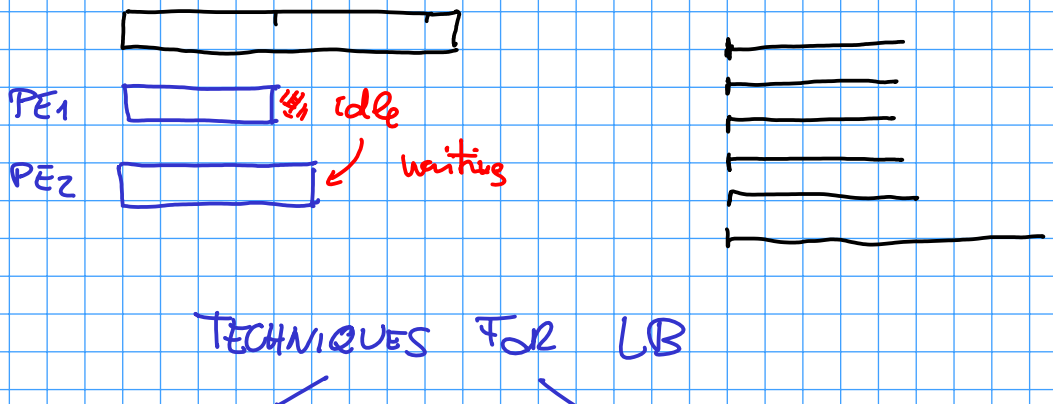
LOAD BALANCING

equal load on all the PE

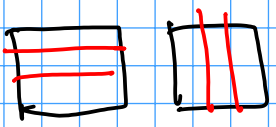


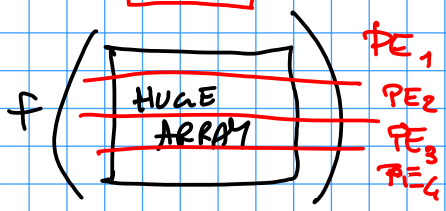
TECHNIQUES FOR LB

- STATIC
- DYNAMIC

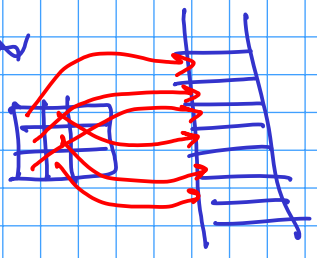
FORTRAN → HPF → BLOCK / CYCLIC

BLOCK of ROWS/COLUMNS

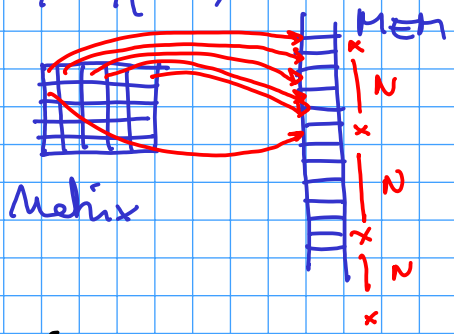




FORTRAN



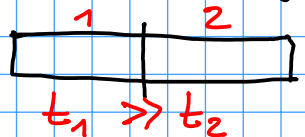
C++/C /(JAVA)



STATIC DISTRIBUTION

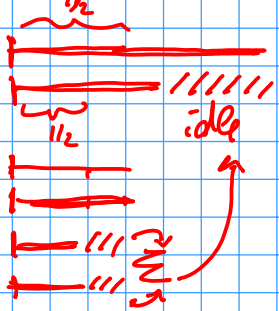
→ IMPACT of GRAIN of DECOMP

→

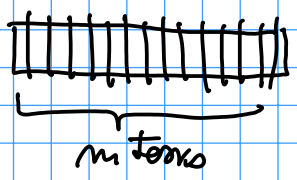


#d1 ≈ #d2

big vector N = 1M

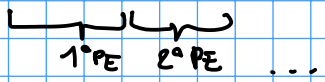


the bigger the idle the smaller the efficiency

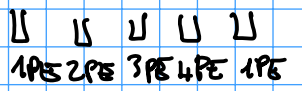


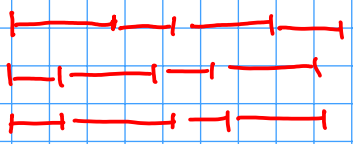
$\#PE = n \ll m$

sched 1 — Block

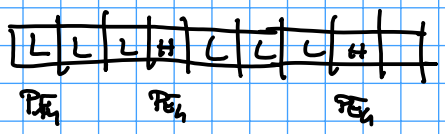


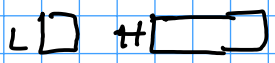
sched 2 — Cyclic





4 PE — 1 heavy each 4 tasks

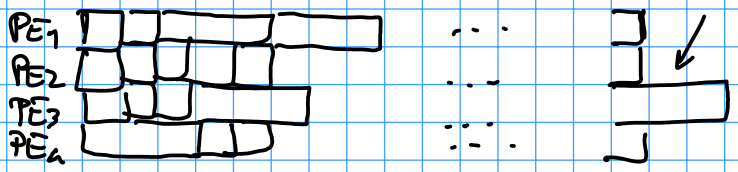




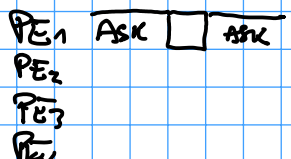
AUTO SCHEDULING STRATEGY (for LB??)

1) generate much more tasks than #PE

2) idle PEs explicitly ask for something to compute

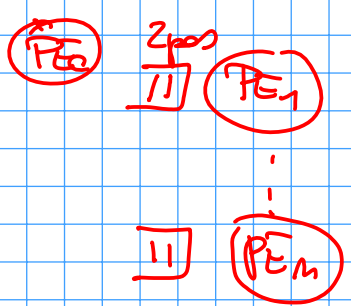


"FULL"



PUSH

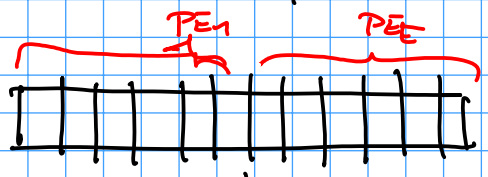
PE_0 pushes tasks into a finite queue in front of the PE_i



if the queue is full it moves to the next PE

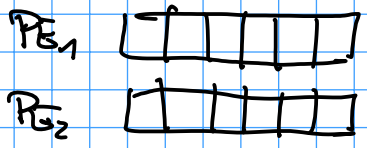
JOB STEALING (DYNAMIC LB TECHNIQUE)

(ao auto scheduling)



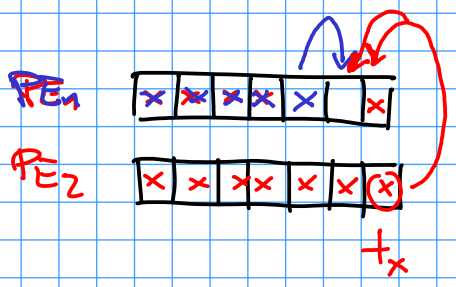
m tasks >> M PE

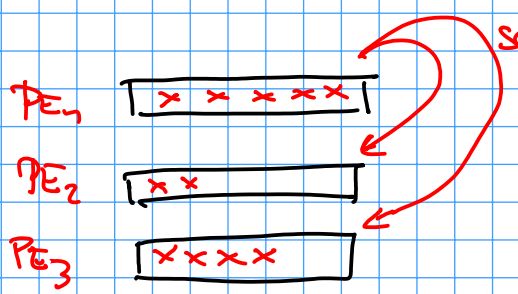
1) Static split of tasks to PEs



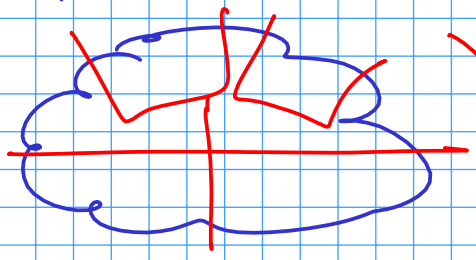
2) ∀ PEi work on the tasks of their local queue

3) When PEi find the local queue empty
it steals a task from a queue (PEj)
most empty





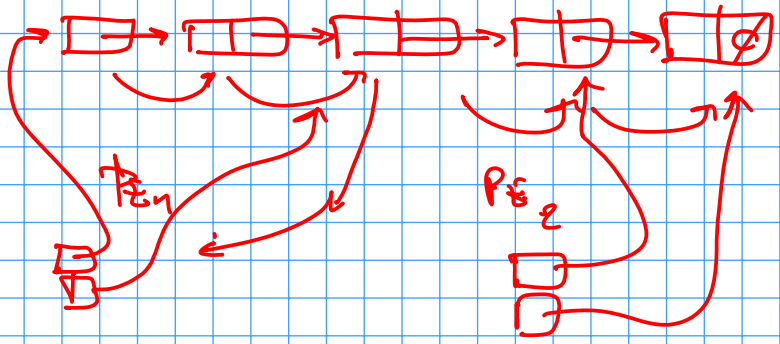
Decomposition strategies



Recursive decomposition strategies

↳ input data "recursively defined"

List = Item | Item :: List



Data decomposition strategies

↳ input data is organized as a collection

&

the collection provides "random access" iterators

PE1 $[0, n/2]$ PE2 $[n/2+1, n]$

Normal decomposition

↳ generate all trees needed to compute the solution of any problem

Speculative decomposition

① $\forall x_i \in$ Vects

if $(c(x_i))$ then $f(x_i) \to x_i$

else $g(x_i) \to x_i$

$t_c \approx t_f \approx t_g$

N item in the array

① $N(t_c + t_f) \to t_c + t_f$ with N PE

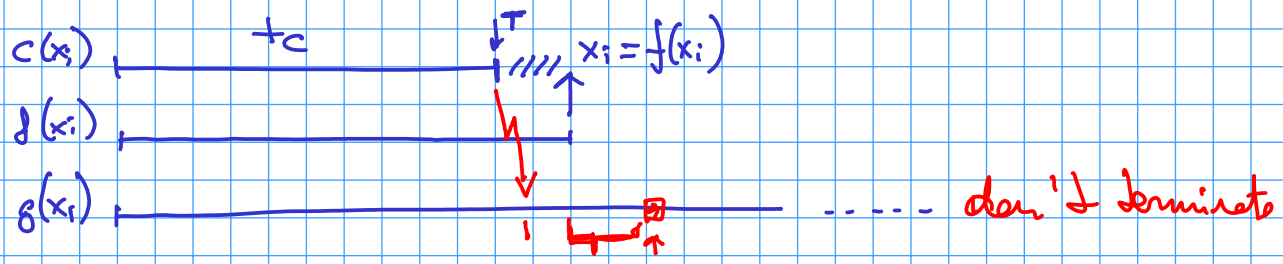
② $N(\max\{t_c, t_f, t_g\}) \to \max(t_f, t_g)$ with 3N PE

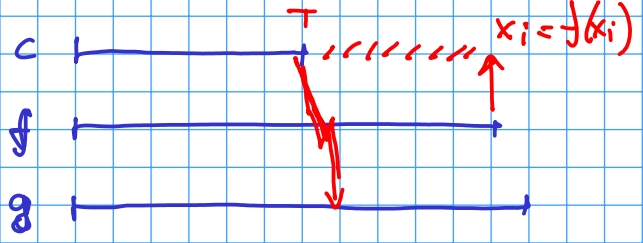
$\forall i$ $c(x_i)$ $f(x_i)$ $g(x_i)$

⟹ extra PEs required / extra computation performed

⟹ works with pure functions (state ?)

⇒ if f or g may not terminate
then need mechanisms to stop f/g





in any case we must ensure there are no side effects to M in g & f before c(x_i) is computed