Gestion des Entrées/Sorties Parallèles dans les grappes SMPs

Adrien Lebre

Directeur de thèse : Brigitte Plateau
Co-encadrant ID : Yves Denneulin
Co-encadrant BULL : Pascale Rosse-Laurent

Laboratoire ID-IMAG (UMR 5132), Grenoble, France.
BULL - HPC, Échirolles, France.

Plan

Contexte

Environnement

- Grappe de SMPs
- Linux
- High Performance Computing
- Applications I/O intensives ⇒ Goulet d'étranglement

Entrées/Sorties parallèles

- Gestion des accès concurrents à un même fichier
- Accès : différences en taille, position. un exemple : la multiplication de matrices

Contexte

Environnement

- Grappe de SMPs
- Linux
- High Performance Computing
- Applications I/O intensives ⇒ Goulet d'étranglement

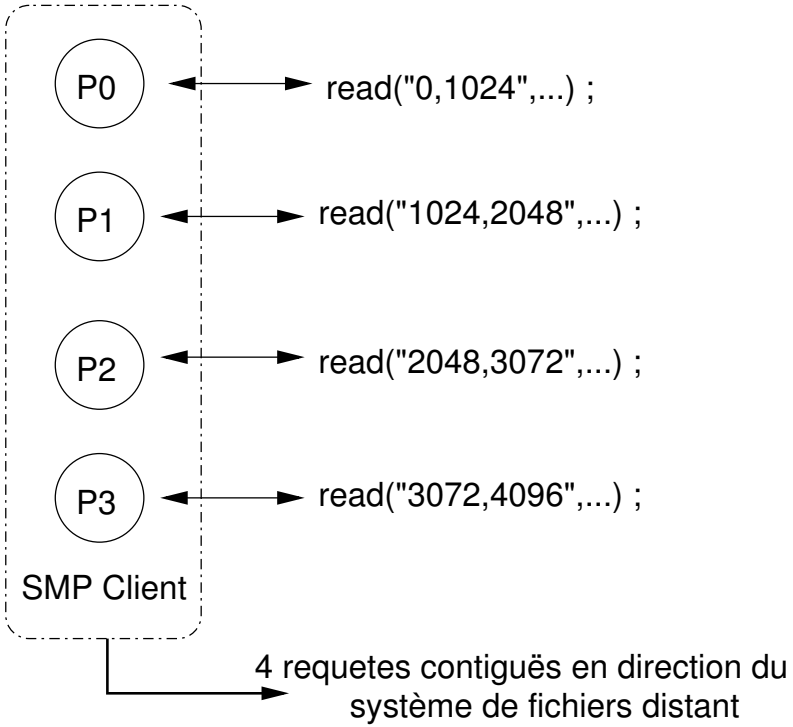
Entrées/Sorties parallèles

- Gestion des accès concurrents à un même fichier
- Accès : différences en taille, position. un exemple : la multiplication de matrices

Entrées/Sorties parallèles - Exemple

Produit de matrices

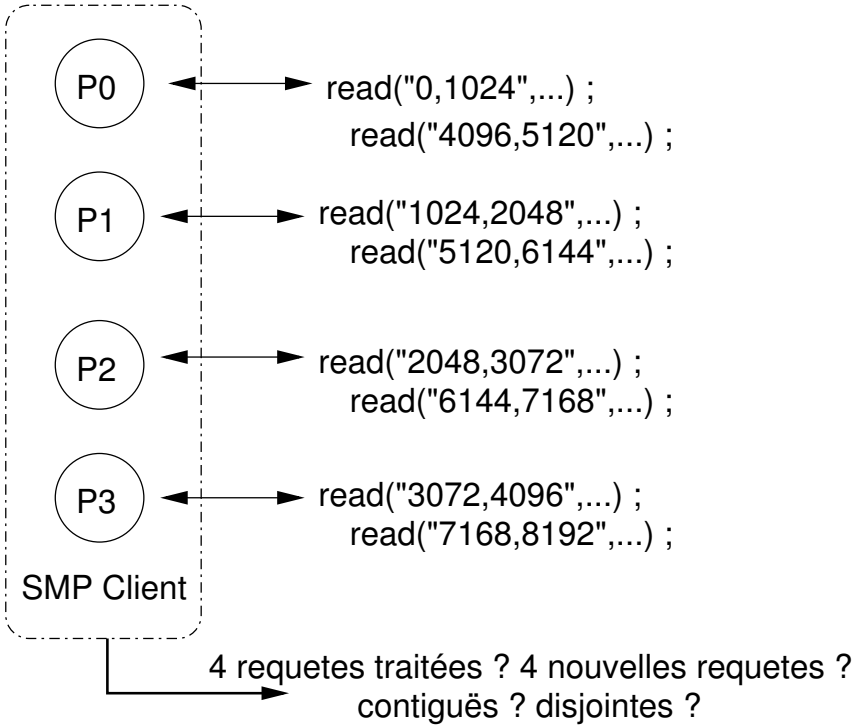
- Parties spécifiques à récupérer (selon le découpage : colonne, ligne, BLOCK/BLOCK, BLOCK/CYCLYC ...)
- Plusieurs requêtes disjointes/contiguës au même instant
- Type de comportement "fatal" pour les performances I/O



Entrées/Sorties parallèles - Exemple

Produit de matrices

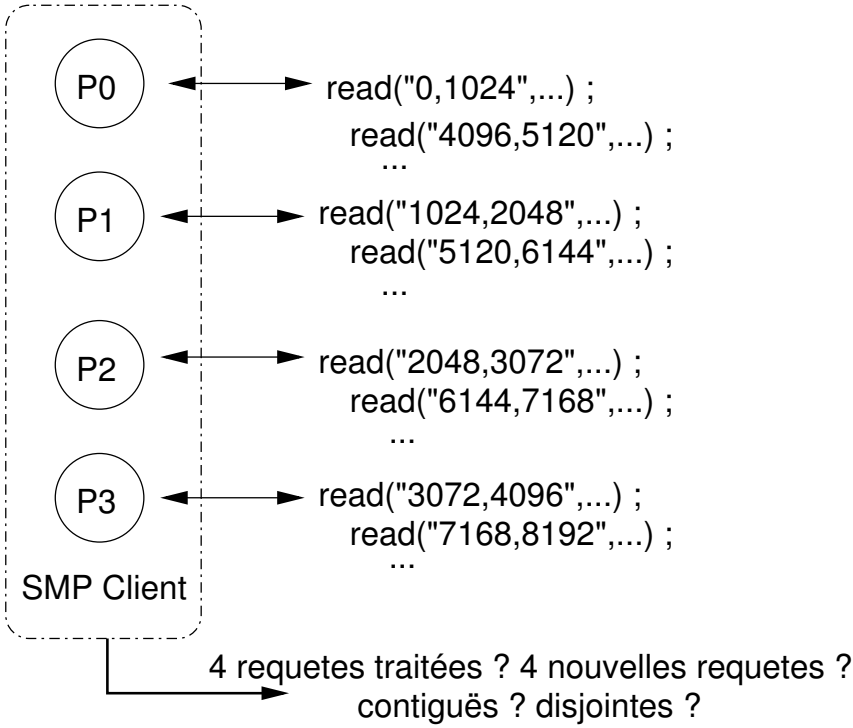
- Parties spécifiques à récupérer (selon le découpage : colonne, ligne, BLOCK/BLOCK, BLOCK/CYCLYC ...)
- Plusieurs requêtes disjointes/contiguës au même instant
- Type de comportement "fatal" pour les performances I/O



Entrées/Sorties parallèles - Exemple

Produit de matrices

- Parties spécifiques à récupérer (selon le découpage : colonne, ligne, BLOCK/BLOCK, BLOCK/CYCLYC ...)
- Plusieurs requêtes disjointes/contiguës au même instant
- Type de comportement "fatal" pour les performances I/O



Entrées/Sorties parallèles

Fonctionnalités requises / contraintes

- Vecteur d'accès (readv) ⇒ API plus complexe
- opération collective ⇒ synchronisation
- Vues logiques (des fichiers) ⇒ placements physiques (blocs disque)

Solutions disponibles

- Systèmes de fichiers parallèles : plus ou moins complets/efficaces
  - "génériques" : PVFS, NFSP, PFS, Lustre
  - spécifiques aux " Parallel I/O" : PIOUS, VESTA ...
- Librairies : maximiser les aspects portabilité
  - Nombreuses : un standard MPI I/O accepté
- API plus sophistiquée ⇒ Dépendance / Coût de développement

Entrées/Sorties parallèles

Fonctionnalités requises / contraintes

- Vecteur d'accès (readv) ⇒ API plus complexe
- opération collective ⇒ synchronisation
- Vues logiques (des fichiers) ⇒ placements physiques (blocs disque)

Solutions disponibles

- Systèmes de fichiers parallèles : plus ou moins complets/efficaces
  - "génériques" : PVFS, NFSP, PFS, Lustre
  - spécifiques aux " Parallel I/O" : PIOUS, VESTA ...
- Librairies : maximiser les aspects portabilité
  - Nombreuses : un standard MPI I/O accepté
- API plus sophistiquée ⇒ Dépendance / Coût de développement

Le système aIOLi

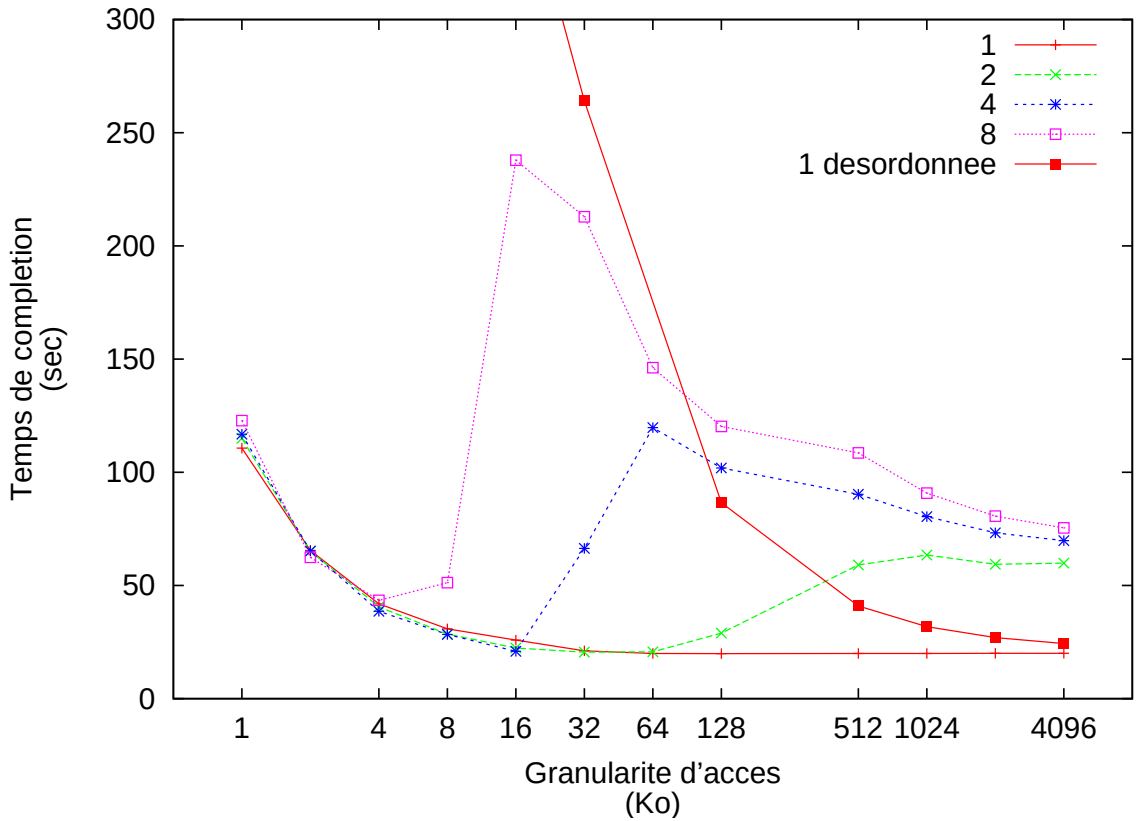
Objectifs

- Intégrer les algorithmes d'Entrées/Sorties parallèles
  - ré-ordonnancement
  - agrégation des accès ⇒ efficacité
  - recouvrement
- Exploiter uniquement les interfaces présentes : POSIX
  - `open/creat/lseek/read/write/close` ⇒ simplicité
- Minimiser les surcoûts de mise en œuvre
  - éliminer au maximum les mécanismes de synchronisation

Évaluations de la pile I/O disque “standard”

Analyse d’une décomposition d’un fichier de 1 Go

(kernel 2.4.27, grappe IDPOT, NFS version 3, mpich 1.2.5)



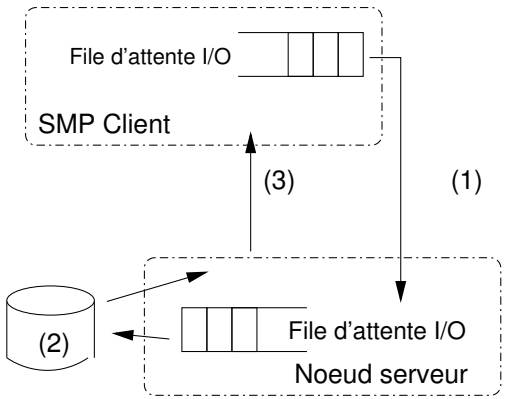
Observations

- 1 processus ⇒ lecture séquentielle (optimale)
- + processus ⇒ - de performance
- 1 processus en accès aléatoire (lecture “sérialisée”) ⇒ + de performance pour les accès larges

Les fondements

Définir une “fenêtre de régulation”

- Maximiser l'utilisation des serveurs de stockages
  - Une requête au moins dans la queue coté serveur
- Appliquer les algorithmes d'Entrées/Sorties parallèles (agrégation/ré-ordonnancement)
  - Juste une requête dans la queue coté serveur !

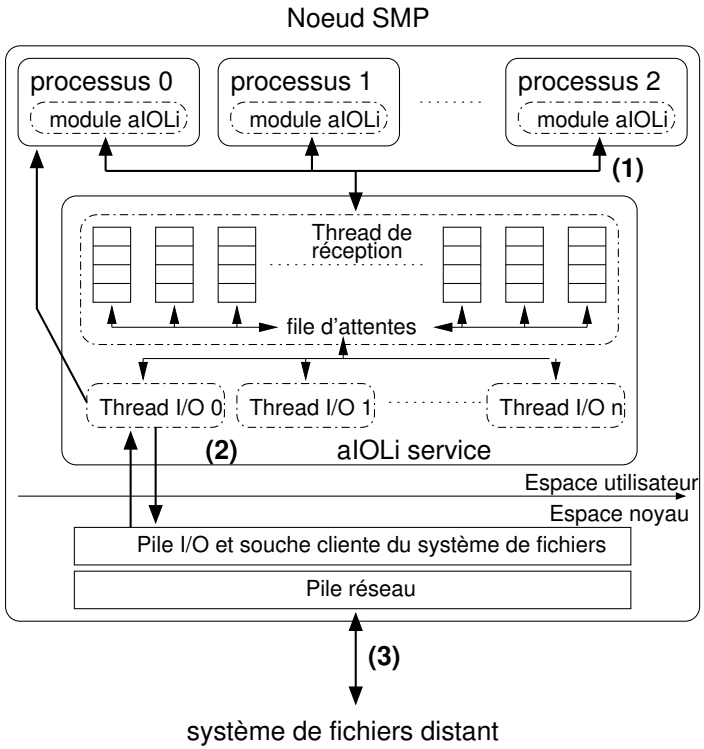


(1) Une première requête est transmise au système de fichiers.
(2) Elle est exécutée sur le périphérique de stockage rattaché.
(3) La réponse est renvoyée au client.

Conception d’un prototype

Librairie en espace utilisateur

- Composant client surchargeant les appels POSIX (lié à l’application)
- Un service aIOLi
  - “Multi-threaded”
  - Réalise les différentes optimisations
  - Exécute les Entrées/Sorties effectives
- Mécanismes IPC et mémoire partagée

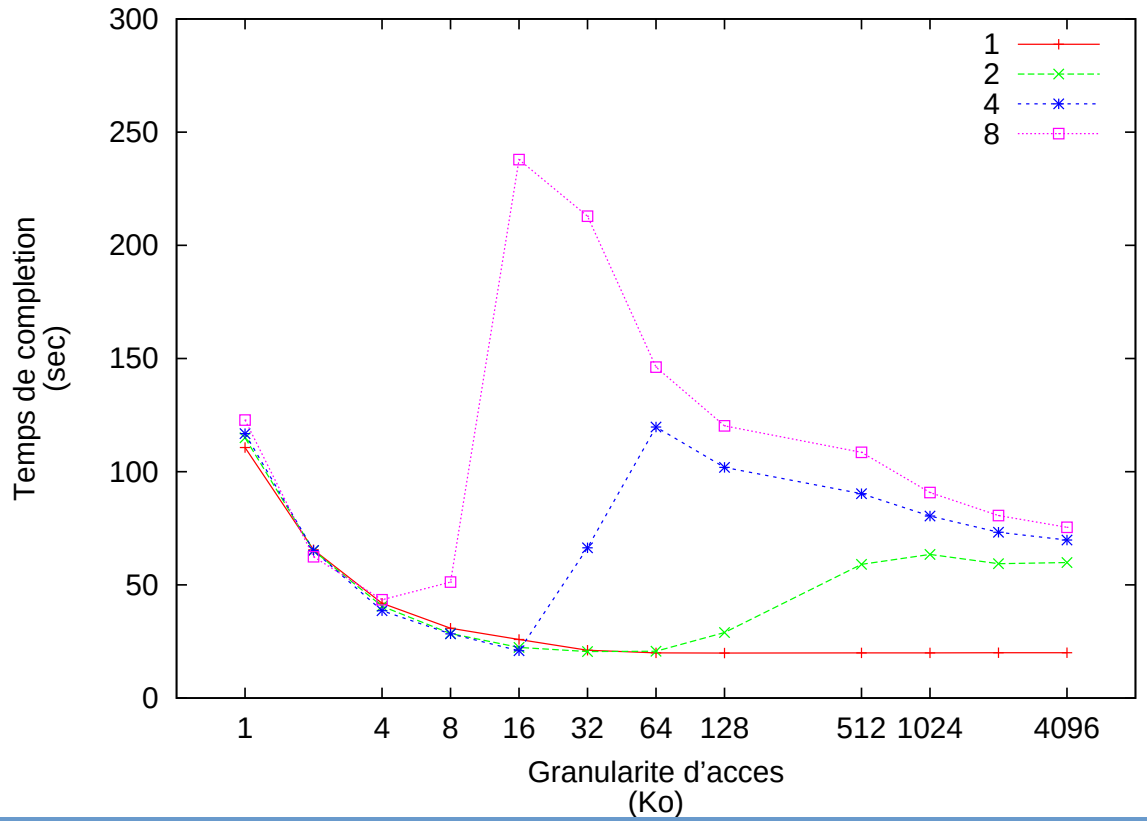


Évaluations : Décomposition aIOLi

Analyse d'une décomposition d'un fichier de 1 Go

(kernel 2.4.27, grappe IDPOT, NFS version 3, mpich 1.2.5)

compilation sans aIOLi

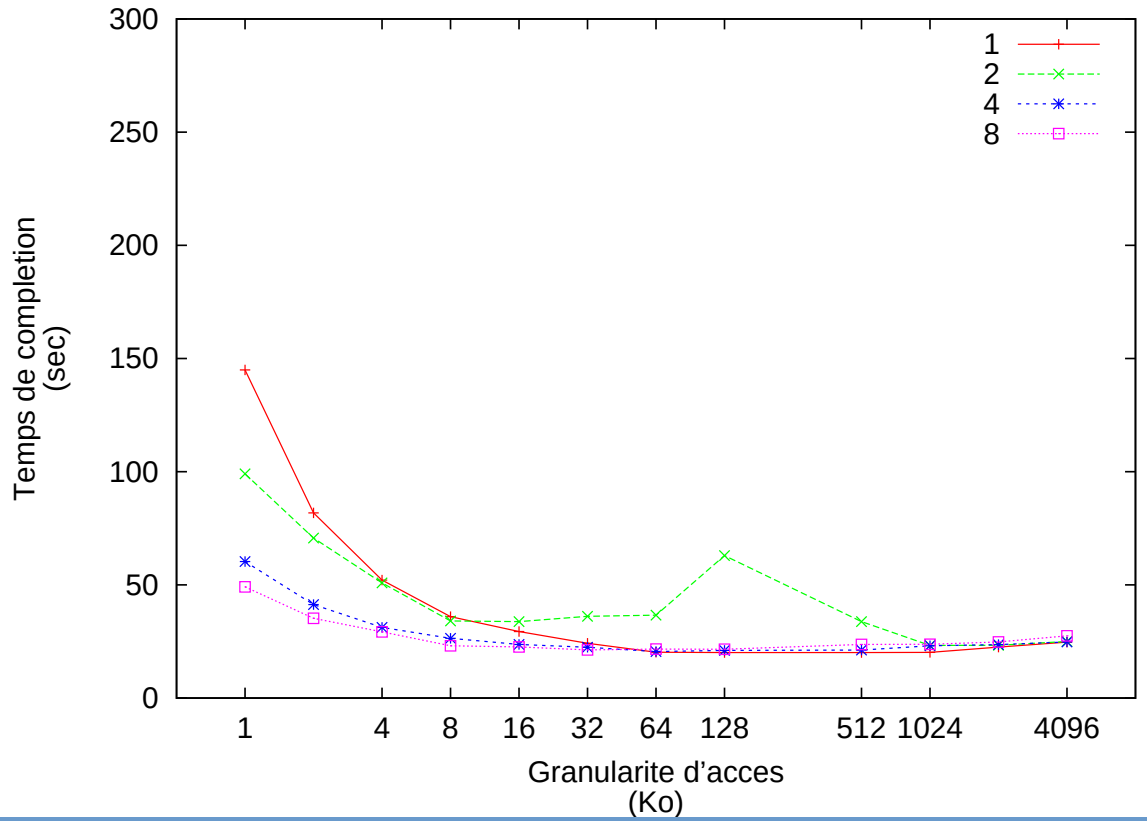


Évaluations : Décomposition aIOLi

Analyse d'une décomposition d'un fichier de 1 Go

(kernel 2.4.27, grappe IDPOT, NFS version 3, mpich 1.2.5)

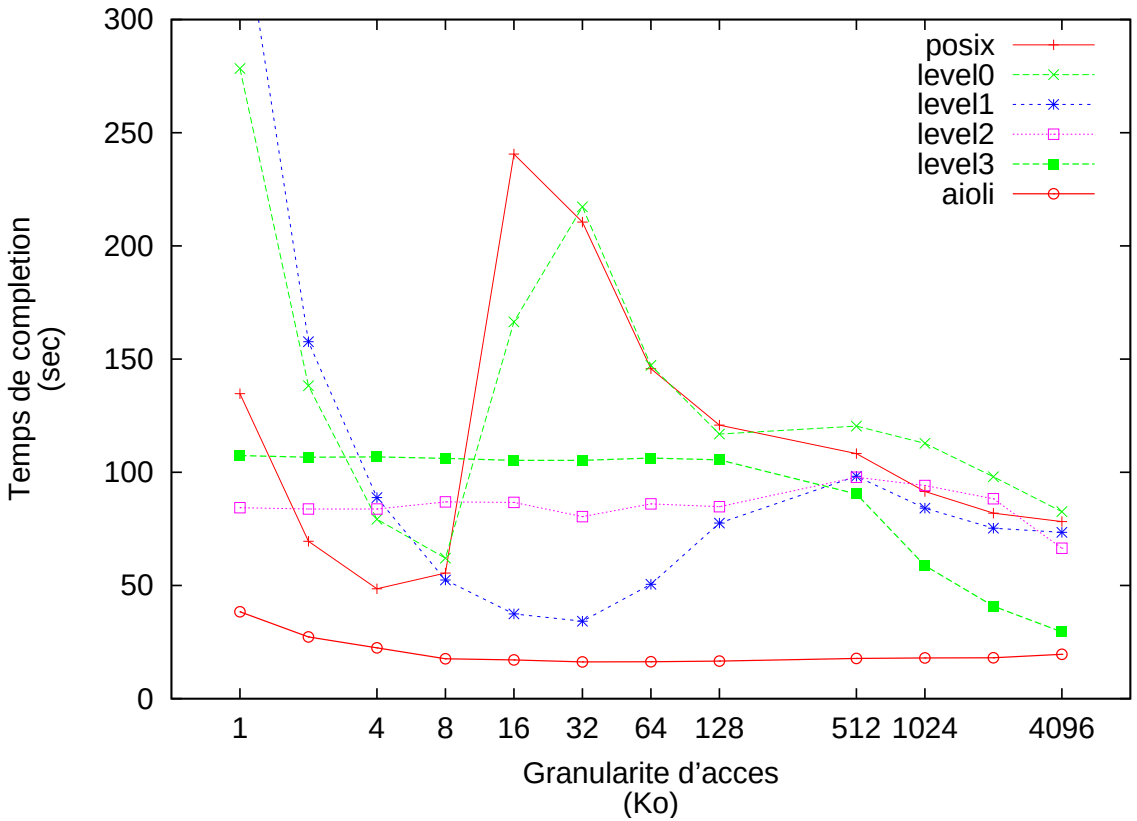
recompilée avec aIOLi



Évaluations : MPI I/O vs aIOLi

décomposition d'un fichier de 1 Go incluant 8 processus

(kernel 2.4.27, grappe IDPOT, NFS version 3, mpich 1.2.5, ROMIO)



Observations

- MPI I/O : fournir de "l'information" sur les accès
- Quelque soit le niveau, aIOLi + performant

Conclusion

Bilan positif

- Efficace, résultats encourageant
- Simplicité du modèle / API standard
- Pas synchronisation intra-processus

Contraintes actuelles

- Gestion du placement physique
- Surcoût sur accès standard
- Dépendant de l'ordonnanceur noyau

Travaux futures

- Ajout des considérations "data stripping"
- Portage en module noyau, été 2005 (stage de 3 mois)
- Analyse sous SMP 16 voies et Lustre
- Coordination entre plusieurs SMPs, en cours (Master).
- Vers les grilles ...

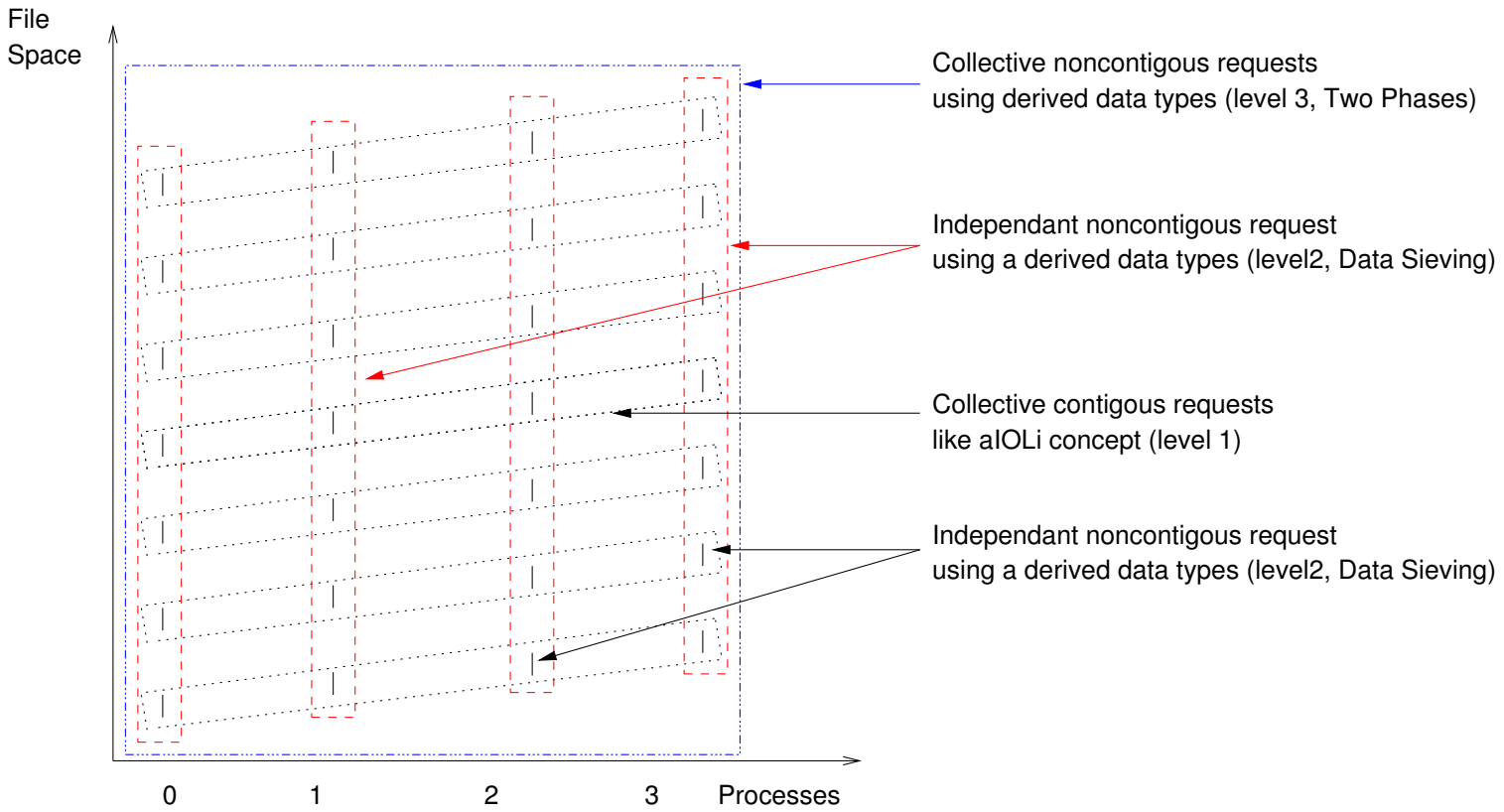
Questions ?

http ://aioli.imag.fr

Projet LIPS
BULL - INRIA - Laboratoire ID

Merci de votre attention

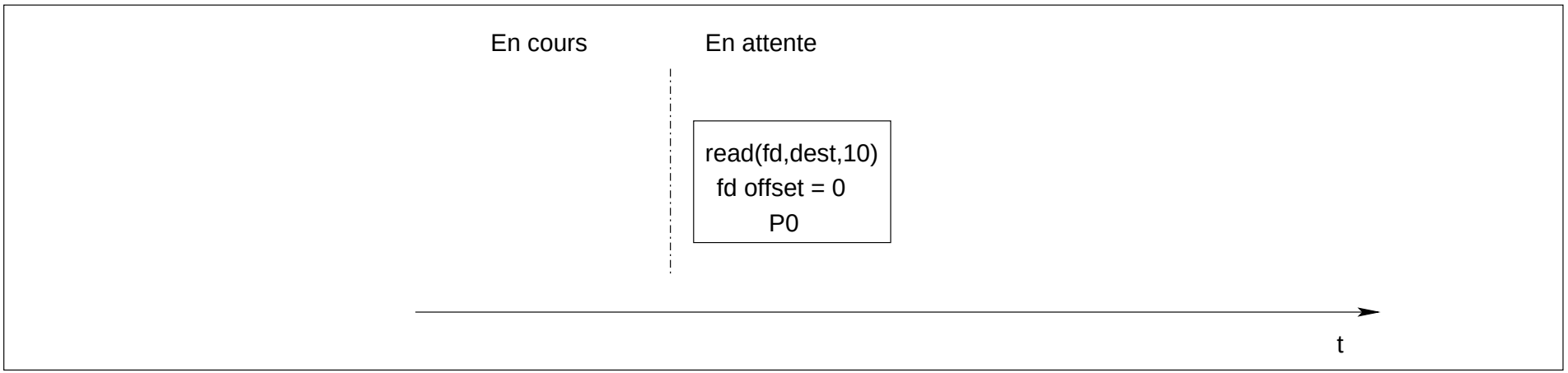
Les optimisations dans MPI I/O



MPI I/O – Four levels
Representing increasing amounts of data per request
[Thakhur/Gropp/Lusk02]

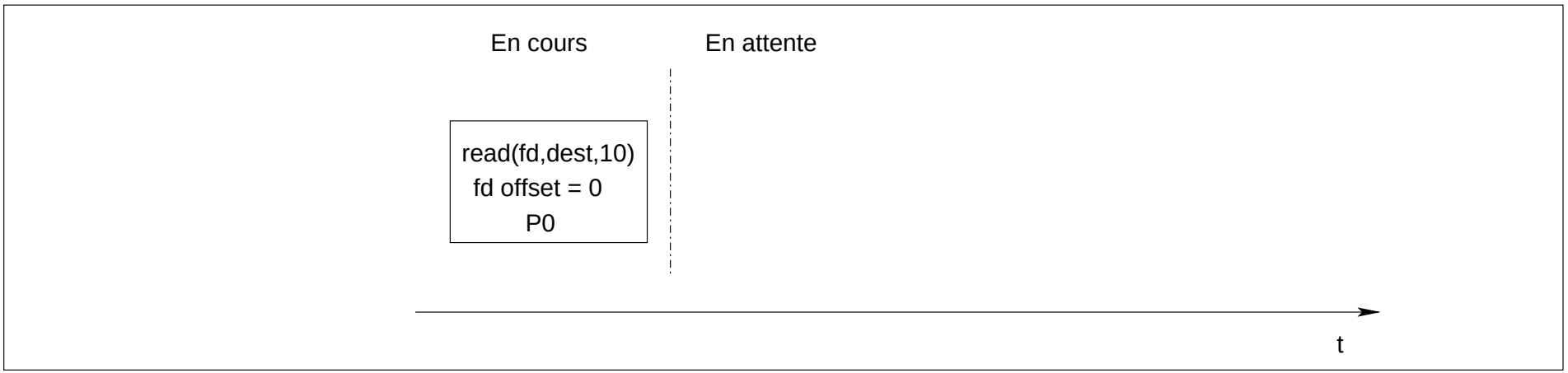
Agrégation et découverte de schéma d'accès

Cas d'école : Décomposition sur 4 processus (grain = 10 octets)



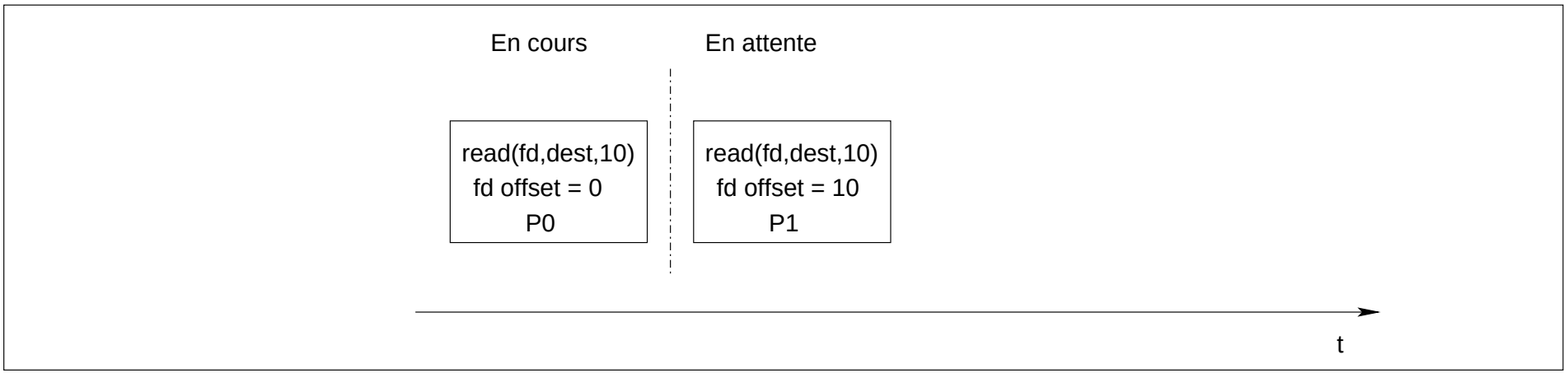
Agrégation et découverte de schéma d'accès

Cas d'école : Décomposition sur 4 processus (grain = 10 octets)



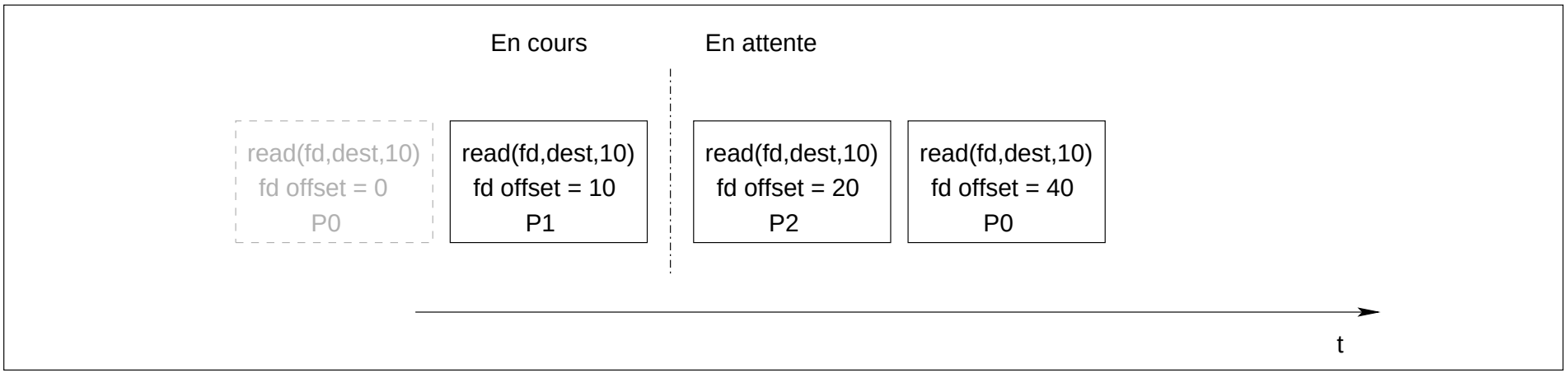
Agrégation et découverte de schéma d'accès

Cas d'école : Décomposition sur 4 processus (grain = 10 octets)



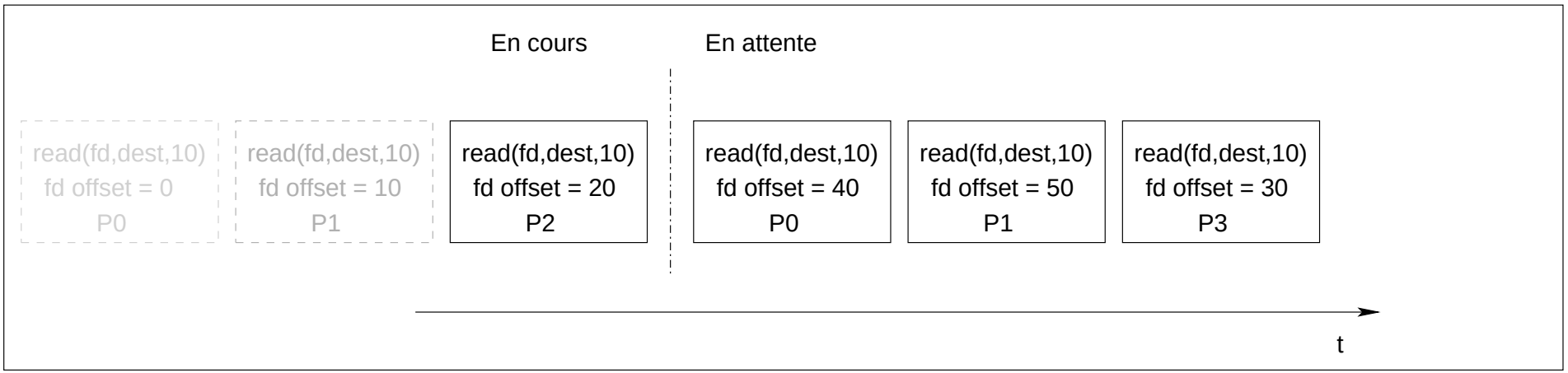
Agrégation et découverte de schéma d'accès

Cas d'école : Décomposition sur 4 processus (grain = 10 octets)



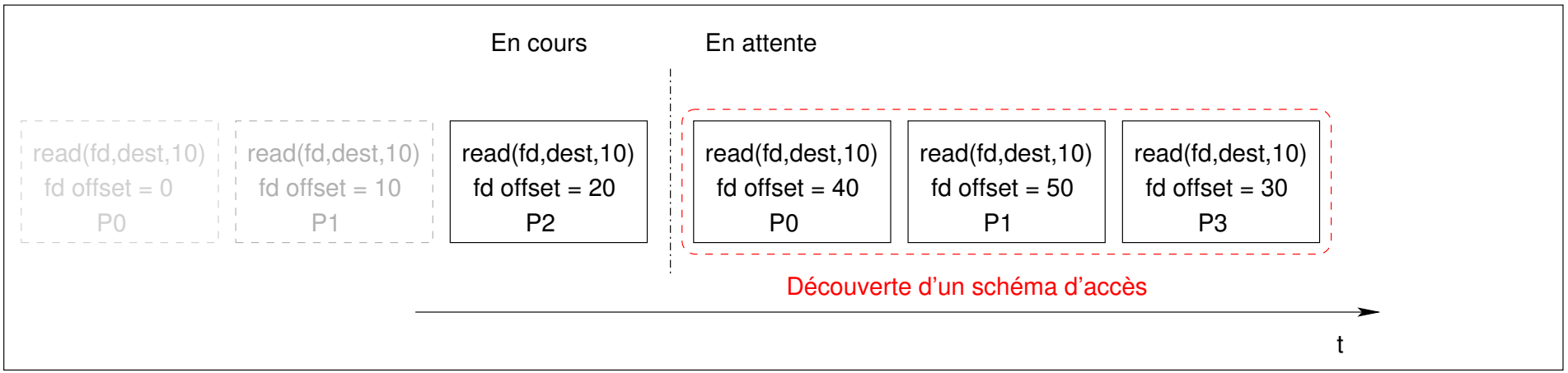
Agrégation et découverte de schéma d'accès

Cas d'école : Décomposition sur 4 processus (grain = 10 octets)



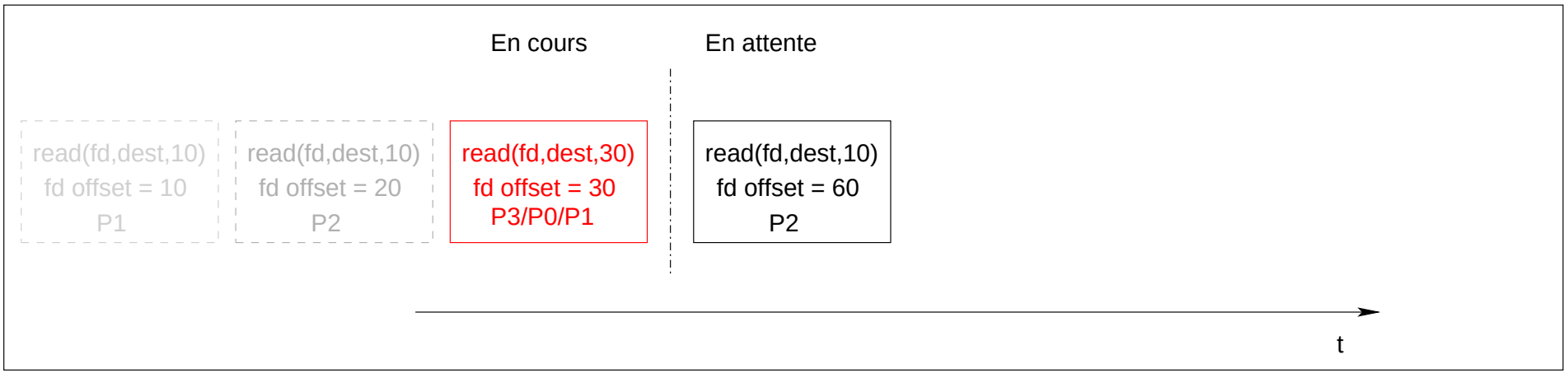
Agrégation et découverte de schéma d'accès

Cas d'école : Décomposition sur 4 processus (grain = 10 octets)



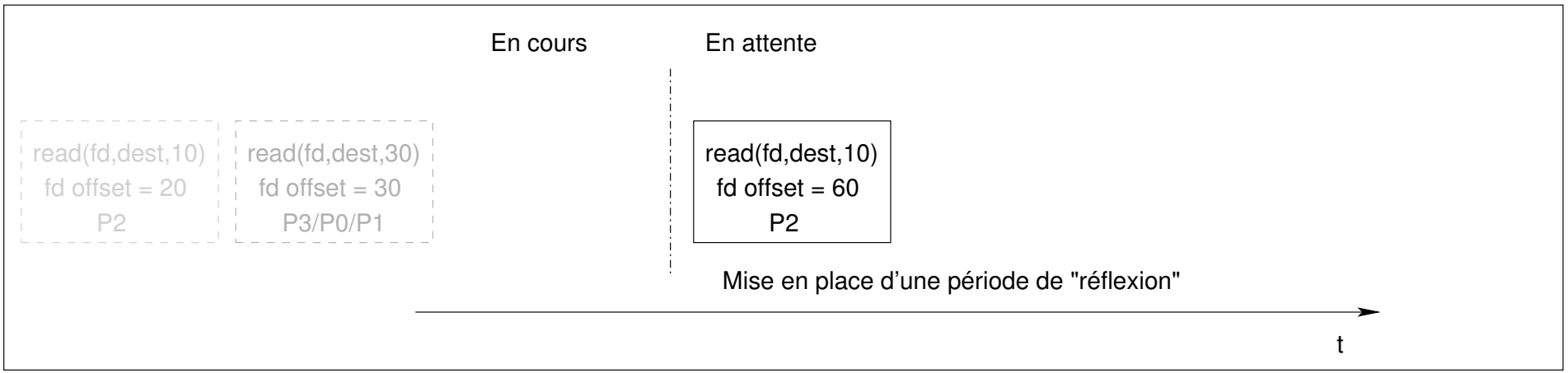
Agrégation et découverte de schéma d'accès

Cas d'école : Décomposition sur 4 processus (grain = 10 octets)



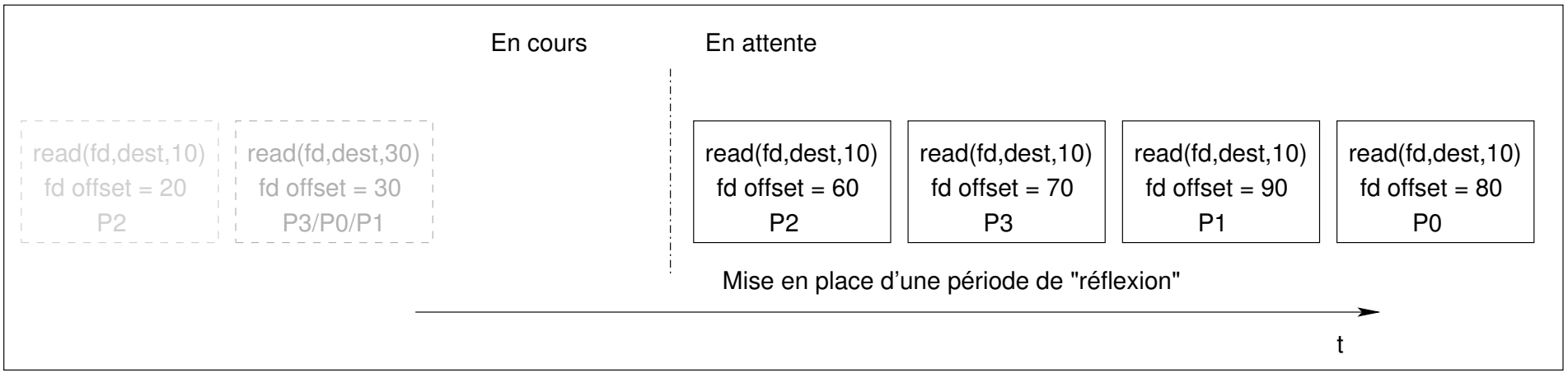
Agrégation et découverte de schéma d'accès

Cas d'école : Décomposition sur 4 processus (grain = 10 octets)



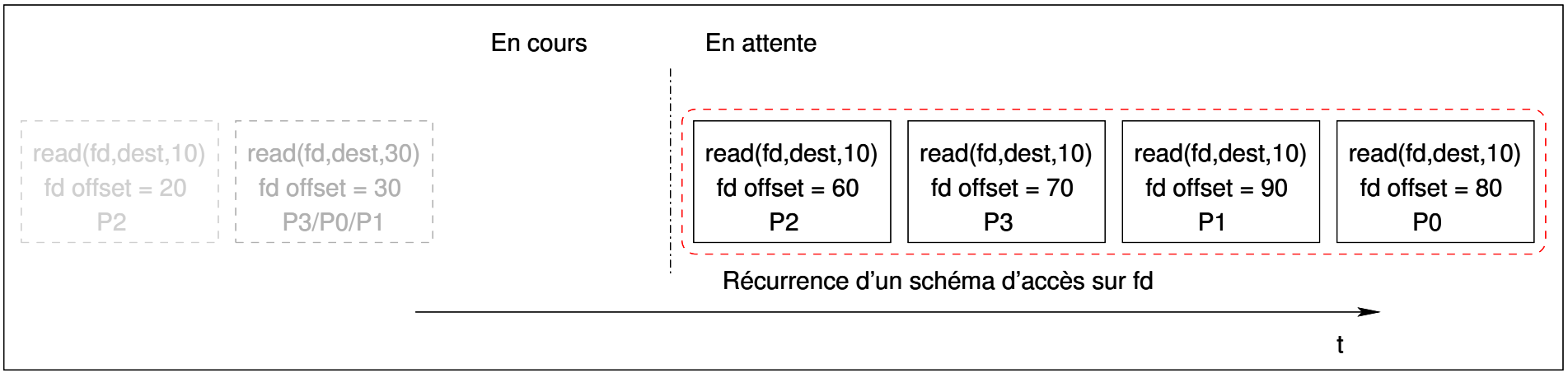
Agrégation et découverte de schéma d'accès

Cas d'école : Décomposition sur 4 processus (grain = 10 octets)



Agrégation et découverte de schéma d'accès

Cas d'école : Décomposition sur 4 processus (grain = 10 octets)



Agrégation et découverte de schéma d'accès

Cas d'école : Décomposition sur 4 processus (grain = 10 octets)