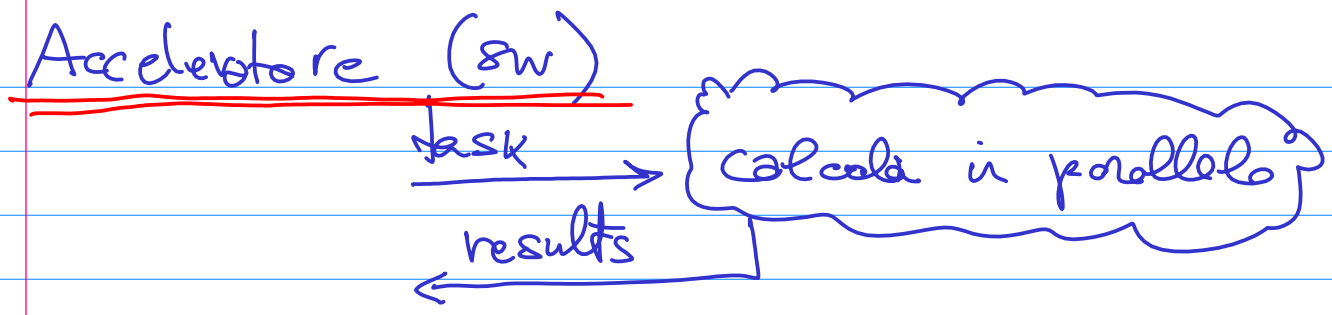
Acceleratore (sw)



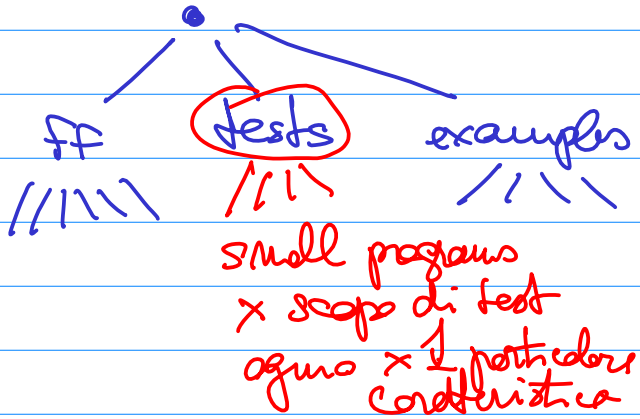
Programmatore applicativo

1) identificare i "task"
   - 1.1 delegare esecuzione dei task all'acceleratore
   - 1.2. attendere la ricezione dei risultati

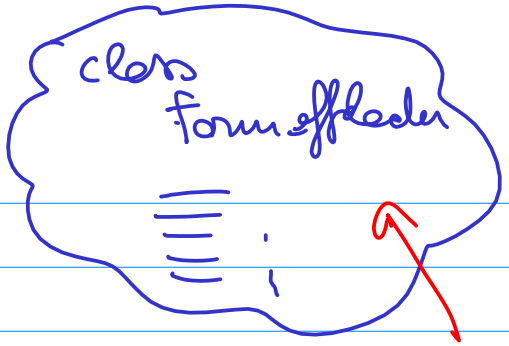
→ Accelerator FastFlow

1) "System programmer"
   - 1.1. capire che forma/pattern serve implementare un esecutore di task secondo quel pattern
   - 1.2.
2) "Application programmer"
   - 2.1 identificare i task iniziali dell'acc e attendere i risultati

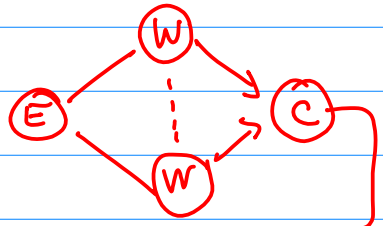
download FastFlow



int main ( ) {



farm-offloader acc(in, out);



offload

load_result

APPL

task (img)

↓ fs

sharpen

↓ balance colors ← data parallel

αF_c

acc = farm (f_s ; f_c)

acc = pipe (f_s ; map (f_c))

SkeTo (Tokio)

-std=c++0x

for (i : x) { f... }

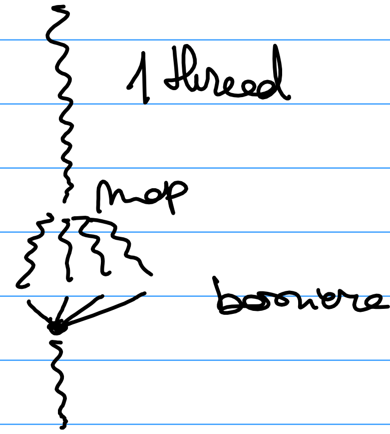
Chiamate di libreria
↳ pattern data parallel
↳ da usare in ambito sequenziale

#include

x[ ]

F

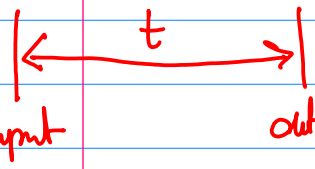
skelo::map(x);



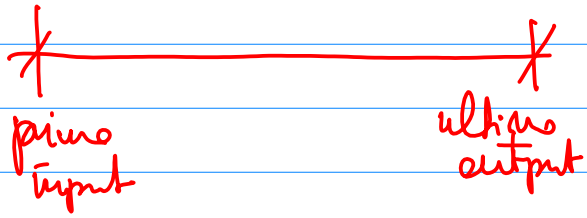
Modelli di performance

Misure base

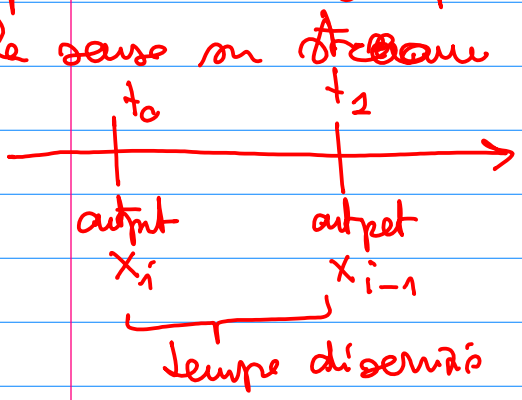
Latenza (computazione) L (sec)



Tempo di completamento T_c (sec)
ha senso su stream (collezioni)



Tempo di servizio (computazione) T_s (sec)
ha senso su stream



Throughput #/sec

Misure derivate

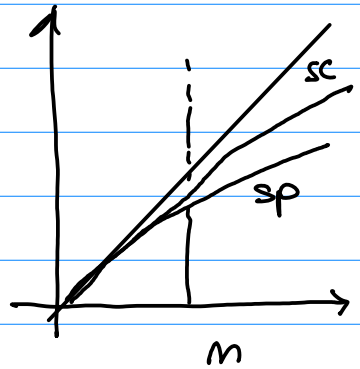
Speedup ← relative al grado di parallelismo (n) → **Scalabilità** **Efficienza**

$$Sp(n) = \frac{T_{seq}}{T(n)}$$

è il migliore sequenziale disponibile

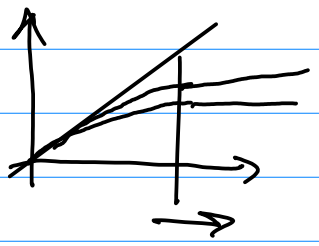
$$Sc(n) = \frac{T(1)}{T(n)}$$

è il tempo col programma parallelo con $n = 1$



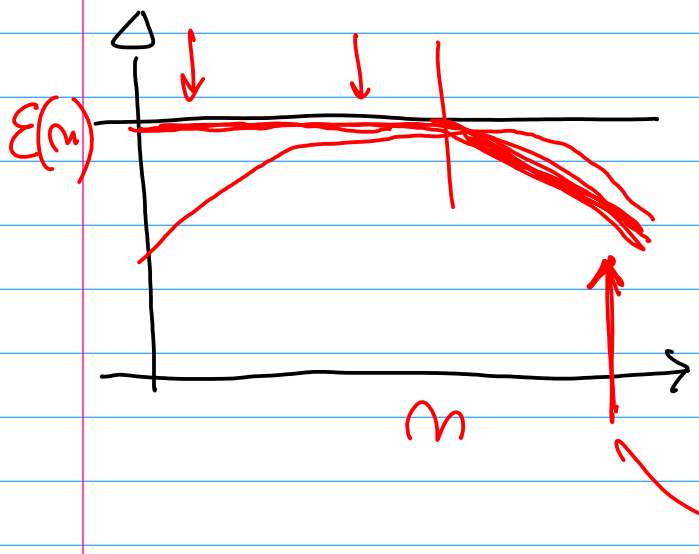
quanto guadagno rispetto al sequenziale

come applicazione si adatta al grado di parallelismo



Efficacité $\varepsilon(n) = \frac{T_{id}}{T(n)} = \frac{T_{seq}}{n\,T(n)} =$

$\varepsilon(n) = \frac{sp(n)}{n}$ $\quad T_{id}(n) = \frac{T_{seq}}{n}$



$\varepsilon(1) = \frac{T_{seq}}{1\,T_{seq}} = 1$