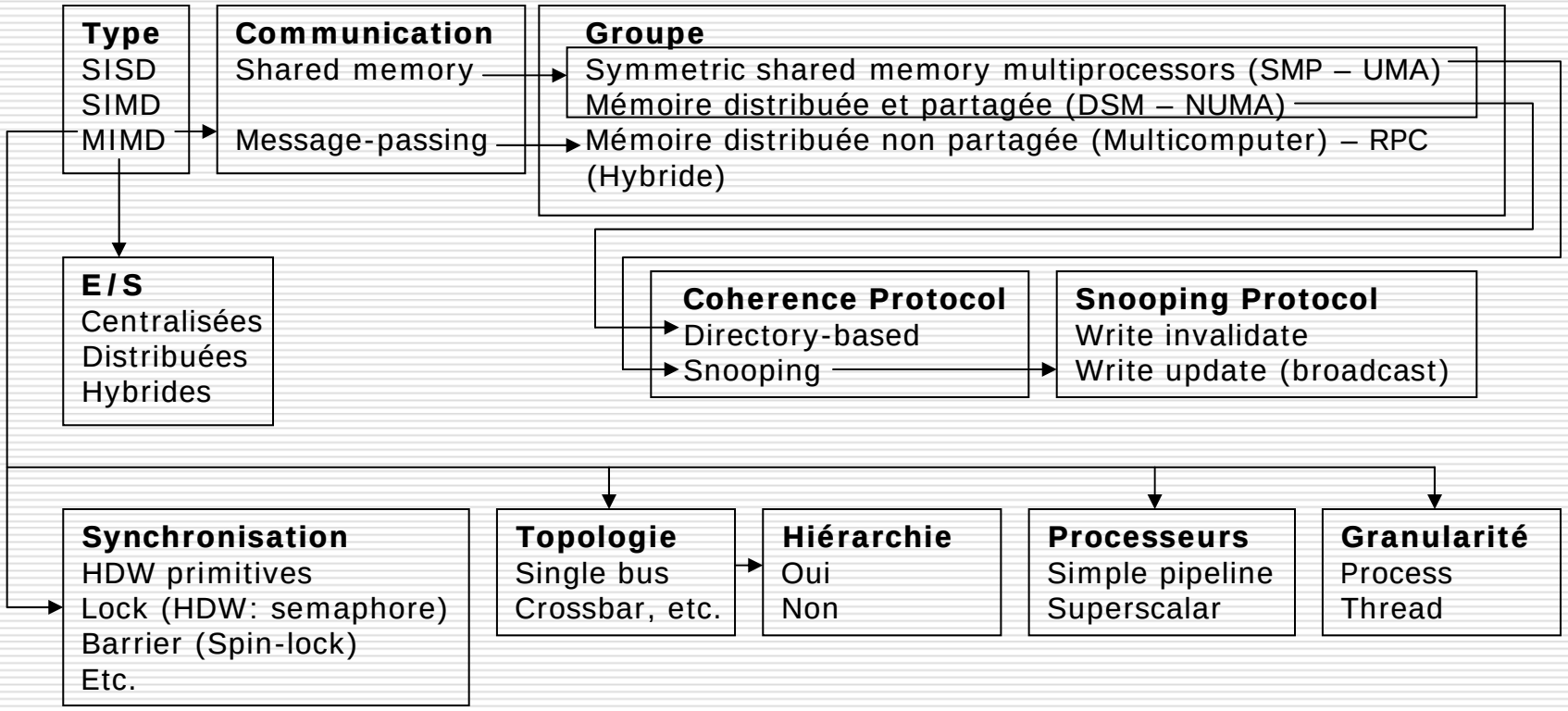
# INF6500 : Structures des ordinateurs

# Cours 4 : Multiprocesseurs

## Multiprocesseurs



# Performance Metrics pour communication entre plusieurs processeurs

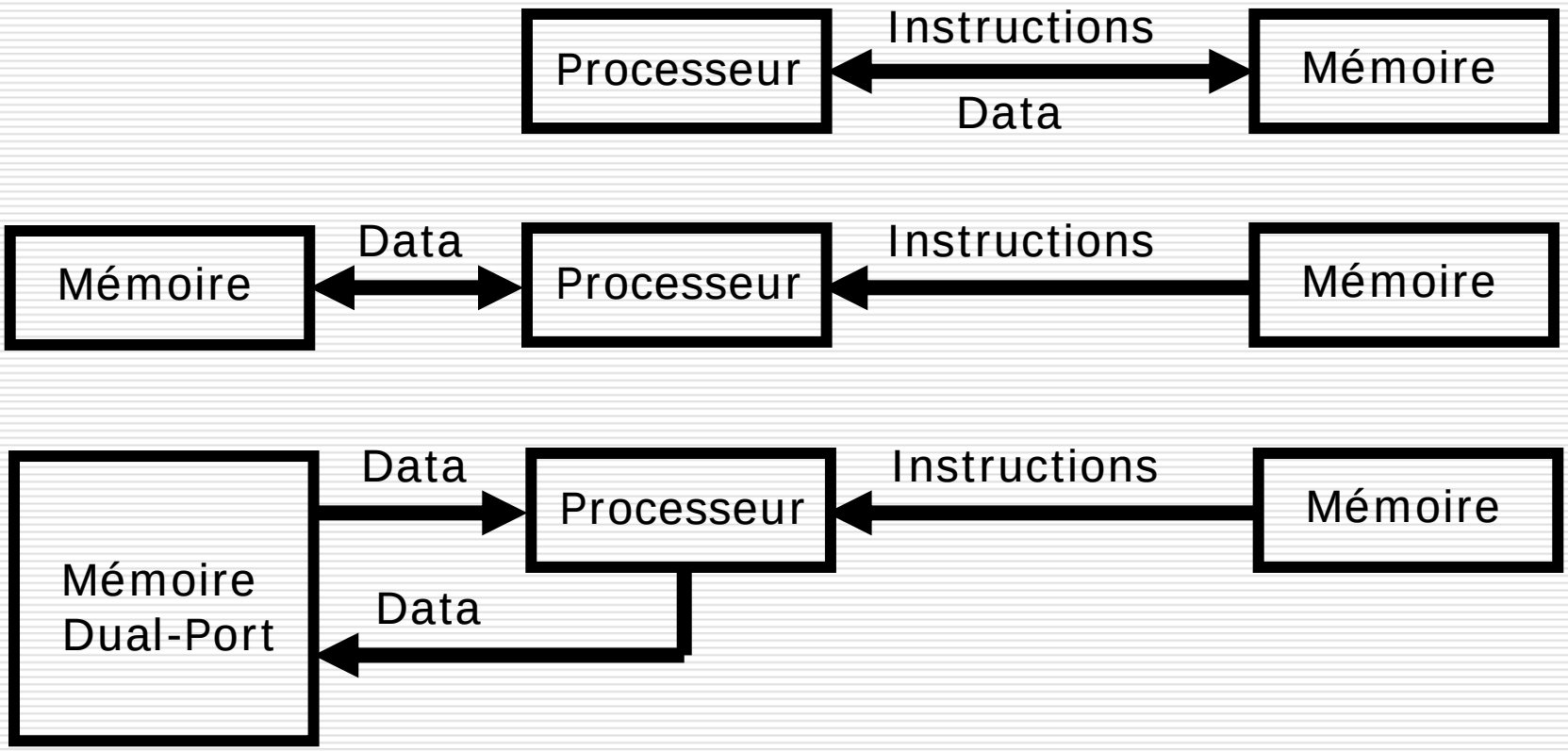
- Communication bandwidth
  - Limitée par le processeur, interconnections, au lieu du mécanisme de communication
- Communication latency
  - Idéalement le plus bas possible, (sender overhead, delais relié au lien de communication (FIFO), receiver overhead)
- Communication latency hiding
  - Latence de communication en parallèle avec computation

- Définitions: Bandwidth (bande passante), Throughput (Débit), Latency (Latence)

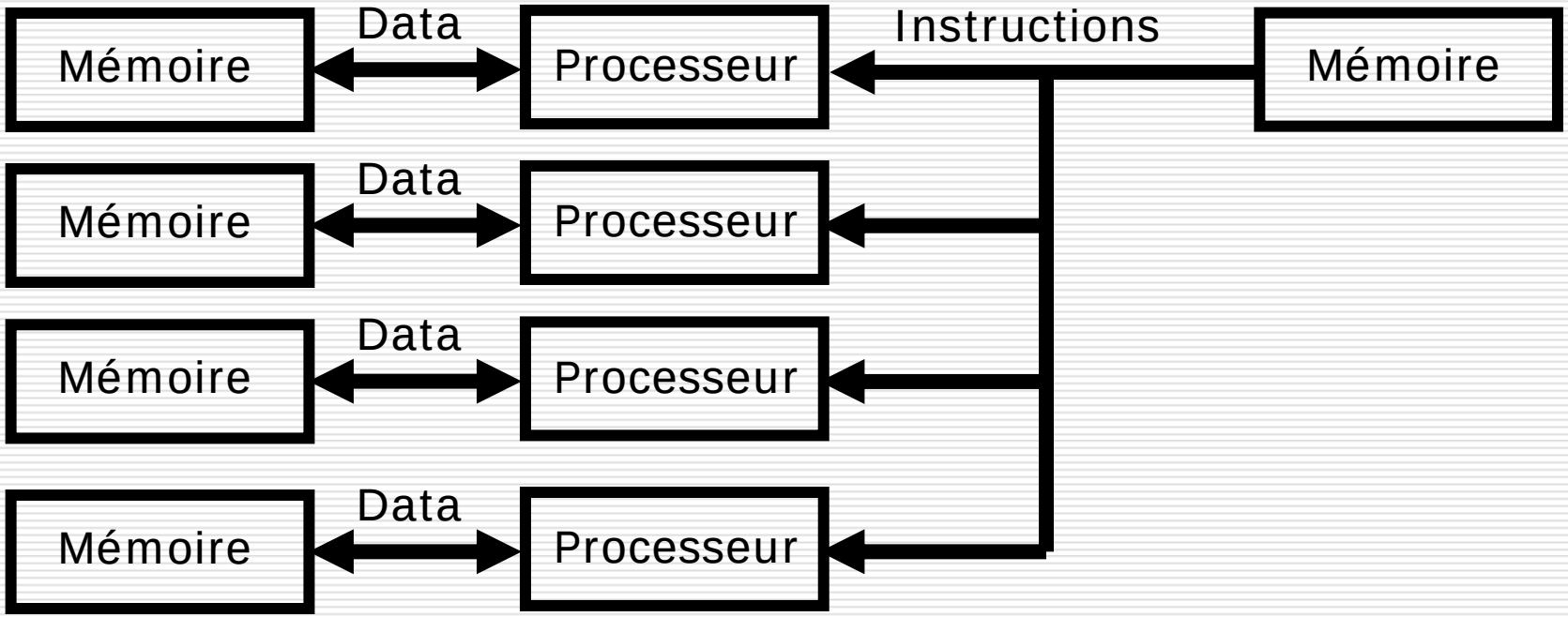
# Taxonomie des architectures parallèles

- Un seul flot d'instruction, un seul flot de données (*single instruction, single datum* - SISD) : c'est le monoprocesseur commun.
- Un seul flot d'instructions, plusieurs flots de données (*single instructions, multiple data* - SIMD): La même instruction est exécutée par plusieurs processeurs utilisant différents flots de données. Chaque processeur a sa propre mémoire données, mais il y a une seule mémoire d'instructions et un seul processeur de contrôle qui lance les instructions.
- Plusieurs flots d'instructions, un seul flot de données (*multiple instructions, single datum* - MISD): aucune machine commerciale de ce type n'a été construite à ce jour.
- Plusieurs flots d'instructions, plusieurs flots de données (*multiple instructions, multiple data* – MIMD): Chaque processeur lit ses propres instructions et opère sur ses propres données.

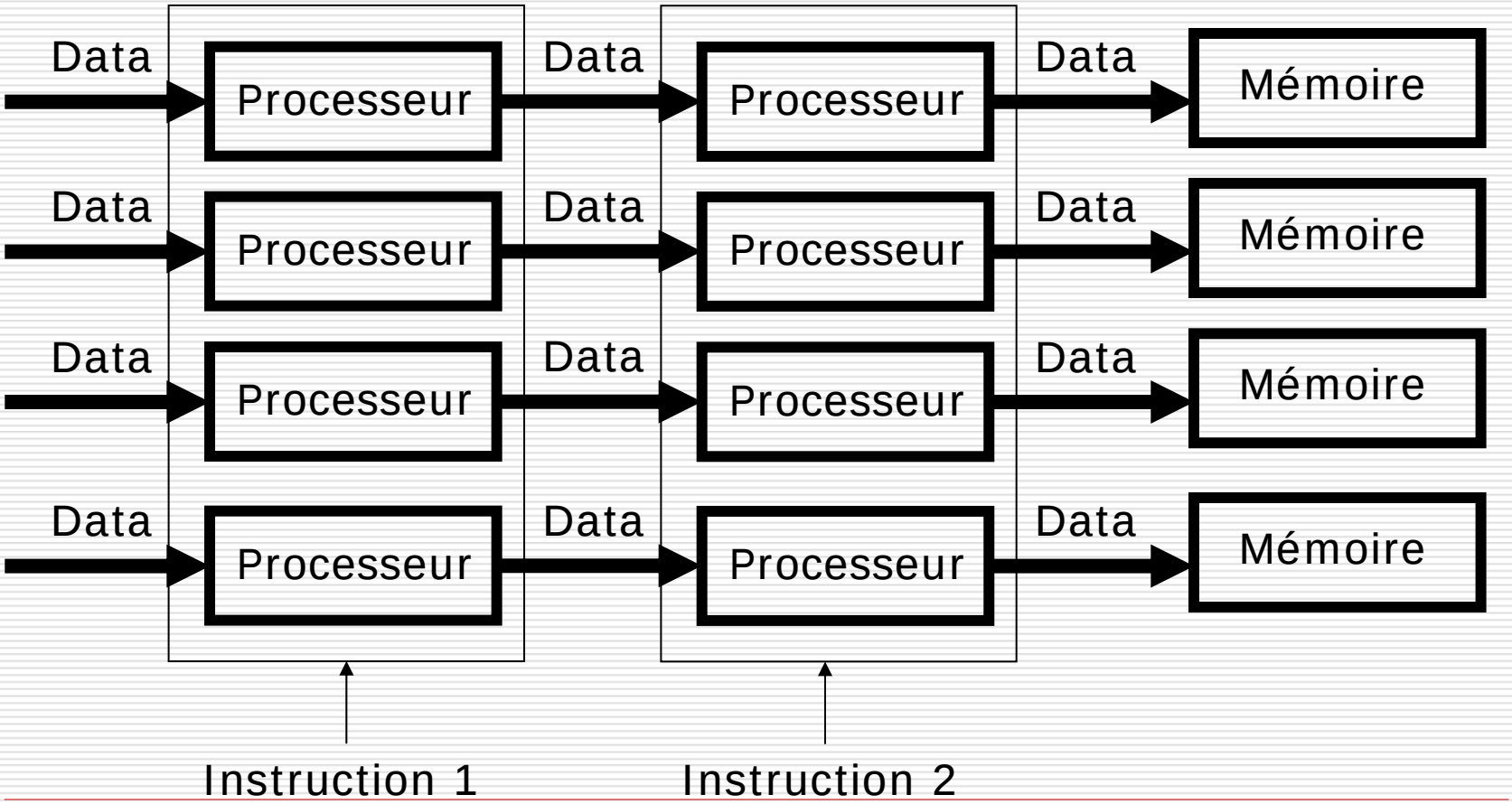
# SISD (Exemples)



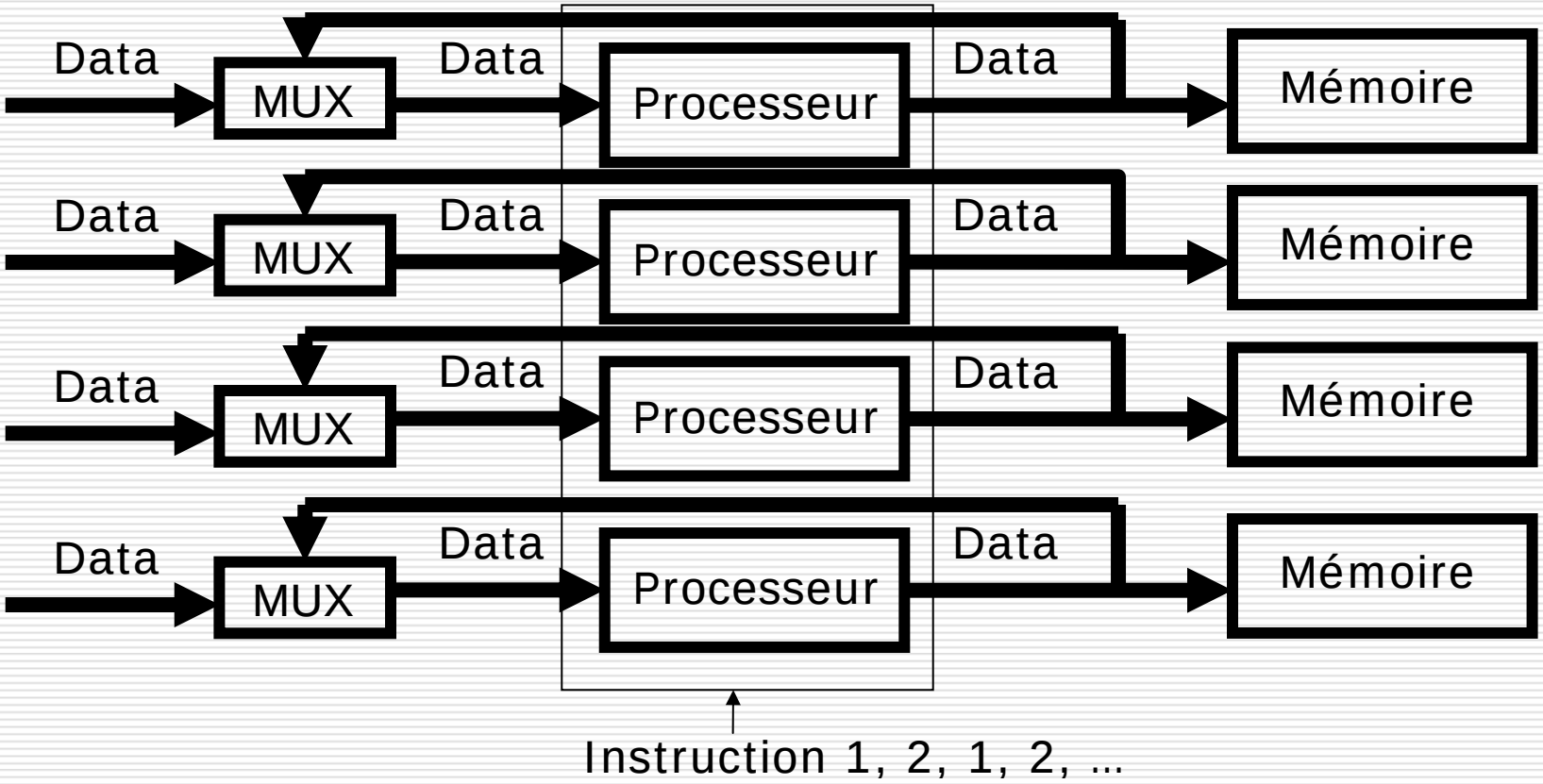
# SIMD (Exemple)



# SIMD (Exemple - suite)



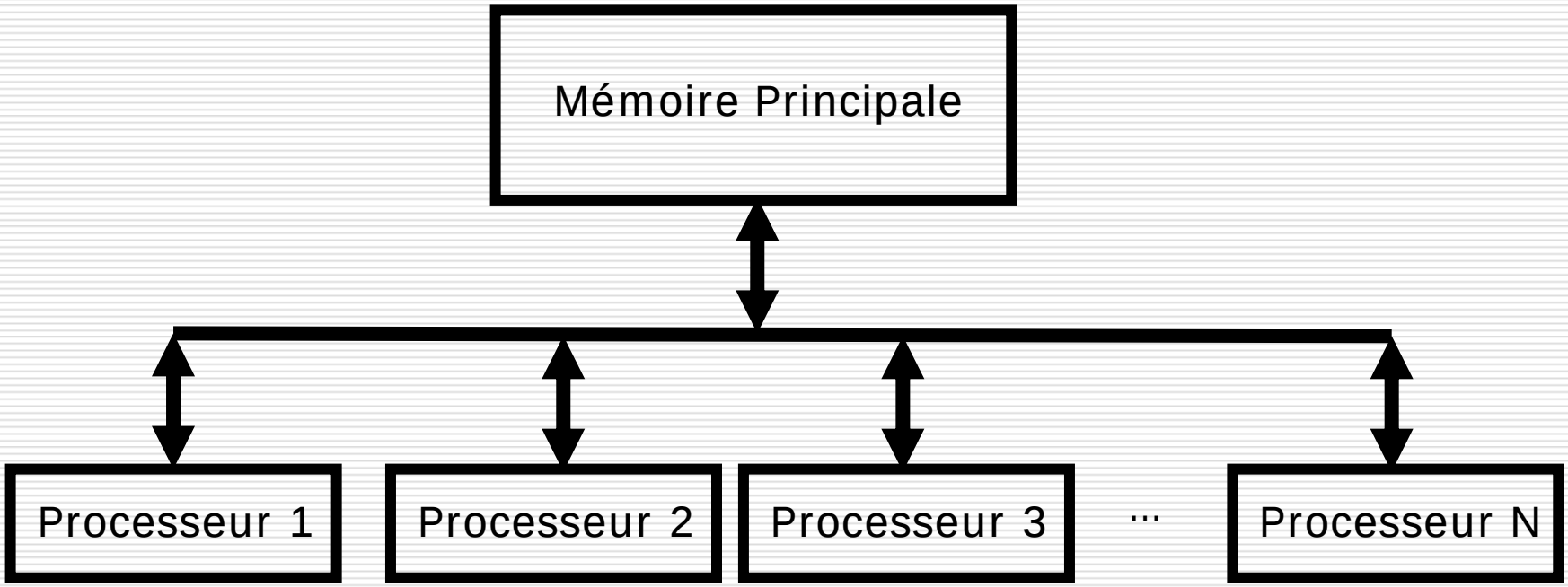
# SIMD (Exemple - suite)



# MIMD avec mémoire partagée ou mémoire distribuée

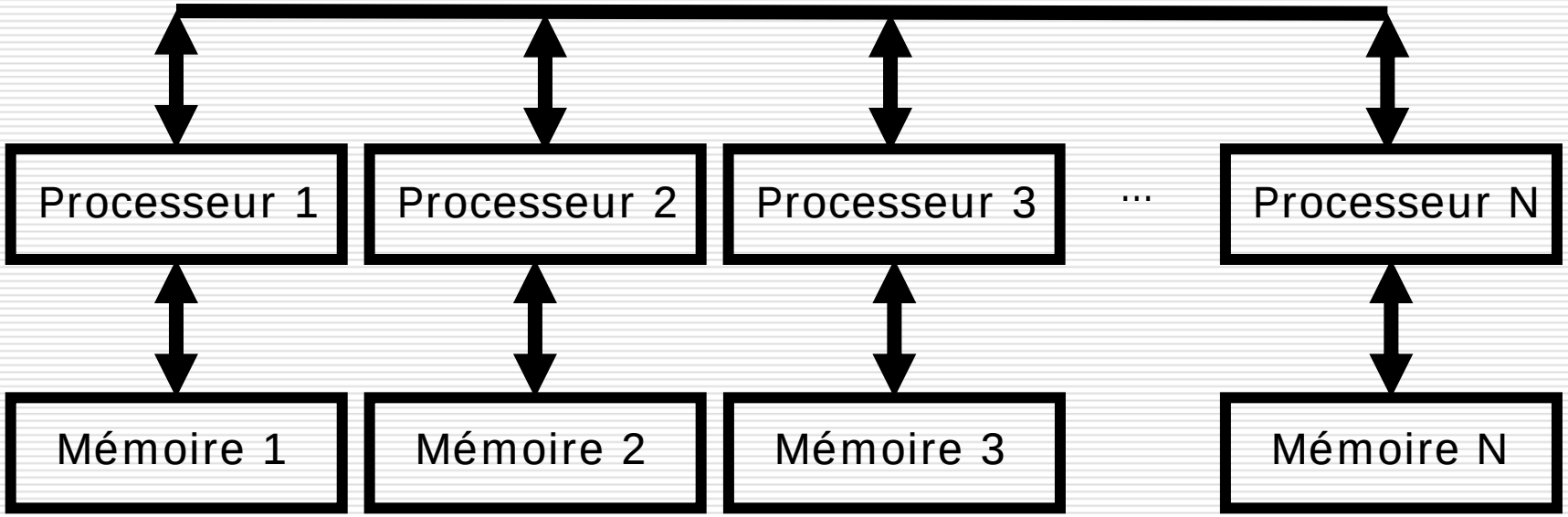
- Les premiers multiprocesseurs étaient SIMD. De nos jours, le modèle MIMD a surgit comme l'architecture évidente à choisir pour le multiprocesseurs d'usage général:
  - Flexible : peut fonctionner comme une machine simple utilisateur destinée à la haute performance pour une application, comme une machine avec multiprogrammation exécutant beaucoup de tâches simultanément, ou selon une certaine combinaison de ces fonctions.
  - Un MIMD peut être construit en s'appuyant sur les avantages coût performance des microprocesseurs standard.
- Les machine MIMD actuelles sont classables en deux groupes, selon le nombre de processeurs.
  - Architectures à mémoire partagée centralisée
    - Au plus quelques douzaines de processeurs en1990
    - Se partagent la même mémoire physique
    - La connexion est typiquement réalisée par un bus
    - Temps d'accès uniforme pour chaque processeur

# MIMD avec mémoire partagée (centralisée) – *Multi-thread Oriented*




Inter-processors: Shared Memory Model

# MIMD avec mémoires distribuées – *Multi-process Oriented*

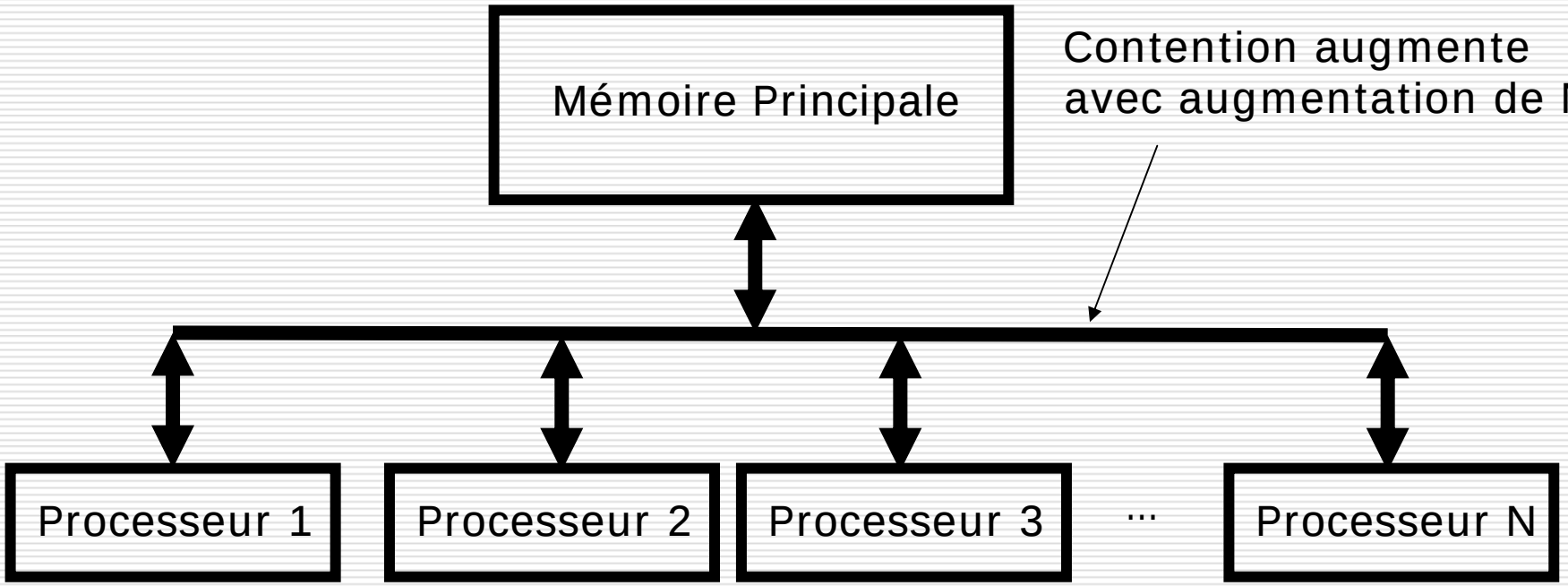



Inter-processors: Message Passing Model

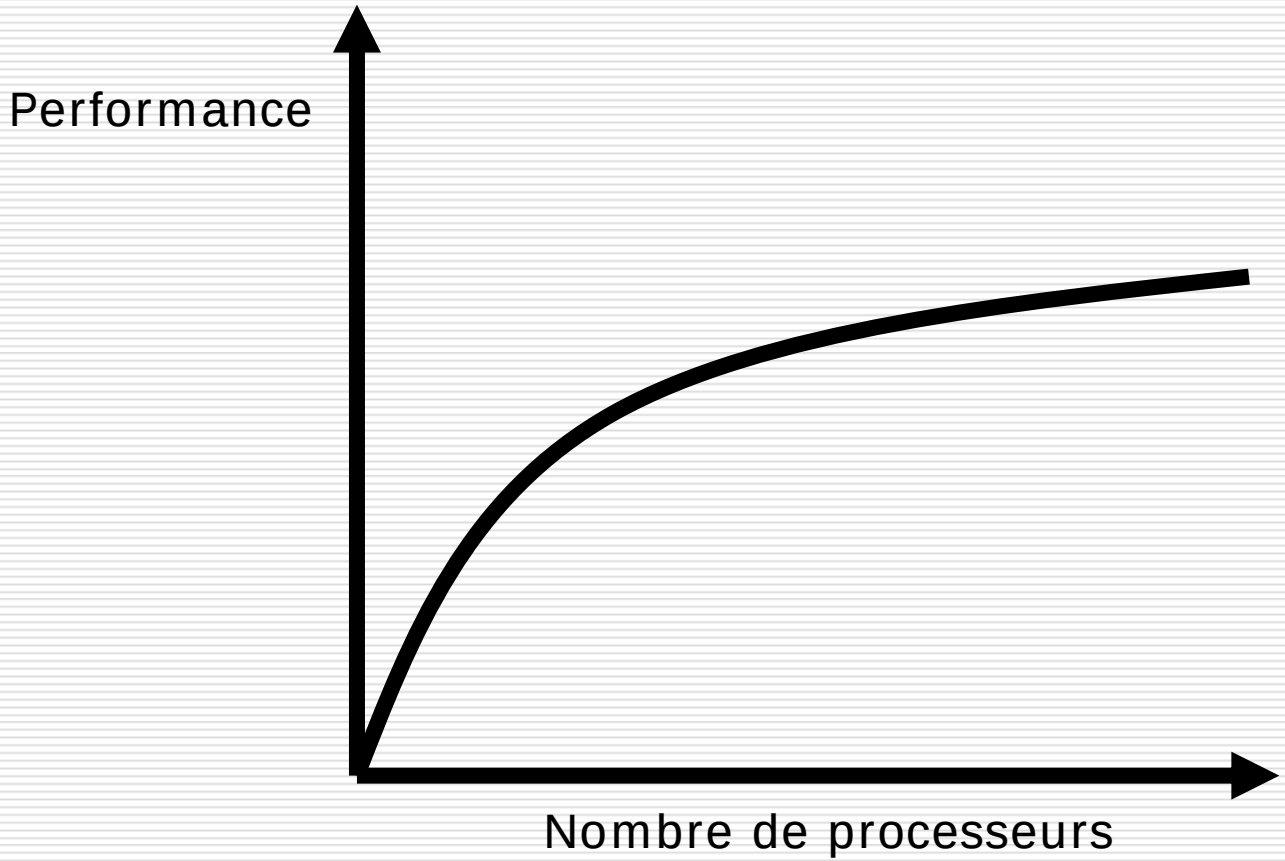
# Processes vs. threads

- Process:
  - Segment de codes qui peut-être exécuté indépendamment par un processeur. Chaque « process » est indépendant des « processes » sur les autres processeurs
- Thread:
  - Lorsque plusieurs « processes » partagent le code et la plupart de leurs adresses mémoires, ils sont référés comme des « threads ».

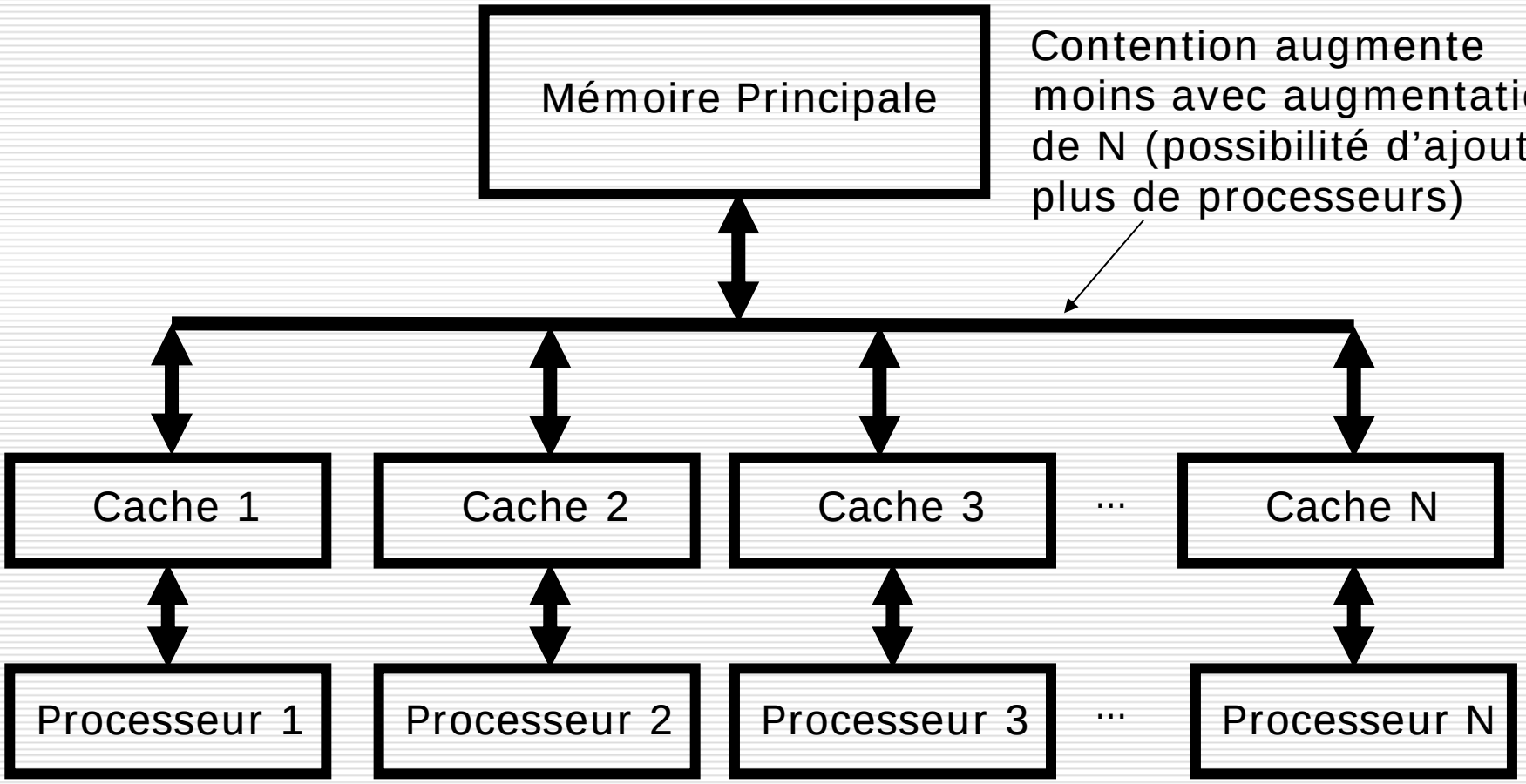
# MIMD avec mémoire partagée centralisée Symmetric **Shared Memory** Multiprocessors (SMP) ou UMA (uniform memory access) – Contention sur le bus



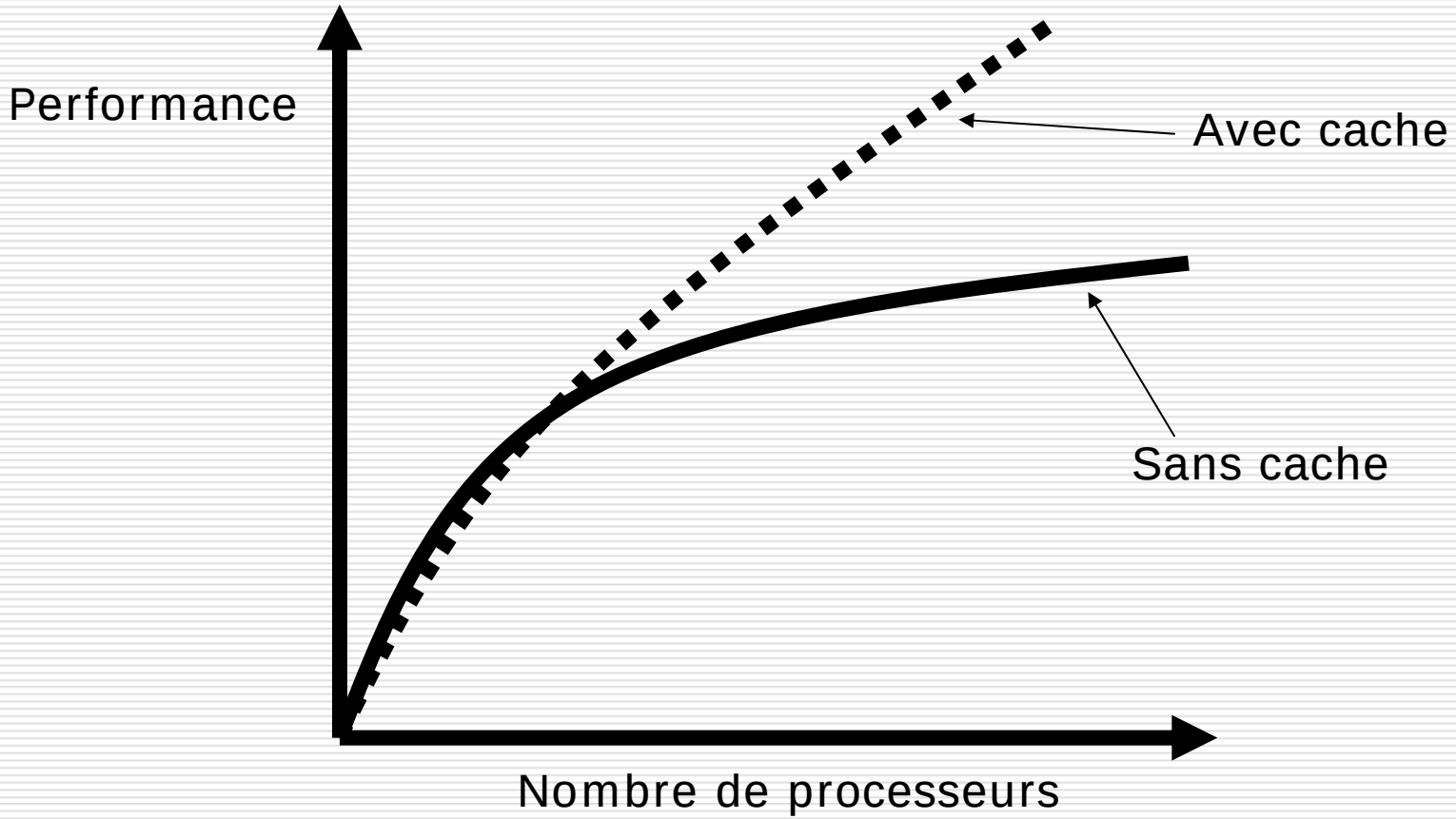
# Réduction de la performance à cause du niveau de contentions



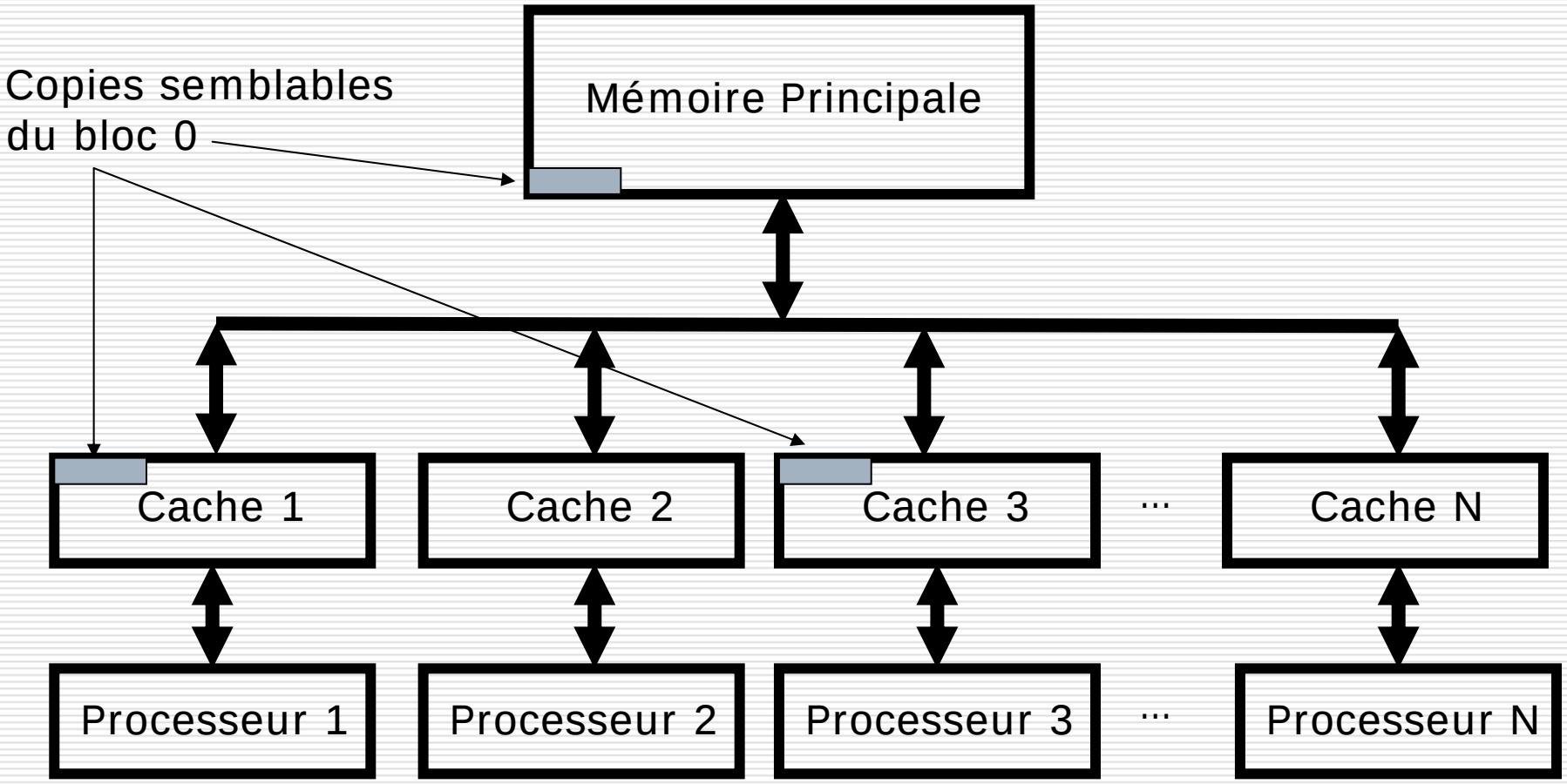
# MIMD avec mémoire partagée centralisée – Réduction du niveau de contention sur le bus avec mémoires caches



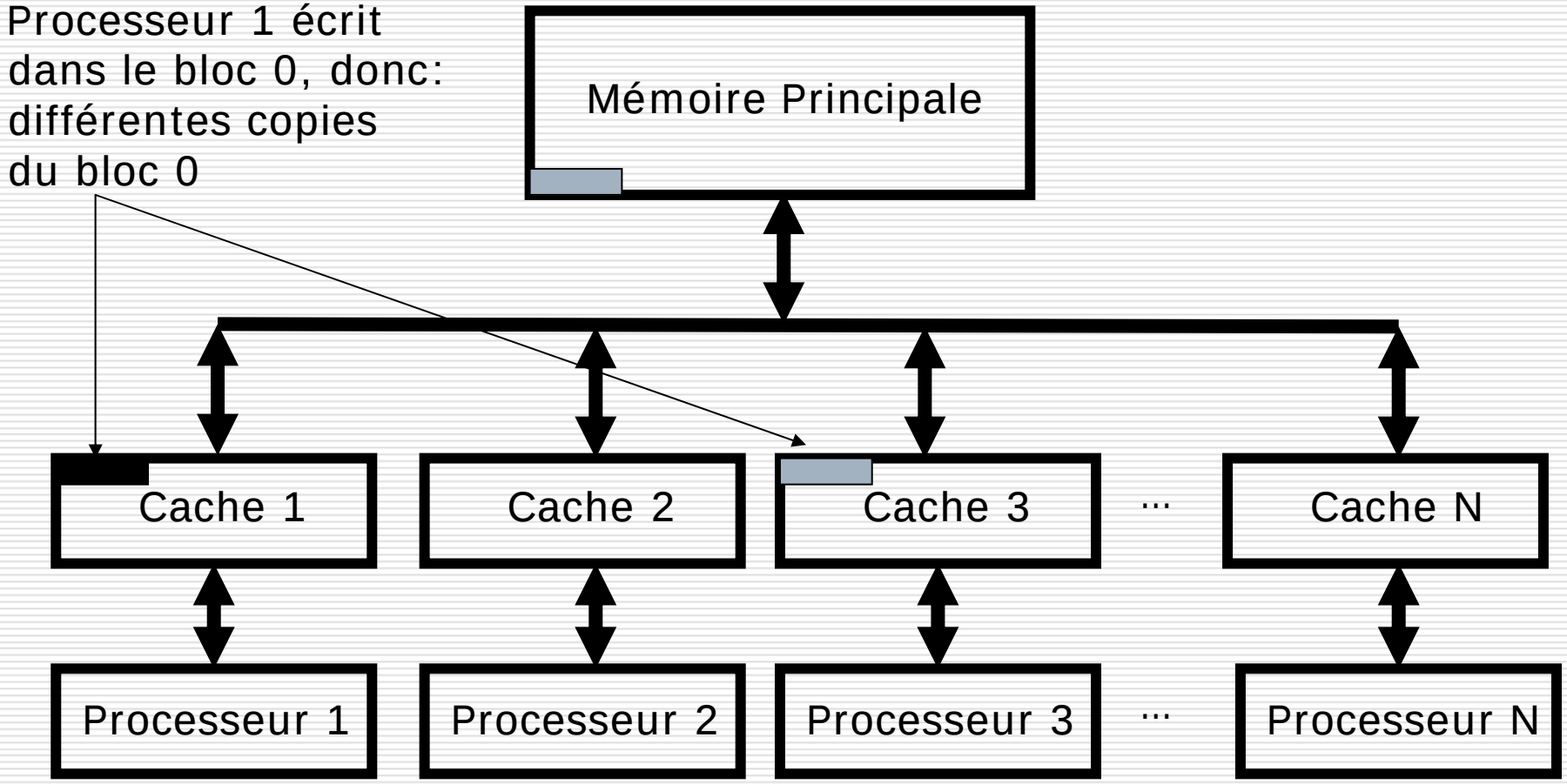
# Réduction de la performance à cause du niveau de contentions



# MIMD avec mémoire partagée centralisée – Problèmes de cohérence entre les mémoires caches (cache coherence)



# MIMD avec mémoire partagée centralisée – Problèmes de cohérence entre les mémoires caches (cache coherence) - Suite



# Protocoles de cohérence de caches (cache coherence protocols)

Snooping Protocols
*(Shared Memory)*

- Write Invalidate Protocol
- Write Update ou Write Broadcast Protocol

Directory-based Protocols
*(Distributed Memory)*

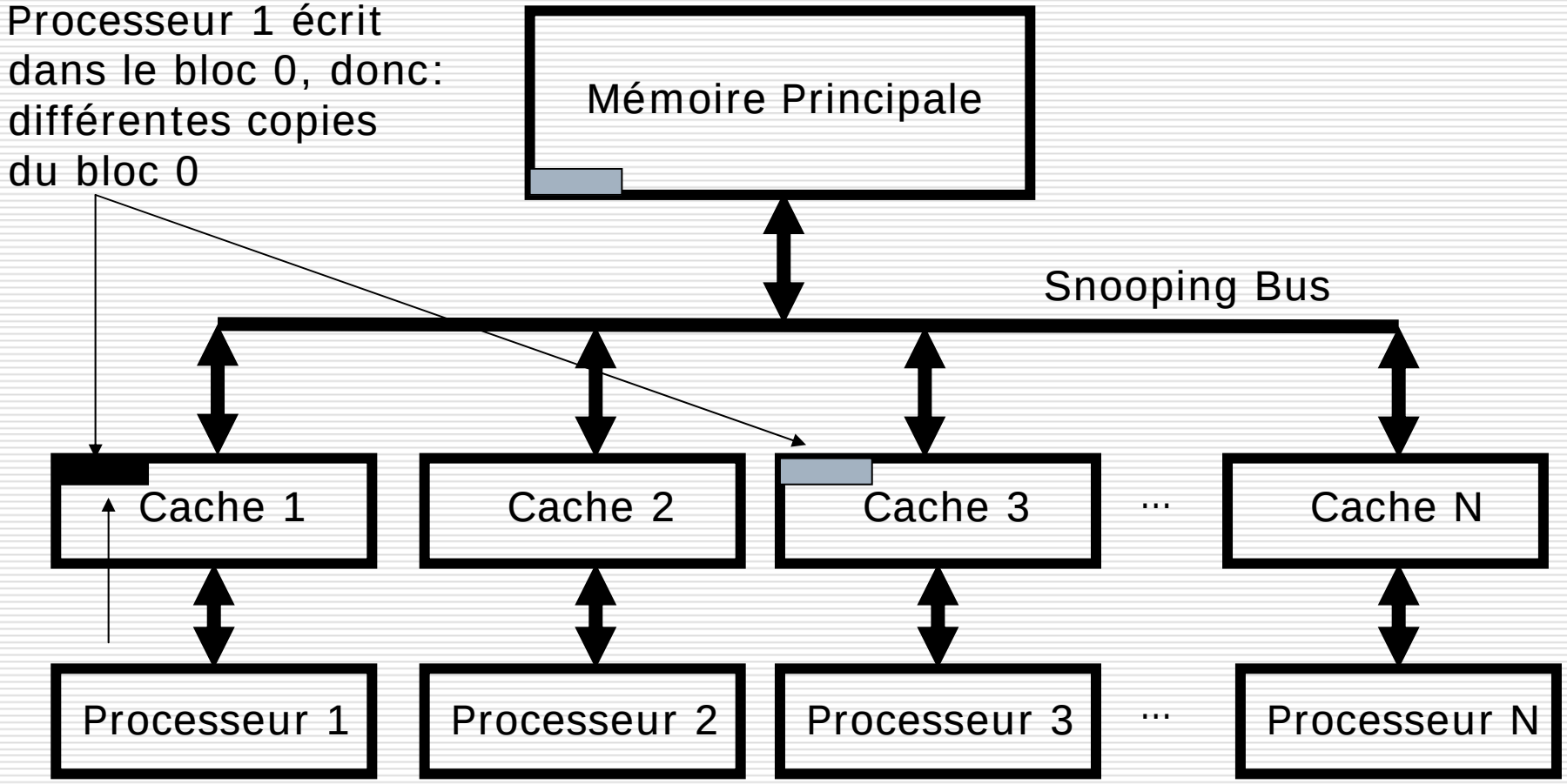
# Protocoles de cohérence de caches (cache coherence protocols)

Snooping Protocols
*(Centralized (single) Shared Memory)*

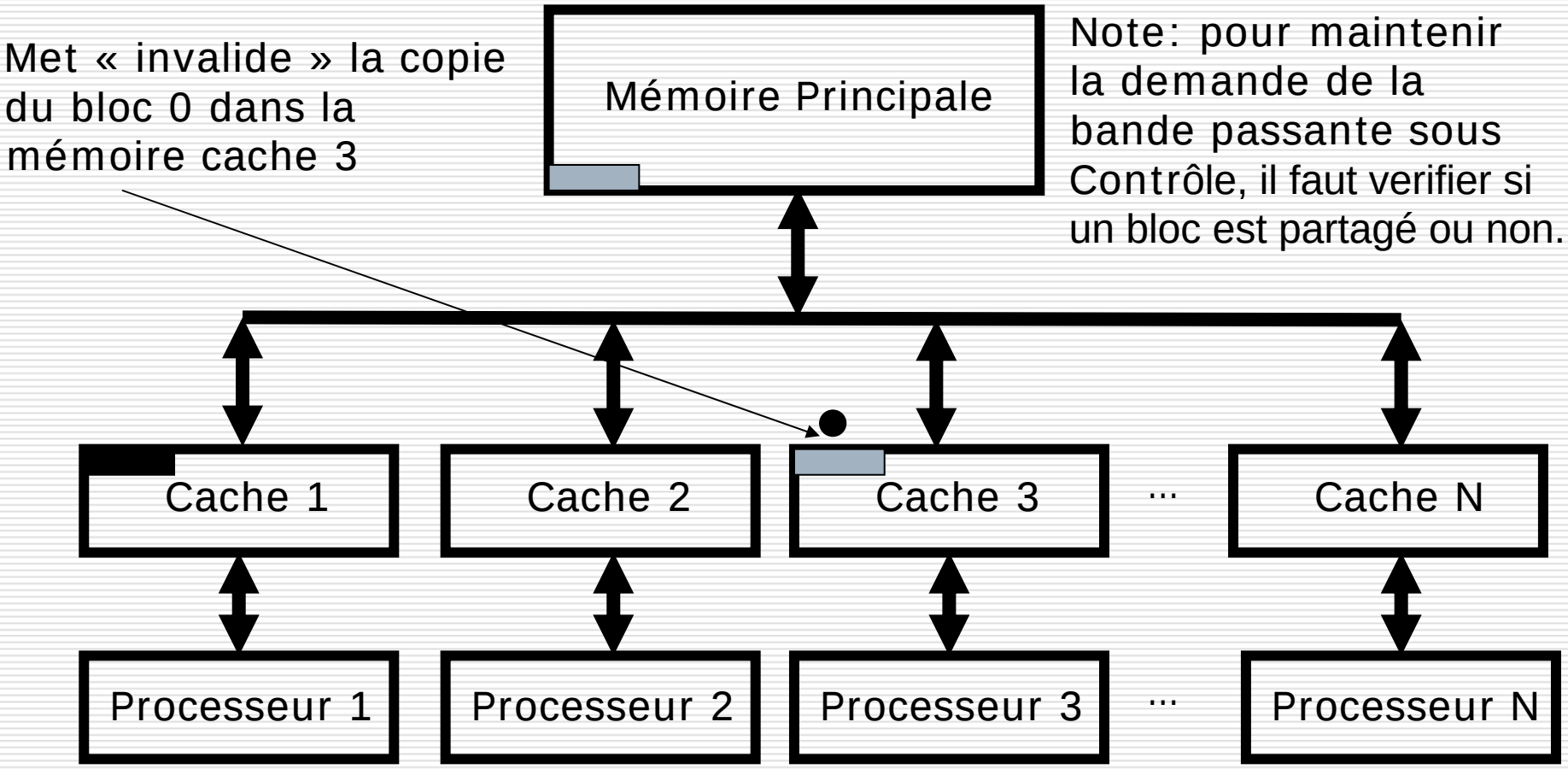
- **Write Invalidate Protocol**
- Write Update ou Write Broadcast Protocol

Directory-based Protocols
*(Distributed Shared Memory)*

# MIMD avec mémoire partagée centralisée – Write Invalidate – T1



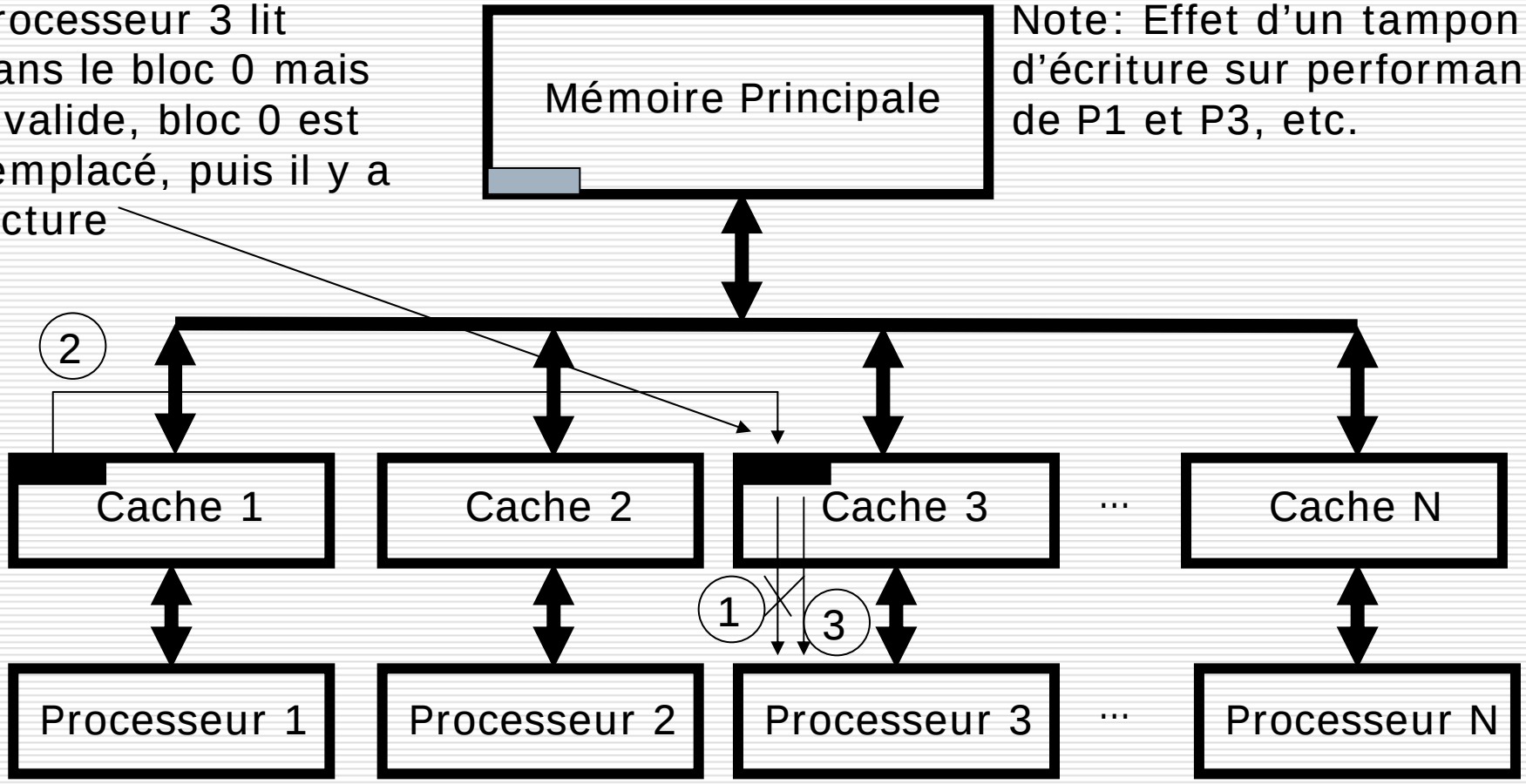
# MIMD avec mémoire partagée centralisée – Write Invalidate – T2



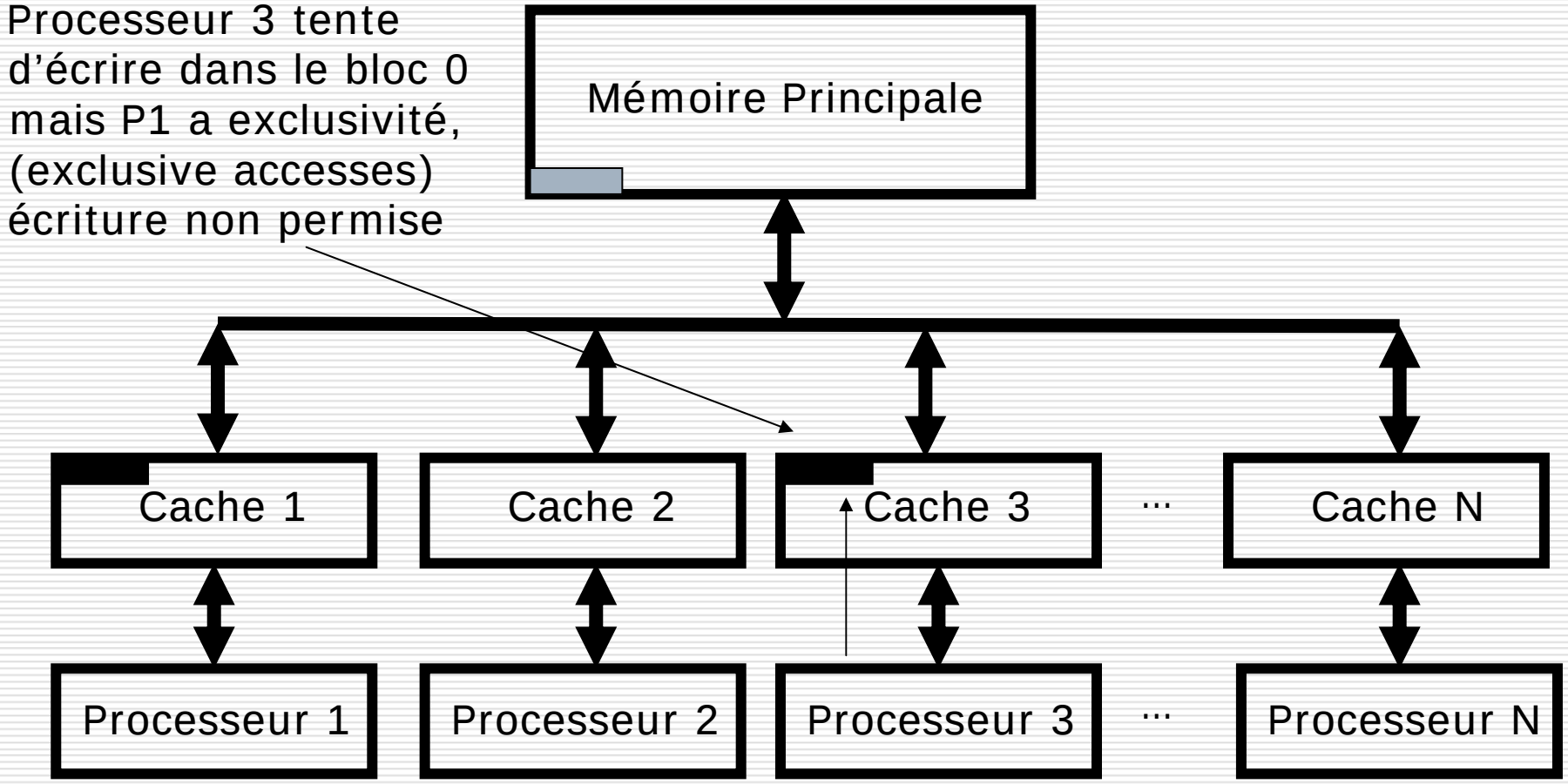
# MIMD avec mémoire partagée centralisée – Write Invalidate – T3

Processeur 3 lit dans le bloc 0 mais invalide, bloc 0 est remplacé, puis il y a lecture

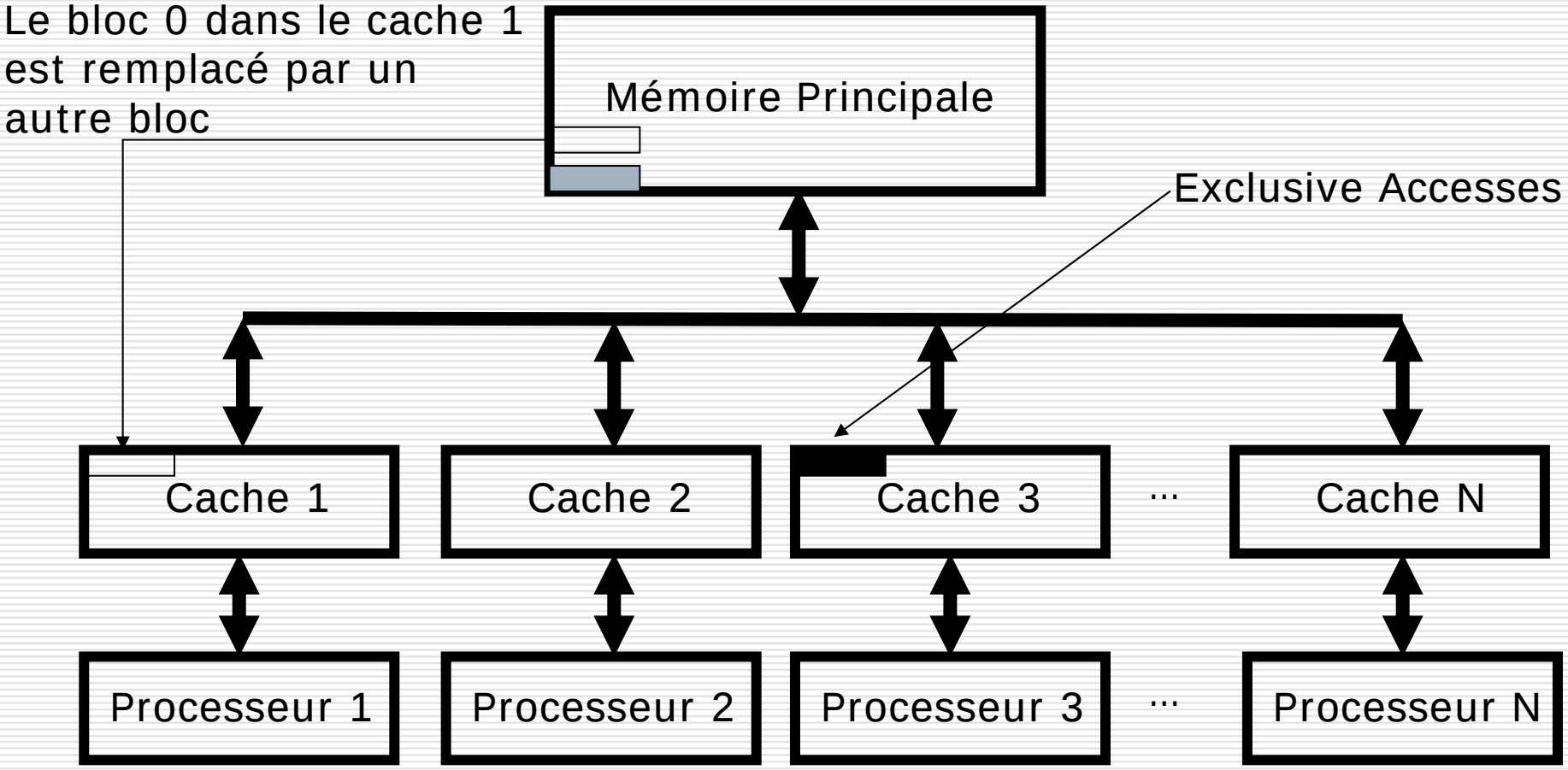
Note: Effet d'un tampon d'écriture sur performance de P1 et P3, etc.



# MIMD avec mémoire partagée centralisée – Write Invalidate – T4

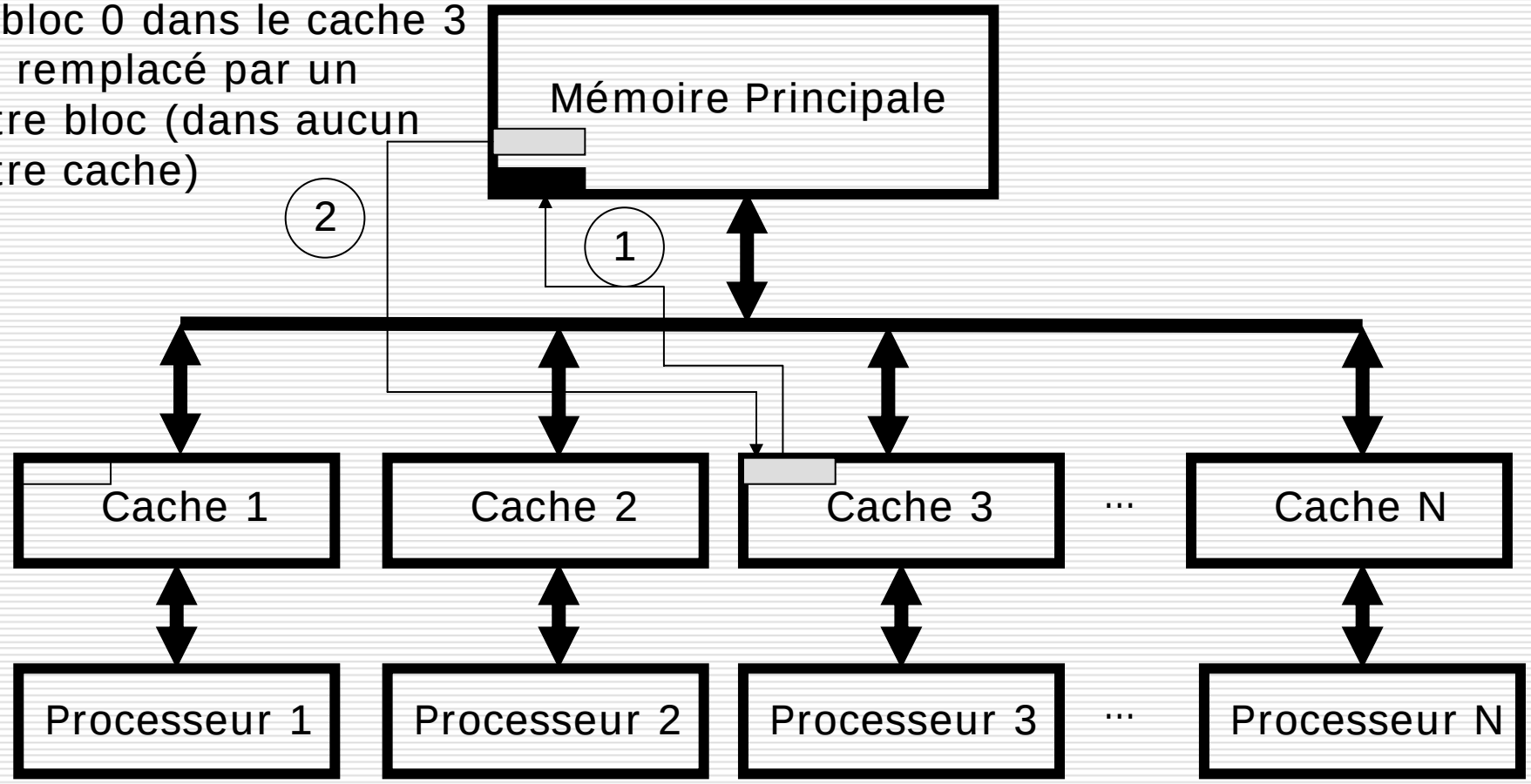


# MIMD avec mémoire partagée centralisée – Write Invalidate – T5



# MIMD avec mémoire partagée centralisée – Write Invalidate – T6

Le bloc 0 dans le cache 3 est remplacé par un autre bloc (dans aucun autre cache)



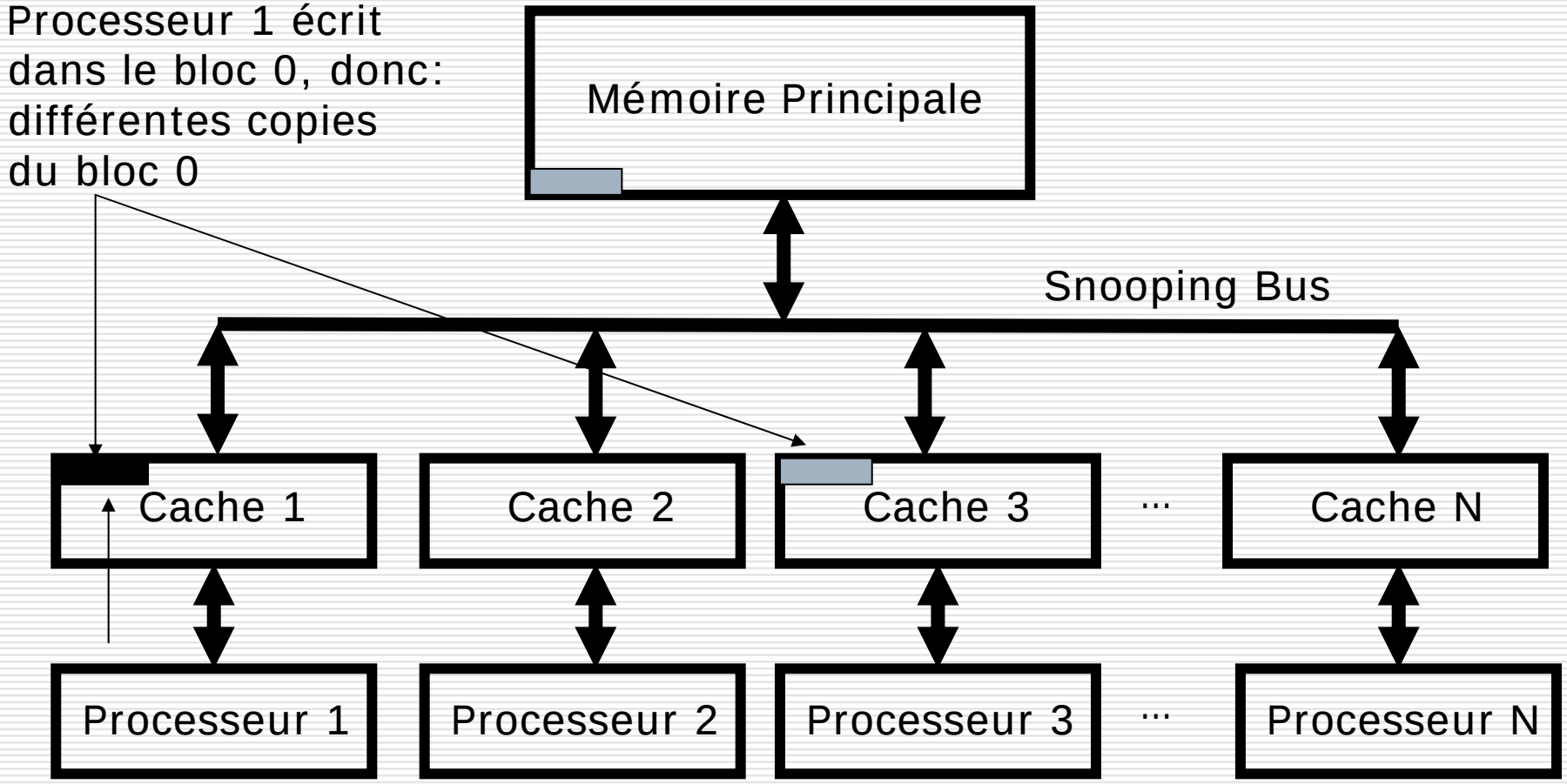
# Protocoles de cohérence de caches (cache coherence protocols)

Snooping Protocols
*(Centralized (single) Shared Memory)*

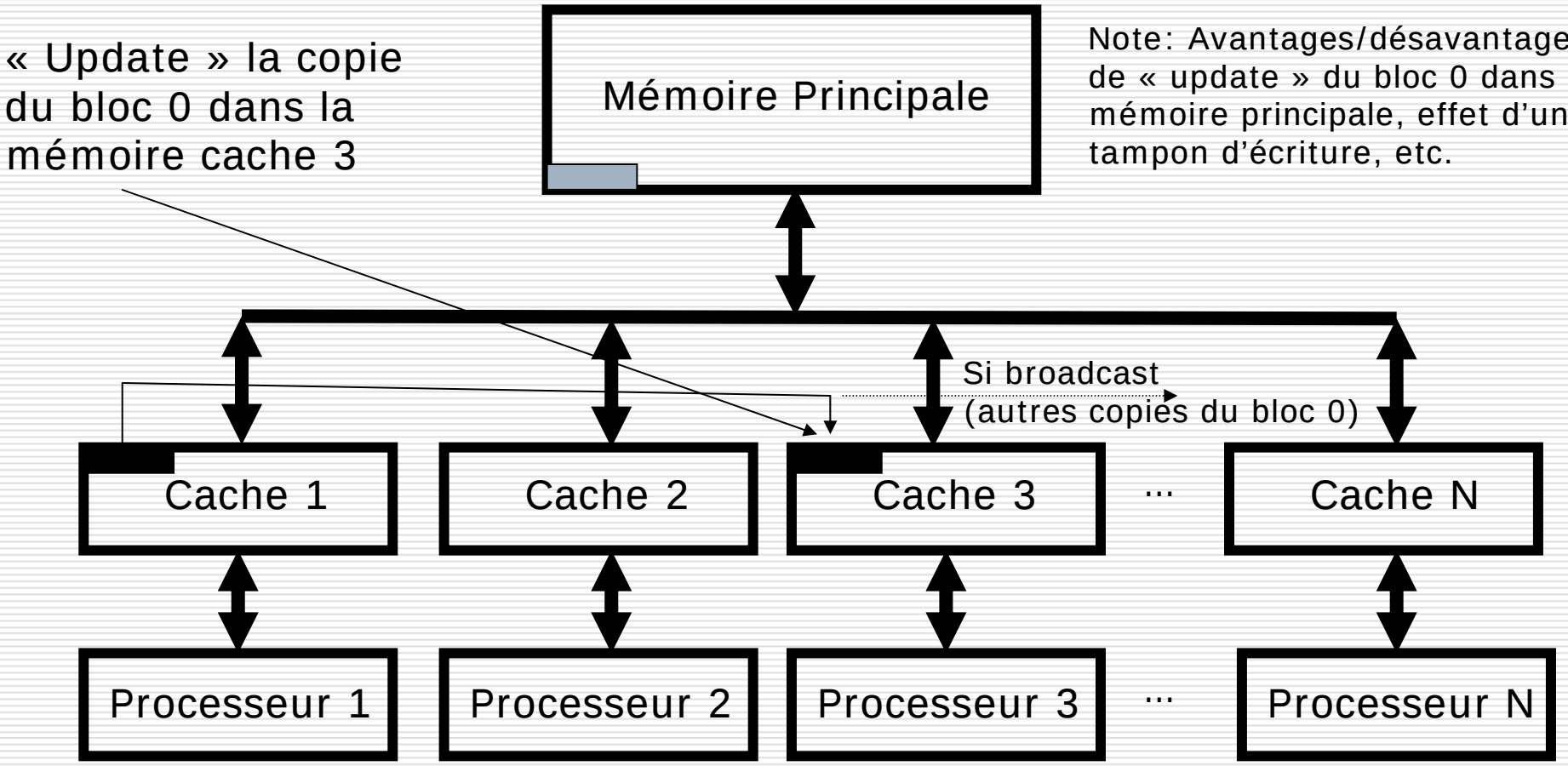
- Write Invalidate Protocol
- **Write Update ou Write Broadcast Protocol**

Directory-based Protocols
*(Distributed Shared Memory)*

# MIMD avec mémoire partagée centralisée – Write Update or Broadcast – T1

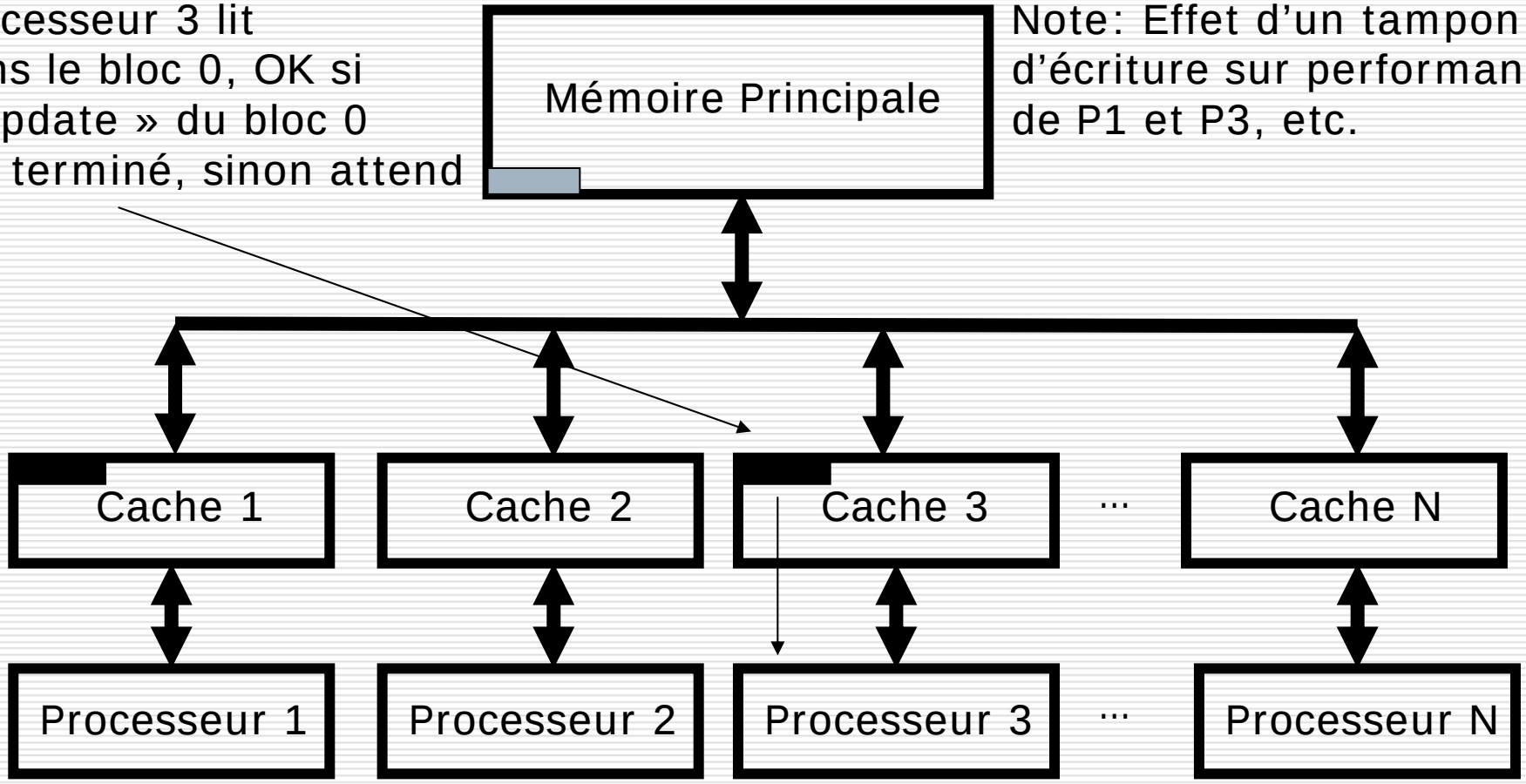


# MIMD avec mémoire partagée centralisée – Write Update or Broadcast – T2



# MIMD avec mémoire partagée centralisée – Write Invalidate – T3

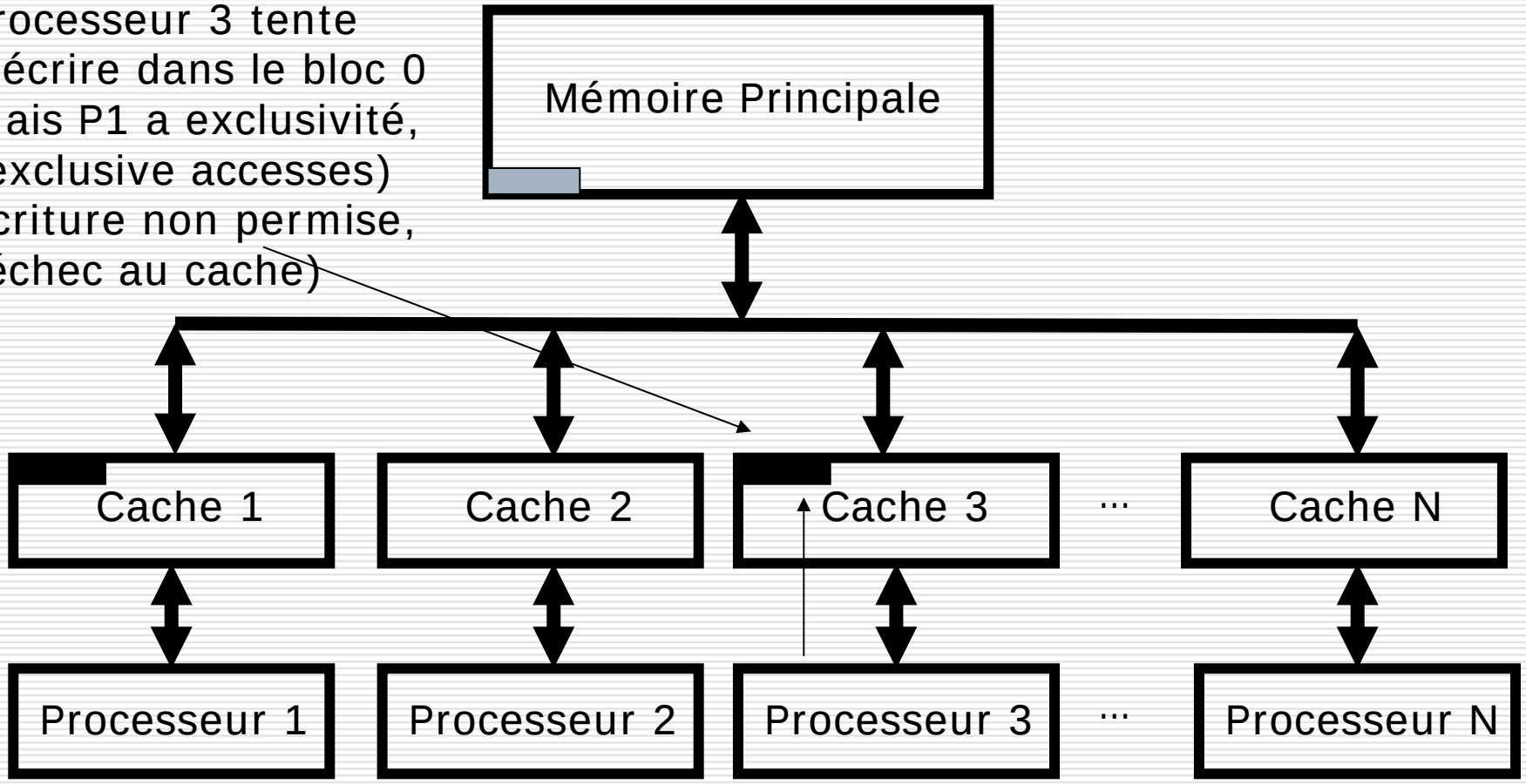
Processeur 3 lit dans le bloc 0, OK si « update » du bloc 0 est terminé, sinon attend



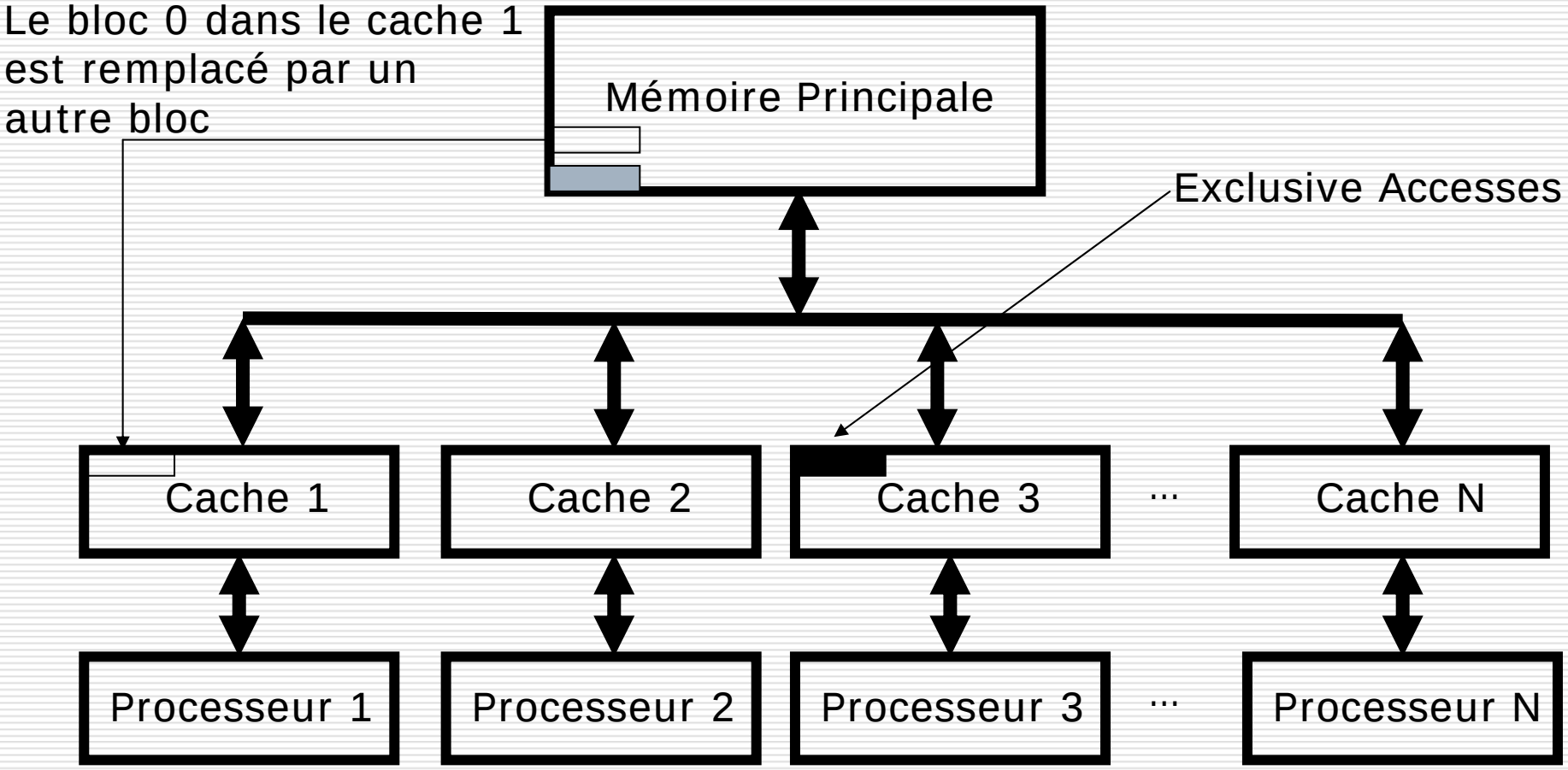

Note: Effet d'un tampon d'écriture sur performance de P1 et P3, etc.

# MIMD avec mémoire partagée centralisée – Write Update or Broadcast – T4

Processeur 3 tente d'écrire dans le bloc 0 mais P1 a exclusivité, (exclusive accesses) écriture non permise, (échec au cache)

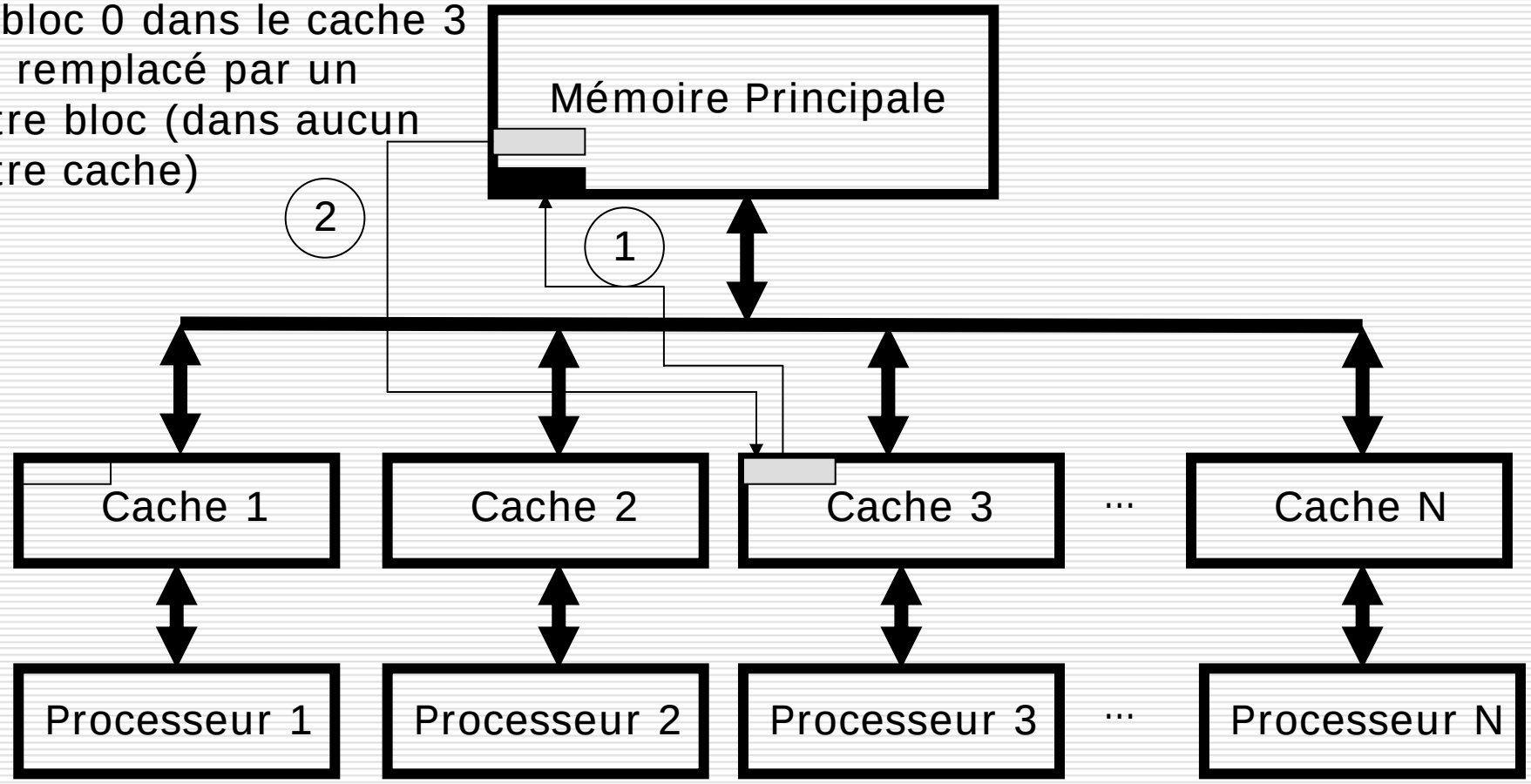


# MIMD avec mémoire partagée centralisée – Write Update or Broadcast – T5



# MIMD avec mémoire partagée centralisée – Write Update or Broadcast – T6

Le bloc 0 dans le cache 3 est remplacé par un autre bloc (dans aucun autre cache)



# Write Invalidate vs. Write Update ou Broadcast

- Write Invalidate plus populaire car moins de trafic sur le bus, mais possibilité dune plus grande latence si un autre processeur a besoin de lire un mot dans le même bloc.

# Architectures de mémoires et mécanismes de communication

- Mémoire distribuée et partagée (Distributed **Shared Memory** – DSM architecture)
  - L'espace d'adresses est partagée entre les processeurs même si la mémoire n'est pas centralisée mais distribuée
  - Connue aussi sous le nom de NUMA (nonuniform memory access) parce que le temps d'accès dépend du lieu ou adresse du mot accédé dans la mémoire
- Symmetric **Shared Memory** Multiprocessors (SMP) ou UMA (uniform memory access)
- Multicomputers (**Message-passing** multiprocessors)
  - Mémoire distribuée mais non partagée
  - Multiples espaces mémoires
  - Clusters: ordinateurs connectés par LAN avec peu d'intéractions

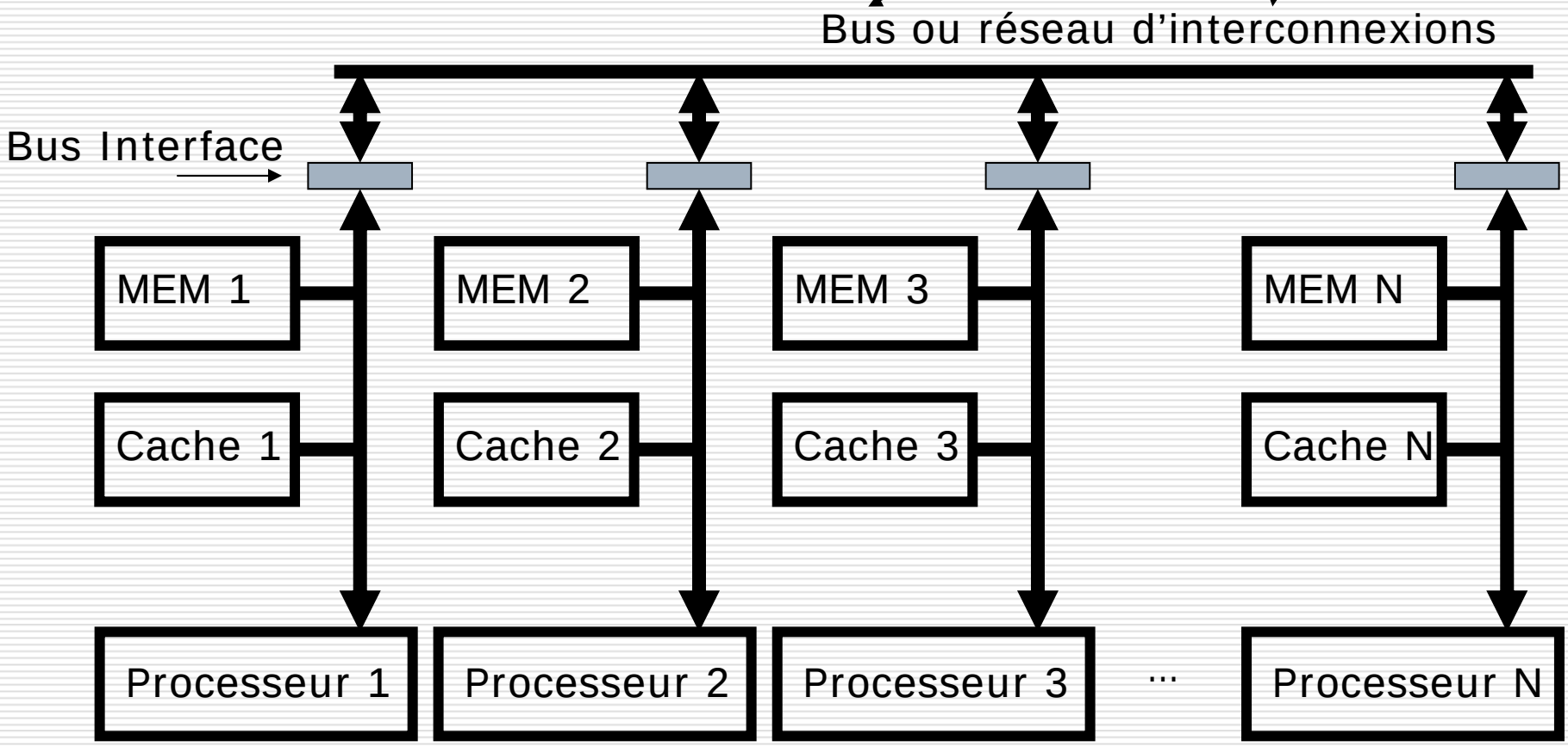
# MIMD avec mémoires distribuées **non-partagées** avec caches

(message-passing multiprocessors)

Préférence

(shared-memory)

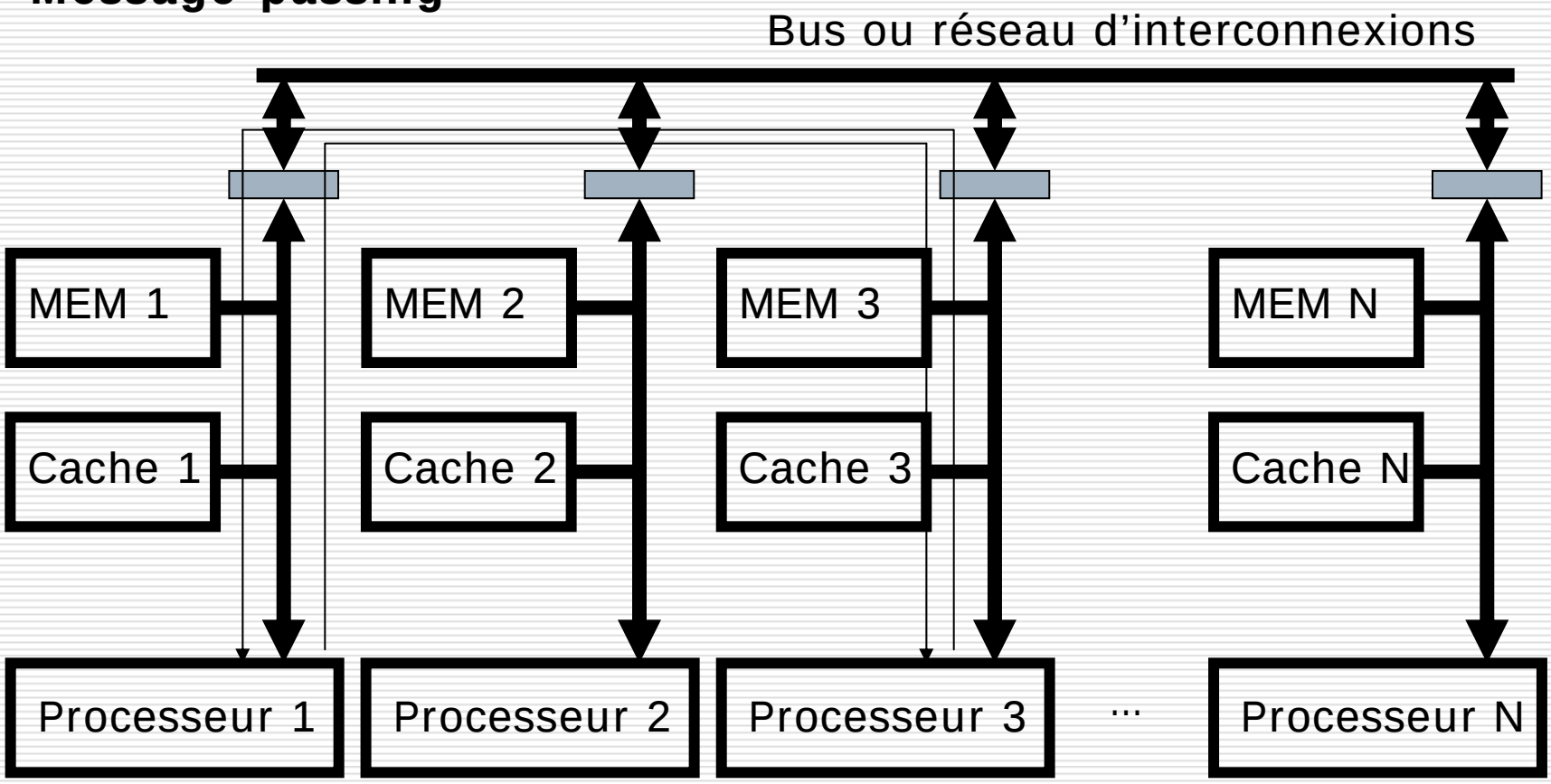
(message-passing)



# MIMD avec mémoires distribuées
# **non-partagées** avec caches – Remote Procedure Call (RPC)

1. P3 requiert DATA dans MEM 1, 2. DATA est envoyé par P1 à P3

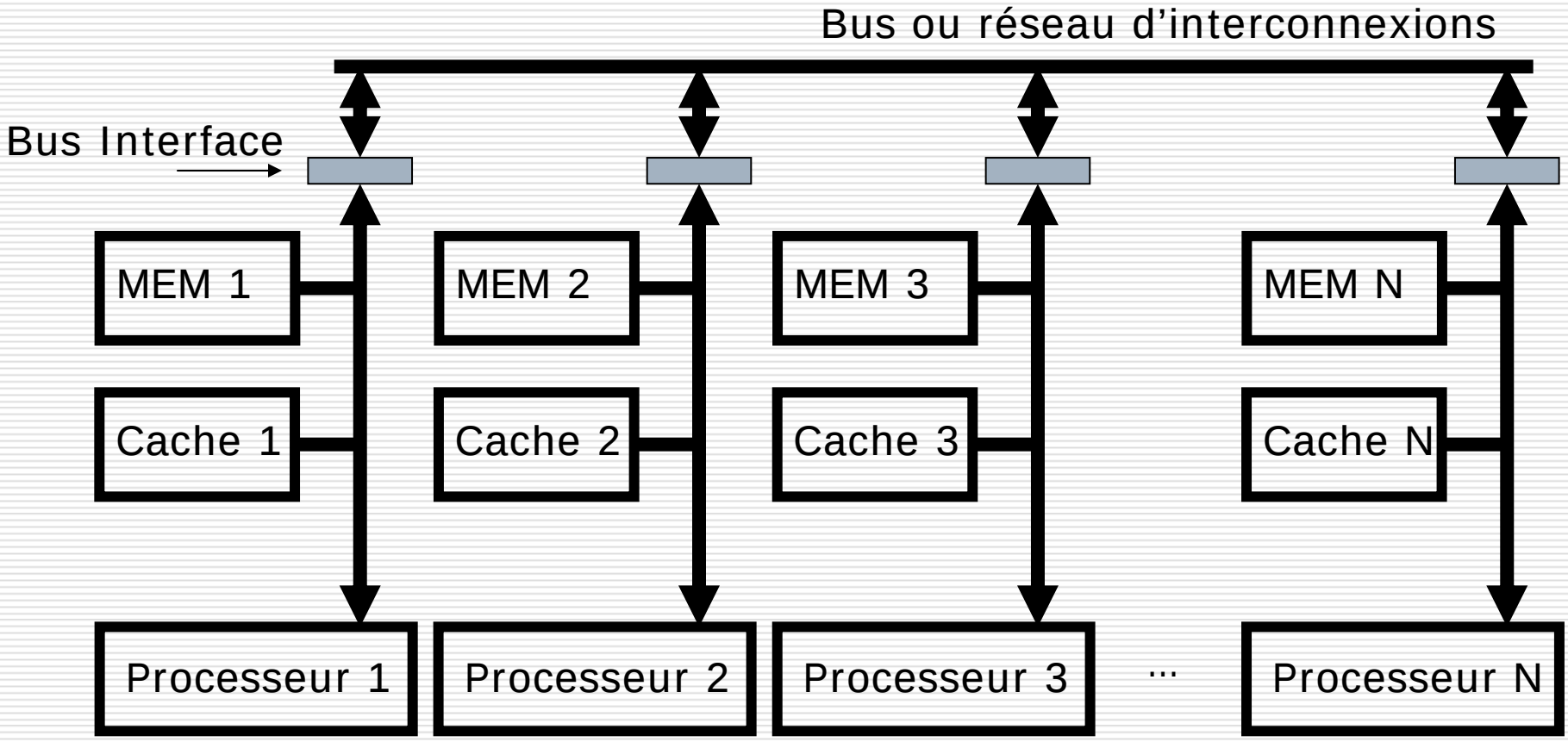
**Message-passing**






Note: Dans le cache ou non, cohérence, etc.

# MIMD avec mémoires distribuées **partagées** (DSM) avec caches

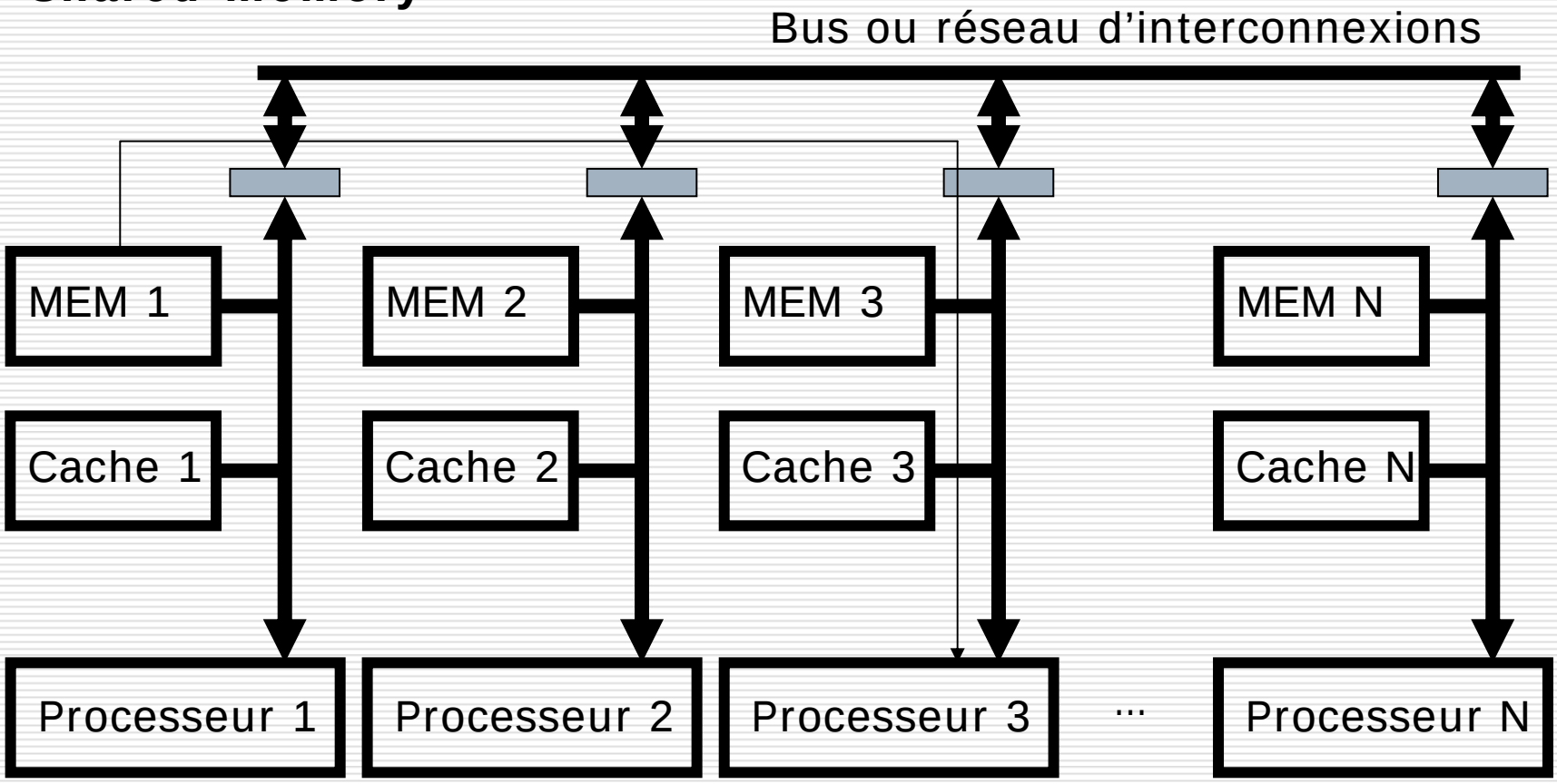


MIMD avec mémoires distribuées

**partagées** avec caches – Accès normale (comme uni-processeur

1. P3 requiert DATA dans MEM 1, 2. P3 accède MEM 1 comme un accès à MEM 3 amis avec plus de cycles d'attente (Wait-states), possibilité de contention sur le bus partagé donc plus long cycle d'accès à la mémoire M1

**Shared-memory**






Note: Dans le cache ou non, cohérence, etc.

# Protocoles de cohérence de caches (cache coherence protocols)

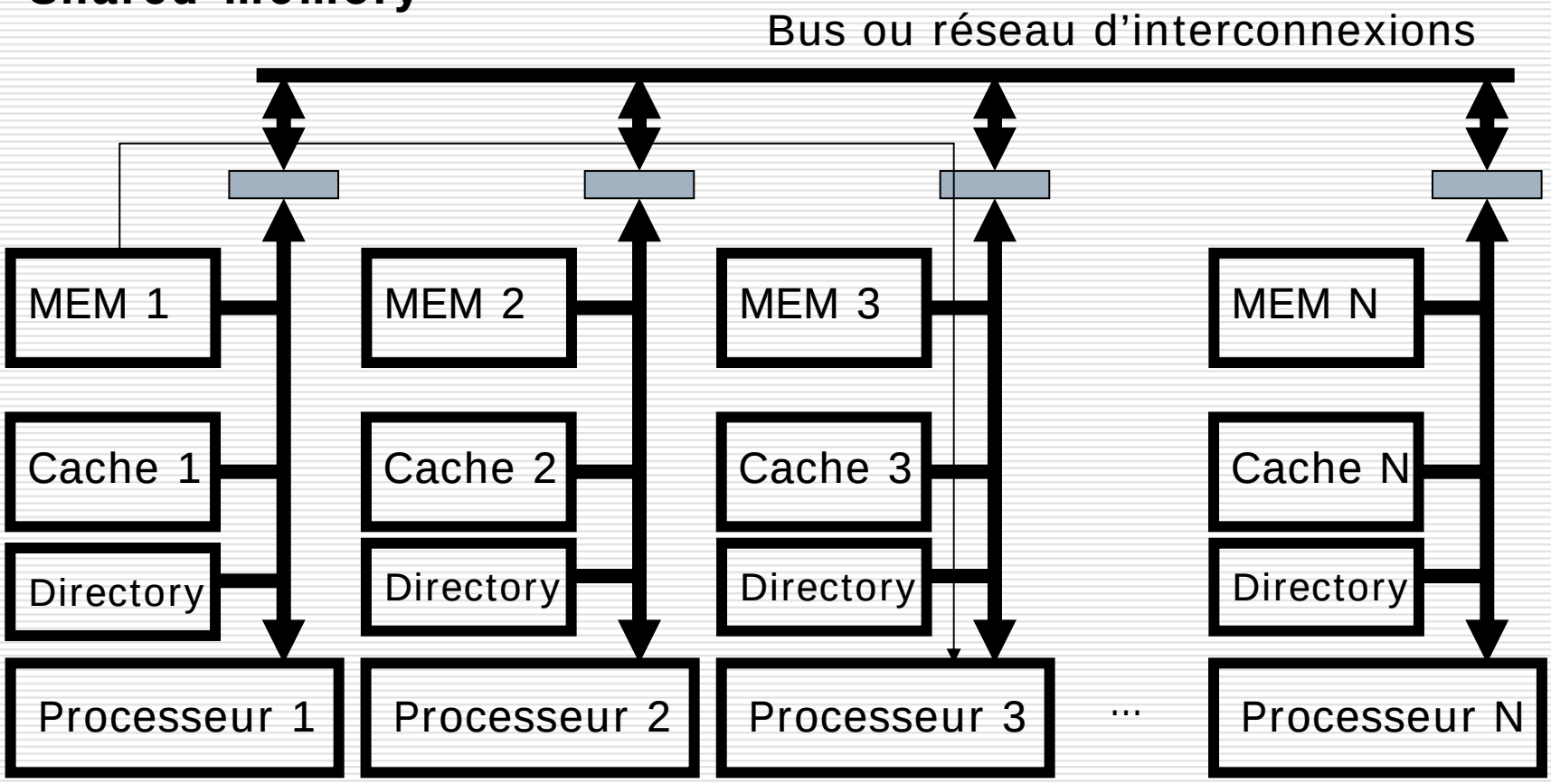
Snooping Protocols
*(Centralized (single) or Symmetric Shared Memory - SMP)*

- Write Invalidate Protocol
- Write Update ou Write Broadcast Protocol

**Directory-based Protocols**
***(Distributed Shared Memory - DSM)***

MIMD avec mémoires distribuées
**partagées** avec caches – Cohérence avec l'ajout de « directories »

**Shared-memory**






Note: Dans le cache ou non, cohérence, etc.

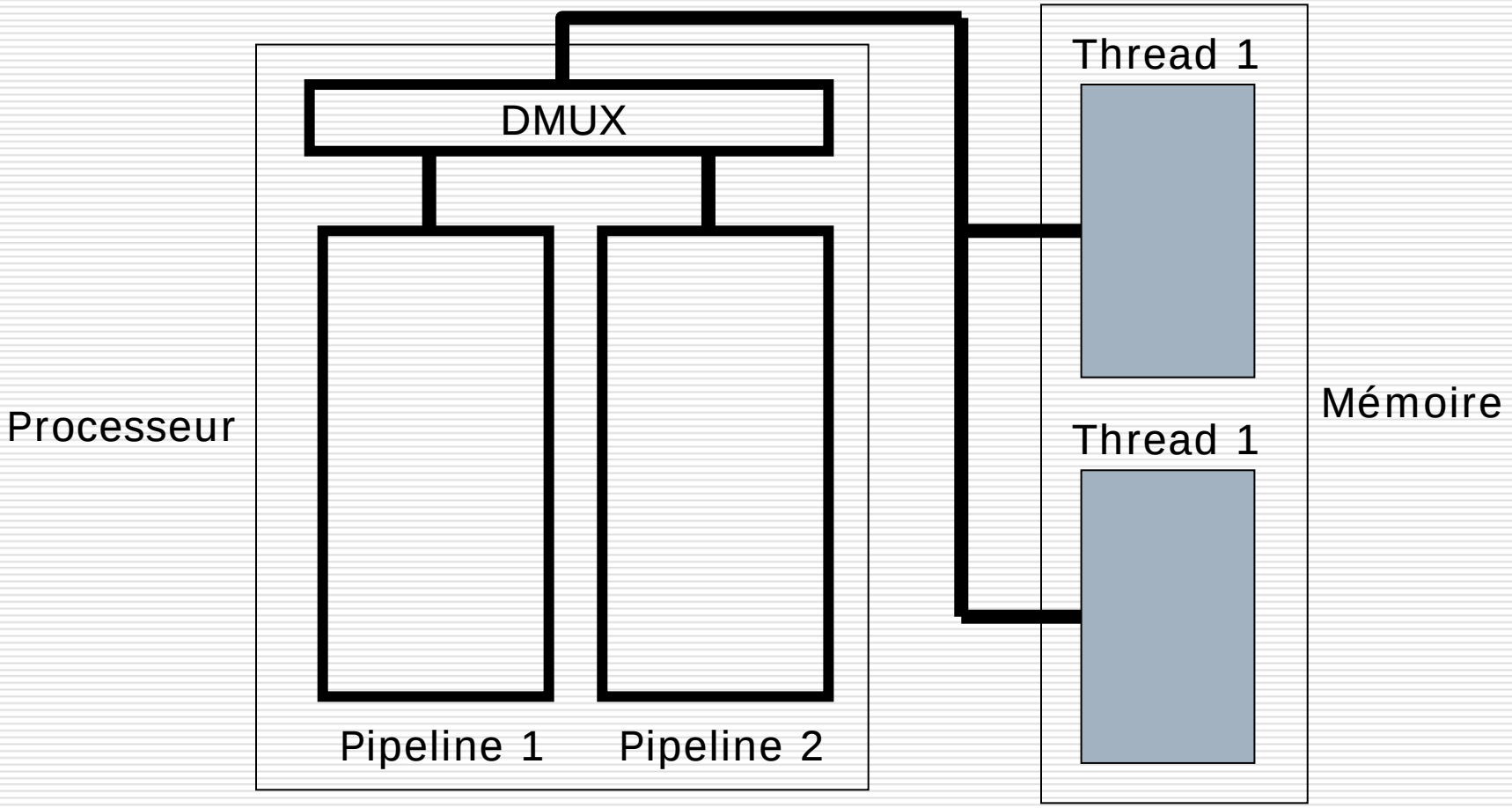
# Directory

- Chaque « directory » est responsable pour le « tracking » des caches qui partagent les mêmes adresses de mémoire dans le noeud (node).
- Le « directory » peut communiquer avec le processeur et la mémoire à travers un bus commun, ou il peut avoir un port séparé à la mémoire, ou il peut être une partie d'un contrôleur central du noeud où toutes les communications intranodes et internode passent.

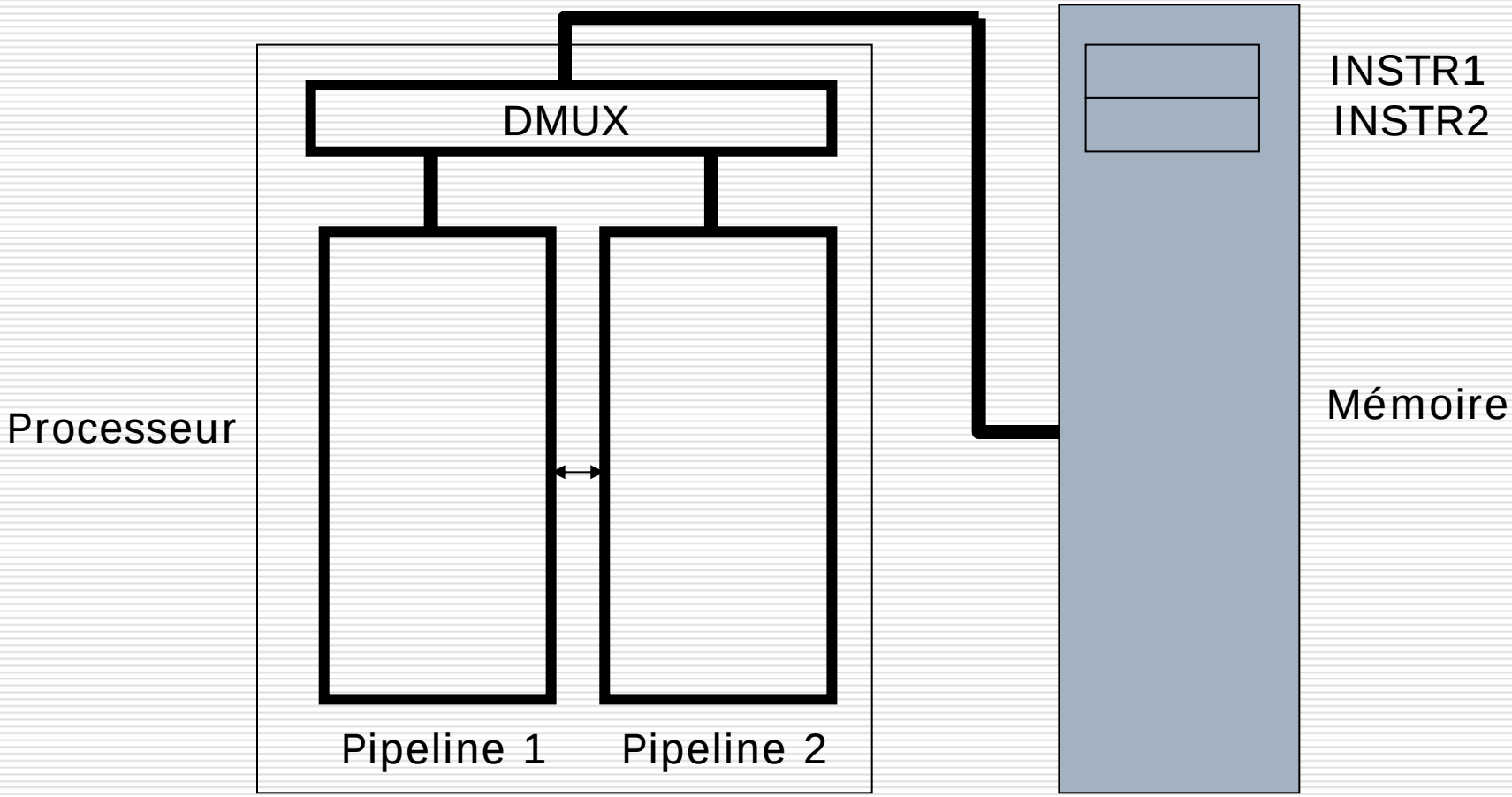
# Synchronisation

- Primitives matérielles de base (Hardware Primitives)
  - Atomic read and modify (test-and-set)
- Autres techniques de synchronization
  - LOCK
  - BARRIER (SPIN-LOCK)

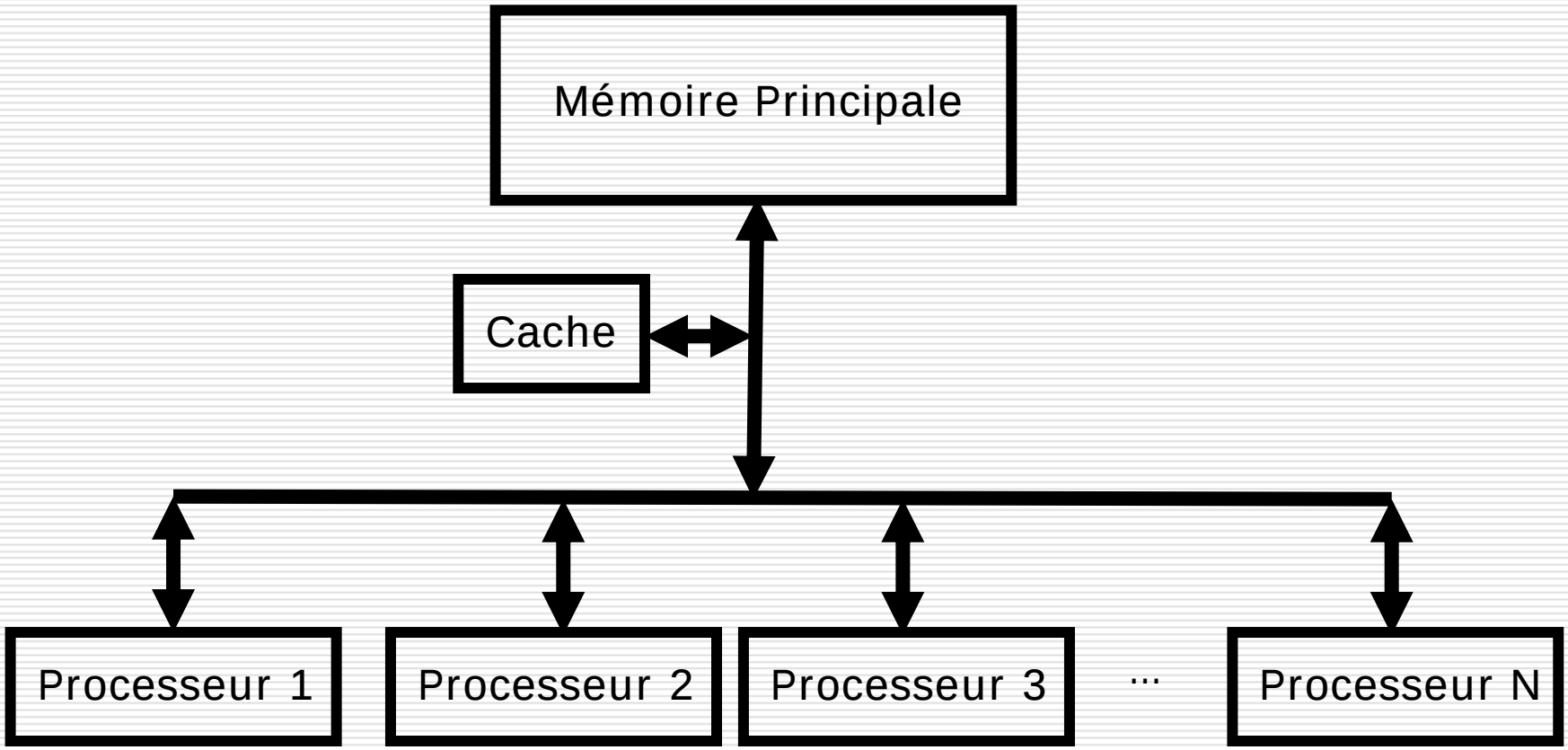
# Multi-threading avec un processeur



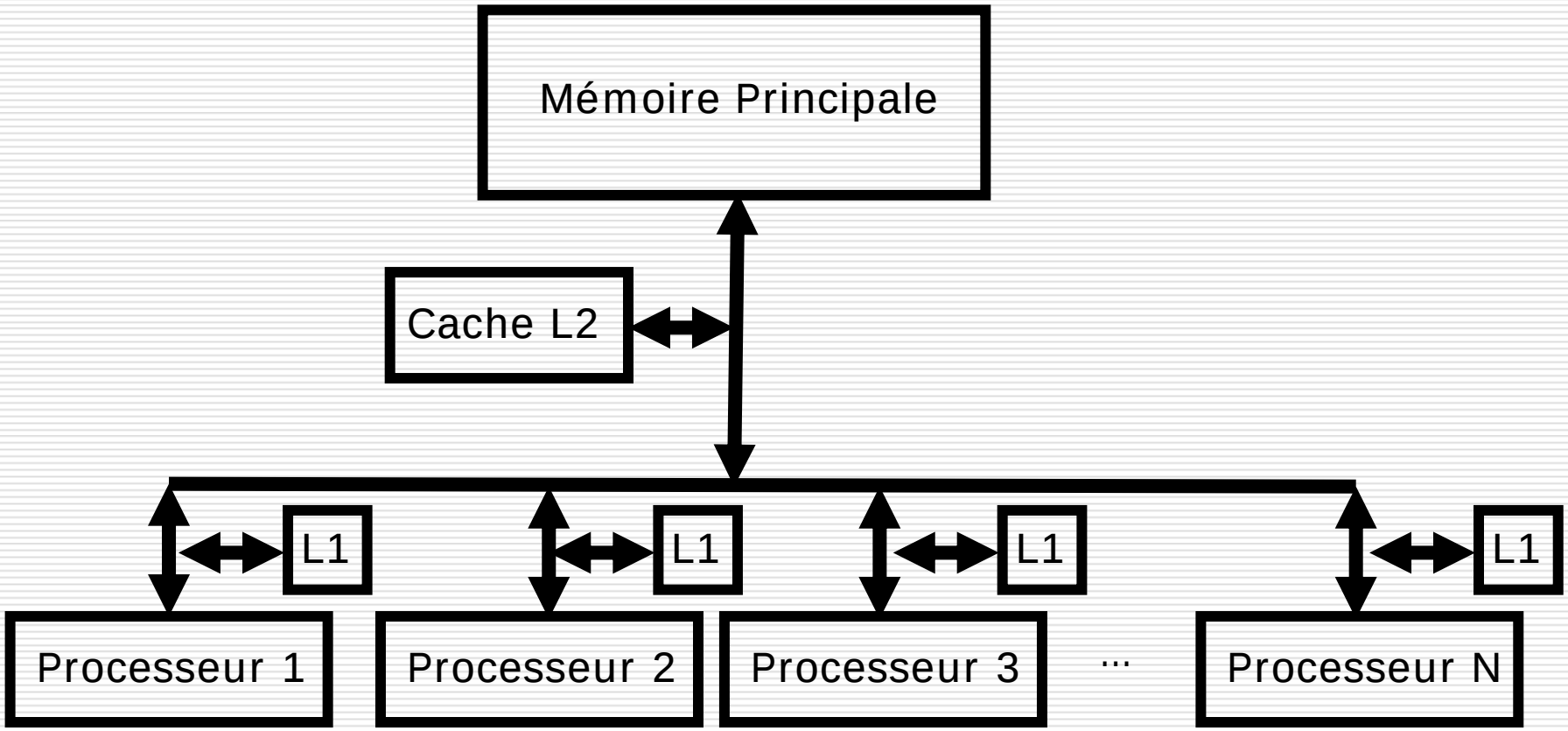
# Multi-threading simultané avec un processeur **superscalaire** (« converting thread-level parallelism into instruction-level parallelism)



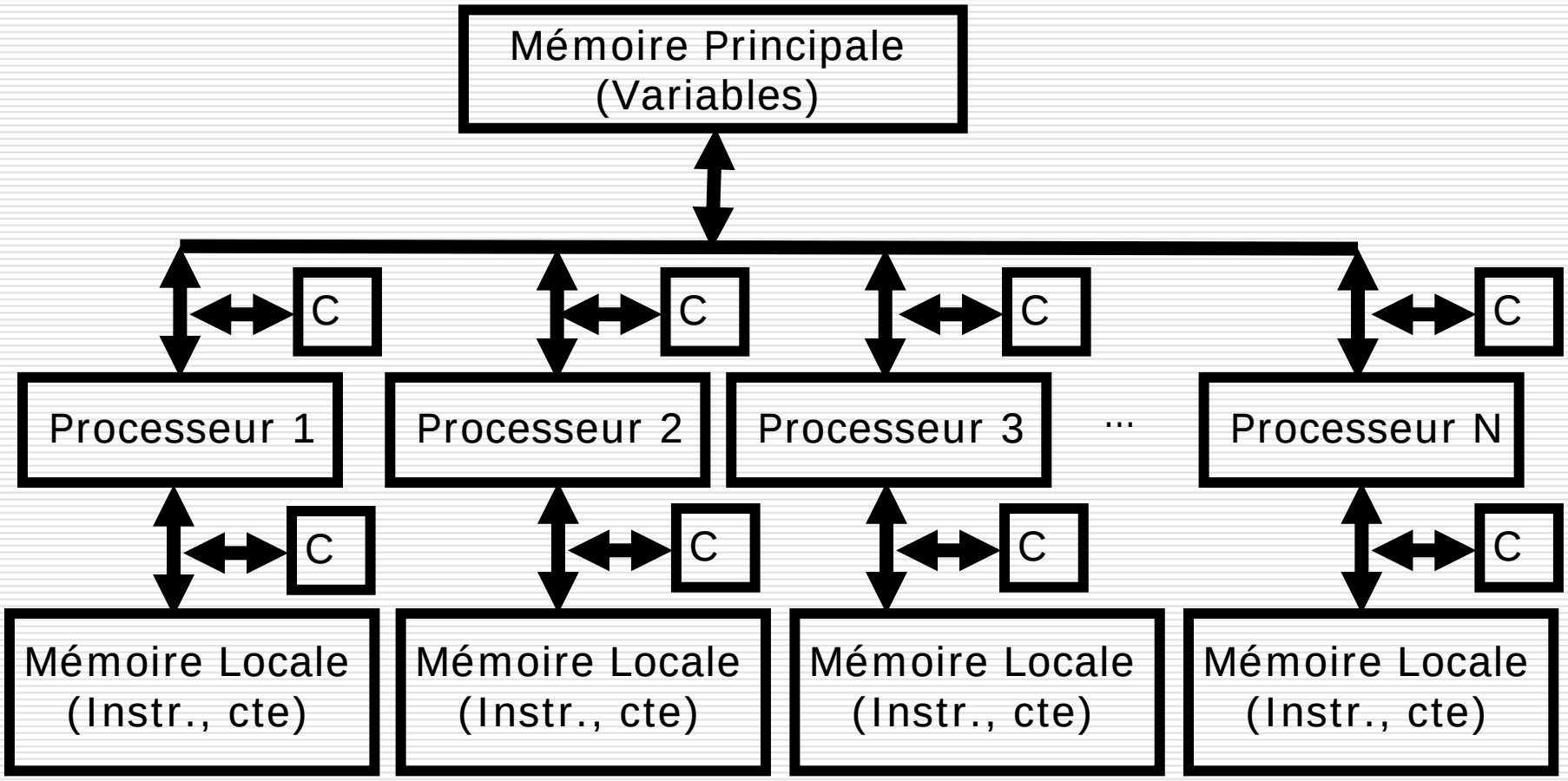
# Autre exemple pour discussion



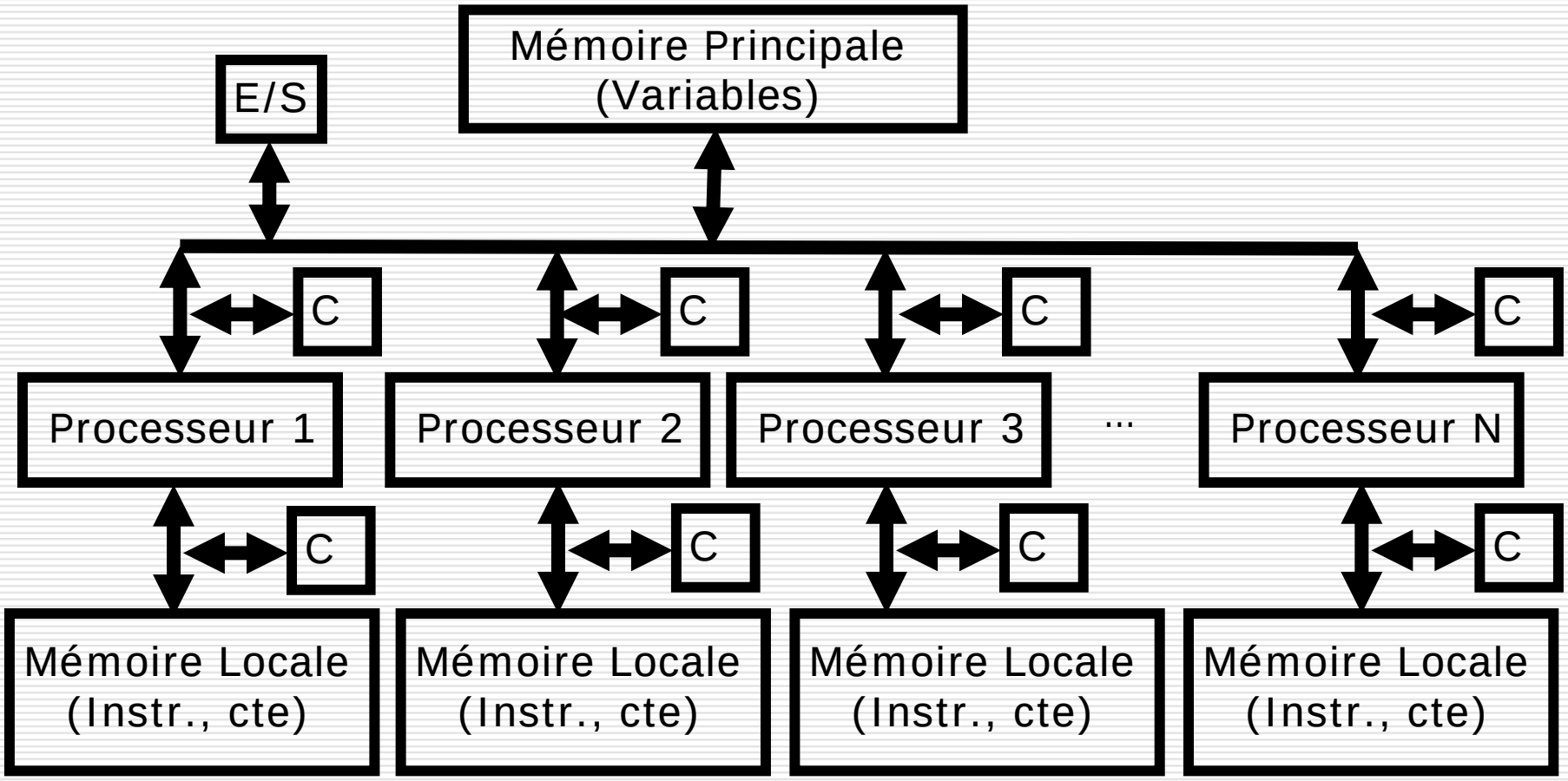
# Autre exemple pour discussion



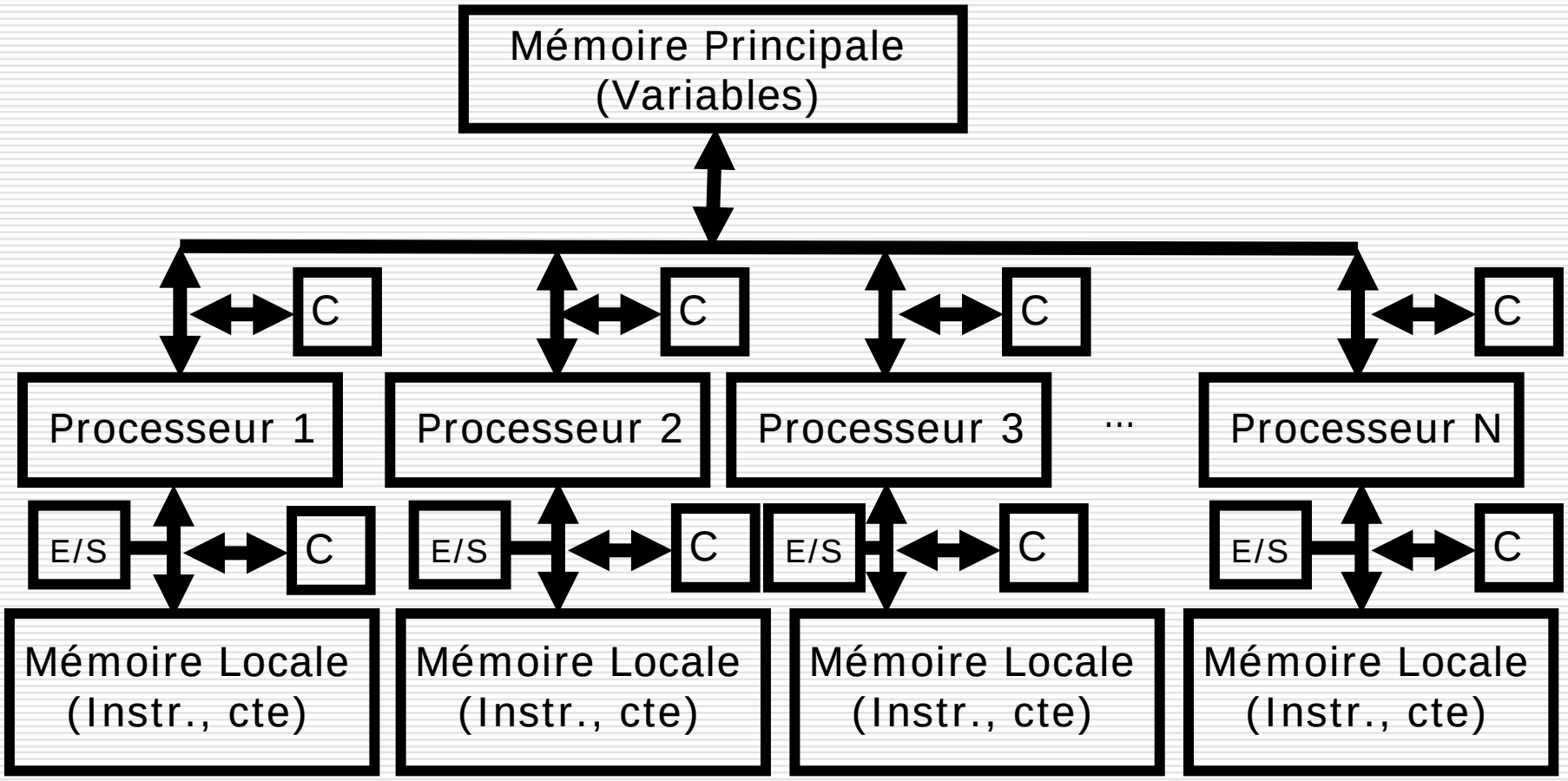
# Autre exemple pour discussion



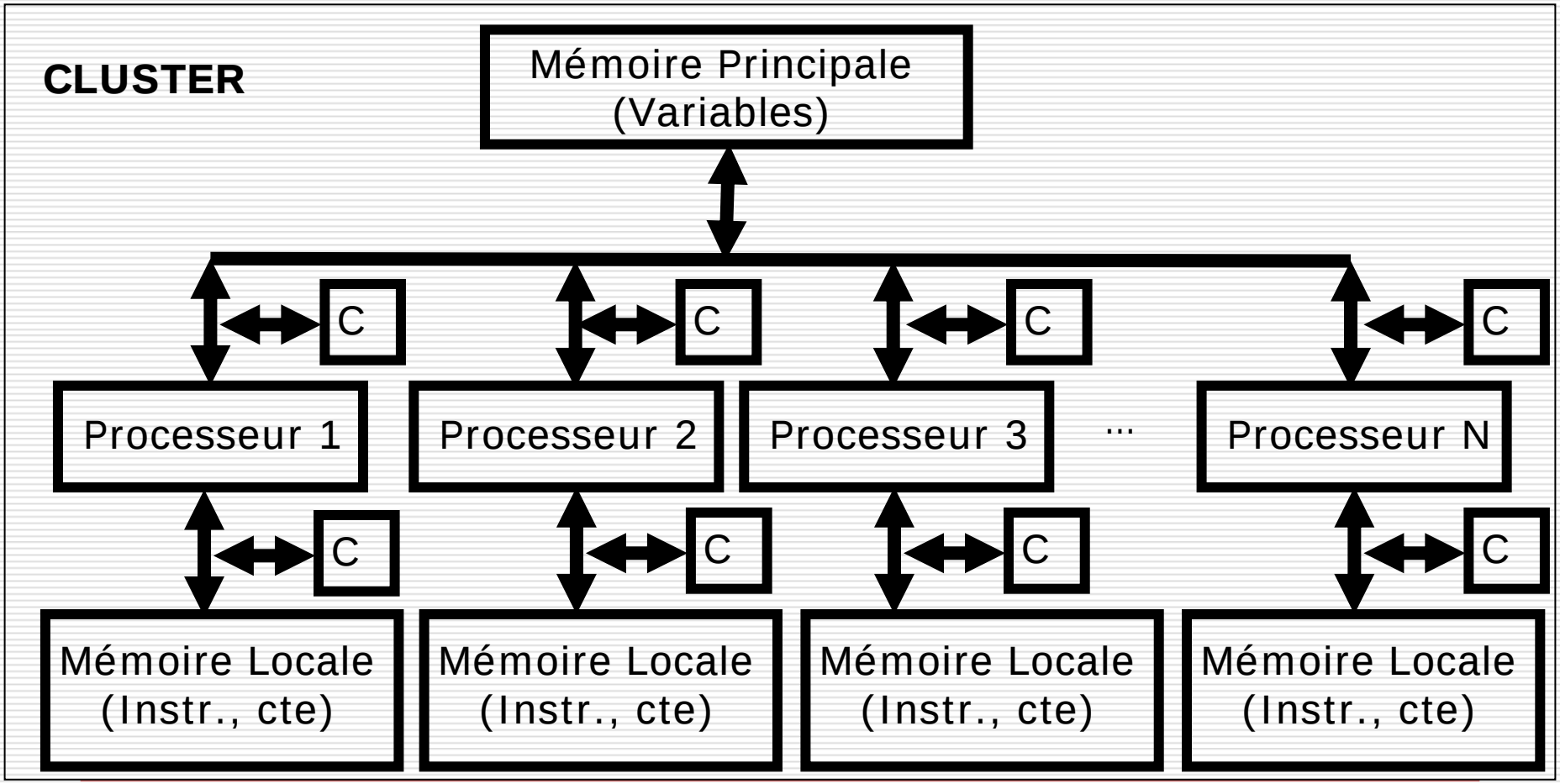
# Autre exemple pour discussion



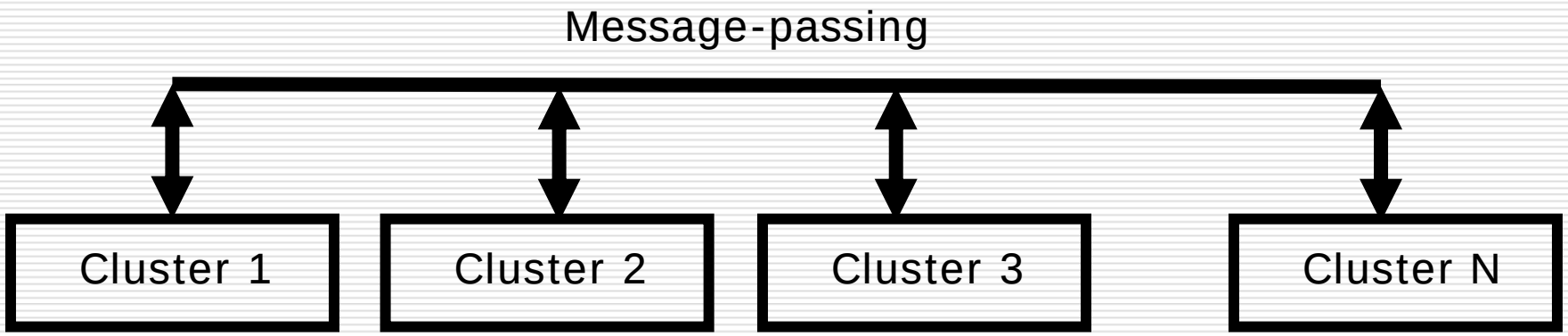
# Autre exemple pour discussion



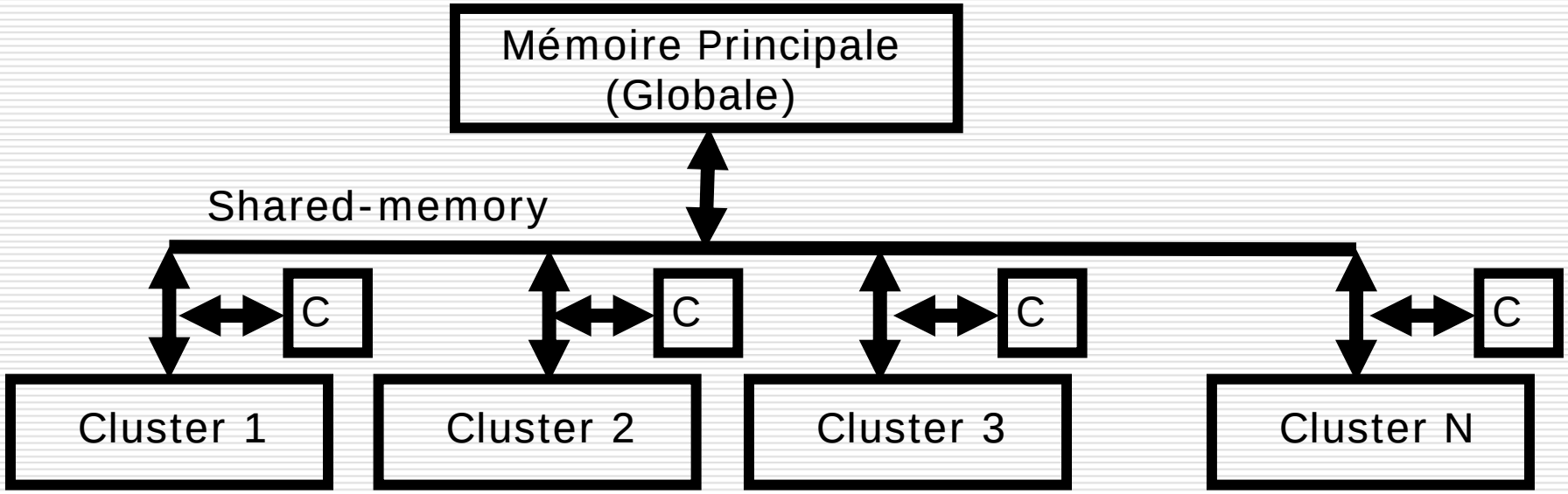
# Autre exemple pour discussion



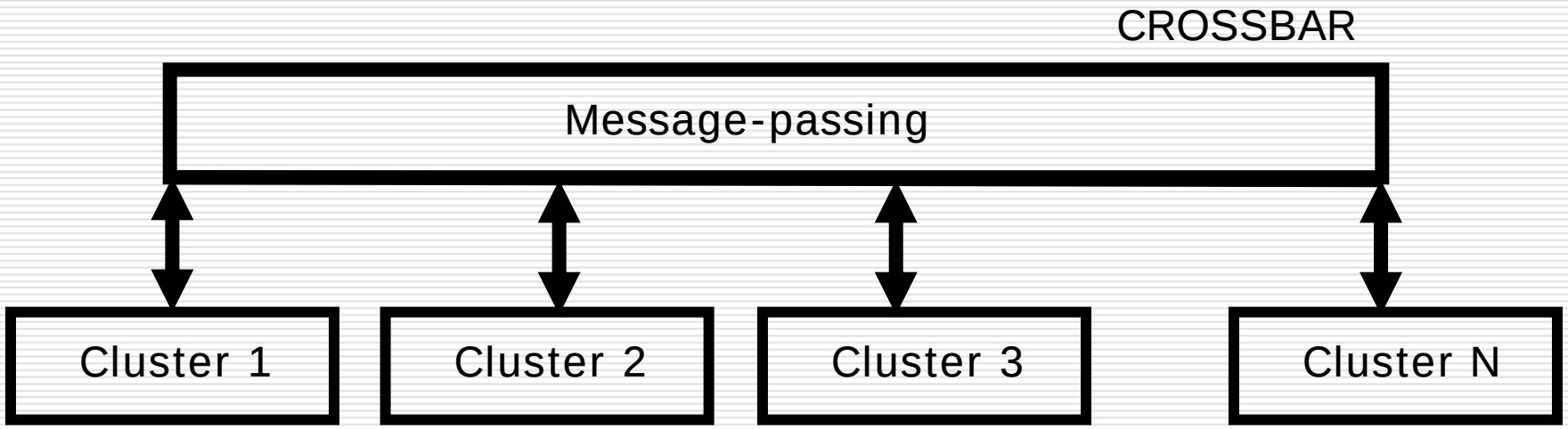
# Autre exemple pour discussion (suite)



# Autre exemple pour discussion



# Autre exemple pour discussion



# Autre exemple pour discussion