1ère partie

Cours Multi Processeurs

(support du cours – CNAM paris-)

Plan du chapitre

LES MULTI – PROCESSEURS

- *La recherche de voies alternatives à la structure classique débuta au milieu des années 60, quand la loi du rendement décroissant commença à prendre effet dans l'effort d'amélioration de vitesse de fonctionnement des ordinateurs ... Les circuits électroniques sont limités en dernier ressort en vitesse de fonctionnement par la vitesse de la lumière ... et beaucoup de circuits fonctionnent déjà dans la gamme des nanosecondes.*

  Bouknight et al, The Illiac IV System (1972)

- *Les ordinateurs séquentiels app rochent d'une limite physique fondamentale dans leur puissance de traitement potentielle. Une telle limite est la vitesse de la lumière ...*

  AL. DeCemaga, The Technology of Parallel Processing, Volume I (1989)

- *Les machines d'aujourd'hui ... sont proches de l'impasse avec les technologies approchant la vitesse de la lumière. Même si les composants d'un processeur séquentiel pouvaient travailler à cette vitesse, le mieux que l'on puisse espérer est bornépar quelques millions d'instructions par seconde.*

  Mitchell, The Transputer : The Time is Now (1989)

Classement de Flynn :

1. **Un seul flot d'instructions, un seul flot de données** 'SISD' – c'est le mono processeur.
2. **Un seul flot d'instructions, plusieurs flots de données** 'SIMD' –
   - La même instruction est exécutée par plusieurs processeurs utilisant différents flots de données. Chaque processeur a sa propre mémoire de données (d'où plusieurs données) mais il y a une seule mémoire d'instructions et un seul processeur de contrôle qui lit et lance les instructions.
3. **Plusieurs flots d'instructions, un seul flot de données** 'MISD' –
   - Aucune machine commerciale de ce type n'a été construite à ce jour, mais pourrait l'être dans le futur
4. **Plusieurs flots d'instructions, plusieurs flots de données** 'MIMD' –
   - Chaque processeur lit ses propres instructions et opère sur ses propres données. Les processeurs sont souvent des microprocesseurs standards. Dans la réalité, certaines machines appartiennent à plusieurs catégories. Nous nous intéresserons essentiellement au MIMD.

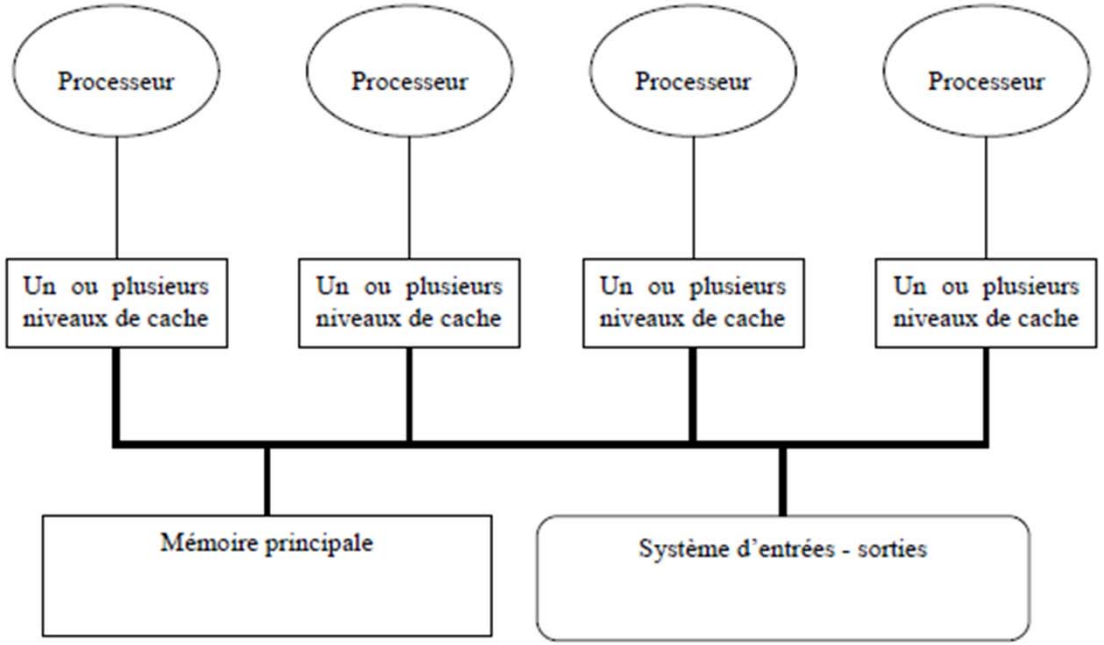
Classement de Flynn :

1. **Un seul flot d'instructions, un seul flot de données** 'SISD' – c'est le mono processeur.
2. **Un seul flot d'instructions, plusieurs flots de données** 'SIMD' –
   - La même instruction est exécutée par plusieurs processeurs utilisant différents flots de données. Chaque processeur a sa propre mémoire de données (d'où plusieurs données) mais il y a une seule mémoire d'instructions et un seul processeur de contrôle qui lit et lance les instructions.
3. **Plusieurs flots d'instructions, un seul flot de données** 'MISD' –
   - Aucune machine commerciale de ce type n'a été construite à ce jour, mais pourrait l'être dans le futur
4. **Plusieurs flots d'instructions, plusieurs flots de données** 'MIMD' –
   - Chaque processeur lit ses propres instructions et opère sur ses propres données. Les processeurs sont souvent des microprocesseurs standards. Dans la réalité, certaines machines appartiennent à plusieurs catégories. Nous nous intéresserons essentiellement au MIMD.

LES MULTI – PROCESSEURS

❑ Mémoire partagée centralisée :

- Au plus quelques dizaines de processeurs.
- La mémoire est partagée à travers un bus.
- Chaque processeur possède une mémoire cache.



❑ Mémoire partagée centralisée :

- Dans ce genre d'architecture, les caches peuvent contenir à la fois des données privées et des données partagées. Ces dernières permettent la communication entre les processeurs.
- Les valeurs partagées peuvent être présentes dans plusieurs caches. Suit alors des problèmes de cohérence de cache..

Temps	Evénements	Contenu du cache pour UC A	Contenu du cache pour UC B	Contenu mémoire pour case X
0				1
1	UC A lit X	1		1
2	UC B lit X	1	1	1
3	UC A range 0 dans X	O	1	0

B voit l'ancienne valeur de X.
Pour maintenir la cohérence des caches, le multi – processeur utilise un protocole :
Le protocole de cohérence des caches.

❑ Deux techniques utilisées dans les protocoles :

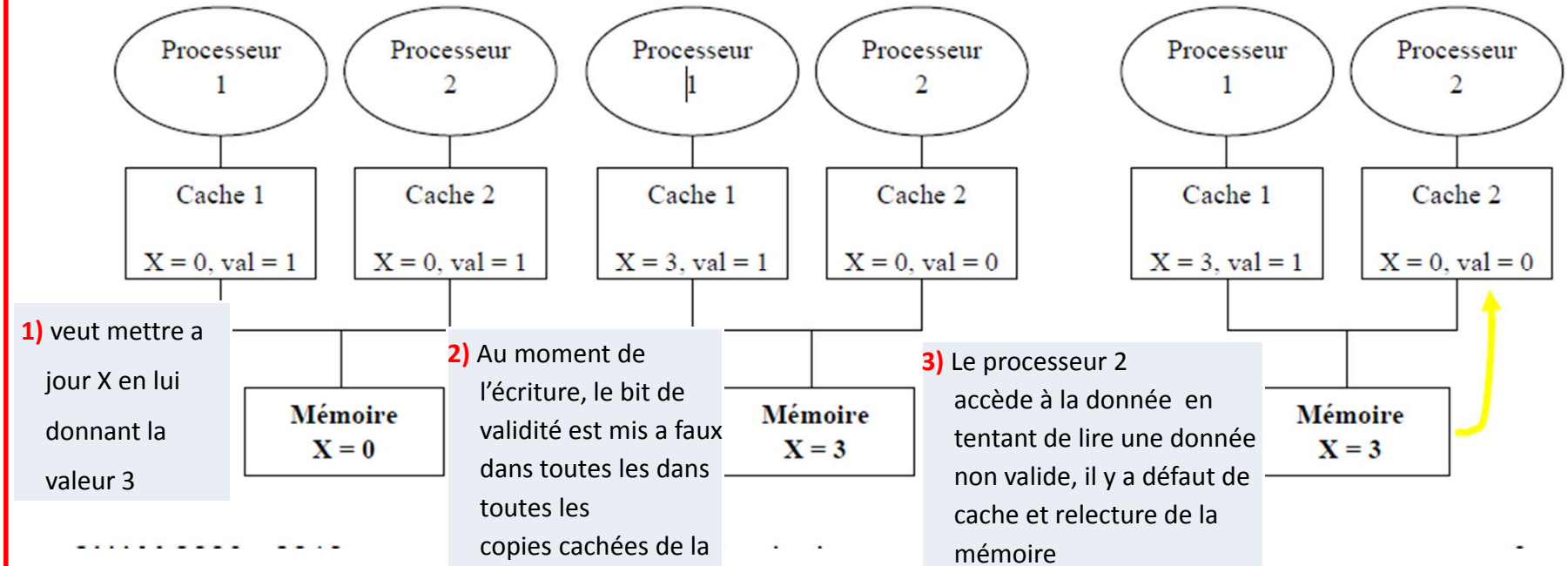
1. **Le répertoire.**
   - L'état de partage d'un bloc de mémoire physique est conservé dans un endroit unique, appelé répertoire. Cette approche est plutôt utilisée dans l'architecture à mémoire partagée.
2. **L'espionnage.**
   - Chaque cache ayant une copie de la donnée d'un bloc de la mémoire physique a aussi une copie de l'état de partage du bloc. Aucune information centralisée n'est utilisée. Tous les contrôleurs de caches surveillent, ou espionnent le bus les connectant à la mémoire pour savoir s'ils on ou non une copie du bloc qui est en transfert sur le bus.

❑ **Deux protocoles existent :**

1. Protocole à invalidation d'écriture.
2. Protocole de mise à jour d'écriture (ou diffusion d'écriture).

❑ Protocole à invalidation d'écriture :

- C'est le plus courant pour l'espionnage et pour les répertoires. Il s'agit de mettre en place un accès exclusif qui assure qu'aucune autre copie de la donnée n'existe quand une écriture intervient.
- Toutes les copies « cachées » d'une donnée sont invalidées. Si un processeur tente alors de lire la donnée, il y a un défaut de cache qui force la lecture (depuis la mémoire) de la donnée pour mettre une copie à jour dans le cache



❑ Protocole à invalidation d'écriture :

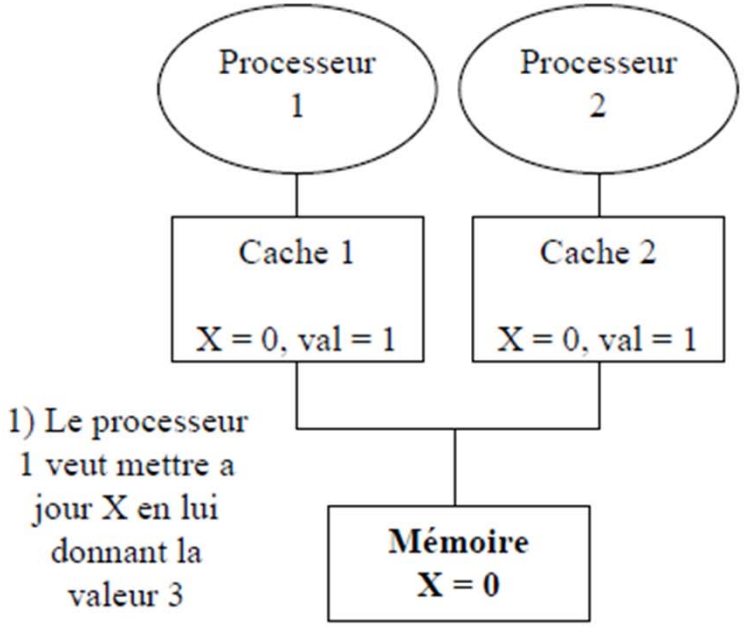
- C'est le plus courant pour l'espionnage et pour les répertoires. Il s'agit de mettre en place un accès exclusif qui assure qu'aucune autre copie de la donnée n'existe quand une écriture intervient.
- Toutes les copies « cachées » d'une donnée sont invalidées. Si un processeur tente alors de lire la donnée, il y a un défaut de cache qui force la lecture (depuis la mémoire) de la donnée pour mettre une copie à jour dans le cache

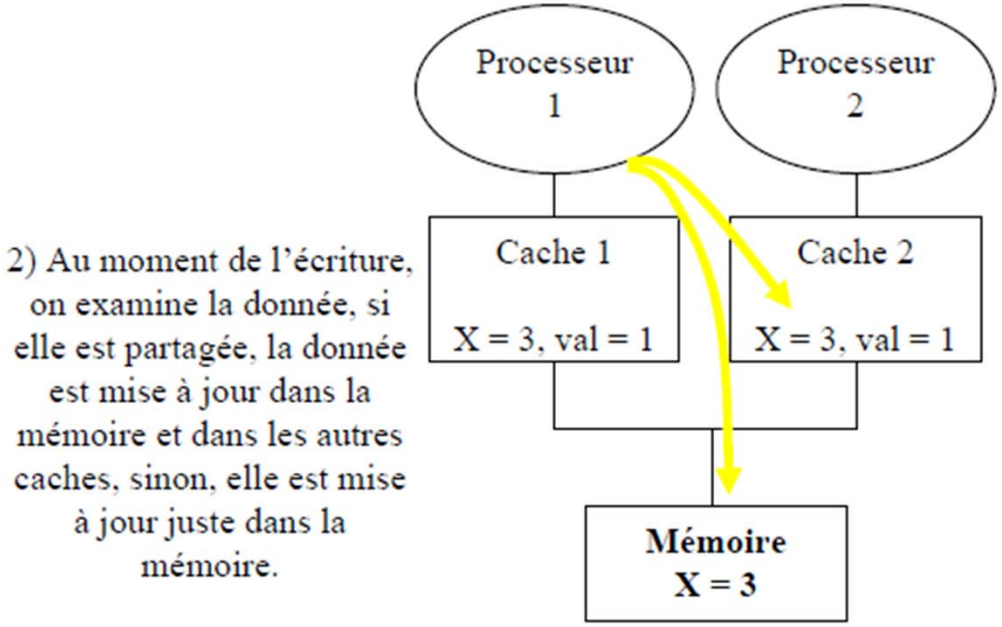
Le processeur

1 veut mettre a jour X en lui donnant la valeur 3	Mémoire X = 0	2) Au moment de l'écriture, le bit de validité est mis a faux dans toutes les copies cachées de la donnée.	Mémoire X = 3	3) Le processeur 2 accède à la donnée en tentant de lire une donnée non valide, il y a défaut de cache et relecture de la mémoire	Mémoire X = 3

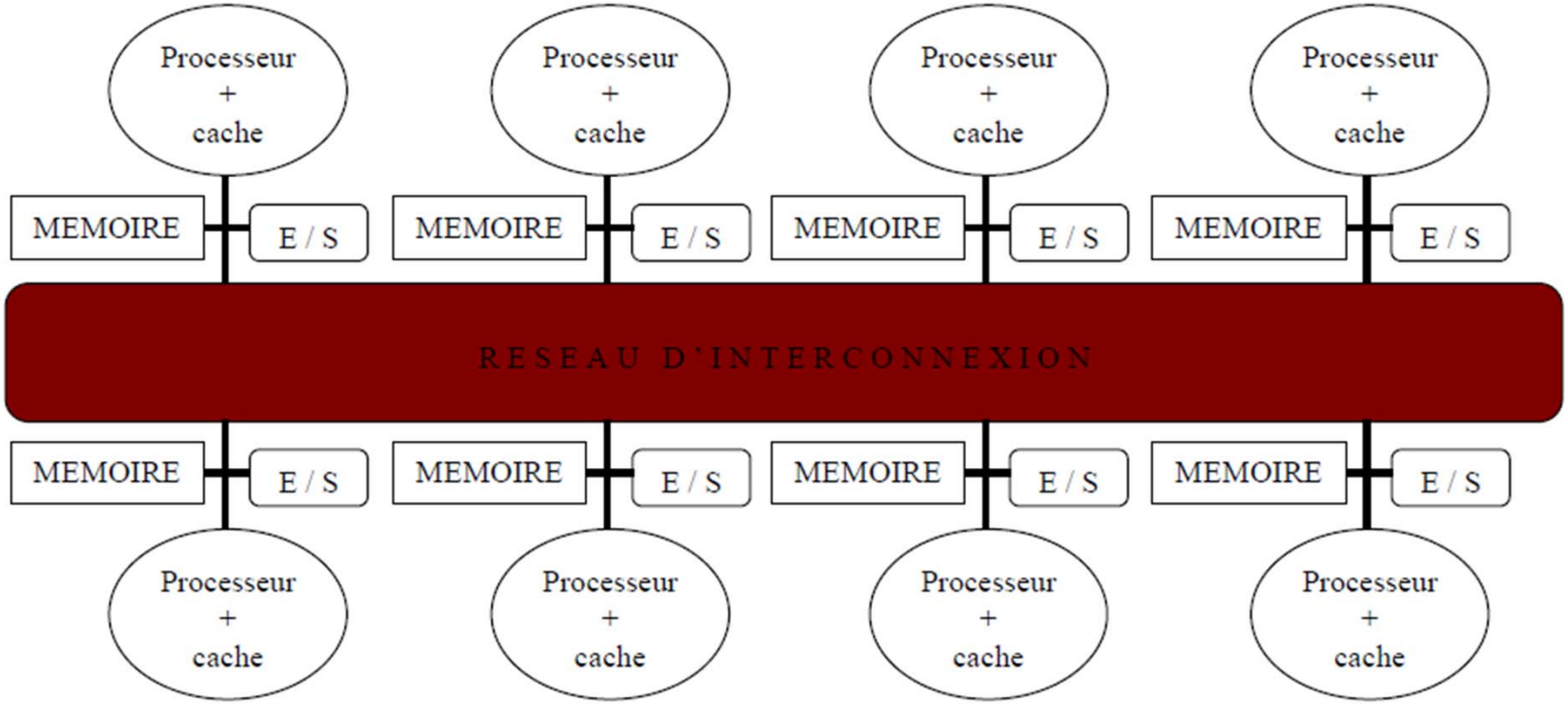
Protocole de mise à jour d'écriture (ou diffusion d'écriture).

- Dans ce protocole, il s'agit de mettre à jour simultanément toutes les copies d'une donnée, ou qu'elles soient.
- Il est donc utile de savoir en permanence si un mot du cache est partagé ou non. Si ce n'est pas le cas, il n'est pas nécessaire de mettre à jour les autres caches.

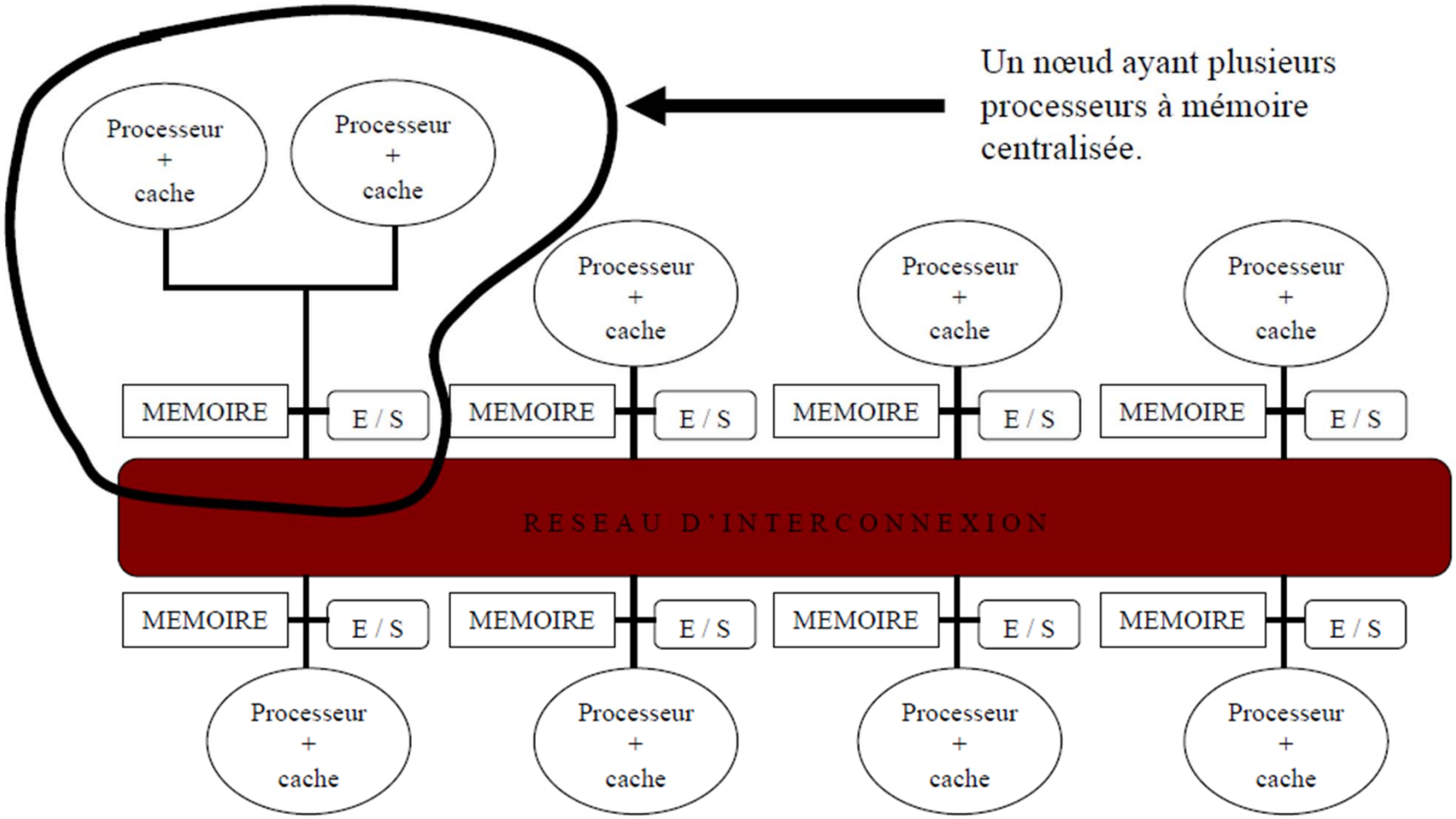




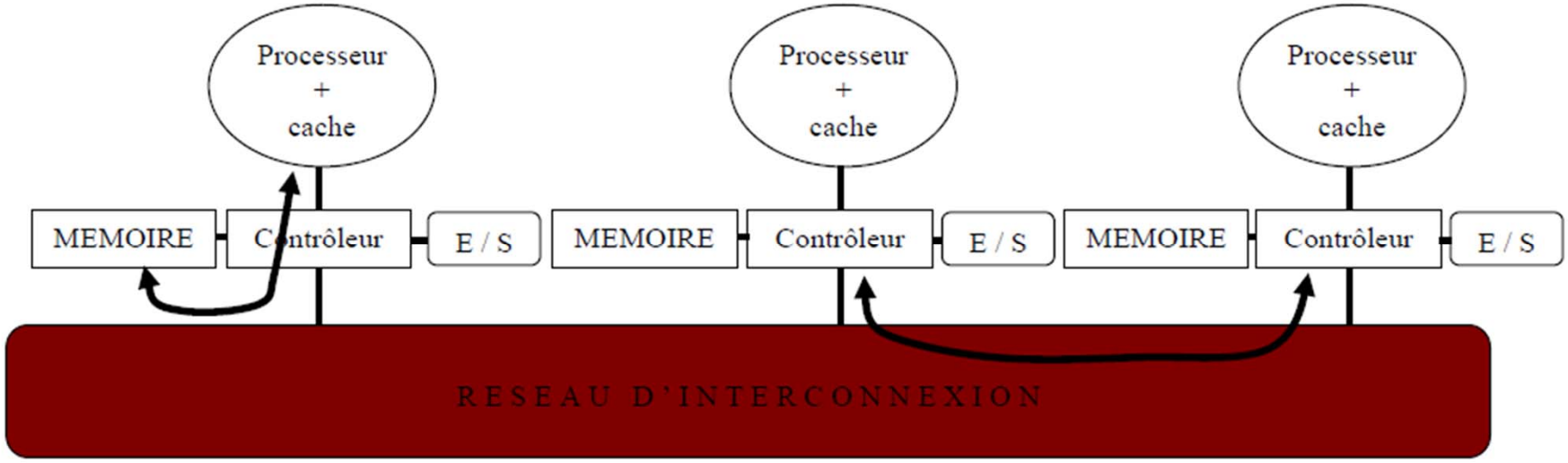
- Les machines possédant un grand nombre de processeurs ne peuvent pas utiliser la mémoire centralisée parce que celle – ci n'a pas assez de bande passante.
  - La mémoire est distribuée entre les processeurs.
  - Augmentation du débit mémoire puisque locale.
  - **Un processeur + son cache + sa mémoire + son mécanisme d'entrées –sorties forme un nœud.**



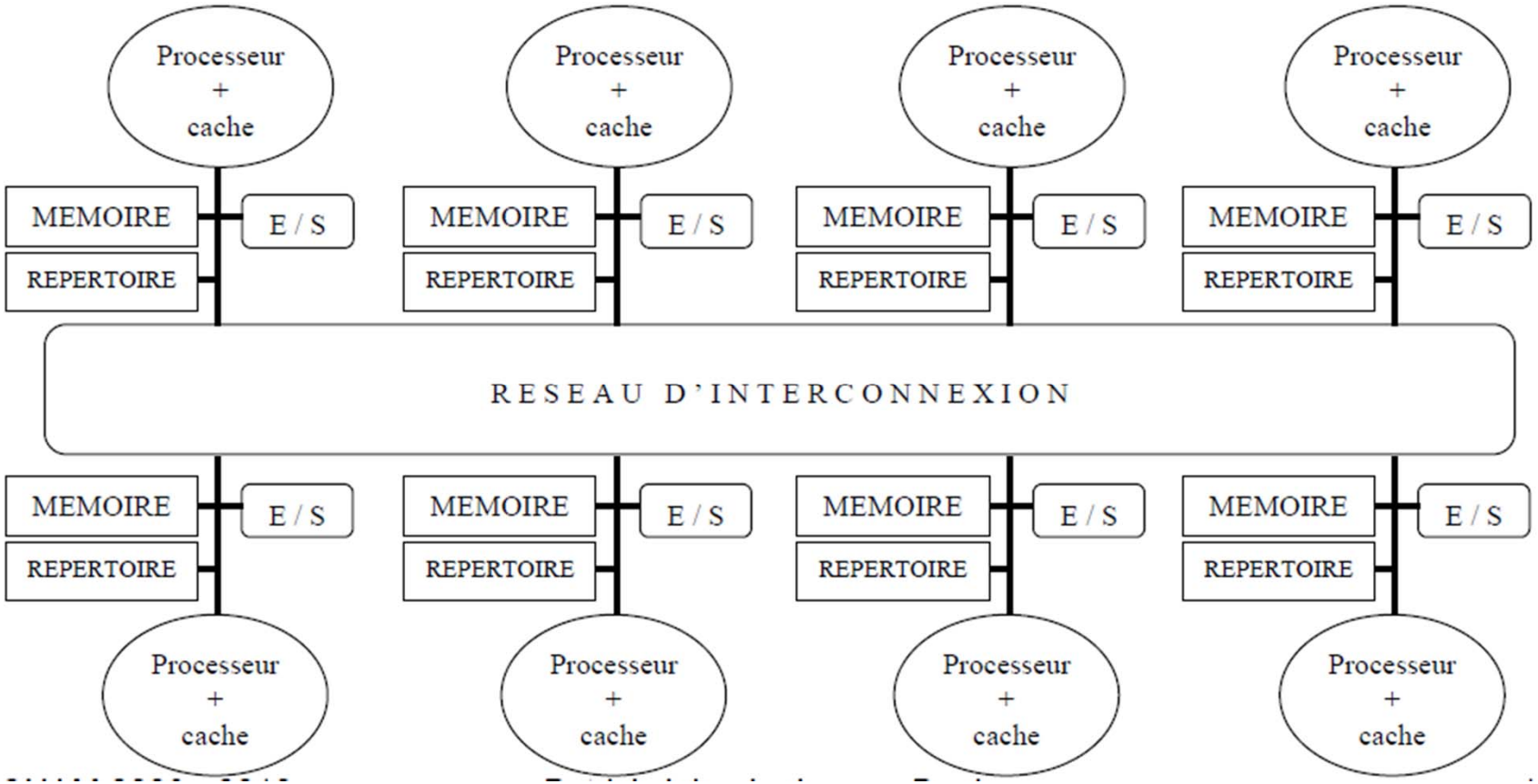
- Un nœud peut contenir plusieurs processeurs entre eux avec une autre architecture comme la mémoire centralisée. C'est pourquoi nous parlerons souvent de nœuds.



- Le Cray T3D est un exemple de machine à mémoire distribuée. Dans cette machine :
  - La mémoire est distribuée dans les nœuds qui sont interconnectés par un réseau.
  - Chaque nœud possède un contrôleur.
  - Les accès peuvent être local au nœud ou lointain. Le contrôleur identifie le type d'accès en fonction de l'adresse mémoire.
  - Si l'accès est lointain, il y a communication entre les deux contrôleurs des nouds concernés pour accéder à la donnée

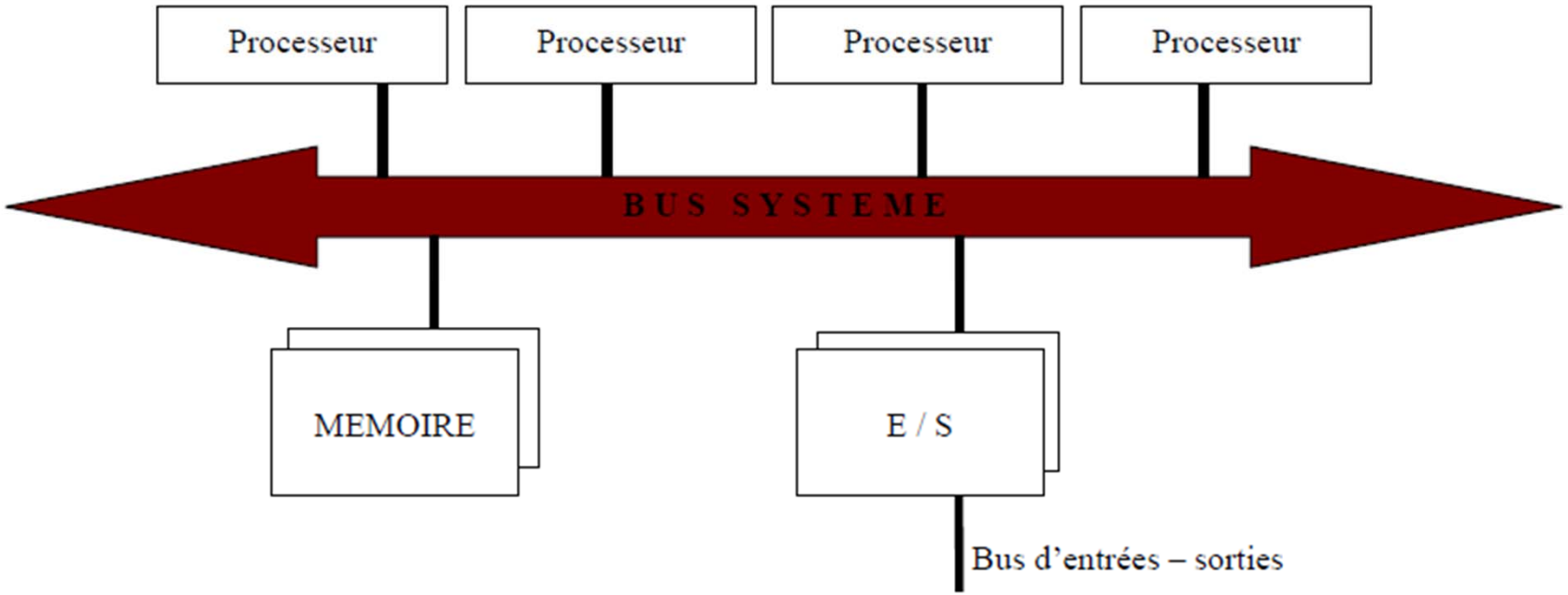


- Pour ce type machine, le protocole à répertoire est plus adapté.
- Les entrées du répertoire sont distribuées avec la mémoire, les répertoires conservent l'état de partage d'un bloc dans un seul endroit. Chaque répertoire suit l'état des caches dans le nœud pour les adresses données dans le nœud. Le répertoire peut faire partie



- On appelle système **multi – processeurs symtriques (SMP)** une architecture multi – processeurs dans laquelle tous les processeurs sont banalisés. Ce type d'architecture porte aussi le nom de multi – processeurs à couplage serré.
- Par opposition, certaines architectures réservent des tâches particulières, comme par exemple initialiser mes opérations d'entrées – sorties à un seul processeur.
- **Ce type d'architecture permet :**
  - Ajuster la puissance de calcul aux besoins réels.
  - La charge de traitement est réparti entre les processeurs du systèmes
  - Cette approche est économique.

- ***C'est souvent une architecture à mémoire centralisée.***
  - Les premières machines datent des années 60 avec les mainframes, puis dans les années 70, les machines Unix se sont emparés de cette architecture.
  - Aujourd'hui elle est présente dans les PC haut de gamme car elle supportée par Windows – NT, 2000, XP, 2003 et Linux.



- Tous les processeurs peuvent jouer le même rôle au sein du système.
- Tous les processeurs peuvent accéder à toutes les ressources du système.
- Le **SMP** fonctionne sous le contrôle d'un seul système d'exploitation.

- **Pour tirer profit de la puissance disponible, il faut que le logiciel prenne en compte l'architecture SMP autant pour les applications que pour le système d'exploitation.**
  - Les logiciels conçus pour les monoprocesseurs peuvent être exploités sur un **SMP.**
  - La performance de l'application est au mieux identique à celle obtenue sur un système mono – processeur.
  - Les applications doivent être conçues sous forme de modules multiples susceptibles de s'exécuter simultanément.
  - Le système d'exploitation doit également avoir été conçu pour le **SMP.**

- **A un moment donné, plusieurs processeurs peuvent exécuter des routines du système d'exploitation**.
  - L'accès à une ressource partagée doit être muni d'un verrou, et toute procédure désirant accéder à cette ressource doit en obtenir l'autorisation en plaçant une demande d'accès au niveau du verrou.

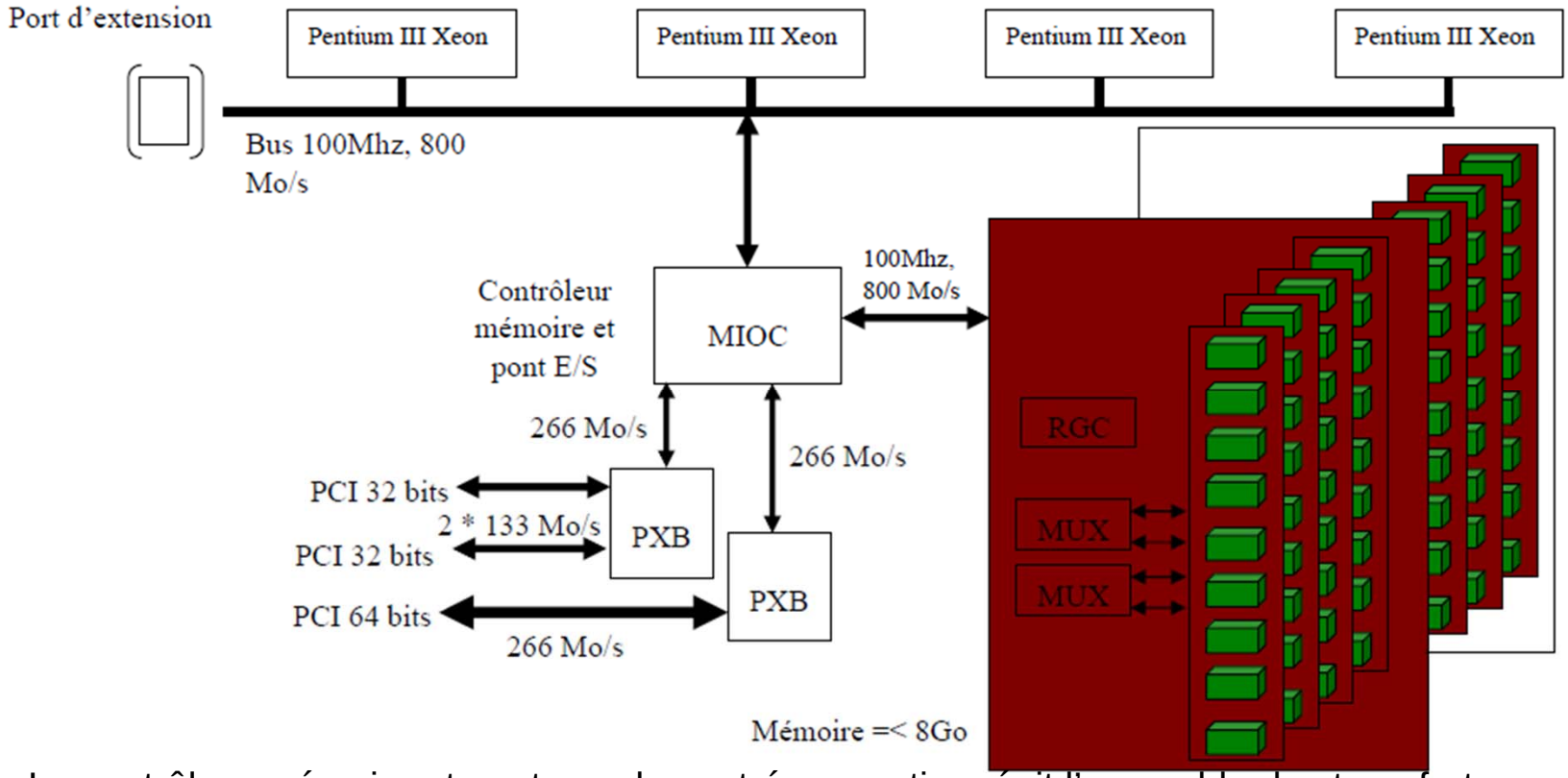
- **L'adaptation d'un système à l'environnement SMP pose les problèmes suivants :**
  - **Allocation des threads aux processeurs :**

    La tendance naturelle serait de laisser l'ordonnanceur choisir l'allocation thread – processeur. Mais la présence des caches conduit à utiliser la notion d'affinité.

    - ✓ On dit qu'un processus a une affinité avec un processeur s'il s'est déjà exécuté dessus. De cette façon les données de ce processus peuvent être dans le cache. On évite ainsi une diminution des performances dues aux défauts de cache.
    - ✓ Si la notion d'affinité est dynamique, il est possible de créer, pour certaines applications critiques, une association thread –processeur permettant d'assurer des temps de réponse courts.
  - **Gestion des interruptions :**
    - ✓ Afin de veiller à un traitement équilibré au sein du système, le système d'interruption, doit dans la mesure du possible, diriger les interruptions vers les processeurs les moins chargés. Pour cela, un registre matériel enregistre l'état de charge des processeurs

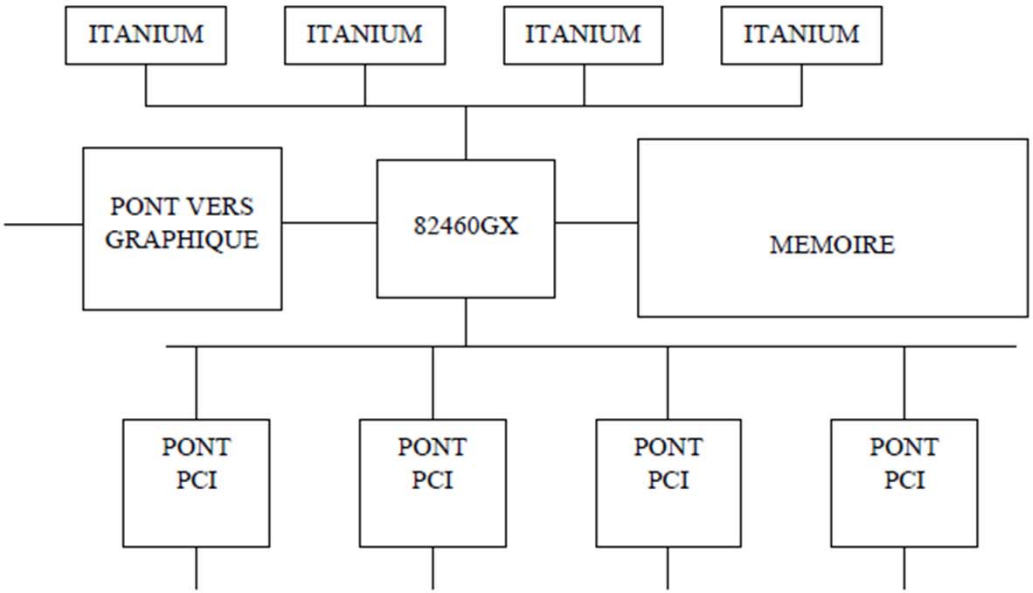
- **L'adaptation d'un système à l'environnement SMP pose les problèmes suivants :**
  - **Communications inter –processeurs**
    - ✓ Un système de communication efficace doit permettre à un processeur de communiquer avec un autre processeur (désigné) ou bien avec l'ensemble des processeurs.
  - **Pour l'utilisation des verrous il faut prendre en compte les problèmes suivants** :
    - ✓ L'attente active du thread désirant obtenir le verrou.
    - ✓ Ou bien l'attente passive.
    - ✓ Veiller au mécanisme d'évitement d'étreinte fatale.
  - **Cette liste de problèmes n'est pas exhaustive, de nombreux autres éléments sont à prendre en compte processeurs**

- **La carte SHV (Standard High Volume) d'Intel, quadri – processeurs reposant sur le chipset MIOC d'Intel:**



- Le contrôleur mémoire et pont vers les entrées –sorties régit l'ensemble des transferts.
- Le pont échangeur vers PCI PXB (PCI exchange) contrôle soit un bus PCI 64bits, soit 2 bus PCI 32 bits.
- Le générateur de signaux lignes et colonnes pour la mémoire RGC (RAS / CAS Generator).
- Le multiplexeur de chemin de données MUX.

- **La même architecture existe pour l'Itanium (IA – 64) d'Intel. Celle – ci repose sur le chipset 82460GX optimisé pour les systèmes à base de 4 processeurs**



- Système composé de **4 processeurs Itanium** partageant le même bus.
- Le chipset 82640GX d'Intel est l'élément central de cette architecture : il contrôle la mémoire et les E/S par l'intermédiaire de puces servant de relais.
- Au delà de 4 processeurs, les constructeurs devront développer leurs propres puces spécifiques.

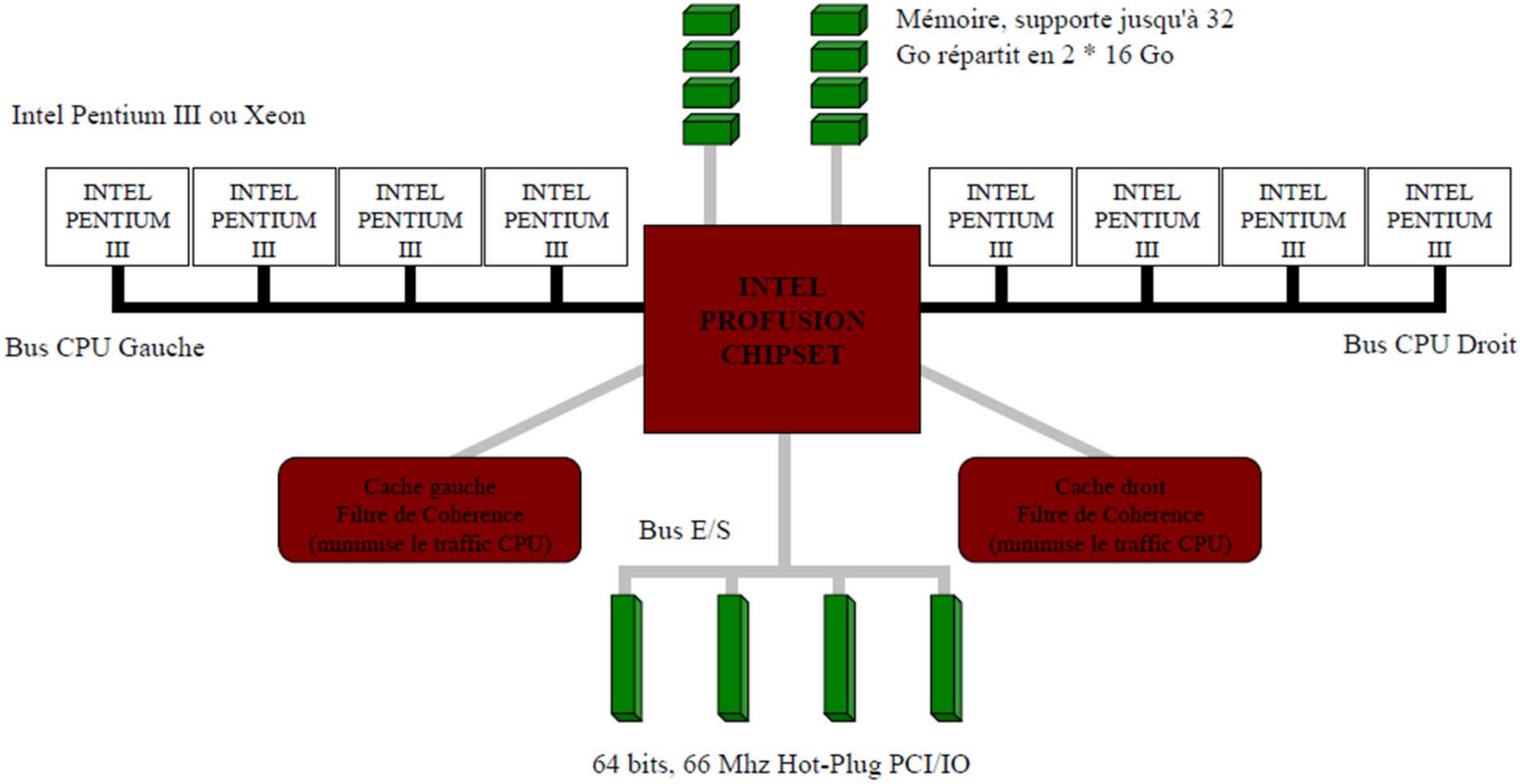
- **L'extension du nombre de processeurs peut se faire de façon simple en interconnectant deux sous – systèmes multi – processeurs. Trois types de structures existent :**

1. La structure de type **Profusion**, proposée par Intel, dans laquelle un composant spécifique permet l'interconnexion de deux sous –systèmes quadri –processeurs.

2. La structure où les sous –**systèmes quadri –processeurs** sont interconnectés au moyen d'un cache de troisième niveau. Le système Express 5000 HV8600 de Nec est un exemple de cette approche.

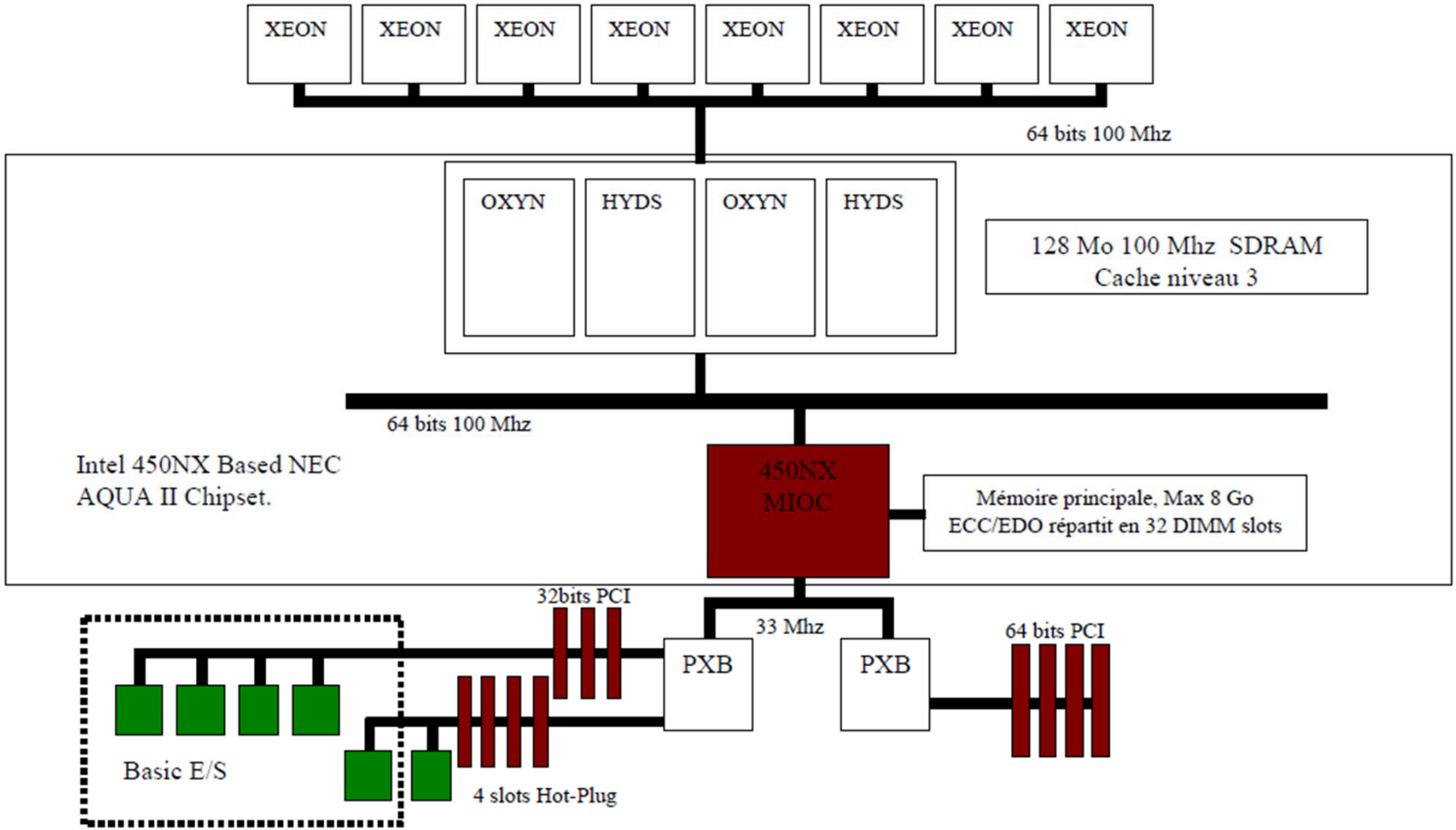
3. La structure dite **cross –bar interconnecte** deux ensembles de quadri –processeurs au travers de deux puces.

- Type **Profusion :**



- Type **Profusion :** Cette architecture repose sur une puce spécifique (le contrôleur Profusion) qui :
  - **Supporte deux sous ensembles quadri –processeurs.**
  - **Intègre deux fonctions :**
    1. ***Contrôleur de mémoire*** supportant deux sous ensembles de puce SDRAM (Synchronous Dynamic Access Memory).
    2. ***Contrôleur de cohérence*** utilisant deux répertoires permettant de minimiser le trafic de cohérence de cache sur les bus processeurs :
       a. Dans chacun des deux sous –ensembles, le protocole de cohérence est assurée par **l'espionnage.**
       b. Le contrôleur mémoire de Profusion propage d'un ensemble à l'autre, par l'intermédiaire des deux filtres les transactions d'un segment de bus si nécessaires.
       c. Un filtre de cohérence mémorise les informations qui sont placées dans les caches des processeurs d'un segment de bus. Si une information n'est pas placée dans l'un des caches des processeurs d'un segment, il n'est pas nécessaire de propager vers ce segment les transactions qui concernent cette information. A l'inverse, si la transaction concerne une donnée placée dans l'un des caches de l'autre segment, la transaction est propagée sur ce segment.
  - Permet de s'interfacer avec un environnement PCI 64 bits.

❑ Type cache troisième niveau:

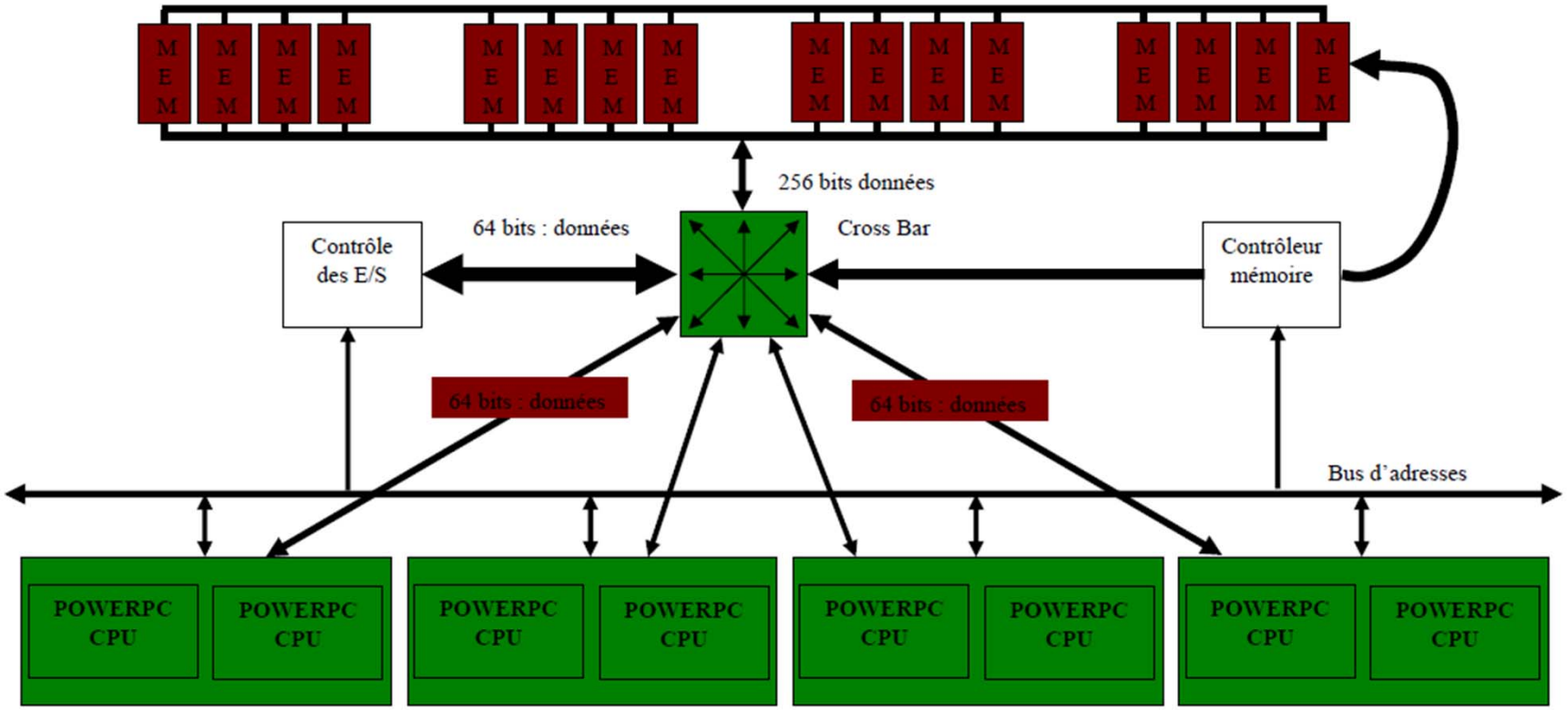


- **Cette architecture utilise :**
  - Un bus processeur et un bus système.
  - Un cache de niveau 3 de 128 Mo interposé entre les deux bus.
  - L'ensemble des composants Intel 450NX PCI Set pour le support de la mémoire et des entrées –sorties.

- L'ensemble de composants contrôlant le cache de niveau 3 est appelé **AQUA II** comprenant des composants désignés par **HYDS** et **OXY** (Hydrogène et Oxygène). **OXYN** est le contrôleur de cache du niveau 3 et **HYDS** est tampon d'accès au cache.

- En présence de la donnée dans le cache, **HYDS** fournit la donnée si nécessaire. En cas d'absence de la donnée, la demande est transmis à la mémoire.

Type cross bar :



- L'objet du **Cross – Bar** est de fournir un chemin indépendants entre tous les éléments du système.
- Les problèmes posés par cette approche se situent au niveau du support d'un nombre **importants de chemins simultanés** et du nombre de broches de la puce (le nombre de broches du Cross – Bar étant directement proportionnel au nombre d'éléments connectés).

Type cross bar :

- Cette architecture a été implémenté par Bull dans le système Escala. Elle a pour objectif de diminuer les contentions processeurs.
  - Le transfert de données effectué simultanément entre les différents composants.
  - Le bus n'est plus utilisé que pour communiquer les adresses associées aux opérations.
- Elle permet de s'affranchir des problèmes de contention entre les processeurs lors des accès mémoire pour les transferts de données importants.

On distingue deux familles de grands SMP :

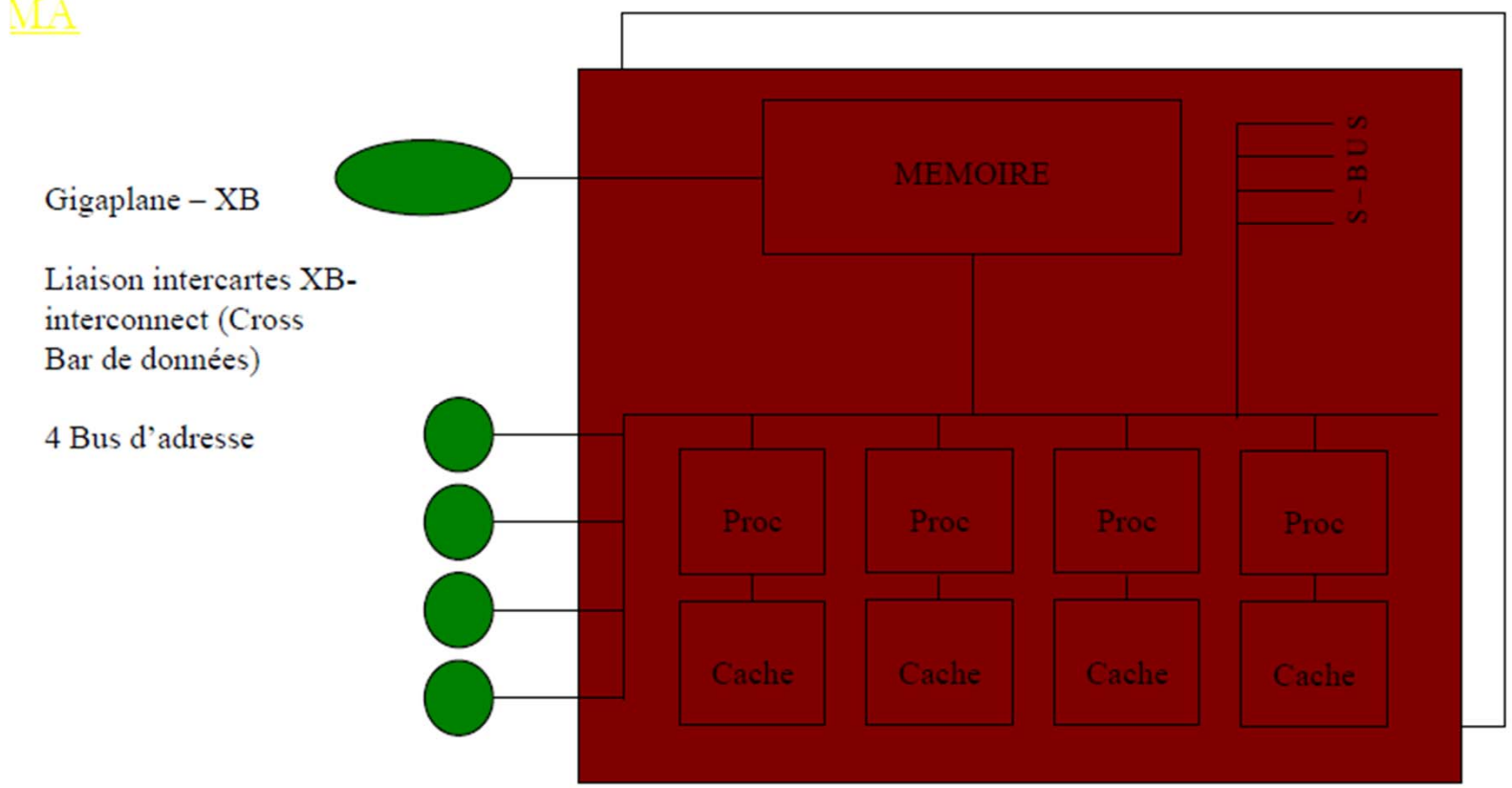
1. UMA ou proche UMA (Uniform Memory Access).

- Ces systèmes ont des temps d'accès mémoire (à partir de n'importe où dans le système) qui ne varie pas plus du double. C'est-à-dire si qu'entre le temps d'accès le plus court et le temps d'accès le plus long, il n'y a pas de rapport supérieur à 2.

2. CC –NUMA (Cache Coherent–Non Uniform Memory Access)

- Lorsque le rapport entre les deux temps d'accès est important, supérieur à 2, on dit que de telles architectures sont **CC – NUMA.**

❑ architectures UMA :

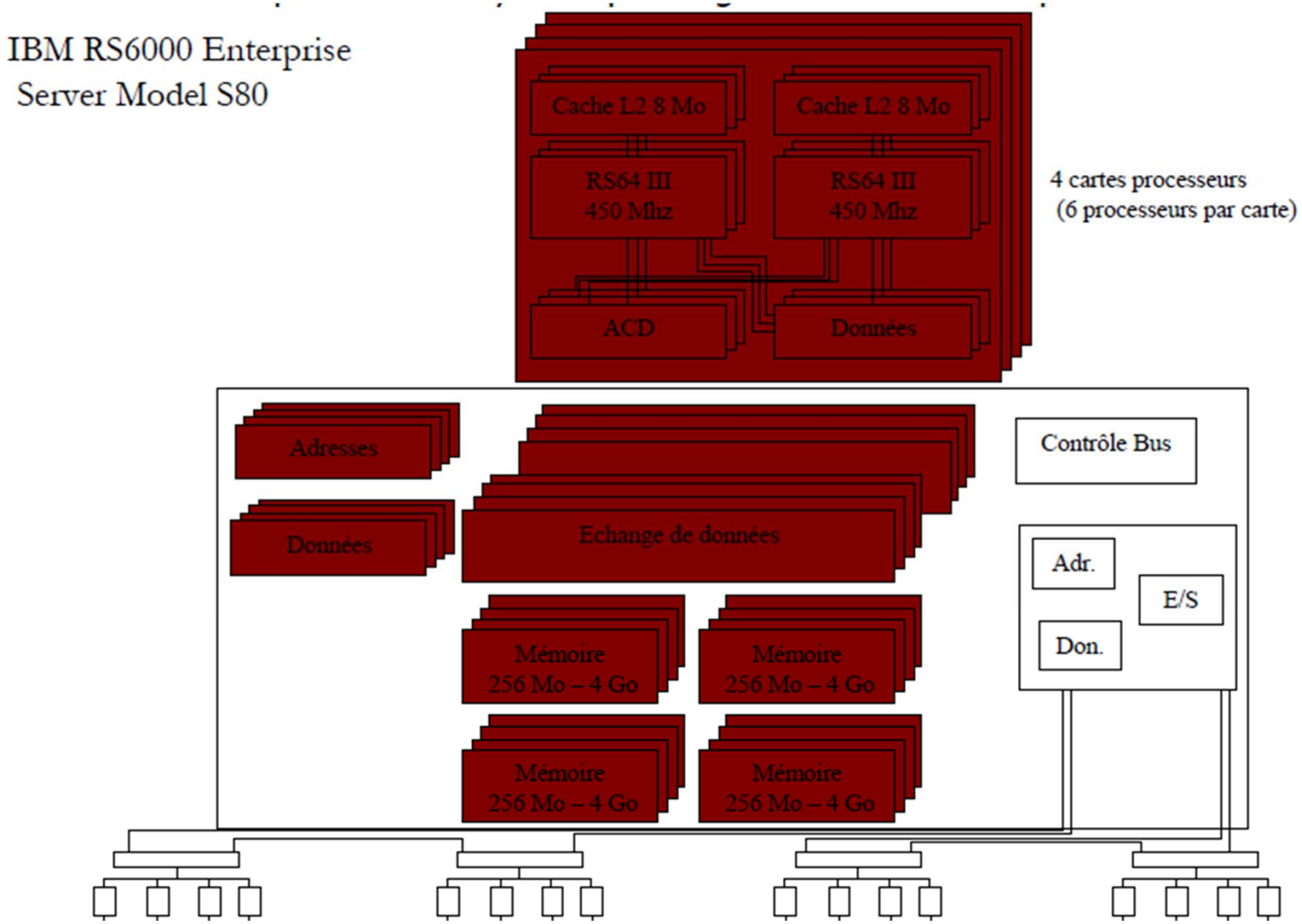


Le système SUN ULTRA Enterprise 10000 (E 10000) utilise comme élément constitutif des cartes quadri – processeurs.

architectures UMA :

- **Chaque carte :**
  - Comprend une mémoire locale.
  - Une interface d'entrées –sorties permettant d'accueillir jusqu'à4 contrôleurs S-BUS
- Ces cartes sont interconnectées au moyen d'un Cross – Bar pour les données et de quatre bus d'adresse. Le débit global est de 12 Go/s.
- Cette architecture est considérée comme quasi UMA.

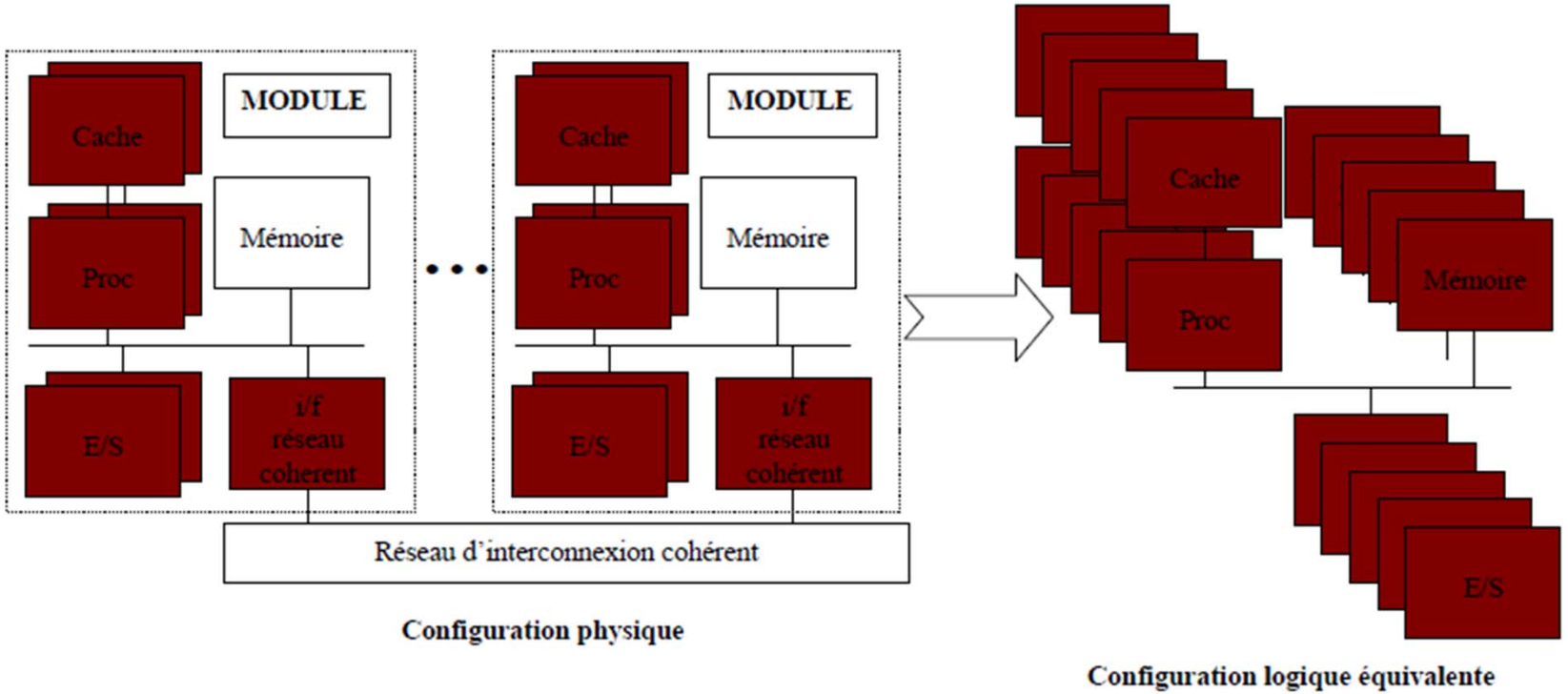
- Multi – processeurs symétriques à grands nombres de processeurs



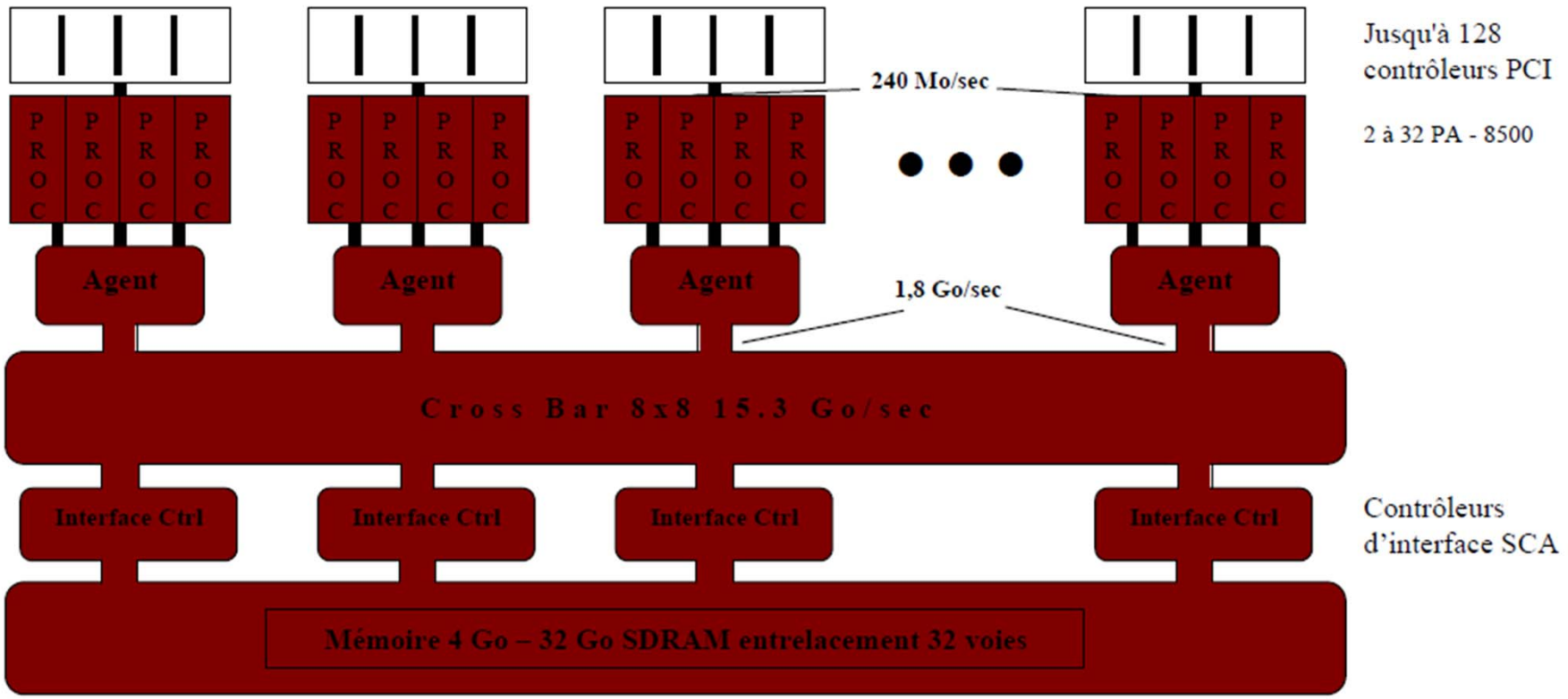
- Ce système utilise des processeurs **PowerPC RS64 – III** cadencé à 450 Mhz qui intègre 128 ko de **mémoire cache.**
- Chaque carte processeur **comprend six processeurs**, chacun ayant un cache de 8 Mo.
- Les bus internes aux cartes processeurs fonctionnent à 150 Mhz et offrent un débit de **2,4 Go/s**, soit pour l'ensemble des bus un débit global de **24 Go/s.**
- Dix bus de données sont utilisés par paire, quatre paires de bus sont utilisées pour la liaison entre les cartes processeur et le contrôleur mémoire, une paire est utilisée pour les entrées – sorties (ils ne sont pas représentés sur la figure).
- Le contrôleur de mémoire possède quatorze ports de données, dix pour les échanges avec les processeurs et les entrées – sorties, les autres pour les échanges avec la mémoire.
- La capacité maximale de la mémoire **est de 64 Go.**
- Ce système a été conçu pour tolérer **des défaillances au niveau de la mémoire** en utilisant un **concept de module de remplacement (voir chapitre : Haute disponibilite)**

- **Architecture NUMA**
  - La mémoire est physiquement répartie sur des cartes différentes constituants le système.
  - Un processeur a des temps d'accès différents aux mémoires en fonction de la localisation des mémoires.
- Si les accès à ces différentes mémoires obéissent à un protocole de cohérence de cache, on dit que l'architecture est du type **CC – NUMA** (Cache Coherent Non – Uniform Memory Access).
- Seules les architectures supportant le **CC – NUMA** au niveau matériel sont intéressantes pour les SMP (le support au niveau logiciel dégrade les performances).
- **Par exemple :**
  - Le module de base d'une carte quadri – processeurs munie de mémoire et de ses propres entrées – sorties, le moyen d'interconnexion des modules doit supporter les mécanismes de cohérence.

Multi – processeurs symétriques CC - NUMA

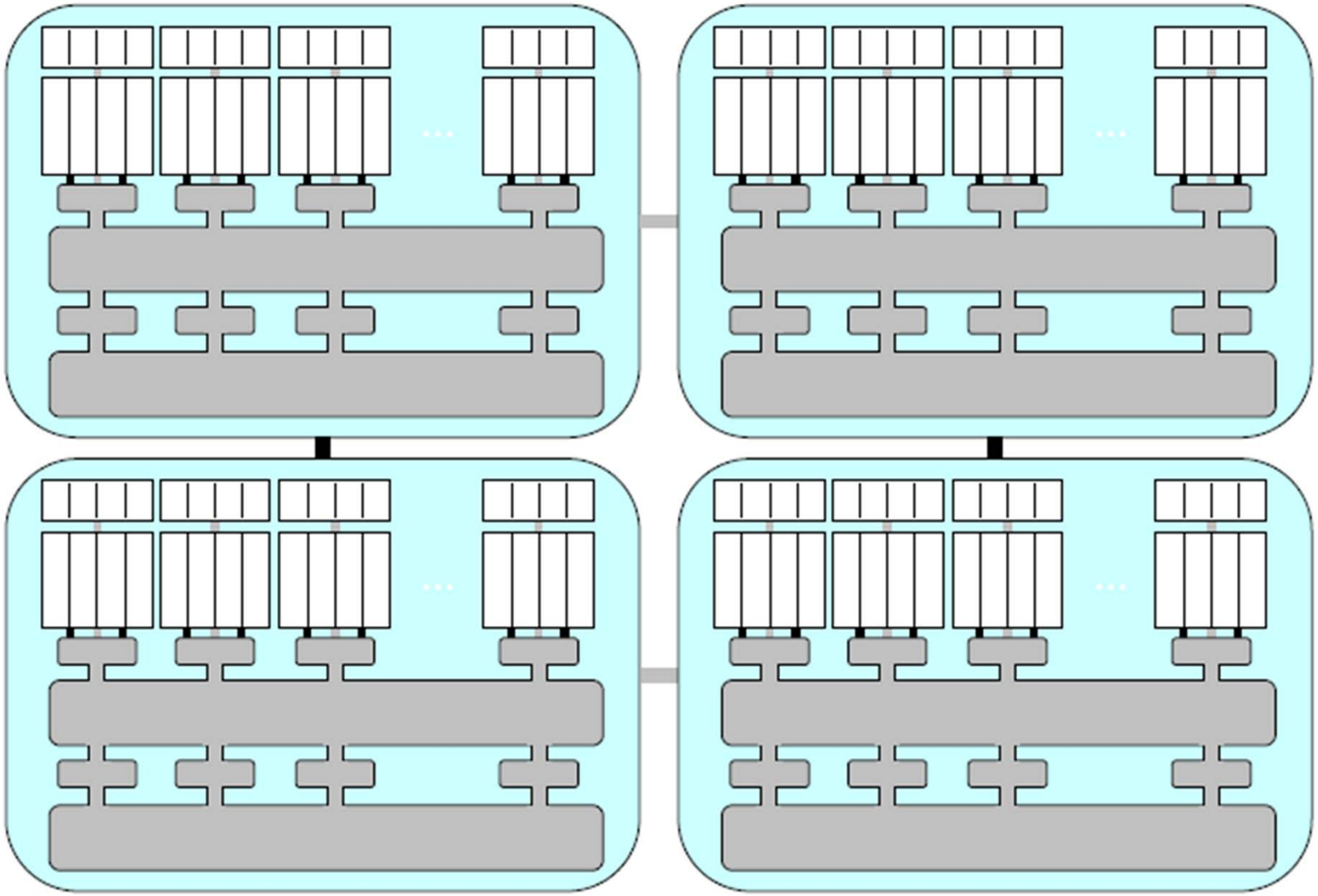


- Un processeur d'un quelconque module peut accéder soit à la mémoire locale (du module), soit aux autres mémoires sur les autres modules.
- Il y a une différence de performance entre ces deux accès.
- Le rapport entre le temps d'accès à la mémoire locale et le temps d'accès aux mémoires distantes est appelée **NUMA Factor.**
- Plus ce rapport est élevé, plus les logiciels systèmes et **SGBD** doivent tenir compte de ces caractéristiques en intégrant les optimisations permettant de rendre la performance du système peu sensible à la localité des références à la mémoire.
- L'objectif des concepteurs est de minimiser ce rapport.
- On considère qu'une valeur de 5 pour le **NUMA Factor** est préjudiciable au bon fonctionnement d'un système.



Module de base du HP 9000 V2500 Entreprise Server

- Supporte de 2 à128 processeurs, le module de base ayant 32 processeurs.
- Les modules de base sont fondés sur un Cross –Bar.
- Les modules de base sont reliés par un réseau d'interconnexion permettant de supporter 4 modules



- Le système est réalisé par l'interconnexion de plusieurs modules.
- Quatre modules peuvent être interconnectés permettant le support de 128 processeurs.
- La liaison entre les modules est réalisé au moyen de liaisons unidirectionnelles point à point entre les contrôleurs d'interface des différents modules (non représentées sur la figure).

Introduction aux clusters

❑ Qu'est ce que le terme Cluster veut dire ?

⇒ Un cluster peut être défini comme un ensemble limité(de l'ordre de la dizaine) de systèmes informatiques interconnecté qui partagent des ressources de façon transparentes.

⇒ Ces ressources partagées sont désignées comme «ressources clusterisées».

⇒ Chacun de ces systèmes peut être considéré comme un système à part entière :

⇒ Il dispose d'un ensemble complet de ressources : mémoire, disques, processeurs etc…

⇒ Les systèmes qui composent un cluster sont appelés nœuds.

⇒ Les clusters sont aussi désignés sous l'appellation de systèmes faiblement couplés car ils ne partagent pas de mémoire.

⇒ Tandem a introduit le concept de cluster à la fin des années 1970 et Digital l'a repris avec le VAX –VMS (VaxCluster) à partir de 1983.

❑ Un Cluster n'est pas un système distribué :

⇒ Un cluster est composé de nœuds similaires. C'est-à-dire provenant d'un même fournisseur, correspondant à des configurations précisément définies et fonctionnant sous le contrôle d'une même version du système d'exploitation.

⇒ Les clusters offrent aux utilisateurs l'illusion qu'ils sont en présence d'un système unique.

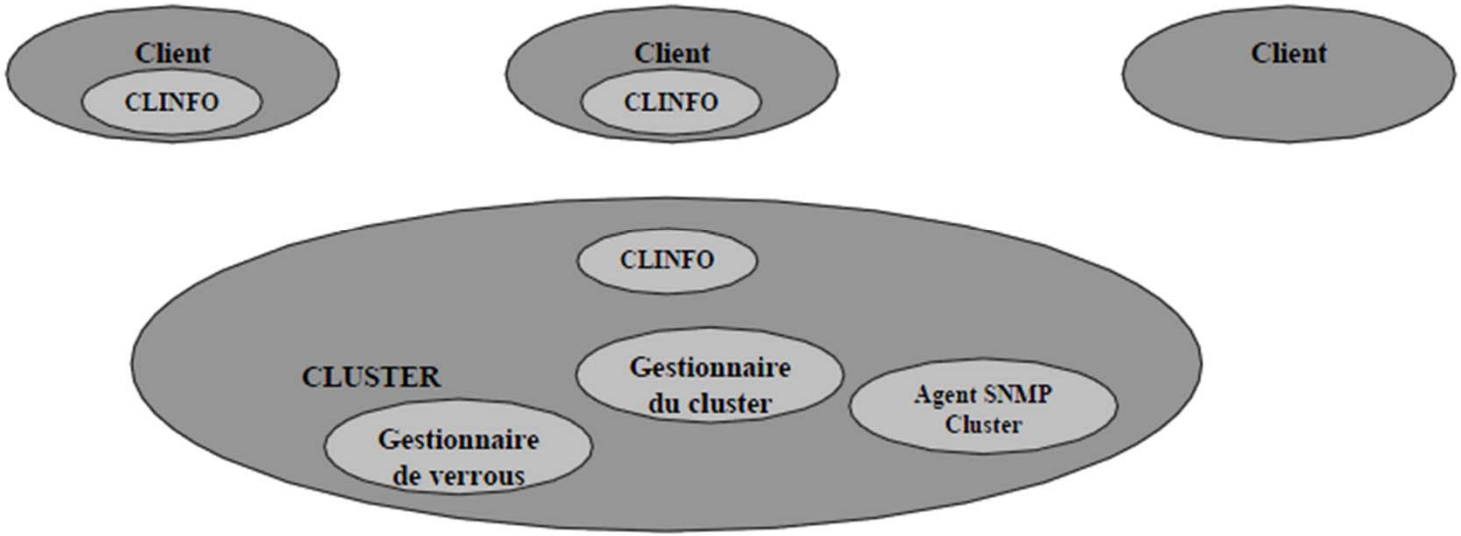
⇒ Les différents nœuds qui composent le cluster sont proches les uns des autres.

⇒ La synchronisation entre les nœuds est assurée au moyen d'un gestionnaire de verrous distribués, appelé **DLM (Distribued Lock Manager).**

⇒ Le système d'exploitation d'un cluster repose soit sur un système spécialement conçu pour cela (cas de Tandem), soit sur un système dérivé d'un système existant.

L'avantage principal est la haute disponibilité parce qu'ils possèdent des caractéristiques de redondance et de résistance aux défaillances (en particulier défaillance logiciel).

Architecture très générale du Cluster HACMP/6000 d'IBM



❑ Architecture très générale du Cluster HACMP/6000 d'IBM :

⇒ **Le gestionnaire du cluster** : Actif sur tous les nœuds du cluster, il se charge de maintenir à jour la configuration du cluster. Son rôle consiste aussi à refléter l'état de fonctionnement du cluster vis-à-vis des autres systèmes.

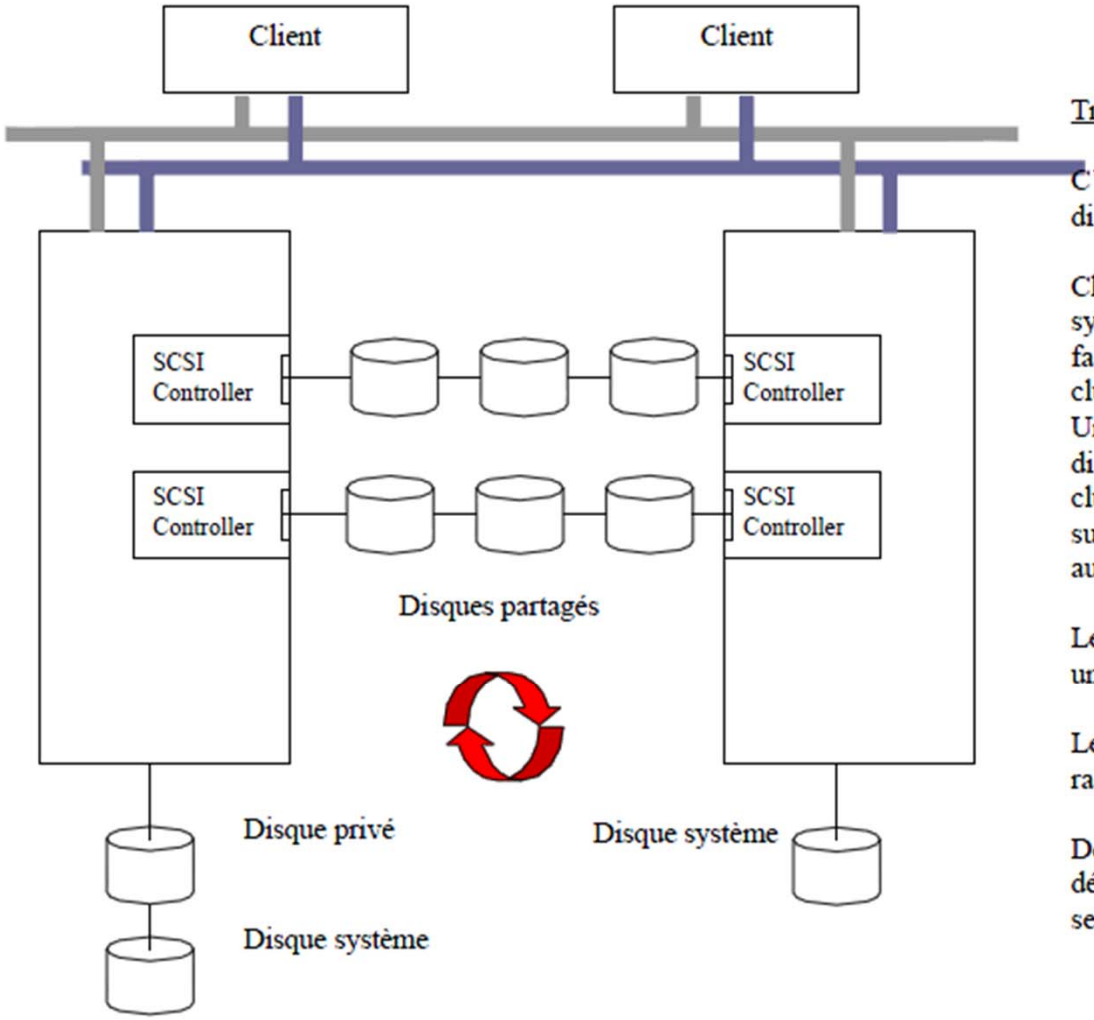
⇒ **CLINFO** (Cluster Information Service). C'est un composant optionnel tant sur le cluster que sur les clients. Il donne des informations aux clients sur l'état du cluster au moyen d'une API.

⇒ Ces informations sont communiquées aussi aux agents **SNMP.**

⇒ L'agent **SNMP,** cluster **SNMP** (Simple Network Management). C'est un produit standard de fait pour l'administration des systèmes distribués dans le monde Unix. Le gestionnaire de cluster lui transmet des informations qu'il rend accessibles à travers **SNMP.**

⇒ Le gestionnaire de verrous fournit un service de synchronisation distribué aux différents services et aux applications fonctionnant sur un cluster.

TruCluster Software de Compaq:



TruCluster Software de Compaq

C'est typiquement un cluster à haute disponibilité.

Chaque nœud est équipé d'un disque système.Un nœud peut s'initialiser de façon autonome avant de rejoindre le cluster.
Un nœud peut aussi posséder ses propres disques qui ne seront pas récupérés par le cluster. Les différents nœuds exercent une surveillance réciproque les uns sur les autres.

Le cluster se gère comme un système unique.

Le système de fichiers clustérisé avec racine unique partagée.

Domaine unique de sécurité, la désignation d'un utilisateur se fait une seule fois.

- Une machine massivement parallèle (MPP pour Massively Parallel Processing) possède un ensemble important (plusieurs centaines) de systèmes appelés nœuds reliés par un réseau d'interconnexion.
- Chaque nœud dispose de ses propres ressources, processeurs, mémoire, contrôleurs d'entrées – sorties et de sa propre copie du système d'exploitation. Chaque nœud est donc totalement indépendant.
- Les différences avec les clusters sont :
  1. Le nombre maximal de nœuds, une dizaine pour un cluster, plusieurs centaines pour les MPP.
  2. L'organisation physique des systèmes, conçue pour supporter un grand nombre de noeuds et en faciliter l'ajout.
  3. Le réseau d'interconnexion spécifique à haute performance, alors que pour les clusters il s'agit plutôt de FDDI ou ethernetà100 Mb.
- Les nœuds peuvent être des SMP, ils sont alors qualifiés de nœuds multi – processeurs.
- Les applications sont développés spécifiquement pour ce type d'architecture. Elles visent surtout le calcul numérique intensif.

- Ce type machines concurrencent directement les machines vectorielles (CrayNec).

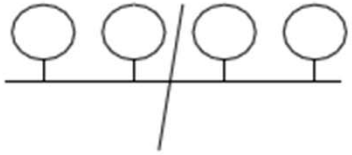
- **Actuellement :**
  - Développement de MPP pour les systèmes d'aide à la décision.
  - Les SGBD ont été adaptés au traitement parallèle pour les grandes bases de données.
- Il est impératif que le réseau d'interconnexion soit performant car une application MPP comprend :
  - Des phases de communication (notamment les données pour les processus qui s'exécutent en parallèle).
  - Des phases de calcul.

La performance du réseau est donc fondamentale pour la performance des applications.

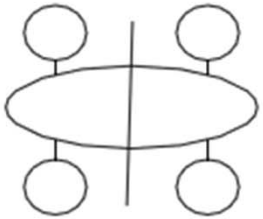
Performance des réseaux d'interconnexion :

- La topologie du réseau.
- Le nombre maximal de nœuds supportés.
- La performance (latence et bande passante).
- La simplicité et la généralité de l'interface matériel –logiciel.
- Le caractère bloquant ou non bloquant du réseau (ici on s'intéresse uniquement aux réseaux non bloquant).
- Le coût.
- La résistance aux défaillances, en général les réseaux sont redondants. En fonctionnement normal, deux réseaux sont utilisés, chacun s'occupe de 50% de la charge, en cas de défaillance de l'un des réseaux, la totalité du trafic est assurée par le réseau valide.
- La conformité à un standard ou la possibilité d'en devenir un

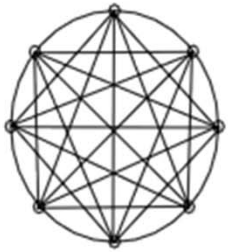
Exemple de topologie d'interconnexion :



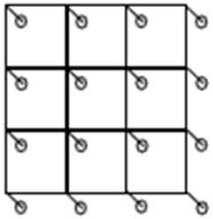
BUS



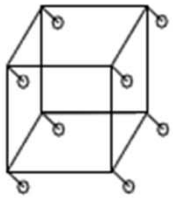
ANNEAU



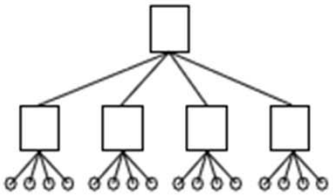
Complètement connecté



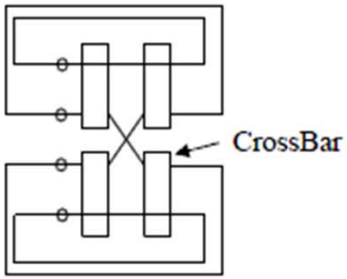
Grille 2D



Grille 2D



Hiérarchisé



Réseau Oméga

❑ Paramètres de caractérisation de la performance d'un réseau d'interconnexion :

1. La latence :

⇒ Le temps nécessaire à l'acheminement d'un message depuis l'espace d'adressage d'un processus jusqu'à l'espace d'un autre processus.

2. Le débit ou bande passante :

- Le nombre d'octets communiqués par unité de temps.
- On distingue :
  - **La bande passante totale**: il s'agit de la bande passante d'un lien multipliée par le nombre de liens existant dans le réseau d'interconnexion. La bande passante totale suppose que les nœuds exploitent en totalité les différents liens et qu'il n'y ait aucun conflit sur ces liens.
  - **La bande passante réellement utilisable:** c'est la moyenne passant par les différents nœuds, la notion de **bissection bandswitch** est utilisée pour définir

- ***La bissection bandswitch :***
  - **Pour un réseau symétrique,** c'est la bande passante observée sur une coupe en deux du réseau d'interconnexion.
  - **Pour un réseau dissymétrique**, c'est la bande passante minimale observée sur l'ensemble de coupes du réseau.

- **Un réseau d'interconnexion parfait :**
  - Une latence constante (indépendante du nombre de nœuds). Le temps nécessaire à l'acheminement d'un message entre deux nœuds quelconques est indépendant du nombre de nœuds du système.
  - Une bissection croissant linéairement avec le nombre de nœuds. L'ajout d'un nœud dans le système apporte alors une contribution constante à la bande passante.

- **Remarque :**
  - Pour les clusters, le réseau est souvent fondé sur les technologies de réseau local, la latence augmente avec le nombre de nœuds car il y a contention tandis que le débit global est constant. Alors que dans le cas des MPP, le réseau d'interconnexion cherche à se rapprocher des caractéristiques idéal : latence constante et débit dépendant linéairement du nombre de nœuds.

Caractéristiques comparées de quelques topologies de réseaux d'interconnexion

Critères		Bus	Anneau	Grille 2D	Hypercube	Complètement connecté
Bande Passante	Bande Passante Totale	1	64	112	192	2016
	Bissection	1	2	8	32	1024
Coût	Port par Switch	Pas applicable	3	5	7	64
	Nombre total de liens	1	128	176	256	2080

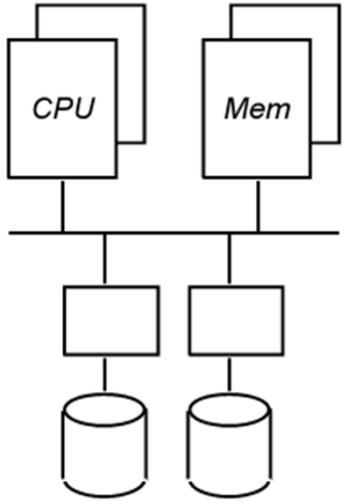
- Un réseau complètement connecté possède une topologie telle que tout nœud possède un lien dédié avec tous les autres nœuds du système.
- Le tableau met en évidence la différence importante qui existe entre la bande passante totale et la bissection.
- Le choix d'une topologie résulte d'un compromis entre sa bande passante et son coût.

LES SERVEURS ET LES SGBD

Une autre vision de l'architecture en relation avec le support des SGBD

- En matière de liaison avec les disques, on distingue ***trois grandes classes d'architectures.***

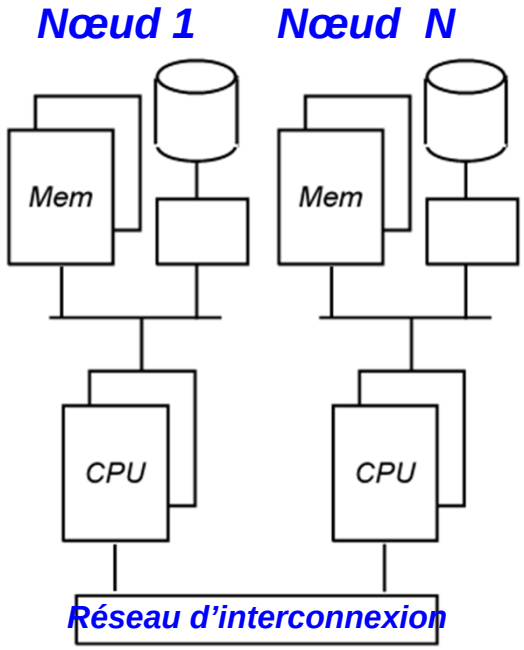
1. Architecture **Share *Everything***, ou « tout partage » : dans cette architecture :
   - L'ensemble des processeurs du système fonctionne sous le contrôle d'un ***seul système d'exploitation***.
   - Une opération d'entrée-sortie peut être initialisée ***par n'importe lequel des processeurs.***
   - Ce modèle est caractéristique de l'architecture ***SMP*** (multiprocesseur symétrique).



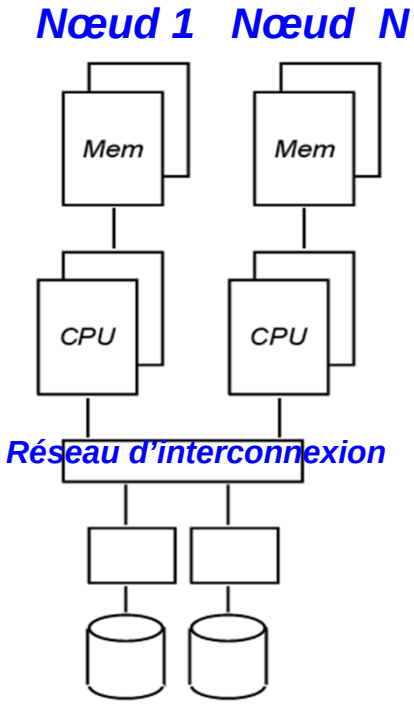
2. Architecture **Share *Nothing***, ou sans partage : dans cette architecture

   – Chacun des nœuds qui composent le système fonctionne sous le contrôle de sa propre ***copie du système d'exploitation*** et a un ***accès exclusif*** aux disques qui lui sont attaches.

   – Ce modèle est typique des MPP, et on le rencontre aussi sur certains clusters.



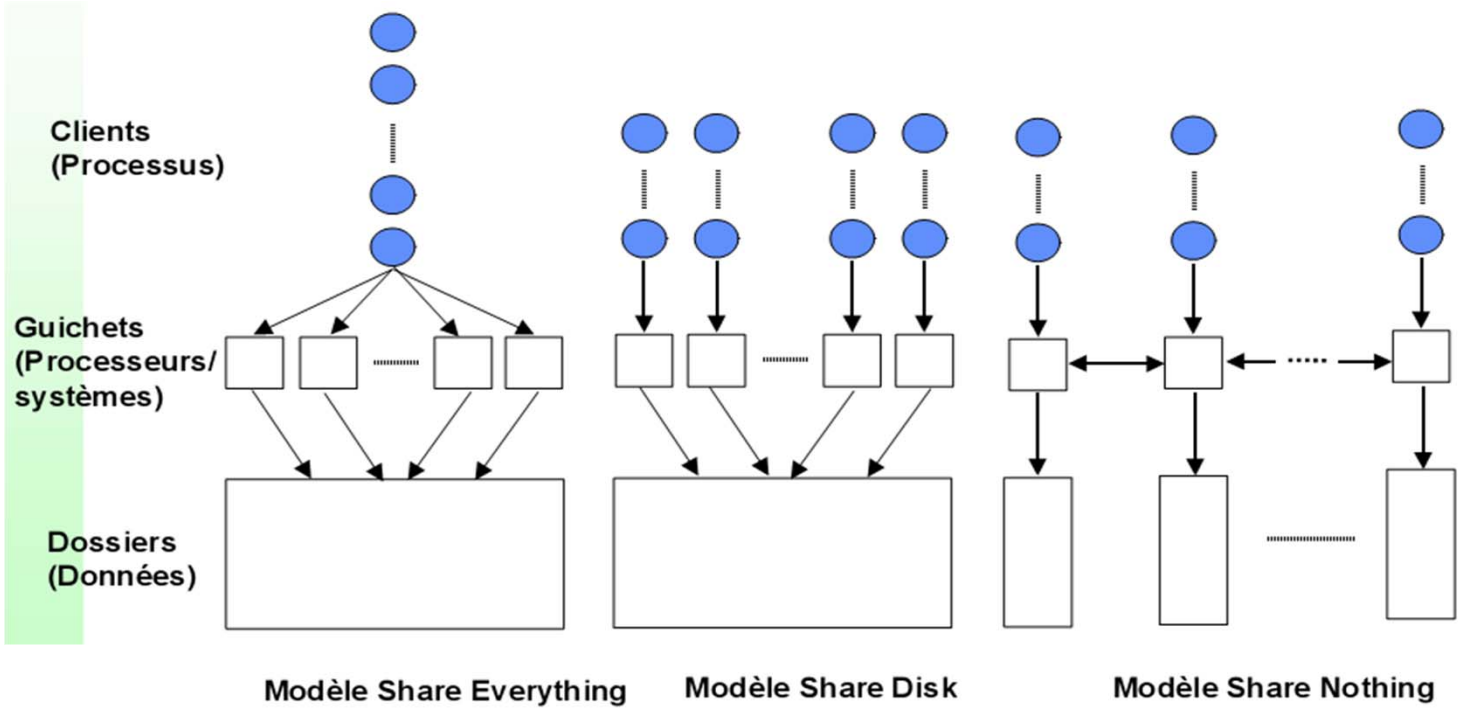
3. Architecture **Shared Disks**, ou disques portagés : dans cette architecture
   - chacun des nœuds qui composent le système ***fonctionne sous le contrôle de sa propre copie du système d'exploitation*** mais peut accéder ***directement aux disques qui sont partagés*** entre les différents nœuds.
   - On rencontre ce modèle sur certains clusters.



Relation SGBD – Architecture Share -1-

- Ces trois modèles correspondent a des réalités différentes du point de vue du ***partage des données*** et de ***l'équilibrage des charges.***
- Une analogie avec les **clients** attendant d'être servis devant des **guichets** permet d'appréhender ***les différences entre les modèles.***

- La Figure ci-dessous illustre les différences entre les modèles au moyen de cette analogie.

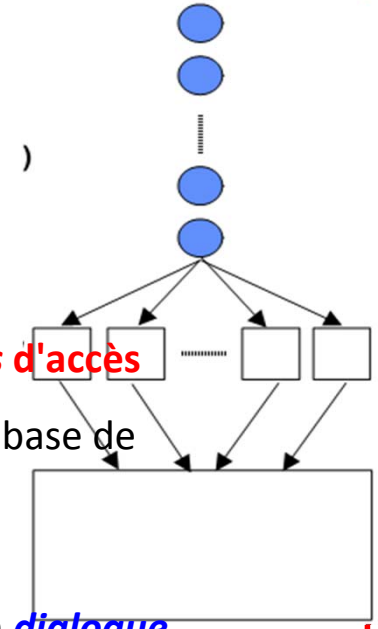


- Cette analogie est fondée sur des **clients** (***processus ou threads***) s'adressant a des **agents** occupant des **guichet**s (***processeur***s) afin d'obtenir ***un service.***

- Ces agents doivent accéder à des **dossiers** (***bases de données***), qui contiennent les renseignements nécessaires au traitement des requêtes des clients.

Relation SGBD – Architecture Share -3-

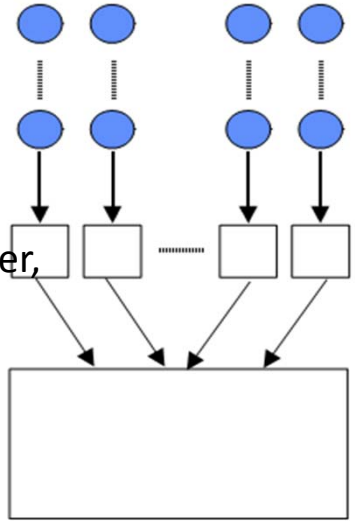
- Dans le modèle *Share Everything, les* clients sont placés dans une **file d'attente commune.**
- Dès qu'un guichet **se libère**, le premier client de la file d'attente se présente a l'agent qui se charge de traitement de sa requête.
- Nous voyons qu'il y a ***un équilibrage de charge naturel***, les clients (processus) se répartissant automatiquement sur les guichets (processeurs) en fonction de leur disponibilité.



- Tous les agents des guichets (***processeur***s) disposent de ***possibilités identiques*** **d'accès aux dossiers,** et l'ensemble des dossiers est partagé par l'ensemble des agents (la base de données est partagée par l'ensemble des processeurs).
- La synchronisation entre les agents pour la mise à jour des dossiers s'opère par un ***dialogue direct*** (les processeurs partagent une même mémoire).
- La limite du système se trouve dans :
  - Le ***nombre maximal d'agents*** qui peuvent être mis en parallèle
  - Et dans le ***débit de l'accès aux données*** (nombre maximal de processeurs en SMP et débit des entrées-sorties avec les disques).

Relation SGBD – Architecture Share -4-

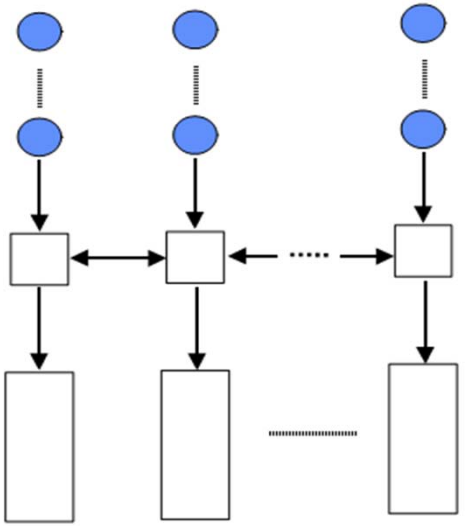
- Dans le modèle ***Shared Disks***, la répartition des clients vers les différents ensembles de guichets (les différents nœuds qui composent le système) ***ne se fait pas naturellement.***



- Il est nécessaire de prévoir ***un agent en amont***, chargé d'équilibrer les flux de clients vers les différents guichets.
- Dans les systèmes informatiques, ce rôle est dévolu **a l'un des nœuds** du cluster, qui recueille des informations sur **la charge des nœuds** et aiguille les requêtes **vers les nœuds les moins chargés**. ***Chacun des agents (les processeurs du système) a accès a l'ensemble des dossiers.***
- La ***synchronisation*** entre les agents pour la mise a jour des données nécessite un dialogue ***entre les différents guichets***.
- Dans le cas des systèmes informatiques, ce dialogue se réalise au moyen du réseau d'interconnexion, et il est bien moins efficace que le dialogue a travers une mémoire commune.

Relation SGBD – Architecture Share -5-

- Dans le modèle ***Share Nothing***, chaque ensemble de guichets accède en propre a un sous-ensemble des dossiers.
- Un guichet désirant accéder à ***des dossiers*** gérés par des guichets appartenant a un autre sous-ensemble doit s'adresser à ce sous-ensemble.
- Pratiquement, ce modèle implique que la répartition des clients sur les sous-ensembles de guichets soit faite *en fonction du sous-ensemble de dossiers auquel le client souhaite accéder.*



- On voit aisément que l'équilibrage de charge est étroitement lié
  - ⇨ à la répartition des dossiers entre les différents sous-ensembles de guichets
  - ⇨ et a la distribution des demandes des clients vis-à-vis de ces données.
- L'accès d'un sous-ensemble de guichets de donnée a des données gérées par d'autres sous-ensembles de guichets nécessite des ***échanges entre ces sous-ensembles.***

- ***Une telle demande peut revêtir deux formes :***

1. ***Envoi de données (Data Shipping) :*** Le guichet traitant la requête demande au sous-ensemble de guichets possédant la donnée de la lui adresser. Ce guichet se charge de la réalisation de l'opération et du renvoi des données (en cas de modification) au sous-ensemble de guichets qui gère ces données.

2. ***Envoi de fonction (Function Shipping) :*** Le guichet traitant la requête adresse au sous-ensemble de guichets possédant les données la définition de la fonction a exécuter. Ce dernier retourne, le cas échéant, les résultats de la fonction qu'il a exécutée.

- L es SGBD qui ont fait le choix de l'approche ***Shared Disks*** a opté pour **l'option «envoi de données »,**
- Les autres SGBD, qui ont fait le choix ***Share Nothing***, ont opté pour l'option ***« envoi de fonction ».***
- Ces options impliquent des échanges sur le réseau d'interconnexion, ce qui constitue une source d'inefficacité.

Caractéristiques comparées des différents modèles

	Share everything	Shared disks	Share nothing
AVANTAGES	⇨ Simplicité pour parallélisme inter requêtes et intra requête ⇨ Bonne utilisation des ressources ⇨ Equilibrage de charge naturel ⇨ Communication inter processeurs efficace (car fondée sur une mémoire partagée cohérente) ⇨ Solution en cours de banalisation en bas de gamme"	⇨ Bonne disponibilité ⇨ Bonne scalabilité (100 processeurs et au-delà) ⇨ Faible Coût du fait de la réutilisation de composants standard ⇨ Bon Équilibrage de charge (les données a fort taux de partage peuvent être répliquées)	⇨ Bonne disponibilité ⇨ Très bonne scalabilité (plusieurs centaines de processeurs) ⇨ Faible coût du fait de la réutilisation de composants standard
INCONVENIENTS	⇨ Disponibilité du système difficile à assurer ⇨ Scalabilité (quelques dizaines de processeurs)	⇨ Interaction entre les nœuds pour la synchronisation des mises à jour des données ⇨ Saturation du réseau d'interconnexion par les transferts entre nœuds et disques ⇨ Coût du maintien de la cohérence de copies multiples (si réplication) en particulier si les mises a jour sont fréquentes	⇨ Equilibrage de charge difficile ⇨ Difficile a administrer et a optimiser du fait du partitionnement des données ⇨ Forte dépendance des performances vis-à-vis des caractéristiques du réseau d'interconnexion ⇨ Coût de la parallélisation, même pour des requêtes simples ⇨ Coût du maintien de la cohérence de copies multiples (si réplication), en particulier si les mises a jour sont fréquentes

Problématique des SGBD parallèles -1-

Problématique des SGBD parallèles -2-

- **Deux types de composants du système concourent à la performance globale en environnement SGBD :**
  1. les ***ressources de traitement*** (processeurs)
  2. et les ***entrées-sorties*** (contrôleurs et unités de disque).
- **Les deux autres types d'éléments qui ont une influence sur la performance sont**
  3. les ***caractéristiques de la mémoire*** (capacité et temps moyen d'accès)
  4. et ***les communications***. Il s'agit ici des caractéristiques du réseau d'interconnexion (latence et débit) que l'on rencontre dans les clusters et les MPP et non du sous-système de communication avec les réseaux distants et les postes clients

Problématique des SGBD parallèles -3-

- On cherche a atteindre ***l'augmentation des performances*** (soit en débit, soit en temps de réponse) par l'augmentation du nombre de composants et leur exploitation en parallèle.
- la recherche d'un **degré élevé de parallélisme** peut avoir des effets adverses :
  - ⇨ coût de la gestion du parallélisme *(overhead),*
  - ⇨ dispersion des temps d'exécution
  - ⇨ et difficultés d'équilibrage de charge.

- Dans un système soumis à une **charge provoquant sa saturation**, nous pouvons observer deux types de comportement:
  1. ***Saturation des ressources de traitement :*** Cette situation est appelée **CPU bound**, parce que la performance du système est limitée par les processeurs.

  2. ***Saturation des entrées-sorties :*** Cette situation est appelée **I/O *bound***, parce que la performance du système est limitée par les entrées-sorties.

Problématique des SGBD parallèles -4-

- Pour le traitement des requêtes complexes, deux possibilités de parallélisations existent :
  1. ***Parallélisme de traitement :*** La requête est décomposée en requêtes élémentaires, qui sont exécutées en parallèle.
  2. ***Parallélisme de données :*** L'exécution de la requête s'opère en parallèle sur des sous-ensembles des données.

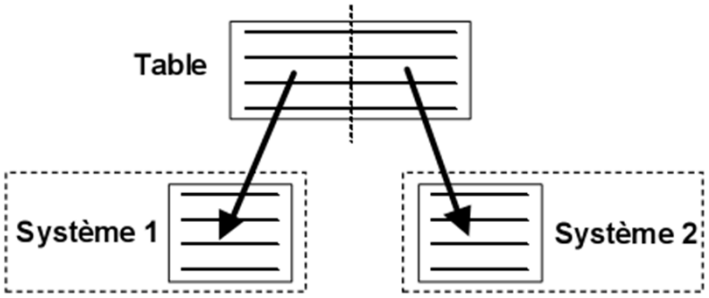
- Il est possible de mettre en oeuvre ces deux formes de parallélisme intra requête sur n'importe lequel des modèles d'architecture présentes a la section précédente : ***Share Everything, Shared* Disks ou *Share Nothing*.**

- **Le partitionnement des données est l'un des paramètres importants des SGBD parallèles.**
  - Il est particulièrement important dans le cas des architectures de type ***Share Nothing.***
  - Dans le cas des architectures **Shared Disks**, le partitionnement des données a une influence sur les performances du système si l'on arrive, par un aiguillage judicieux des requêtes vers les nœuds, **à créer une affinité nœuds-données qui favorise l'effet de cache**. Cela signifie que les mêmes données sont accédées par le même nœud et que le cache de la base de données placé sur ce nœud a une plus forte probabilité de contenir les données accédées par les requêtes.

- **Le partitionnement cherche a atteindre les buts suivants :**
  - ⇨ **Réduction de la charge système au niveau de l'accès aux données.**
    - **Plusieurs ressources parallèles sont mises au service de l'accès aux données en cherchant a éviter que les différentes applications fonctionnant sur le système accèdent aux même ressources.**
  - ⇨ **Equilibrage de charge. Les traitements sont repartis en fonction de la répartition des données sur les différents nœuds composant le système.**
  - ⇨ **Accroissement de la capacité de travail du système (du fait de la mise en parallèle de plusieurs nœuds).**
- **Il existe deux modèles de partitionnement des données**
  - ➢ **partitionnement vertical ;**
  - ➢ **partitionnement horizontal.**

Partitionnement et placement des données -3-

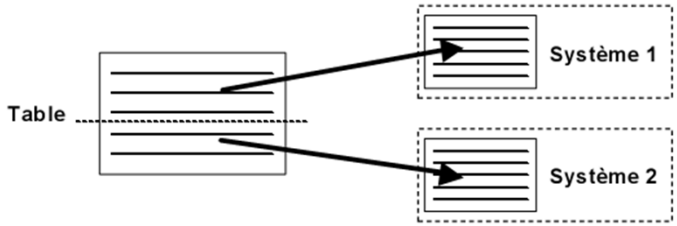
- **Partitionnement verticale :**
  - Avec le partitionnement vertical, les attributs d'une relation sont repartis entre plusieurs systèmes.
  - Ce type de partitionnement peut avoir des conséquences sur la programmation des applications.
  - ***Deux cas peuvent se présenter :***
    - Dans le cas où le SGBD supporte les bases de données distribuées, Il y a transparence vis-à-vis de l'application, le SGBD se chargeant des problèmes liés a la distribution.
    - En revanche, lorsque le SGBD ne supporte pas les bases de données distribuées, la programmation de l'application doit tenir compte de la partition des données.
  - *Dans tous les cas, les performances sont impactées en cas de mise a jour des données portant sur plus d'une partition, car Il est alors nécessaire de recourir au protocole de validation a deux phases.*



- ***Projection, dont un attribut commun, sur chacun des sites***
- ***Table globale reconstituée par une jointure selon cet attribut***

- ***Note : Relation avec l'organisation des applications (minimisation des interactions entre les systèmes et validation a deux phases)***

- **Partitionnement horizontale :**
- Le partitionnement horizontal ne présente pas les mêmes contraintes, et son utilisation ne pose pas de problème au niveau de la conception des applications

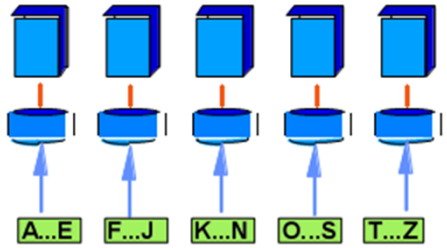


- ***Sélection, selon des valeurs disjointes d'un critère sur chaque site***
- ***Table globale reconstituée par UNION***

- ***Note : Prise en compte de la validation a deux phases par le SGBD***

Méthodes de partitionnement horizontal (exemples)

Domaine



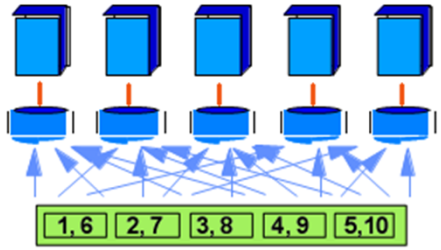
Les données sont reparties en fonction des domaines de valeur des clés

Le partitionnement fondé sur les domaines de valeur de la clé

- Permet des optimisations car il est possible d'aiguiller les requêtes vers un nœud en fonction de la valeur de la clé.
- Ce type de partitionnement présente en revanche le risque de déséquilibre des partitions et de la charge.
- Pour illustrer ce type de partitionnement, le plus simple est d'imaginer que les clés **sont des noms** et que la partition se fait en **fonction de la première lettre du nom** : les noms débutant par les lettres A a E. par exemple, appartiennent à la première partition, ceux débutant par les lettres F à la seconde, etc.

- **Méthodes de partitionnement horizontal (exemples)**

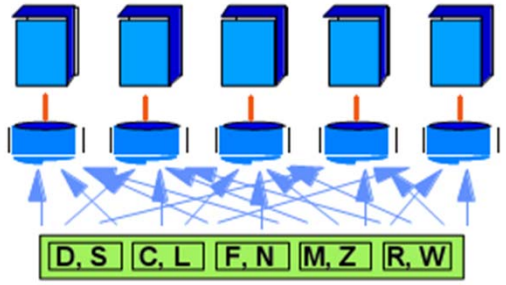
Round robin



Chaque article est rangé dans
la partition suivante en séquence

- ***Le Round Robin possède***, par définition, la propriété d'équilibrer les partitions. En revanche, la répartition de la charge n'est pas assurée (puisqu'elle dépend des articles qui sont accédés).

- **Méthodes de partitionnement horizontal (exemples)**



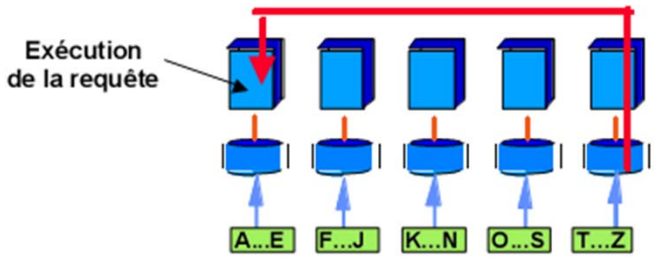
Un algorithme appliqué a la clé de l'article détermine son numéro de partition)

- ***La méthode fondée sur une fonction de hachage***, ou *Hashing,* pose le problème du choix de la fonction de hachage. Un exemple de fonction de hachage qui produit des résultats généralement satisfaisants est le suivant :
  - on considère que la représentation (binaire) de la clé est un nombre entier de longueur variable, et la valeur de la fonction hachage est le reste de la division de la clé par un nombre premier.
  - Une fonction de hachage doit être simple a évaluer et fournir un bon équilibre des partitions.

Deux modèles fonctionnels dans le cas d'une architecture de SGBD réparti

- Dans le cas d'une architecture de type Share Nothing, **deux modèles fonctionnels** sont possibles pour un nœuds ayant besoin d'accéder a des données se **trouvant sur d'autres nœuds.**

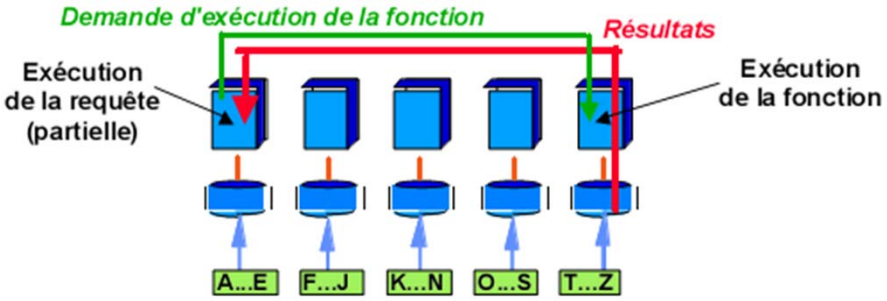
1. Dans le modèle ***Expédition de données***, le nœud désirant ***effectuer une opération*** sur des données gérées par un nœud distant expédie a ce nœud distant **une demande d'accès aux données**. Le nœud distant réalise l'accès et expédie les données, *via* le réseau d'interconnexion, au nœud



Envoi de données (Data Shipping). Le guichet traitant la requête demande au sous-ensemble de guichets possédant la donnée de la lui adresser. Ce guichet se charge de la réalisation de l'opération et du renvoi des données (en cas de modification) au sous-ensemble de guichets qui gère ces données

Expédition de données « Data Shipping »

2. Dans le modèle ***Expédition de fonction***, le nœud désirant ***effectuer une opération*** sur des données gérées par un nœud distant expédie au nœud distant une **demande d'exécution** de la partie de la requête (appelée ici une fonction) qui concerne les données gérées par ce nœud. Le nœud distant exécute la fonction sur les données concernées et expédie le résultat (c'est-à-dire des données), *via* le réseau d'interconnexion, au nœud demandeur. Une requête peut ainsi être exécutée par plusieurs nœuds.



Expédition de fonction « Function Shipping »

Envoi de fonction (Function Shipping). Le guichet traitant la requête adresse au sous-ensemble de guichets possédant les données la définition de la fonction a exécuter. Ce dernier retourne, le cas échéant, les résultats de la fonction qu'il a exécutée.

- Les avantages et inconvénients des deux modèles dépendent des quantités de données à « exporter » et de la quantité de traitement « exportée ».

Optimisation des requêtes réparties-01-

- Établissement d'un plan d'évaluation optimal
- Optimisation d'une fonction de coût ou de temps de réponse de la forme :

coût global = a x coût(E/S) + b x coût(Processeur) + c x coût(Communication)
+ d x coût(Transfert des données)

- Rappel : la compilation d'une requête SQL produit un arbre d'évaluation composé d'un certain nombre d'opérateurs de base :
  - Projection : Π_X R projection de la relation R sur la liste d 'attributs X
  - Sélection : σ_P R sélection des tuples de R vérifiant le prédicat P
  - Équijointure : R1 $\bowtie_A$ R2 jointure des relations R1 et R2 selon l'attribut A (R1.A=R2.A)
  - Produit cartésien : x produit de deux relations
  - Union : $\cup$ union de 2 relations
  - Intersection : $\cap$ intersection de deux relations
- L'optimisation vise à transformer l'arbre d'évaluation en un arbre optimal

Exemple - Définition de la base de données

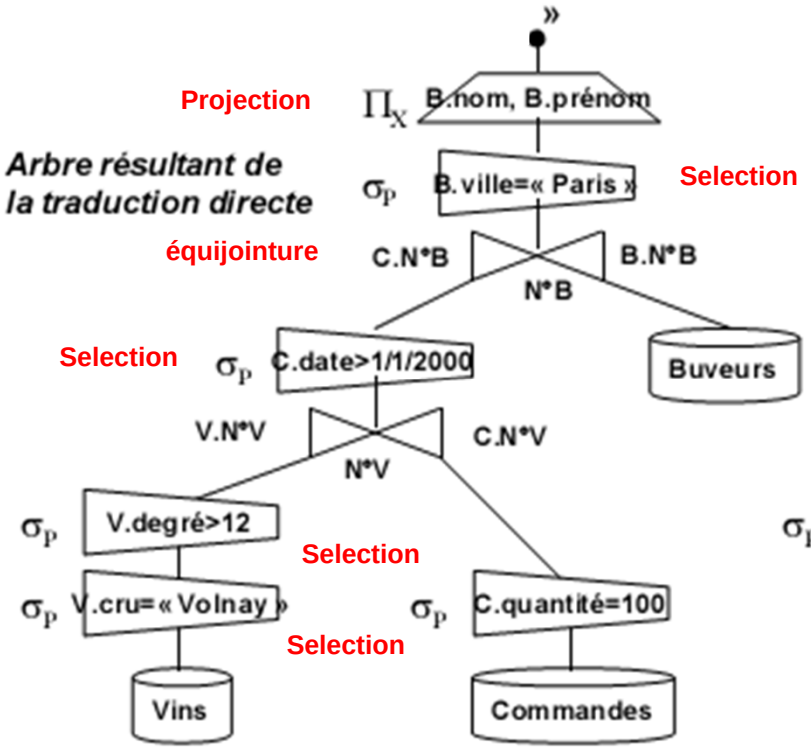
- **B :** Buveurs (**N°B, nom, prénom, ville**)
- **V :** Vins (**N°V, cru, année, degré**)
- **C :** Commandes (**N°V, N°B, date, qté**)
- **Exemple de requête de sa traduction et de son optimisation en l'absence de répartition :**

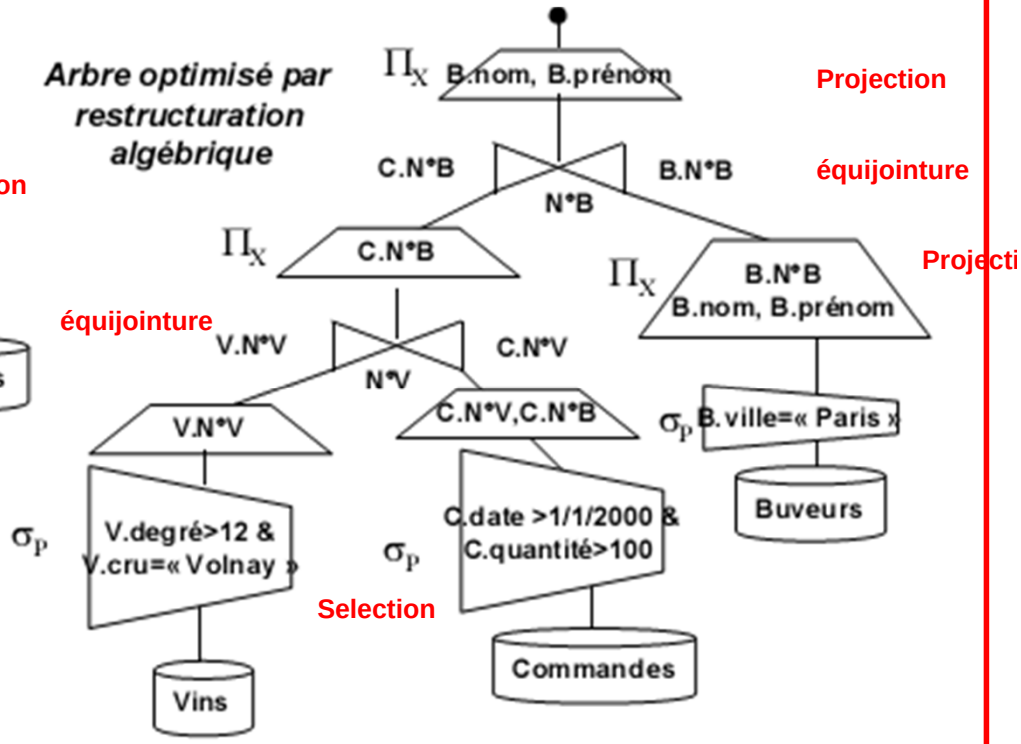
```
SELECT nom, pénom FROM buveurs (B), vins (V), commandes (C)
WHERE V.cru =« Volnay »
AND     V.degré > 12
AND     C.quantité > 100
AND     C.N°V = V.N°V
AND     C.date >1/1/2000
AND     B.N°B = C.N°B
AND     B.ville =« Paris »
```

Optimisation des requêtes réparties-03-

SELECT nom, prénom **FROM** buveurs (B), vins (V), commandes (C)
WHERE V.cru =« Volnay » **AND** V.degré > 12 **AND** C.quantité > 100
AND C.N°V = V.N°V **AND** C.date >1/1/2000 **AND** B.N°B = C.N°B **AND**
AND B.ville =« Paris »

B :	Buveurs	(N°B, nom, prénom, ville)
V :	Vins	(N°V, cru, année, degré)
C :	Commandes	(N°V, N°B, date, qté)





Optimisation des requêtes réparties-04-

- Hypothèse de volume :
  - Buveurs (B) : 10 000 tuples
  - Vins (V) : 1 000 tuples
  - Commandes : 200 000 tuples
- Hypothèse de distribution des BD :
  - Paris : Buveurs (B)
  - Dijon : Vins 1 (V1) restriction N°V<= 400
    Commandes 1 (C1) restriction N°V<= 400
  - Bordeaux : Vins 2 (V2) restriction N°V > 400
    Commandes 2 (C2) restriction N°V>400
- Requête émise sur le site de Paris :
  « Noms des buveurs parisiens n'ayant pas commandé en décembre 2000 »
  Sélectivités supposées : 20% de parisiens et 1% n'ayant pas commandé
- Stratégies :
  - Simpliste : transférer C1 et C2 vers Paris (200 000 tuples) et faire C = C1 ∪ C2 et évaluer SELECT B.nom FROM Buveurs (B) WHERE B.ville = « Paris » AND B.N°B NOT IN (SELECT N°B FROM C WHERE C.date>1/12/2000 AND C.date<1/1/2001)
  - Améliorée : Transférer vers Dijon et Bordeaux Buveurs.N°B des seuls parisiens (= 2 x 2 000 petits tuples). Évaluer sur les sites de Dijon et Bordeaux : Buveurs.N°B NOT IN (SELECT N°B FROM Ci WHERE Ci.date>1/12/2000 AND Ci.date<1/1/2001)
    Transférer les résultats vers Paris (= 2 x 20 petits tuples) et faire l'intersection des résultats