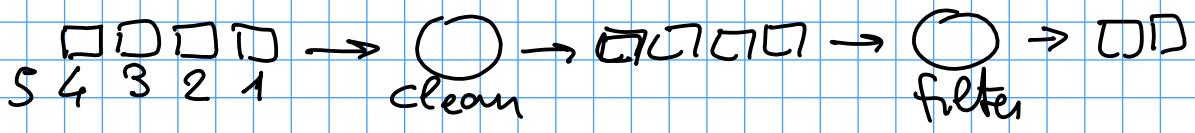
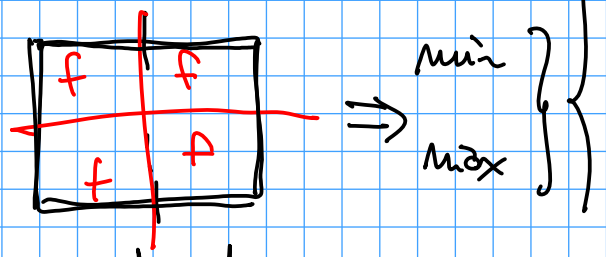
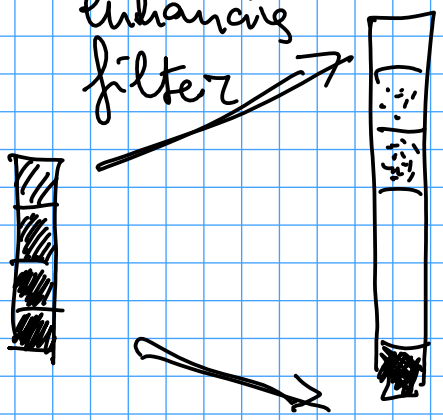


DATA PARALLELISM



contrast enhancing filter

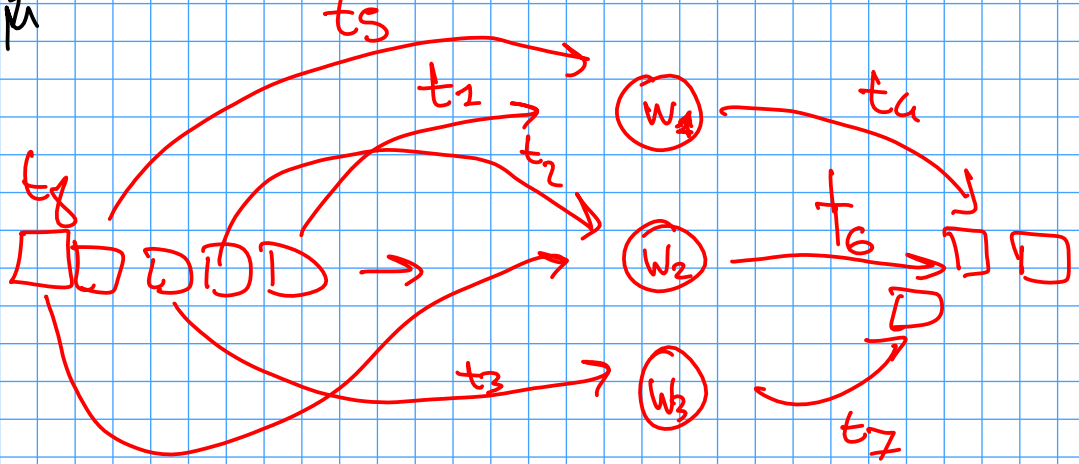


$\min(\forall pixels)$

$\{\forall pixels_{even}\}$ $\{\forall pixels_{odd}\}$

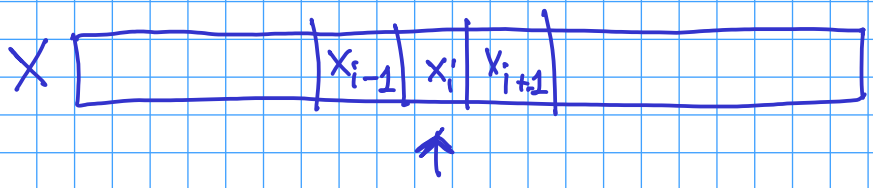
$\min$ $\min$

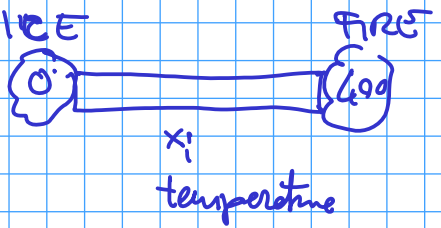
$\min$



STREAM PARALLELISM

DATA PARALLELISM

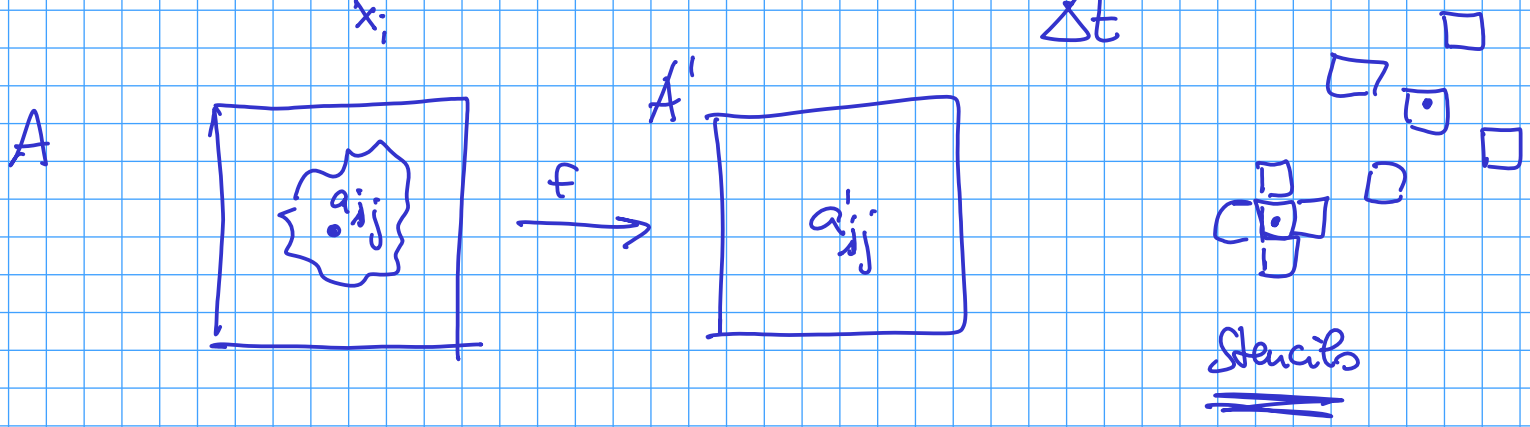




$$f(x_i, x_{i-1}, x_{i+1}) = \frac{x_i + x_{i-1} + x_{i+1}}{3}$$

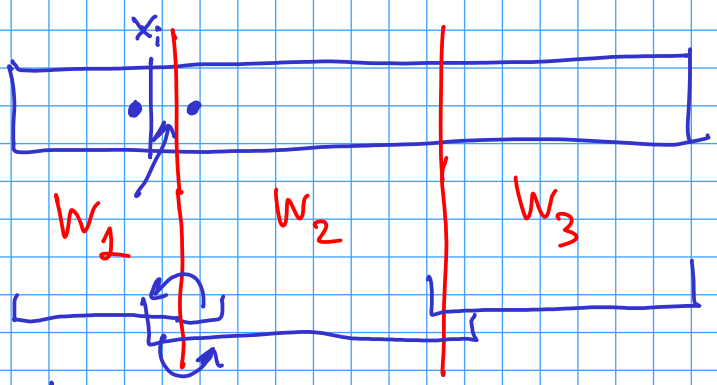
x_i'

Δt



Stencils

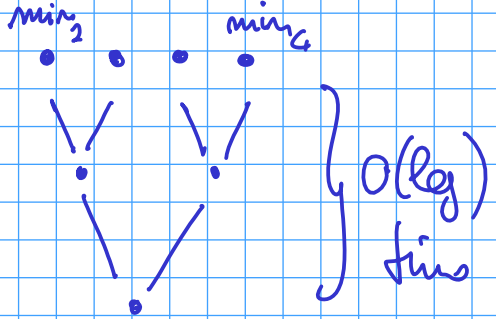
DATA PARALLEL

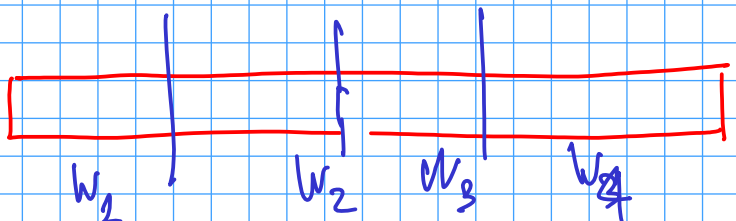


2nd Solution
use "communications"

1st solution
(overlapping portions to the workers)

Min (vector)





assign portions to W_i

$\forall W_i$ computes the min

compute the min of mins

$X \quad \{x_i\}$ $\quad \oplus = \min$

$X_0 \oplus X_1 \oplus \quad \oplus X_{n-1}$

$x_1 \; x_2 \mid x_3 \; x_4 \mid x_5 \; x_6 \mid x_7 \; x_8$

2 tours nous rather than 4

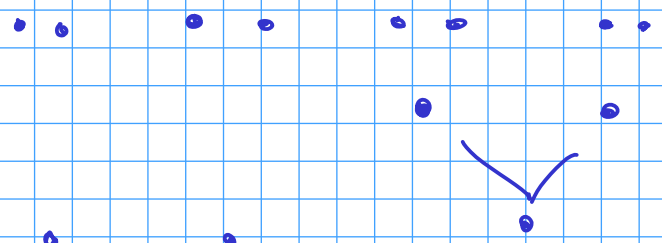
$(x_1 \oplus x_2)\ (x_3 \oplus x_4)\ (x_5 \oplus x_6)\ (x_7 \oplus x_8)$ $\qquad (x_1 \oplus x_2 \oplus x_3 \oplus x_4)$

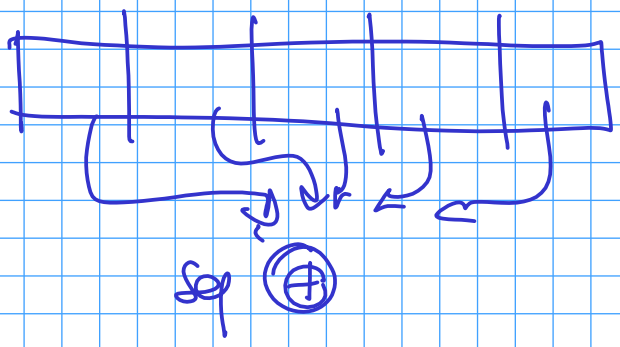
$\oplus$

$((x_1 \oplus x_2) \oplus (x_3 \oplus x_4))\ ((x_5 \oplus x_6) \oplus (x_7 \oplus x_8))$ $\qquad (x_5 \oplus x_6 \oplus x_7 \oplus x_8)$

$\oplus$

$\oplus$ associative commutative operation

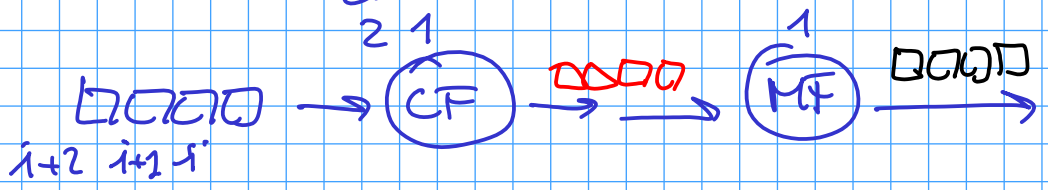




STREAM PARALLELISM

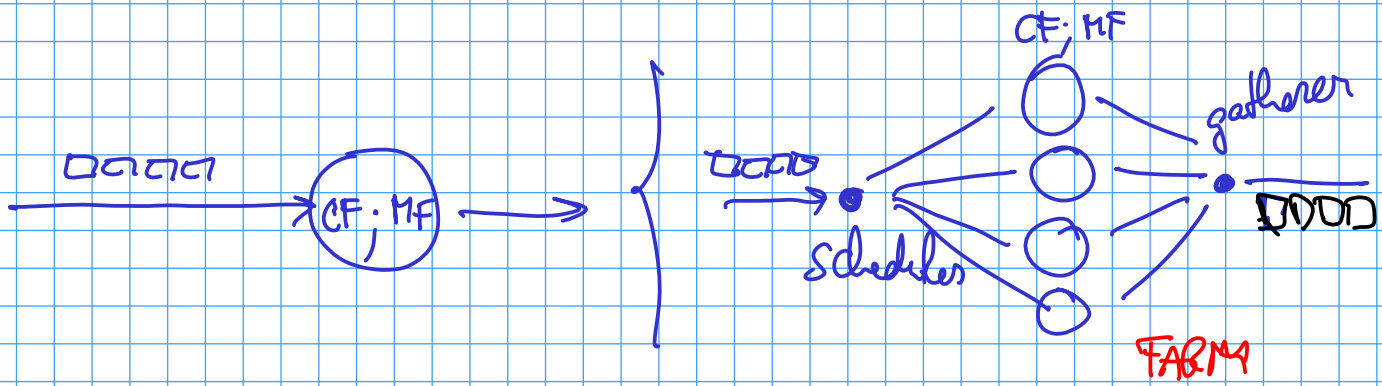
apply (clean filter) then apply (motion filter)

CF MF



PIPELINE

CF will compute $CF(X_i)$ while MF will compute $MF(CF(X_{i-1}))$



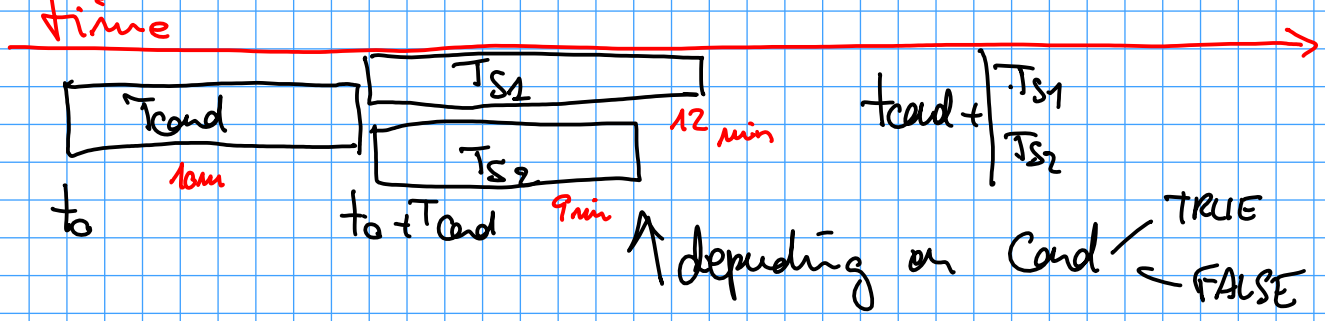
FARM

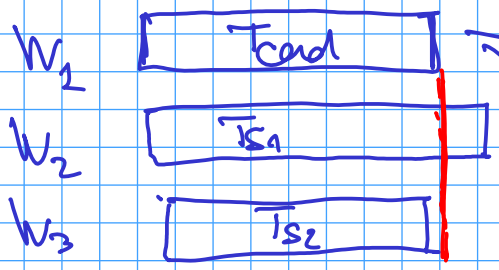
if (cond) then { S_1 } else { S_2 }

T_{cond} comparable with T_{S_1} and T_{S_2} (and long!)

SPECULATIVE

PARALLELISM





TRUE keep results by S_1 & abort S_2

FALSE " " " S_2 & " S_1

$\max \{ T_{cond}, \max \{ T_{S_1}, T_{S_2} \} \}$

CONTROL PARALLELISM

typical control flow statements

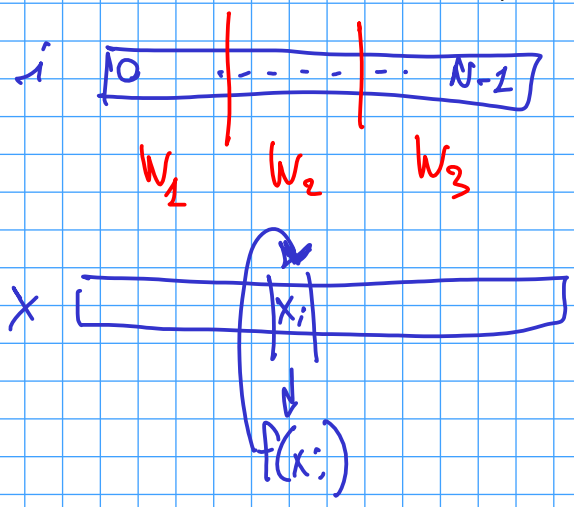
if-then-else ⟹ speculative parallelism

Iterative statements

```
for (i=0; i<N; i++) { S }
```

S: `x[i] = f(x[i]);`

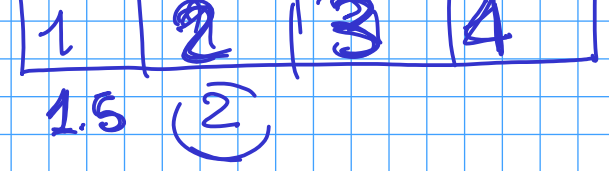
if independent (each other)



S i=k doesn't interfere with the computations of S with i=k

Sample NON INDEPENDENT (iterations)


```
for(  ) {
  x[i] = (x[i] + x[i-1] + x[i+1])/3;
}
```



iteration i depends on i-1

independent loops

```
for(  ) {
  Y[i] = (X(i) + X(... ..)) / 3
}
for(  ) {
  X[i] = Y[i];
}
```

main ( ) { C/C++

pragma omp parallel for

for ( ) {

y[i] = (x[i] + x[i-1] + x[i+1])/3;

}

}

g++ -fopenmp

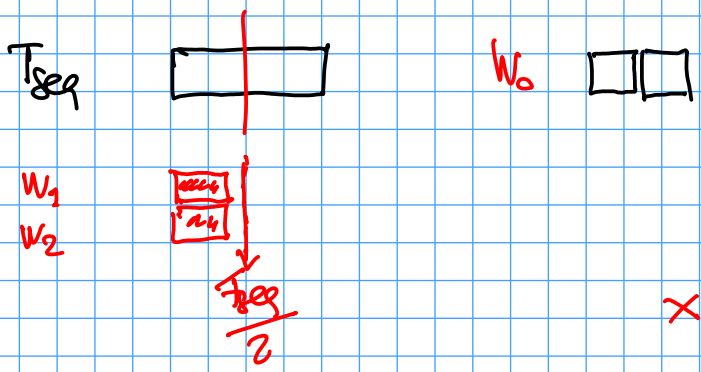
on a multicore with k threads

split into k portions (of the iterations) executed in parallel

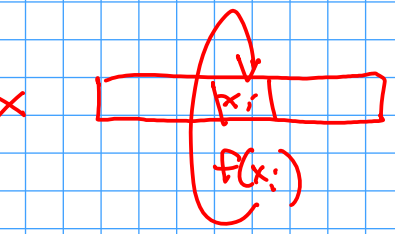
T_{seq} time spent executing a task sequentially

$T_{par}(n)$ $T(n)$ time spent executing the task in parallel having a parallelism degree of n

$$speedup(n) = \frac{T_{seq}}{T(n)}$$ limit $sp(n) \to n$



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

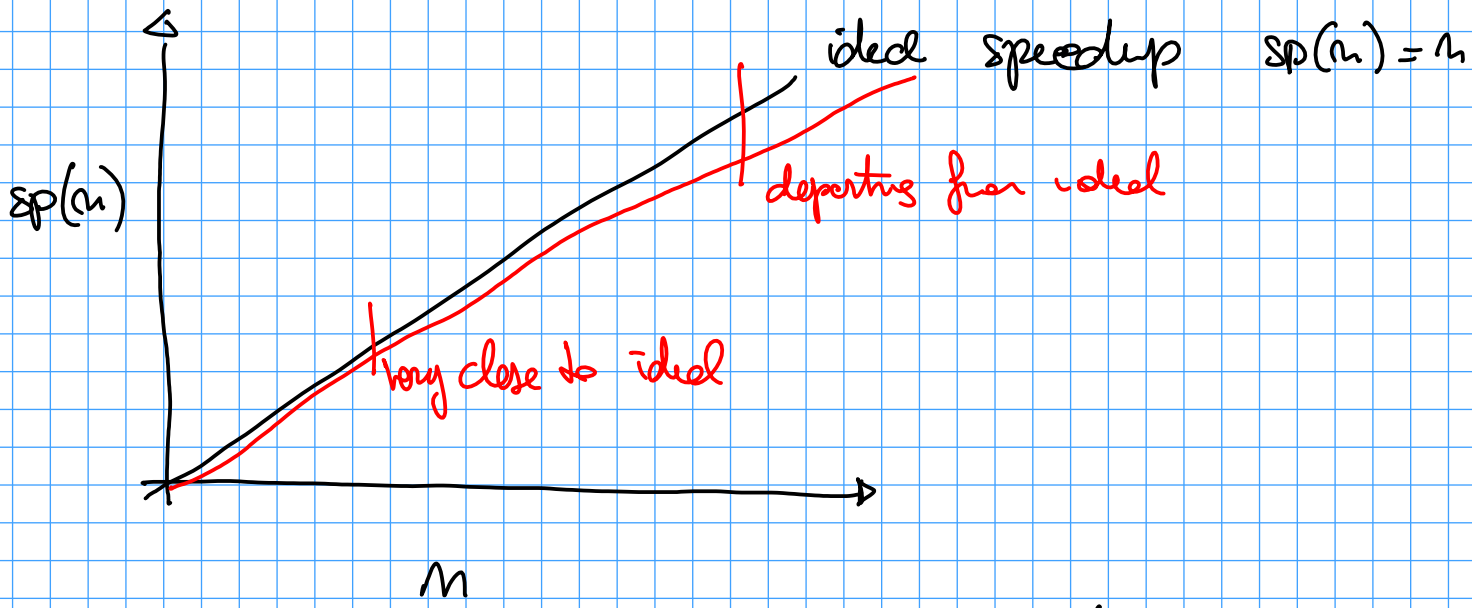


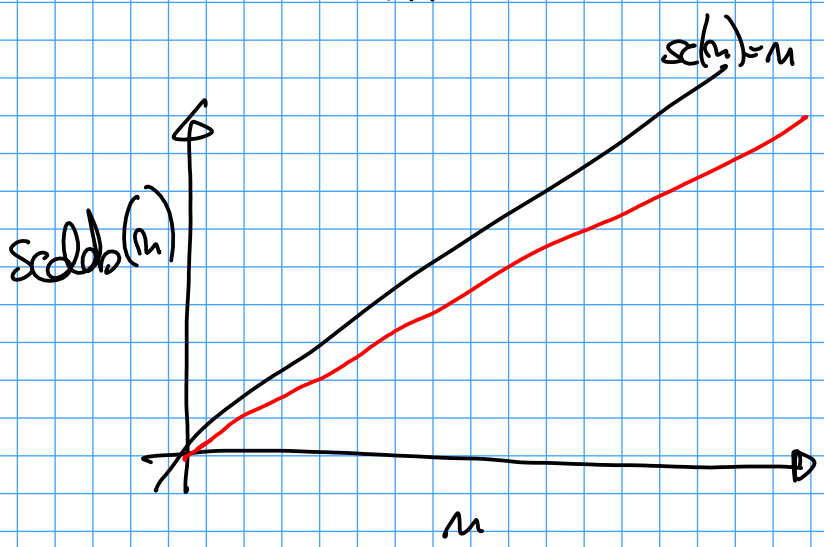
T_{seq} uses the "best" sequential algorithm

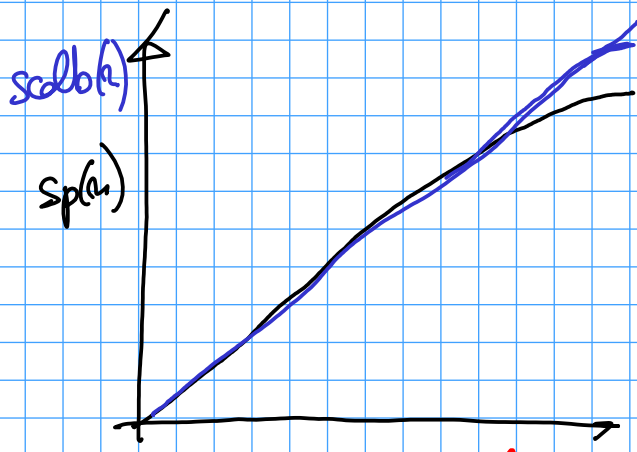
T_{seq} → $\forall i$ substitute x_i by $f(x_i)$

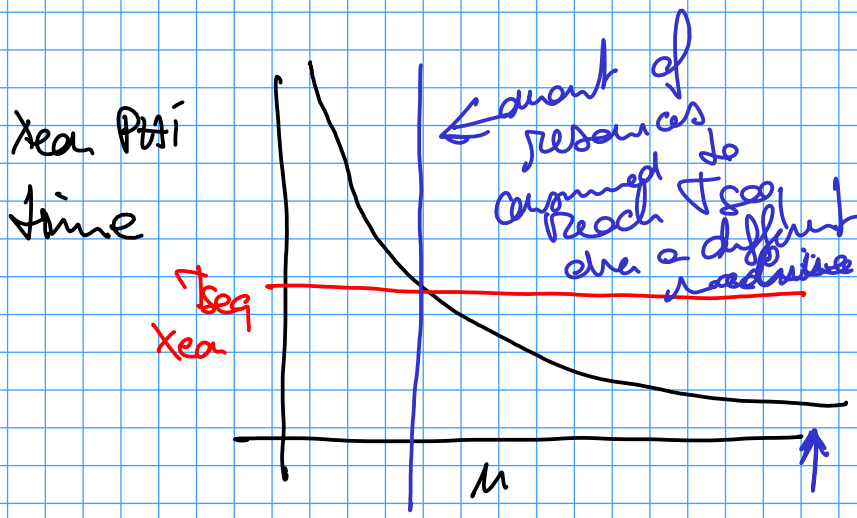
$T(1)$ → Split x in 1 part
Assign to W_1
Get result from W_1

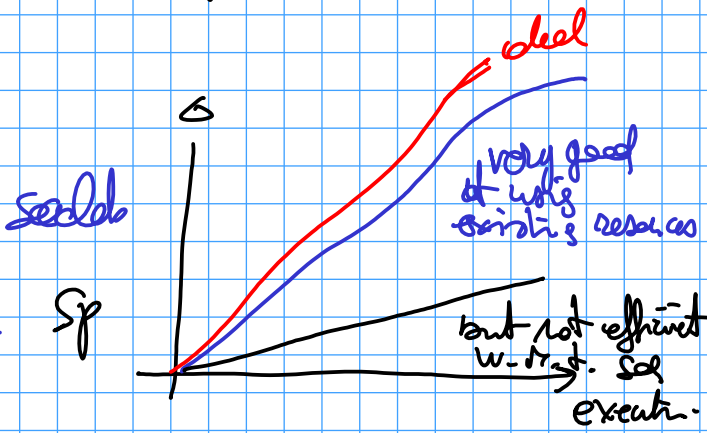
$T(1)$ uses the parallel algorithm with no parallelism











$t_{seq}/n \Rightarrow scalab(n) \approx n$