Parallélisme des opérations de haut niveau sur les fichiers

Une des manières d'améliorer les performances (en présence de volumes importants de données à traiter) est l'utilisation du parallélisme

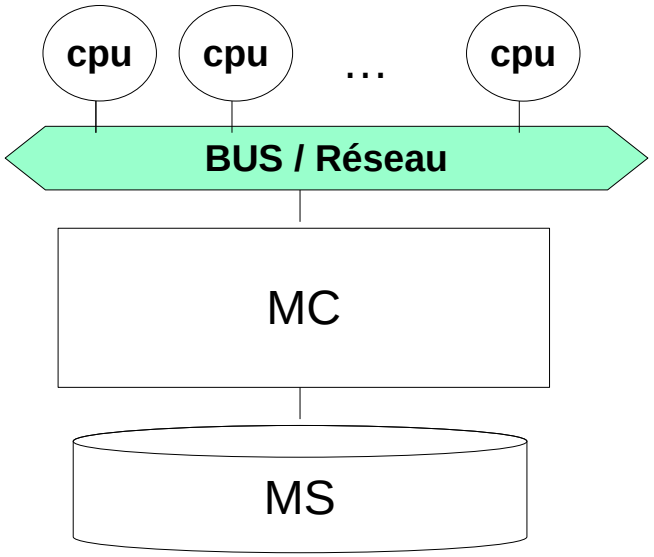
Plan

Architectures parallèles

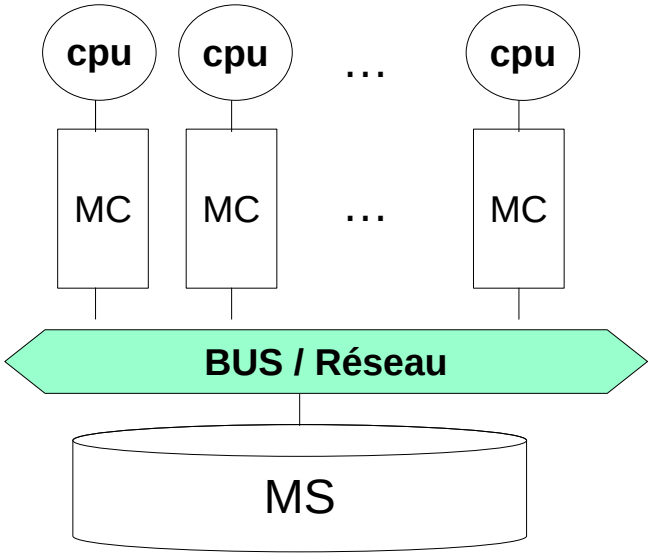
Tri, Fusion et Jointures parallèles

Architectures Parallèles

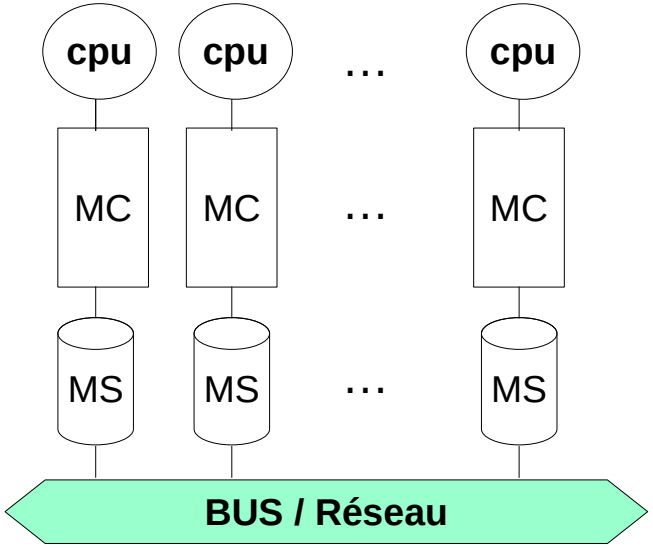
Shared Memory



Shared Disk



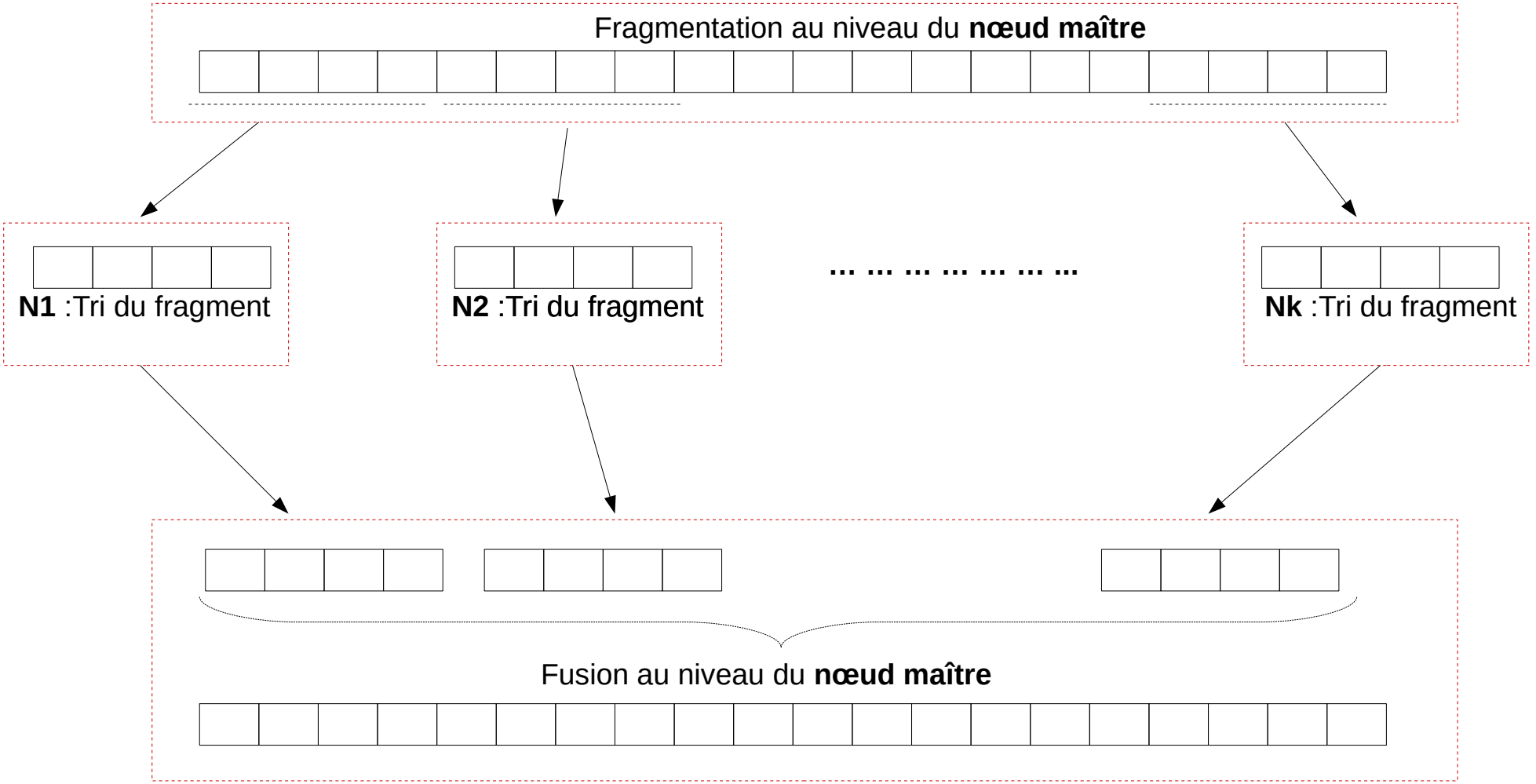
Shared Nothing (cluster)



Tri parallèle (sur cluster)

Première approche

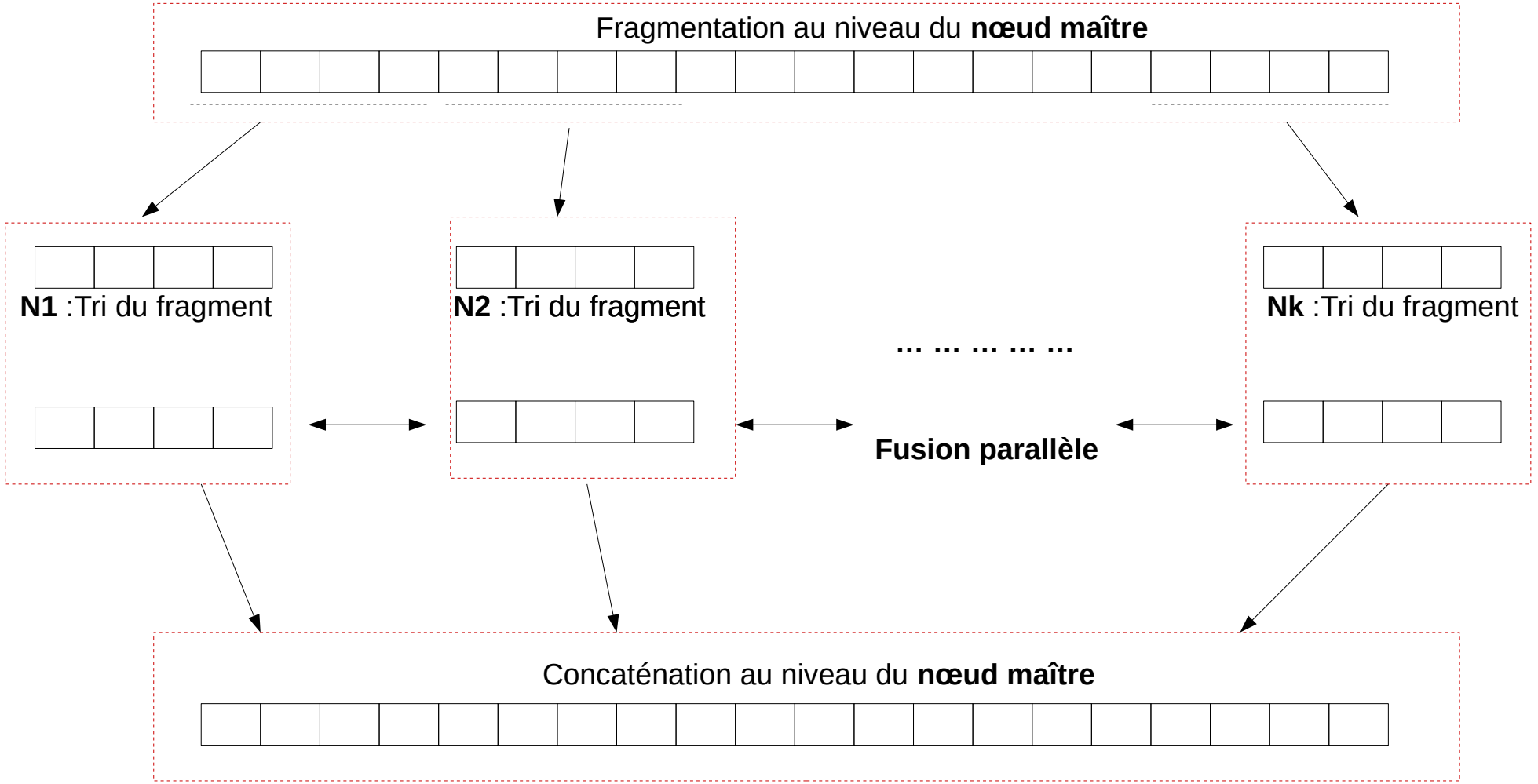
- Fragmentation par position (découpage en fragments de même taille)
- Tri des fragments (au niveau de chaque nœud)
- Fusion des résultats



Tri parallèle (sur cluster)

Première approche (améliorée)

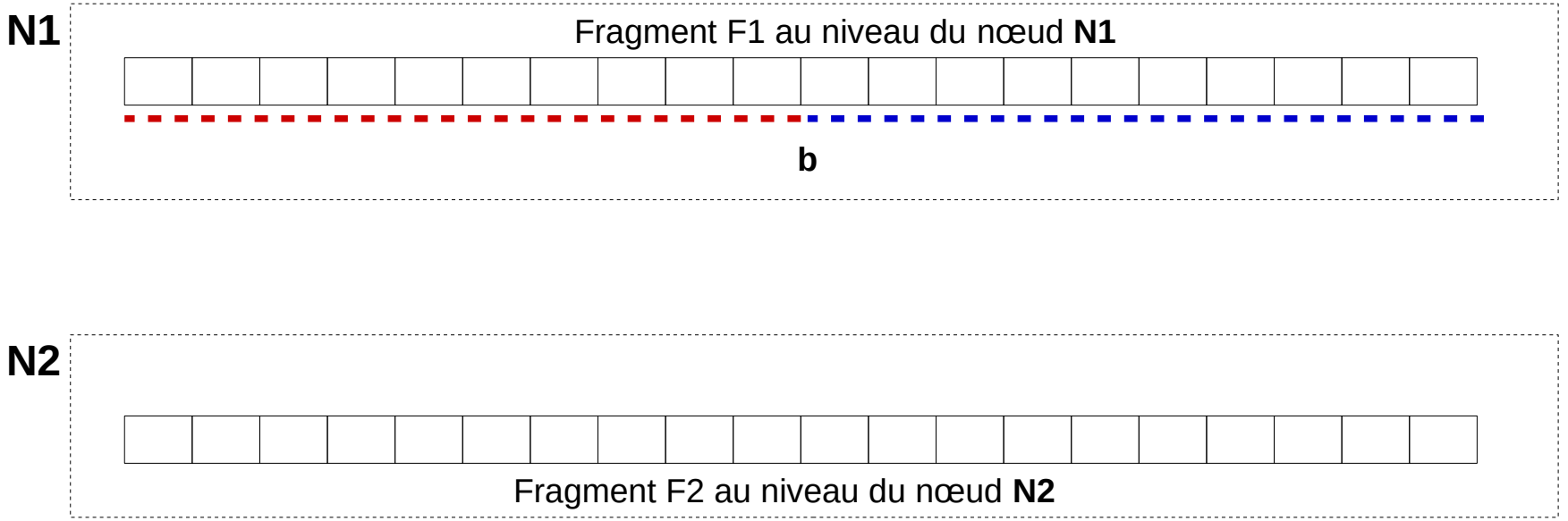
- Fragmentation par position (découpage en fragments de même taille)
- Tri des fragments (au niveau de chaque nœud)
- Fusion des résultats



Tri parallèle (sur cluster)

Fusion parallèle de 2 fragments (avec 2 nœuds N1 et N2)

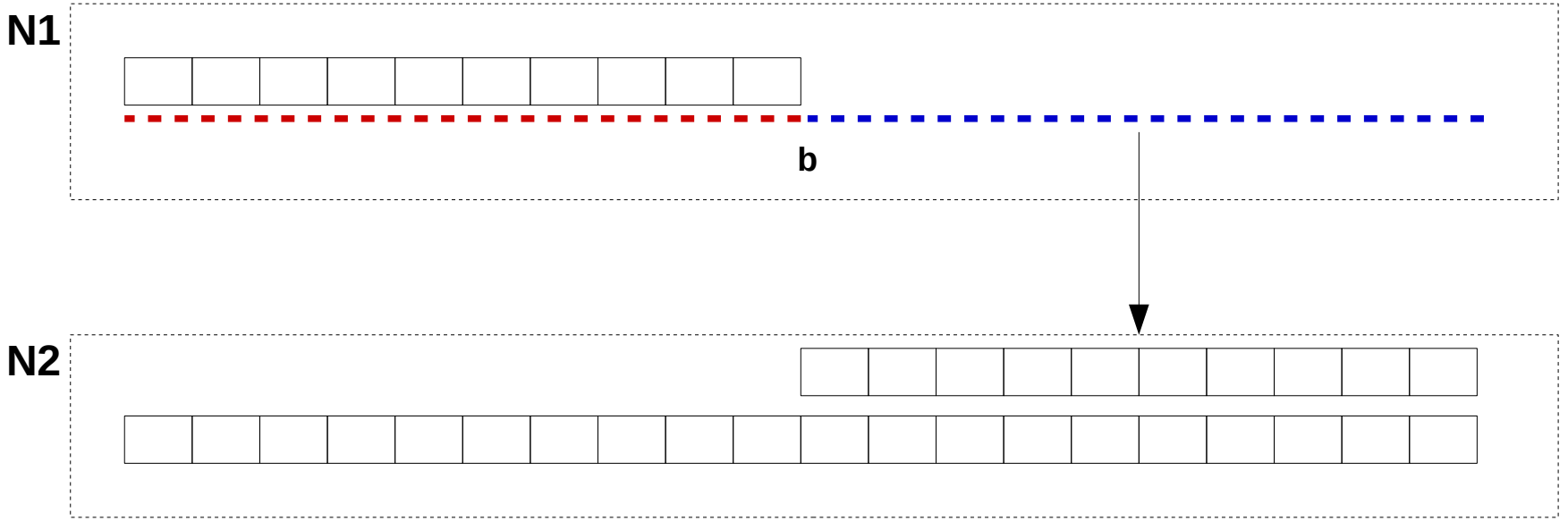
- Le nœud N1 dispose d'un fragment trié → F1
- Le nœud N2 dispose d'un fragment trié → F2
- N1 décompose son fragment en 2 parties égales : partie(1,1) et partie(1,2)



Tri parallèle (sur cluster)

Fusion parallèle de 2 fragments (avec 2 nœuds N1 et N2)

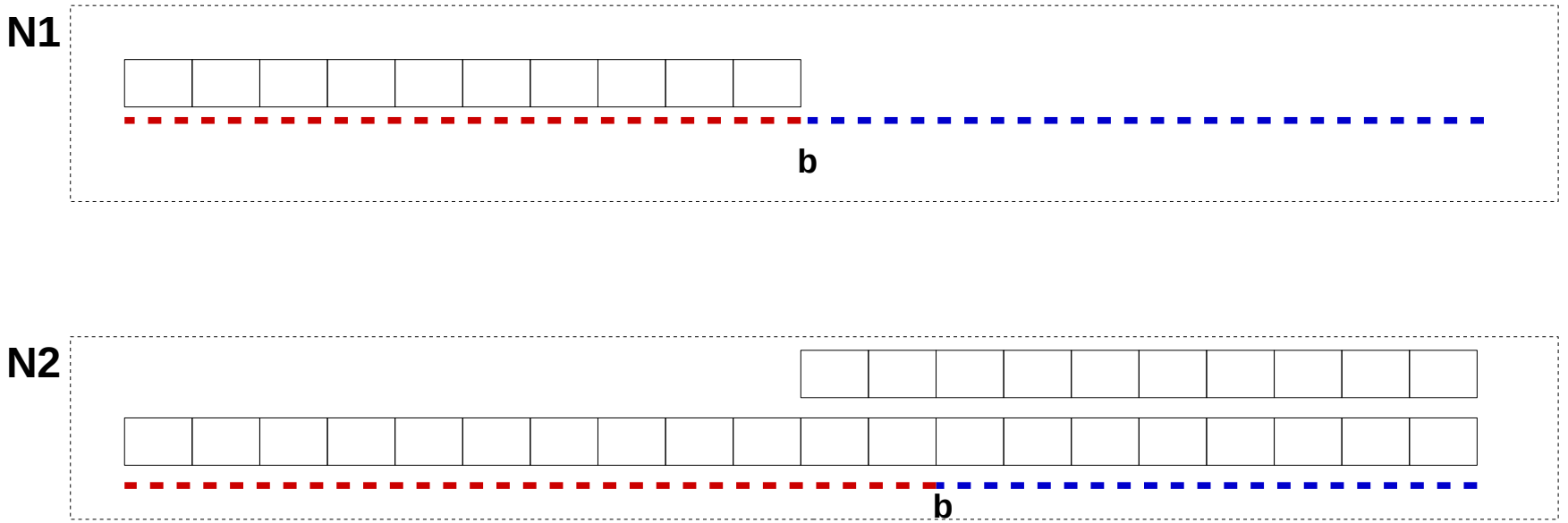
- Le nœud N1 dispose d'un fragment trié → F1
- Le nœud N2 dispose d'un fragment trié → F2
- N1 décompose son fragment en 2 parties égales : partie(1,1) et partie(1,2)
- N1 envoie une des 2 parties (avec la borne de séparation **b**) à N2



Tri parallèle (sur cluster)

Fusion parallèle de 2 fragments

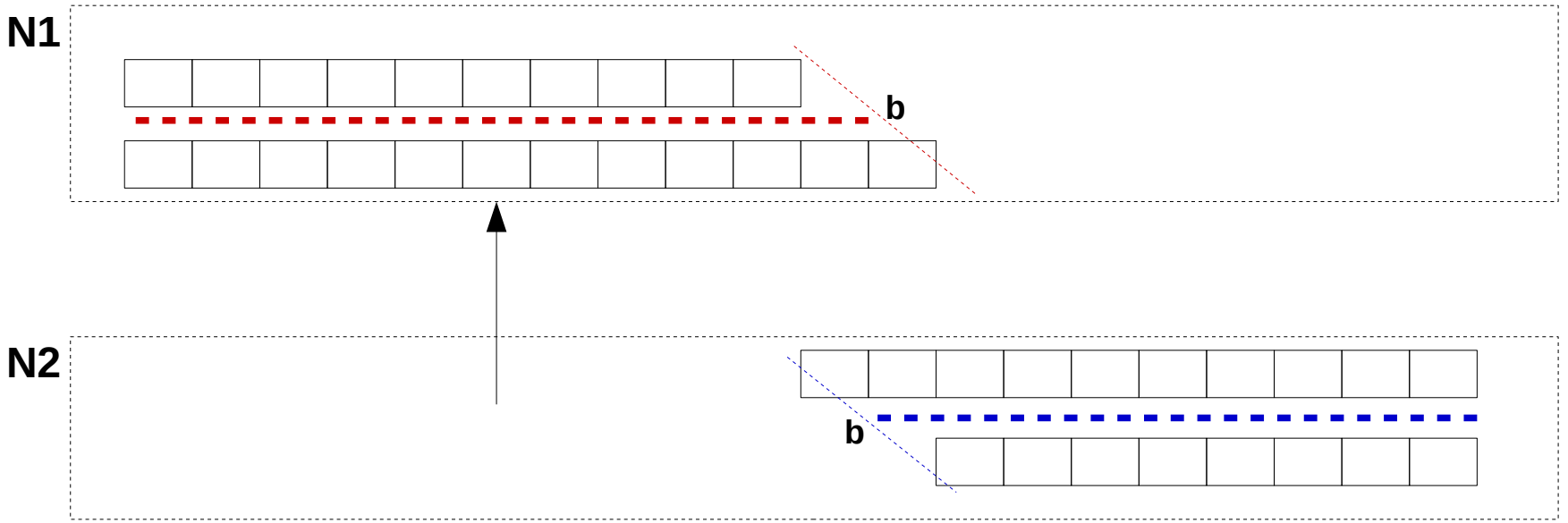
- Le nœud N1 dispose d'un fragment trié F1
- Le nœud N2 dispose d'un fragment trié F2
- N1 décompose son fragment en 2 parties égales : partie(1,1) et partie(1,2)
- N1 envoie une des 2 parties (avec la borne de séparation **b**) à N2
- N2 décompose alors son fragment en 2 parties : partie(2,1) et partie(2,2)



Tri parallèle (sur cluster)

Fusion parallèle de 2 fragments

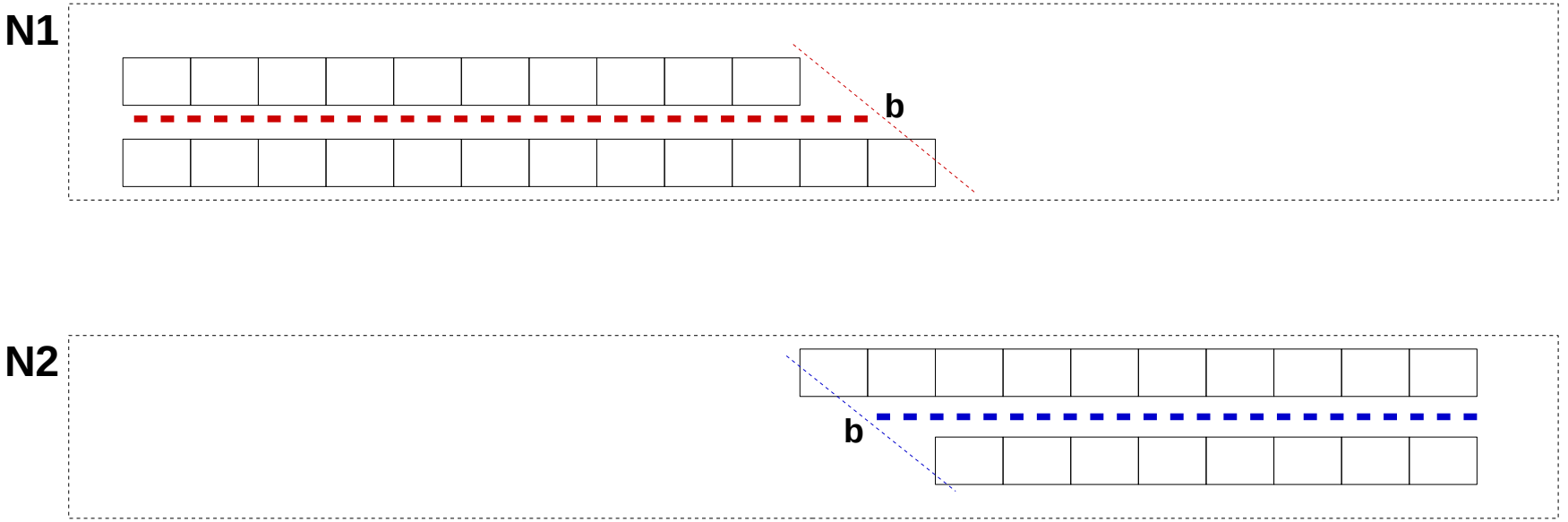
- Le nœud N1 dispose d'un fragment trié F1
- Le nœud N2 dispose d'un fragment trié F2
- N1 décompose son fragment en 2 parties égales : partie(1,1) et partie(1,2)
- N1 envoie une des 2 parties (avec la borne de séparation **b**) à N2
- N2 décompose alors son fragment en 2 parties : partie(2,1) et partie(2,2)
- N2 envoie l'autre partie de son propre fragment à N1



Tri parallèle (sur cluster)

Fusion parallèle de 2 fragments

- Le nœud N1 dispose d'un fragment trié F1
- Le nœud N2 dispose d'un fragment trié F2
- N1 décompose son fragment en 2 parties égales : partie(1,1) et partie(1,2)
- N1 envoie une des 2 parties (avec la borne de séparation **b**) à N2
- N2 décompose alors son fragment en 2 parties : partie(2,1) et partie(2,2)
- N2 envoie l'autre partie de son propre fragment à N1
- N1 et N2 effectuent en parallèle la fusion des parties en leur possession
  Dans N1 : Fusion (partie(1,1) , partie(2,1))
  Dans N2 : Fusion (partie(1,2) , partie(2,2))
- Les résultats des fusions sont envoyés au nœud maître pour être concaténés.



Généralisation : fusion parallèle de k fragments (sur k nœuds)

Principe

Chacun des k fragments (Fi) est découpé en k partie : [P(i,1) , P(i,2) , … P(i,k)]
Chacun des k nœuds (Ni) effectue une multi-fusion des parties numéro i de chaque fragment (P(1,i),…P(k,i))

Tâche de N_1 :

- N1 décompose son fragment en k parties égales : p(1,1) , p(1,2) … p(1,k)
  et diffuse les k-1 *bornes de séparation* aux autre nœuds (N2, N3, … Nk)
- N1 garde pour lui la première partie et envoie les autres parties aux nœuds restant :
  p(1,2) → N2 ; p(1,3) → N3 … p(1,k) → Nk
- N1 se met en attente des parties qui le concernent (provenant des autres nœuds)
- Dès que N1 reçoit toutes les parties p(*,1) des autres nœuds, il entame la multi-fusion des parties en sa possession : MultiFusion(p(1,1) , p(2,1), … p(k,1))
- Le résultat es envoyé au nœud maître.

Tâche de N_i (i = 2 .. k) :

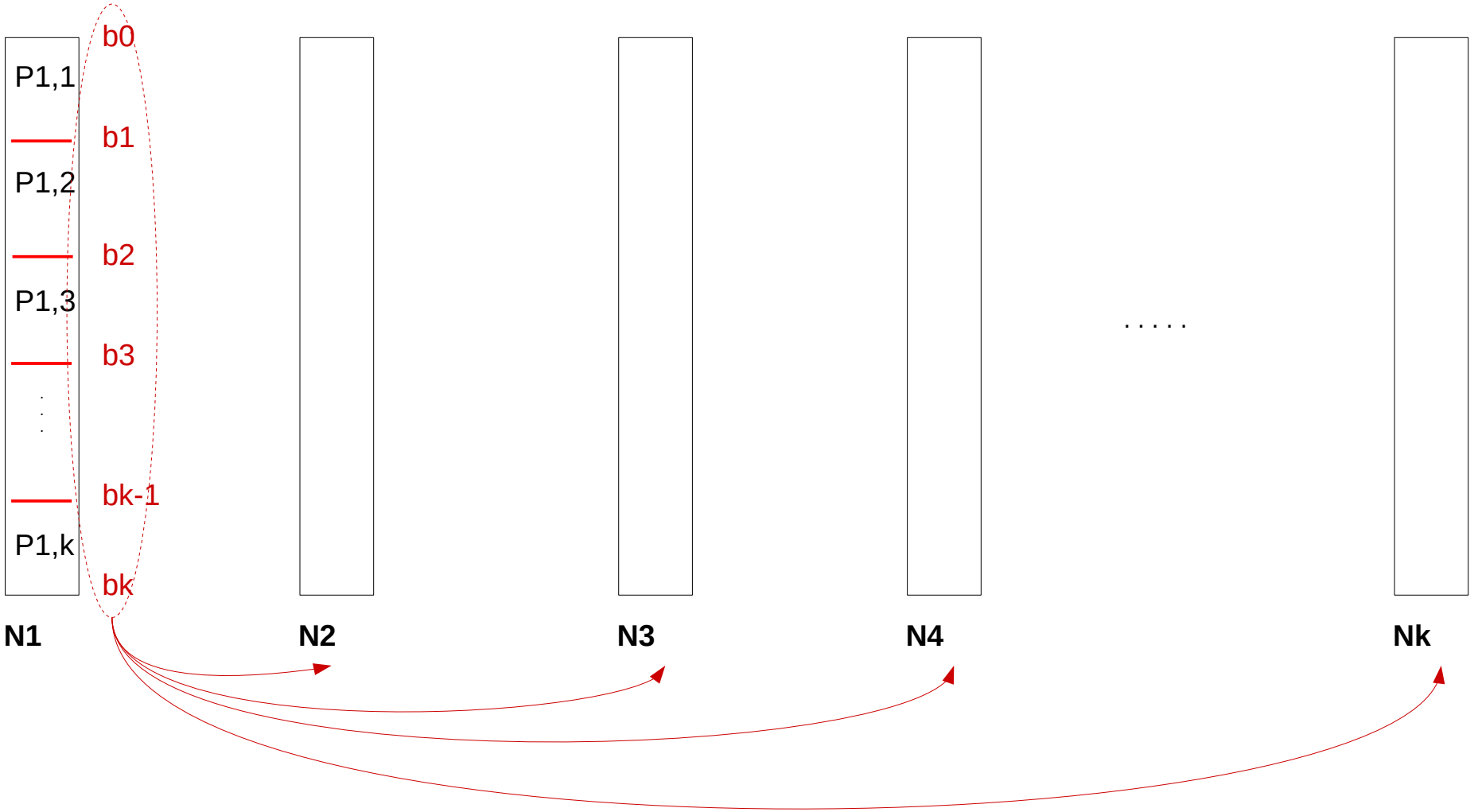
- Ni se met en attente des *bornes de séparation* (provenant de N1)
- Ni décompose son fragment local en k parties (à l'aide des bornes reçus de N1) :
  p(i,1) , p(i,2) … p(i,k)
- Ni garde pour lui la partie p(i,i) et envoie le reste aux autres nœuds :
  p(i,1) → N1 ; p(i,2) → N2 … p(i,k) → Nk
- Ni se met en attente des parties qui le concernent (provenant des autres nœuds)
- Dès que Ni reçoit toutes les parties p(*,i) des autres nœuds, il entame la multi-fusion des parties en sa possession : MultiFusion(p(1,i) , p(2,i), … p(k,i))
- Le résultat es envoyé au nœud maître.

Généralisation : fusion parallèle de k fragments (sur k nœuds)

N1 découpe son fragment en k parties égales

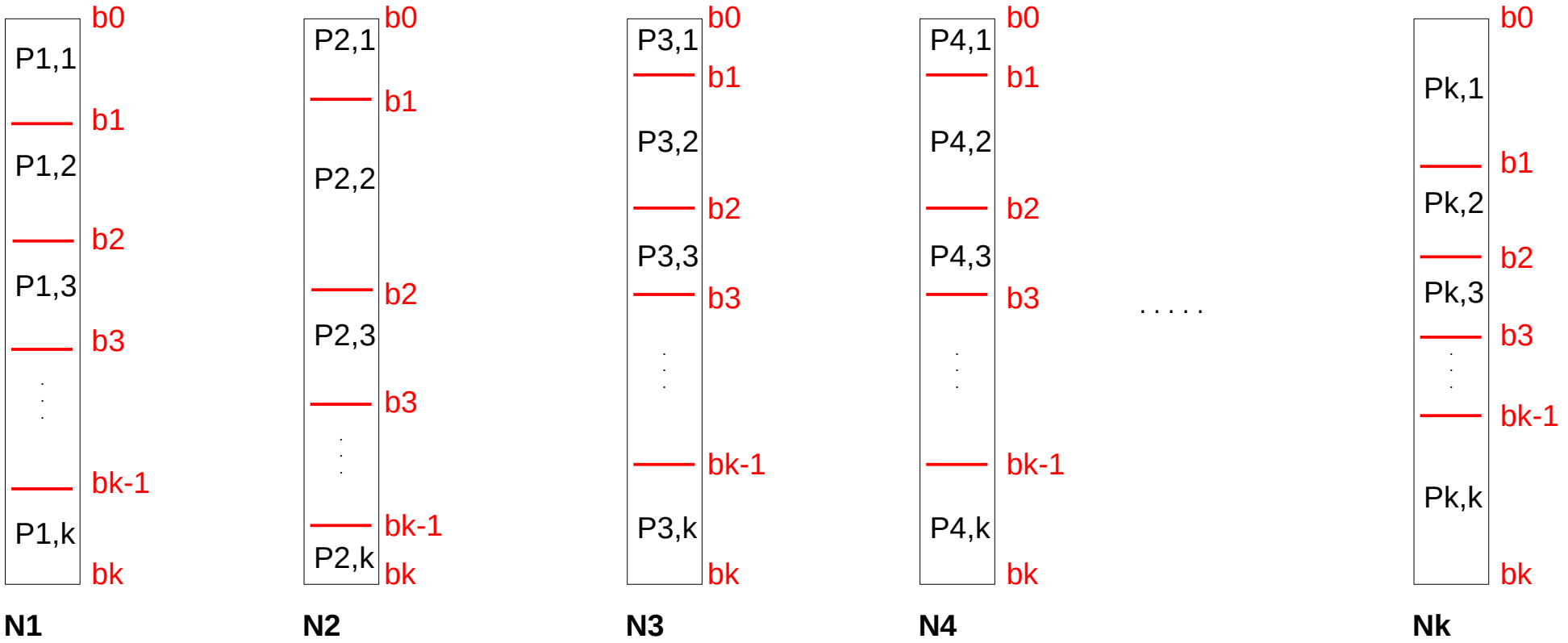
→ cela détermine les bornes de séparation : b0, b1 , … bk

→ les bornes de séparation sont envoyées aux autres nœuds

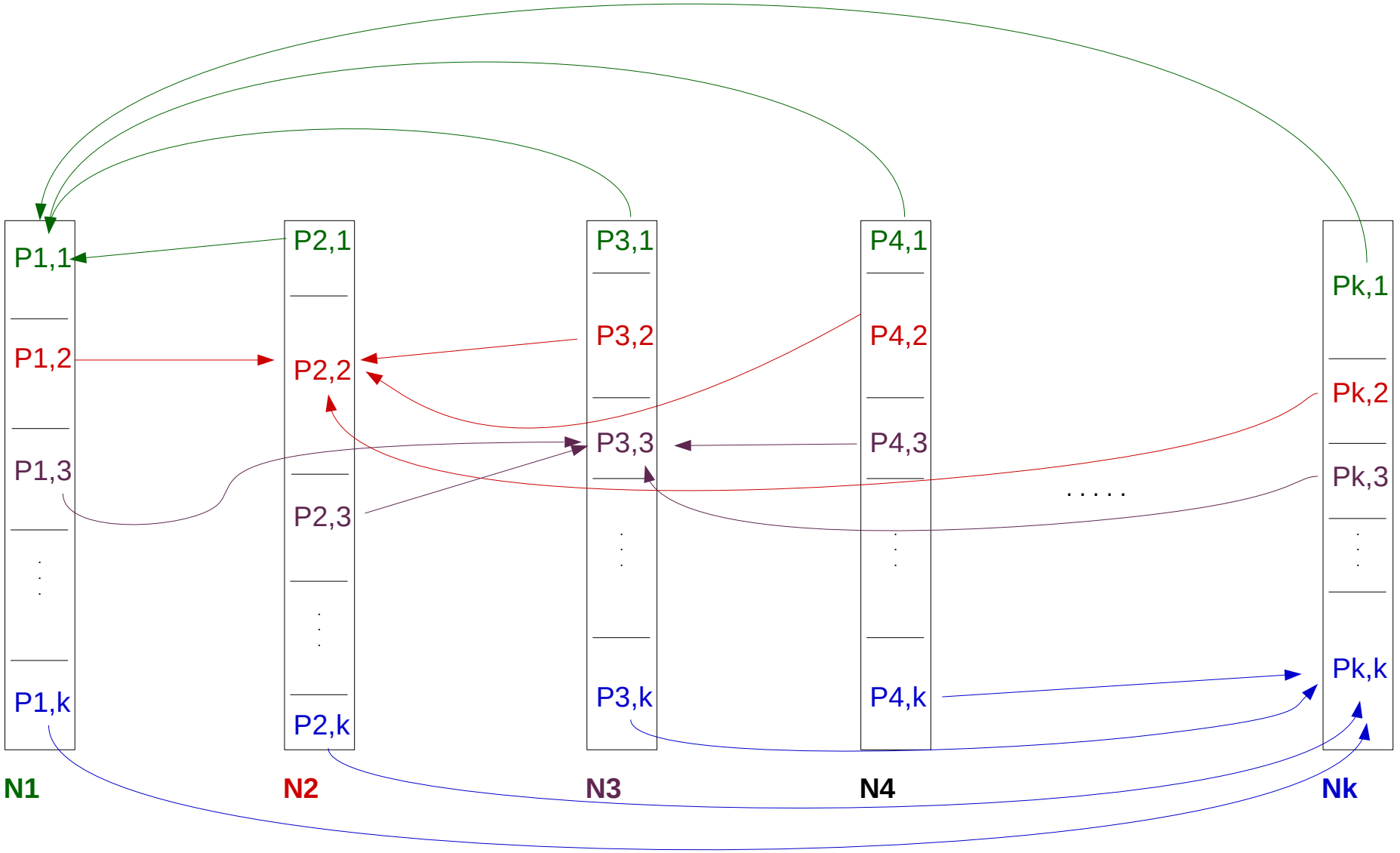


Généralisation : fusion parallèle de k fragments (sur k nœuds)

Les autres nœuds découpent leurs fragments respectifs à l'aide des bornes reçus de N1



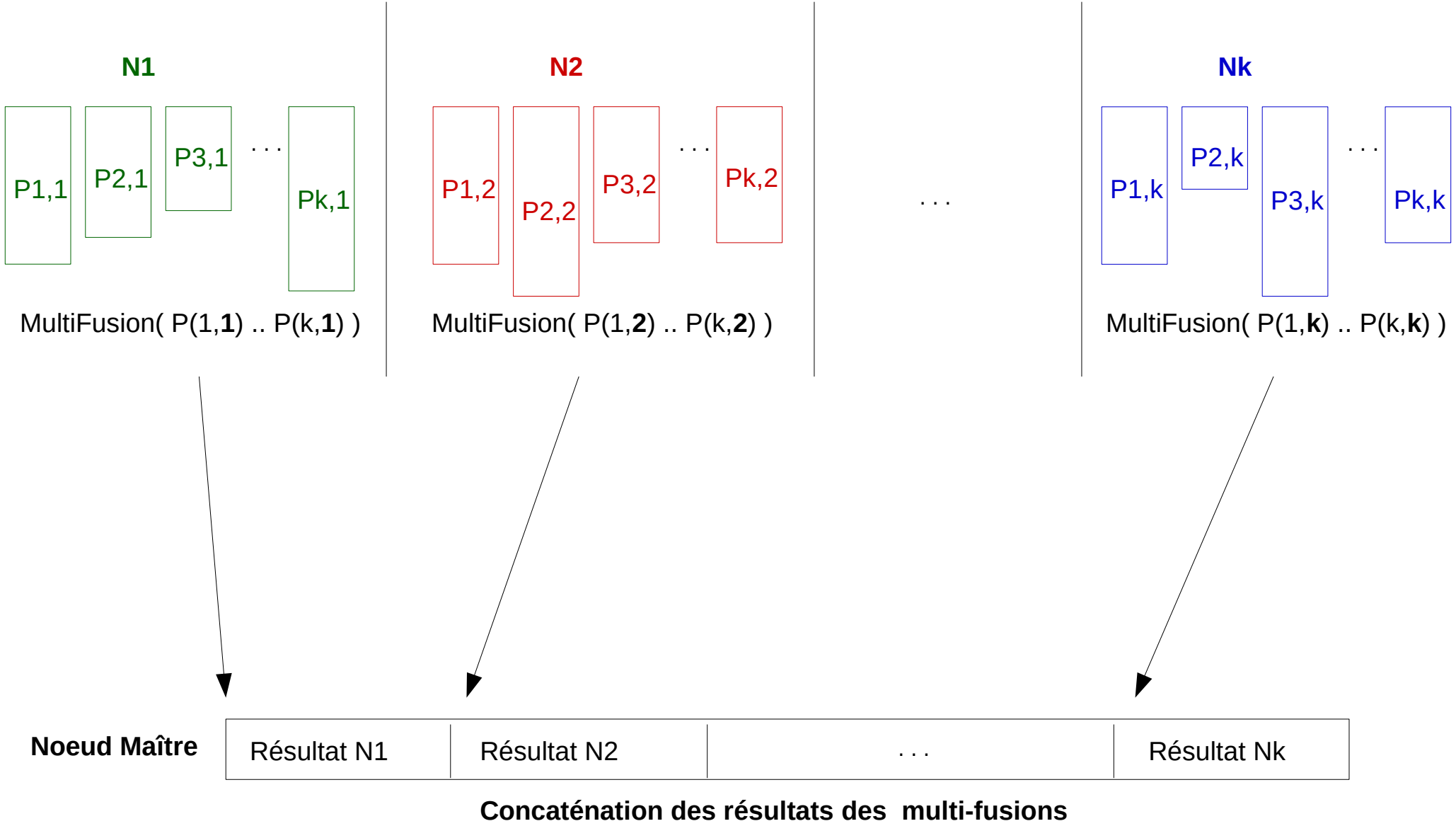
Généralisation : fusion parallèle de k fragments (sur k nœuds)



Chaque nœud **Ni** garde pour lui la partie **P(i,i)** et envoie les autres parties **P(i,j)** aux nœuds concernés : **Nj** (avec j ≠ i)

Généralisation : fusion parallèle de k fragments (sur k nœuds)

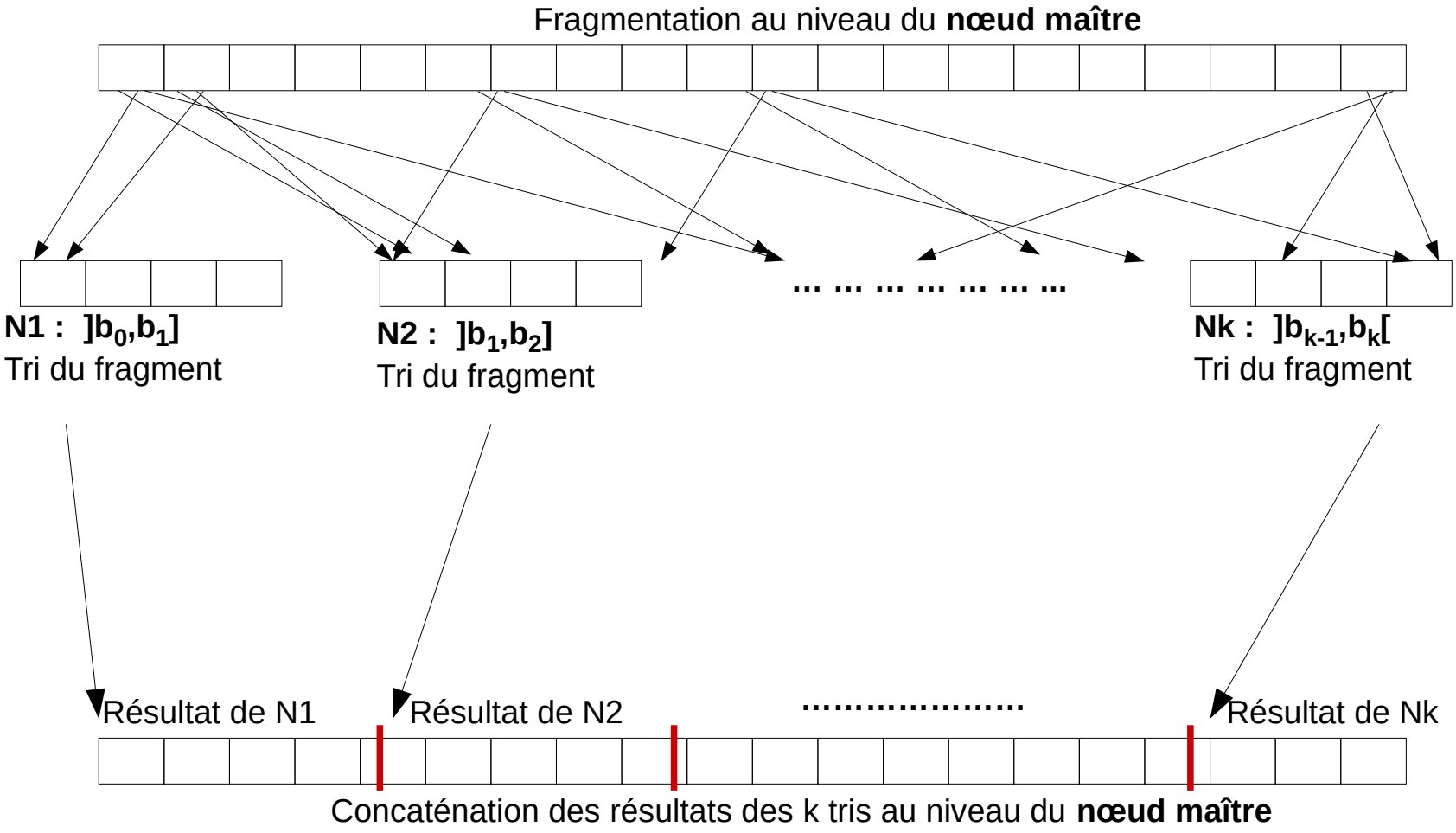
Chacun des nœuds **Ni** effectue une **multi-fusion** des parties qui le concernent
et envoie le résultat au **nœud maître** pour être concaténé avec les autres résultats



Tri parallèle (sur cluster)

Deuxième approche

- Fragmentation par valeur (par intervalle)
- Tri des fragments (au niveau de chaque nœud)
- Concaténation des résultats



Jointure parallèle (sur cluster)

Première approche (découpage par position)

- Le nœud maître fragmente R en k parties de tailles égales : r_1, r_2, … r_k
- Chaque partie ri est envoyée avec le 2^e fichier S au nœud Ni
- Chaque Nœud Ni effectue la jointure ri * S et renvoie le résultat au nœud maître
- Le nœuds maître concatène les résultats (dans un ordre quelconque)

Deuxième approche (découpage par valeur)

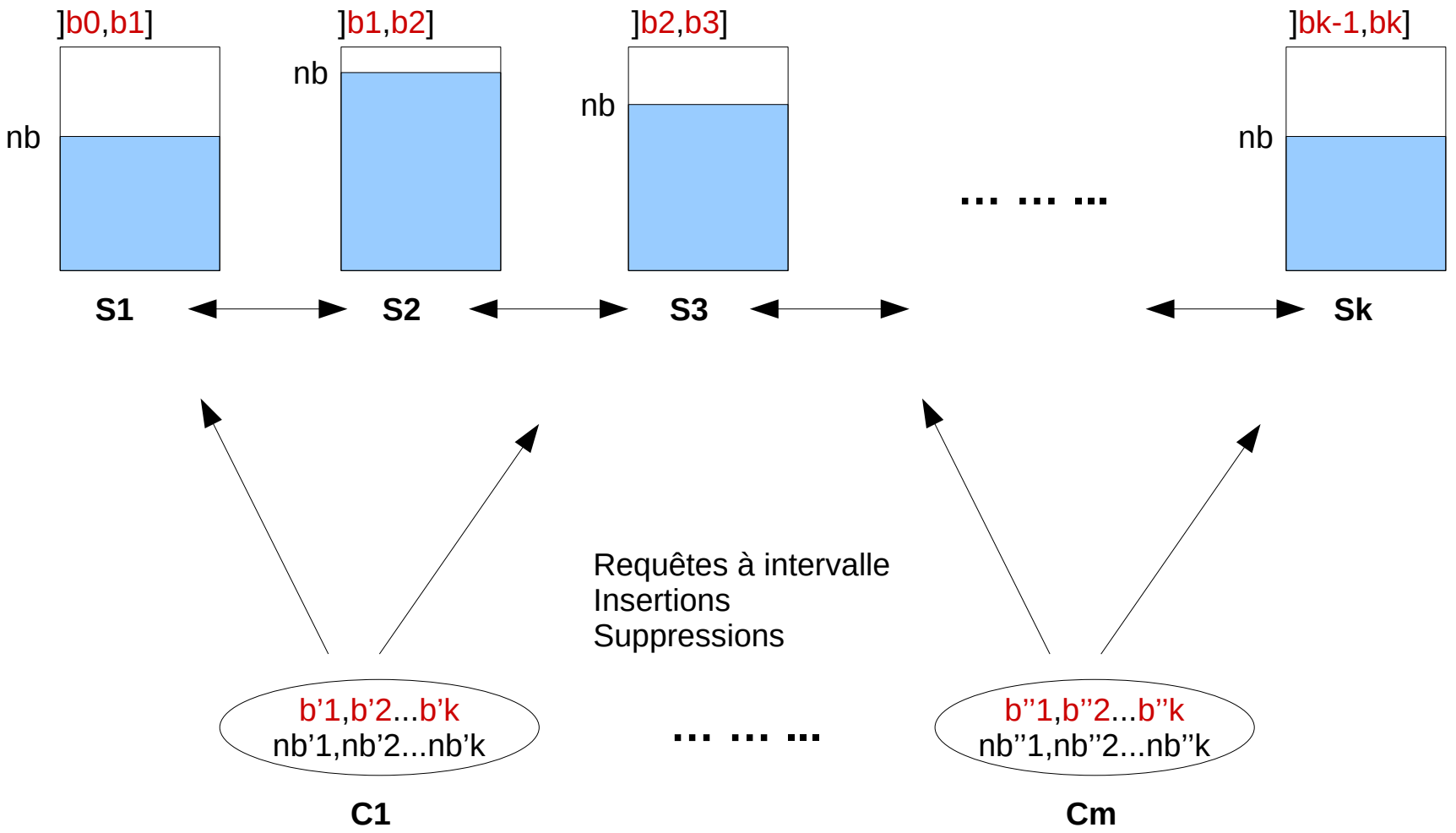
- Le nœud maître fragmente R en k parties : r_1, r_2, … r_k (selon les valeurs d'enreg)
- Le nœud maître fragmente S en k parties : s_1, s_2, … s_k (selon les valeurs d'enreg)
- Chaque couple de parties < r_i, s_i > est envoyée au nœud N_i
- Chaque Nœud N_i effectue la jointure r_i * s_i et renvoie le résultat au nœud maître
- Le nœuds maître concatène les résultats (dans un ordre quelconque)

→ Le découpage par valeur peut se faire dans ce cas soit avec une fragmentation par hachage, soit avec une fragmentation par intervalle

→ risque de déséquilibre de charge causée par les données (data skew)

Exemple d'application
Système de stockage réparti, fragmenté par intervalle à bornes dynamiques

On a k serveurs : S1, S2, … Sk et m clients : C1, C2, … Cm



Chaque client possède une **image** ± correcte de la répartition des données

→ les bornes (bi) et les charges (nb) des serveurs

→ en cas d'erreur d'adressage, le client est partiellement corrigé par le serveur ciblé

Chaque serveur possède aussi une **image** ± correcte de la répartition des données

→ les bornes (bi) et les charges (nb) des autres serveurs

→ utilisée pour des besoins de rééquilibrage asynchrone entre serveur