NFF 214
Partie 1

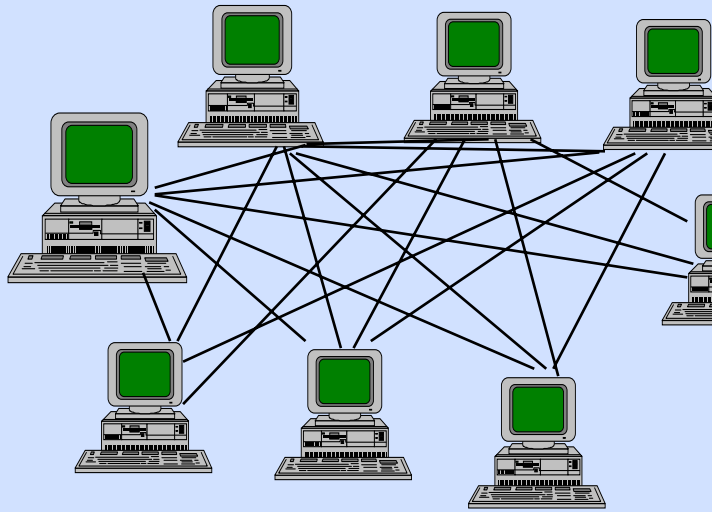
Calculateurs hautes performances
Architectures et programmation

B. Lécussan

bernard.lecussan@cnam.fr

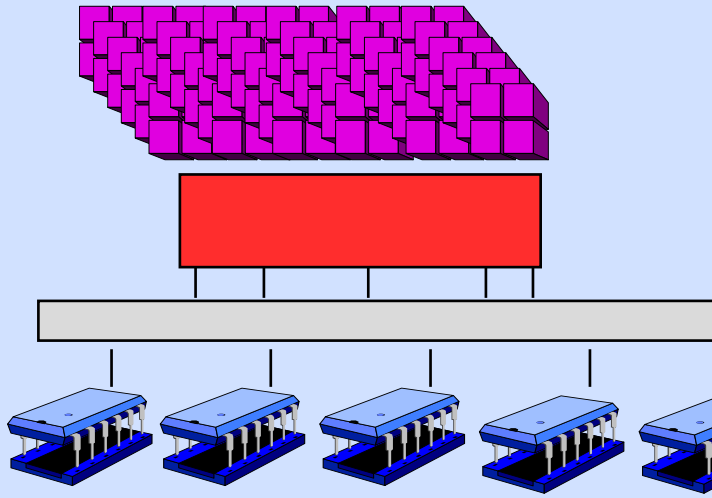
Objectifs du parallélisme

turer un ensemble de
ocesseurs en opérations efficaces.



iter les goulets d'étranglements
e peuvent constituer :

- **La mémoire**
- **Le support de communication**
- **Les E/S**



Plan du cours

- **Les modèles de programmation et d'exécution**
- **Les types de parallélisme**
  - **Vectoriel**
  - **Mémoire Distribuée**
  - **Mémoire Partagée**
- **La programmation des calculateurs parallèles**
- **Les évolutions prévisibles**

ref

Top_lis

IBM

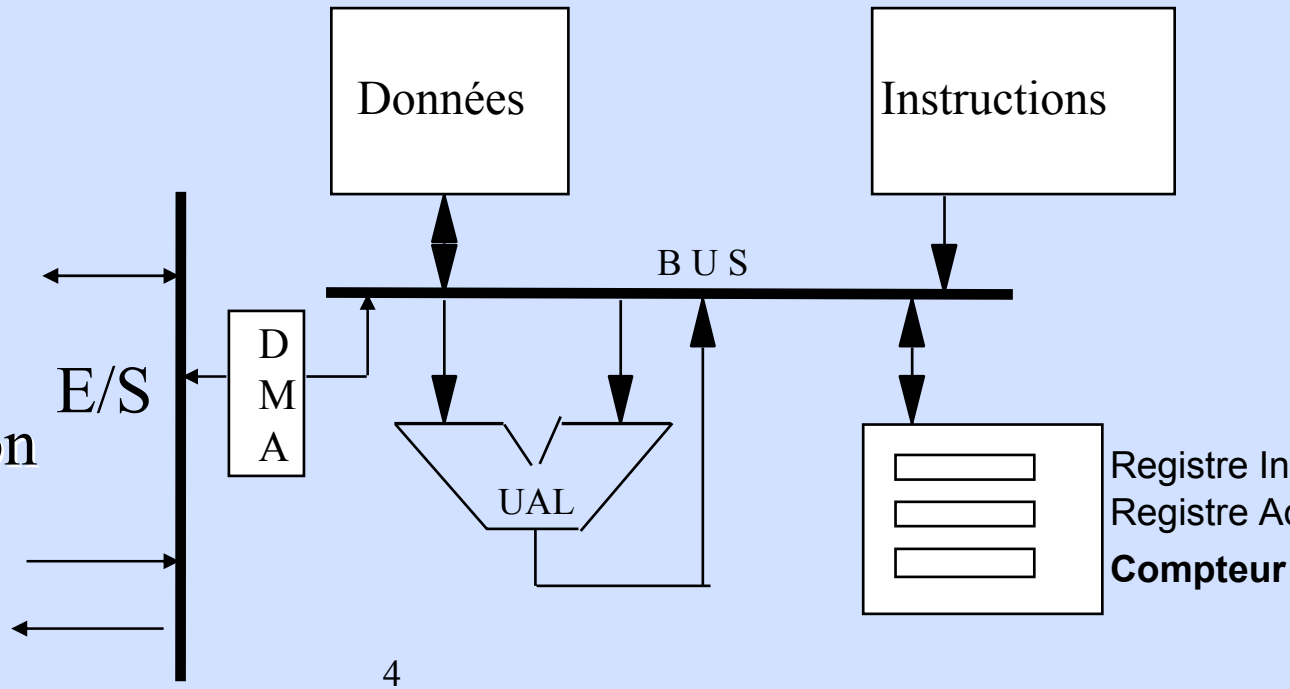
madeW

1) Les modèles de programmation et d'exécution

composantes d'un modèle de programmation

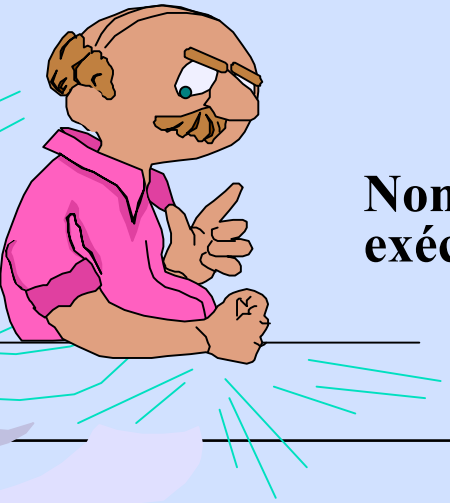
- Les données
- Le programme
- Le contrôle
- Les opérations
- La communication

Les ressources d'un ordinateur



1.1) Performances : a) la tyrannie de la loi d'A

$$T = N * C * Tc$$



- **N** : Nombre d'instructions exécutées
- **C** : Nombre moyen de cycles de base par instruction
- **Tc** : Temps decycle du processeur

$$Speed\_up = \frac{1}{(1 - Part_{amélioration}) + \frac{Part_{amélioration}}{Speed\_up_{amélioration}}}$$

Exemples

- L'augmentation de performances attendue, en améliorant une partie la machine, est limitée par la fraction du temps pendant laquelle cette partie est utilisée.

$T = Ts + Tp/n$

Ts = Temps d'exécution de la partie séqu
Tp = Temps d'exécution de la partie para
n = Nombre de processeurs

Soit f la fraction du temps de la partie séquentielle

Le facteur d'accélération est donné par :

$$S(n)=\frac{Ts}{f*Ts+(1-f)*Ts/n}=\frac{n}{1+(n-1)*f}=\frac{1}{f+\frac{1-f}{n}}$$

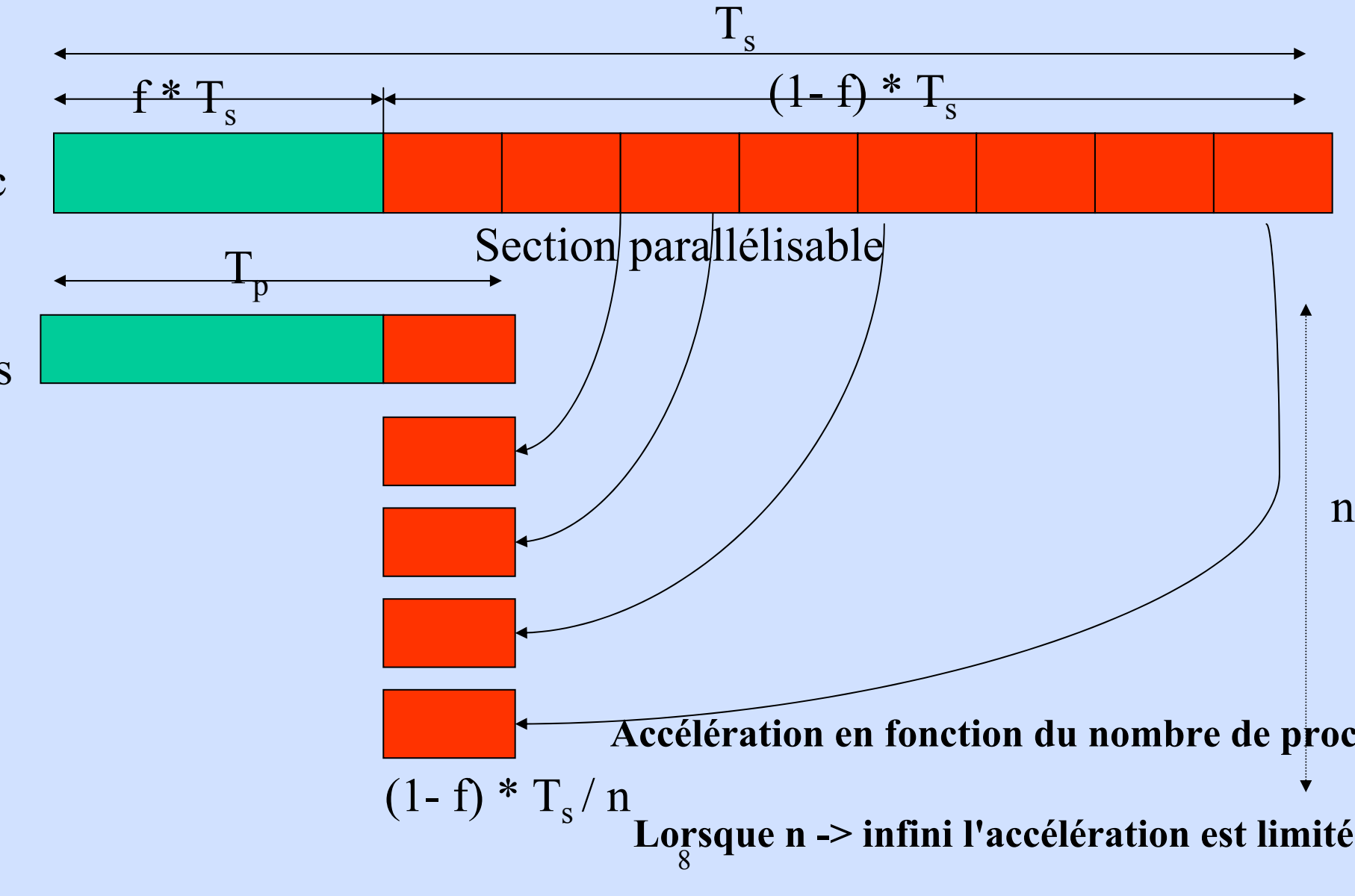
• **Exemple1: 80 : 20**
80% parallèle sur 80 PE
20% séquentiel

$$Speed\_up = \frac{1}{(1-0,80) + \frac{0,80}{80}} = 4,76$$

• **Exemple2: 99 : 1**
99% parallèle sur 100 PE
1% séquentiel

$$Speed\_up = \frac{1}{(1-0,99) + \frac{0,99}{100}} = 50,25$$

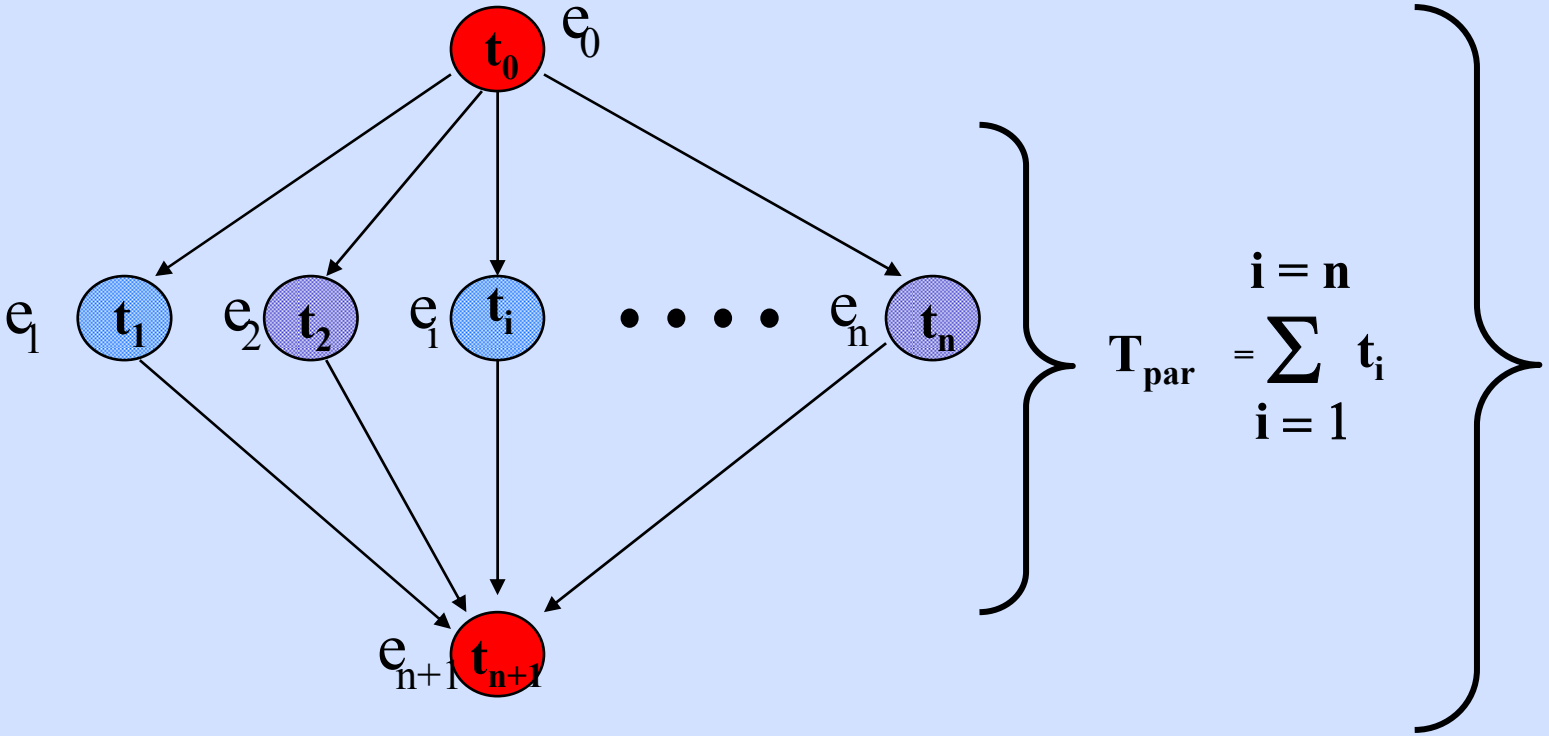
Maximum Speedup - Loi d'Amdahl



Question : Avec seulement 5% du calcul total restant séquentiel, quel est le nombre suffisant de processeurs d'un calculateur parallèle pour atteindre le maximum d'efficacité?

b) Un modèle analytique des performances

oit un graphe de tâches à un seul niveau représentant un travail partic
unions disjointes de tâches s'exécutant sur n processeurs.



Temps de calcul parallèle

$$T_p = t_0 + t_{n+1} + \max t_i$$

t_i : i-th processor computation time, $1 \leq i \leq n$

Efficacité parallèle

$$E = \frac{T_s}{n * T_p}$$

T_s : Best sequential computation time

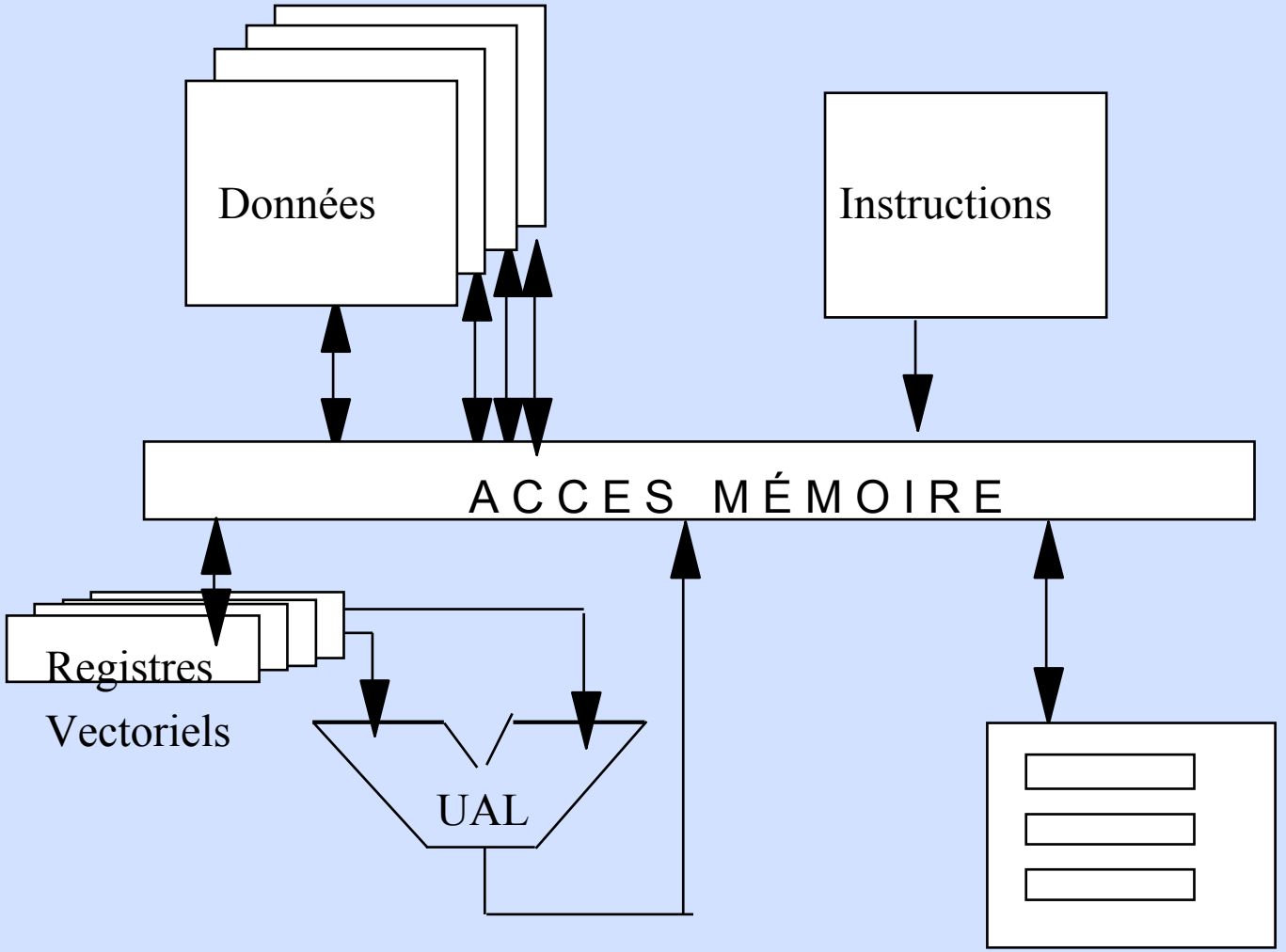
Maximum d'efficacité

$$T_{min} = (t_0 + t_{n+1}) + \frac{T_{par}}{n}$$

$t_0 + t_{n+1}$: Sequential part computation time

T_{par} : Parallel part computation time

1.3) Le Calculateur Vectoriel

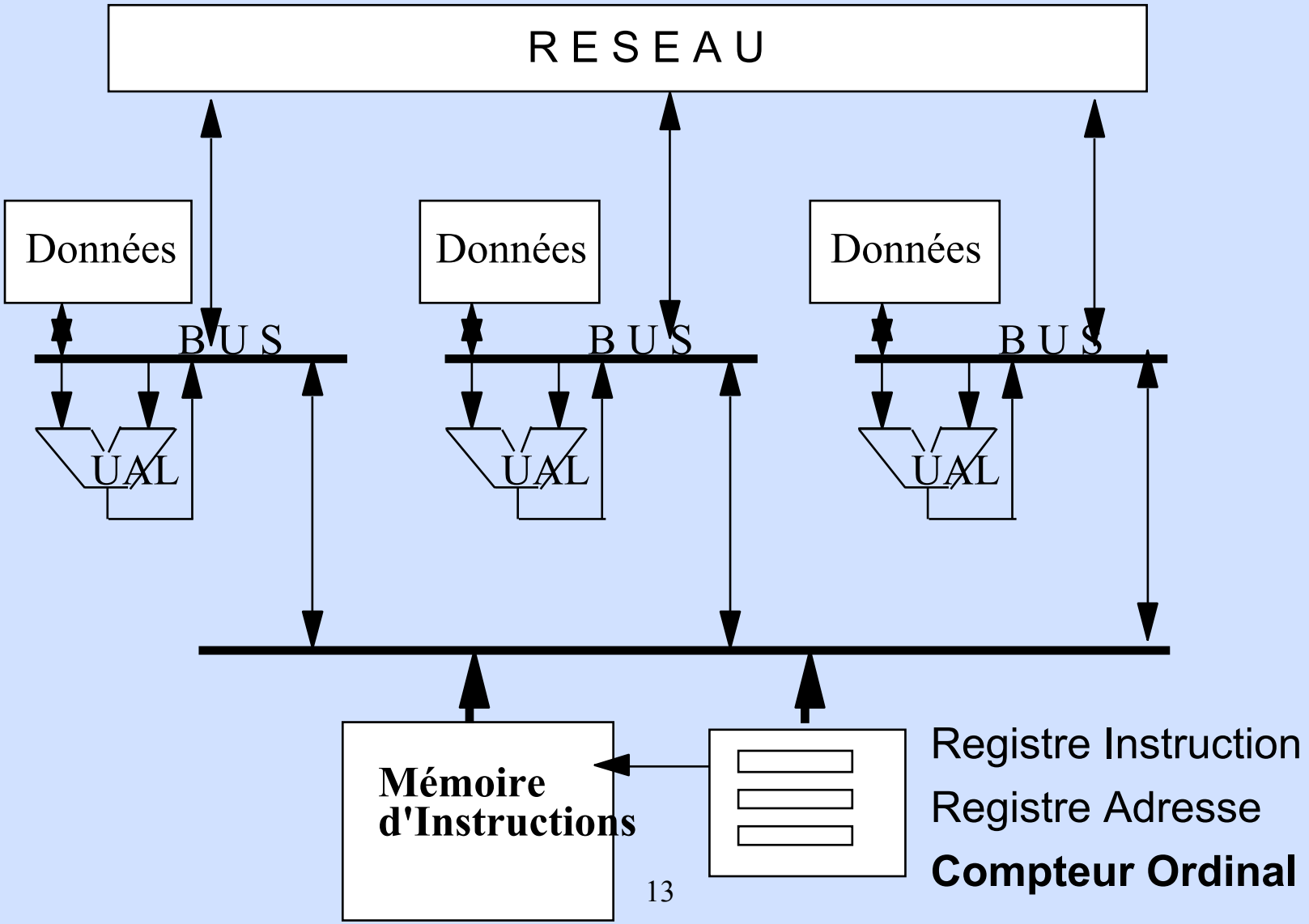


Earth Simula

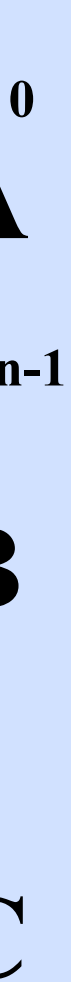
1
2
3

Registre Instruct
Registre Adress
Compteur Ordi

1.4) SIMD Single Instruction Multiple Data

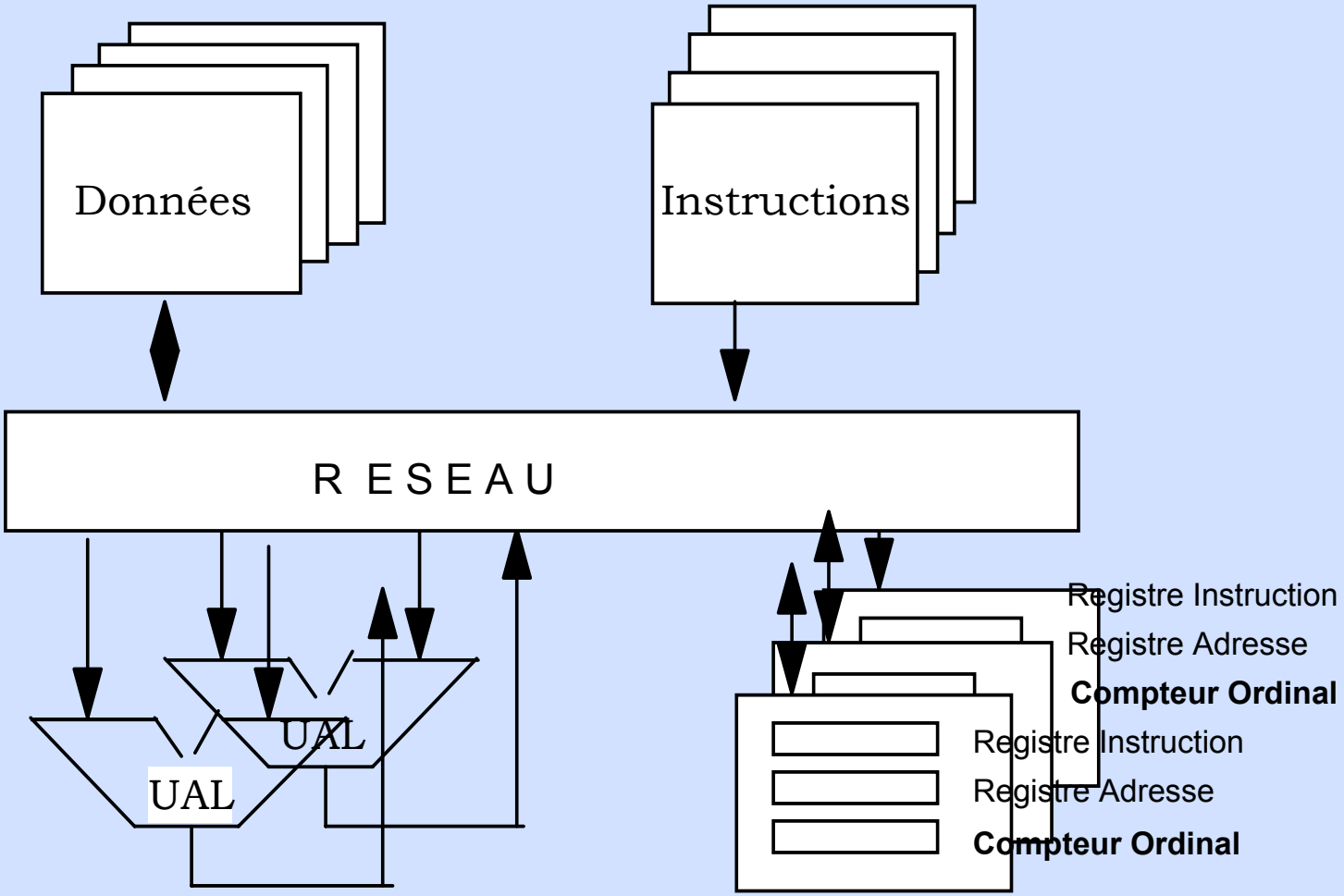


Rangement des données en mémoire

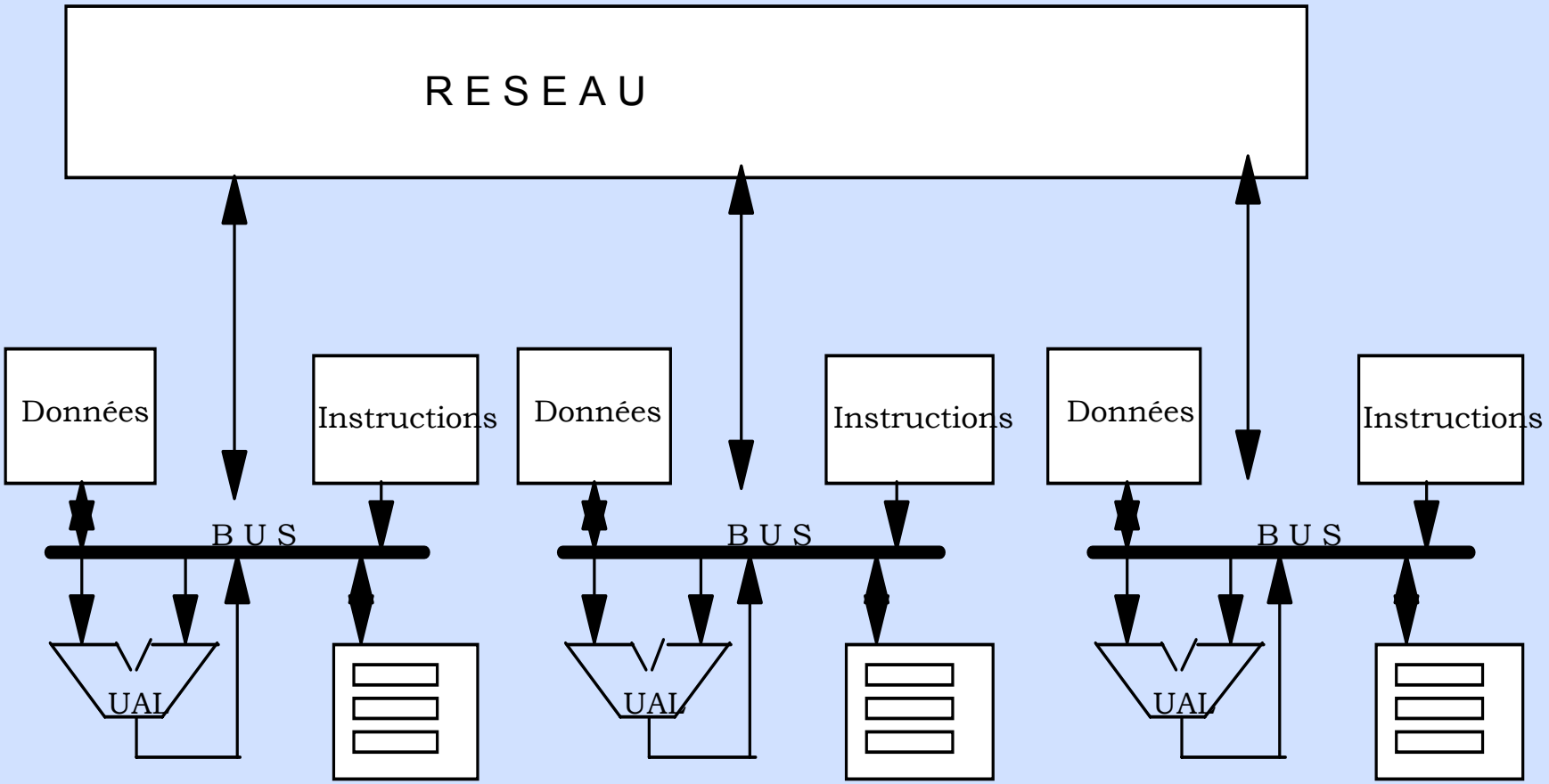


Mémoire 0	Mémoire i	Mémoire n-1
A(0,0)	A(0,i)	A(0,n-1)
A(1,0)	A(1,i)	A(1,n-1)
A(2,0)	A(2,i)	A(2,n-1)
A(n-1,0)	A(n-1,i)	A(n-1,n-1)
B(0,0)	B(0,i)	B(0,n-1)
B(1,0)	B(1,i)	B(1,n-1)
B(2,0)	B(2,i)	B(2,n-1)
B(n-1,0)	B(n-1,i)	B(n-1,n-1)
C(0,0)	C(0,i)	C(0,n-1)
C(1,0)	C(1,i)	C(1,n-1)
C(2,0)	C(2,i)	C(2,n-1)
C(n-1,0)	C(n-1,i)	C(n-1,n-1)

1.5) Le multiprocesseur à mémoire partag



1.6) Le multiprocesseur à mémoire distrib



1.7) Les trois paramètres des calculateur parallèles

- la latence pour accéder à une donnée
  - **latence de la hiérarchie mémoire (caches, mémoire centrale, disque)**
  - **latence réseau**
- la vitesse de transfert des données d'une tâche s'exécutant dans un processeur vers une tâche d'un autre processeur
- la performance scalaire d'un processeur : le nombre d'instructions par cycle (IPC)

Latence versus débit

Avion	Paris/DC	Vitesse	Passagers	Débit (p*km/h)
Boeing 747	6,5 heures	976 km/h	470	458 720
Concorde	3 heures	2160 km/h	132	285 120

Exercice

:

calculateur parallèle dont les accès mémoires distants coûtent 2.000ns
processeur a une horloge à 2 Ghz et exécute une instruction par cycle C
lement 0,5 % des instructions engendrent un accès distant

Calculer le CPI d'une telle machine :

- **CPI = 1 + 0,5 % * 2000/0,5 = 21**

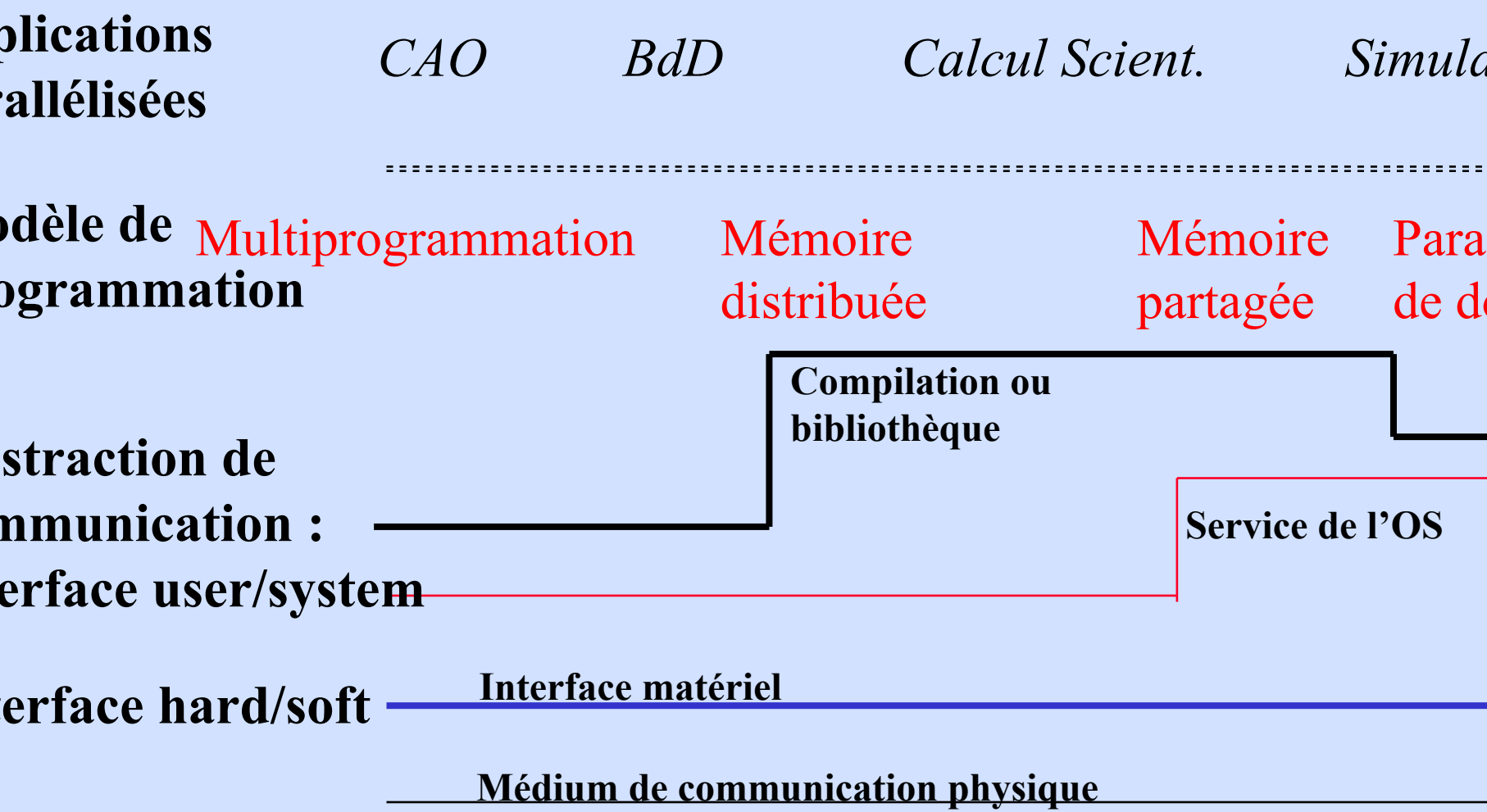
1.8) Une vision logicielle : les modèles de programmation parallèle

Un programme parallèle est composé de un ou plusieurs flots d 'exécution opérant sur un ensemble de données.

Un modèle de programmation parallèle doit spécifier :

- Les données accédées par les flots
  - accès à des variables partagées
  - accès à des variables distantes
- Les opérations qui peuvent être effectuées sur ces données
  - R/W atomique sur des données partagées
  - send ou receive (N° de process // adresse locale)
- L'ordre d'exécution
  - Exclusion mutuelle sur la modification de certaines donnée
  - Les événements comme moyen de synchronisation

1.9) Les niveaux d'abstraction



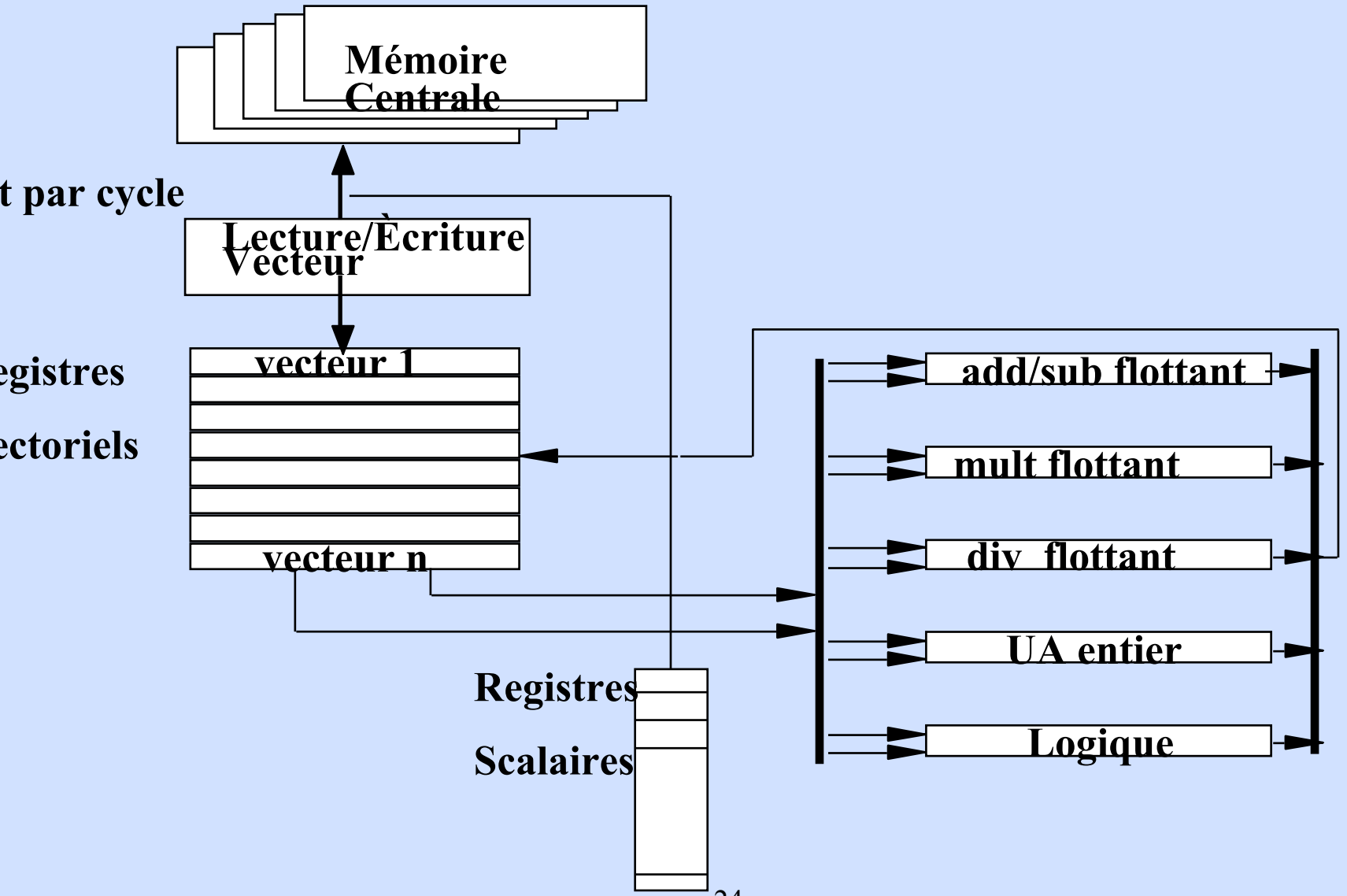
2) Les types de parallélisme

2.1) Le parallélisme de données

- **Par des calculateurs vectoriels**

- **Par des multiprocesseurs**

2.1.1) Architecture du calculateur vectorie



Considérons l'opération vectorielle classique suivante :

Y = a * X + Y avec X et Y vecteurs et a scalaire

Soit des vecteurs de longueurs L = 64, taille des registres vectoriels.

Programme scalaire

```
DDI    R4, Rx, #512    ; dernière adresse à lire
D      F0, a
D      F2, 0(Rx)       ; chargement de X(i)
ULTD   F2, F0, F2      ; calcul a x X(i)
D      F4, 0(Ry)       ; chargement de Y(i)
DDD    F4, F2, F4      ; a x X(i) + Y(i)
D      F4, 0(Ry)       ; rangement à Y(i)
DDI    Rx, Rx, #8      ; suivant de X
DDI    Ry, Ry, #8      ; suivant de Y
UB     R20, R4, Rx     ; calcul de la limite
NZ     R20, loop
```

Exécution de 578 instructions

Programme vectoriel

```
LD      F0, a          ; chargement du sca
LV      V1, Rx         ; chargement du vec
MULTSV  V2, F0,V1      ; multiplication scal
LV      V3, Ry         ; chargement du vec
ADDV    V4, V2, V3     ; addition vectorielle
SV      Ry, V4         ; rangement du résu
```

Exécution de 6 instructions

2.1.3) Gestion de l'allocation des données mémoire

omposantes des vecteurs
nt pas forcément consécutives
moire d'ou pb pour
nter
gistres vectoriels.

Ex. Produit de matrices de taille 100

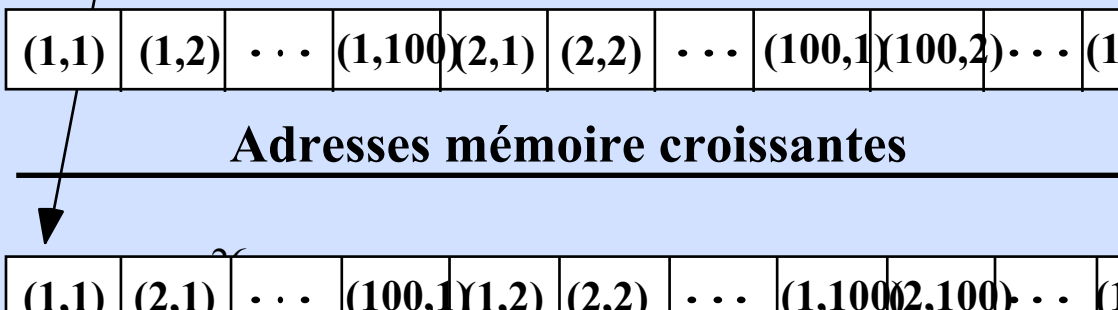
```
      do  10  i = 1, 100
          do  10  j = 1, 100
              A(i,j) = 0.0
              do  10  k = 1, 100
10                A (i,j) = A(i,j) + B(i,k) x C(k,j
```

k) et B (i, k+1) sont distants de 100.
j) et C(k+1,j) sont distants de 1.

Rangement en mémoire :
- par lignes (tous les langages sauf FORTRAN)
- par colonnes (FORTRAN)

egistre *Stride* contient la
ur de l'intervalle

de 800



2.1.4) Gestion des vecteurs de longueur quelconque

it l'opération :

valeur connue à l'exécution

```
do   10   i = 1,n
Y(i) = a x X(i) + Y(i)
```

- **Soit VLR un registre spécialisé contenant la valeur de la longueur du vec**
- **Valeur Max de VLR = Taille des registres vectoriels (MVL)**

```
      low = 1
      VL = (n mod MVL)                    ; taille du premier sous vecteur
      do   1  j = 0, ( n / MVL)           ; boucle vectorielle externe
         do   10  i = low, low + VL -1
            Y(i) = a x X(i) + Y(i)        ; boucle interne sur chaque sous-vecteu
10    continue
      low = low + VL                      ; passage au vecteur suivant
      VL = MVL                            ; taille maximale
1     continue
```

2.1.5) Exemples de vectorisation

ramme scalaire :

t = 0.0

10 i=1, 64

*dot = dot + A(i) * B(i)*

64 multiplications et additions

séquentielles

Vectorisation :

do 10 i = 1, 64

*10 dot (i) = A(i) * B(i)*

do 20 i = 2, 64

20 dot (1) = dot (1) + dot (

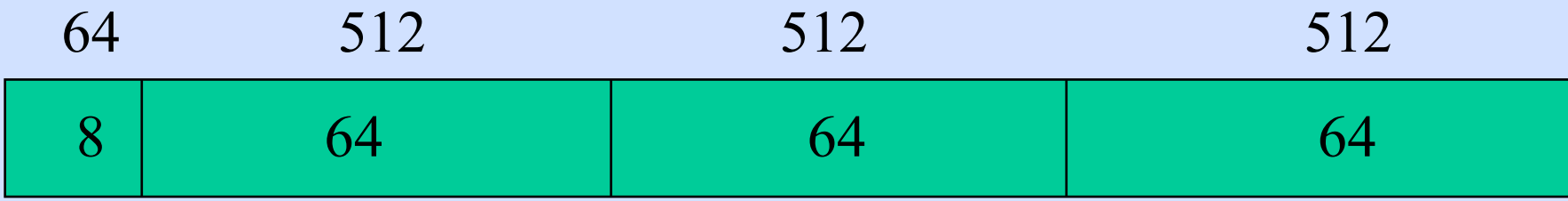
64 multiplications parallèles

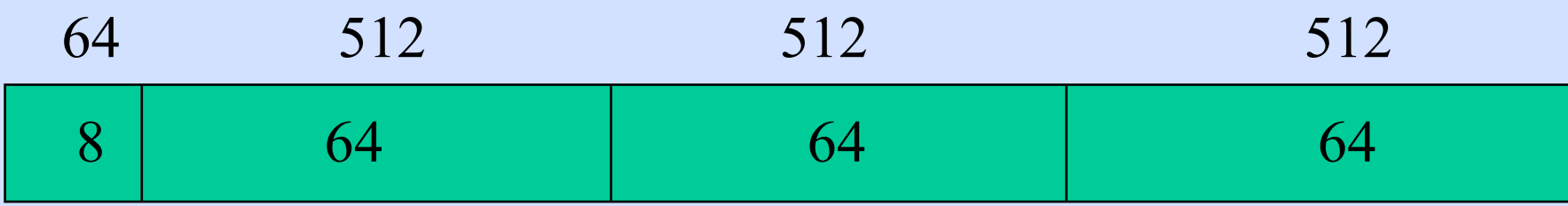
Sommation séquentielle de
64 sous-produits

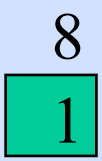
Ex: A = B*s A et B vecteurs de longueur 200 élé

s scalaire

Vi registres vectoriels de 64 flottants de 8







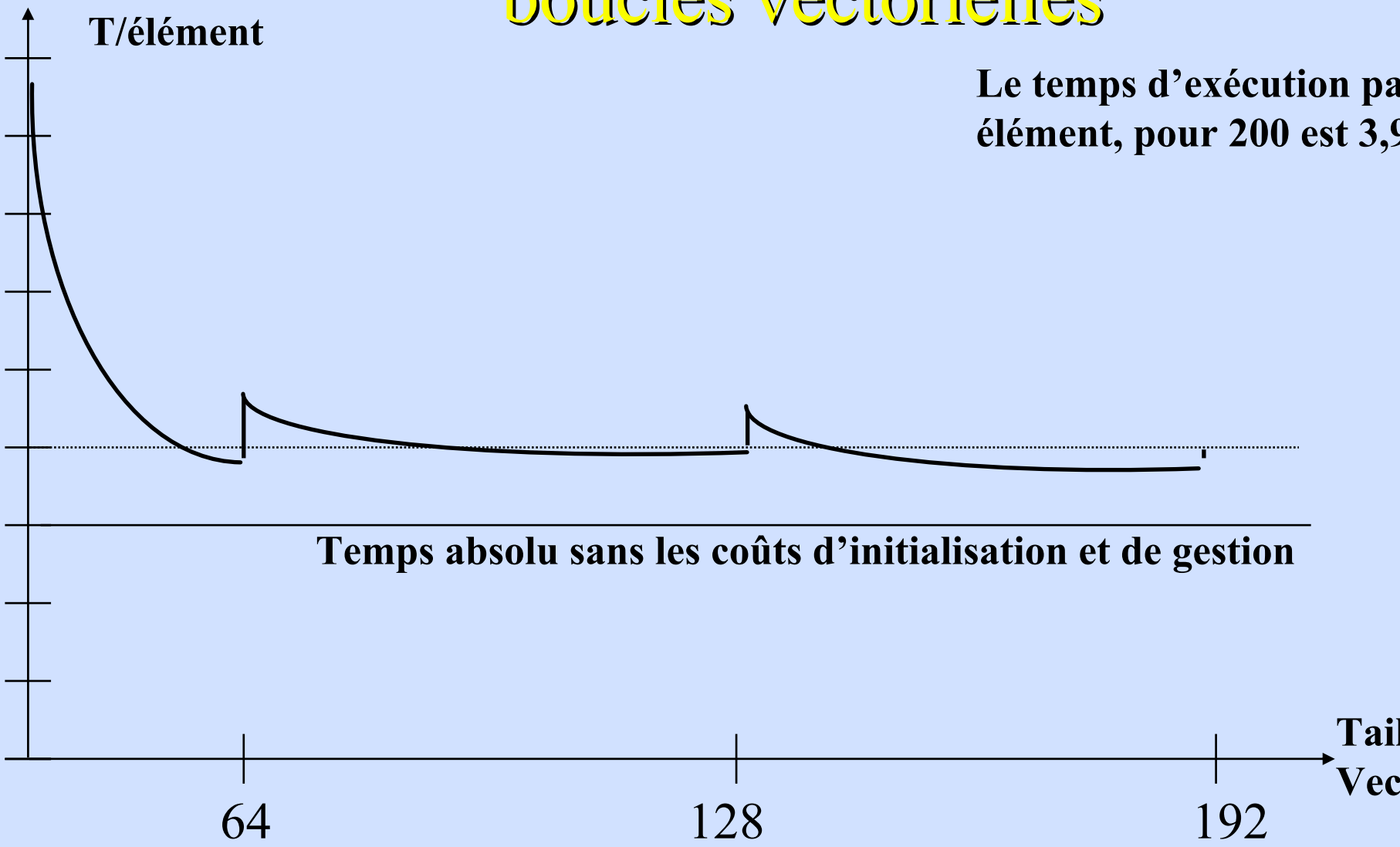
Ra = Adresse de A Rb = Adresse de B Fs <-

R0 contient 0

```
        ADDI      R2,R0,#1600     R2 <- Nb octets du vecteur
        ADD       R2,R2,Ra        R2 <- Adresse de fin de A
        ADDI      R1,R0,#8        Calcul de la taille du premier sou
        MOVEI2S   VLR,R1          VLR <- Lg du sous vecteur N°1
        ADDI      R1,R0,#64       R1 <- lg du premier vecteur
        ADDI      R3,R0,#64       R3 <- lg des vecteurs suivants

loop:   LV        V1, Rb          Chargement du vecteur V1
3*n     MULTVS    V2,V1,Fs        V2 <- V1 * Fs
        SV        Ra, V2          Rangement du registre V2
                                  Passage aux vecteurs suivants
        ADD       Ra,Ra,R1        de A
        ADD       Rb,Rb,R1        de B
        ADDI      R1,R0,#512      R1 <- 64 * 8
        MOVEI2S   VLR,R3          VLR <- 64
        SUB       R4,R2,Ra        R4 <- R2 -Ra
        BNZ       R4, loop        Brch si R4 != 0
```

2.1.6) Évaluation de performances des boucles vectorielles

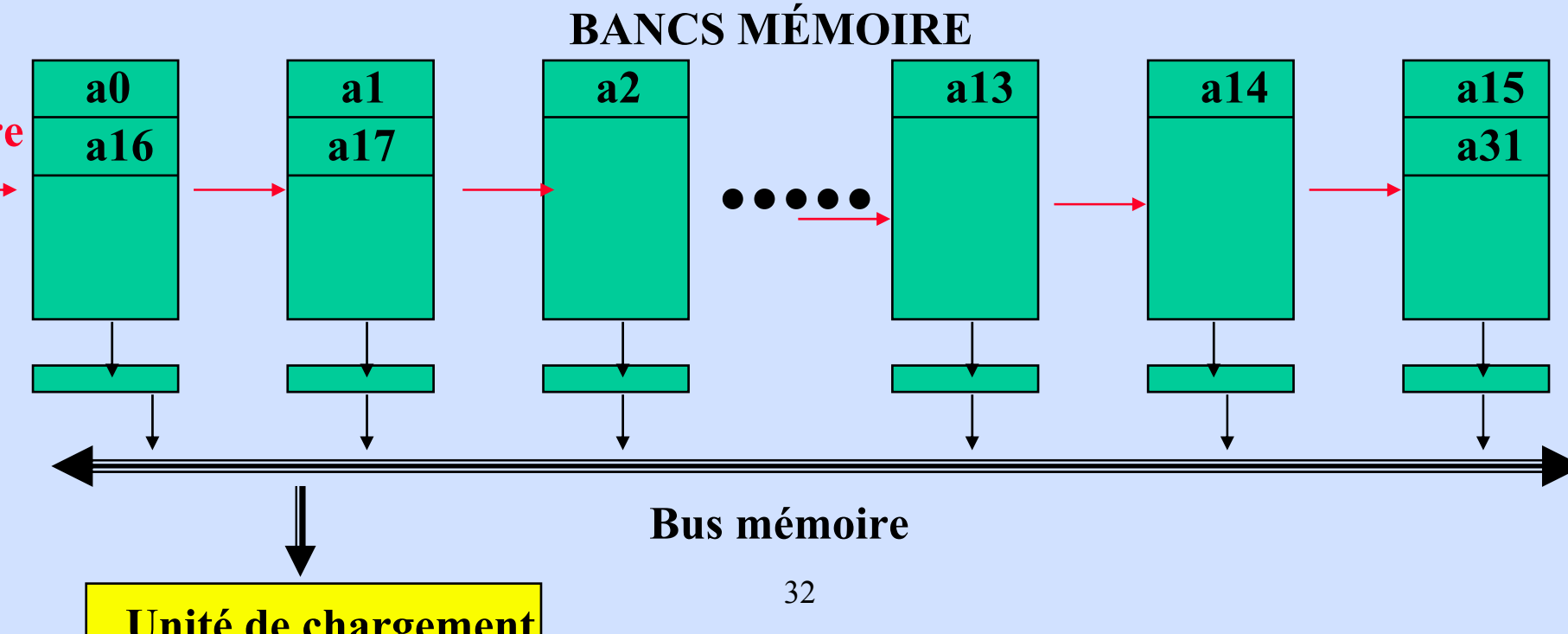


2.1.7) Évaluation de l'accès mémoire

ne mémoire centrale à 16 bancs et un temps d'accès à chaque banc de 12 cycles.
s mémoire délivre un élément tous les cycles.
ien de cycles sont nécessaires pour charger dans le processeur un vecteur de 64 é

a) consécutifs (stride de 1)

b) séparés de 32 éléments (stride de 32)



2.2) Le modèle Data Parallel sur multiprocesseu

```
for   i <- 0  step 1  until   n-1  do
      begin
      C[ i, k] := 0                                   (o <= k <=
      for j <- 0 step 1 until  n-1  do
      C[ i, k] := C[ i, k]  + A[ i, j] * B[ j, k] (o <= k <=
      end
```

t être diffusé à tous les processeurs

Couple (banc mémoire, proce

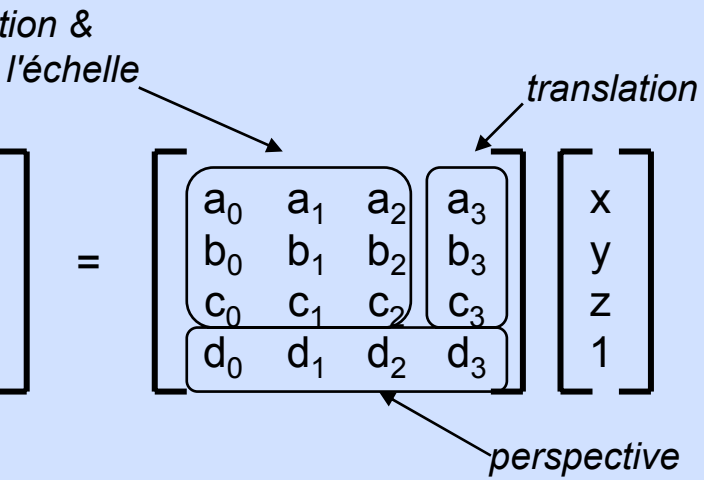
- Calcul simultané de tous les éléments de la ième ligne
- Addition en série mais multiplication en parallèle

Exemple SSE2 Intel

- graphique 3D, encodage et décodage vidéo, reconnaissance de la parole, audio, internet (encryptage, décryptage), applications scientifiques.
- 70 nouvelles instructions SIMD
- Calculs sur 128 bits
  - 2 flottants double précision
  - 2 fois plus d'entiers que MMX
- Amélioration du contrôle du cache
  - Ne pas cacher trop de données
  - Préchargement
- 8 registres 128 bits

Calcul pour le graphique 3D

- Les objets 3D sont représentés par des milliers de polygones
  - Chaque somment est représenté par un quadruplet X, Y, Z, W avec W une information de correction de perspective.
  - Dès qu'un objet se déplace ou tourne ou grandit ou diminue, une transformation géométrique doit être appliquée pour recalculer la forme du polygone via une multiplication de matrice.
  - 16 multiplications et 12 additions pour chaque sommet.



$x' = a_0x + a_1y + a_2z + a_3$

Sans SSE

16 multiplications et 12 additio

Avec SSE

Une instruction PMADD par li

la matrice i.e. 4 instructions

Le modèle Data Parallel sur multiprocesseur

```
For k:= 0 step 1 until n-2 do
   Fork next
     k:= n-1 ;
next :        For i:= 0 step 1 until n-1 do
                  C[i,k] := 0 ;
                      For j:= 0 step 1 until n-1 d
                      C[i,k] := C [i,k] + A[i, j] *
   Join N ;
```

r le *Fork* une tâche (le code de *next* à *Join*) **est alloué à un ocesseur libre**

us les éléments de la ième ligne sont calculés en parallèle, sur de ocesseurs parallèles associés à une valeur de k.

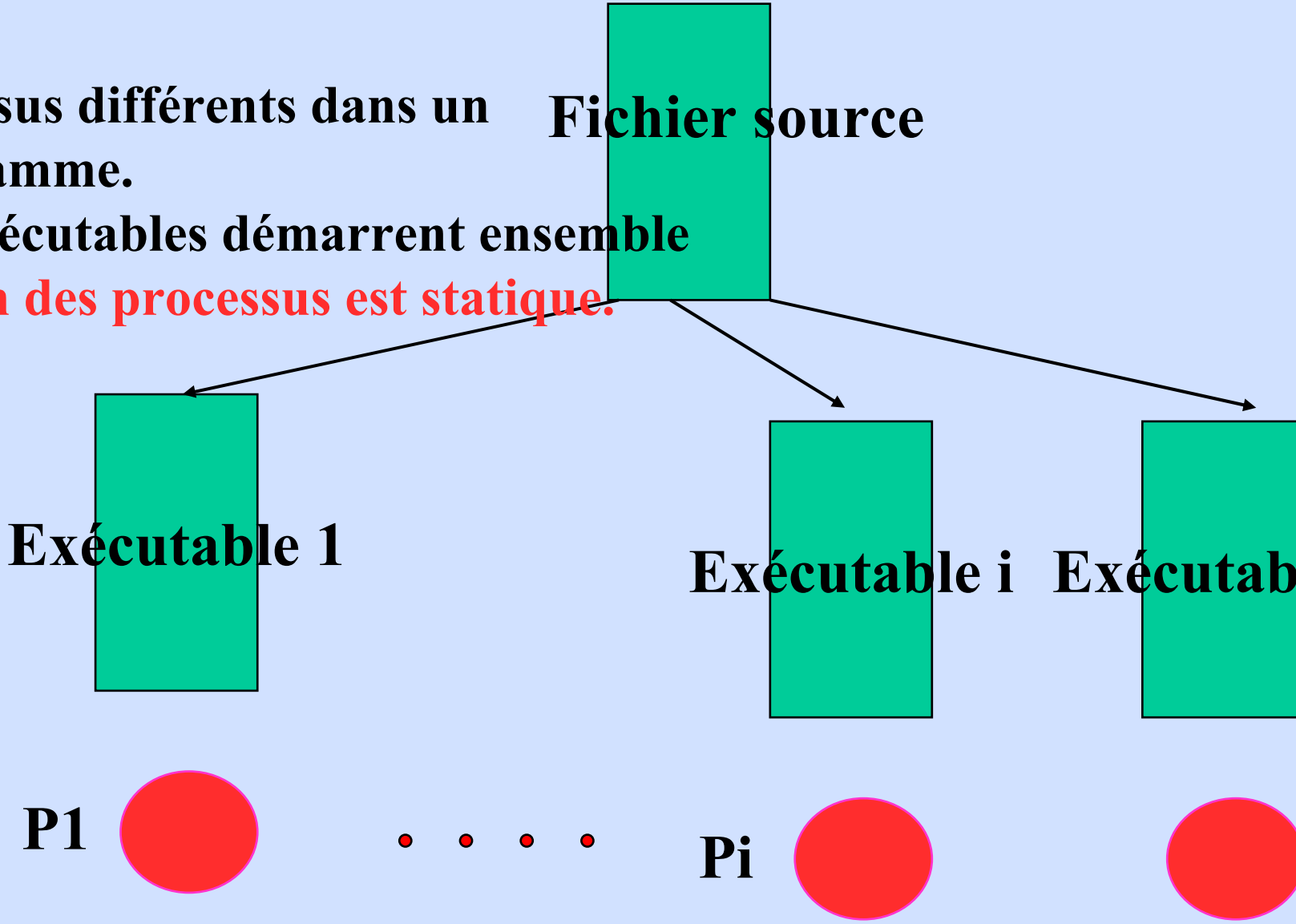
r *Join* N <- N + 1 ; si C >= N alors poursuite en séquence du

2.3) Le modèle SPMD

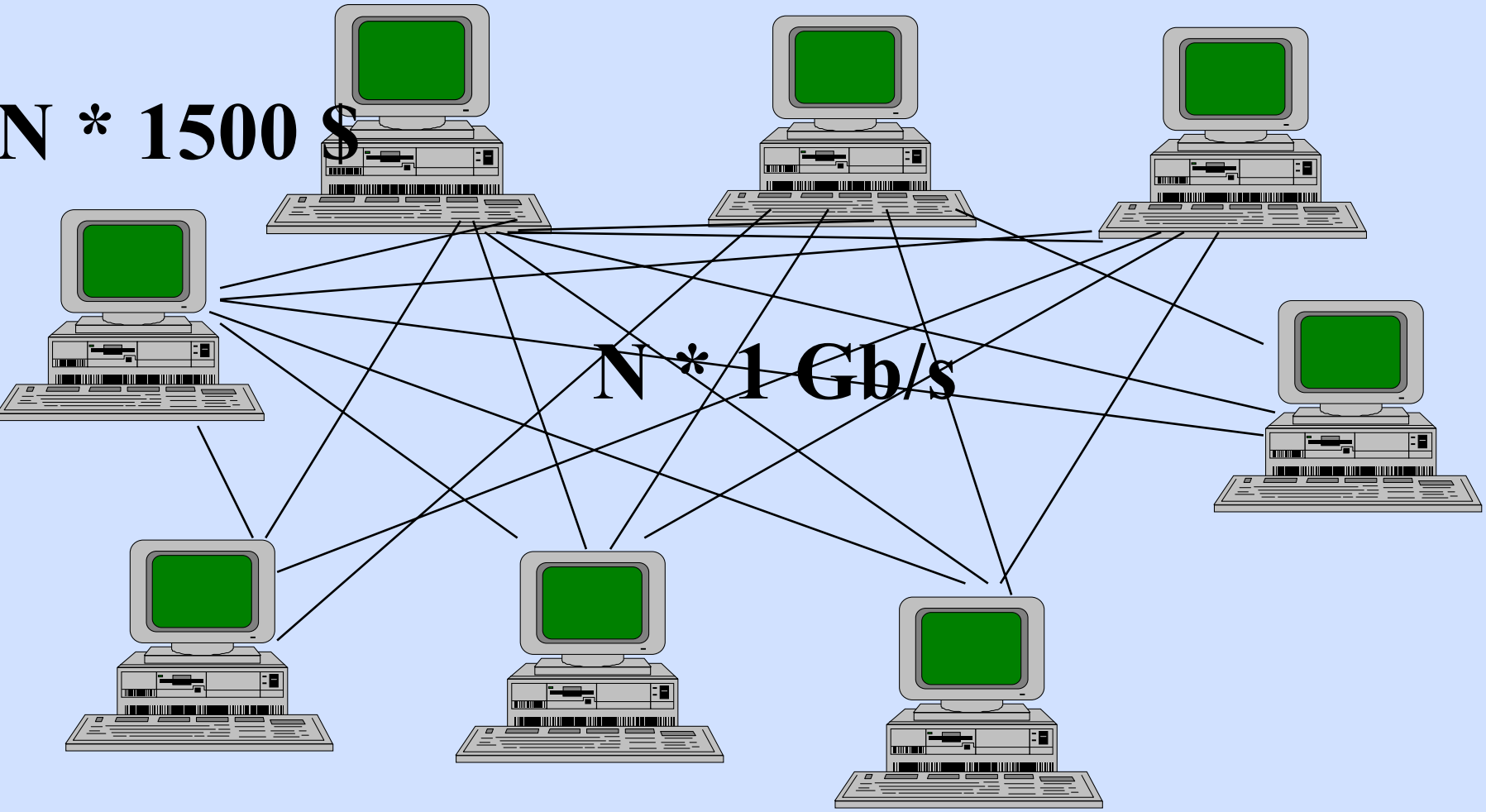
processus différents dans un programme.

s les exécutables démarrent ensemble

réation des processus est statique.

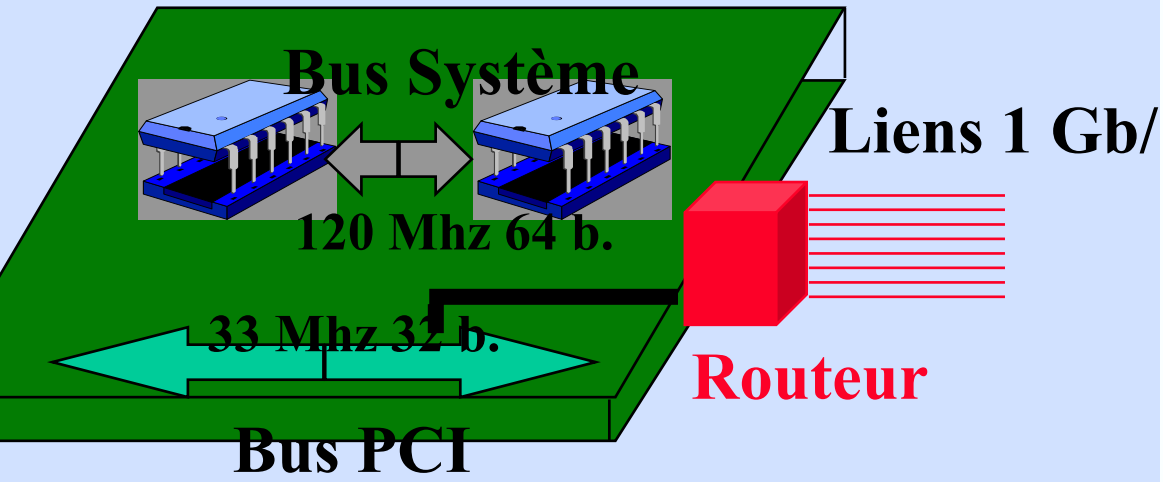


a) Clusteurs et NOW : SCI, Myrinet,...



b) Exemple Cluster de PC

Processeurs Intel/Pentium

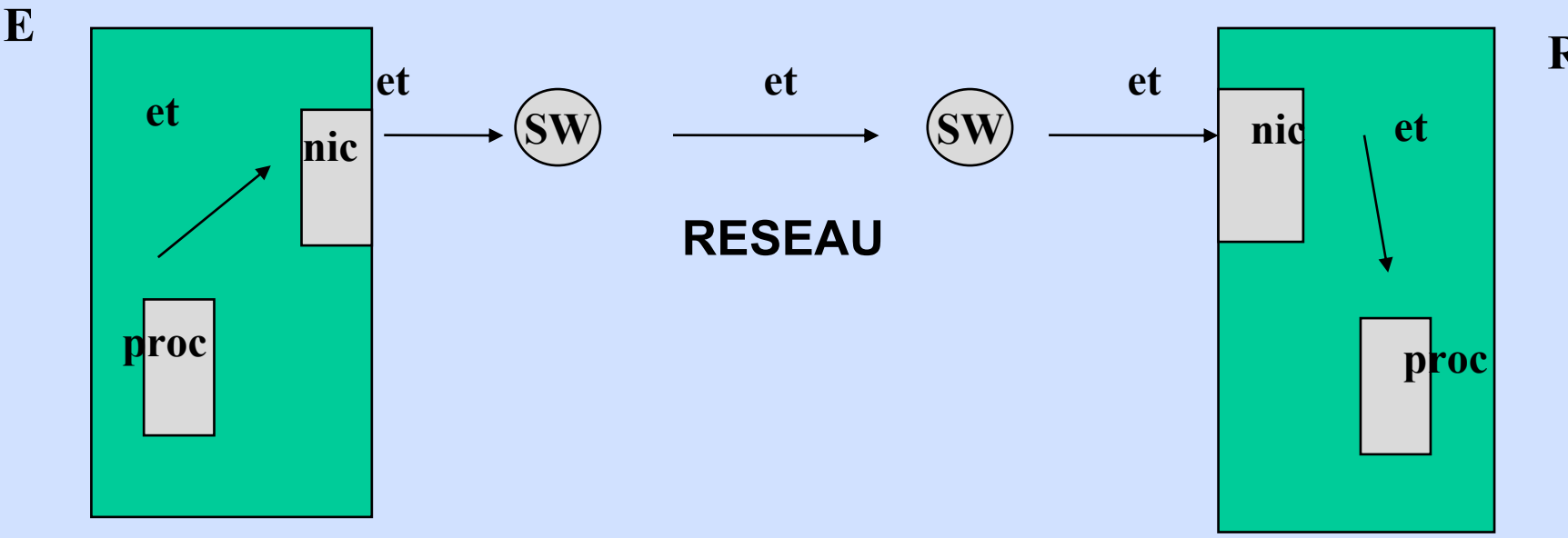



06
10
14
18
22
26
30
34
07
11
15
19
23
27
31
35
16
20
24
28
32
36

c) Exemples de réseau haute performance

- Scalable Coherent Interface (SCI)
  - Interconnexion pour clusters : Espace d'adressage global
- Asynchronous Transfert Mode (ATM)
  - Commutation de paquets de taille fixe (53 0ctets)
- Myrinet
  - Gigabit/s communication de paquets (machine parallèle)
- HiPPI
  - Canal point à point parallèle de 32 ou 64 bits à 25 Mbit/s par ligne

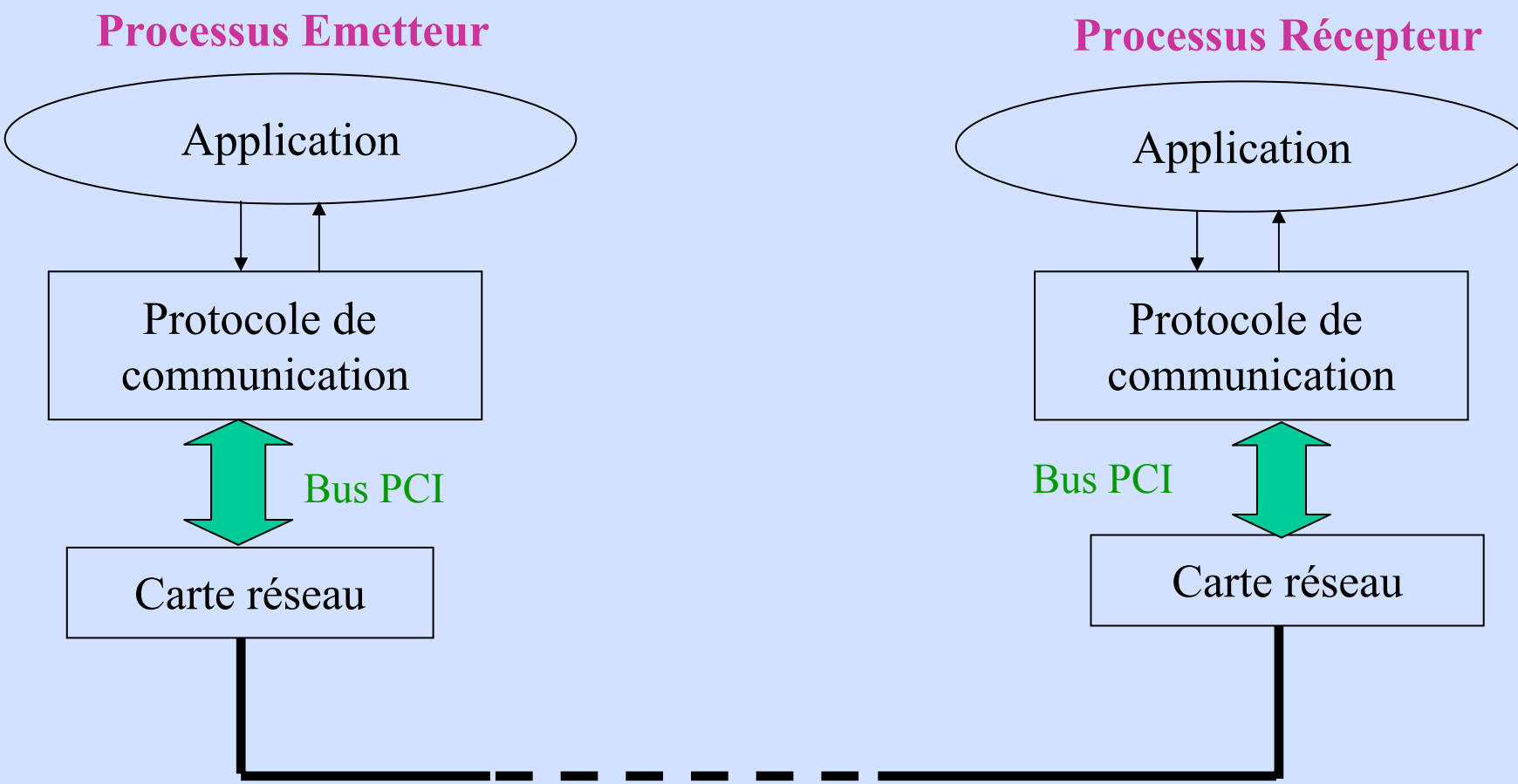
Schéma d'une communication



2.4.3) Définitions

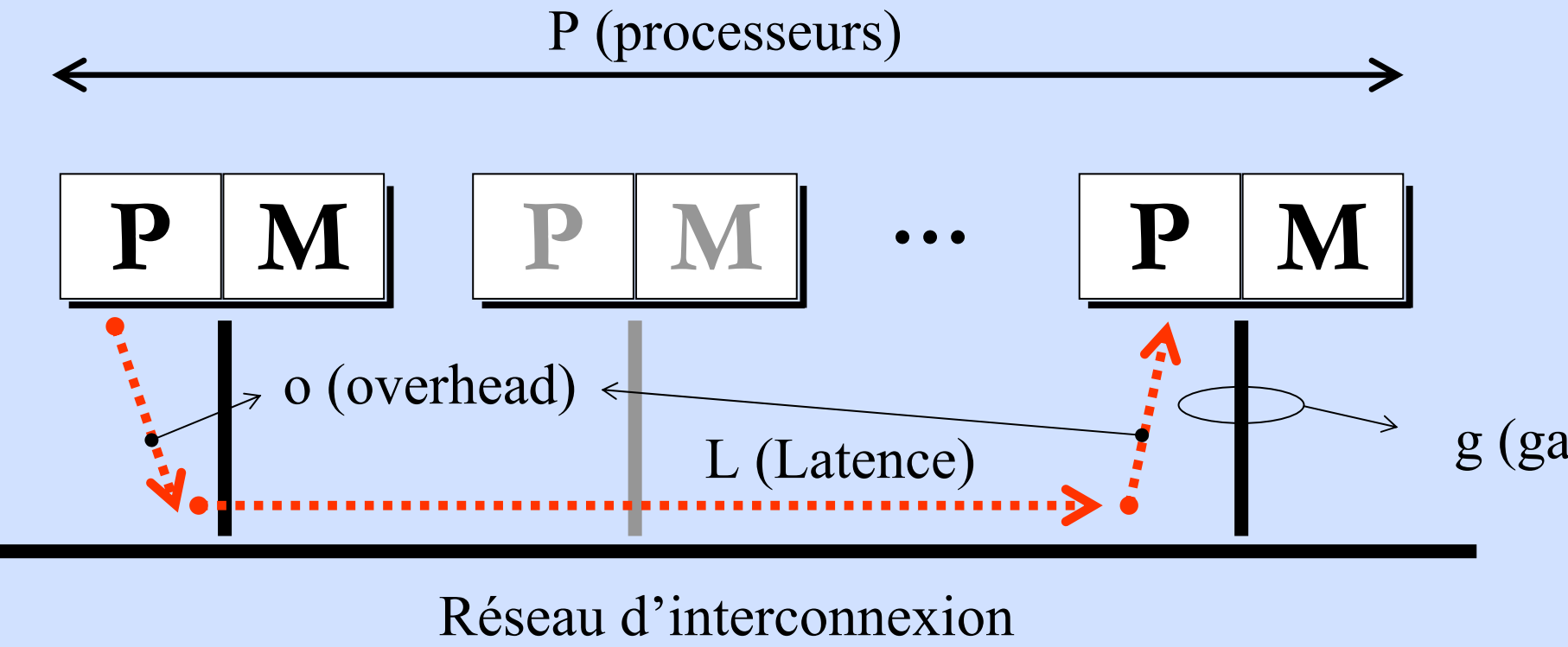
- Bande passante : Débit maximal pour propager l'information après son entrée sur le réseau
- Temps de transmission : Taille du message / bande passante (sans contention)
- Temps de transit : Temps pour que le premier bit arrive au récepteur
- Coût émission : Temps pour un processeur, pour délivrer un message sur le réseau
- Coût réception : Temps pour un processeur, pour retirer un message du réseau

Réseau d'Interconnexion

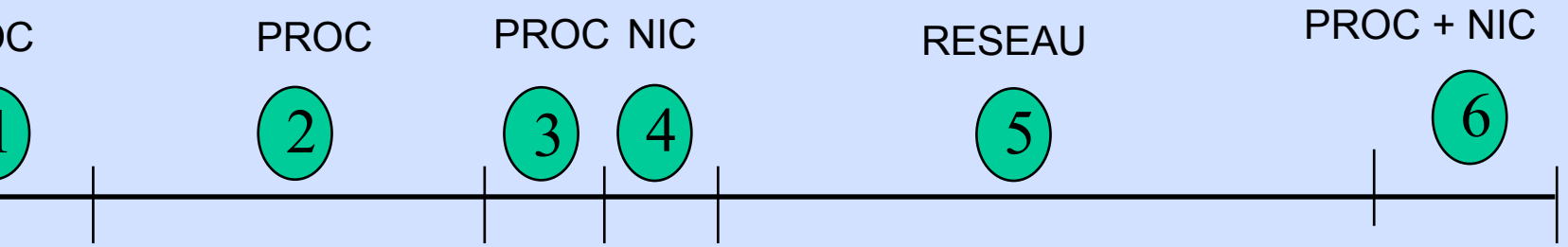


T(n)E-R = overhead + routage + latence canal + content

Le modèle LogP



2.4.4) Le coût d'une émission



Émetteur place le message dans un buffer de sortie

Préparation des paquets et transfert dans une file pour émission

Activation du handler de messages

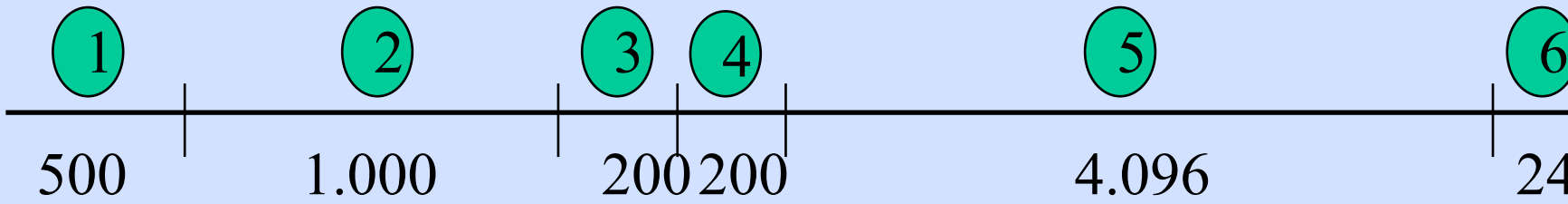
Exécution du handler de messages

Temps d'occupation de la bande passante du réseau

Temps d'occupation du support hardware

Exemple

Processeur 1 Ghz : une instruction/cycle
Réseau : BP 2 Gb./s.
Messages : 1 Ko.



Coût de la communication : 6.020 instructions dont 1.924 dans le

2.4.5) Calcul du temps d'exécution en paral

$$T = T_{seq} + T_{com}$$

q est le temps d'exécution dans chaque processeur
m le temps de LA MESSAGERIE.

$$T = n * g + Nc\,(Ts + Lc*Tb)$$

n = nombre de " paquets " d'instructions dans chaque proces
g = nombre d'instructions par " paquets "
Nc = nombre de communications
Ts = temps d'établissement de la communication
Lc = longueur du message
Tb = temps d'émission par octets transmis

Si exécution et communication peuvent être recouvertes alors :

T = max (n*g + Nc*Ts, Nc*Lc*Tb)

- Exemple IBM/SP2

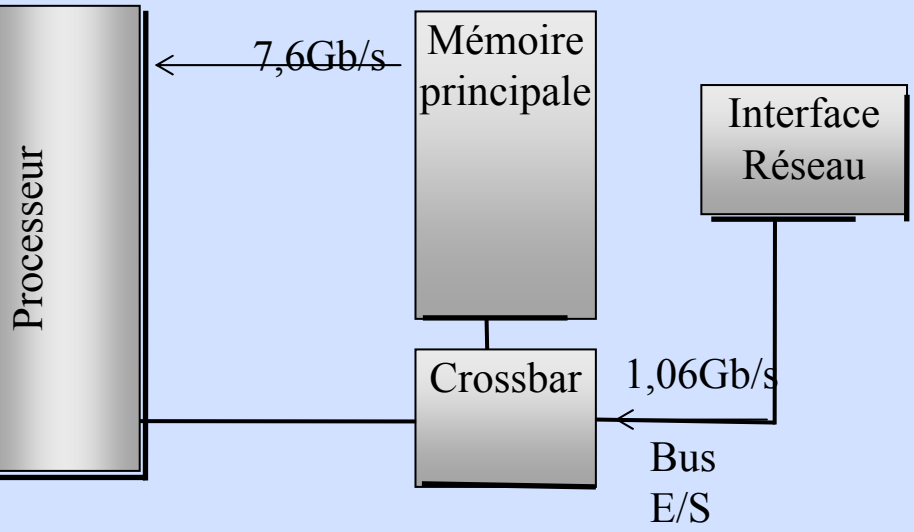
Ts = 39 micros.
Tb = 0,028 micros.
Perf. du proc. = 328 Mflops/s. => 328 inst.flot/micros.

- Intel Pentium IV 3 Ghz
ou AMD Athlon 1.8 Ghz

Com ~ 60,000 instruction

renant 2 messages par flots (Nc =2) et Lc=1024 octets:
max (n*g + 78, 57) =
n*g + 78 micros. => latence équivaut à 328*78 = 25.000 instructio

Exemple : Réseau Myrinet sur un Cluster de



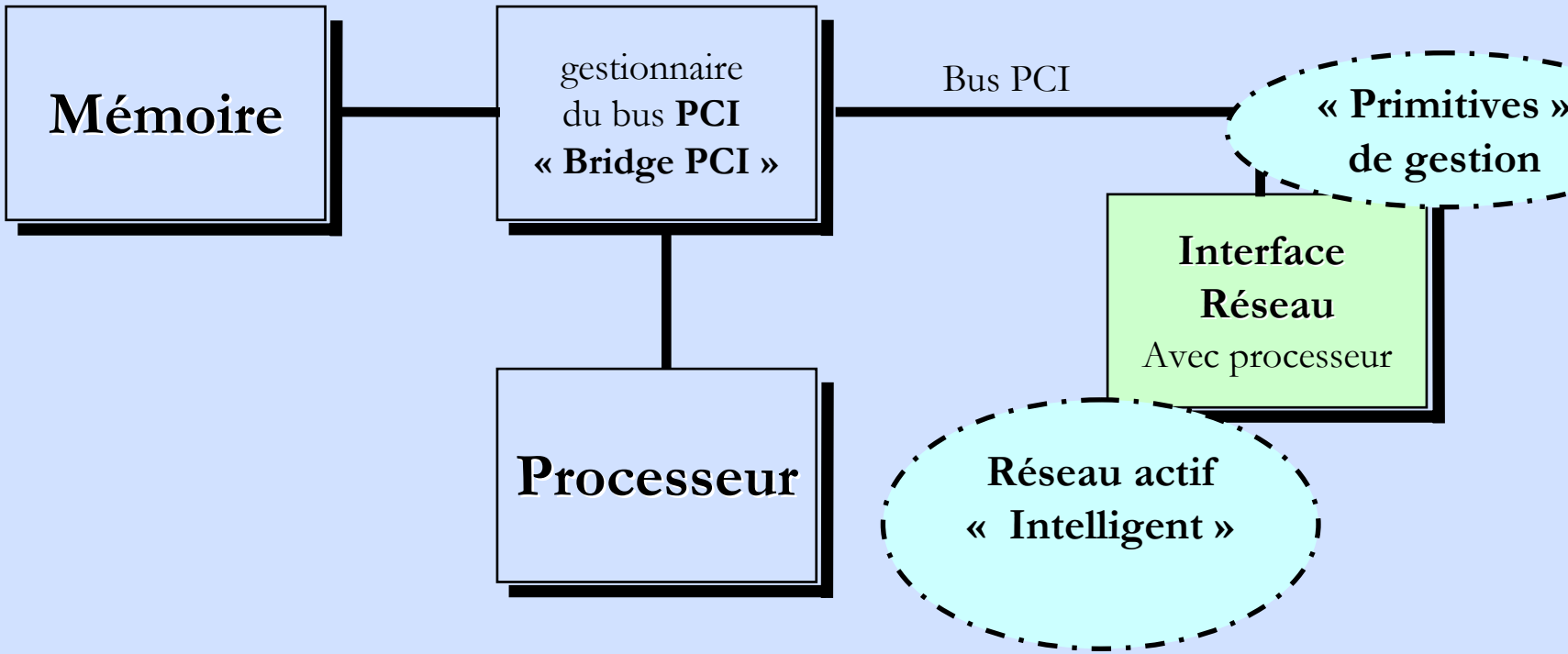
La bande passante

- Mémoire Principale **960 Mo/s =>**
- Bus PCI **32bits à 66 MHz => 2.12**
- Le réseau **de 10 Mb/s à 1 Gb/s**

Les latences

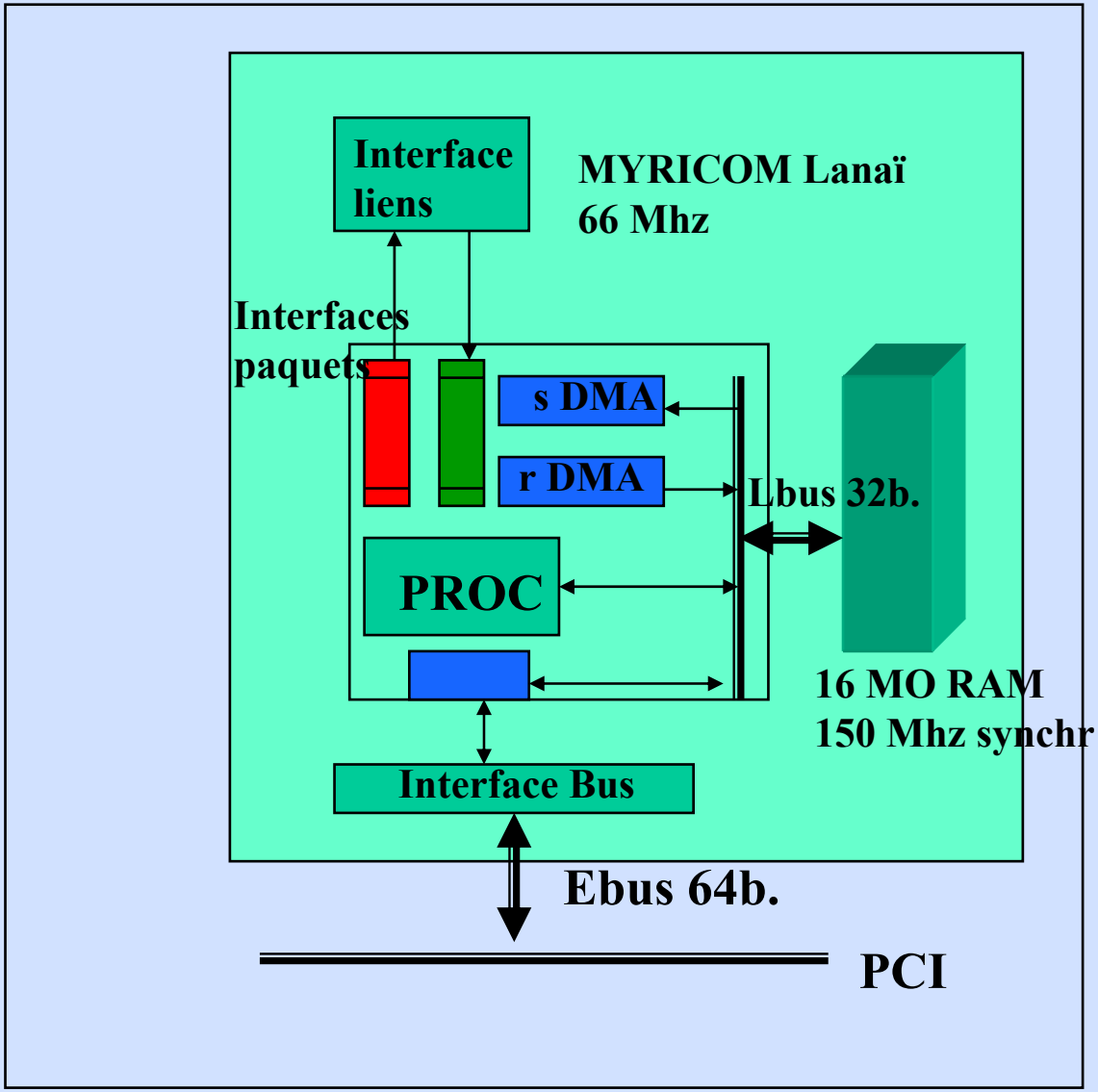
- Latence de transport entre 100 et 600 n
- Latence du protocole de gestion, en µS

Overhead « Interface PCI-Lanaï»

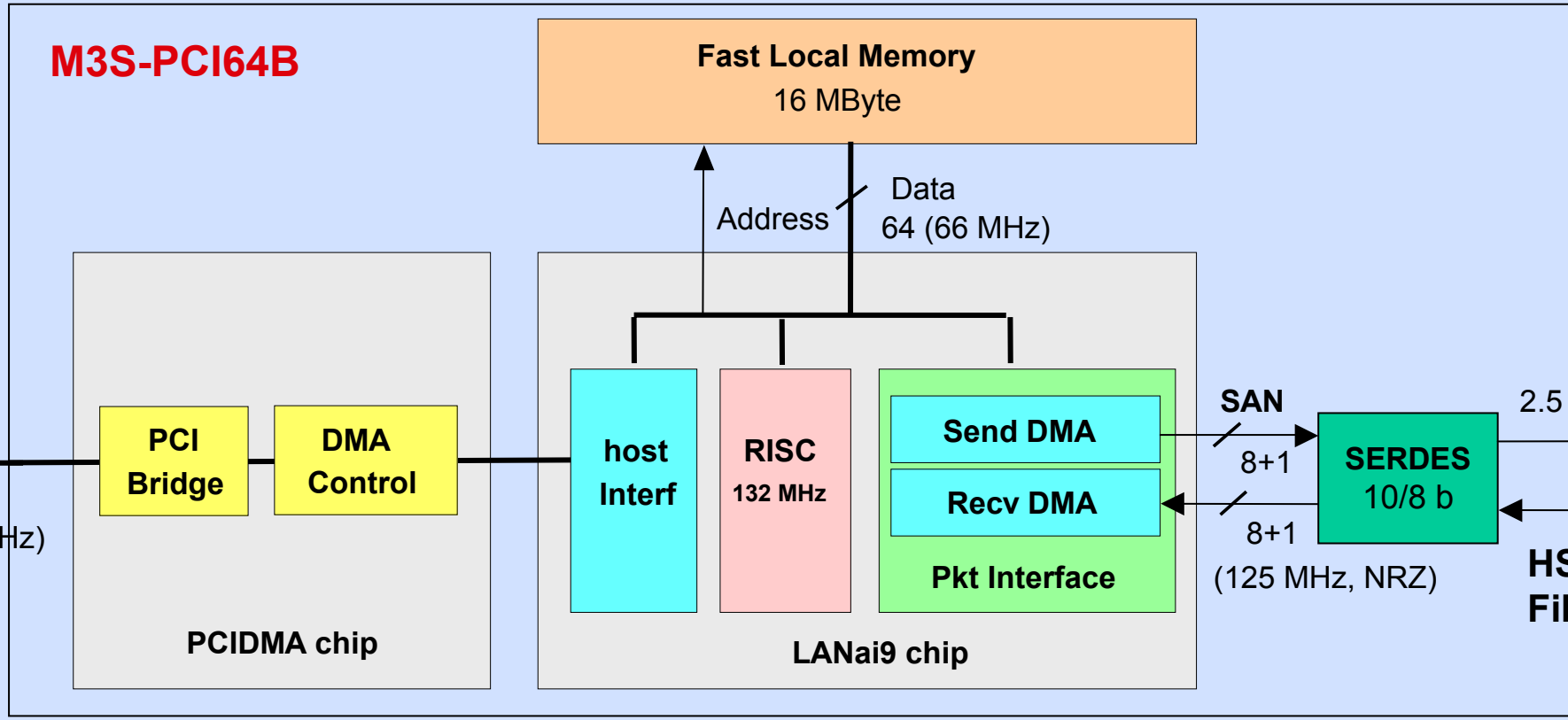


250 Mb/s

8 ports switch whormole



LANai Chip Architecture, V9



emory system: supply 64b x 132 MHz = 1056 MB/s

peak demand	PCI	512 MB/s
	RDMA	250 MB/s
	SDMA	250 MB/s
left for CPU		44 MB/s

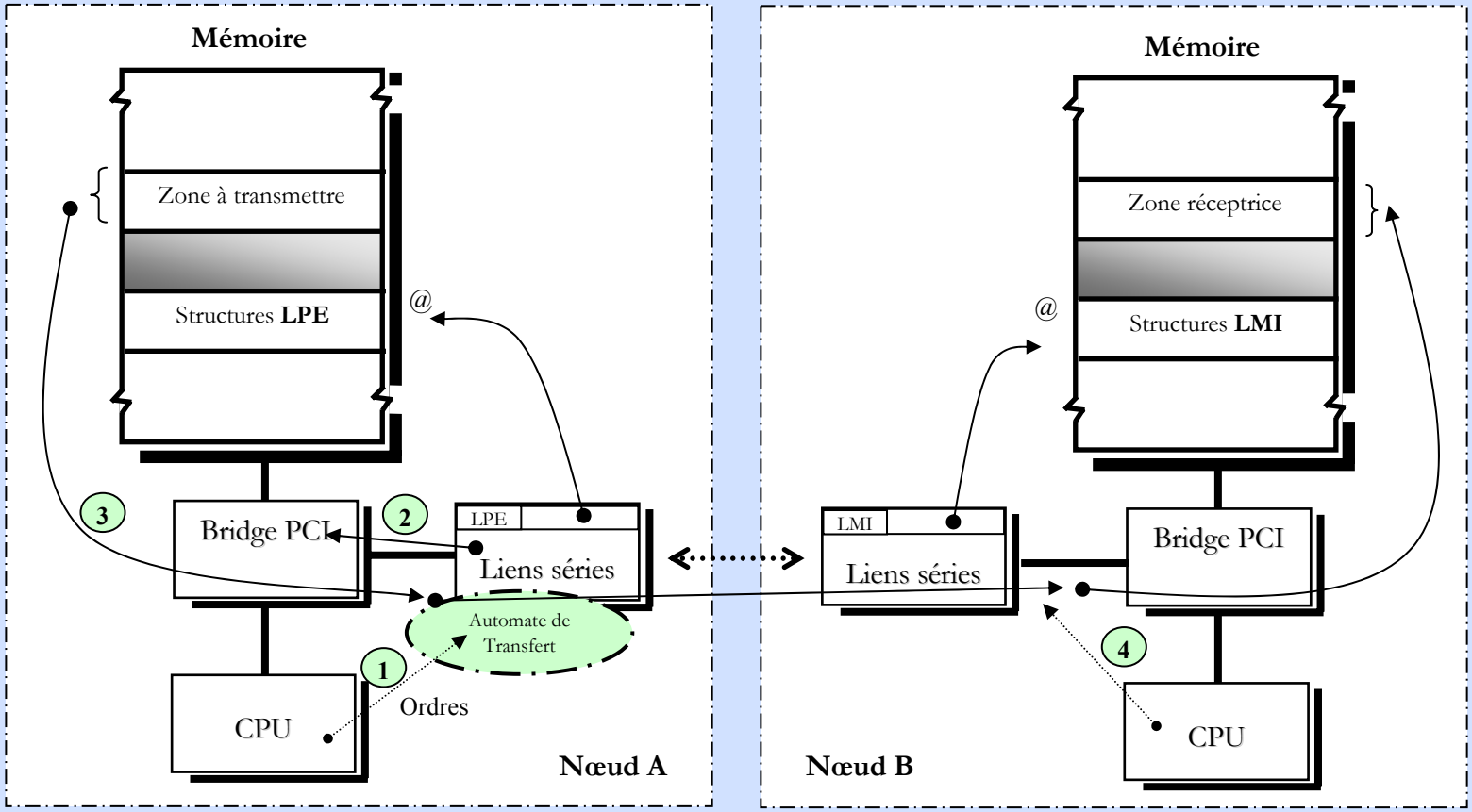
priority

The NIC is implemented as four basic elements:

- A bus interface
- A link interface
- A static memory (SRAM)
- The Lanaï chip
  - embedded processor
  - three DMA engines
  - two link FIFOs

The design goal of the firmware is to keep all three DMA engines active simultaneously.

Several user processes can have the ability to write messages into the N and read messages directly from the NIC, or to write message descripto containing pointers to data that is to be DMA transferred through the ca

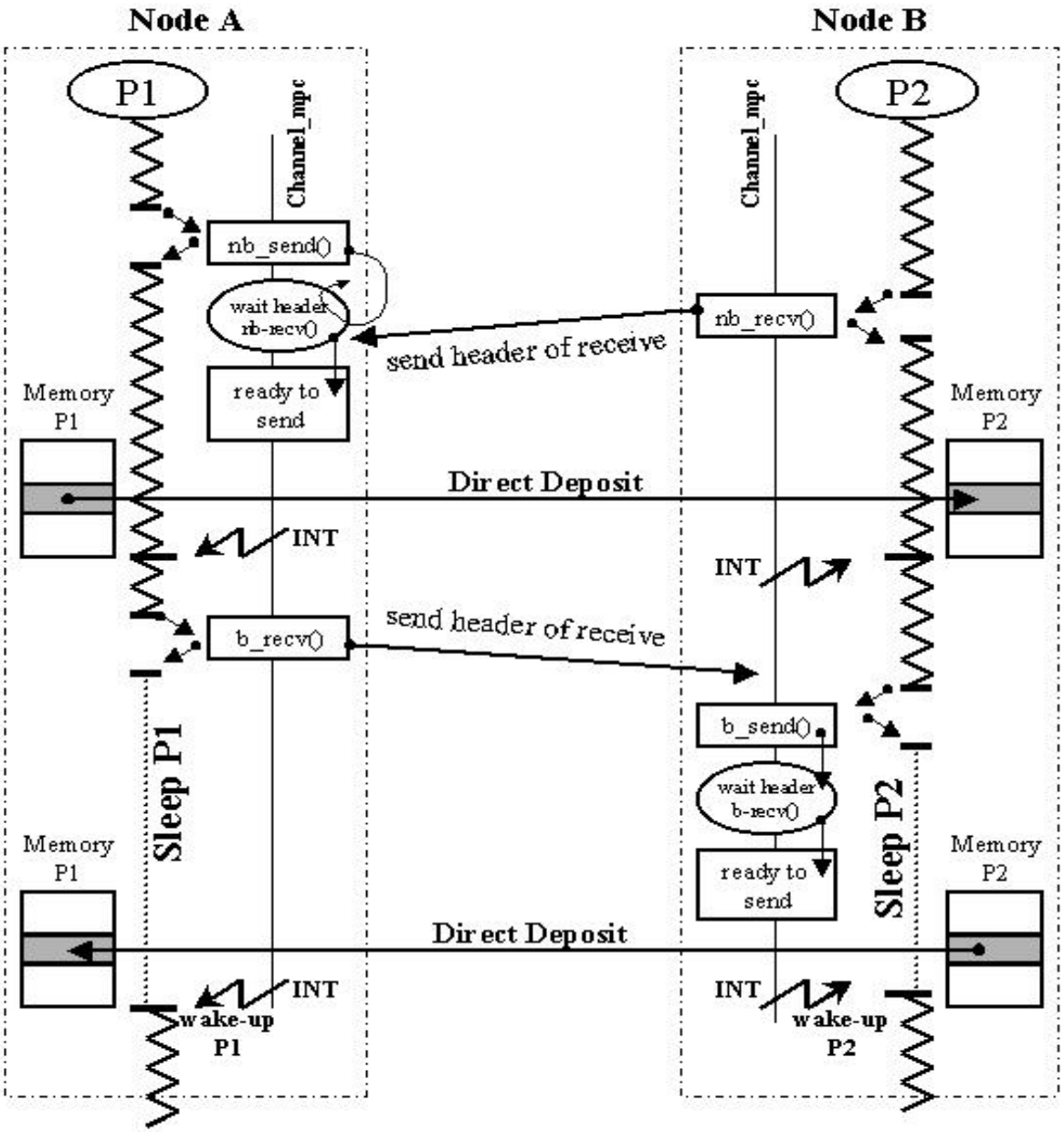


(1) Ordre du CPU à l'automate pour transférer.

(2) L'automate déclenche le transfert grâce au DMA.

(3) Le transfert est réalisé directement de mémoire à mémoire.

(4) Le CPU récepteur scrute la réception des données .

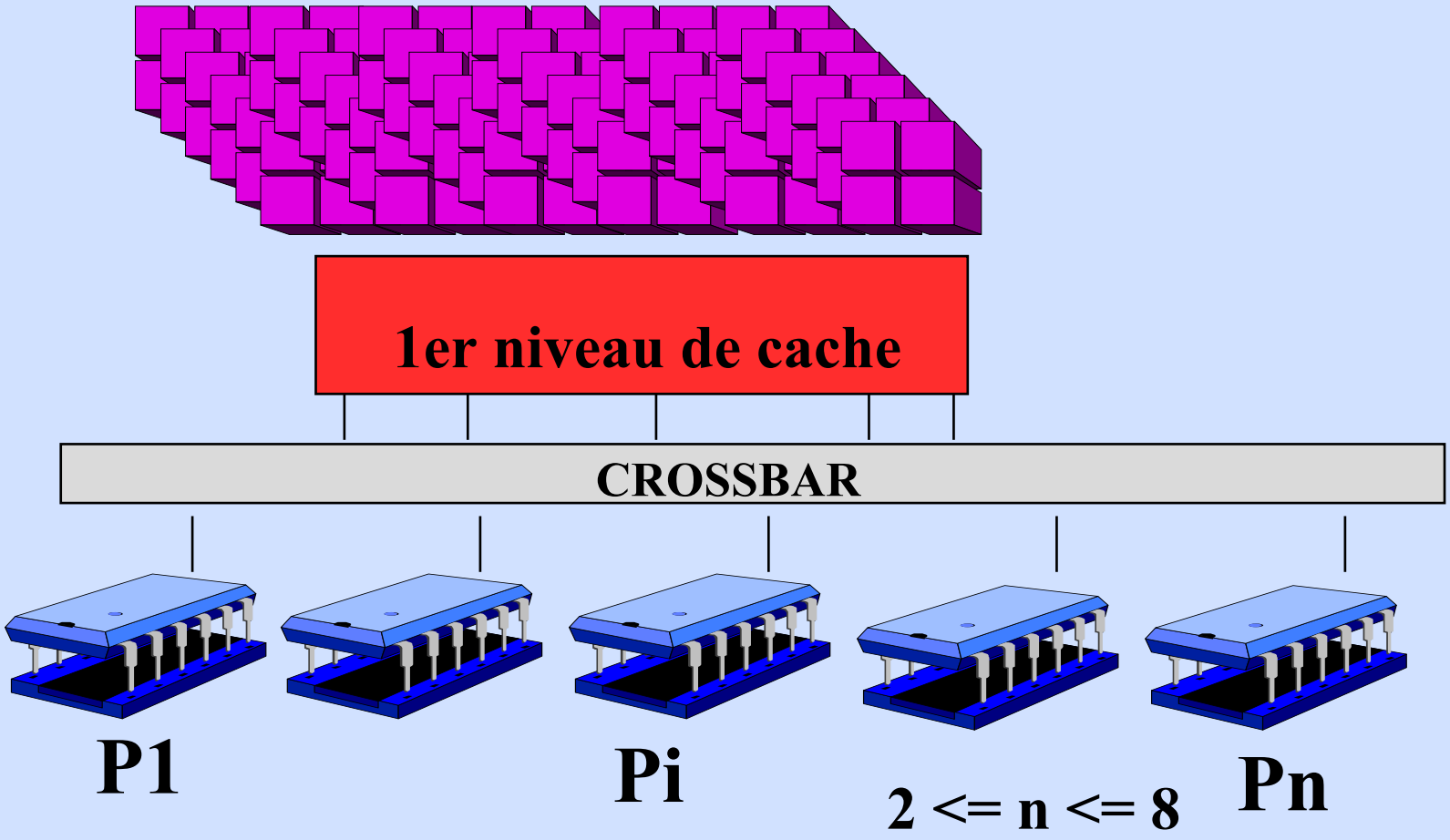

Node A
Node B
P1
P2
Channel_mpc
Channel_mpc
nb_send()
wait header
nb-recv()
ready to
send
nb_recv()
send header of receive
Memory
P1
Memory
P2
Direct Deposit
INT
INT
b_recv()
send header of receive
b_send()
wait header
b-recv()
ready to
send
Sleep P1
Sleep P2
Memory
P1
Memory
P2
Direct Deposit
INT
INT
wake-up
P1
wake-up
P2

Les calculateurs parallèles à mémoire partagée : cohérence et consistance

1) Architecture (1)

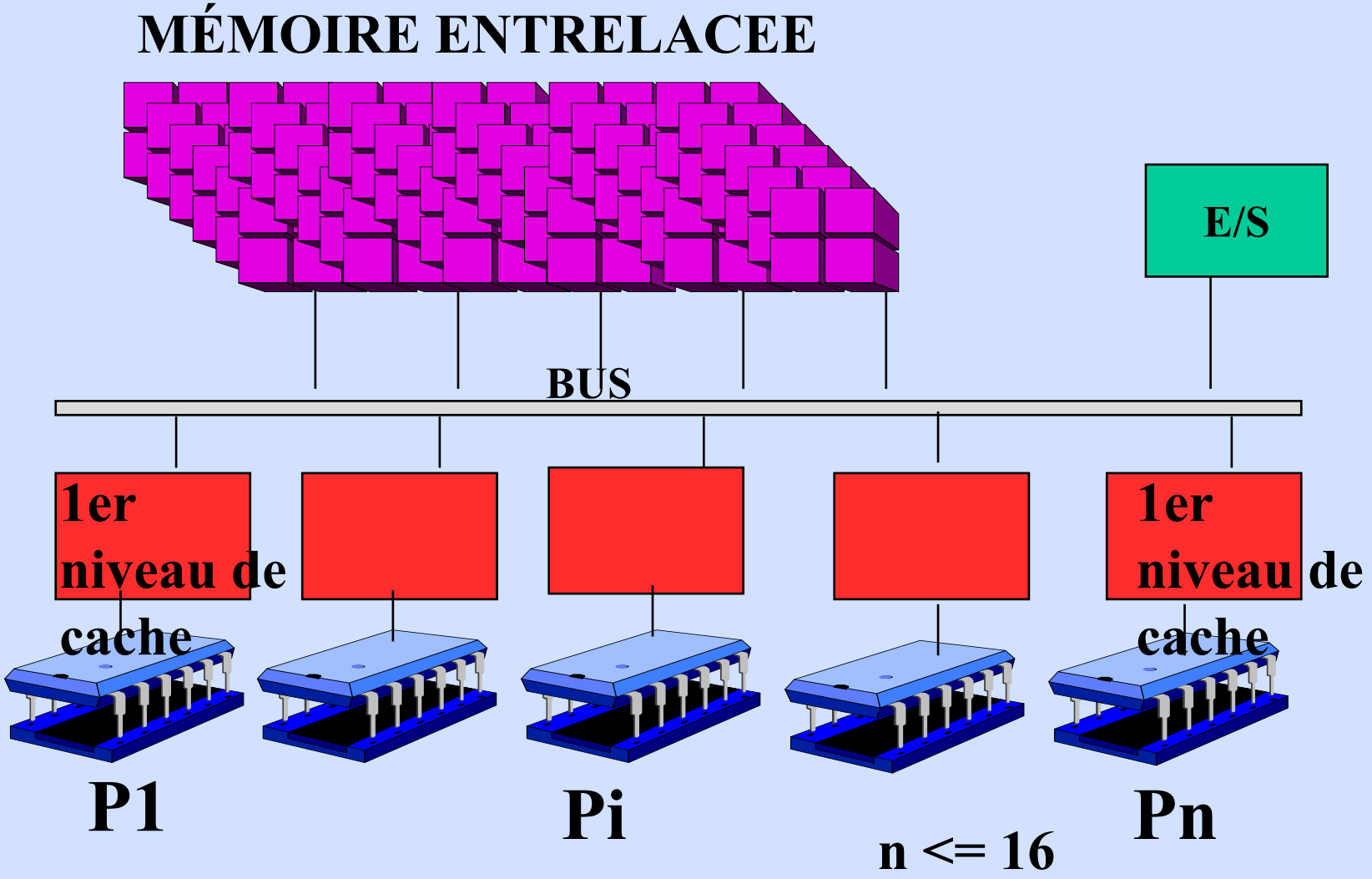
a) Cache partagée

MÉMOIRE ENTRELACEE



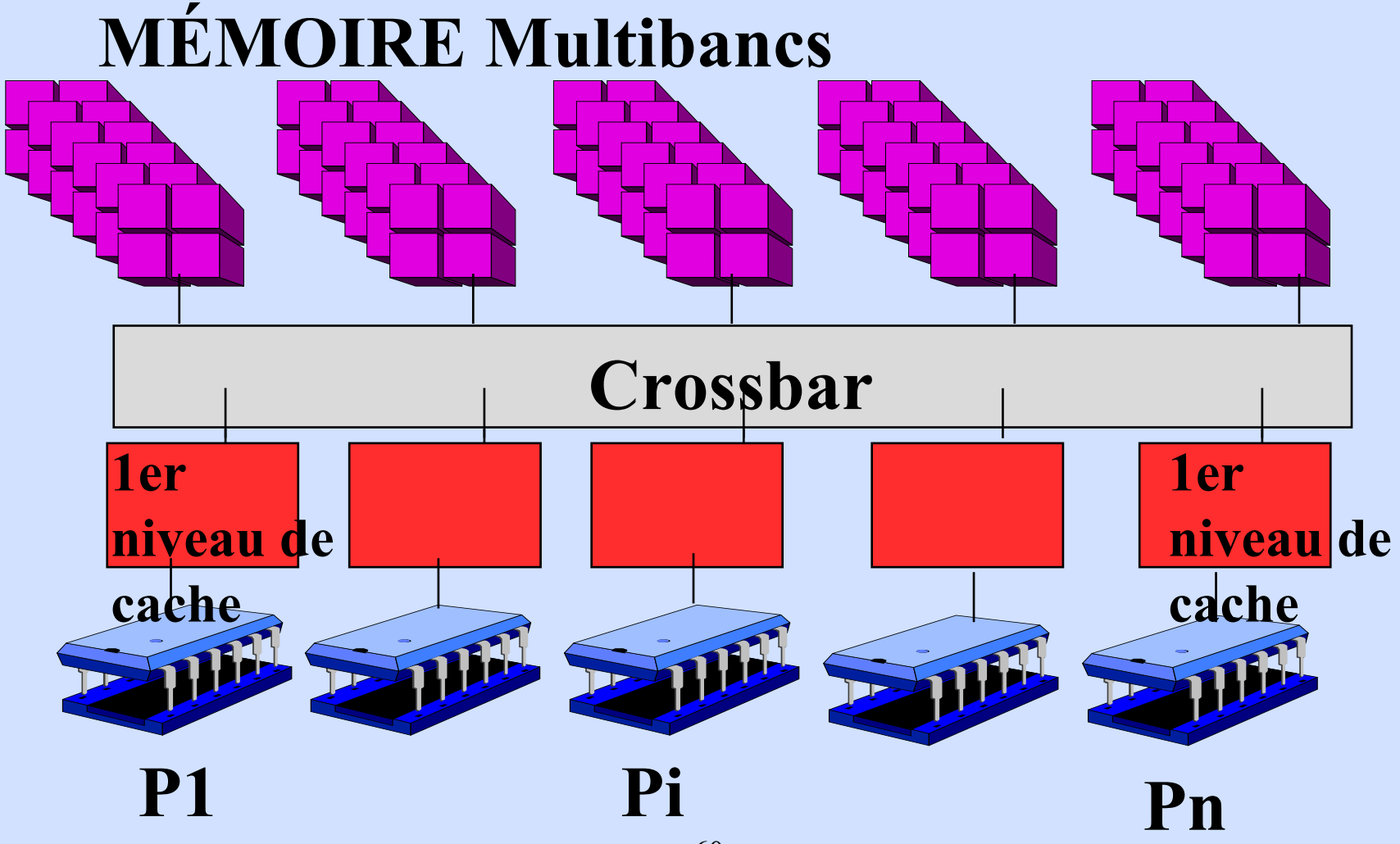
Architecture (2)

b) Bus mémoire partagé



Architecture (3)

c) Architecture hall de dans



2) Caractéristiques

• Toutes les mémoires principales sont à égale distance de chaque processeur.
• Les caches jouent un rôle essentiel pour réduire la latence d'accès à cette mémoire

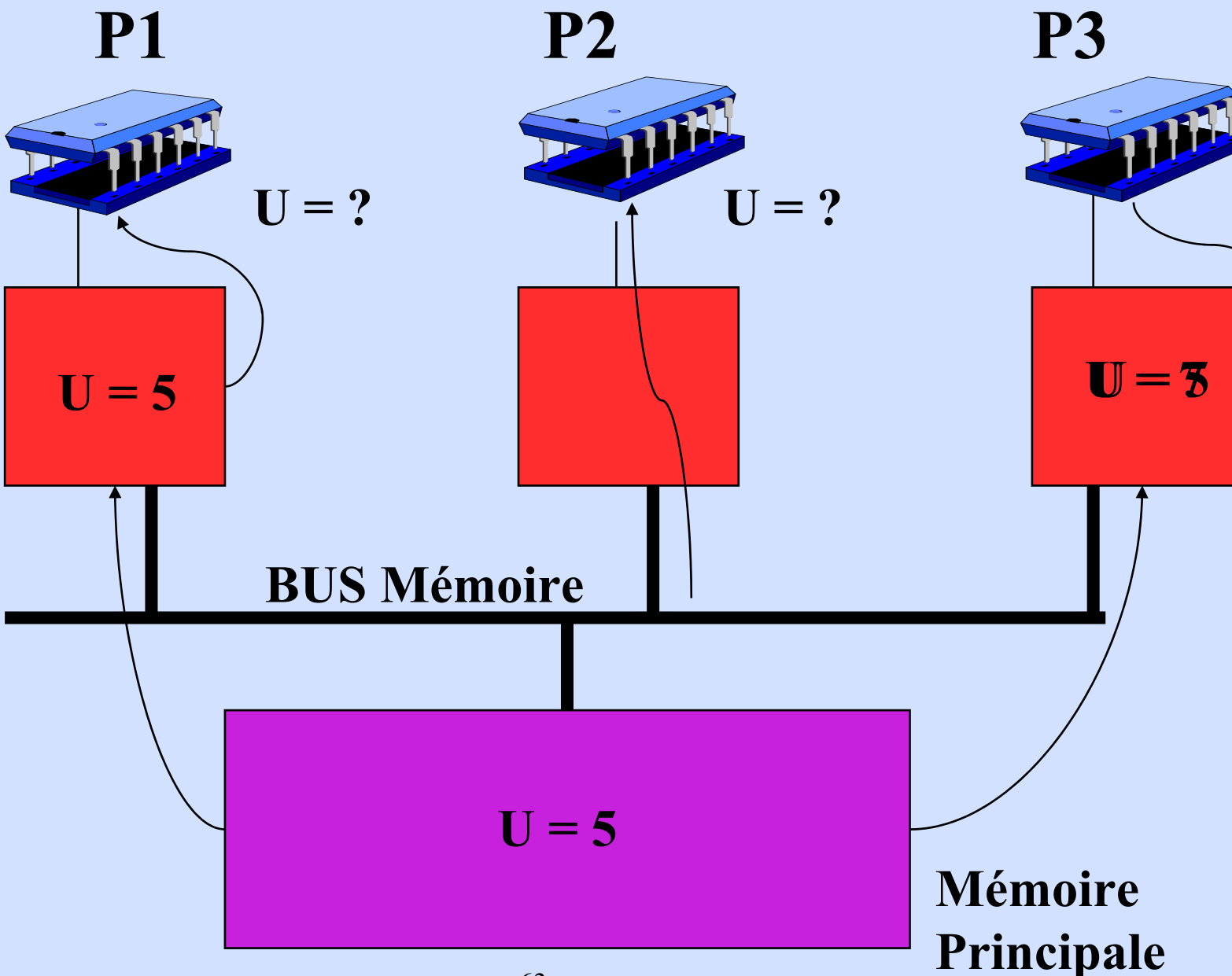
Problème N°1 : la cohérence des caches
La lecture d'un mot doit retourner la valeur de la dernière écriture effectuée à cet emplacement.

Problème N°2 : la consistance de la mémoire
Un modèle de consistance mémoire spécifie des contraintes sur l'ordre dans lequel toutes les opérations mémoire sont effectuées.

3) Cohérence

Un système mémoire est cohérent si le résultat de n'importe quelle exécution est tel que, pour chaque donnée partagée il est possible de construire un ordre complet (toutes les L/E de tous les processus) conforme avec le résultat d'une exécution pour lequel

- Les opérations émises par n'importe quelle tâche se réalisent dans l'ordre dans lequel elles auraient été émises par cette tâche
- La valeur retournée par chaque lecture est la dernière valeur écrite dans cette même mémoire

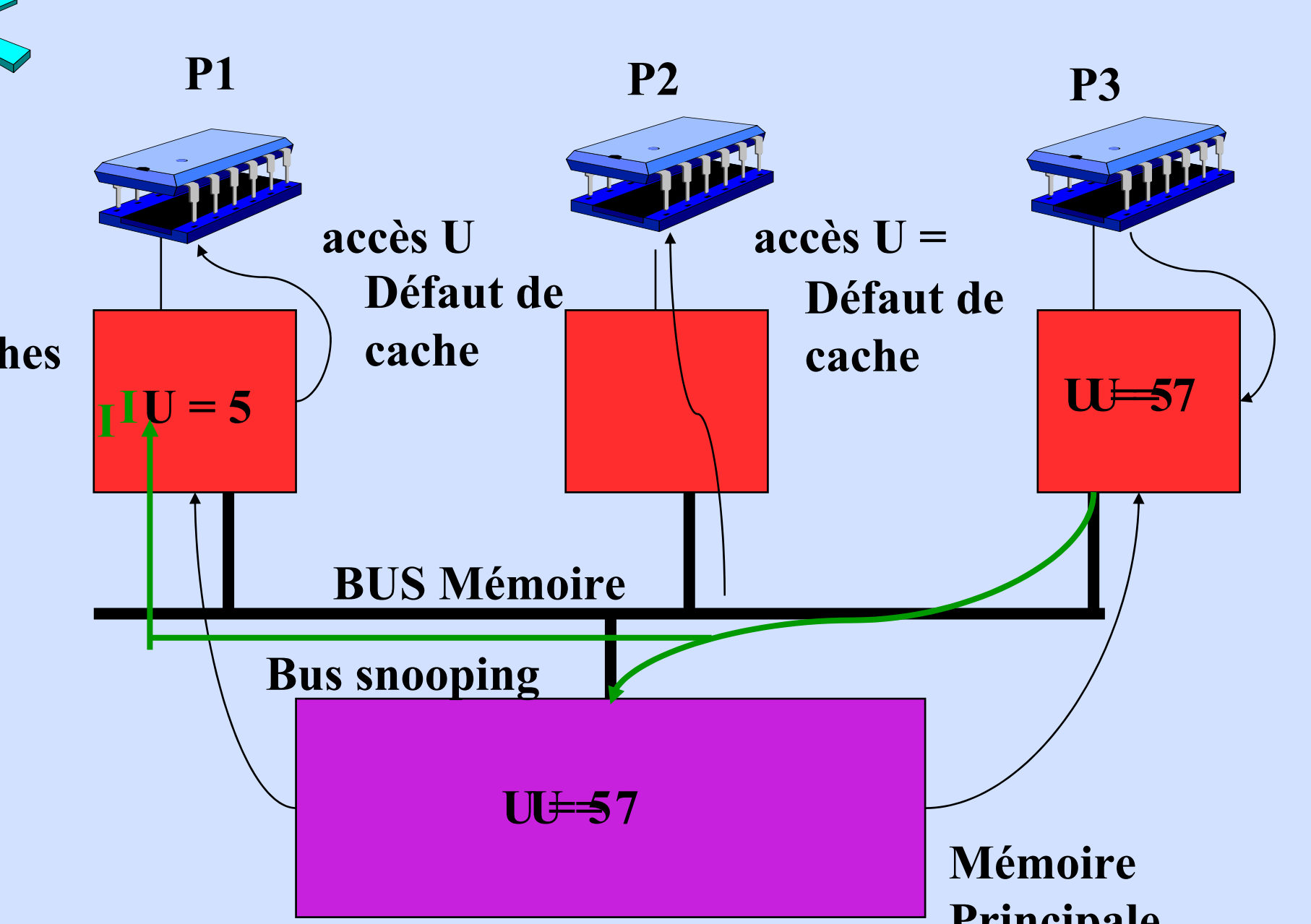

P1
P2
P3
U = ?
U = ?
U = 5
U = 7
BUS Mémoire
U = 5
Mémoire
Principale

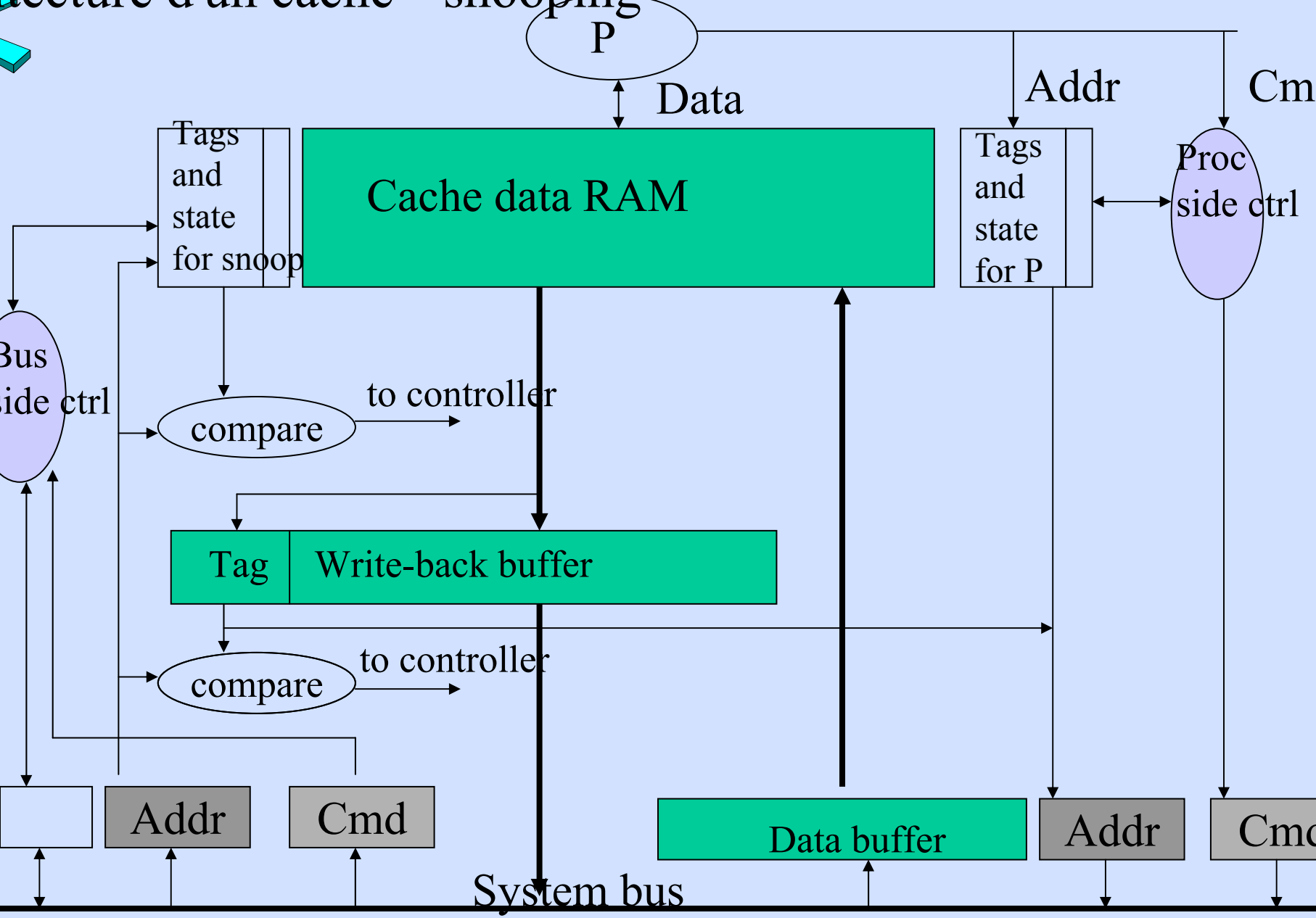
4) Solutions

- Bus snooping

Utilise la propriété inhérente à la structure de BUS : toutes les transactions passent par le BUS, une seule à la fois et tous les contrôleurs de cache ont accès à ce bus unique.

Tous les contrôleurs de cache espionnent le BUS et analysent les transactions : si l'adresse vue sur le BUS correspond à une adresse contenue dans une ligne de son cache alors il INVALIDE ou MET_A_JOUR la copie


P1
P2
P3
accès U
Défaut de
cache
accès U =
Défaut de
cache
hes
U = 5
U=57
BUS Mémoire
Bus snooping
U=57
Mémoire
Principale


P
Data
Addr
Cm
Tags and state for snoop
Cache data RAM
Tags and state for P
Proc side ctrl
Bus side ctrl
compare
to controller
Tag
Write-back buffer
compare
to controller
Addr
Cmd
Data buffer
Addr
Cmd
System bus

5) Analyse de performance

- Toutes les écritures mémoires font directement accès à la mémoire (Write_through) => consommation de la bande passante mémoire notamment en multiprocesseur :

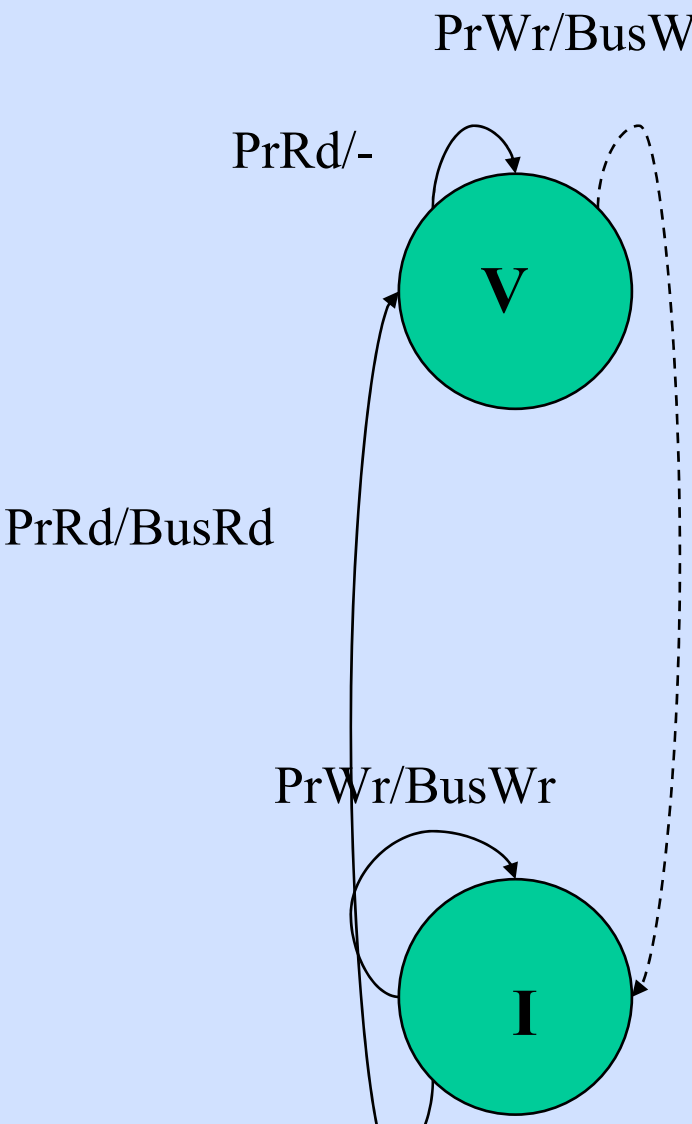
Exemple :

Soit un Proc à 1Ghz ; CPI =1 et 5% des instructions sont des stores 8 octets. Combien de procs saturent la bande passante mémoire pour un bus à 1,6 GB/s ? (ignorant une contention par ailleurs)

Nb écritures : 0,05 * 1 * 109 = 50 millions/s**
Bande passante : 8B*50M/s = 400 MB/s
Nb procs : 1600/400 = 4

Valeur pour un proc à 2Ghz ?

6) Invalidation-based Protocole pour un cache cohérent write-through



Transaction initiée par le Bus-Snop

Transaction initiée par le Pr

BusWr/-

A/B *A est observé, B est gé*

Exemple : un protocole d'invalidation à 3 cache write back.

M Modifié — Seulement ce cache a une copie valide, et la copie en mémoire n 'est pas à jour.

S Shared — Le bloc est présent dans le cache, à jour en mémoire et dans 0 ou N caches

I Invalide — Le bloc est invalide

PrWr/BusRdX

Sur une écriture d'un bloc PrWr qui passe de l'état I
l'état S vers l'état M, toutes les copies de ce bloc doi
être invalidées par une transaction bus RdX (copie
exclusive). Cette transaction ordonne les écritures
d'invalidation et donc la cohérence des caches

PrRd/- PrWr/-

Les lectures/écritures peuvent s'appliquer sur un blo
existe ou non dans le cache. Si le bloc n'existe pas, u
existant doit être remplacé et son contenu écrit en mé

PrRd/BusRd

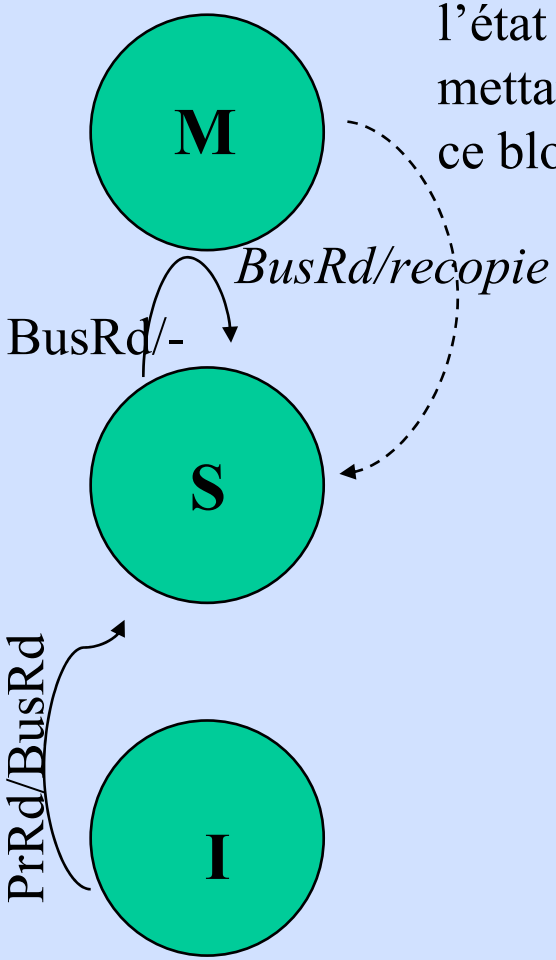
Un défaut de cache engendre un accès bus qui doit fo
la donnée. Celle-ci provient d 'un cache ou de la mém
et n'a pas a être modifiée

BusWr

Cette transaction est engendrée par un contrôleur de c
sur une écriture (Write Back). La mémoire est mise à
avec la valeur contenue dans le cache

Cas N°1

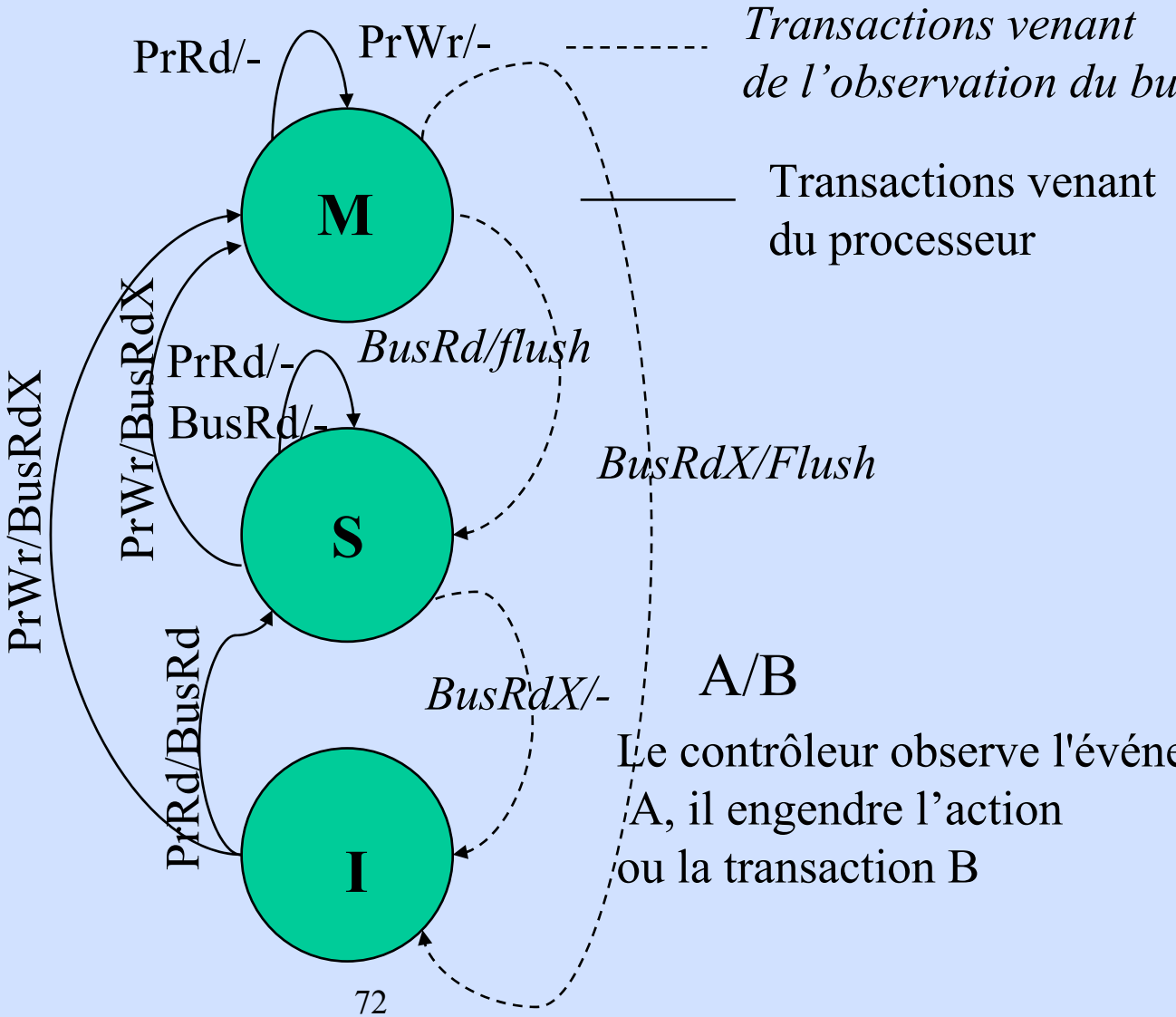
Si un cache possède ce bloc dan
l'état modifié, il écarte ce bloc e
mettant sur le bus, puis fait passe
ce bloc dans l'état partagé.



oc passe de I a S ;
s autres blocs dans
res caches dans l'état S
ent le Bus/Rd sans rien faire,
ttant à la mémoire de
re avec les données.

rocesseur lit un bloc invalide
non présent) dans le cache
ui provoque un BusRd.

Généralisation (protocole MSI)



Modifié

hared

nvalide

Le protocole MSI sur l'exemple (T7)

Action du processeur	Etat de P1	Etat de P2	Etat de P3	Action sur le bus	Données transmises
P1 lit U	S	_	_	BusRD	Par Mémoire
P3 lit U	S	_	S	BusRD	Par Mémoire
P3 écrit U	I	_	M	BusRDX	A la mémoire
P1 lit U	S	_	S	BusRD	par Cache P3
P2 lit U	S	S	S	BusRD	par Mémoire