Les SGBD Parallèles

Didier DONSEZ

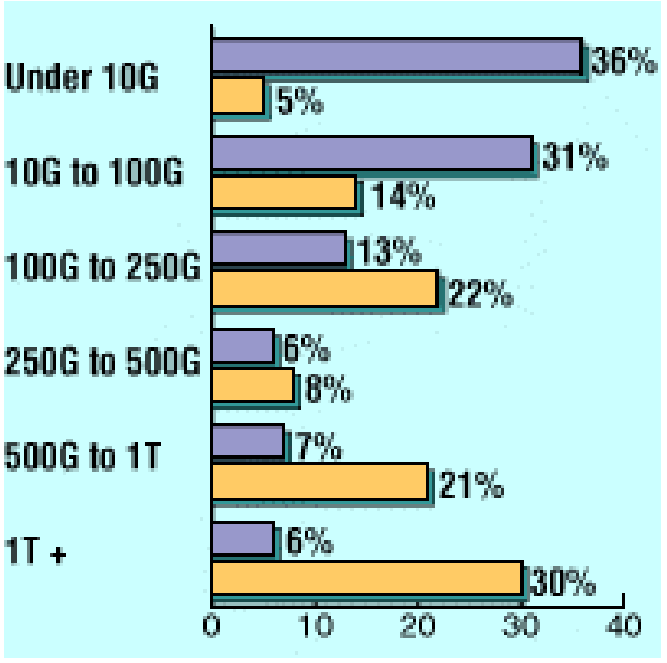
Université Joseph Fourier (Grenoble 1)

IMA – LSR/ADELE

Didier.Donsez@imag.fr, Didier.Donsez@ieee.org

Le problème des systèmes d'Information

- Les entreprises dépendent de l'information à jour disponible à temps.
  - augmentation du volume d'information :
    - 14 fois de 1990 à 2000
    - 1000 premières entreprises mondiales gèrent chacune un total de 400 To d 'infos en moyenne en 2000 contre 28 To en 1990
    - Taille des Data Warehouses d 'entreprise
      1996 en bleu, 1999 en orange
      (source : Meta Group)



  - augmentation du volume des transactions :
    - 10 fois dans les 5 prochaines années

Le problème des systèmes d'Information

- La charge de travail change :
  - simple OLTP (transactionnel)
  - transactions complexes
    (e.g., comme support au décisionnel)
  - transactions très complexes
    (e.g., générées par des systèmes experts)
- Le Besoin :
  - des serveurs bases de données qui fournissent
    à haut débit et avec de bons temps de réponse
    des charges de travail variées sur des très grosses bases de données.

Le problème des Bases de données

- Gros Volumes de données
  - => utilisation de disques et de grandes mémoires
- Prédiction d'accroissement :
  - vitesse des microprocesseurs : 50% par an
  - capacité des DRAM : 4 fois toutes les 4 ans
  - débit des disques : 2 fois sur les 10 dernières années
- Conclusion:
  - Le goulot d'étranglement sont les Entrées/Sorties (E/S)

La Solution

- Améliorer le débit d'E/S
  - Partitionnement des données
  - accès parallèle aux données
- Origine (années 80) : les Machines BDs
  - orientées vers le matériel spécialisé
    - => mauvais rapport coût / performance
  - exception notable : CAFS ISP d'ICL
- Années 90 : la même solution mais avec des composants standards (Off the shelf) intégrés dans un multiprocesseur
  - solution logiciel
  - => profite des améliorations constantes de la technologie

Les Objectifs des Machines Parallèles

- Hautes performances pour un meilleur rapport coût /performance que les mainframes

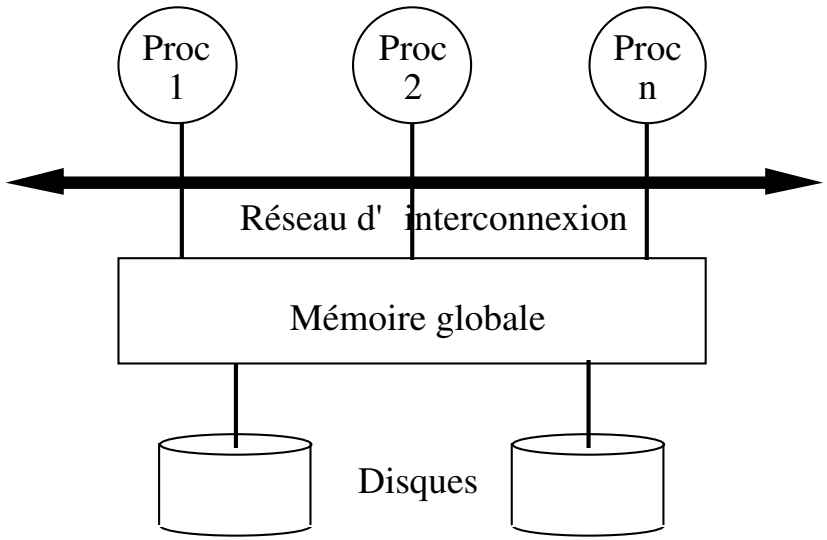
- Utiliser de nombreux nœuds *(avec un bon coût/performance)* et les faire communiquer par un réseau
  - des composants très répandus
  - un bon débit de communication
- Les tendances:
  - microprocesseur et DRAM du marché
  - réseau de communication : custom
- Le vrai challenge :
  - paralléliser les applications
    et les exécuter en équilibrant la charge des nœuds

Les Architectures de Systèmes Parallèles

- Les extrêmes
  - Architecture à Mémoire Partagée (Shared Memory)
  - Architecture à Mémoire Distribuée (Shared Nothing)
- L'intermédiaire
  - Architecture à Disque Partagé (Shared Disk)

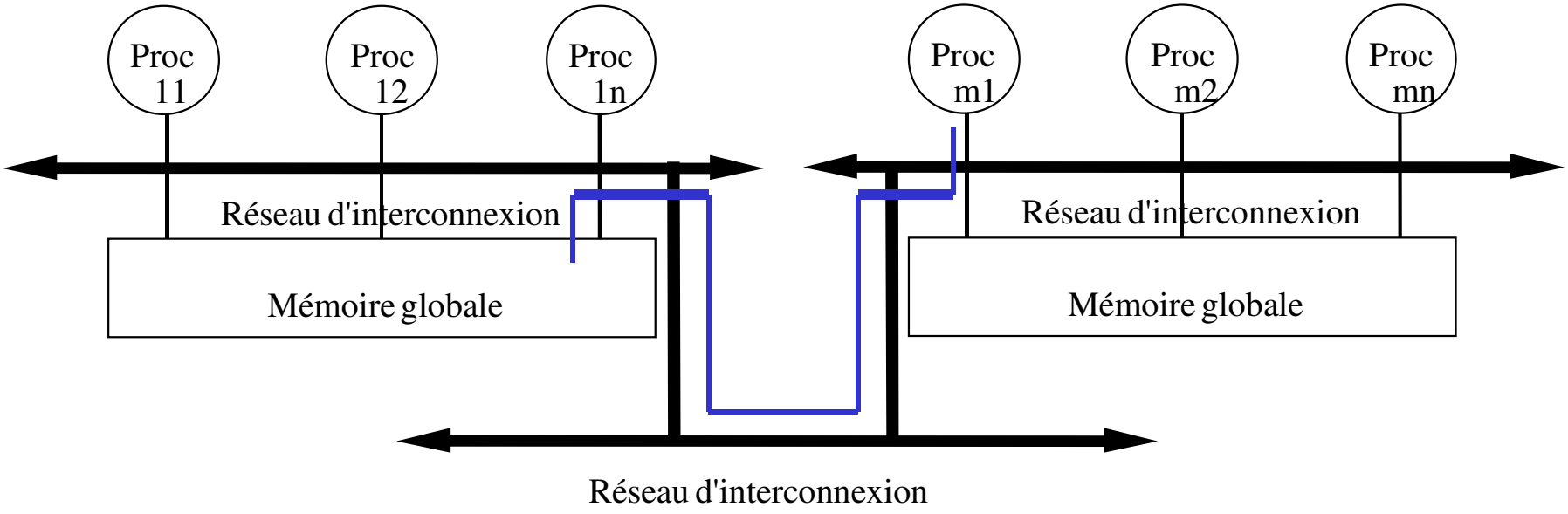
Architecture à Mémoire Partagée (Shared Memory) ou Machine à couplage fort

- Exemples
  - multiprocesseurs symétriques (Sequent, Encore, Bull Escala) sous Unix ou Windows NT
  - SGBD // : XPRS (U. de Berkeley), DBS3 (Bull)
- Modèle de programmation:
  - mémoire partagée (multiprocessus, multithreading)
- Avantages :
  - simplicité, équilibrage de la charge
- Inconvénients :
  - Coût du Réseau (bus parallèle ou multibus) , Faible extensivité, faible disponibilité



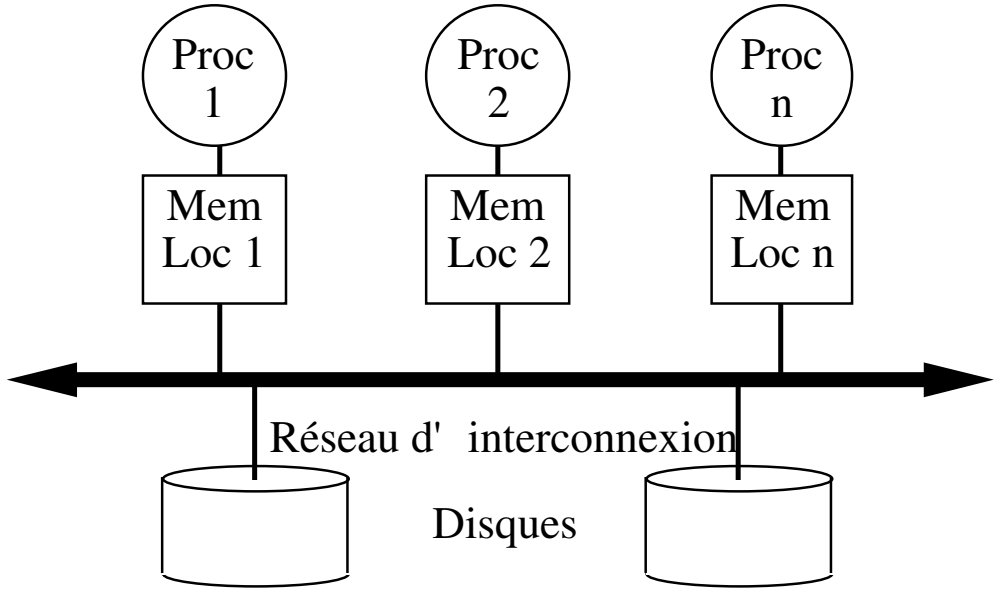
Architecture NUMA (Non Uniform Memory Access)

- Machines:
  - Architecture VIA (GigaNet), Sequent NUMA-Q 2000 (IQ/Link à 1Go/s entre clusters)
    Sun Ultra Enterprise 1000, IBM S/70 RS6000, Compaq/Digital AlphaServer SMP
  - Informix, Oracle, IBM DB2
- Modèle de programmation:
  - mémoire partagée (multiprocessus, multithreading)
- Avantages :
  - Simplicité, Extensibilité Moyenne
- Inconvénients :
  - équilibrage de la charge



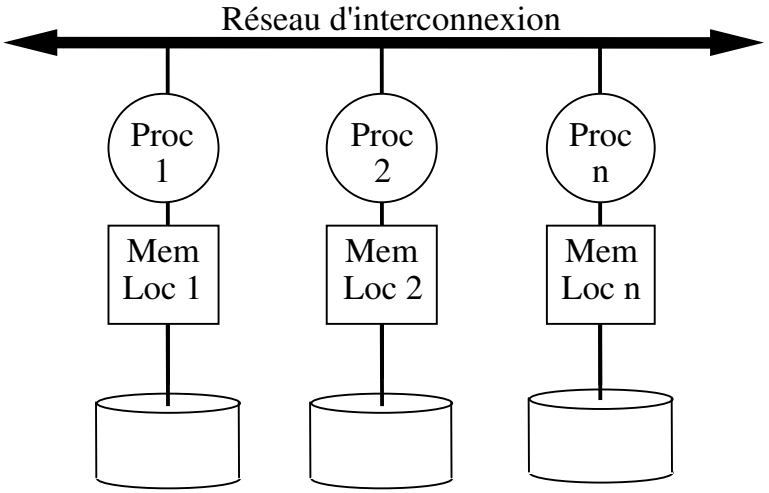
Architecture à Disque Partagé
Shared Disk ou Cluster

- Exemples :
  - TruCluster (DEC), IMS/VS Data Sharing, Parallel SysPlex S390, HACMP AIX (IBM), MS Cluster Server (ex NT Wolfpack), MC/ServiceGuard (HP), Unix LifeKeeper (NCR), Solstice HA Parallel DB (Sun)
- Avantages :
  - Coût du Réseau, extensibilité, migration depuis les monoprocesseurs, sûreté de fonctionnement (les CPU peuvent être séparées de plusieurs centaines de mètres par un RI de type FibreChannel)
- Inconvénients :
  - complexité

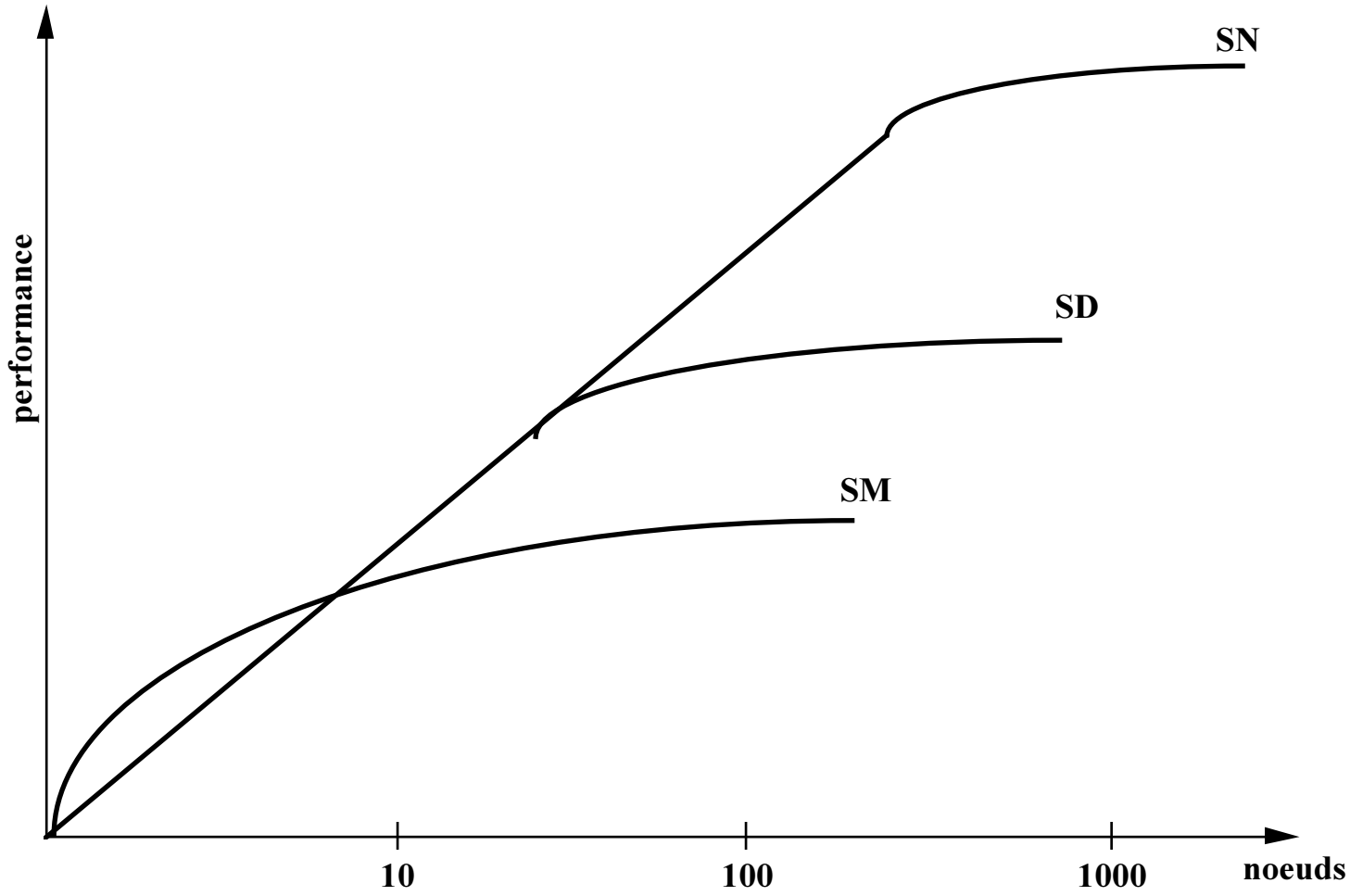


Architecture à Mémoire Distribuée (Shared Nothing) ou machine à couplage lâche (faible)

- Exemples :
  - réseaux d'interconnexion : Grille, Tore, Anneau, HyperCube, ...
  - nCube, Thinking Machine, Intel IPSC, SuperNode, ...
  - Teradata (ATT GIS), NonStop SQL (Tandem), Gamma (U. de Wisconsin), Bubba (MCC)
- Avantages :
  - Coût, extensibilité, disponibilité
- Inconvénients :
  - complexité, équilibrage difficile de la charge

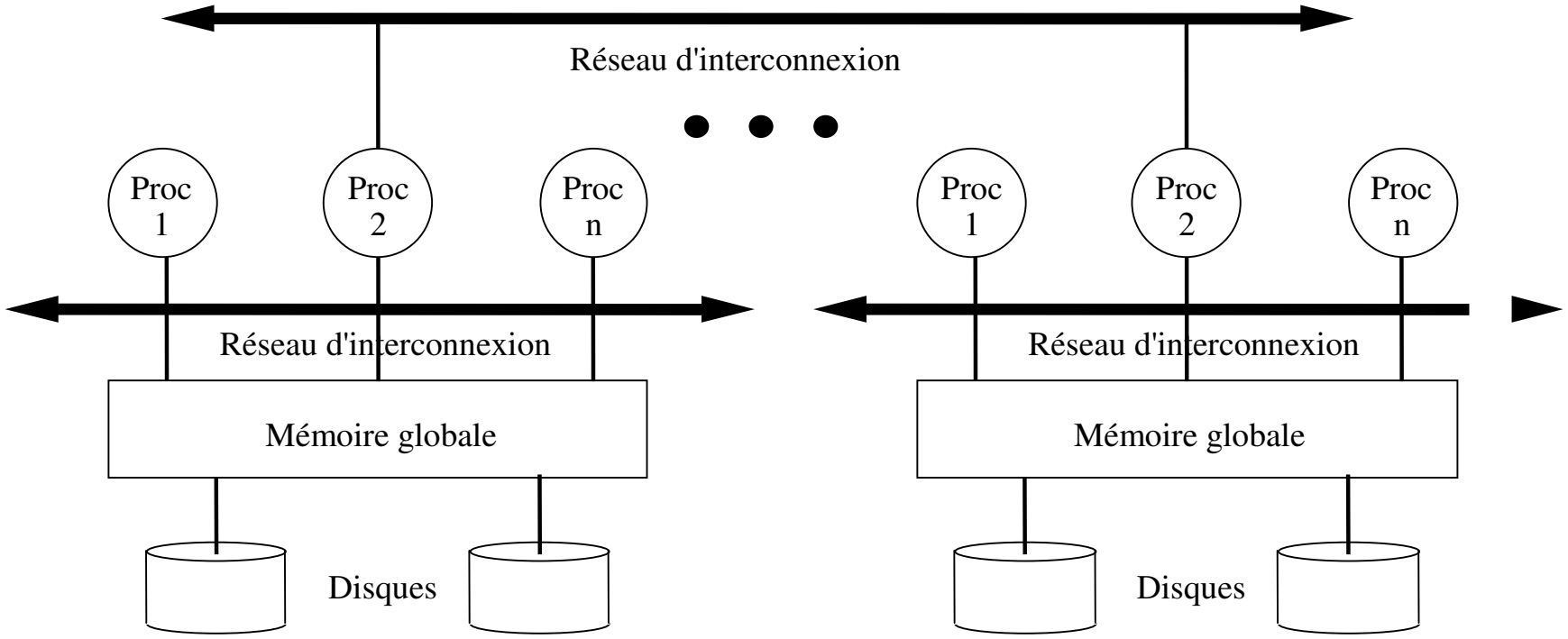


L 'extensibilité (*Scalability*) des Performances



Architecture Hybride : Grappes de nœuds Mémoire Partagée (cluster)

- Combiner
  - l'équilibrage de la charge (SM)
  - et l'extensibilité (SN)



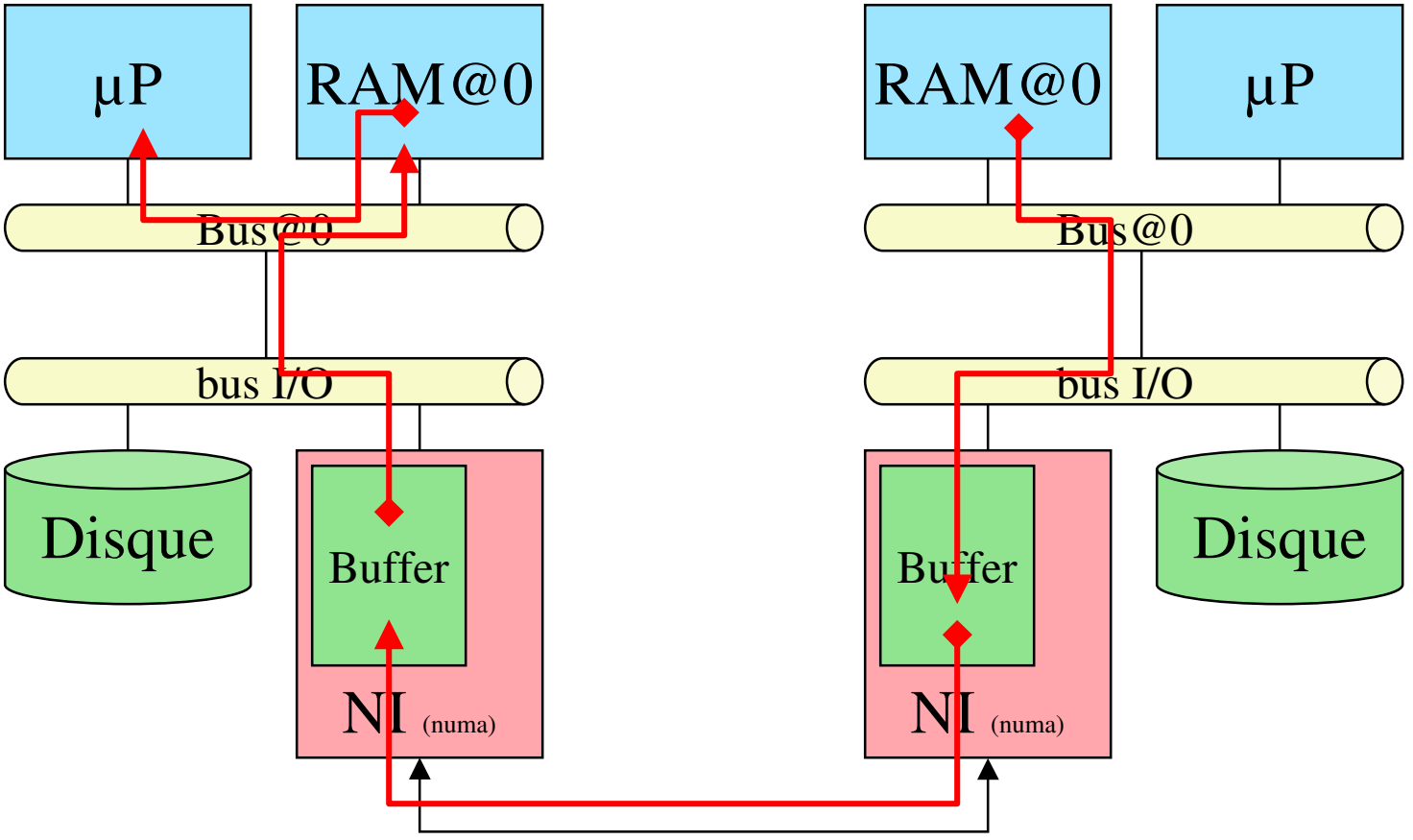
Structures d 'Interconnexion
System Area Networks

- Caractéristiques
  - Interconnexion de machines à très haut débit et très faible latence (très faible taux d'erreur)
- Modèles de partage
  - Cluster partage de disques
  - Echange mémoire-mémoire
    - NUMA : Non Uniform Memory Access
- Marché émergent
  - Liaisons Privés
    - Fibre Channel, HIPPI, Synfinity, Myrinet, Sequent Q-Link, …
    - VIA (Virtual Interface Architecture), Giganet, ...
    - Switch Ethernet, FDDI, ATN
  - **Liaisons Opérateur (VPN)**
    - ATM
  - Exploitation par les OS et les SGBDs
    - WinNT, Unix, VMS,OS390 … Linux, SCO UnixWare
    - Oracle, DB2, Informix, Sybase, ...

Principe de la NI (*Network Interface*) dans les SAN

La NI est considérée comme un organe d 'I/O

Temps de latence important, Nombreux cycle CPU, ...

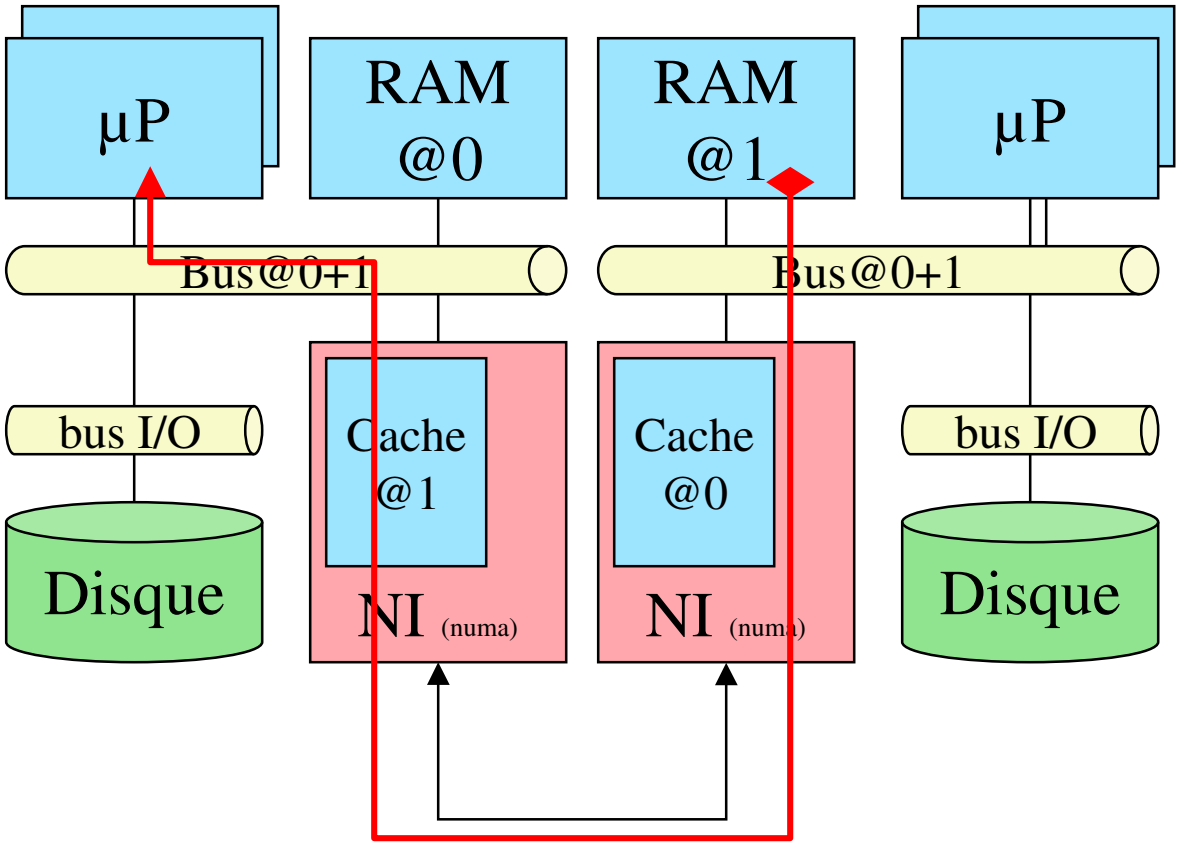


Principe de la NI (*Network Interface*) dans une Mémoire NUMA

La NI n 'est plus considérée comme un organe d 'I/O

C 'est une mémoire adressable directement par le(s) processeur(s)

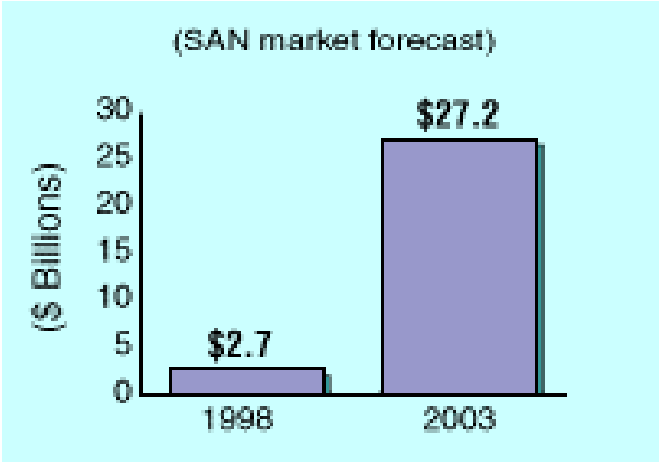
cc:NUMA : Cache Coherence for NUMA



Systèmes de Stockage de Masse
Storage Area Networks (un autre SAN) vs Network-Attached Storage (NAS)

- Principe
  - Clusters de Stockage et d 'Archivage
    - Baies de Disques (RAID)
    - Jukebox de Bandes
  - partagé via un réseau haut débit
- SAN
  - Réseau dédié au stockage
    - Fibre Channel

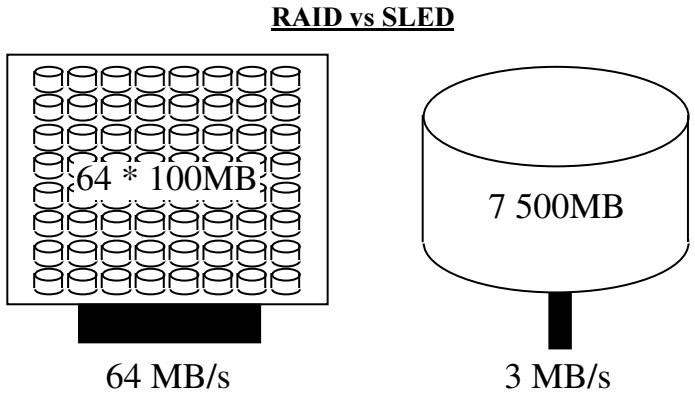
      Bus Série à 100 Mo/s sur fibre optique (10 Kms au maximun)
- NAS
  - Réseau partagé à les autres « applications »
    - Ethernet, ATM, ...
- Acteurs
  - HP, IBM, Storagetek, Legato, ...
  - www.fibrechannel.com, www.snia.org



Source : Meta Group

Clusters de Stockage

- RAID - Redundant Array of Inexpensive Disks [Patterson 88]



Niveau 1 • : Les Disques Mirroirs (TANDEM Mirrored Disks)
Niveau 2 : Code de Hamming pour ECC (CM2 DataVault)
Niveau 3 : Un Seul Disque de Controle par Groupe de Disque
Niveau 4 : Lectures et Ecritures indépendantes
Niveau 5 • : Contrôle réparti sur les disques du Groupe
Niveau 0 • : Pas de redondance

- Contrôleur RAID actuel
  - Disques Hot Plug 80 Go par groupe de 5 pour du RAID niveau 0,1,5
  - interface : SCSI, FiberChannel

Nouvelles architectures d 'E/S (NGIO : Next Generation I/O)

- Forum NGIO : Next Generation I/O
  - http://www.ngioforum.org
  - supporté par HP, IBM, SUN, DELL, … MicroSoft, ...
- Réseau switché d 'échange Processeur/Mémoire/Cluster de Disque
  - 2,5 Gbit/s

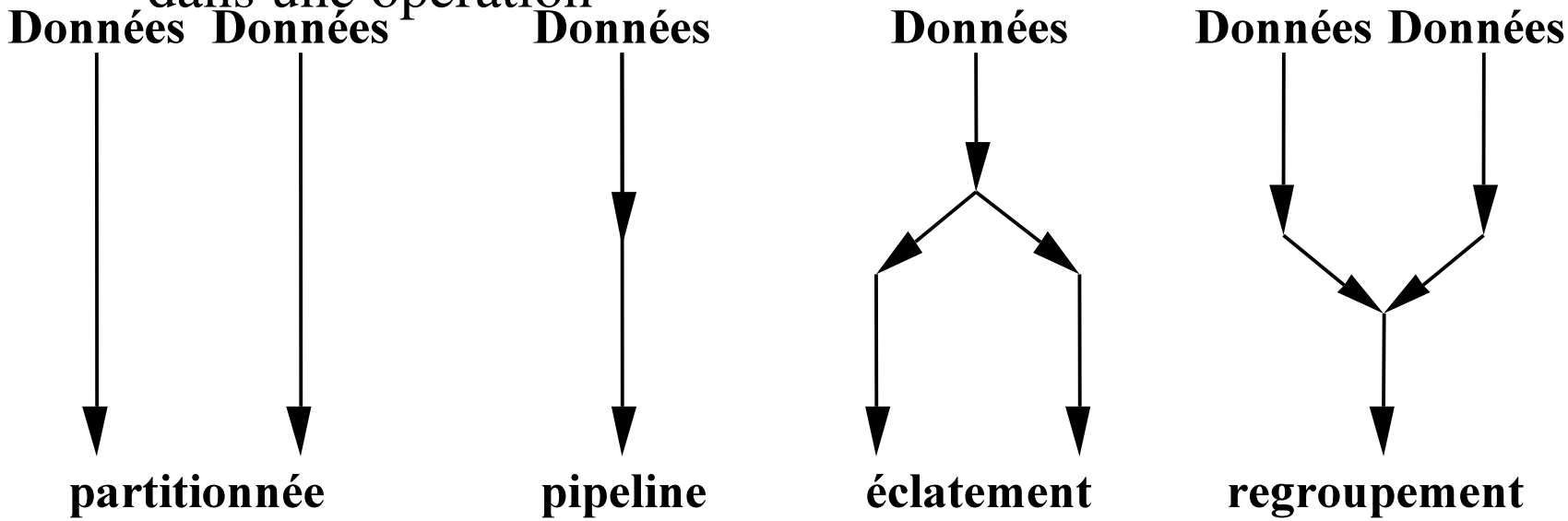
Le Calcul Parallèle

- Les 3 voies d'exploitation des multiprocesseurs
  1. détection automatique du parallélisme dans des programmes séquentiels (e.g, FORTRAN)
  2. ajouter des structures de contrôle parallèles dans un langage existant (e.g., FORTRAN 90, C*)
  3. offrir un nouveau langage dans lequel le parallélisme est automatiquement inféré (e.g., SQL)

- Critiques
  1. dur de développer des (compilateurs) paralléliseurs
  2. permet au développeur d'exprimer le parallélisme mais trop bas niveau
  3. permet de combiner (1) et (2)

Le Parallélisme des Données

- Inférer le parallélisme dans les requêtes SQL
  - entre les requêtes
  - dans une requête
  - dans une opération



Les SGBDs Parallèles

- Définition approximative
  - un SGBD implanté sur un multiprocesseur fortement couplé
- Autres Alternatives
  - portage directe d'un SGBD Réparti
    - (l'approche des vendeurs de logiciels)
  - combiner du logiciel avec le nouveau matériel
    - (l'approche des fabriquants de machines)
- Extension naturelle aux SGBDs Répartis
  - 1 serveur par site

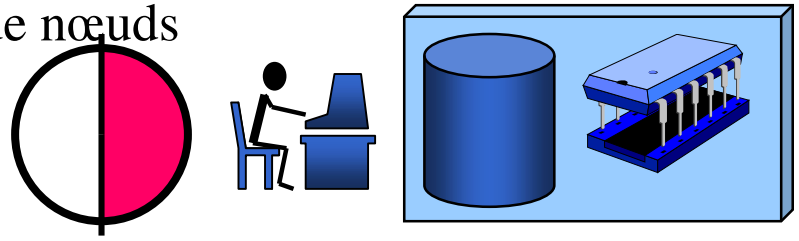
Les SGBDs Parallèles : Objectifs

- Meilleur ratio Coût/Performance qu'une solution mainframe
- Haute performance par le parallélisme
  - haut débit de transactions : parallélisme inter-requête
  - temps de réponse plus courts : parallélisme intra-opération
- Haute disponibilité par la réplication des données
- Extensibilité (Scalability) avec des buts idéaux
  - Speed Up : amélioration linéaire du temps de réponse
  - Scale Up : accroissement linéaire de la charge

Speed-up et Scale-up

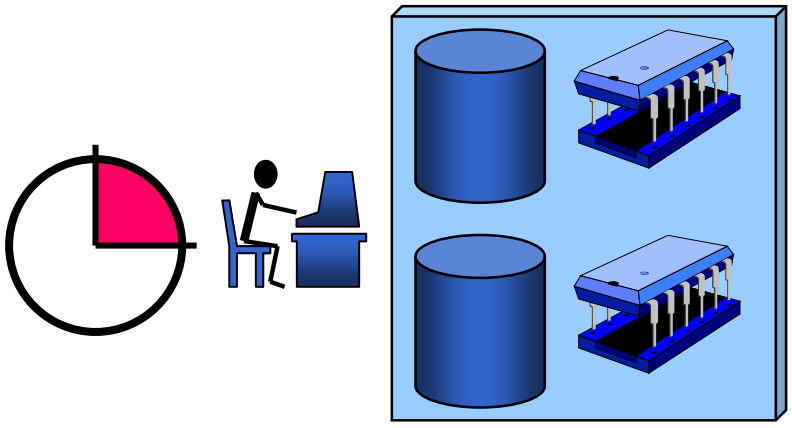
Speed-up

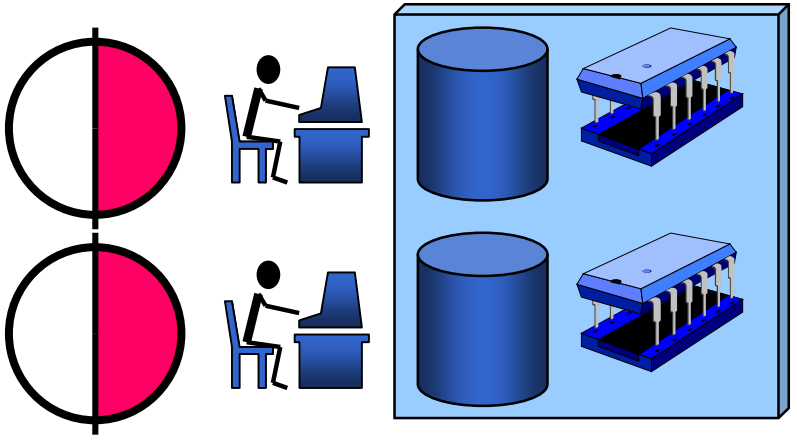
- Diminution du temps de réponse en augmentant le nombre de nœuds



Scale-up

- Accroissement linéaire de la Charge de Travail





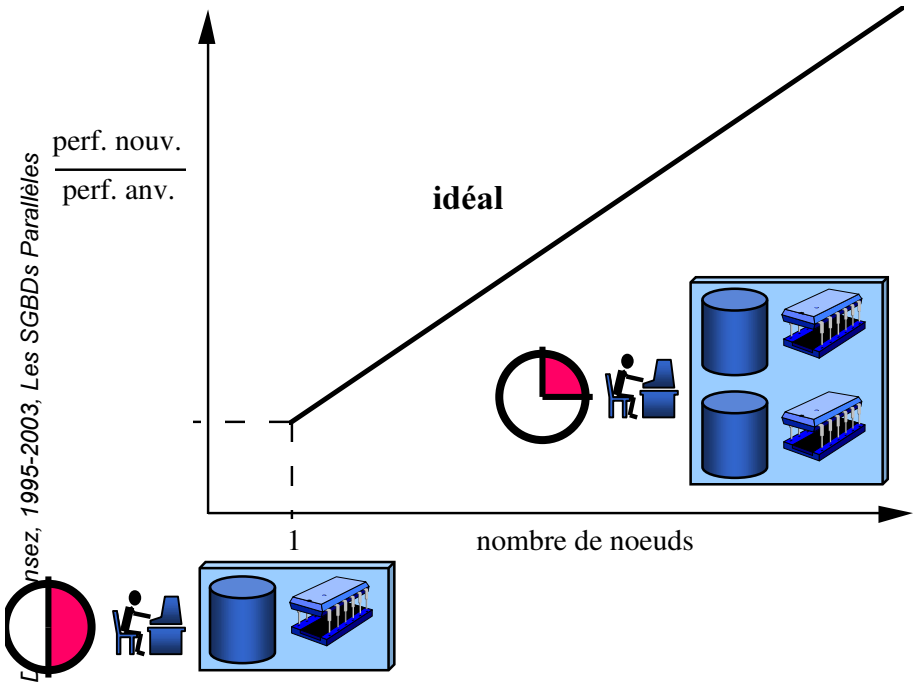
Le Gain Idéal : l ’accélération Linéaire

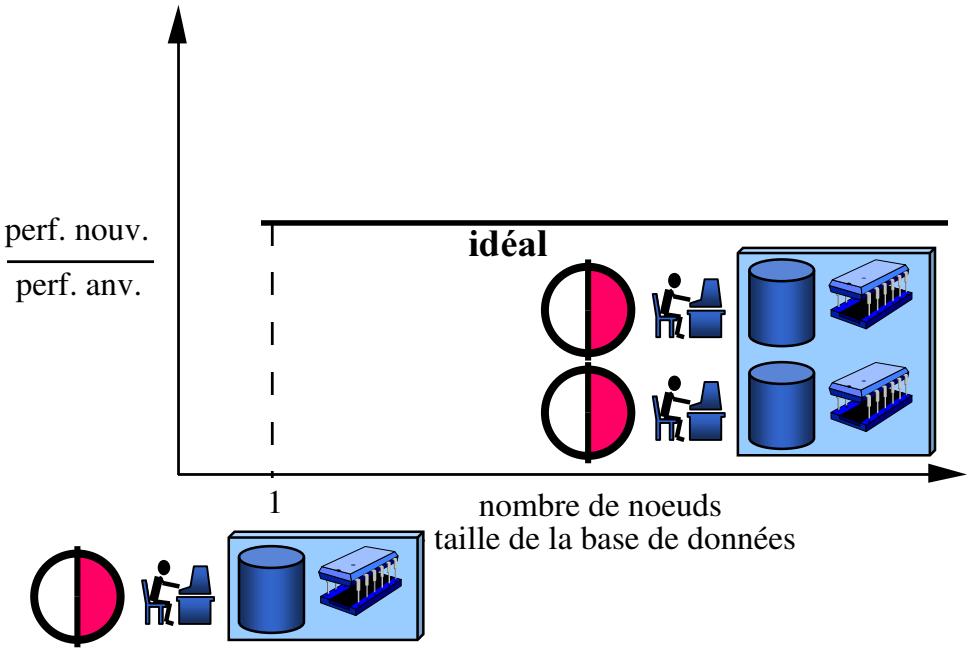
Speed-up

diminution du temps de réponse en augmentant le nombre de nœuds (avec BD constante)

Scale-up

conserver le même temps de réponse en augmentant proportionnel la taille de la base avec le nombre de nœuds





Les Barrières au Parallélisme

- démarrage (start-up):
  - temps nécessaire pour démarrer une opération parallèle
    - *peut dominer le temps de calcul*
- interférence
  - un processeur ralentit les autres quand il accède à des ressources partagées
    - *hot-spot*
- biais (skew)

  temps de réponse d'un ensemble de processus parallèles
  = *temps de réponse du plus lent*

- Les SGBDs parallèles visent à limiter ces barrières

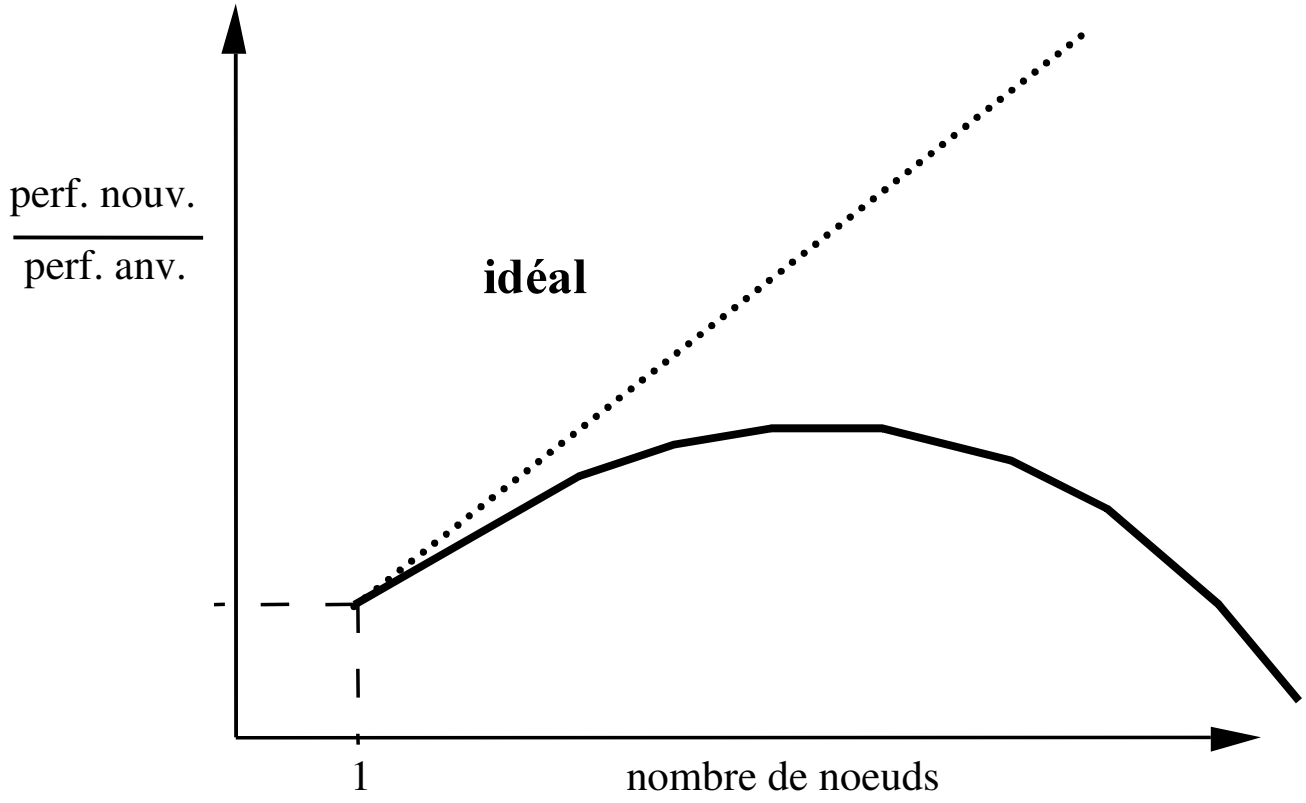
Le Biais (Data Skew)

nuït à l ’accélération linéaire

- Déséquilibre intrinsèque aux données
  - Attribute Value Skew: AVS
    - distribution non uniforme des valeurs d ’attributs
- Déséquilibre lié au partitionnement des données
  - Tuple Placement Skew (TPS)
    - variation de la distribution des tuples entre les partitions.
  - Selectivity Skew (SS)
    - variation de la sélectivité des prédicats de sélection entre les processeurs.
  - Redistribution Skew (RS)
    - variation du nombre de tuples arrivant après la phase de redistribution.
  - Join Product Skew (JPS)
    - quand les jointures locales sont de tailles inégales.

Les Barrières au Parallèlisme

- Performance Réelle / Nombre de processus



La parallèlisation dans les BDs

- Programme à parallèliser
  - une requête relationnelle
- Parallélisme intra-opération
  - Basé sur un parallélisme de données
    - même traitement à appliquer à chaque tuple (ligne) d 'une ou plusieurs relations
  - Fragmentation de chaque table source entre plusieurs nœuds
    - (disques ou processeurs)
  - Décomposition des opérations en sous-opération
- Parallélisme inter-opération
  - enchaînement des opérations en Pipeline (Unaire)
  - enchaînement des opérations en Dataflow (Binaire)

Techniques utilisées

- Placement des données
  - Fragmentation des tables
    - prise en compte du biais
  - Allocation physique des fragments sur plusieurs noeuds
  - Placement statique / dynamique, Réorganisation
- Algorithmes Parallèles pour les opérateurs relationnels
  - unaire : facile
    - sélection, projection, partition, agrégat
  - binaire : plus difficile
    - jointure, star-join, ...
- Optimisation des Requêtes Parallèles
  - parallélisation des requêtes
  - choisir le meilleur plan d'exécution
  - déclenchement des opérations
- Gestion des transactions
  - similaire à la gestion des transactions distribuées

Placement des données

- *Soit un Nœud = Disques en SD et SE, Processeurs en SN, Clusters en Hybride*
- *Soit N le nombre de Nœuds du système*

Fragmentation (ou Partitionnement) des tables

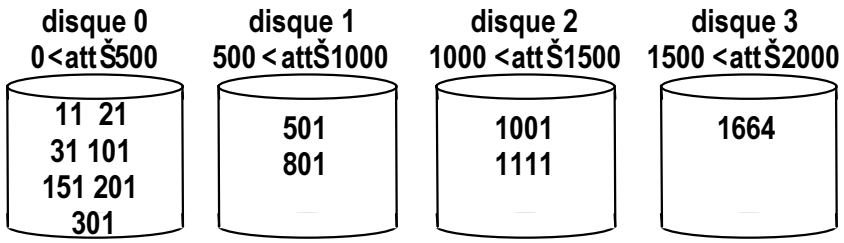
- Degré de Partitionnement (Degree of Declustering DD)
  - nombre de nœuds sur laquelle la table va être distribuée
- Full Declustering
  - pour chaque table Ti, DDi == N
- Partial Declustering
  - pour chaque table Ti, DDi < N (cas des machines massivement parallèles)

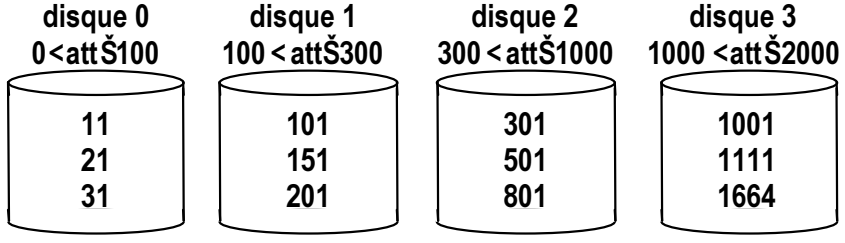
Placement des tables

- Allocation physique des fragments de Ti sur un groupe de DDi nœuds parmi N pour toutes les tables T1 à TN
  - en fonction de la charge de travail type
- Placement statique / dynamique, Réorganisation Périodique
  - prise en compte du biais réel
  - coût de la réorganisation << gain performance

Partitionnement des Données

- Chaque table est divisée en DD sous-tables
  - DD = f (fréquence d'accès , taille relation)
- Plusieurs techniques
  - Round Robin (*Donneur de Carte*)
    - simple, équilibré mais calcul sur l' ensemble des nœuds
  - Ranged (*Intervalle de Valeur*)
    - équilibrage statistique, requête par intervalle
  - Hachage
    - calcul exact
- Intervalles Uniformes / Spécifiés par le DBA



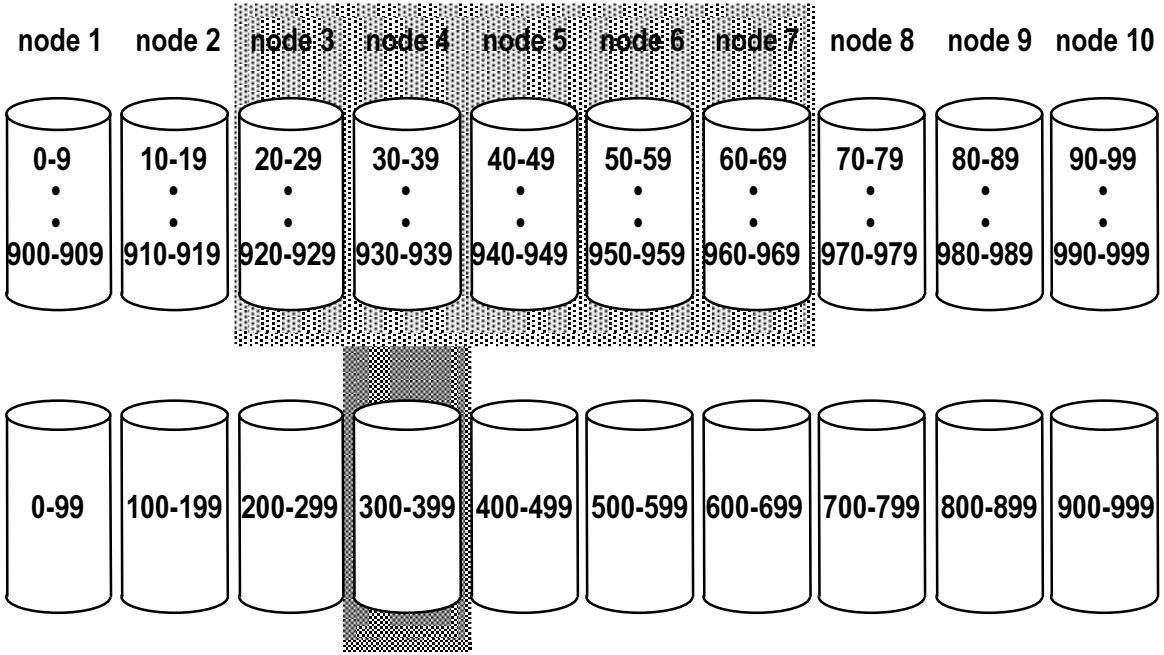


Hybrid-Range Partitioning [Gamma]

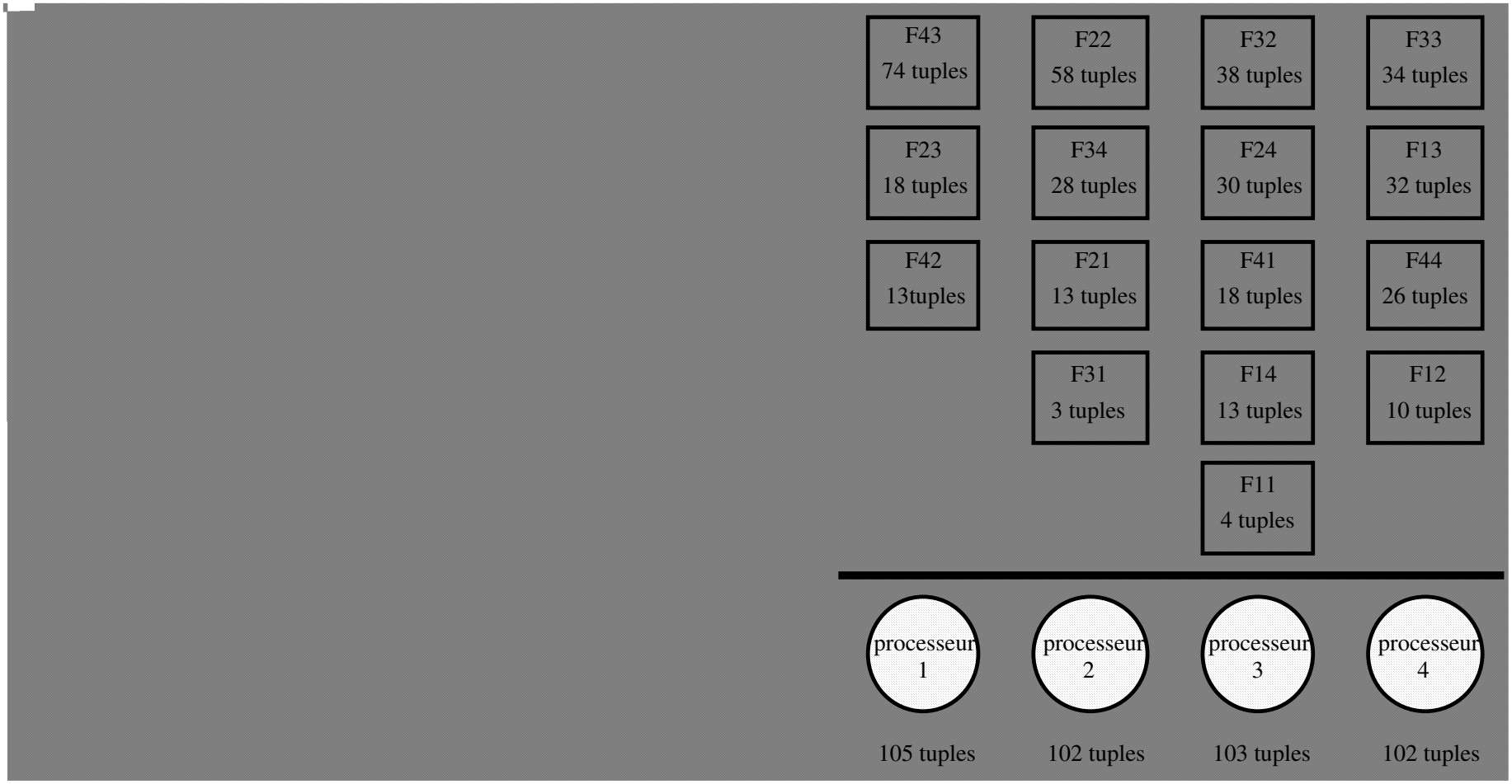
Extensions du Placement Multiprocesseur

- But : augmenter le parallélisme dans les opérations RANGE-SELECT
  - Technique : Déterminer le bon nombre P pour la majorité des questions
    - fragmenter en M = f(P,…) paquets
    - répartition non contiguë des M paquets
  - Performance = Max (perf_Hash, perf_Range)
- Ex: 10 processeurs / P=5

*select * from REL where **ATT BETWEEN 300 AND 399***



Placement multi-attributs [Hua & Lee 1990]

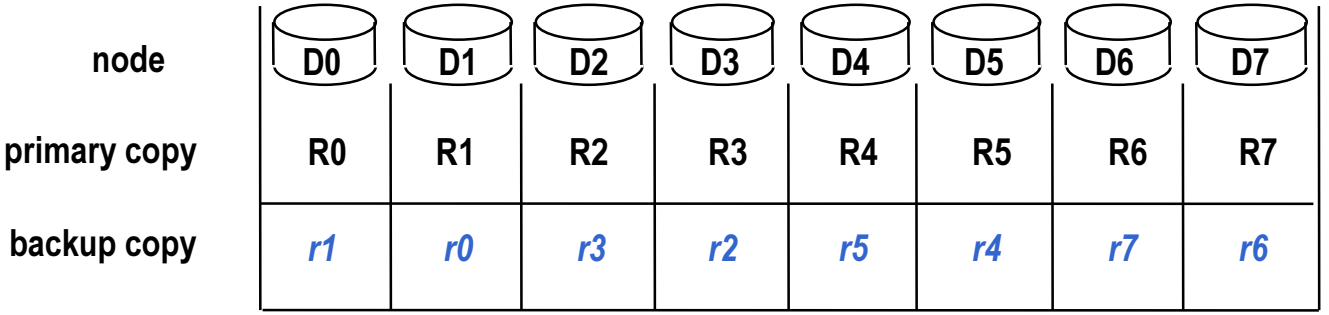


Tolérance aux Pannes (Reliability)

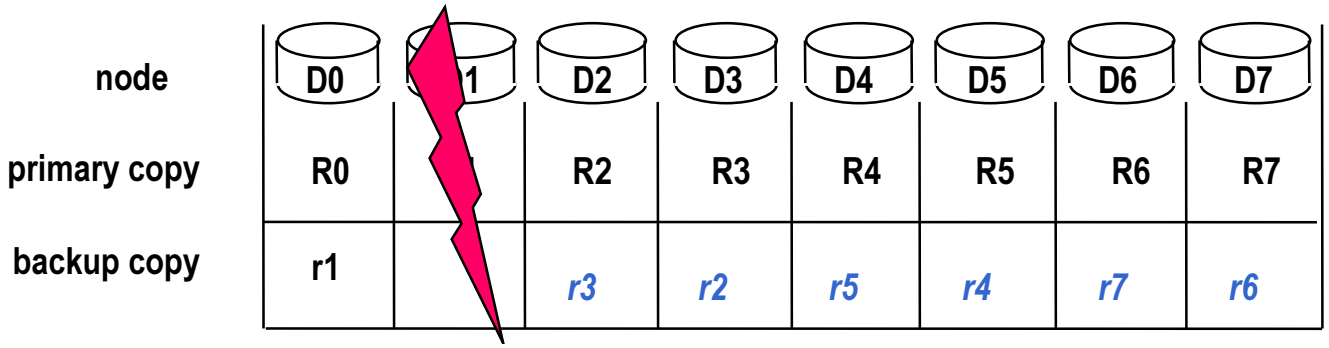
- Réplication dans le Partitionnement et le Placement
  - plusieurs partitionnements
    - même critère vs critères différents
    - même DD ou DD différent
  - plusieurs placements
    - disjoints (tuple à tuple)
- Avantages / Inconvénients
  - ☺ Tolérance à la panne d 'un nœud ou plus
  - ☺ Performance en consultation
  - ☹ Capacité de stockage
  - ☹ Mise à jour synchrone
- Techniques
  - Miroir (Tandem)
  - Chained Declustering (Gamma)
  - Direct-Copy - IF-Copy (Bubba)

Miroir (Tandem)

- Redondance des Données sur N sites
  - Tolérance à la Panne d 'N/2 site
- Fonctionnement à demi-perf lors de la panne
- Fonctionnement
  - Hors Panne: lecture sur copie primaire



  - Panne d'1 nœud D1 : lecture répartie sur les 2 copies



Direct-Copy - IF-Copy [Bubba]

- 2 copies d'une table sur des processeurs différents
  - Direct-Copy
  - IF-Copy:
    - Fichiers inversés (Inverted Files) + fichier reste (RC)

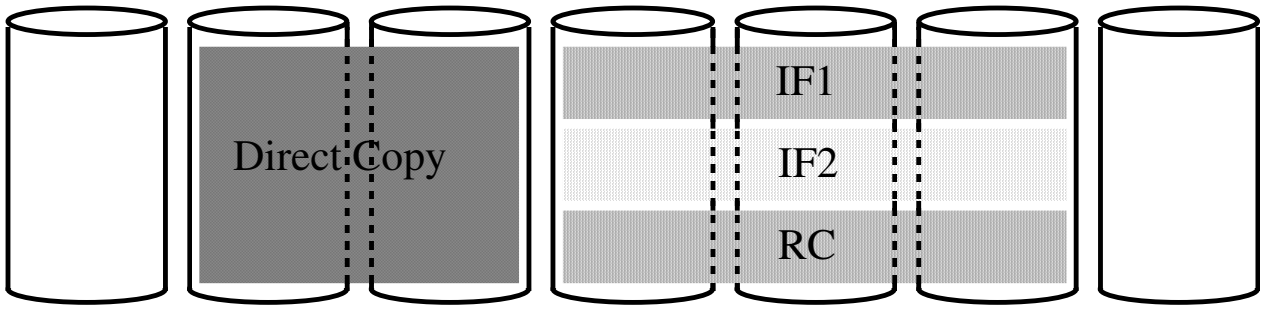
DC	Nom	Prnom	Ville	Age
#1	DUPONT	Pierre	Paris	45
#2	DUPONT	Paul	Nice	36
#3	DURANT	Pierre	Cannes	21

IF 1	Nom	#tuple
	DUPONT	#1
	DUPONT	#2
	DURANT	#3

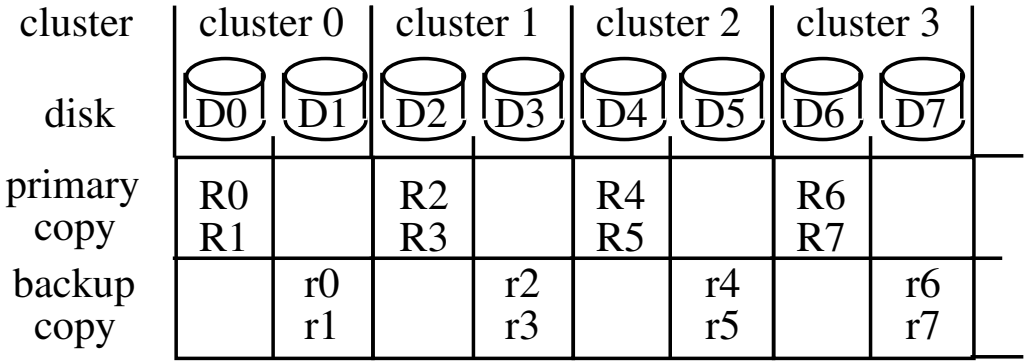
IF 2	Prnom	#tuple
	Paul	#2
	Pierre	#1
	Pierre	#3

RC	#tuple	Ville	Age
	#1	Paris	45
	#2	Nice	36
	#3	Cannes	21

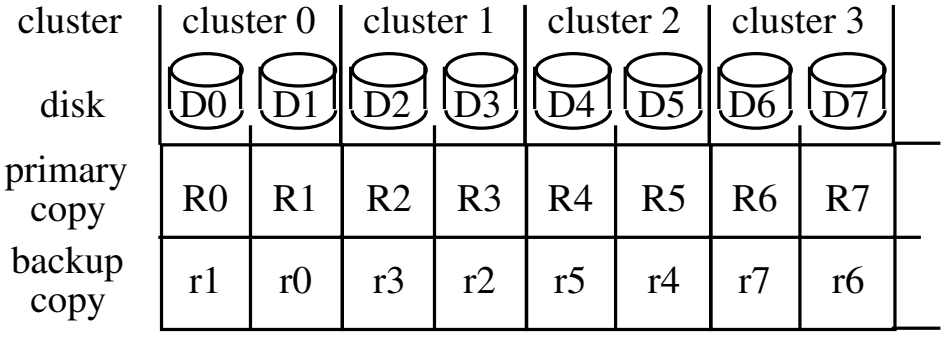
- Répartition des tables sur plusieurs noeuds
  - Direct-Copy : hachage sur attribut clé
  - IF-Copy : IFx sur attribut inversé "x", RC sur l'identifiant
  - DC et IFs+RC sur processeurs ≠ ⇒ résistance aux pannes



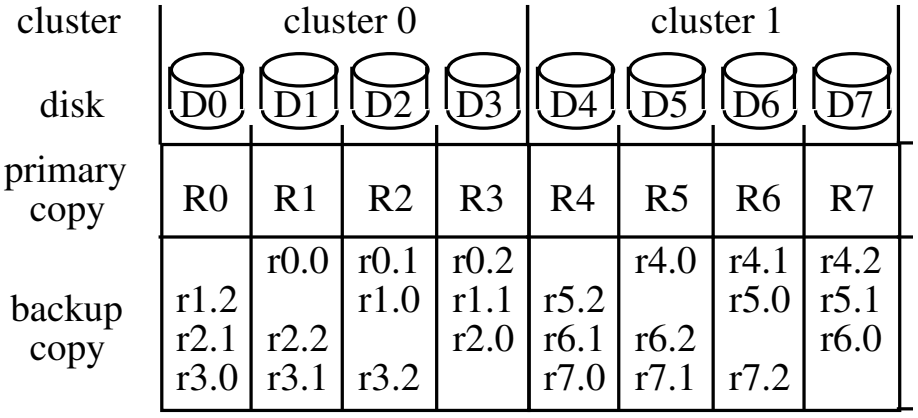
Mirrored Declustering vs Interleaved Declustering



Mirrored Declustering



Interleaved Declustering (S=2)



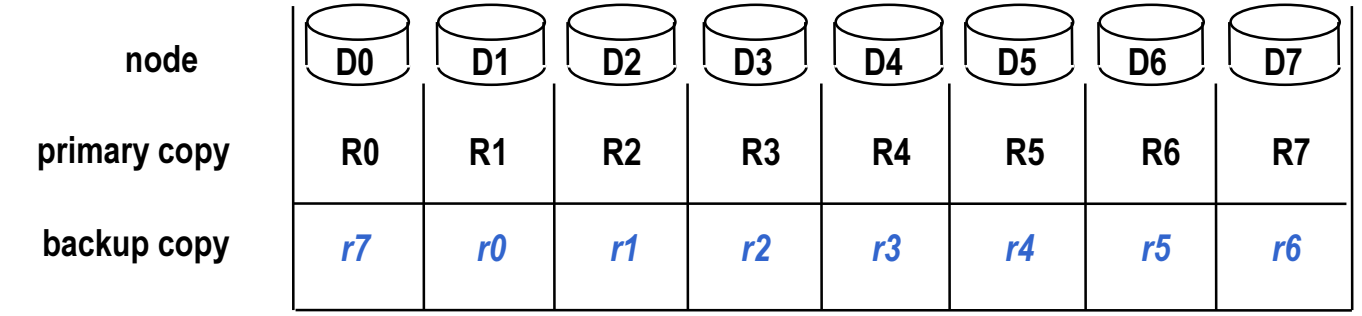
Interleaved Declustering (S=4)

Mirrored Declustering vs Interleaved Declustering

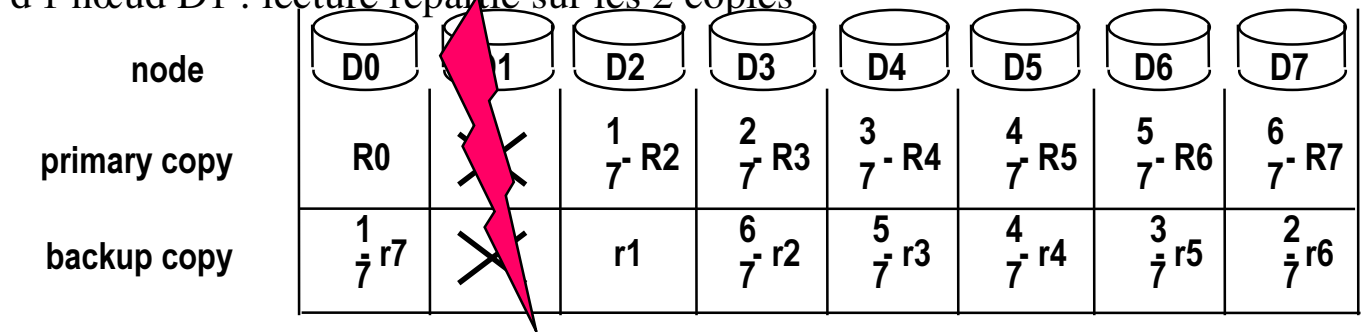
- Copeland and Keller 1989
- MD
  - Offre une grande simplicité et est universelle.
  - La reprise sur panne peut être implantée dans les couches basses de l'architecture logicielle/ matérielle (au niveau du contrôleur par exemple).
  - Pour des architectures qui ne partagent pas de disques, elle permet aux index (de clusters) locaux et globaux d'être indépendents.
  - De plus, MD ne requière pas que les données soient "clusterisées" (i.e. réparties sur plusieurs disques).
- ID
  - offre des améliorations significatives dans le temps de réprise, dans le temps moyen de pertes des 2 copies, dans le débit en fonctionnement normal, et dans le temps de réponse des opérations au cours de la reprise.
  - Pour toutes les architectures, ID permet aux données d'être réparties sur plus de 2 disques pour améliorer l'équilibre de charge entre les disques.

Chained Declustering [Gamma]

- Redondance des Données sur N sites
  - Tolérance à la Panne d'1 site
- Equilibre de la charge lors de la panne
  - Charge augmentée d' 1/N sur chaque site
- Fonctionnement
  - Hors Panne: lecture sur copie primaire



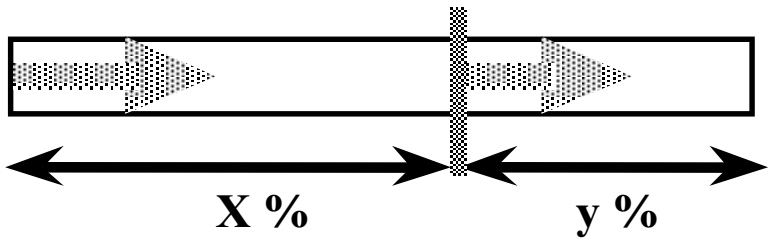
  - Panne d'1 nœud D1 : lecture répartie sur les 2 copies



Chained Declustering : Implantation (ii)

- Panne d'1 site
  - X % des lectures physiques sur la copie primaire
  - y % des lectures physiques sur la copie de secours
- 2 types de lectures logiques
  - SCAN
    - 1 scan logique = 2 scans physiques partiels par 2 nœuds différents

      Calcul de la page bornant les 2 scans



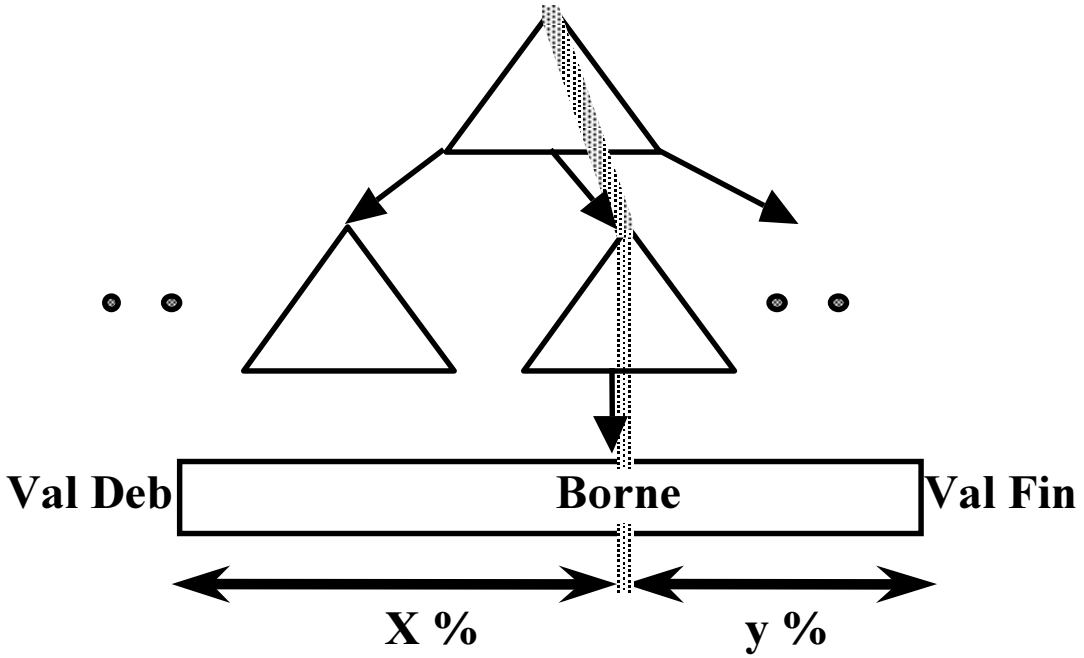
Chained Declustering : Implantation (ii)

INDEX (Restriction)

- 1 restriction dans l'index = 2 restrictions dans 2 sous-indexes par 2 noeuds
  - Déterminer dans l'index la valeur de borne
  - Rajouter à la Question l'opération RANGE-SELECT

    SELECT [ValDeb Borne [@ primaire

    SELECT [Borne ValFin] @ sécondaire



Parallélisme intra-opération

- Basé sur le parallélisme de données

Opérations unaires

- *sélection, projection, partition/agrégat*
- l 'opération est découpée en sous-opérations
- chaque sous-opération travaille indépendamment sur un fragment de la relation

Opérations binaires ou plus

- *jointure, star-join, ...*
- l 'opération est découpée en sous-opérations
- chaque sous-opération travaille indépendamment sur un fragment de la relation 1 et sur l 'ensemble des fragments de la relation 2

Calcul des Jointures

- 4 algorithmes parallèle de base pour la jointure
  - Boucle Imbriquée (Nested Loop)
    - pas de supposition
  - Associative
    - une des relations est distribuée sur l'attribut de jointure (équi-jointure)
  - Hachage
    - équi-jointure
  - Hybride [Gamma]
    - équi-jointure
- Certains algorithmes s'appliquent également aux autres opérations (intersection, élimination des doubles ...)

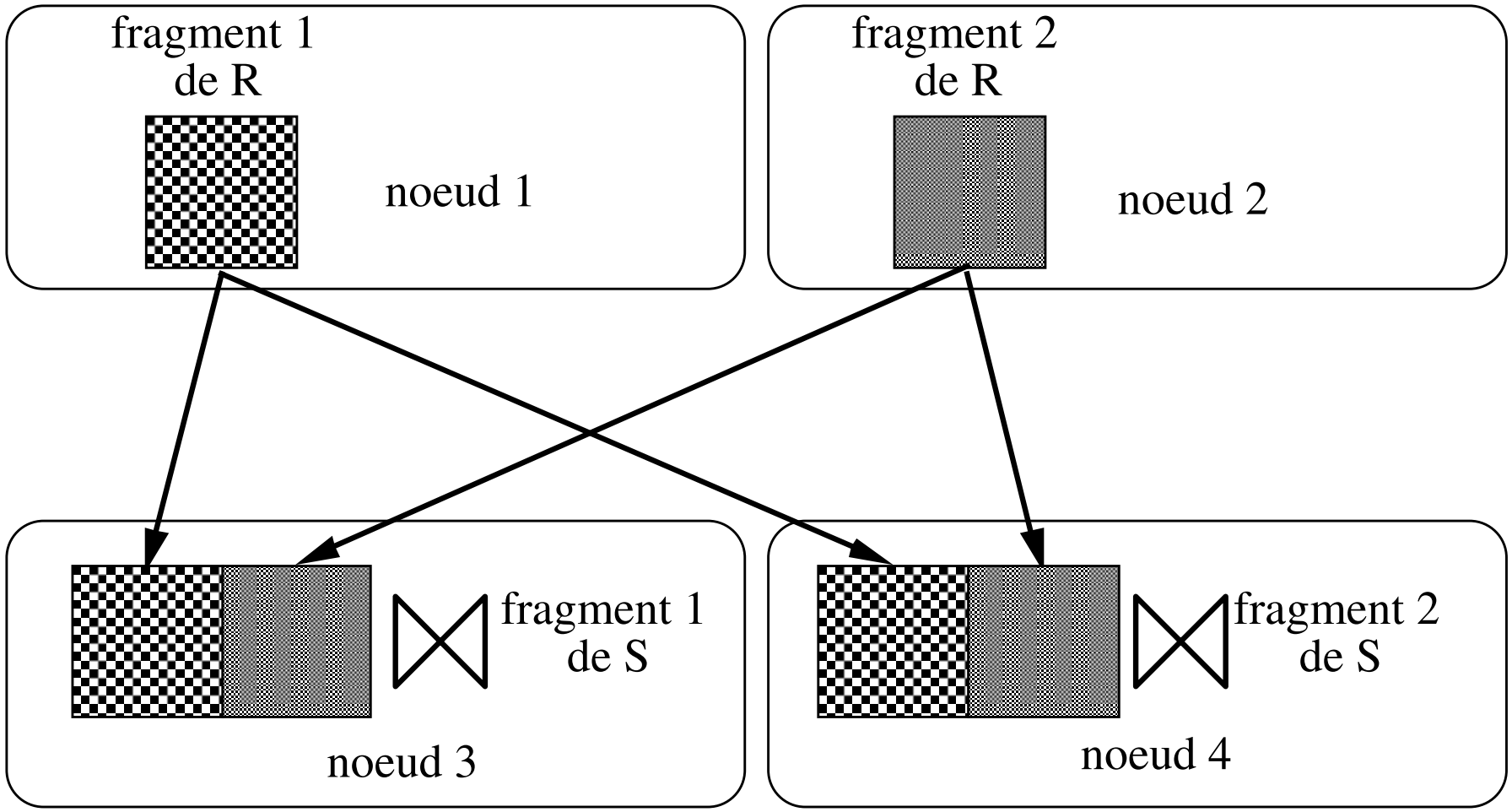
Jointure // par Boucle Imbriquée (Nested Loop)

- $S = S1 \cup S2 \qquad S1 \cap S2 = \emptyset$
- $R = R1 \cup R2 \qquad R1 \cap R2 = \emptyset$
- $R \ltimes S = (R \ltimes S1) \cup (R \ltimes S2)$

- $S = S1@node3 \cup S2@node4$
- $R = R1@node1 \cup R2@node2$

- transfert de R1 vers node3
- transfert de R2 vers node3
- transfert de R1 vers node4
- transfert de R2 vers node4
- $R \ltimes S = ((R1 \cup R2) \ltimes S1)@node3 \cup ((R1 \cup R2) \ltimes S2))@node4$

Jointure // par Boucle Imbriquée (Nested Loop)



Remarque au rédacteur

- Mettre des valeurs d ’attribut dans les tables
- Faire des paquets proportionnels en taille

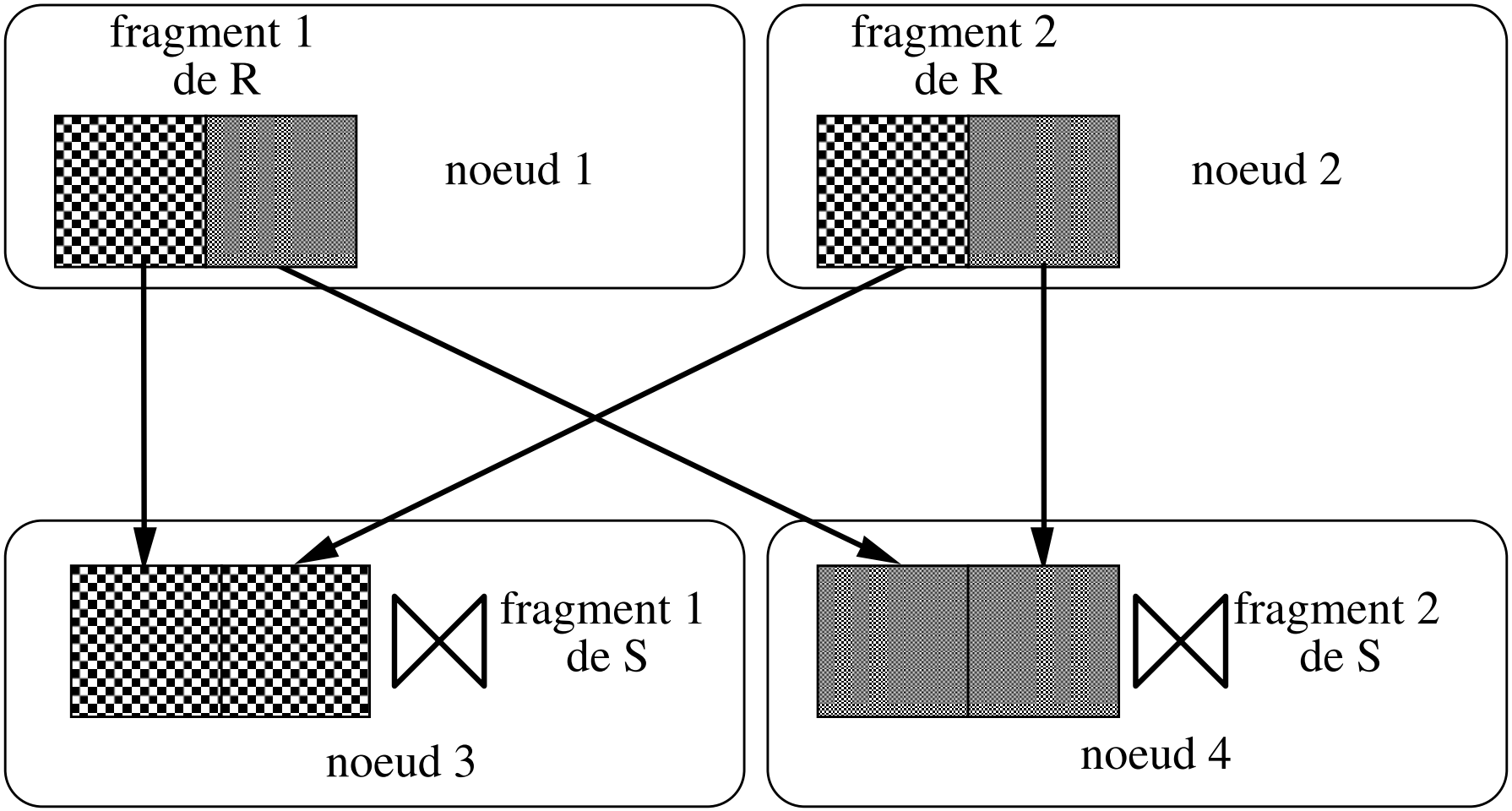
Jointure Parallèle Associative

- $S = S1 \cup S2$ $\quad S1 \cap S2 = \emptyset$
- $R = R11 \cup R12 \cup R21 \cup R22$ $\quad R11 \cap R12 \cap R21 \cap R22 = \emptyset$
- $R \ltimes S = ((R11 \cup R21) \ltimes S1) \cup ((R12 \cup R22) \ltimes S2)$

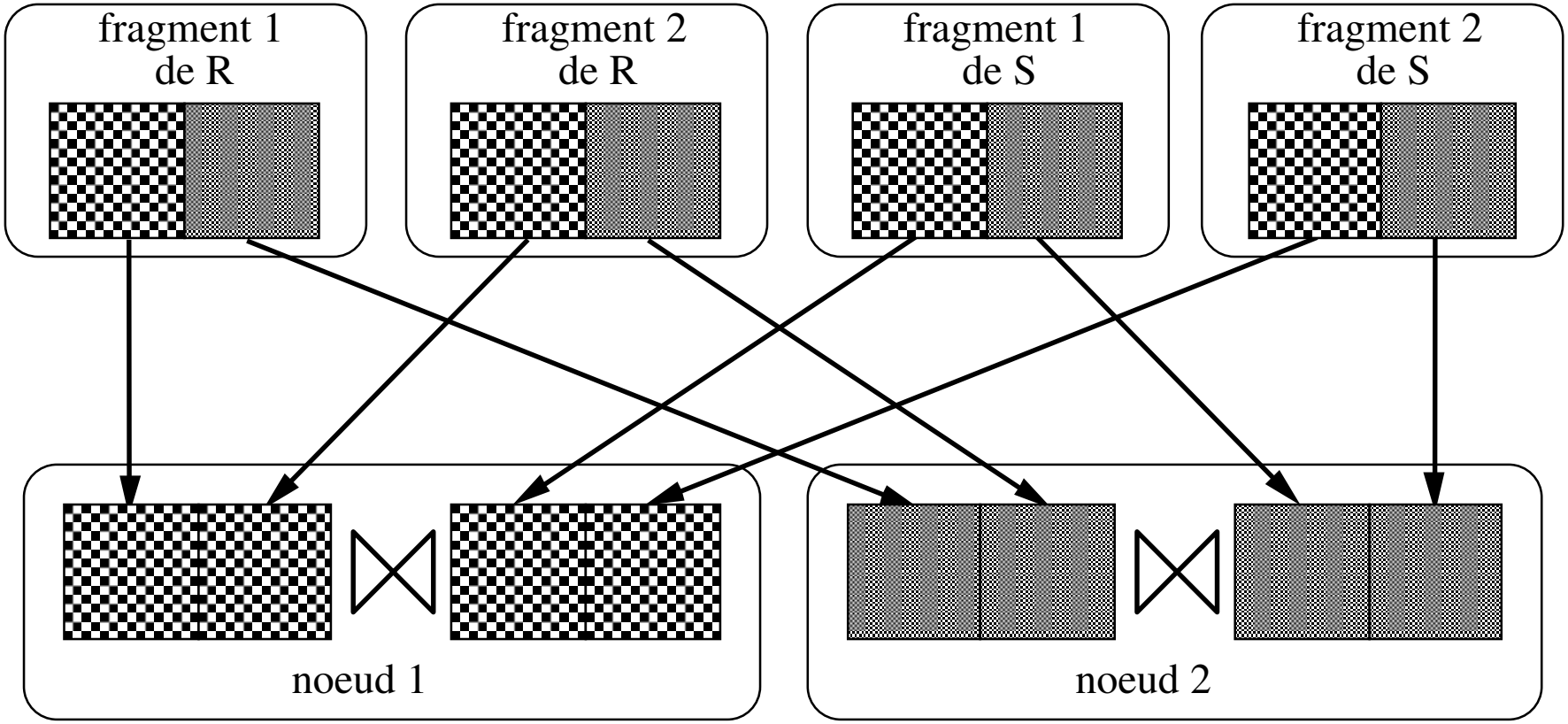
- $S = S1@node3 \cup S2@node4$
- $R = (R11 \cup R12)@node1 \cup (R21 \cup R22)@node2$

- transfert de R11 vers node3
- transfert de R21 vers node3
- transfert de R12 vers node4
- transfert de R22 vers node4
- $R \ltimes S = ((R11 \cup R21) \ltimes S1)@node3 \cup ((R12 \cup R22) \ltimes S2))@node4$

Jointure Parallèle Associative



Jointure Parallèle par Hachage



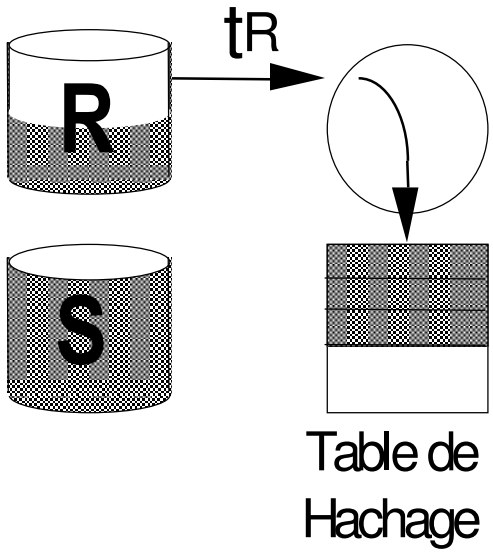
Jointure Hybride Parallèle [Gamma] (i)

Rappel Jointure Hybride

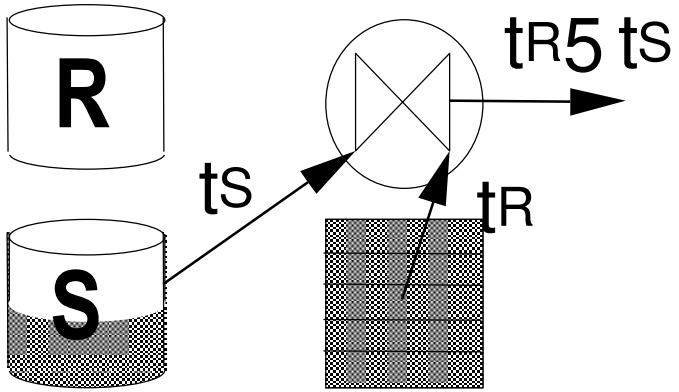
- bonne performance ∀ mémoire allouée

2 phases

Building Phase

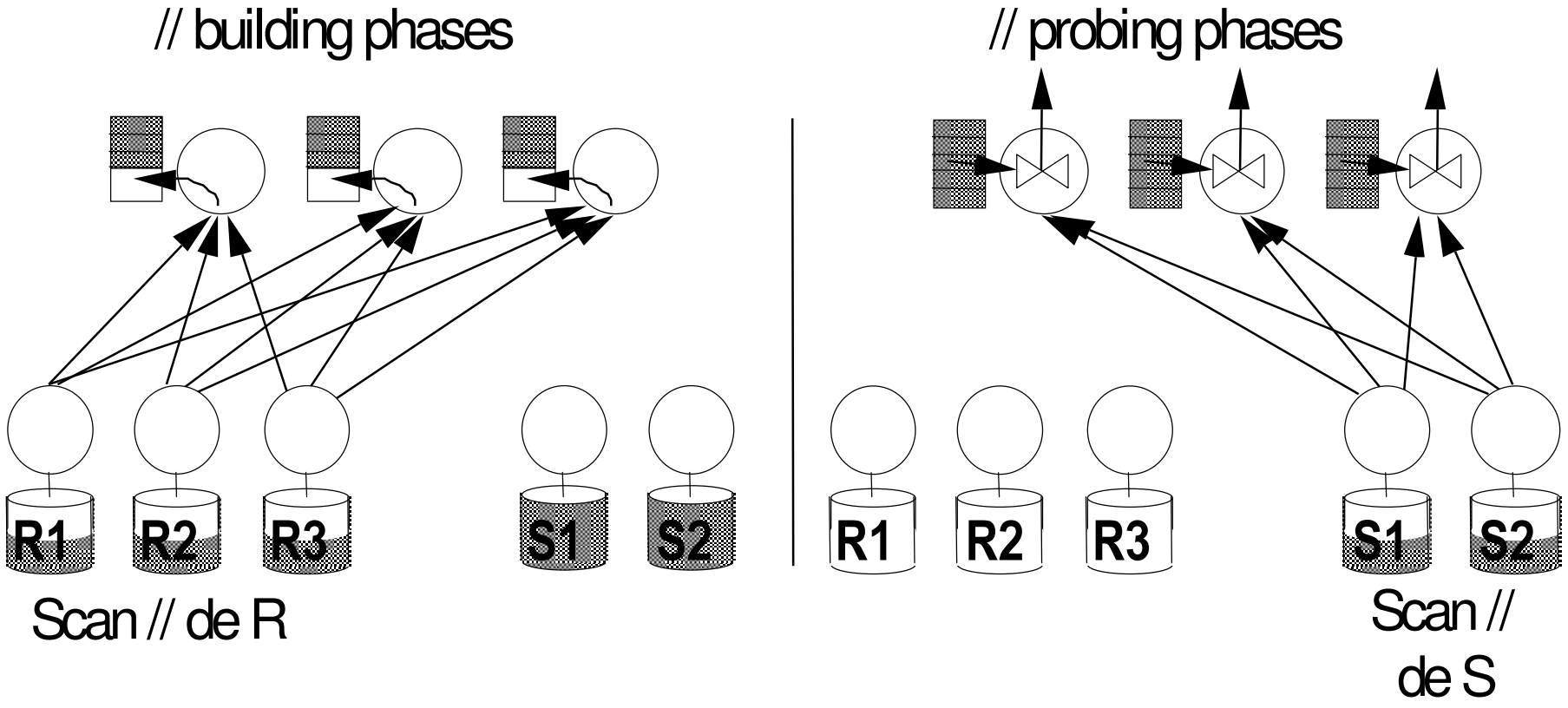


Probing Phase

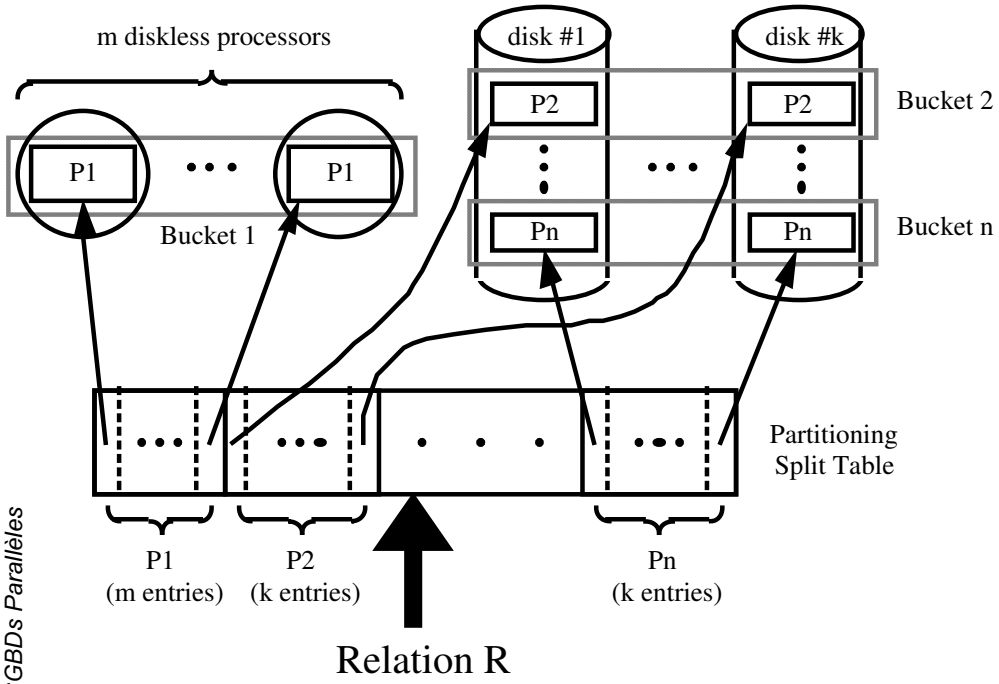


- plusieurs phases si R ne tient pas dans la table de hachage

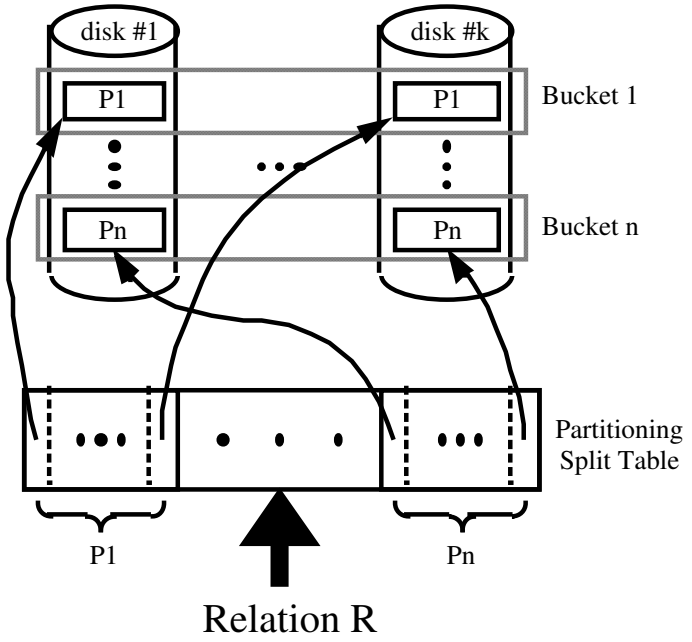
Jointure Hybride Parallèle [Gamma] (ii)



Jointure Hybride Parallèle [Gamma] (iii)



Partitioning of R into N logical buckets for Hybrid-Hash-Join



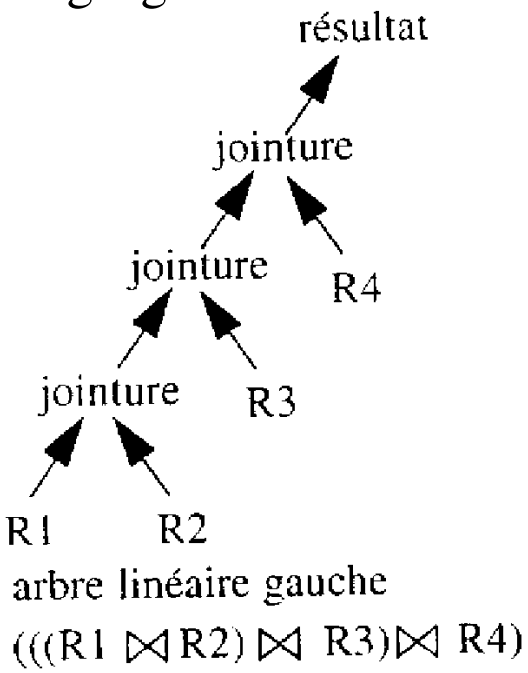
Partitioning of R into N logical bucke for Grace-Hash-Join

Optimisation Parallèle des Requêtes

- Elaboration des plans d'exécution
  - Degré de parallélisation des opérateurs
    - = F(distribution des tables sources)
  - Placement des opérations sur les nœuds
    - = F(Charge courante, localité des sources et des résultats intermédiaires)
- Alternatives de plan d ’exécution
  - arbres d'exécution dataflow
- Choix du plan d'exécution optimal
  - Fonction à optimiser :
    - Temps de Réponse (OLAP) vs Débit de Transaction (OLTP)
- Modèle de coût
  - schéma physique (statistiques)

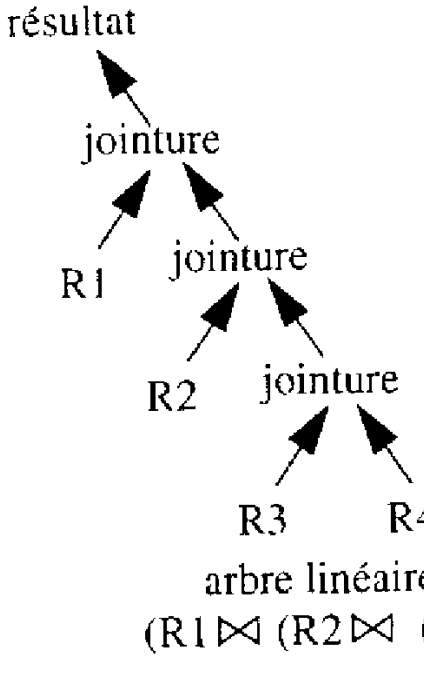
Alternatives de plan d ’exécution

- Types d’arbres
  - left-deep
  - right-deep
  - bushy
  - right-deep segmenté
  - zigzag
- Défini un plan d'exécution dataflow



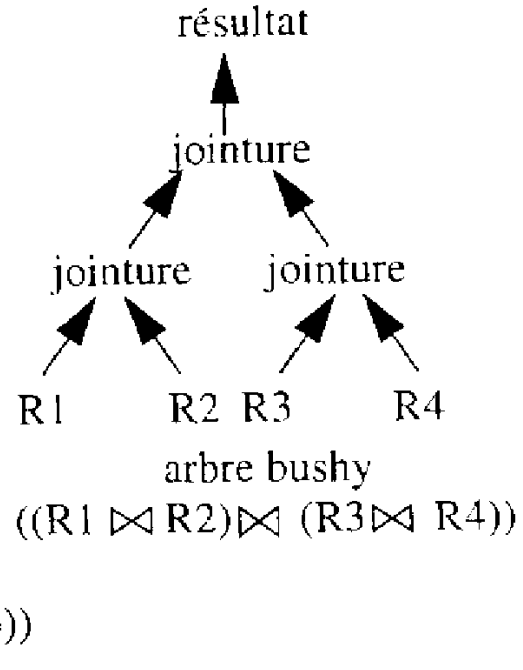
arbre linéaire gauche

(((R1 ⋈ R2) ⋈ R3)⋈ R4)



arbre linéaire droit

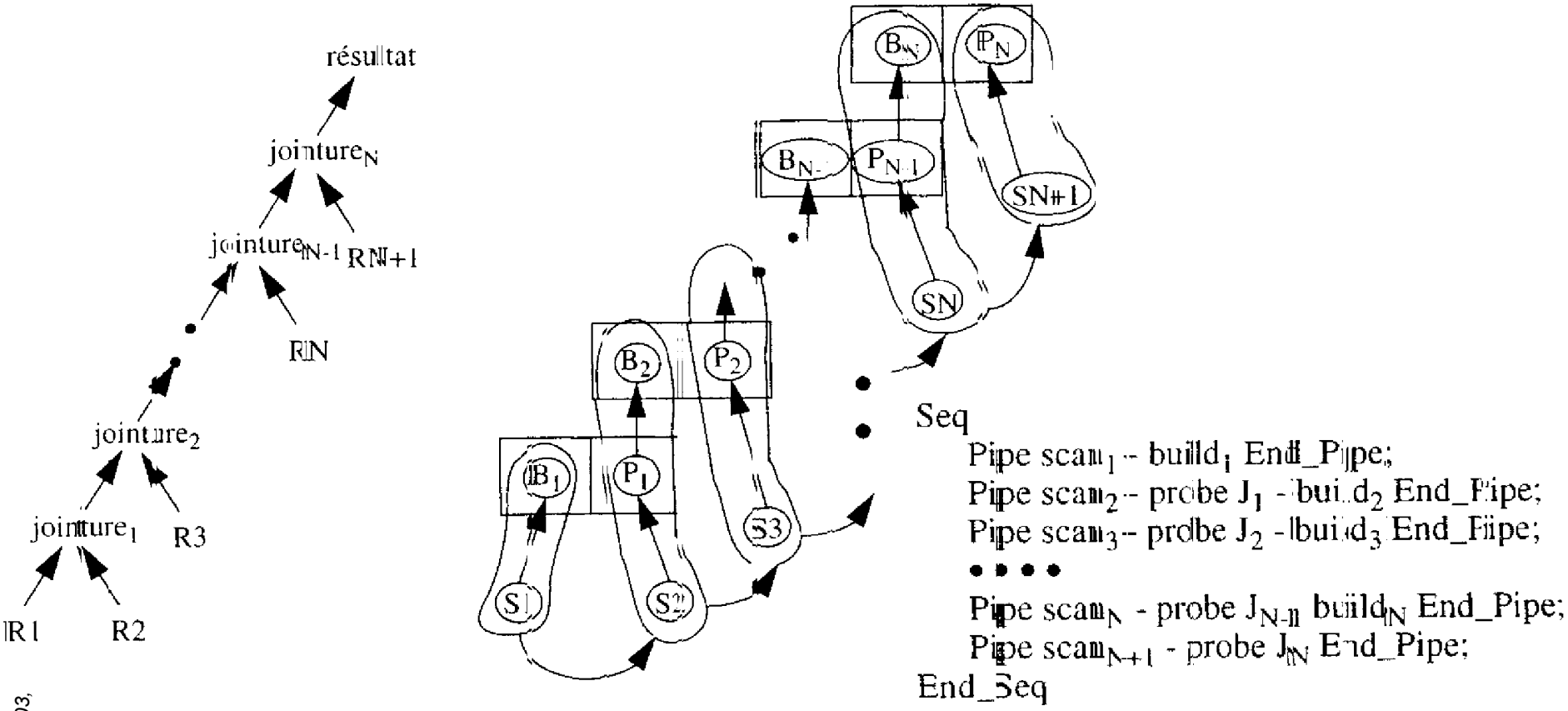
(R1⋈ (R2⋈ (R3⋈ R4)))



arbre bushy

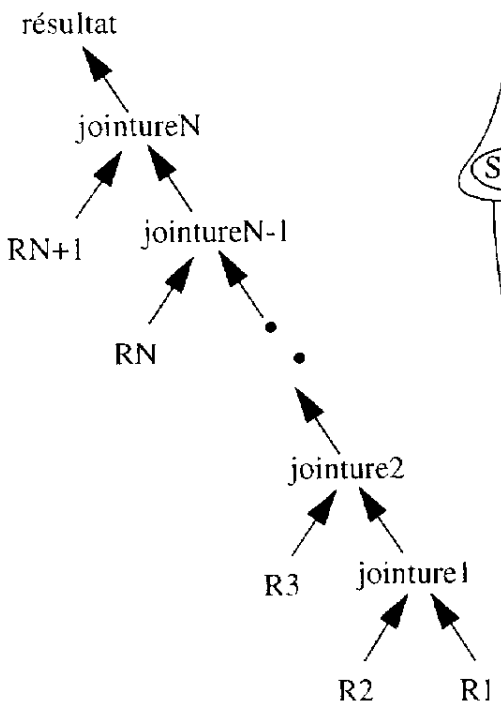
((R1 ⋈ R2)⋈ (R3⋈ R4))

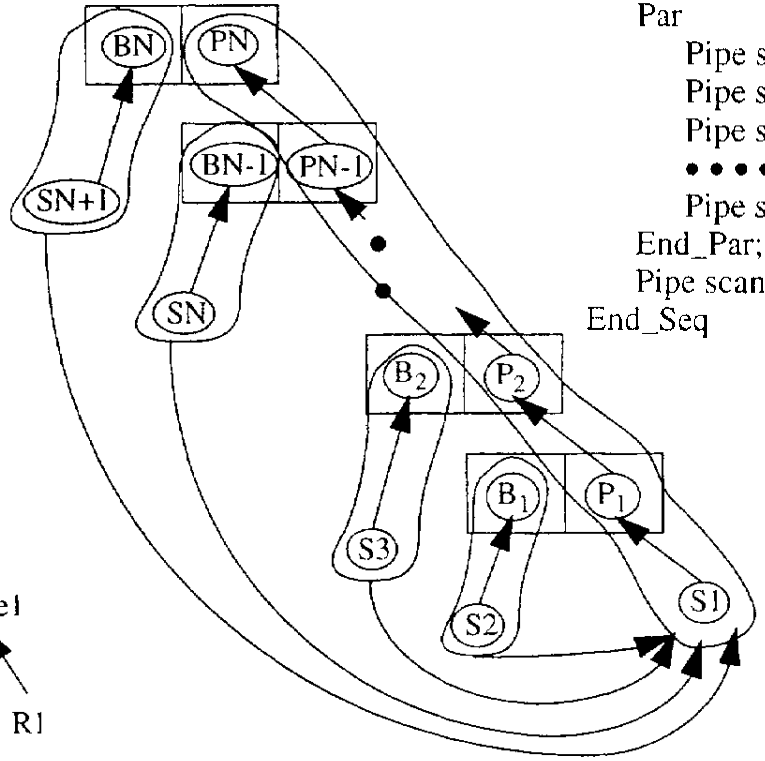
Left-Deep Tree (arbre linéaire gauche)



- Peu de parallélisme inter-opération
- Nombreuses phases

Right-Deep Tree (arbre linéaire droit)



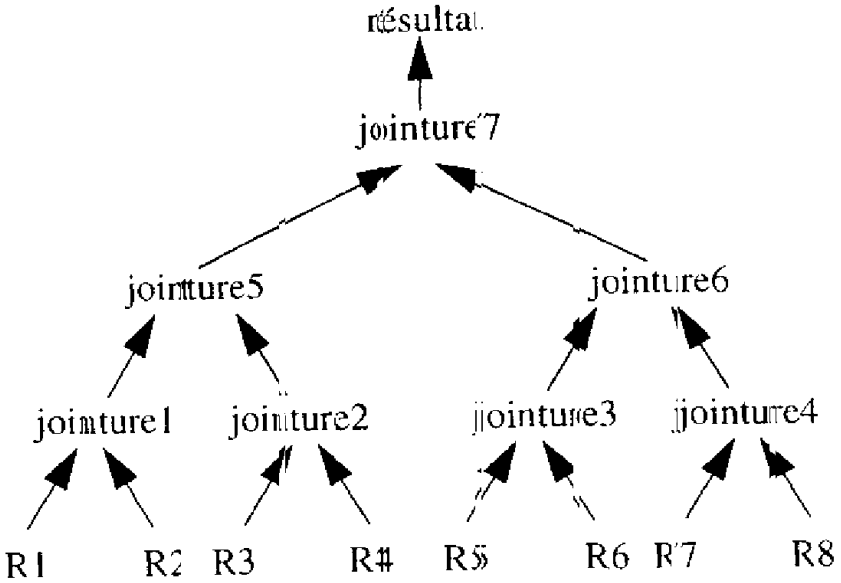


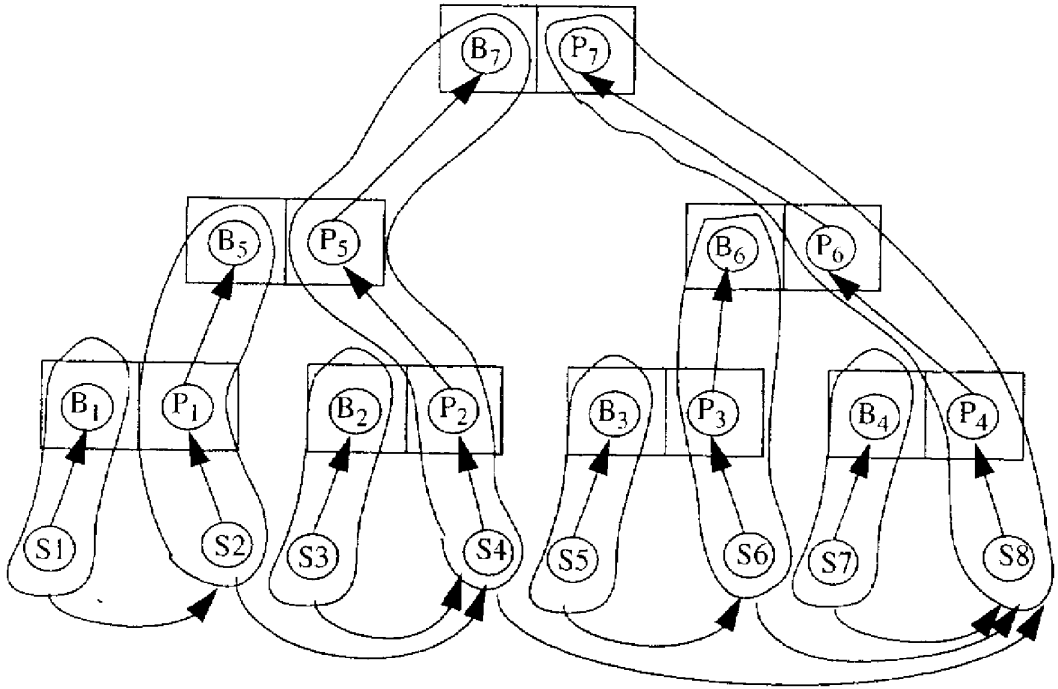

```
Seq
    Par
        Pipe scan2 - build1 End_Pipe
        Pipe scan3 - build2 End_Pipe
        Pipe scan4 - build3 End_Pipe
        ••••
        Pipe scanN+1 - buildN End_Pipe
    End_Par;
    Pipe scan1 - probe J1 - probe J2- ••• probe JN End_Pipe;
End_Seq
```

- Améliore le speedup (Temps de réponse)
- Beaucoup de parallélisme inter-opération
- Nécessite une grande quantité de ressource

Bushy-Tree

- Equilibre entre left-deep tree et right-deep tree
  - Temps de réponse
  - Consommation des ressources





- Exercice : Donnez l 'ordonnancement de cet arbre

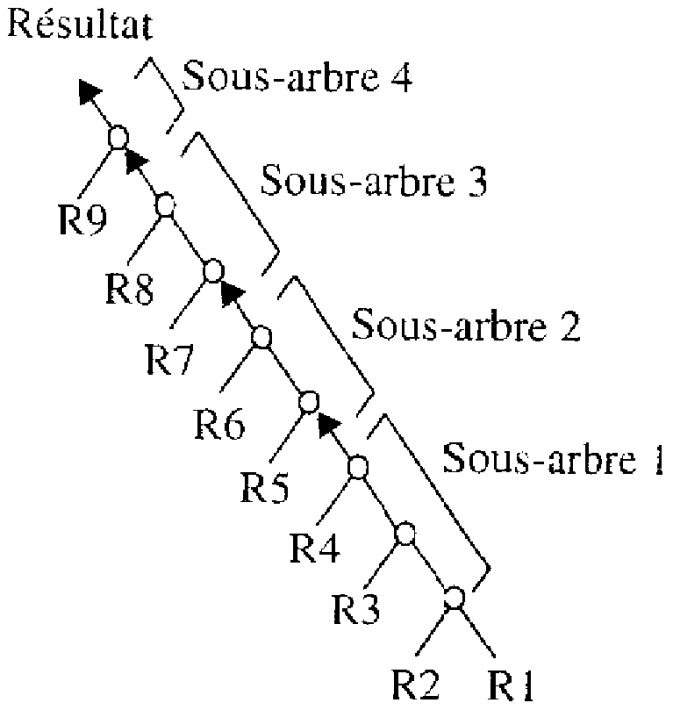
Bushy-Tree : l 'ordonnancement

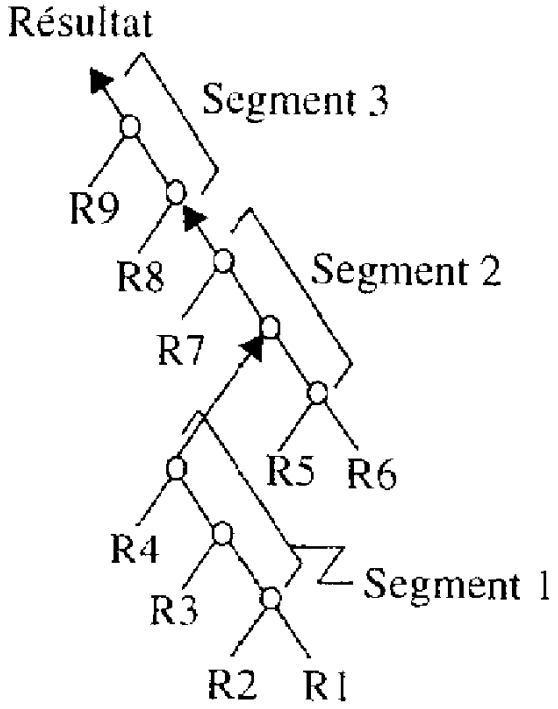
Réponse à l 'exercice

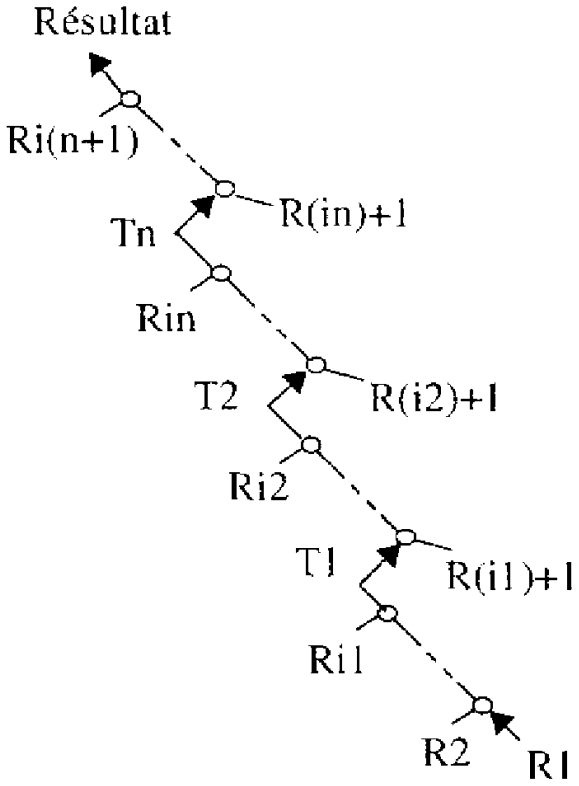
Seq
    (E1) Par
        Pipe $scan_1$ - $build_1$ End_Pipe
        Pipe $scan_3$ - $build_2$ End_Pipe
        Pipe $scan_5$ - $build_3$ End_Pipe
        Pipe $scan_7$ - $build_4$ End_Pipe
    End_Par;
    (E2) Par
        Pipe $scan_2$ - probe J_1 - $build_5$ End_Pipe
        Pipe $scan_6$ - probe J_3 - $build_6$ End_Pipe
    End_Par;
    (E3) Pipe $scan_4$ - probe J_2 - probe J_5 -$build_7$ End_Pipe
    (E4) Pipe $scan_8$ - probe J_4 - probe J_6 - probe J_7 End_Pipe
End_Seq

Arbre linéaire droit segmenté (i)

- But : tenir compte de la limitation mémoire
  - découpé l 'arbre droit en sous-arbres droits







- Zig-zag : Arbre linéaire droit découpé
  - le résultat d 'un sous arbre est utilisé comme source pour la phase de build du sous arbre suivant

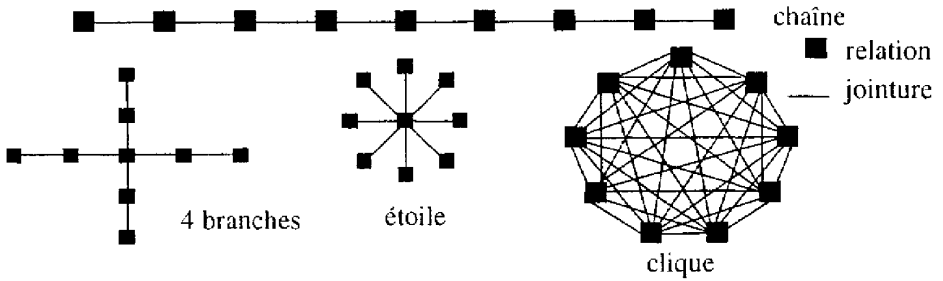
Arbre linéaire droit segmenté (ii)

Heuristiques de construction

- Minimal Work (MW)
  - sélection des relations d ’un segment pour un temps de réponse minimal du segment
- Balanced Consideration (BC)
  - idem BC mais élimine la sélection des petites relations dans le premier segment par un mécanisme de pénalité

Choix d ’un arbre optimal

- Types de requêtes
- Espace des arbres

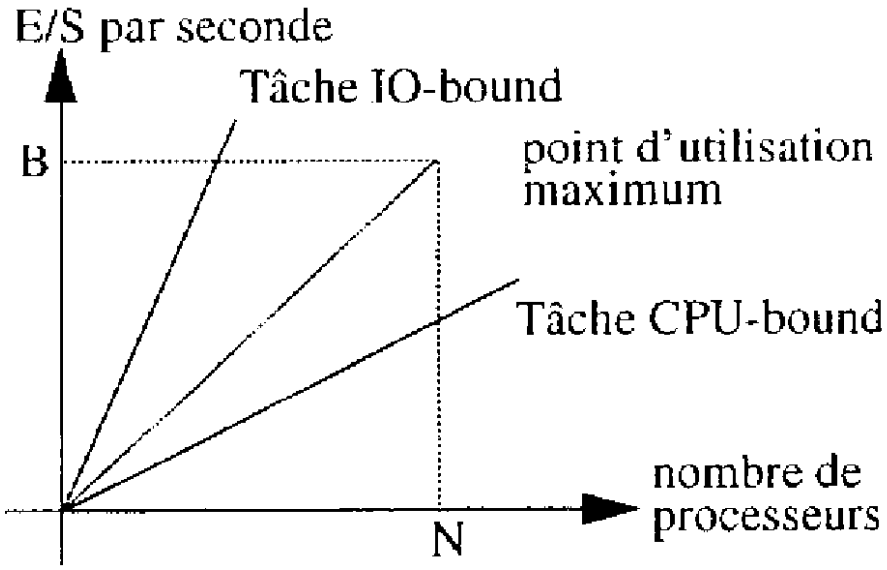


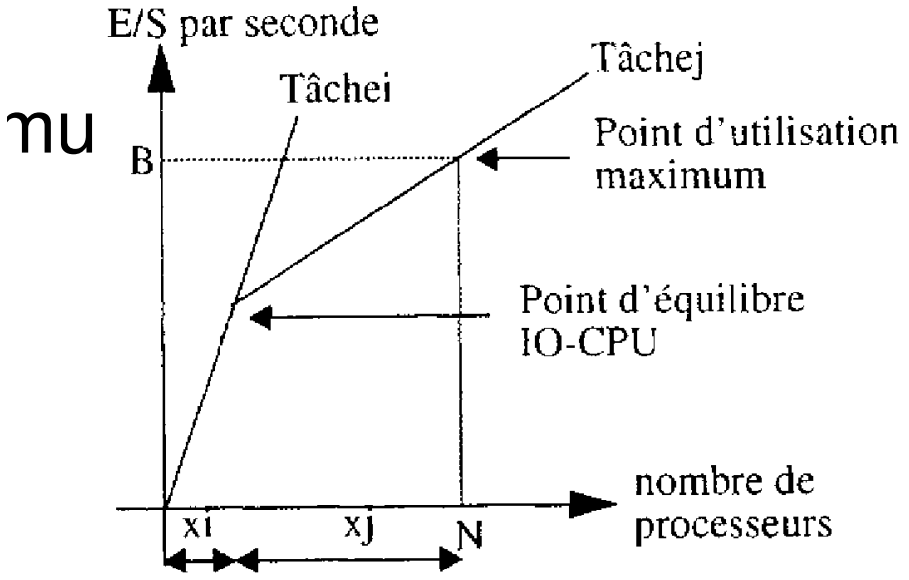
Nombre de relations N	Espace linéaire gauche (ou droit)			Espace bushy		
	chaîne 2^{N-1}	étoile $2(N-1)!$	clique $N!$	chaîne $\frac{2^{N-1}}{N}\binom{2N-2}{N-1}$	étoile $2^{N-1}(N-1)!$	clique $\frac{(2N-2)!}{(N-1)!}$
3	4	4	6	8	8	12
4	8	12	24	40	48	120
5	16	48	120	224	384	1.680
6	32	240	720	1.344	3.840	30.240
7	64	1.440	5.040	8.448	46.080	665.280
8	128	10.080	40.320	54.912	645.120	17.297.280
9	256	80.640	362.880	366.080	10.321.920	518.918.400
10	512	725.760	3.628.800	2.489.344	185.794.560	17.643.225.600

- Optimisation
  - heuristiques, recuit-simulé, génétique, recherche tabou

Classification des tâches

- Tâche IO-Bound (bornée par les E/S)
- Tâche CPU-Bound (bornée par le nombre de processeurs)





Gestion de l'exécution (Query Manager)

- Ordonnancement intra-requête
  - Déclenchement des phases dataflow d 'une requête
    - Déclenchement des sous opérations dans chaque phase

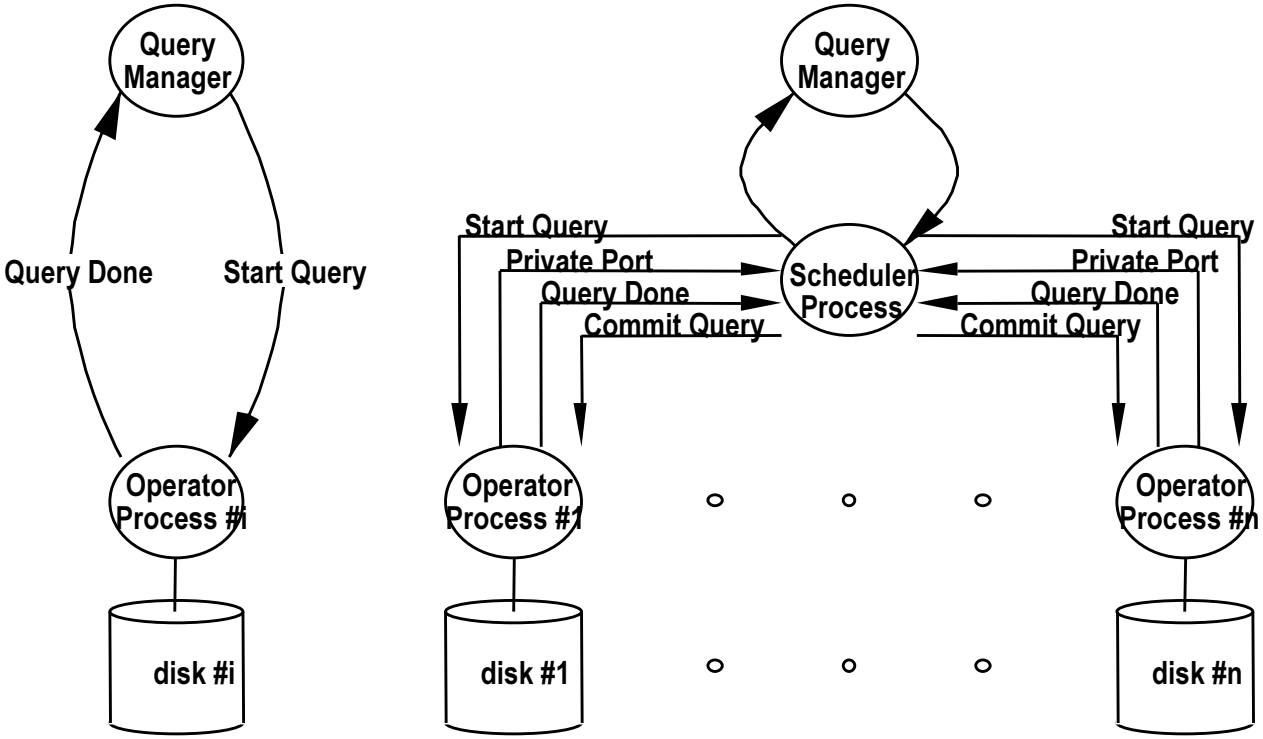
      seules les sous-opérations ayant un calcul (i.e. des données sources) sont activées

      voir Gamma et Bubba
- Ordonnancement inter-requêtes
  - déclenchement des opérations des différentes requêtes
  - en fonction des ressources (mémoire et CPU) disponibles

- NB : Ré-évaluation d 'un plan d'exécution en cours d'exécution
  - en fonction du biais, de la taille réelle des résultats intermédiaires et des ressources disponibles

Gestion centralisée de l 'exécution [Gamma]

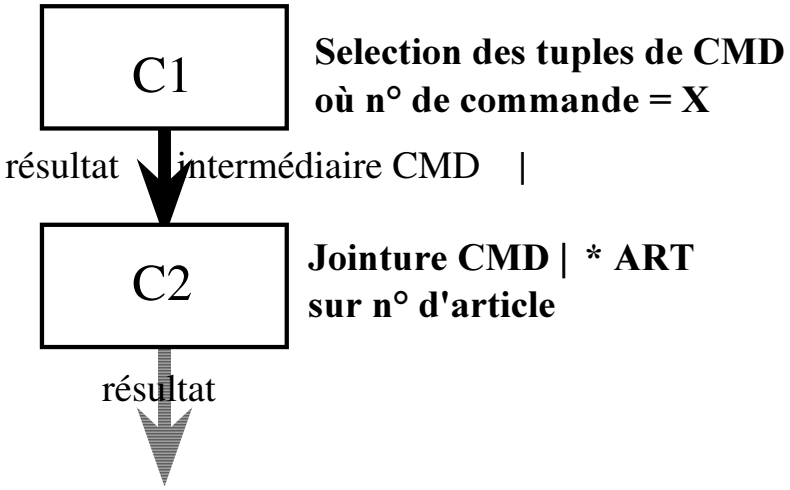
- Question mono-site
  - QM envoie la question directement au processeur approprié (déterminé au Runtime)
- Question multi-site
  - QM envoie la question compilée au SP
  - SP active les OP sur les processeurs sélectionnés pour l 'exécution

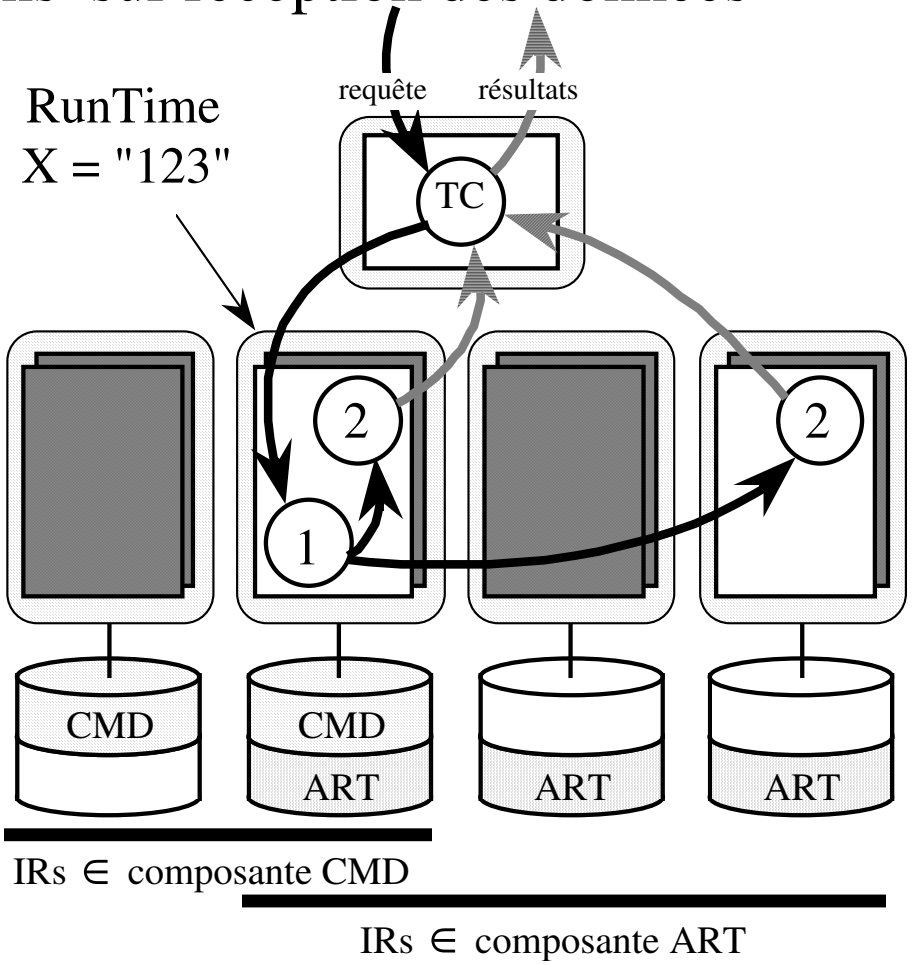


Gestion Dataflow de l ’exécution [Bubba]

Enchaînement Dataflow

- Activation des sous opérations sur réception des données sources pour le calcul



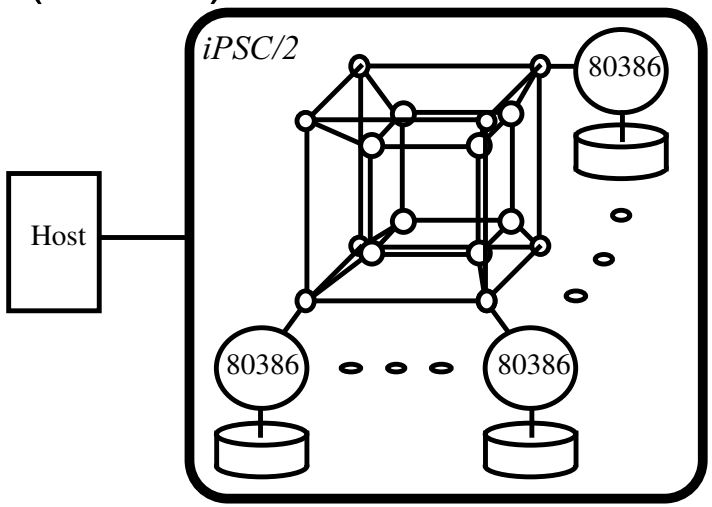


Les SGBDs Parallèles

- Les Prototypes de Recherche
  - EDS et DBS3 (Bull)
  - Gamma (U. du Wisconsin)
  - Bubba (MCC, Austin)
  - XPRS (U. de Berkeley)
  - GRACE (U. de Tokyo)
- Les Produits Parallèles
  - Teradata (ATT-NCR GIS)
  - NonStopSQL (Tandem/Compaq)
- Les Extensions Parallèles
  - Oracle
  - Informix
  - IBM-DB2, Ingres, Sybase-Navigator, MS SQL Server...
- Les Entrées/Sorties Parallèles
  - RAID (Patterson)

Gamma (Univ. du Wisconsin)

- Machine BD parallèle expérimentale (87-90)
  - parallélisme Shared Nothing moyen
  - Cluster de Vax/VMS
  - Hypercube iPSC/2



- Placement
  - Full Declustering, Hybrid Range Partitionning
  - Chained Declustering pour la tolérance aux pannes
- Jointure Hybride Parallèle
  - 2 phases
- Optimisation
  - Arbres linéaires droits et gauches, Arbres bushy
- Exécution Dataflow

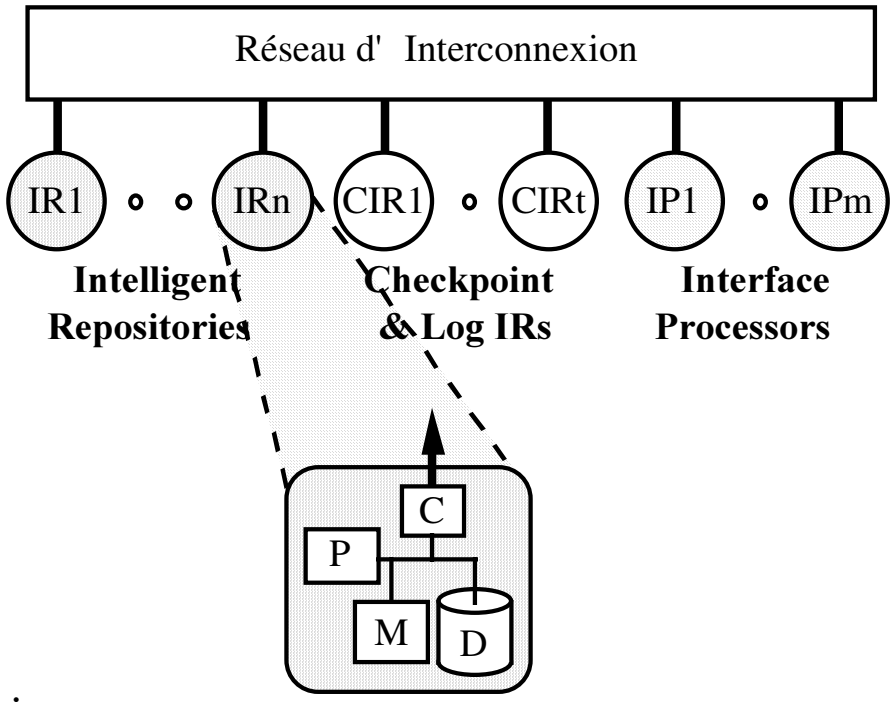
Bubba (MCC, Austin)

- Machine BD massivement parallèle expérimentale (87-90)

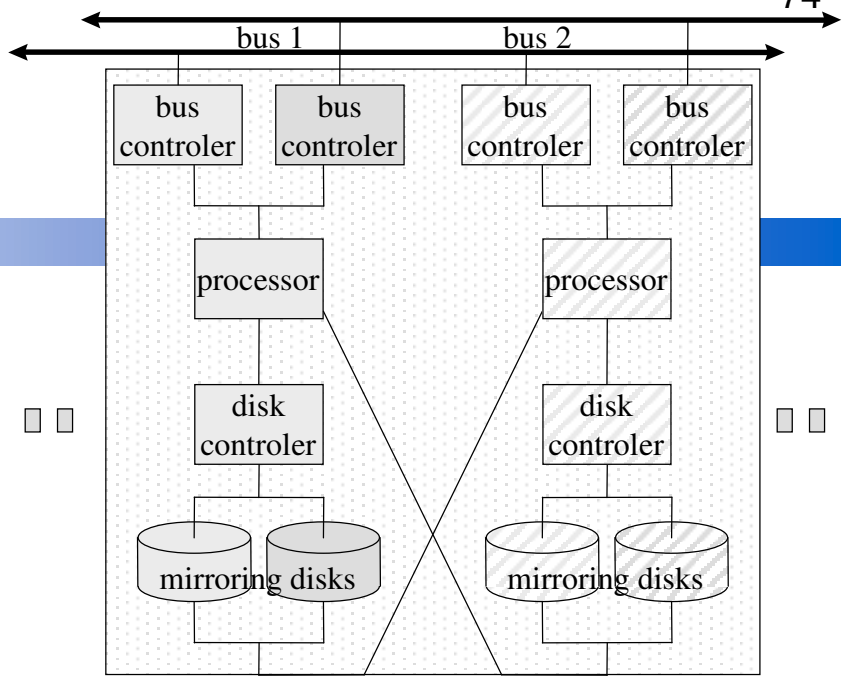
  Shared Nothing

  jusqu'à 1000 processeurs

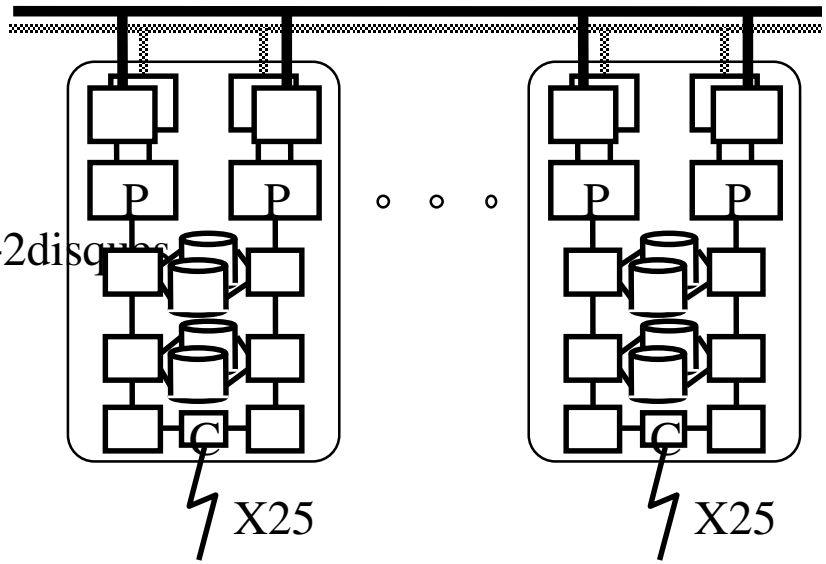
- But Exploiter et étudier le parallélisme massif
  - Partial Declustering
    - Placement redondant des données
    - heuristique de placement
  - Parallélisation automatique
  - Contrôle DataFlow de l'exécution
    - activation dataflow des sous opérations
  - Techniques de reprise sur panne
  - Langage BD général
  - Gestion d'objets / Support OS



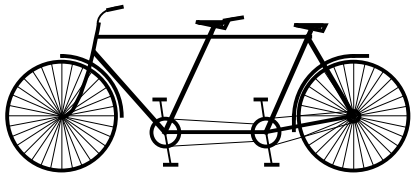
Compaq-Tandem Architecture



- Gamme Non Stop Himalaya
  - K100 (2 ~ 1020 proc.)
  - K1000 (2 ~ 4080 proc. R3000)
  - K10000 (2 ~ 4080 proc. R4400)
- Disponibilité Continue (Non Stop)
  - Tolérance aux Pannes
  - Haute disponibilité
  - Transfert de la technologie vers OpenVMS
- Redondance du Hardware
  - No SPOF (No Single Point Of Failure)
    - 1 Nœud = 2processeurs+2contrôleurs+2disques
  - Choix du Shared-Nothing
    - Isolation des Pannes Logicielles
      Fast-Fail
  - Disponibilité des Données
    - Disques "Miroirs"
      D1=D3 et D4=D2



Compaq-Tandem Non Stop SQL



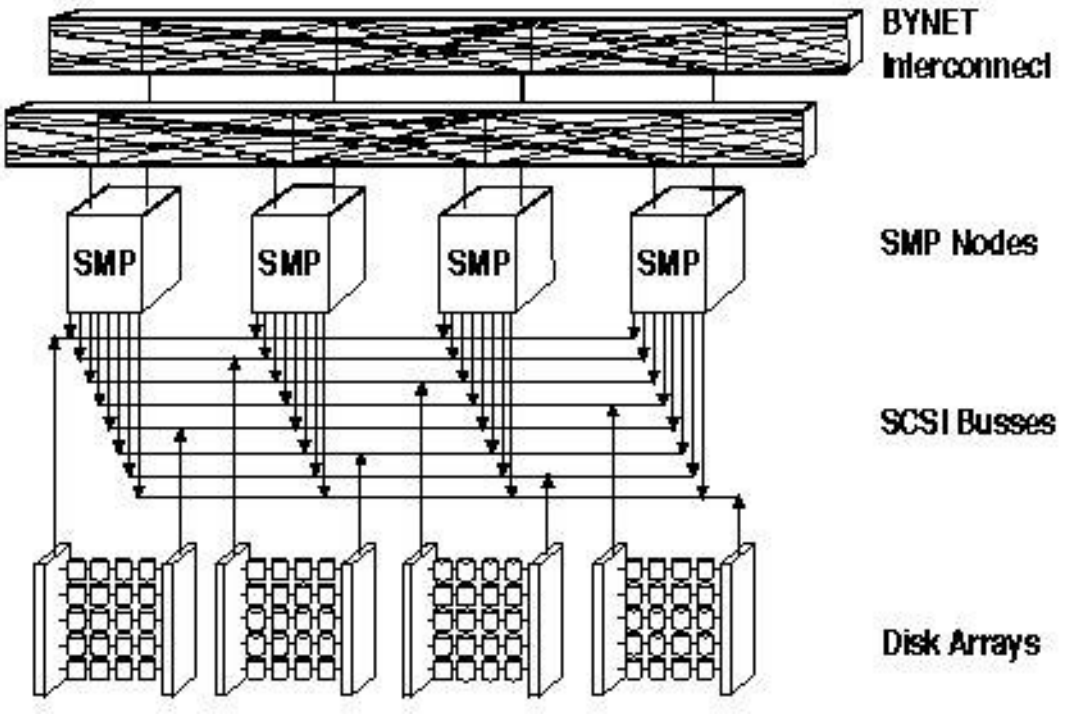
- En Général, 1 site en panne
- => Toutes les transactions du site échouent

Mécanisme de processus "pairs" (kernel)

- 1- Exécution de l'application A
  - Création du processus X (primaire) sur 1 site.
  - Création du processus Y (de secours) sur le site pair
- 2- Seul X s'exécute
  - Y reçoit l'état de X à des points de checkpoint
- 3- Le système signale que X est mort
  ou Quand X n'envoie plus de messages "Je suis Vivant!"
  - Y reprend l'exécution de A au dernier checkpoint.

Utilisable et Utilisé par les applications et par le système (processus I/O Miroir)

Teradata (NCR)



- Initialement
  - SMP
- Worldmark (2003)
  - Matériel
    - MPP (jusqu'à 512 P4)
    - Interconnexions redondantes '*BYNET*'

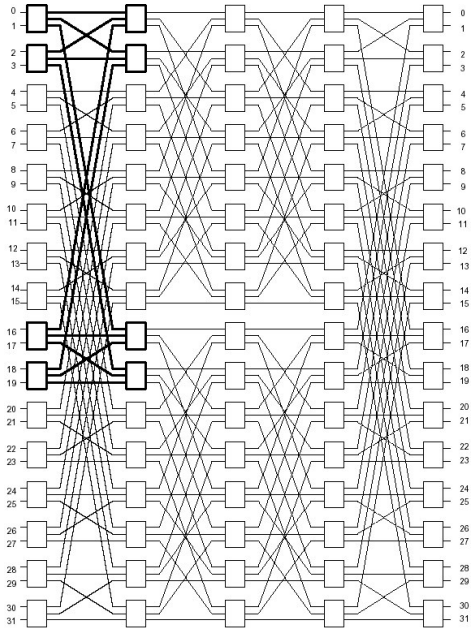
      switché Folded banyan

      jusqu'à 4096 nœuds (120 Mo/s par nœud)

      point-à-point, multicast, broadcast

      sémaphores globaux

      (First done, last done" and counting)
  - SGBD



Informix

- Monde en 93 : CA 353 M$
- France en 93 : CA 77 MF

Moteur SGBD

- Informix Workstation & Workgroup Server
  - Transactionnel
    (Petites et Moyennes Applications)
- Informix 6,7- Online
  - Transactionnel lourd, Décisionnel
    (Grosses Applications, Disponibilité 24/24, ...)
- Informix 9 -Universal Server
  - Relationnel Objet
    (Multimédia, …)

Outils

- Administration, Développement (HyperScript, ViewPoint)

Le Parallélisme chez Informix

- Online 6.0
  - MultiThreading (couplage fort)
  - Parallel Data Query (PDQ)
    - Fonctions Parallélisées
      - Tri, Jointure ... (10-30 fois plus performants)
      - Construction des Index, Archivage et Reprise
- Online 7.0 (SEQUENT)
  - Parallélisme Intra-RequÍte
- Online 8.0
  - Dynamic Scalable Architecture (DSA)
    - machines à couplage lâche
- Online XPS
  - eXented Parallel Server

Online XPS (eXented Parallel Server)

- Architecture Parallèle
  - Cluster, MPP (Couplage faible) et nouveau SMP (Couplage fort)
- Fonctionnalité
  - OLTP lourd et Décisionnel (Data Warehousing)
- Partitionnement
  - Hachage, Prédicat, Round-Robin (Tourniquet)
- Parallélisme
  - Parallélisme intra requête et intra opération
    - Pipeline
- Tolérance aux Pannes
  - Pour architecture en cluster
- Fusion de XPS et de Universal Server
  - Objet Relationnel et Parallélisme

Oracle

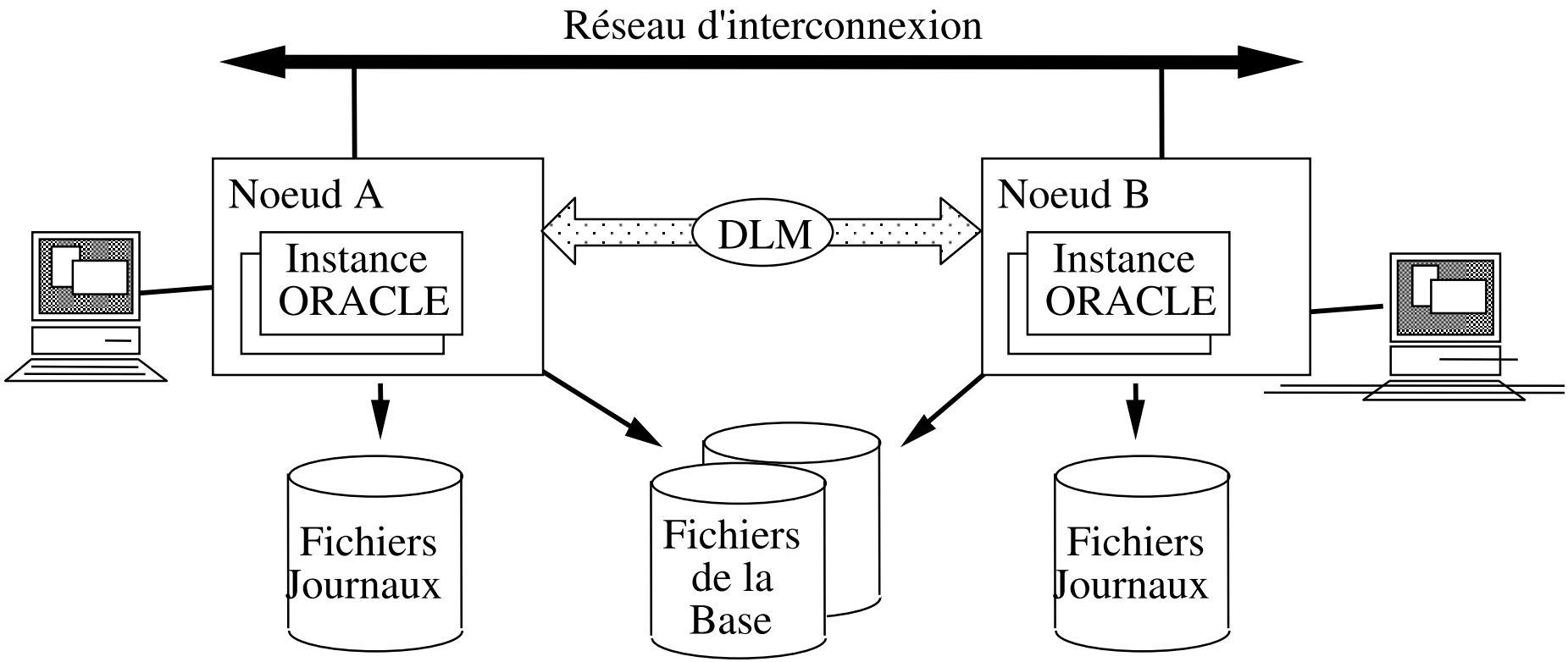
- 3éme éditeur mondial (après MicroSoft et Computer Associate)
- Monde en 98: 7500M$ de CA et 37000 personnes

L'offre parallèle d'Oracle

- Oracle V7
  - Multithreading
- Oracle V7.1
  - Oracle7 Parallel Server
  - Oracle7 Parallel Query Option
- Oracle V8
  - Décisionnel
  - Objet-Relationnel
  - NCA
  - Web et Commerce Electronique

Oracle Parallel Server

- adapté aux architectures en grappe (cluster)



Oracle Parallel Query Option

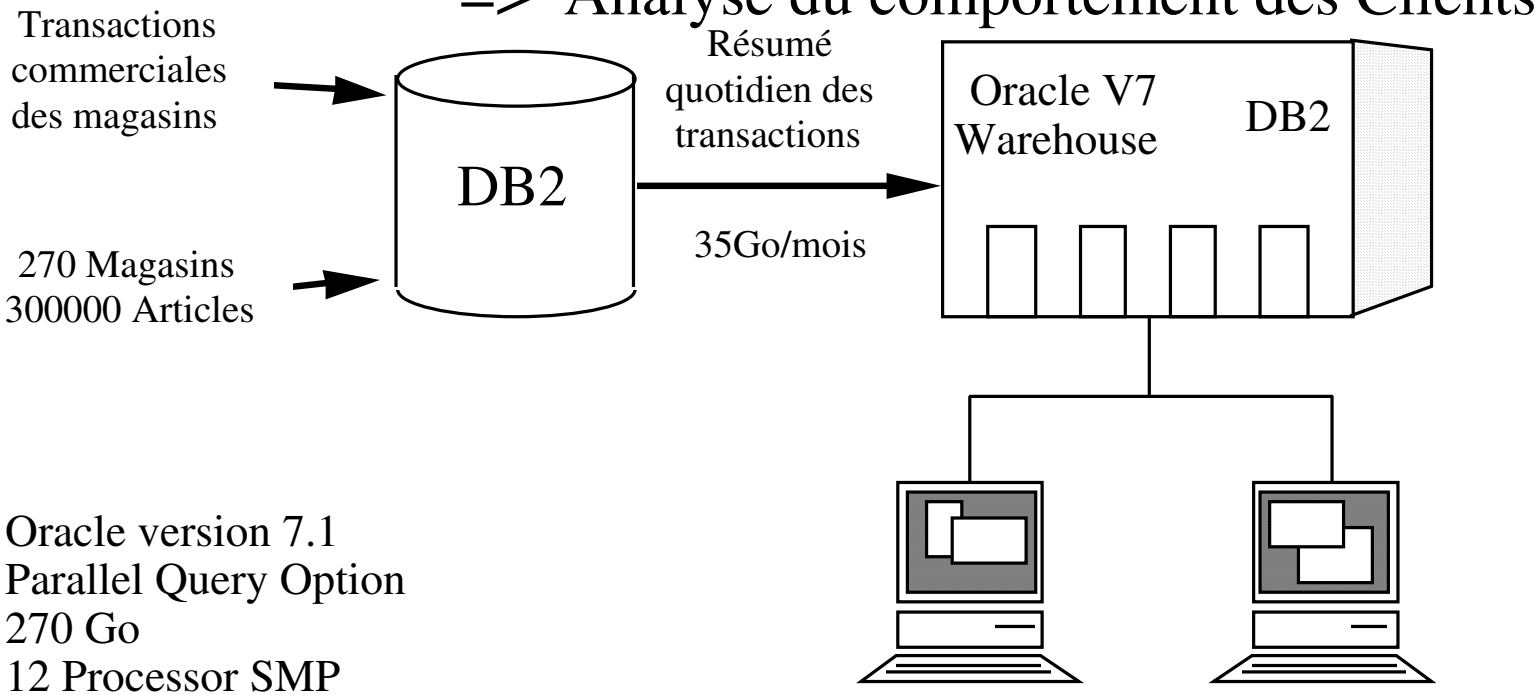
- Architectures Clusters ou Shared Nothing
- Partitionnement Dynamique
  - Striping (Bande) ou Hachage sur la clé
  - Equilibrage Dynamique
- Parallélisation des fonctions
  - balayages, tris, jointures, star-join, agrégats
  - index B-Tree et BitMap, archivage, reprise
- Parallélisation inter et intra requête
  - Pipeline, Synchronisation des Caches entre les nœuds
- Requête Parallélisable (Option) pour le OLCP
  - But: Aide à la décision, infocentre

Oracle : l'exemple d'un InfoCentre

Mervin's (USA) en 94

- Aide à la Décision

=> Analyse du comportement des Clients



- Oracle version 7.1
- Parallel Query Option
- 270 Go
- 12 Processor SMP

Oracle7 : l'exemple d'un InfoCentre

ING Bank International (Pays Bas)

- Configuration actuelle (94):
  - 6 ES9000, 10 millions de transactions / jour DB2, IMS,CICS, VSAM
  - 1 nCUBE avec ORACLE Parallel Server.

- Travail Parallèle
  - Chargement massif des fichiers VSAM
  - Rapports standards
  - Mainframe : 24 heures
  - MPP : 15 minutes

Sybase MPP

- 2 niveaux de parallélisme
  - Sybase VSA (Virtual Server Architecture)
    - Architecture Shared Memory
  - Sybase MPP
    - Navigator (Sybase et ATT GIS)
    - Architecture Shared Nothing
- Sybase MPP
  - Architecture
    - un processus Navigator
    - un nœud = 1 SQLServer
    - Navigator découpe la requête
      et place les sous requêtes plusieurs nœuds
  - Partitionnement entre les noeuds
    - Schéma (1 table sur un nœud)
    - Hachage, Intervalle

IBM DB2

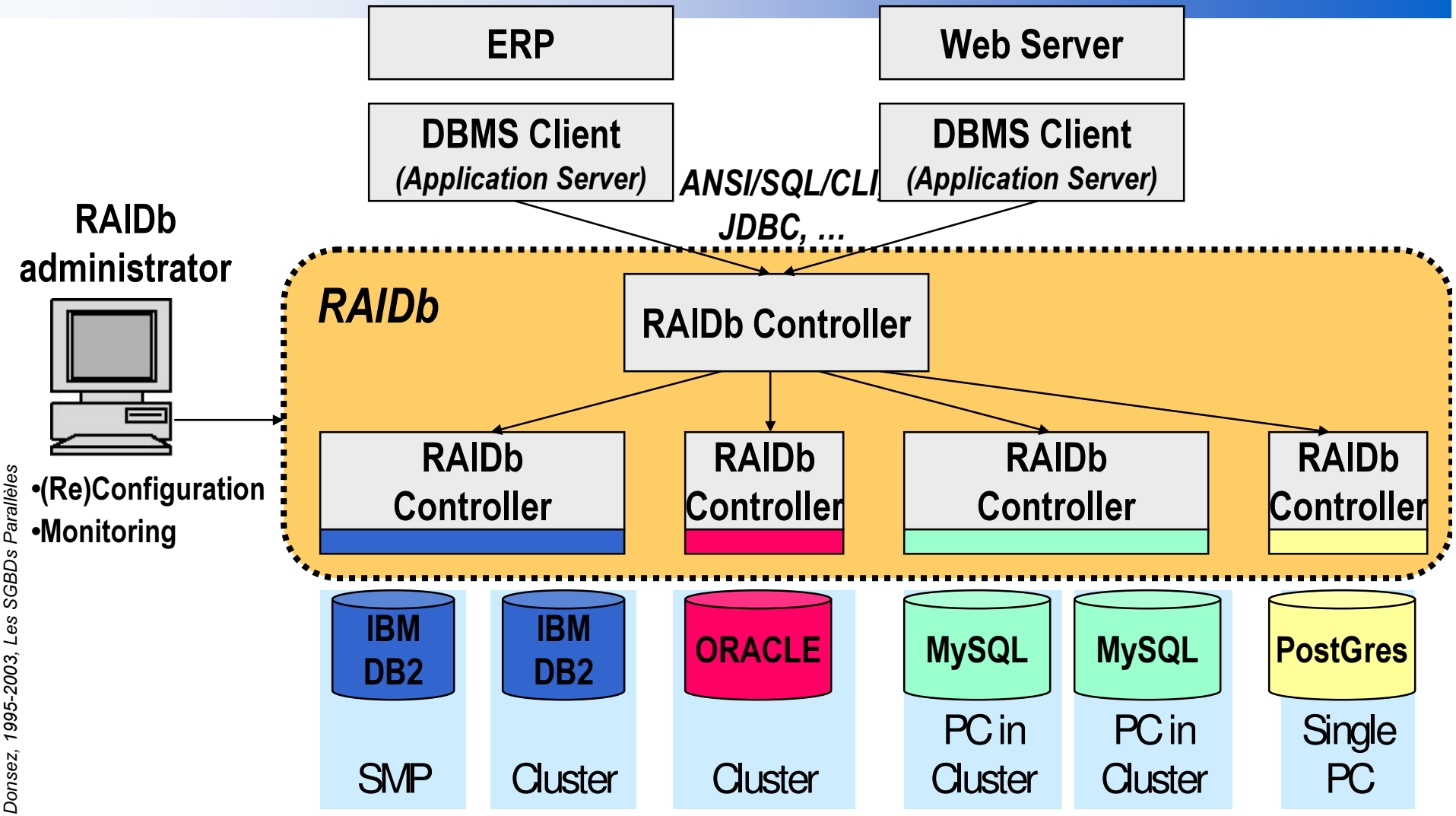
- Architecture Shared Nothing, NUMA, Cluster
  - initialement sur IBM/SP2, maintenant sur nombreux OS
- Requêtes Transactionnelles et Décisionnelles
- Partitionnement
  - Hachage multi-colonnes
  - Réorganisation non bloquante
- Gestion de l ’exécution
  - Processus serveur coordinateur
  - Processus esclaves pour chaque sous-opération
  - Pipeline

RAIDb
Redundant Array of Inexpensible Databases

- Constat
  - Coût des licences de SGBDs commerciaux : Oracle, DB2, …
  - Non coût des SGBDs libres : MySQL, PostGres, …
- Principes
  - Répartir/Réplication la base entre des SGBDs hétérogènes et gratuites sur des grappes de PC (sous Linux)
- RAIDb controller
  - (Re)Configuration of the database
    - Partitionnement verticale/horizontal, Réplication
    - Security
  - Query evaluation/dispatching
  - Monitoring
  - Distributed Join
  - …
- Exemple : C-JDBC (http://www.objectweb.org/c-jdbc)

RAIDb

Redundant Array of Inexpensible Databases



Différents Niveaux de RAIDb

- A faire

SGBD Objet-Relationnel et Parallélisme

- Modèle Objet-Relationnel + Parallélisation
  - OR : de plus en plus populaire
  - Machines parallèles de plus en plus présentes
- Problème très difficile
  - Like trying to mix oil and water
    - DeWitt -http://www.cs.wisc.edu/~dewitt/vldbsum.ps VLDB 97
- Nouvelles techniques de parallélisation nécessaires pour
  - Partitionnement des Types complexes
    - Row, Set, Array, Reference ...
  - Partitionnement des Données deTypes étendus
    - (Series Temporelles, Spatial, SIG, Video, Image …)
  - Optimisation
    - requêtes navigationnelles pour des traitements ensemblistes
  - Règles

Conclusion

- De petites niches pour les MBD dédiés
  - Très haute disponibilités, Très haut débits
- Triomphes de SGBDR parallélisés
  - Oracle, DB2, …

Bibliographie

- Miranda & Ruols, « Client-Serveur : Concepts, moteurs SQL et architectures parallèles », Ed Eyrolles, ISBN 2-212-08816-7, 1994
- Hameurlain, Bazex, Morvan, « Traitement parallèle dans les bases de données relationnelles », Ed Cépadués, ISBN 2-85428-414-3
- Mahdi Abdelguerfi, Kam-Fai Wong, « Parallel Database Techniques », Ed IEEE CS Press, 1998, ISBN 0-8186-8398-8.
- H. Lu, B.-C. Ooi, and K.-L. Tan. « Query Processing in Parallel Relational Database Systems ». IEEE Computer Society Press,1994.
- M. Tamer Özsu, Patrick Valduriez, « Principles of Distributed Database Systems », Prentice-Hall Intl Eds., Second Edition ISBN 0-13-659707-6,1999

Compléments sur Gamma

Didier DONSEZ

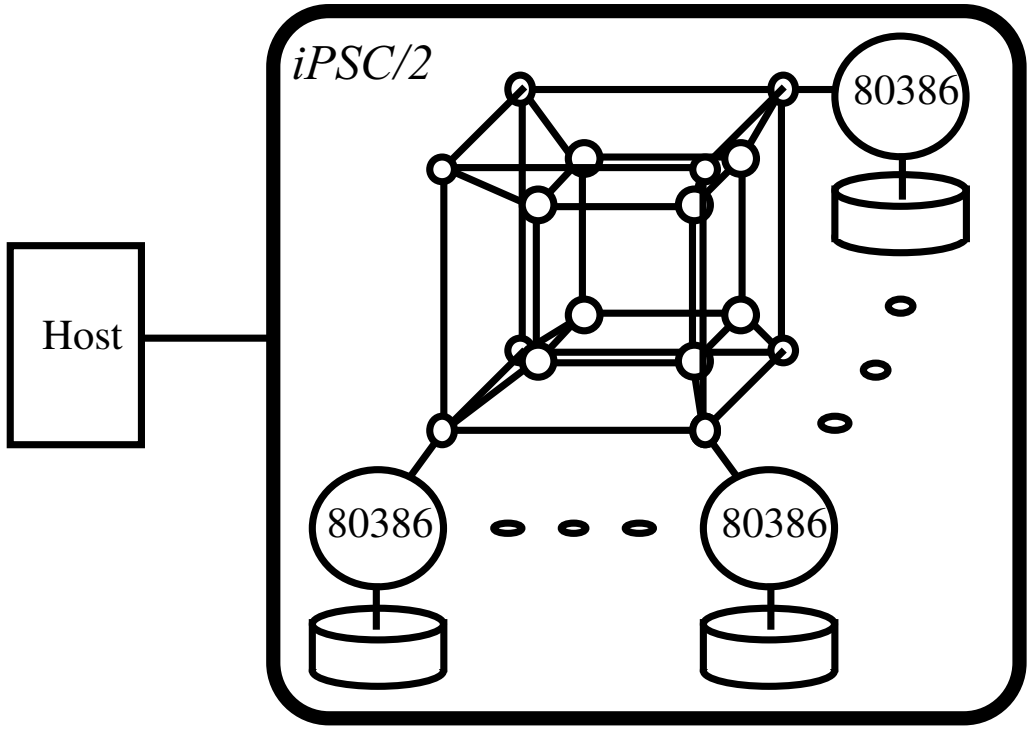
Université Joseph Fourier (Grenoble 1)

IMA – LSR/ADELE

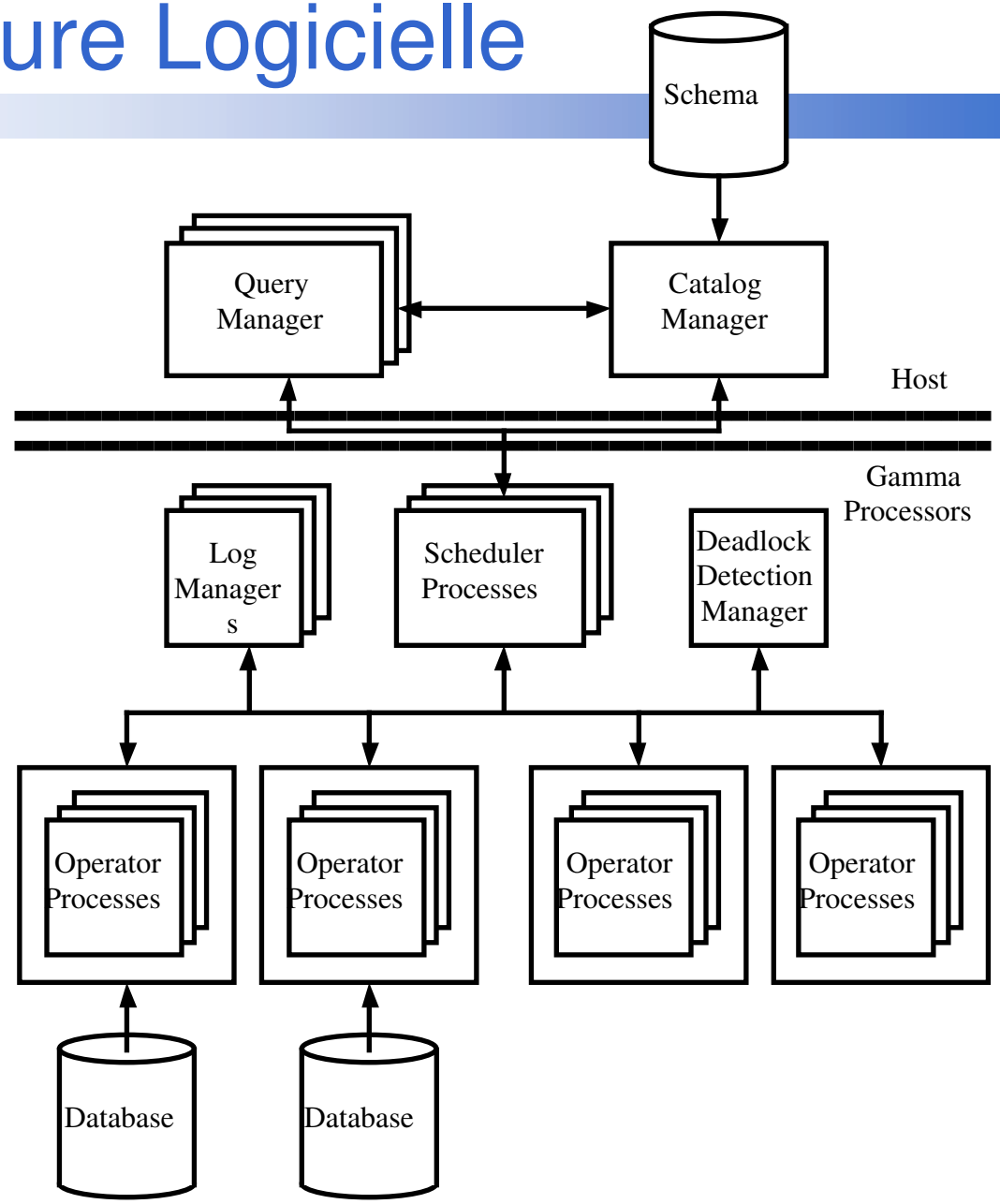
Didier.Donsez@imag.fr, Didier.Donsez@ieee.org

Architecture des machines

- V1 : Réseaux de VAX interconnectés
- V2 : Hypercube Intel iPSC/2

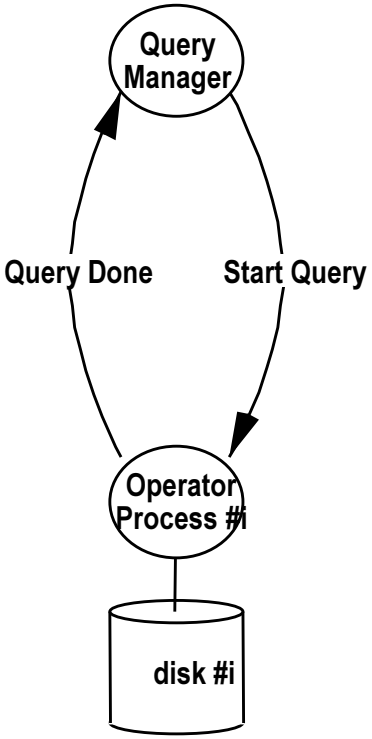


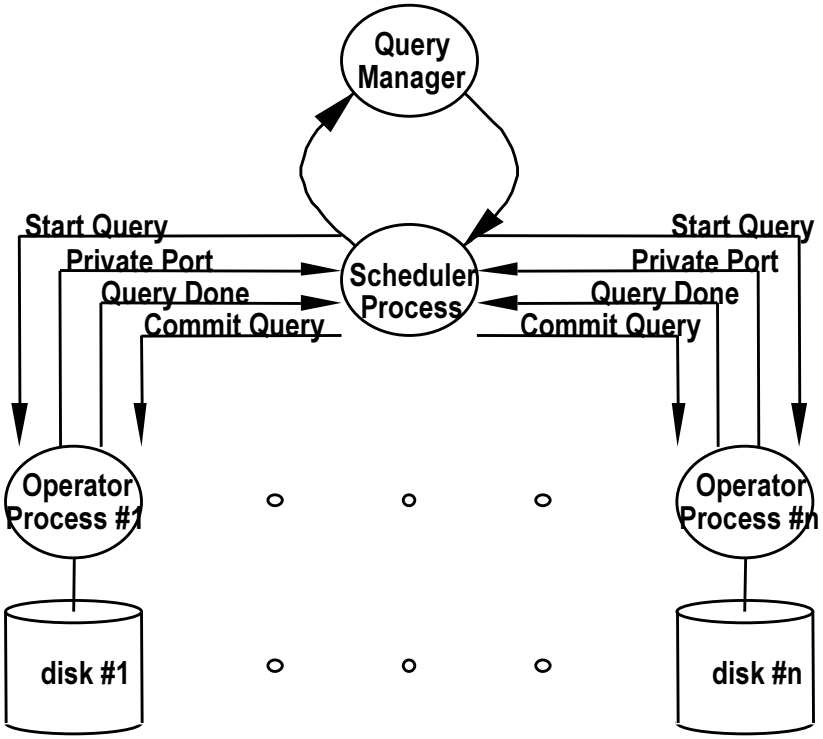
Gamma
Architecture Logicielle



Gamma
Gestion de l 'exécution

- Question mono-site
  - QM envoie la question directement au processeur approprié (déterminé au Runtime)
- Question multi-site
  - QM envoie la question compilée au SP
  - SP active les OP sur les processeurs sélectionnés pour l 'exécution





Gamma
Algorithmes de opérations relationnelles

Restriction

- Utilisation du partitionnement horizontal
- si attribut de placement

Question vers	VALUE SELECT	RANGE SELECT
Round-Robin	Tous les sites	
Hashed	1 seul site	Tous les sites
Range	1 seul site	1 site

- sinon requête de toutes les sites

Jointure basée sur le hachage

- Multiprocesseurs / Grandes mémoires

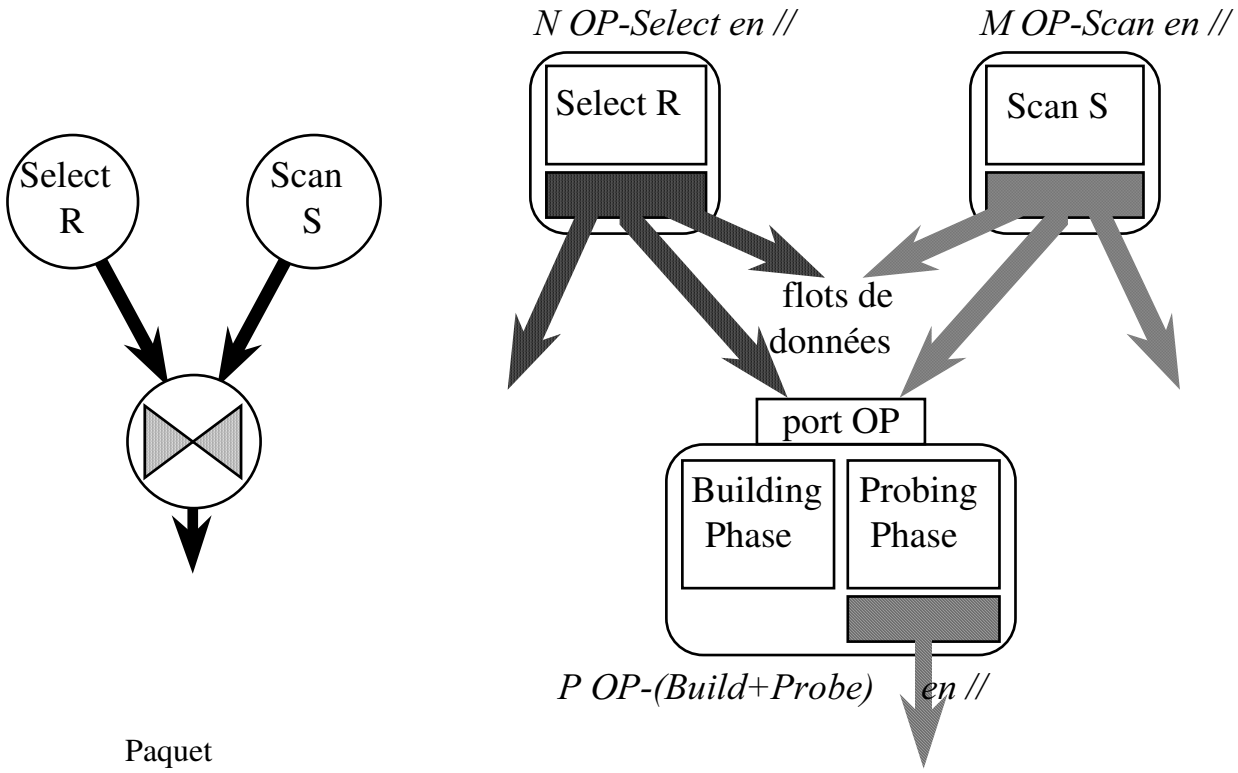
Mise à jour

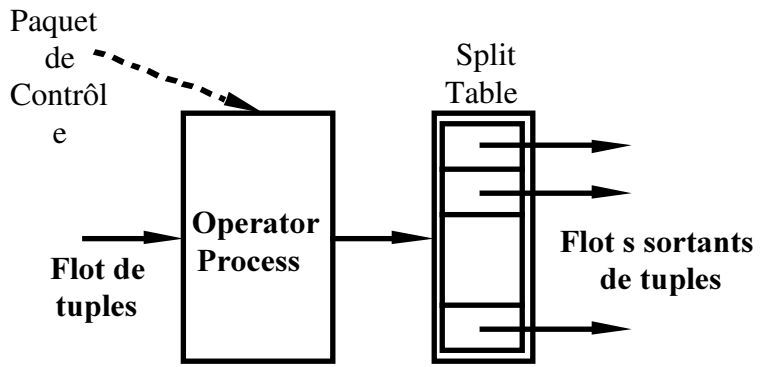
- Techniques classiques

Gamma
Gestion de l'exécution

- OS
  - 3 Processus "Light-Weight"
  - 3 Flot de Données vers 1 port (Stream)
- Contrôle de type DataFlow
  - Pas d'autonomie des sites OP
  - Contrôle intégrale d'1 question par 1 SP

Gamma
Exemple d'exécution contrôlée





Gamma
Contrôle de la Concurrence

- Verrouillage 2-phases
  - 5 modes de Verrouillage
  - Granularité niveau Fichier ou niveau Page
- Détection centralisée des interblocages
  - le Lock Manager (LM) de chaque site envoie périodiquement son graphe d'attentes de verrou à un processus de détection globale (DDM)

Gamma
Reprise sur Panne

- Journalisation Parallèle
  - plusieurs (soit M) Log Managers (LogM) en parallèle
- Principe
  - Chaque Site #i envoie ses info de recovery
    au LogM #i mod M.
  - 1 Numéro de Séquence (LSN) par site.

Gamma
Hybrid-Range Partitioning

- Questions RANGE-SELECT
  - Q1: `select * from R where 320< A < 369`
- Si Placement Hash, Calcul sur tous les sites
  - Surcoût du calcul //
- Si Placement Range Calcul sur 1 (ou plus) site
  - temps de réponse long
- Solution Hybride
  - P un nombre de Processeur
  - tps de réponse(Q1) optimal

Compléments sur Bubba

Didier DONSEZ

Université Joseph Fourier (Grenoble 1)

IMA – LSR/ADELE

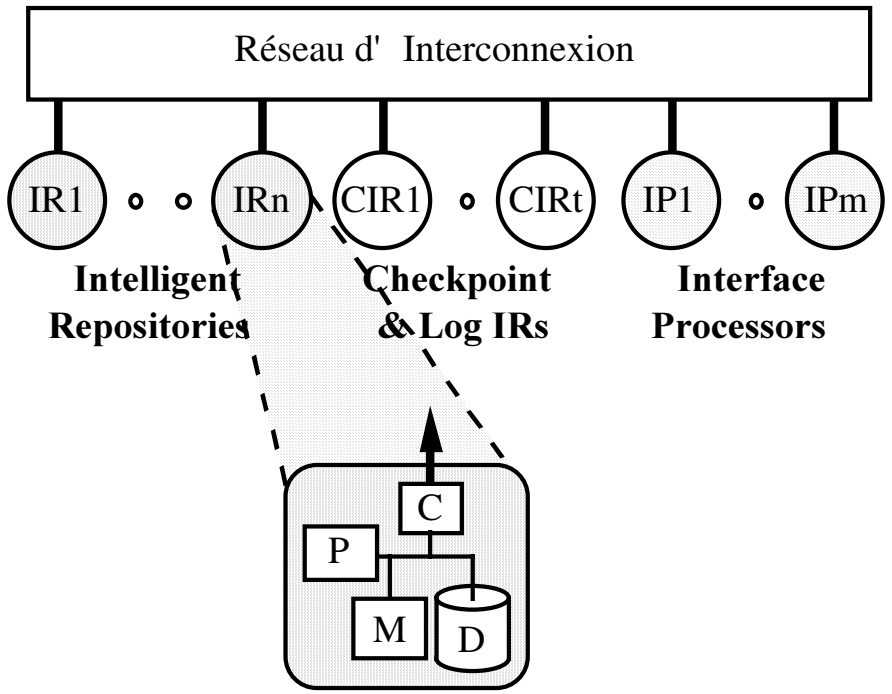
Didier.Donsez@imag.fr, Didier.Donsez@ieee.org

BUBBA

- Projet de machine bases de données parallèles (MCC, Austin)
  - Architecture Matérielle
  - Organisation des Données
  - FAD: Langage BD
  - Contrôle de l'exécution
  - Gestion d'Objets
  - Support OS

Bubba : Architecture matérielle

- Machine BD massivement parallèle
  - jusqu'à 1000 processeurs en 88
- Exploiter le parallélisme
  - Placement des données
  - Parallélisation automatique
  - Contrôle DataFlow de l'exécution
  - Techniques de reprise sur panne
  - Langage BD général
  - Gestion d'objets
  - Support OS



Placement de Données dans Bubba (i)

- Calcul des requêtes
  - Les opérations s'exécutent sur les nœuds où résident les données
- Equilibre de la charge des processeurs
  - implique un « bon » placement des relations entre les processeurs.
  - Equilibrer la charge des nœuds
  - Résident en mémoire / Résident sur disque
  - Hot Set / Cold Set (règle des 5 minutes)
  - 2 copies d'une relation sur des processeurs _
- Heuristique de placement
  - critères:
    - Chaleur de la relation :
      nb d'accès aux objets par UT
    - Température de la rel :
      nb d'accès aux objets/taille par UT
    - Proportion des accès entre
      Direct Copy (x%) / IF-Copy (y%)

Bubba
Placement de Données dans Bubba (ii)

- Algorithme de placement
  - Déterminer le degré de déclusterisation DegDecl
    - le nb de noeuds pour chaque copie de chaque relation
  - Placement de copies en Hot-Set (Memory Resident)
    - copies ordonnées par Température décroissante
    - trouver une place sur DegDecl_copy processeurs avec équilibre des températures cumulées
- Placement de copies en Cold-Set (Disk Resident)
  - copies ordonnées par chaleur décroissante
  - trouver une place sur DegDecl_copy processeurs avec équilibre des chaleurs cumulées
- Expérience : DegDecl optimal
  - DegDecl fixe
  - Direct Copy x% DegDecl
  - IF-Copy y% DegDecl
- Réorganisation (décidée par l'administrateur)

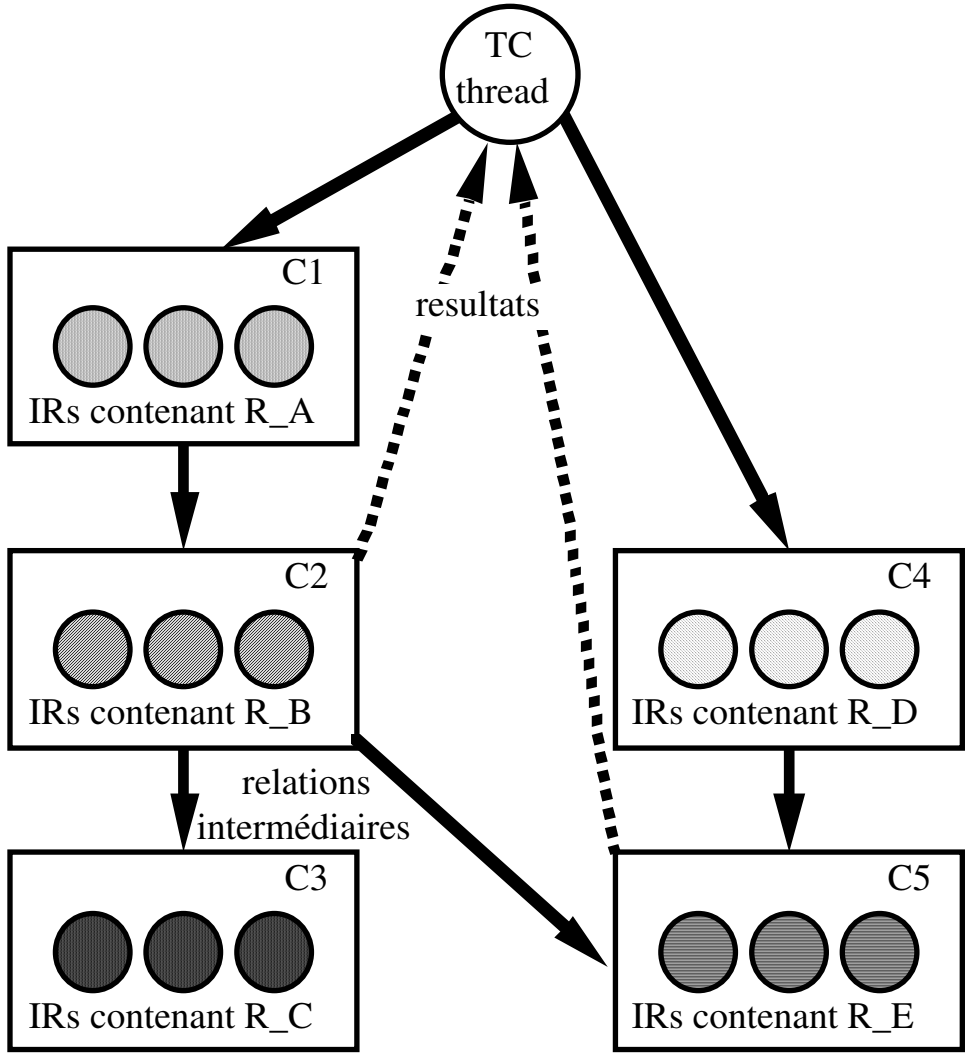
Bubba Langage

FAD:

- 1) Langage à Usage Général
  - les noeuds exécutent l'application et pas seulement les requêtes BD (Embedded SQL)
    => pas de mouvement de données
    entre l'application et le SGBD
- 2) Parallèlisation automatique
  - Programme Monolythique
    => Threads multiples communicantes

Bubba
Contrôle de l'exécution

- Modèle Opératoire
  - Transaction = graphe de composantes
  - Composantes = morceau compilé de prgm FAD
    - ↳ IRs d'une relation source permanente
    - ↳ 1 Thread par IR
  - 1 Thread Coordinatrice
  - Interface avec l'usager
  - Maître de la Validation 2-Phases



Bubba - Chargement et Activation des sous opérations

- Au RunTime, dans une composante, qu'une partie de thread s'exécutera sur les Irs

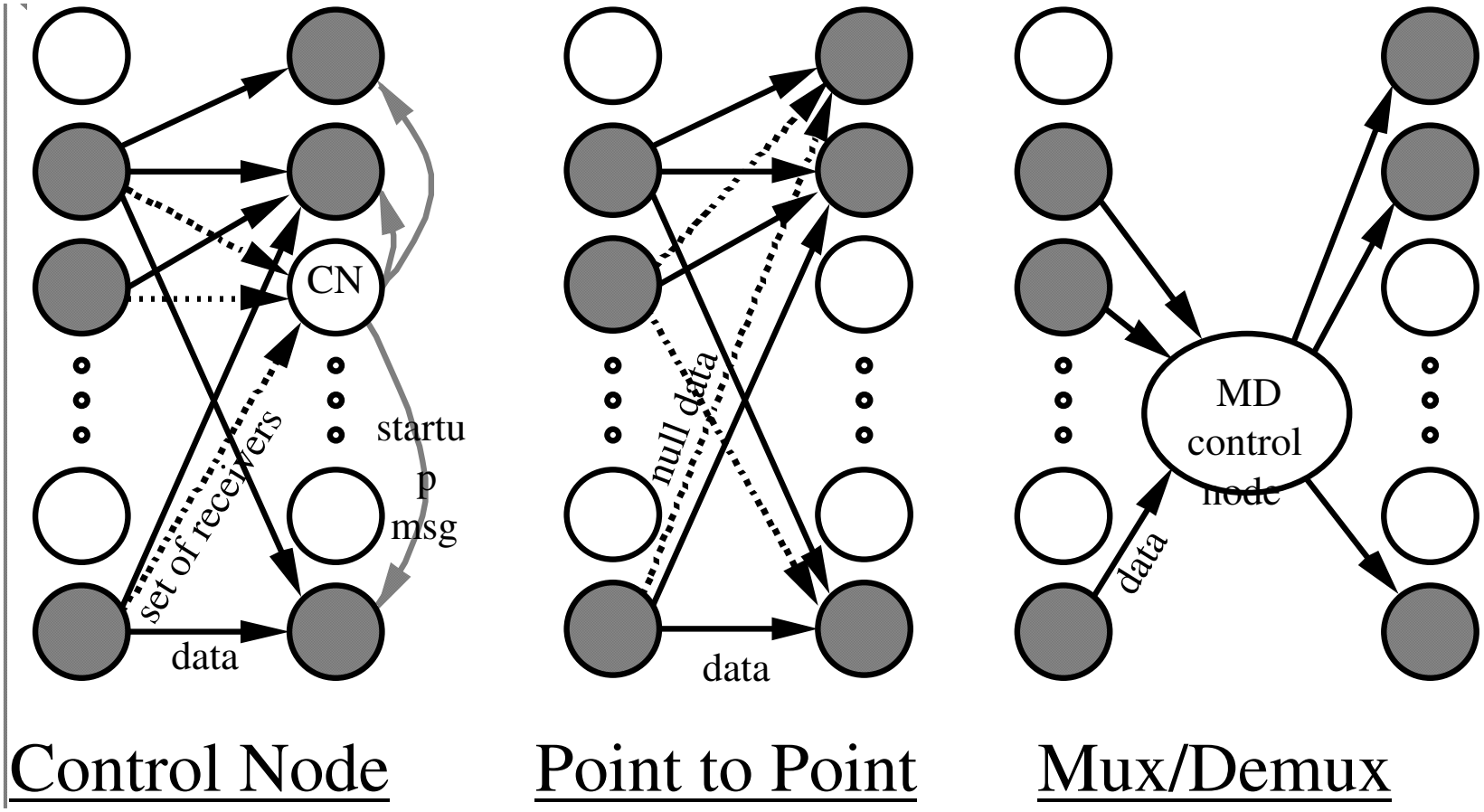
- Changement du code sur les IRs d'une composante
  - PréChargement du Code sur tous les Irs après la compilation
  - Chargement Dynamique sur les IRs où les threads seront actives
- Activation de Threads sur les IRs d'une composante
  - PréActivation des threads sur tous les Irs
    - puis Tuer les threads non activées
  - Activation Dynamique des threads utiles
    - activation sur réception du message (support OS)

Bubba
Contrôle DataFlow de l'exécution

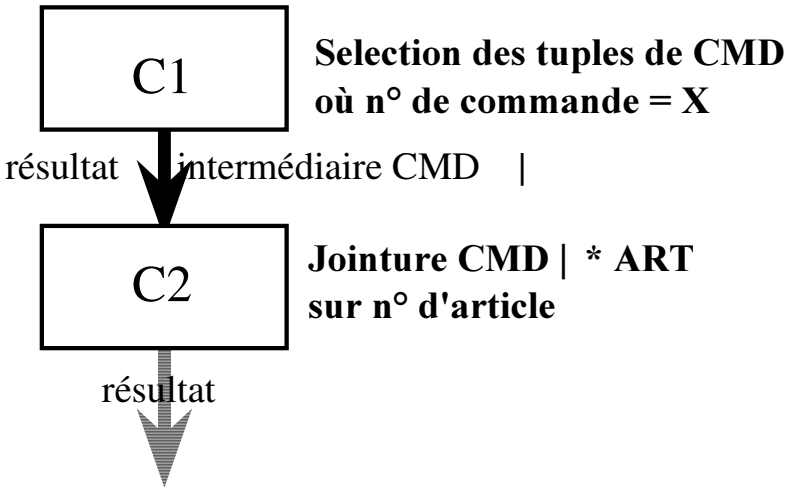
- Transfert de résultats temporaires de la composante S vers D
  - une partie des threads de S actives
  - une partie des threads de D reçoivent des données et sont activées
  - Fin du transfert ↳ démarrage des threads réceptrices
- Pb: Déterminer la fin du transfert

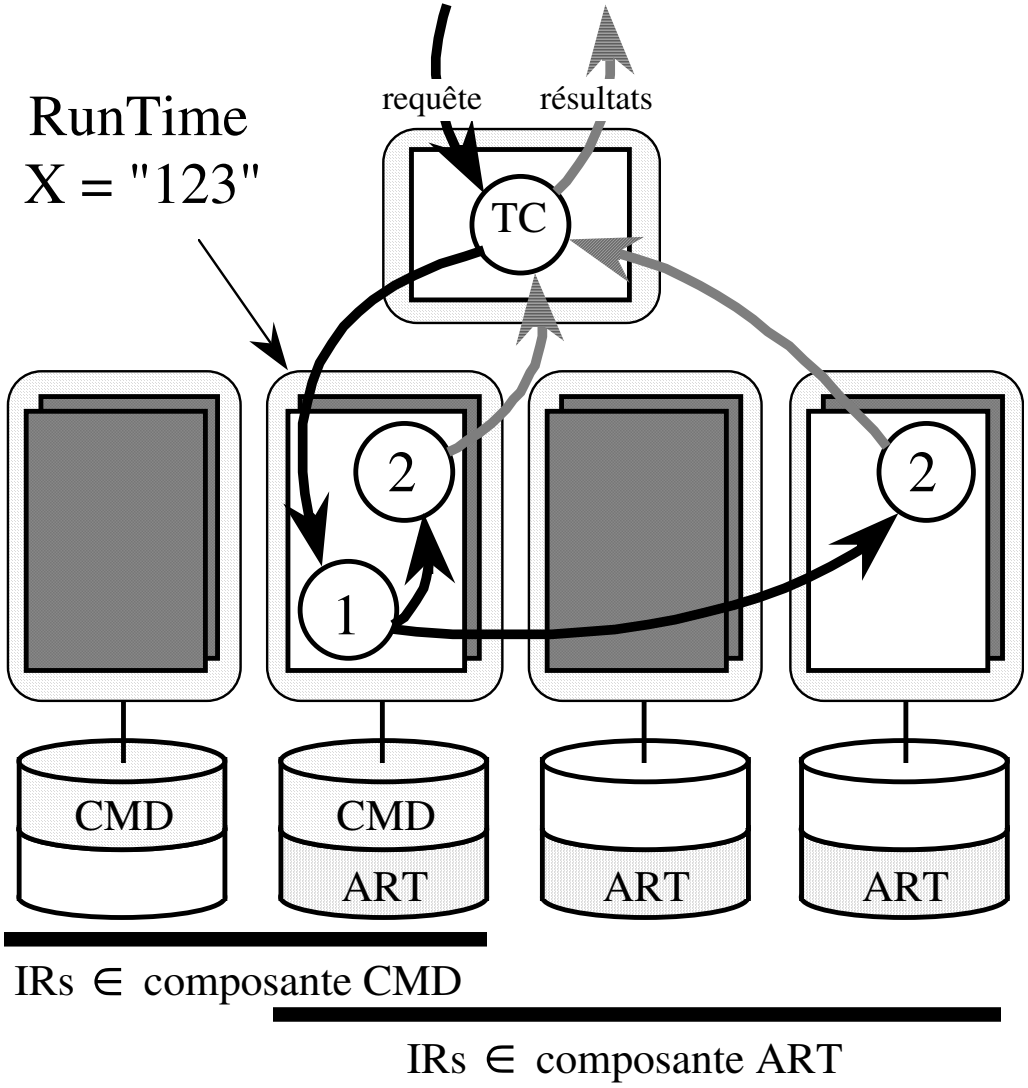
Bubba
Contrôle DataFlow de l'exécution

- Pb: Déterminer la fin du transfert
  - 3 solutions de Transferts N vers M



Bubba
Exemple d'exécution d'une transaction





Bubba
Validation et Abandon

- Quelles sont les threads qui ont participée au calcul ?

hiérarchie de contrôle

- TC envoie le Commit à une Thread de la composante
- cette Thread est maître du commit dans sa composante

Bubba
Gestion d'Objets

- Une seule vue des objets
  - mémoire / disque (one level storage)
  - temporaire / persistent (orthogonalité du type)
- Objets longs (Index vers des blocs)
- Gestion de la Concurrence
- Support OS
  - BOS: OS pour BD
  - Gestion de la Mémoire
  - Gestion de Processus
    - supporter Multi-Threading
    - supporter des messages orphelins
    - ↳ chargement et activation dynamique de threads
  - Messages
    - pas de copie de message d'une thread à l'autre
    - ↳ utilisation de la MMU (Remapping)