Université Paul Sabatier, Toulouse 3

Bases de Données Parallèles

Méthodologies d'optimisation - parallélisation de requêtes décisionnelles

I. *Avantages et limites des systèmes relationnels*
II. *Introduction aux BD parallèles*
III. *Parallélisation des opérateurs relationnels*
IV. *Méthodlogies d'optimisation-parallélisation de req. décisionnelles*
V. *Optimisation des coûts de communication dans les BDP*
VI. *Conception d'un modèle de coût*

A. Hameurlain *<hameur@irit.fr>*

Université Paul Sabatier, Lab. IRIT,
118 Rte de Narbonne
31062 Toulouse - France

I RAPPELS : SYSTEMES RELATIONNELS

1. DEFINITION &TERMINOLOGIE

☛ *MODELE RELATIONNEL [CODD 1970 & 1979]*

- *RELATION ⊂ D1 x D2 x ...x Dn D : Domaine*
- *TUPLE : ELEMENT D'UNE RELATION*
- *ATTRIBUT --> DOMAINE*

SCHEMA RELATIONNEL :

UN SCHEMA DE RELATION :

NOM DE LA RELATION
LISTE DES ATTRIBUTS
[CONTRAINES D'INTEGRITE]

SCHEMA DE RELATION <---> PARTIE INTENTIONNELLE

LES TUPLES DE LA REL. <--> PARTIE EXTENSIONNELLE

☛ *LANGAGE DE DEFINITION DE DONNEES/SCHEMAS*

☛ *LANGAGES RELATIONNELS : SQL, QUEL, QBE,...*

☛ *DEFINITION DES VUES*

☛ *PROGRAMMATION DES APPLICATIONS : SQL +{/C/C++/ADA/Java/,..*

2. AVANTAGES DES SYSTEMES RELATIONNELS

☛ *SIMPLICITE DES CONCEPTS ET DU SCHEMA*

☛ *INDEPENDANCE DONNEE-PROGRAMME*

- *INDEPENDANCE LOGIQUE*
- *INDEPENDANCE PHYSIQUE*

☛ *LANGAGES RELATIONNELS : DECLARATIFS*

- *SPECIFIER LES DONNEES RECHERCHEES*
- *SANS DONNER LE CHEMIN D'ACCES*

➥ *OPTIMISATION DE TRAITEMENT DES REQUETES*

⇨ *STRATEGIES D'OPTIMISATION*
⇨ *METHODES D'ACCES*

☛ *CONCEPT DES VUES*

- *DEFINITION DES RELATIONS DEDUITES(VIRTUELLES)*

- *AVANATAGES :*
  - ➥ *SECURITE ET CONFIDENTIALITE*
  - ➥ *AMELIORE L'INDEPENDANCE LOGIQUE*
  - ➥ *SIMPLICITE D'EXPLOITATION*

- *AUGMENTER LA PUISSANCE DES LANGAGES REL.*

☛ *MANIPULATION ENSEMBLISTE DES DONNEES*

➥ *EXPLOITATION DU PARALLELISME*
⇨ *Haute Performance*

☛ *Objectif :*
- *Minimiser le temps de réponse TR d'une requete : $T_{cpu} + T_{E/S} + T_{com}$*
- *Maximser l'exploitation des ressources : <CPU, Disque , Réseau, Mémoire>*

☛ *gérer de manière efficace et économique :*
- *l'allocation des ressources*
- *les coûts de communication : données, contrôle*

II. Introduction aux bases de données parallèles

1. Motivation

☛ *Langages relationnels : déclaratifs*

Modèle d'exécution monoprocesseur ⇨ Modèle d'exécution multiprocesseur
⇨ *aucun impact sur les programmes d'application*
⇨ *Meilleure utilisation des compétences des programmeurs*

☛ *Nature du parallélisme : <Explicite, Implicite >*

⇨ *Les processus : Optimisation - Parallélisation - Allocation de ressources sont TRANSPARENTS aux programmeurs*

2. Objectifs des BD parallèles

☛ *Meilleur Coût / Performance par rapport à la solution gros systèmes (DPS8 / GCOS, IBM 30390 / VMS, ...)*

☛ *Haute performance :*

- *Minimiser le temps de réponse*
- *Maximiser l'utilisation des ressources du système //*

☛ *" Scalability " : Dimensionnement évolutif*

- *Ajout de nouvelles ressources (CPU, Disque, Mémoire, Réseau)*
- *Ajout de nouveaux utilisateurs*

⇨ *Sans perte de performance*

☛ *Disponibilité des données (en cas de pb sur des noeuds)*

3. Modèles d'Architectures Parallèles

☛ Classes d'Architectures Parallèles : Mémoire et Processeurs

- **Modèle d'exécution SIMD** (Single Instruction Multiple Data Stream)
- **Modèle d'exécution MIMD** (Multiple Instruction Multiple Data Stream)

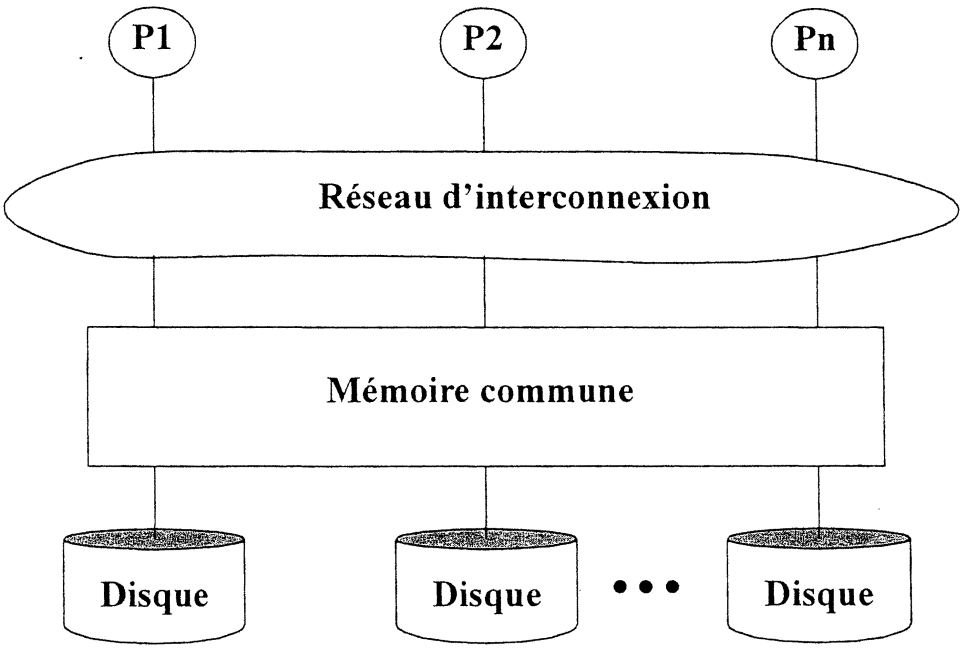
En fonction de l'organisation des données en mémoire :

⇨ *Multiprocesseur à mémoire commune (Couplage fort)*

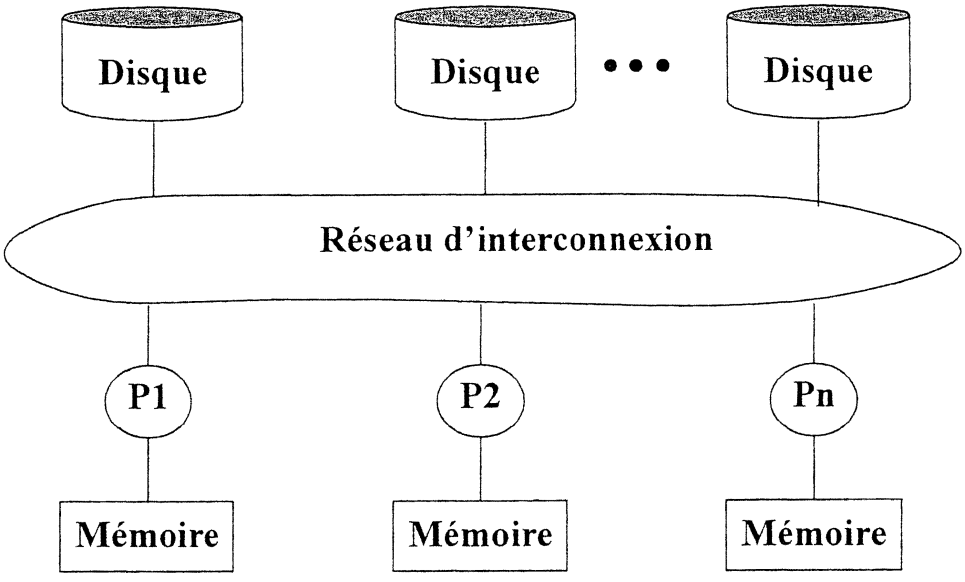
⇨ *Multiprocesseur à mémoire distribuée (Couplage faible)*

☛ Contexte Bases de Données : Mémoire, Processeurs et Disques

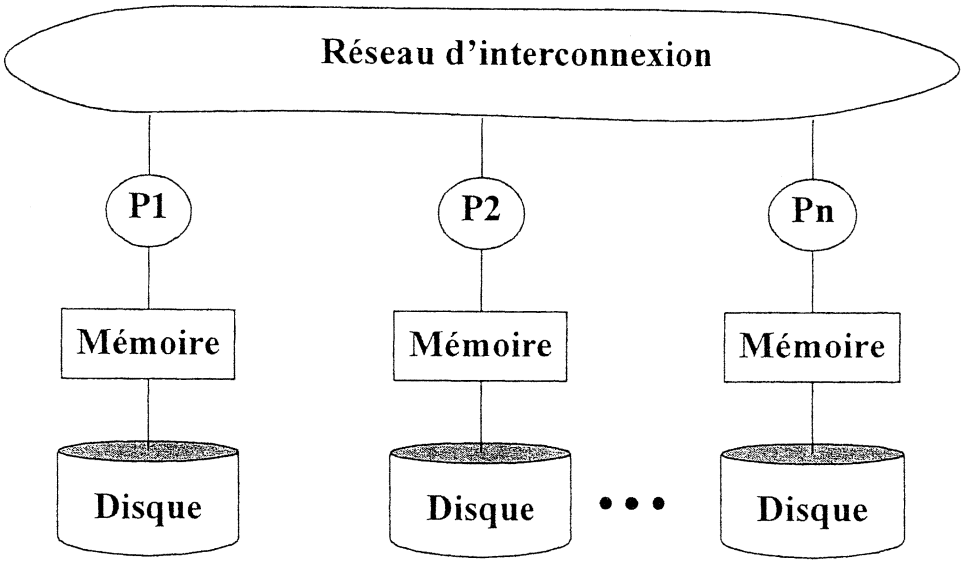
- *SGBD Parallèles basés sur des Arch. Mult. à Mém. Commune*
- *SGBD Parallèles basés sur des Arch. Mult. à Disques partagés*
- *SGBD Parallèles basés sur des Arch. Mult. à Mém. Dist.*



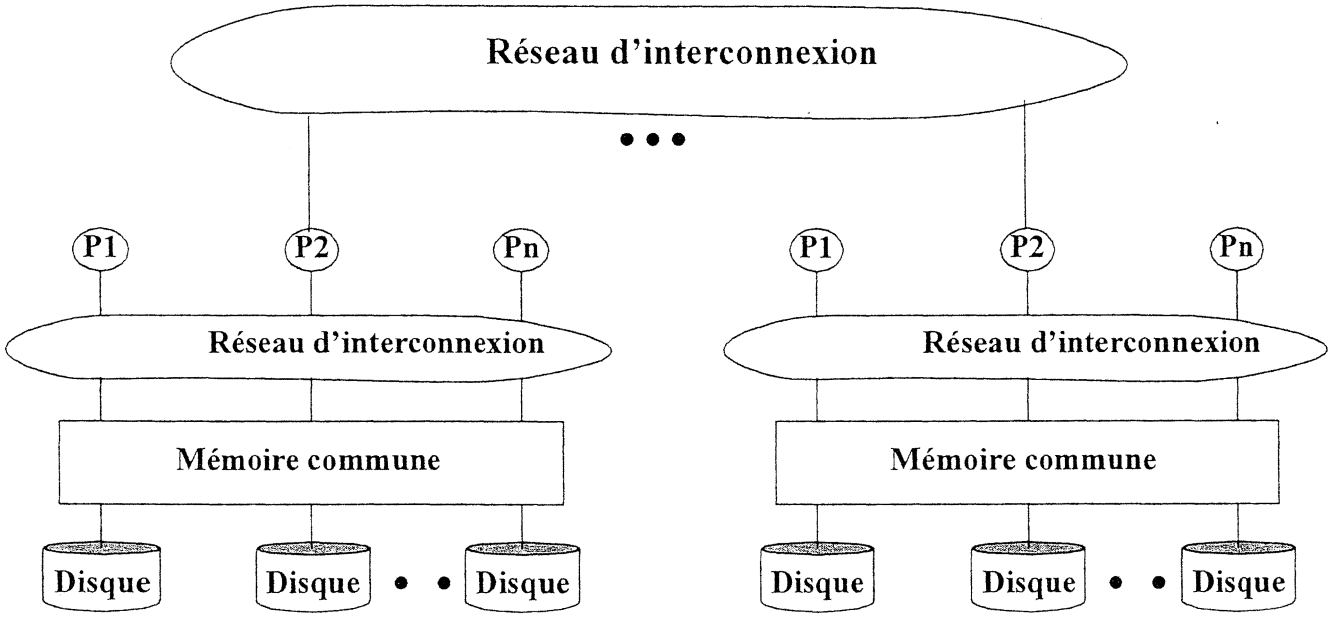
Architecture à Mémoire Commune



Architecture à disques partagés



Architecture à Mémoire Distribuée



Architecture Hybride

(i) Avantages et Inconvénients des Architectures (voir en TD)

☛ *à Mémoire Distribuée*

☛ *à Mémoire Commune*

(ii) Principaux produits et prototypes

☛ *à mémoire distribuée*

- *Teradata : DBC/1012 Database computer*
- *Tandem NonStop SQL*
- *Serveur Parallèle ORACLE (>=V7, 1994)*
- *DB2 (SP2 et SPQS (IBM))*
- *DSA/PDQ Informix (1994)*
- *Gamma : Dewitt et al. USA [Dew 90]*
- *Bubba : Boral et al. USA [Bor 90]*
- *Oracle /Supernode : PARSYS*
- *Prisma/DB [Ape 92]: Phillips et univ. Thewente (NL)*

☛ *à mémoire commune*

- *XPRS : Stonebraker et al., Univ. Berkley [Sto 88]*
- *DBS3 /KSR1 [Cas 94]: Inria & Bull (Mém. Log. Partagée)*
- *Volcano (USA) [Gra 90]*

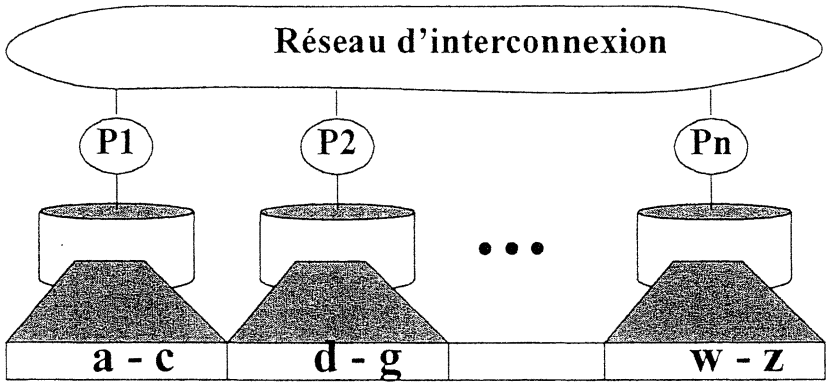
☛ *Disques Partagés :IMS Data Sharing (IBM),*
ORACLE V7 Parallel Server

4. Approches et méthodes de répartition de données [Dewitt 92]

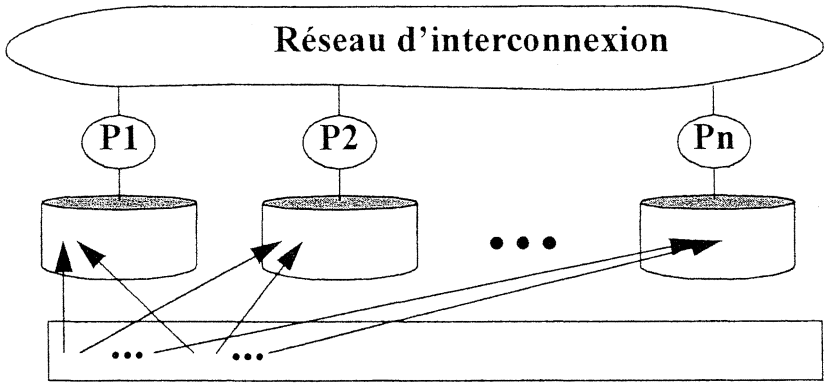
☛ *Approches :*

- *Partitionnement total (Full declustering)*
- *Partitionnement partiel (Partial declustering)*

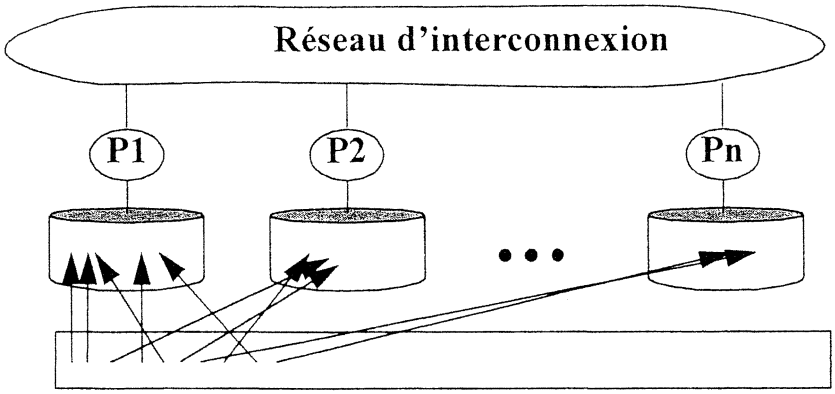
☛ *Méthodes :*



Range Partitioning



Round-Robin



Hashing

5. Niveaux, formes et types de parallélisme

☛ *niveaux de parallélisme : intra-requête, inter-requête.*

☛ *formes du parallélisme : intra-opération, inter-opération.*

☛ *types du parallélisme :*

Pipeline *indépendant (partitionné)*

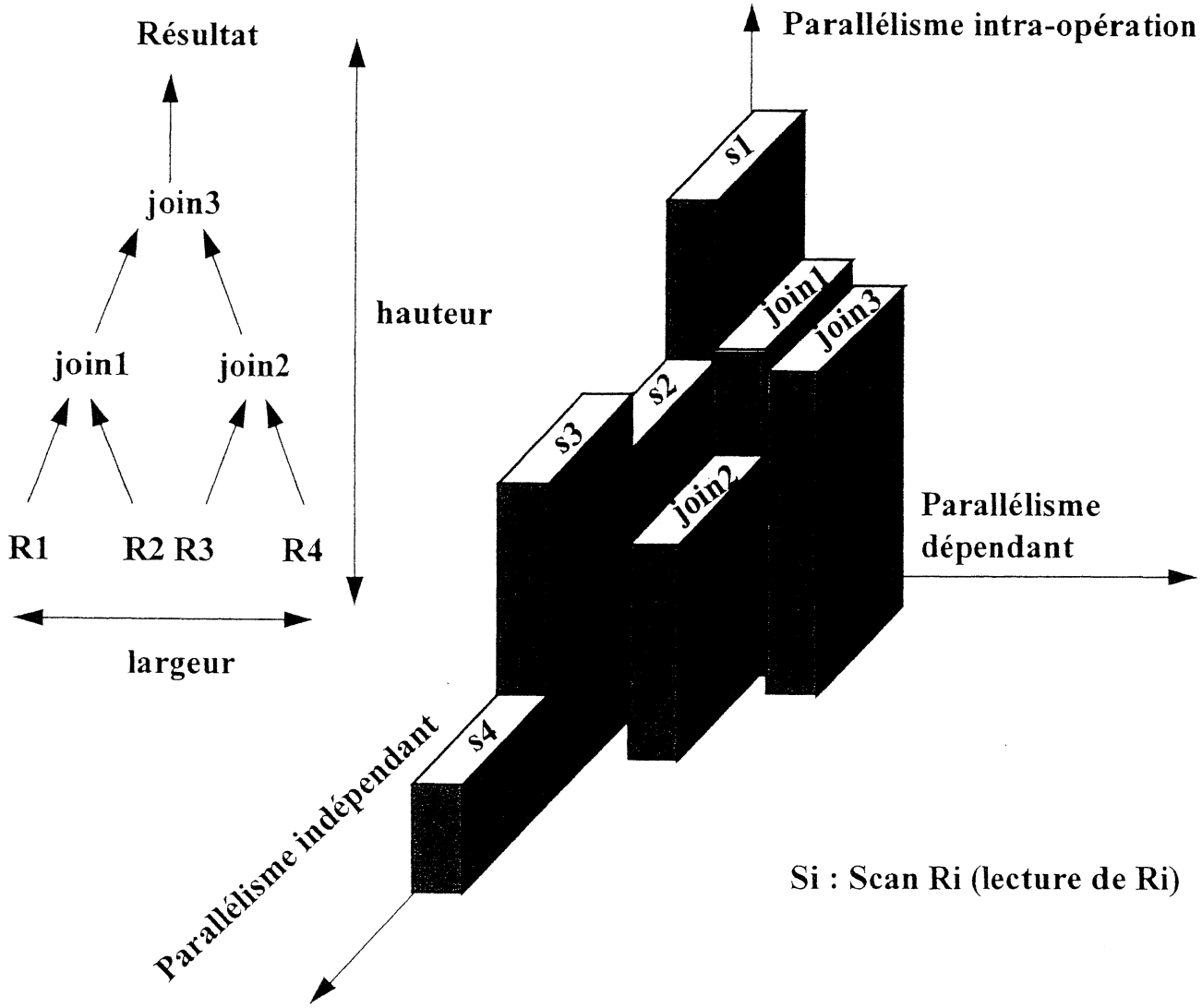
(a) Pipeline : Chaque fois que 2 opérations échangent les données

+ *Evite de stocker des résultats intermédiaires*
— *Efficacité limitée : Chaîne de pipeline, fonctions,..*

(b) Parallélisme indépendant

- *Distribution des données*
- *Eclater l'opération en sous-opérations*

+ *Sans communication et contrainte de synchronisation*
— *Répartition des données (équilibrer la charge de travail)*



Références :

A. Hameurlain, P. Bazex, F. Morvan; "Traitement parallèle dans les BD relationnelles", Cépaduès-Editions, Oct. 1996, 314 pages.

G Gardarin, "Bases de données : objet & relationnel"; Ed. Eyrolles, Janv. 1999, 788 pages

M. Tamer Ozsu, P. Valduriez, "Principles of Distributed Database Systems", Second Edition, Prentice Hall Intl. Editions (http://www.prenhall.com), 1999, 666 pages.

III. Parallélisation des opérateurs relationnels

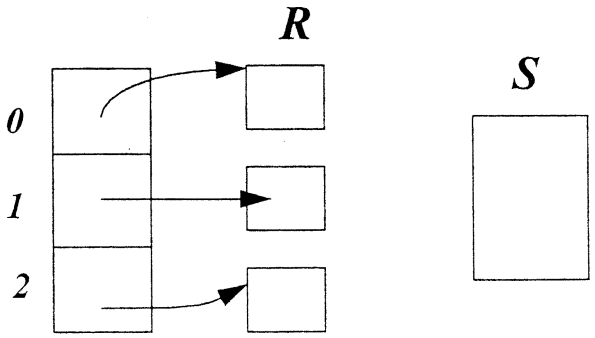
1. Algorithmes de Jointure dans un contexte centralisé

☛ *Jointure par Produit Cartésien : R ⋈ S*

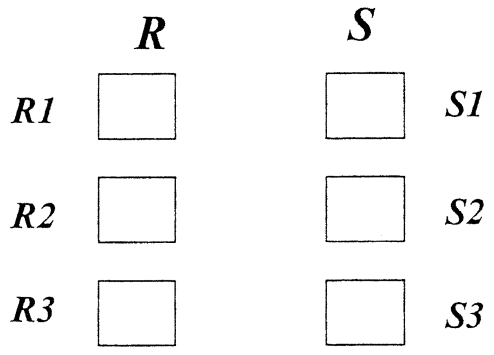
Chaque tuple de R est comparé avec chaque tuple de S

☛ *Jointure par hachage*

- *Simple : Build (R) {table de hachage} ; Probe (S)*



- *Hybride : Fragmenter (découper) R avec une fonct. F;*
  Fragmenter (découper) S avec une fonct. F;
  Pour i=1 à n faire Ri ⋈ Si (hachage simple)



☛ *Jointure par Tri-Fusion*

(1) Trier R; (2) Trier S;
(3) Fusion;

2. Algorithmes de Jointures parallèles [Val 84] [Sch 89]

☛ *Jointure par produit cartésien [Val 84]*

R est répartie sur les disques des processeurs

S → *en Mémoire Centrale si mémoire commune*

S → *dupliquée sur les noeuds si mémoire distribuée*

☛ *Jointure parallèle par hachage [Sch 89]*

R et S réparties sur les disques des processeurs avec une fonction F1

- *Simple*

  build (Ri) avec une fonction F2

- *Hybride*

  Ri et Si fragmentées (découpées) avec une fonct. F2
  build (Rij) avec une fonction F3

☛ *Jointure par Tri-Fusion [Val 84][Sch 89]*

(1) Répartir (R, S, d)	*(1) Répartir R avec une fonction de F1*
(2) Tri parallèle (log d)	*(2) Répartir S avec une fonction de F1*
(3) Fusion centralisée	*(3) Tri-Fusion en parallèle*

Répartition d'une relation sur les disques des processeurs (F1)

$\neq$

Construction des paquets/table hachage de données (F2/F3)

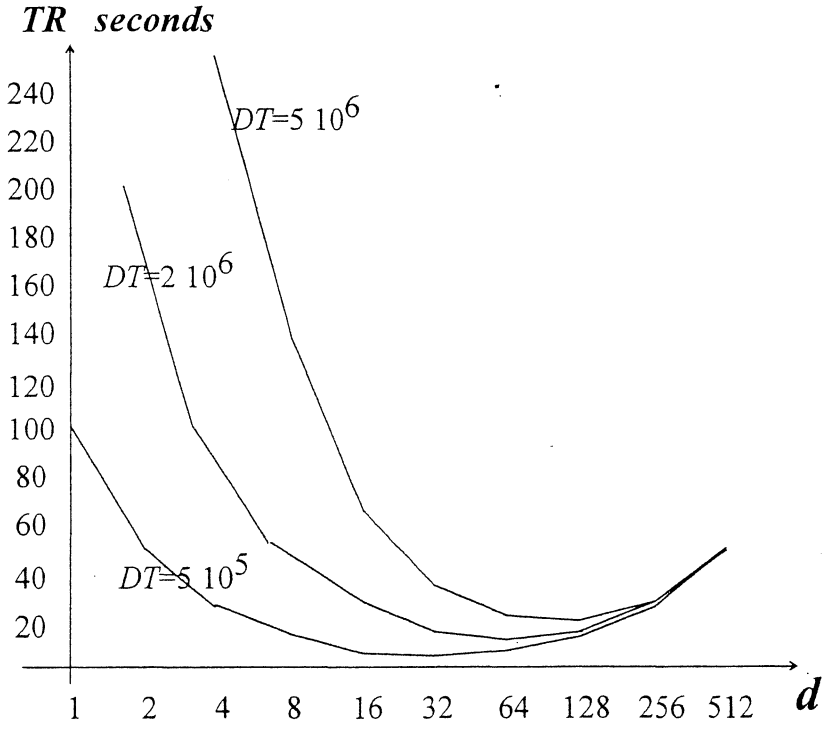
3. Degré de parallélisme intra-opération : Méthode de détermination du nombre de processeurs pour une exécution parallèle

☛ *Motivations*

$$TR\ (d)_Q = A.d + B/d$$

R = 10^6 tuples; msg = 1 ms; trf = 5 µs; t = 0,2 ms; p (=h)=100

t : *temps pour produire un tuple,*
trf : *temps pour transférer un tuple,*
msg : *temps pour envoyer un message*



Temps de réponse en fonction du nombre de processeurs

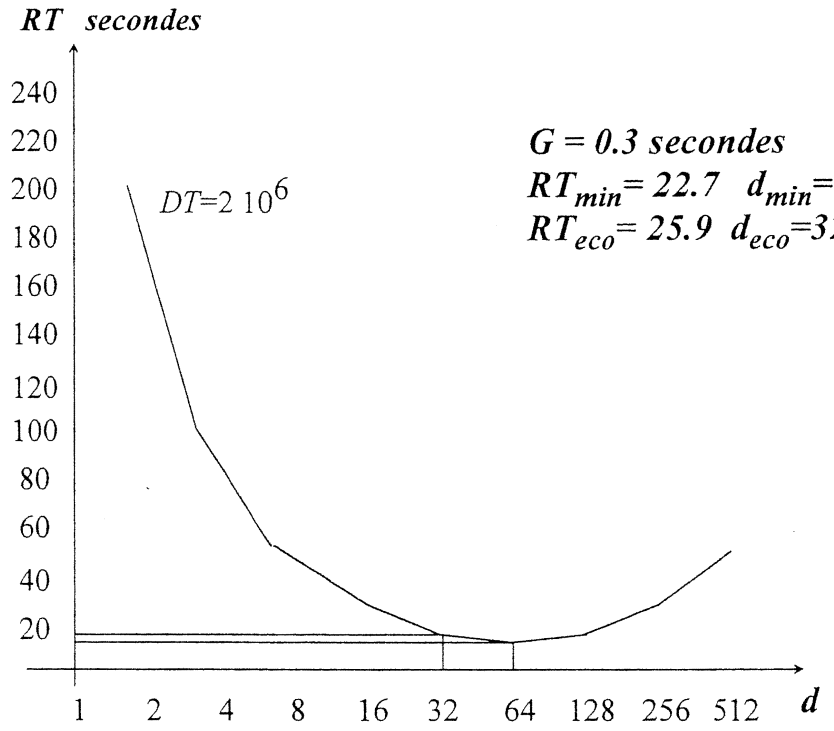
Si DT=5 10^5 tuples alors le minimum RT est atteint pour d=32
Si DT=2 10^6 tuples alors le minimum RT est atteint pour d=64
Si DT=5 10^6 tuples alors le minimum RT est atteint pour d=128

➥ *Méthode d'estimation de DT*

DT: Le nombre de tuples déduits par : une fermeture transitive ou une fermeture externe

☛ Détermination du nombre de processeurs économique

$$\Delta t = |RT_{min} - RT_{acc}| \Longrightarrow \Delta d = d_{min} - d_{eco}$$



G: seuil de rentabilité (break-even point)

- *Définition de D_{eco}:*
  - *(i) $TR(d_{eco}) - TR(d_{eco}+1) < G$;*
  - *(ii) $TR(d_{eco}-1) - TR(d_{eco}) >= G$.*
- *Méthode de recherche :*
  - *Approache dichotomique dans $[1, d_{max}]$*
  - *d_{max} : le nombre total de la machine cible*

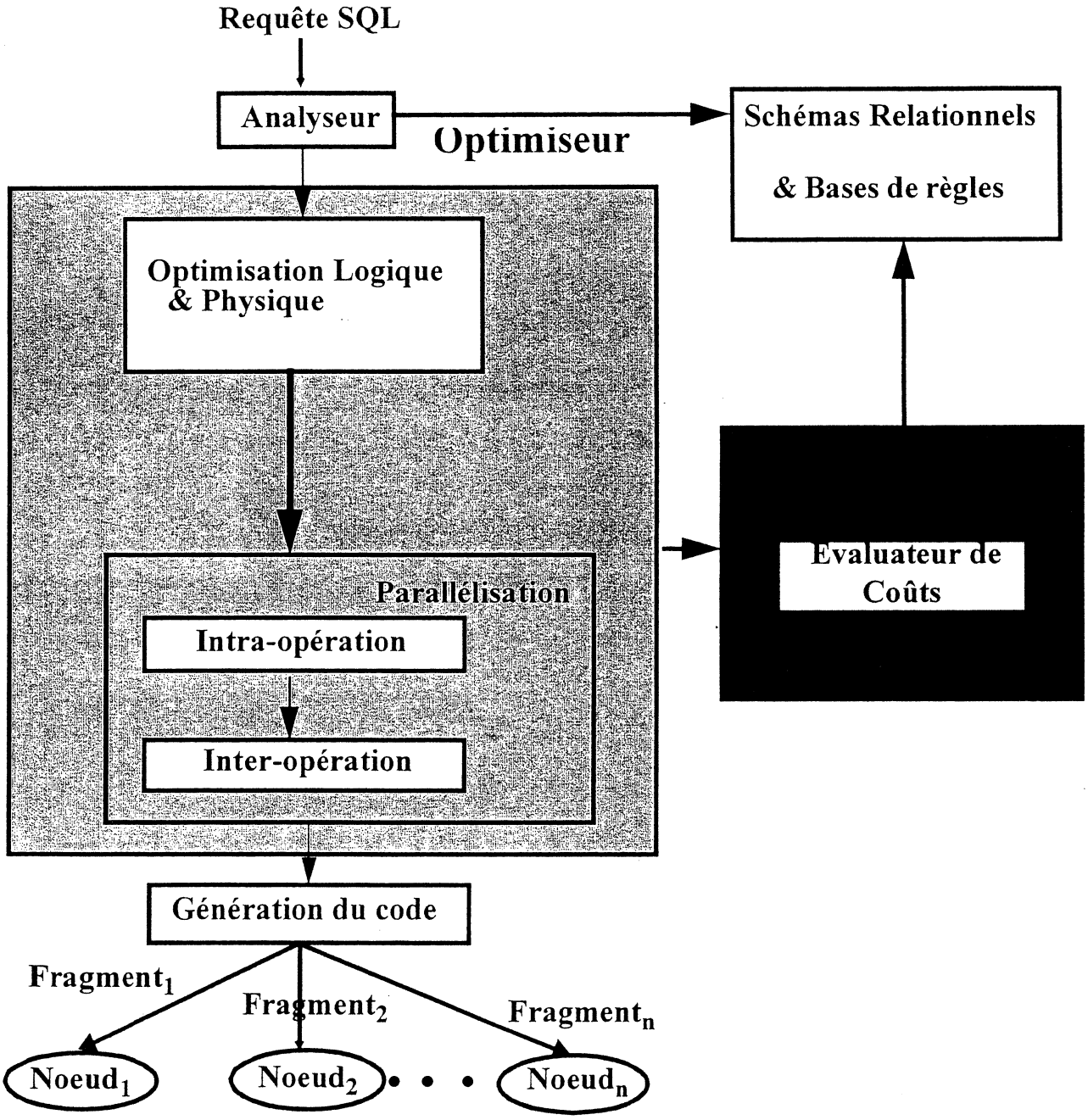
M = milieu de $[1, d_{max}]$
Si $TR(M-1, R, S, p) - TR(M, R, S, P) < G$ ➨ $[1, M-1]$
Si $TR(M-1, R, S, p) - TR(M, R, S, P) >= G$ ➨ $[M, d_{max}]$

P : nombre de processeurs de l'opération successeur (communi.)

IV. Méthodologies d'optimisation - parallélisation de requêtes décisionnelles

1. Introduction

☛ *Architecture d'un compilateur de requêtes SQL*



☛ *Optimiseur - Paralléliseur - Alloc. Res.*

☛ *Problème de l'optimisation de requêtes [Gan 92]*

$q \in Q$, $p \in PE$ (*Plans d'exécution*), *Coût* $(q)_p$:

⇨ *trouver le PE calculant q tel que le coût (q) est minimal*

♦ *Coût (q) = Min (Temps Rep.) & Max (Capacité du Système)*

☛ *Optimiseur* $= < S, E, C >$

- *S : Stratégies de recherche = {Optim. Physique, Parallélisation}*
- *E : Espaces de recherche ={ Arbres linéaires et arbres bushy}*
- *C : Evaluateur de coûts =< Environnement du système parallèle, Métriques >*

⇨ *Stratégies de Recherche :*

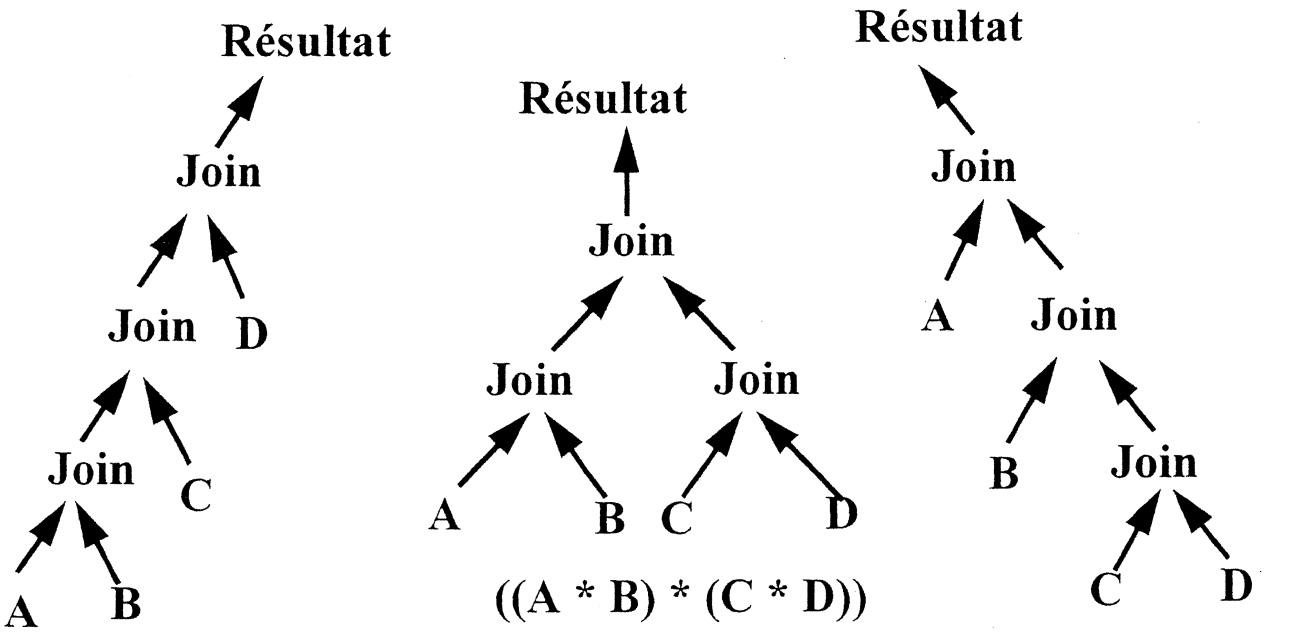
♦ *<u>Optimisation Physique</u> (monoprocesseur):*
- Choix du meilleur Alg. de jointure &
- Génération d'un ordre efficace des opérateurs de jointure
[Swa 88, Ioa 89, Lan 91,...]

- *Stratégies Enumératives:*
  - Breadth-First (System R [Sel79]),
  - Depth-First (AH, AH)*

- *Stratégies Aléatoires:*
  - Iterative Improvement II
  - Simulated Annealing SA

♦ *<u>Parallélisation</u> (Multiprocesseur) :*
Scheduling (// Extraction) & Allocation de Ressources

⇨ *Espaces de recherche :*



2. Stratégies de parallélisation des opérations d'une requête

☛ *Objectif :*

Engendrer un plan d'exécution PE parallèle (~ Oprimal)

☛ *Approches : < mono-phase et à deux phases>*

Φ_1 : *Génération d'un PE sans considerer les ressources*

Φ_2 : *Allocation des ressources (proche de l'optimale)*

⇨ *Approche mono-phase :* $\{\Phi_1 * \Phi_2\}^*$

⇨ *Approche à deux phases :* $\{\Phi_1 ; \Phi_2\}$

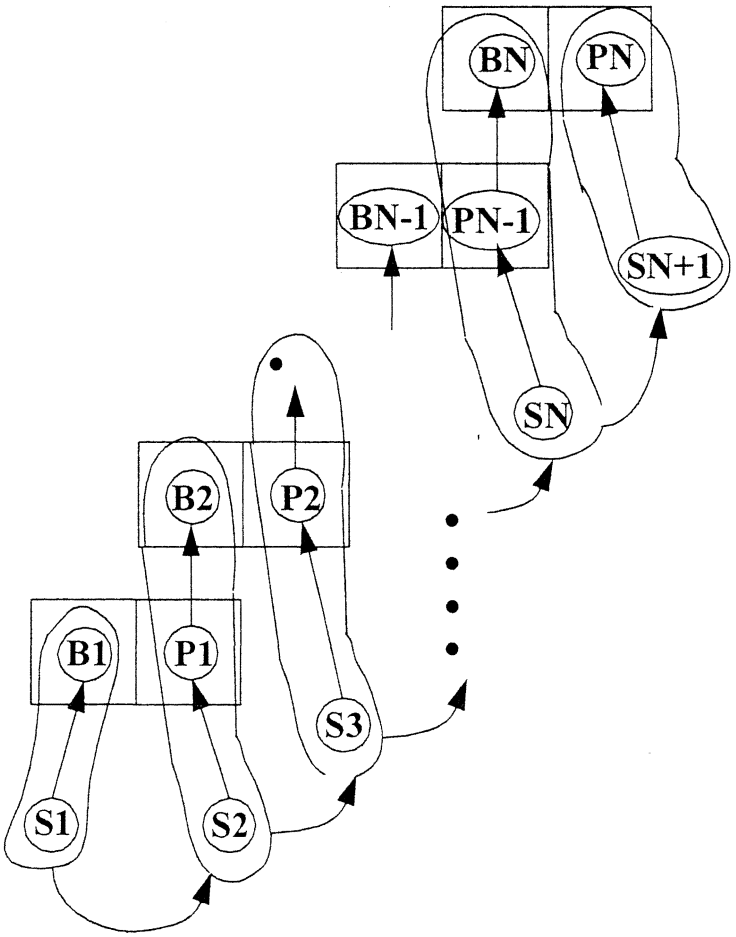
☛ *Approche mono-phase :* $\{\Phi_1 * \Phi_2\}^*$

➥ *Extension des Stratégies de recherche à l'optimisation de requêtes parallèles.*

- *Extension du Modèle de Coûts = Capture tous les aspects du < Parallélisme & Alloc. Res. >*

➥ *Allocation de ressource : <Memory>*

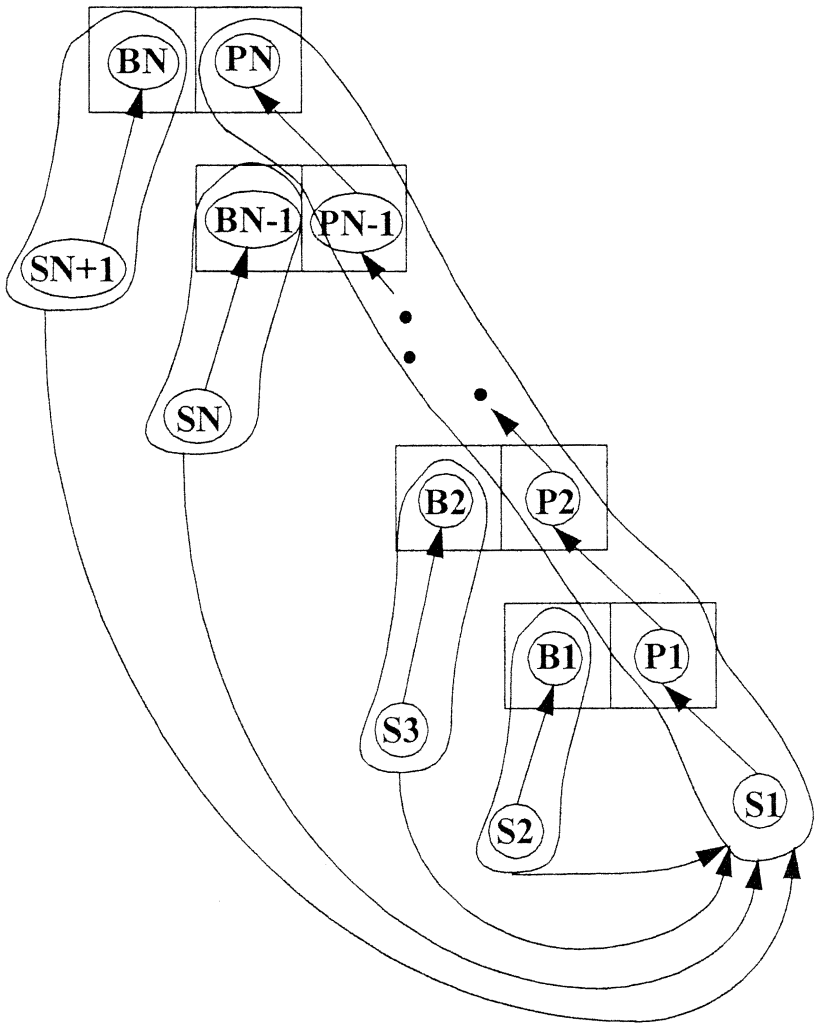
3. Approches mono-phase (Gamma [SCH90] & DBS3 [ZIA 93])



SEQ
PIPE Scan S1 - Build J1 ENDPIPE;
PIPE Scan S2 - Probe J1 - Build J2 ENDPIPE;
PIPE Scan S3 - Probe J2 - Build J3 ENDPIPE;
• • • •
PIPE Scan SN - Probe JN-1 - Build JN ENDPIPE;
PIPE Scan SN+1 - Probe JN ENDPIPE;
ENSEQ

SCAN (R) = Sélection (R) + Répartition (R)

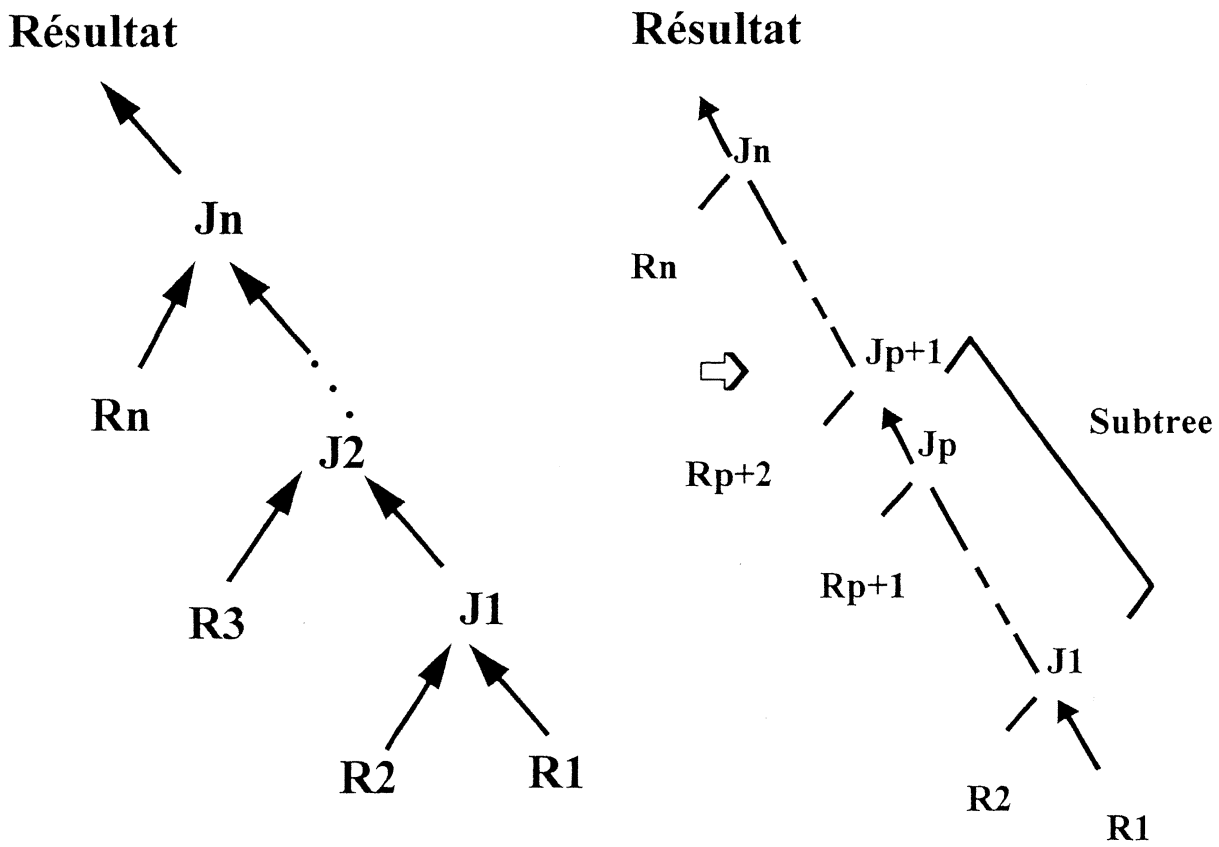
Arbre linéaire gauche



SEQ
PAR
PIPE Scan S2 - Build J1 ENDPIPE
PIPE Scan S3 - Build J2 ENDPIPE
PIPE Scan S4 - Build J3 ENDPIPE
• • • •
PIPE Scan SN+1 - Build JN ENDPIPE
ENPAR
PIPE Scan S1 - Probe J1 - Probe J2- • • • Probe JN ENDPIPE
ENSEQ

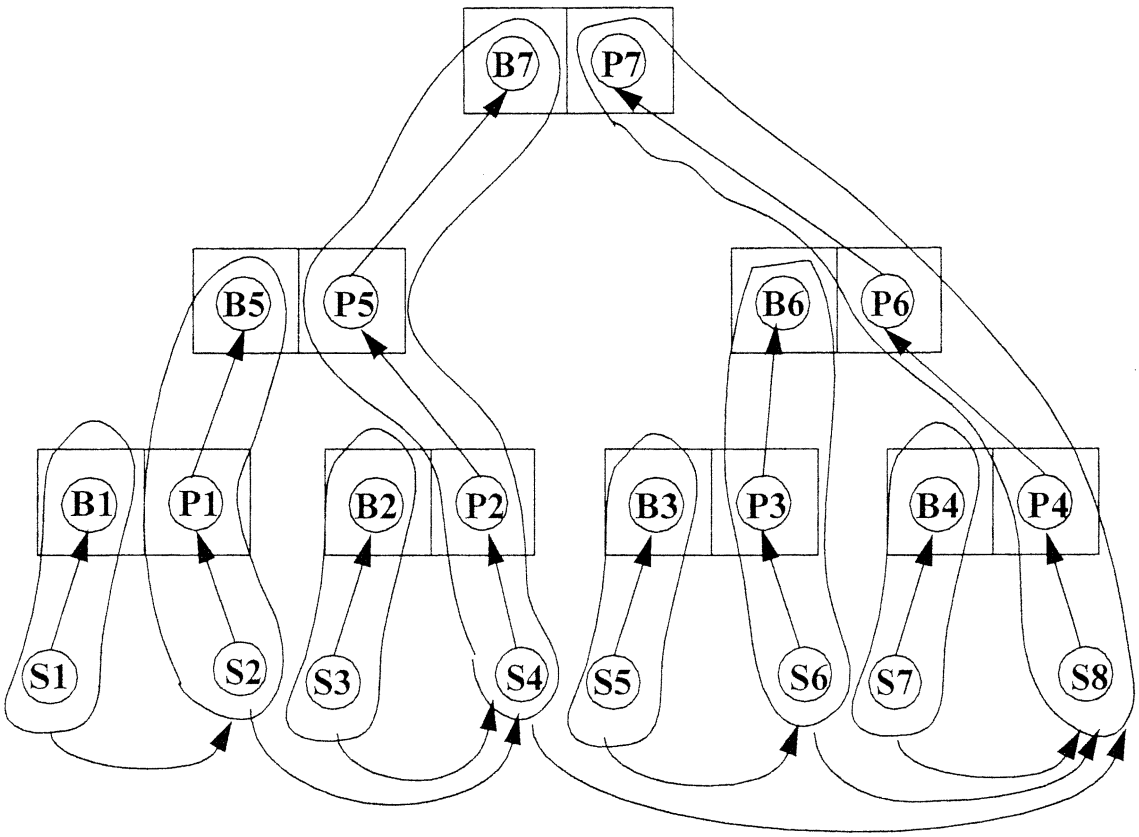
Arbre linéaire droit

Contention Mémoire [Sch90]



(Rn * ... (R3 * (R2 * R1)))
Right-Deep Tree

Sliced Right-Deep Tree




```
SEQ
    PAR
        PIPE Scan S1 - Build J1 ENDPIPE
        PIPE Scan S3 - Build J2 ENDPIPE
        PIPE Scan S5 - Build J3 ENDPIPE
        PIPE Scan S7 - Build J4 ENDPIPE
    ENPAR;
    PAR
        PIPE Scan S2 - Probe J1- Build J5 ENDPIPE
        PIPE Scan S6 - Probe J3 - Build J6 ENDPIPE
    ENDPAR;
    PIPE Scan S4 - Probe J2 - Probe J5 -Build J7 ENDPIPE
    PIPE Scan S8 - Probe J4 - Probe J6 - Probe J7 ENDPIPE
ENSEQ
```

Arbre ramifié (Bushy)

4. Approches à deux phases [Hon92, Has 95, Garo 96/97, Kab 98...]

☛ *Principe :* Φ_1 ; Φ_2

Φ_1 : *Génération d'un PE seq. optimal sans considerer les ressources (Opt. Physique)*

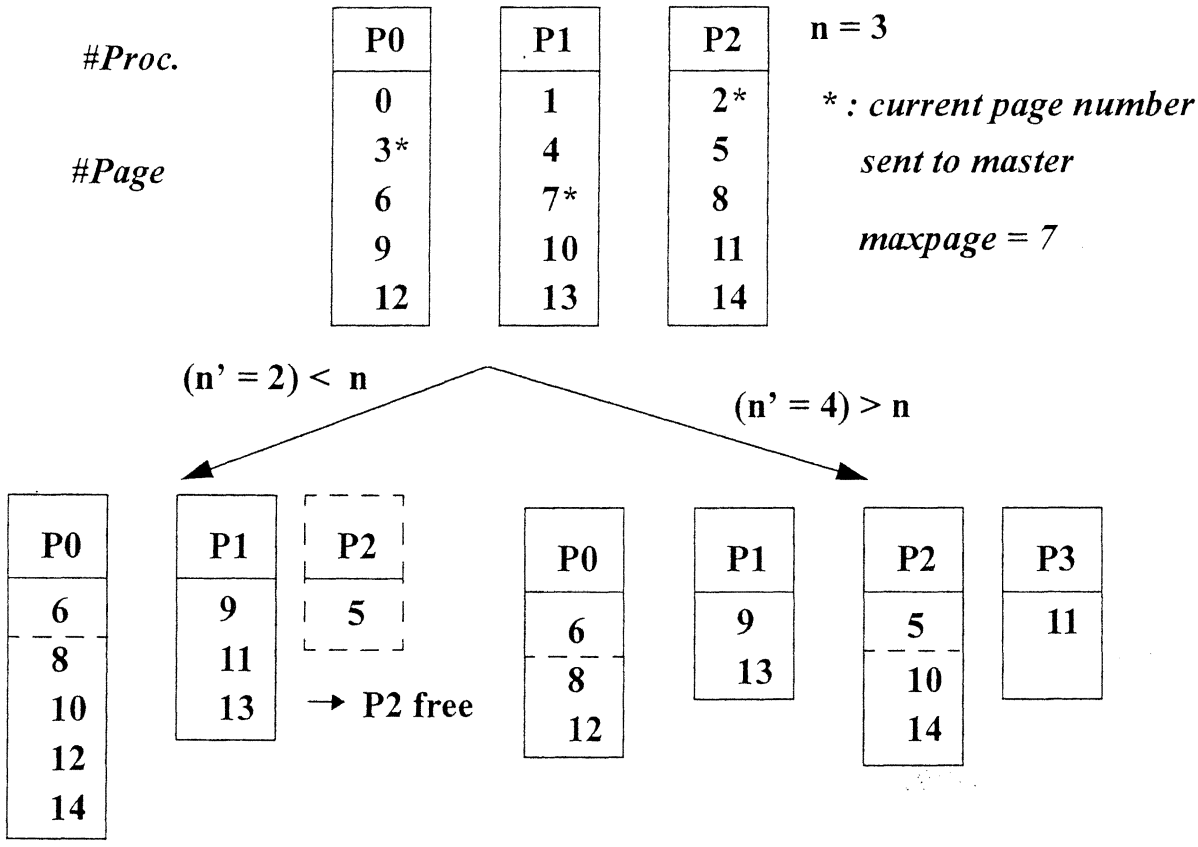
Φ_2 : *Allocation de ressources*

Φ_2 : *<Extraction du Parallélisme ; Allocation de Ressource>*

☛ *Méthodes d'ajustement du degré de parallélisme*

M1 : Page Partitioning (Répartition circulaire)

☛ *Ces méthodes sont exploitées par l'alg. d'ordonnancement adaptatif/point d'équilibre IO-CPU*



M1 : Page Partitioning Parallelism Adjustement

5. Formalisation du processus de placement [And88, Cos93]

Définition Informelle

"Placer" une application sur une architecture parallèle

i.e.

Allouer physiquement les processeurs de l'architecture parallèle aux tâches de l'application de manière à optimiser une fonction de coûts en respectant des contraintes d'allocation, de façon à garantir une utilisation quasi-optimale des ressources.

1. Definition formelle : Un placement est une application alloc qui associe un processeur à une tâche :

$$\textbf{alloc} : T \rightarrow P \;/\; \forall\, t \in T\,,\, \exists\, q \in P,\, alloc(t) = q$$

avec

T, l'ensemble des n tâches de l'application à placer et

P, l'ensemble des p processeurs de l'architecture parallèle

*** Nombre de placements possibles :** p^n

☛ *Approaches :*
- ◆ *Strategies Enumeratives*
- ◆ *Strategies Approximatives/Aleatoires*

2. Hypothèses :

{Type d'archi.//, Nature des Proc., RI (f/v), Mode d'exploitation, Type de placement}

3. Contraintes d'allocation

C0 : Contrainte de localité des données :
$Scan_i$ s'exécute là où se situent les données.

C1 : Contrainte de localisation des données :
$Alloc(Build_i) = Alloc(ProbeX_i)$

C2 : Contrainte de parallélisme indépendant :
$Alloc(T_i) \neq Alloc(T_j)$, si $i \neq j$ et
$\exists$ t / T_i et T_j s'exécutent simultanément

C3 : Contrainte de parallélisme pipeline :
$Alloc(ProbeX_i) \neq Alloc(pred(ProbeX_i))$ et
$Alloc(Build_i) \neq Alloc(pred(Build_i))$

4. Prise en compte du parallélisme intra-opération

⇨ *Allocation logique d'un ensemble de processeurs à chaque tâche du GdD*

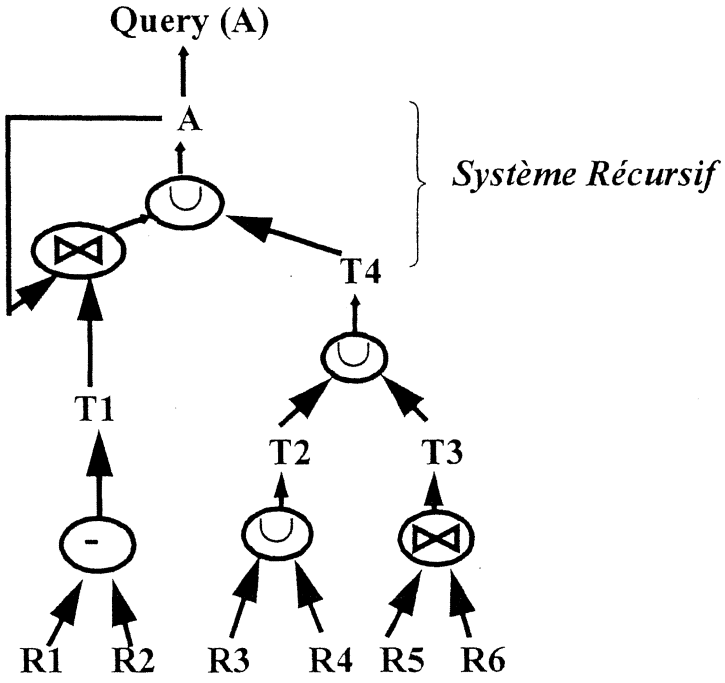
6. Synthèse des méthodes d'optim. de req. parallèles

Table 1: *Paramètres permettant une aide à la décision dans le choix d'une approche d'optimisation*

Approche *Paramètres*	*Mono-phase*	*Deux-phases*
Espace de rech	*Plus large*	*Plus réduit*
Strat. d'opti. //	*Extesiond'une strat. opt. phys.*	*Conception d'une strat.*
Coût d'opti. //	*Plus élevé*	*Moins élevé*
Modèles de coûts	*Extension d'un MC :*	*MC1 + MC2*
Implémentation	*Plus difficile*	*Moins difficile*

7. Performance des Formes de Parallélisme

- *Architecture à Mémoire Distribuée*
- *Benchmark [Bit 83, Gra93], & Paramètres [Sch 90,Val 88]*



- *Jointure par Hachage simple;*

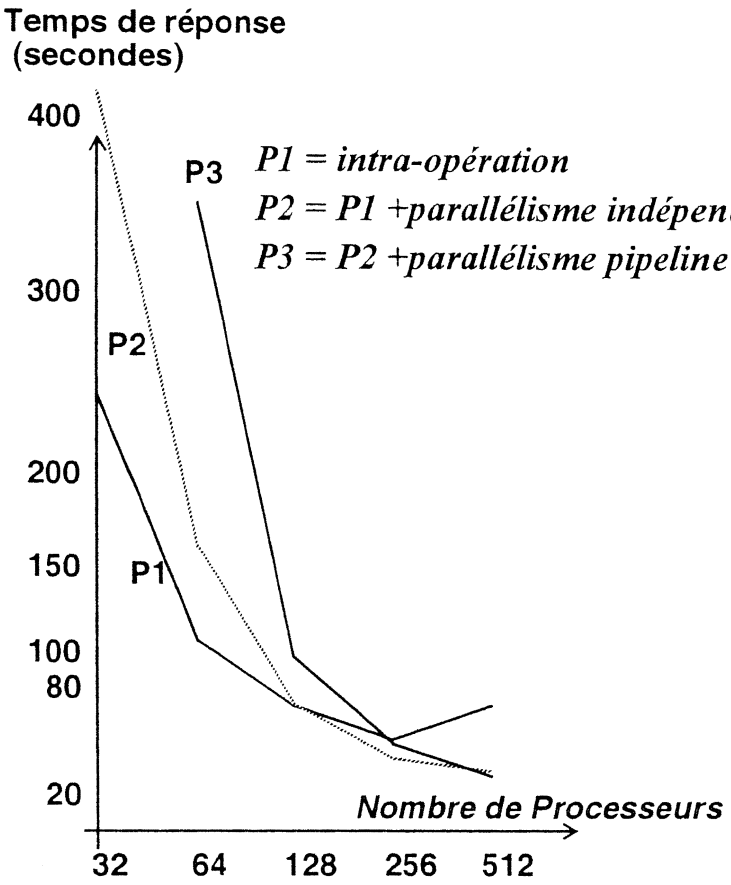
$$\text{TRL}\ (T \leftarrow R \bowtie S) = T_{ef} + T_d + T_{com} \quad \text{Où}$$

T_{ef}= (|R|/d).th + ((||R||/d) + (||S||/d)).CR + (||R||/d/q).(||S||/d).CJO + |T|.I + ||T||/d.CW

temps build + temps de lecture + temps de comparaison + temps de déplacement + temps d'écriture

T_d = (|T|/d).th and T_{com} = ((|T|/d).trf + p.msg). $\lceil d/p \rceil$

|R|	: Nombre de Tuples de R=10^6	Trf : Temps pour transf. 1 tuple
||R||	: Nombre de pages de R	msg : Temps pour envoy. 1 message
th	: Temps pour répartir 1 tuple = 3 µs	
CR	: Temps de lect. 1 page (18KB)= 8 ms	
CW	: Temps d'écriture 1 page= 16 ms	
I	: Temps pour déplacer un tuple	
CJO	: Temps pour joindre 2 pages non triées	
d	: Nombre de proc. de l'opér. source	
p	: Nombre de proc. de l'opér. destination	



◆ *Efficacité du Parallélisme :*

- *Intra-opération avec un petit NB de processeurs*
- *Pipeline avec un grand NB de processeurs*

➥ *Coûts de Communication*

V. Optimisation des Coûts de Communication

- *Optimisation Logique :*

  ⇨ *Réduire le volume de données manipulées*

  ☛ *Approche : Application des régles de Transformation des Arbres Algébriques*
  {Join Select --> Select Join ; Union Select --> Select Union,....}

- *Optimisation Physique : Ordonnancement des jointures*

  ⇨ *Réduire les E/S*

  ☛ *Approche : Exploitation des accélérateurs (clés et index)*

- *Parallélisation inter-opération (Ordonnancement parallèle & Placement):*

  ➜ *Coûts de comm. Inter-opération*

  - *Limiter le volume de données transférées*

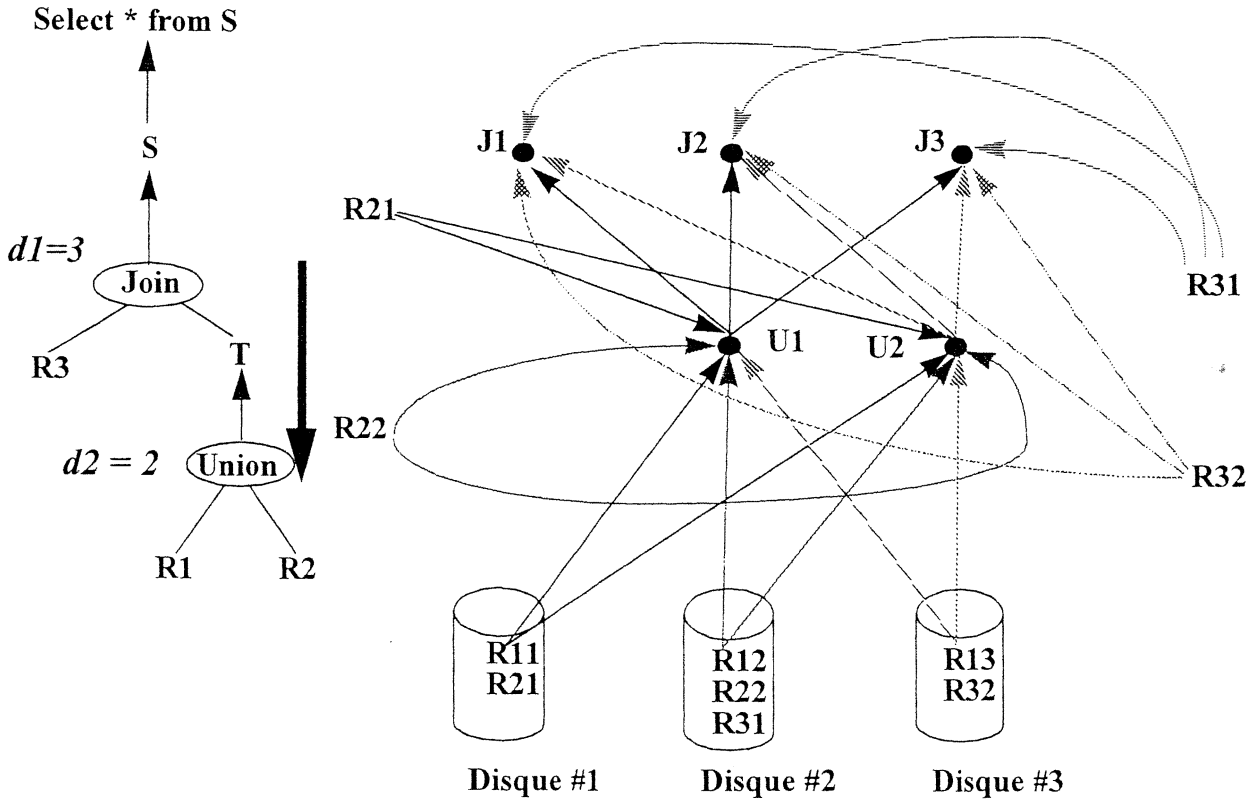
  ➜ *Coûts de com. Inter-processeur*

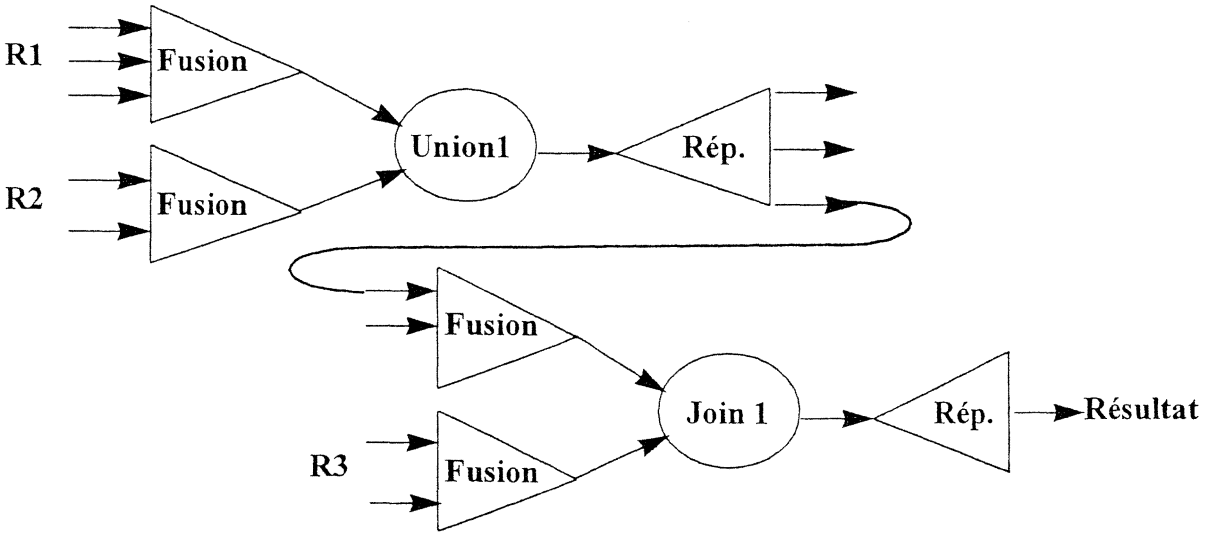
  - *Minimiser les temps de transfert inter-proc. (Topo. RI)*
  - *Minimiser l'impact des temps start-up pipeline*

1. Minimisation des Coûts de Com. Inter-opération

➥ *Coûts de la Redistribution de données > Coût d'Exécution*

Méthodes : Propagation <Attr_répart., Nb_proc> [Ham 93]
Coloriage d'un arbre [Has 95]



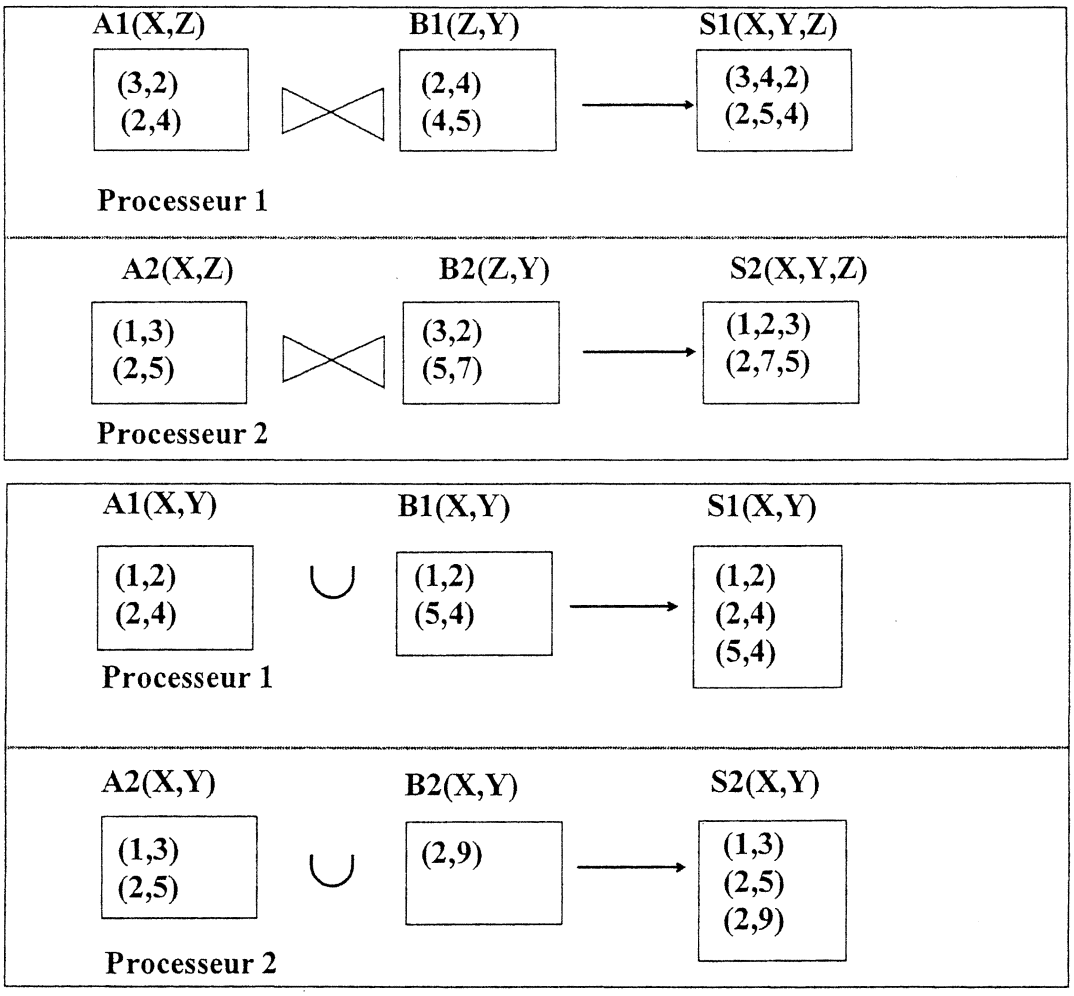


⇨ *NB important de messages de données et de contrôle*

A. Conditions de Redistribution des données

- ***Fonction de répartition :*** *Une relation produite par une opération n'est pas consommée avec la même fonction de répartition par l'opération succ.*
- ***Attribut de répartition :*** *Une relation produite est consommée par l'opération successeur avec un attribut de répartition différent.*
- ***Allocation de processeurs:*** *Les nombres de processeurs alloués à une opération et son successeur sont différents.*

B. Classe des Opérations : C1, C2



C. *Méthode de Propagation : Attributs de Répartition & Nombre de Processeurs*

- *Hypothèse : même fonction de répartition*

Pour chaque OP (i) ∈ graphe de la requête faire

E1 : OP(i) ⇨ (meilleur Algorithme, NB_P, TRL)

☛ *Evaluateur de coûts :*

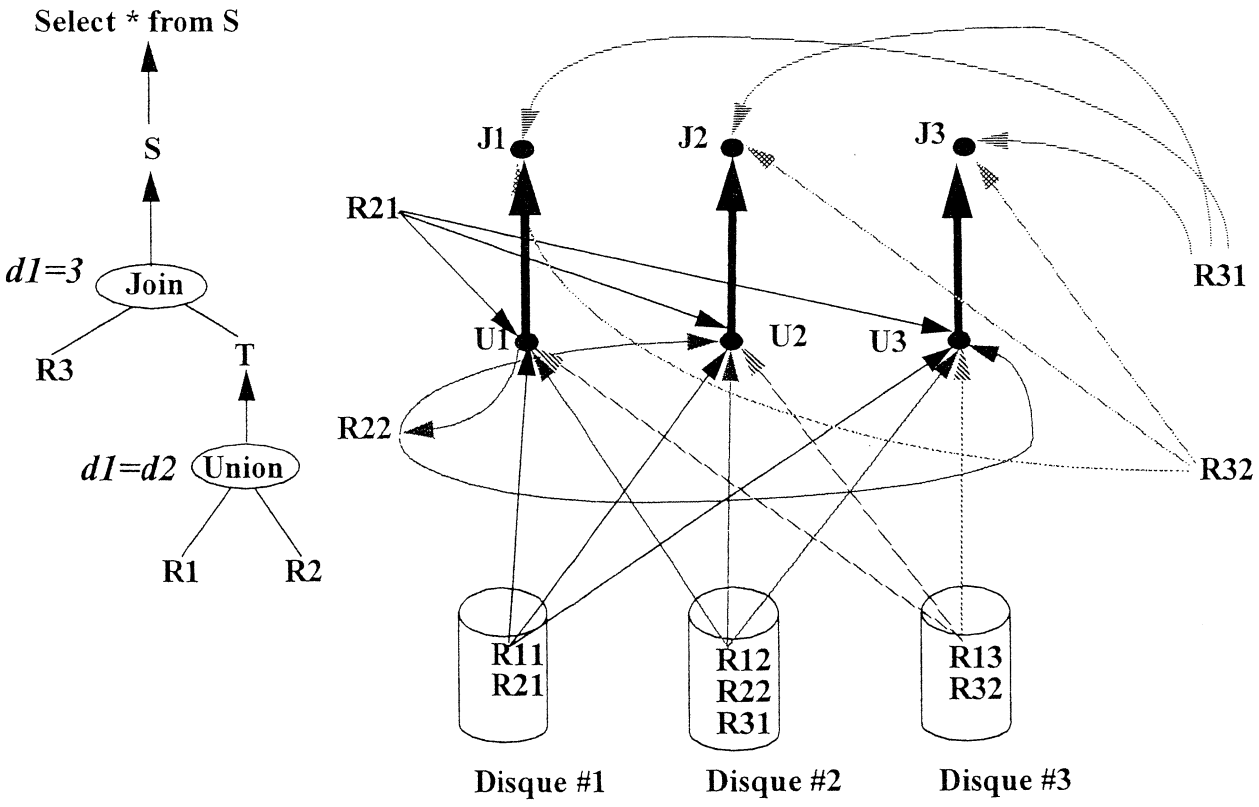
(i) Recherche le meilleur Algorithme (TR)
➜ Table de décision (|relations|)

(ii) Détermine : (NB_P, TRL1)

E2 : Propager (Attr_Succ, NP_Succ) si les 2 cond. suivantes sont vérifiées :

(1) OP(i) ∈ C2 ou Attr_Succ correspond à un attribut de jointure pour OP (i)

(2) le processus de propagation améliore le TRL.



VI. Conception d'un Evaluateur de Coûts

☛ *Motivations*

➥ *Optimisation physique: Choix d'une Stratégie de recherche*

- ◆ *Algorithmes Enumératives (exacts): RLD, AH, AH**
- ◆ *Algorithmes Aléatoires : AI, RS, TRS et Génétiques*

➥ *Détermination du degré du parallélisme :*

- ◆ *Intra-opération : Choix d'un algorithme (seq/par)*
- ◆ *Propagation des attr. de répartition et du nb Procs*

➥ *Placement des opérations :*

- ◆ *fonction de coût à minimiser*

☛ *Modèles de Coûts clasiques :*

— *Ne prennent pas en compte l'env. / Modèles de Coûts*

— *Ne sont pas compatibles avec les autres systèmes*

➥ *Séparer l'optimiseur et l'évaluateur de coûts [And 91]*

Raisonnement ⇔ ***Connaissances***

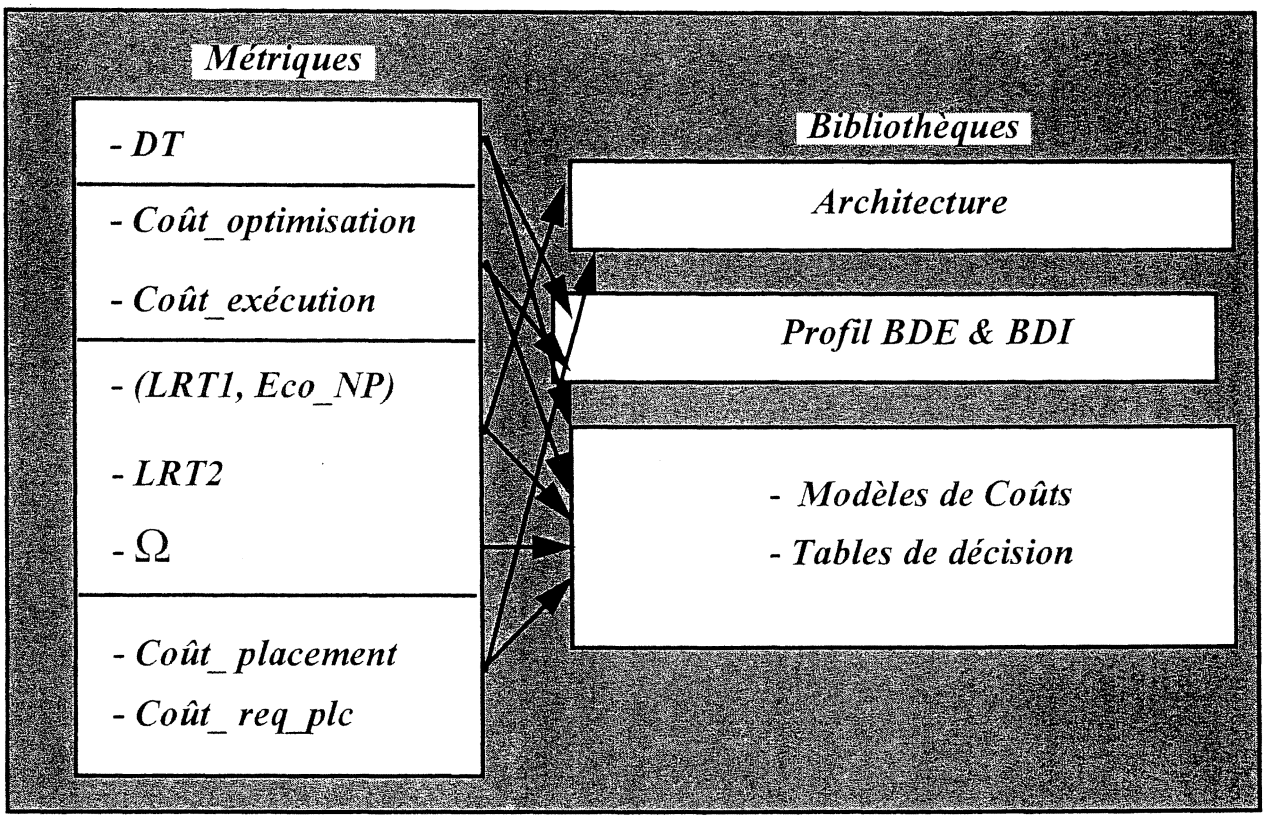
- • *Optimisation physique*
- • *Détermination du nombre processeurs économique*
- • *Optimisation de requêtes récursives*
- • *Optimisation des communications de données*

➥ *EC = < Métriques + Librairies >*

- *Avantages :*
  - *Extensibilité : Méthodes d'Accès, Algo., Profile BD,...*
  - *Flexibilité : Ajout de nouvelles métriques*
  - *Adaptabilité : aux diff. machines BD.*

- *Evaluation des métriques :*

  ➥ *Représentation de l' environnement ⇨ Bibliothèques*



- *Métriques de l'Evaluateur de Coûts*
  - *Nombre de tuples déduits : DT*
  - *Cout d'optimisation : Topt*
  - *Qualité d'optimisation (trade-off)*
  - *Nombre de processeurs et TRL1*

  OP(i) ⇨ meilleur algo. NBP_ECO, TRL

  (i) Recherche du meilleur alg. (table de décision 1)
  (ii) Détermination (NBP_ECO, TRL1)

- *Propagation des attr. de répartition et du NBP_ECO*
  - ➥ *Table Décision 2 (|Ri|, NBPROC) ⇨ Algo.*

- *Placement des opérations : fonction de coût*

4. Représentation de l'environnement d'une machine parallèle

➥ *Profile de la BD*

- *Cardinalité des relations de base*
- *Valeur minimale et maximale d'un attribut*
- *Nombre de valeurs distinct d'un attribut*
- *Taille d'un attribut*
- *Fin, Fout, base*

➥ *Modèles de Coûts*

- *Fonctions de coûts des métriques (DT,...)*
- *Tables de décision*

➥ *Architecture*

- *Type d'architecture (Type Mémoire, Type réseau, Type cpu, Type système, nombre de processeurs)*
- *Mémoire (taille mémoire , tps d'accès mémoire)*
- *CPU (tps d'une opération simple, tps de distribution)*
- *Système (taille d'1 page, tps lecture/écriture d'1 page)*
- *Réseau (tps pour envoyer 1 tuple, tps pour envoyer 1 message)*