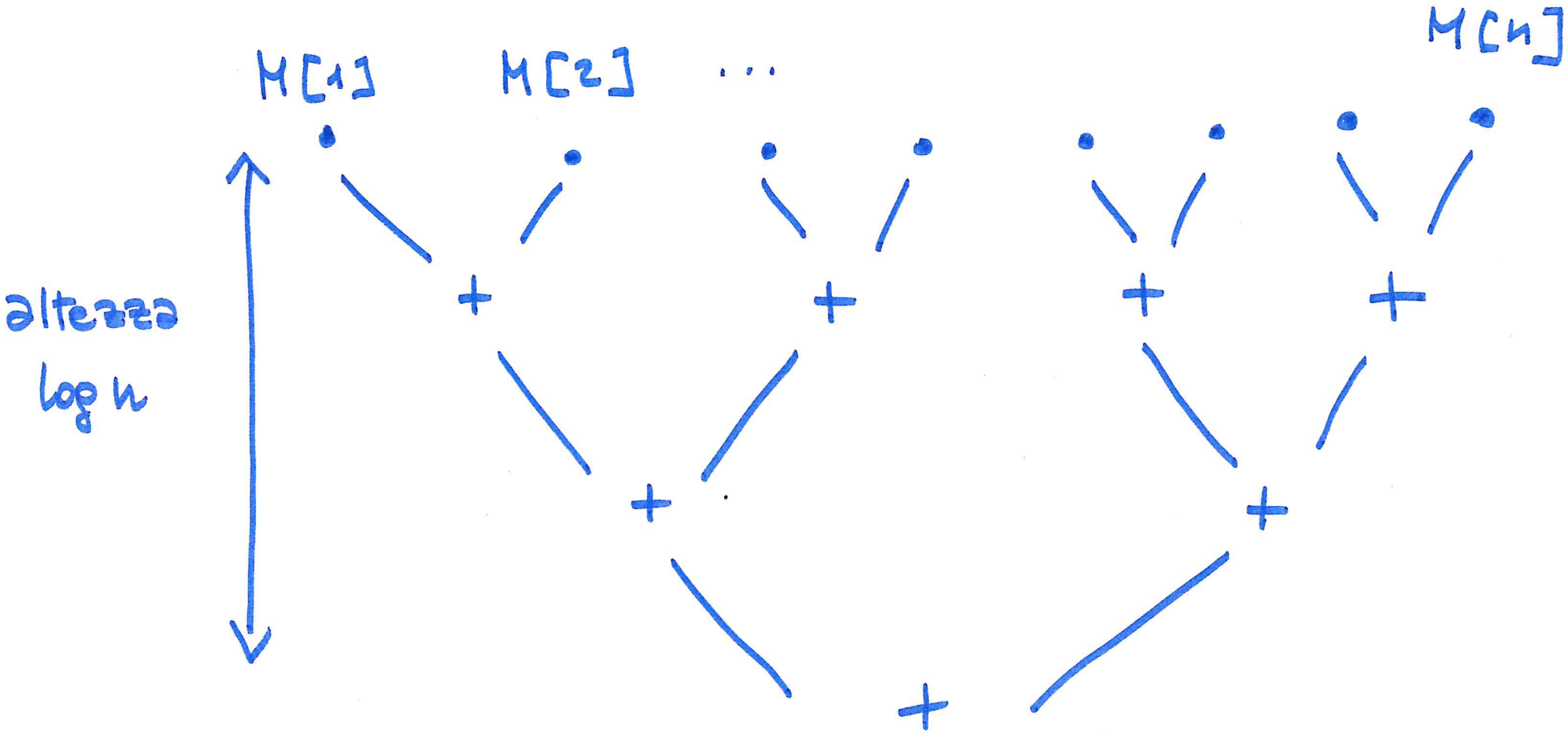
$T(n,1) = n-1$

Idea per parallelizzare: "una somma a processore"



$$E = \frac{n-1}{(n+1)\cdot(n-1)} \to 0$$

In alternativa:

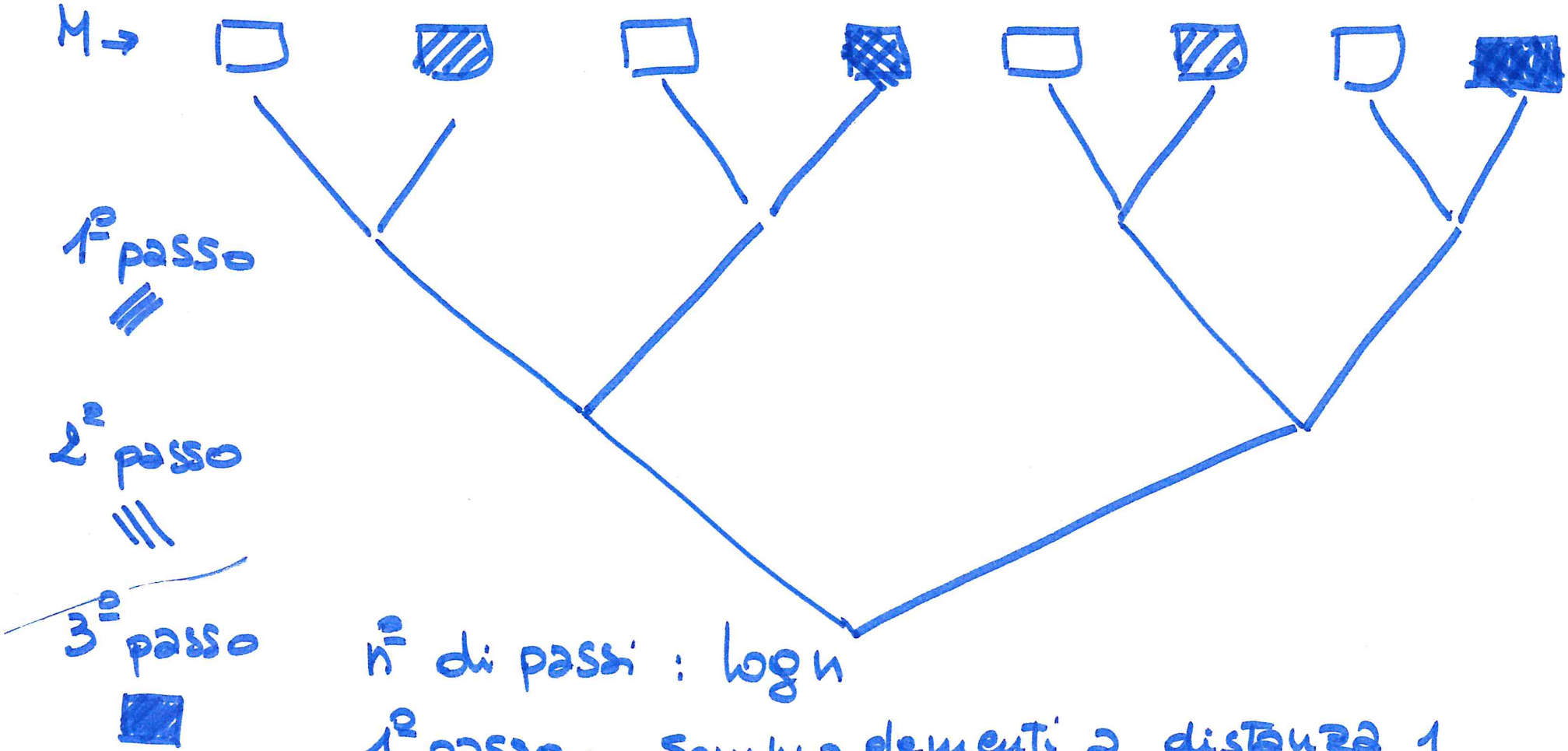


Si può fare perché il + è associativo:

es: ((a+b)+c)+d = (a+b)+(c+d)

Sia $n = 2^t$

Algoritmo EREW



n° di passi : $\log n$

1° passo : somma elementi a distanza 1

2° passo : " " " " 2

3° passo : " " " " 4

Log n° " : ...

Istruzioni:

1° passo: $M[2k] = M[2k] + M[2k-1]$ $\quad 1 \le k \le \frac{n}{2}$

2° passo: $M[4k] = M[4k] + M[4k-2]$ $\quad 1 \le k \le \frac{n}{4}$

3° passo: $M[8k] = M[8k] + M[8k-4]$ $\quad 1 \le k \le \frac{n}{8}$

⋮

$\log n$° passo: $M[n] = M[n] + M[n/2]$ $\quad 1$

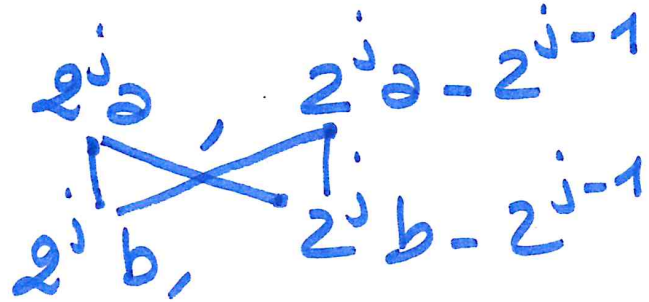
```
for j = 1 To log n
    for k = 1 To n/2^j par do
        M[2^j k] = M[2^j k] + M[2^j k - 2^(j-1)]

return M[n]
```

k = n° processi.

È EREW?

No lettura e scrittura simultanea alla stessa cella di M

Processori a, b processori $a \neq b$



celle coinvolte da a

" da b

Confronti:

$2^j a \neq 2^j b$ si per $a \neq b$

$2^j a \neq 2^j b - 2^{j-1}$ per assurdo: $\Rightarrow \quad 2a = 2b - 1 \Rightarrow a = \frac{2b-1}{2} \notin \mathbb{N}$

...

$\Rightarrow$ ho un algoritmo EREW

Per la correttezza dimostriamo:

$$(*) \quad M[2^j k] = M[2^j k] + \ldots + M[2^j(k-1)+1]$$

Per $j = \log n$, ovviamente $k = 1$, e la (*) è:

$$M[n] = M[n] + \ldots + M[1]$$

Dimostriamo (*) per induzione:

CASO BASE: $j = 1$ e $1 \leq k \leq n/2$

l'istruzione dell'algorit. $\rightarrow$ $M[2k] = M[2k] + M[2k-1]$

quindi vale (*)