# Haute performance et Entrées/Sorties au sein des applications parallèles

**Adrien Lebre**

4 Mars 2004
(révision du 29 Janvier 2004)

Directeur de thèse : Brigitte Plateau
Co-encadrant : Yves Denneulin

Bull

Projet Lips – Pascale Rossé

Laboratoire Informatique et Distribuer | CNRS Centre National de la Recherche Scientifique | Institut National Polytechnique de Grenoble | INRIA | Université Joseph Fourier

## Contexte
### Super-calculateurs et HPC

- Super-calculateurs
- Grappes
  - Grappe « Beowulf » :
    - icluster
  - Grappes de super-calculateurs :
    - icluster 2
- Architecture 64 bits

Puissance de calcul ↗

## Contexte
### HPC et Entrées/Sorties

- Applications scientifiques exploitent la puissance de calcul disponible
  - Climatologie, génomique, imagerie ...
  - Les données intrinsèques à l'application ↗
    - *Ecart considérable entre puissance CPU et Débit I/O.*

*[amdahl67] gain maximal dans un processus de parallélisation est limité par le sous système le plus lent.*
*machine équilibrée : 1Mo RAM, 1MIPS, 1Mb/s I/O 2GHz /2Gb/s*
*[Hennessy96] iconfirmation : I/O 10%/an vs CPU 40%/an*

- *Exploitation intensive (voir saturée) du système de données*



## Problématique
### Parallélisme et Entrées/Sorties

- *Performance d'un disque : 500Mb/s, latence élevée*
- *Quadrics, Myrinet : Débit élevé, latence faible*

⇒ *Comportement de type goulet d'étranglement.*

- *Les solutions*
  - *Dédiées (SAN, RAID) : coûteuses, complexes et bornées*
  - *Optimisations logicielles (placement intelligent, système de fichiers « modernes », bibliothèques) – qu'en est il ?*



## Problématique
### Optimisation des Entrées/Sorties // (1)

- *Analyse du code - phase pré-compilatoire*
  - *Construction statique du graphe de flots de données*
  - *Fonctionne dans des cas simples*
    - *Irrégularité, dépendances des instructions et des données ...*
  - *Ne répond pas à la problématique définie par BULL*
- *« Intergiciel » - systèmes de fichiers //*
  - *Parallélisation des requêtes (PVFS, NFSp, PFS ...)*
    - *+ ou – performant, + ou – complexe*
    - *API + ou – spécifique Portabilité ?*



## Problématique
### Optimisation des Entrées/Sorties // (2)

- *Bibliothèques – optimisation pendant l'exécution*
  - *Complète l'interface standard (open/read/write/close) d'UNIX inadaptée, API spécifique Standard ?*
  - *« Abstraction » de haut niveau basée sur des systèmes sous-jacents parfois spécifiques (PASSION, PANDA ...)*

⇒ *dépendance architecturelle moins forte mais toujours présente*

- *Cohésion des interfaces : MPI I/O (1997)*

*Comparaison échanges de messages / accès I/O*
*2004 : acceptation du standard ? implémentation ? efficacité ?*

## Plan

- Introduction
  - Contexte – cluster et HPC
  - Problématique – les I/O parallèles – MPI I/O
- Caractérisation des comportements
- Techniques d'optimisation
- MPI I/O au sein de la pile I/O
- Bilan et perspectives



## Plan

- Introduction
- Caractérisation des comportements « I/O »
  - Notions élémentaires
  - Classification des applications I/O intensives
  - Caractérisation des accès
- Techniques d'optimisation
- MPI I/O au sein de la pile I/O
- Bilan et perspectives



## Caractérisation des comportements
### Notions élémentaires

- Terminologie spécifique à la gestion et au partage de données
  - Transparence de localisation, sémantique de partage, granularité, faux-partage.
  - Réactivité : temps requis afin de satisfaire une opération

(latence bus/net/hdd + analyse requête + formulation réponse)

- Mode d'accès: Lecture / Ecriture / Lecture / Ecriture — Le mode influe sur le coût du partage



## Caractérisation des comportements
### Classification des applications (1)

- 3 principales catégories :
  - Compulsive : accès conséquent intervenant au même moment : Lecture en début / écriture en fin d'exécution

Difficulté de rendre transparentes les opérations I/O puisque les données sont des paramètres prépondérants de l'application.

*Agrégation des requêtes*

  - Contrôle : opérations sur de faibles quantités se produisant tout au long de l'exécution

Fichiers « checkpoint », fichiers temporaires

*Asynchronisme, cache*



## Caractérisation des comportements
### Classification des applications (2)

- Sur dimensionnée : « Out-of-core », les données nécessaires à l'application sont trop volumineuses pour être stockées en mémoire.

*Mémoire virtuelle (« swap »),*

*Solution temporaire : augmentation de la RAM*

*Pagination optimisée de la mémoire, cache adaptable*

*Classification établie dans le cadre de l'initiative « Scalable I/O » en 1995 [Crandall95].*



## Caractérisation des comportements
### Caractérisation des accès (1)

- Peu de fichiers ouverts en lecture/écriture
- Opérations de lecture largement majoritaires
  - 90% des appels récupèrent moins de 100 octets

*Concentrer ses efforts sur ce type d'accès.*

- *Ecritures principalement locales et indépendantes*
- *Mais, centralisation parfois requise (cohérence)*

Spécification du mode d'accès par un paramètre supplémentaire : Descripteur individuel vs partagé

*Accès à l'échelle d'un noeud, d'un groupe ...*

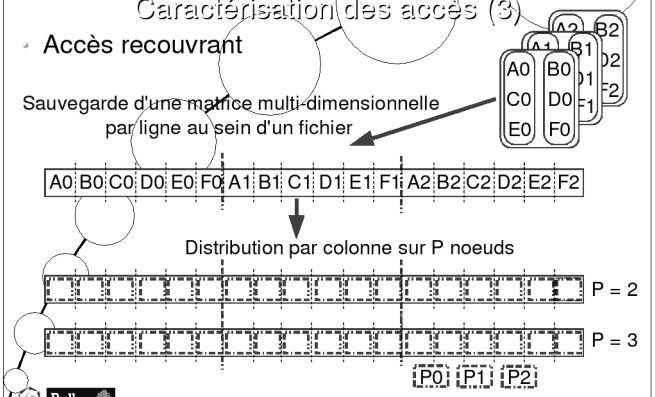
## Caractérisation des comportements
### Caractérisation des accès (2)

- Accès séquentiel
  - La requête débute à un « offset » supérieur à celui de la précédente
- Accès consécutif
  - La requête débute à l'offset où la dernière s'est terminée (principalement lors des écritures)
- Accès recouvrant
  - Données contigües accédées par plusieurs processus *séquentiel du point de vue d'un processus, consécutif du point de vue de l'application.*



## Caractérisation des comportements
### Caractérisation des accès (3)

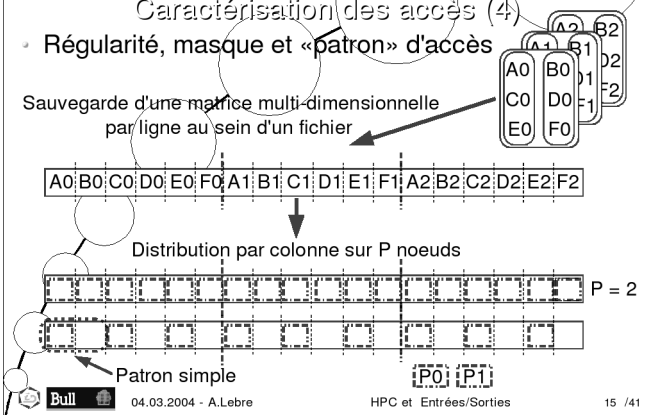
- Accès recouvrant






## Caractérisation des comportements
### Caractérisation des accès (4)

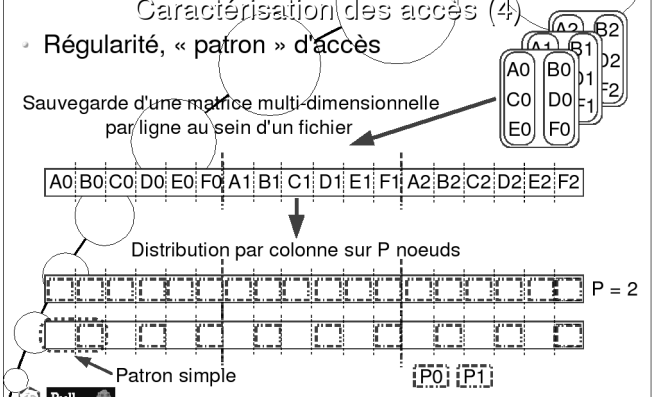
- Régularité, masque et «patron» d'accès






## Caractérisation des comportements
### Caractérisation des accès (4)

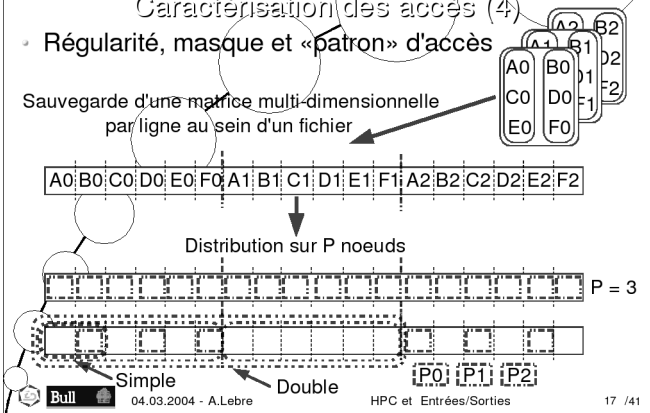
- Régularité, « patron » d'accès






## Caractérisation des comportements
### Caractérisation des accès (4)

- Régularité, masque et «patron» d'accès






## Plan

- Introduction
- Caractérisation des comportements « I/O »
  - Notions élémentaires – Corpus défini : réactivité
  - Classification – compulsive, contrôle, « out-of-core »
  - Caractérisation des accès – de faible taille, régularité
- Techniques d'optimisation
- MPI I/O au sein de la pile I/O
- Bilan et perspectives

## Plan

- Introduction
- Caractérisation des comportements « I/O »
- Techniques d'optimisation
  - Agrégation des requêtes
  - Cache et asynchronisme
- MPI I/O au sein de la pile I/O
- Bilan et perspectives



## Techniques d'optimisation
## Agrégation des requêtes (1)

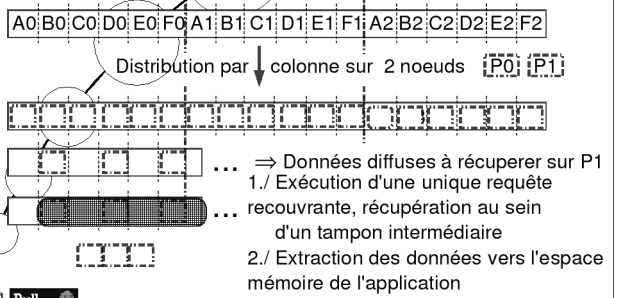
- But : Diminuer le nombre d'appels
  - *Réduire les phénomènes de goulet d'étranglement sur les unités de sauvegarde*
  - *Améliorer le temps de réactivité*
- *3 techniques :*
  - *A l'échelle d'un noeud – « Data sieving »*
  - *Approche collective, à plusieurs niveaux : disque, serveur, client : « Two phases »*
  - *« Stream based I/O » - minimise les flux réseaux*



## Techniques d'optimisation
## Agrégation des requêtes (2)

- « Data sieving », un noeud

Sauvegarde d'une matrice multi-dimensionnelle par ligne




⇒ Données diffuses à récuperer sur P1

1./ Exécution d'une unique requête recouvrante, récupération au sein d'un tampon intermédiaire

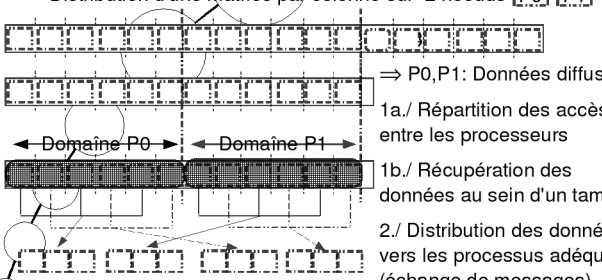
2./ Extraction des données vers l'espace mémoire de l'application



## Techniques d'optimisation
## Agrégation des requêtes (3)

- « Two phases » - approche collective

Distribution d'une matrice par colonne sur 2 noeuds P0 P1




⇒ P0,P1: Données diffuses

1a./ Répartition des accès entre les processeurs

1b./ Récupération des données au sein d'un tampon

2./ Distribution des données vers les processus adéquats (échange de messages)



## Techniques d'optimisation
## Agrégation des requêtes (4)

- *Approches présentées*
  - *Nécessité d'un « masque » de fichier*
  - *Recopies entre les différents tampons coûteuses*
  - *Transfert de données inutiles sur les divers supports*
  - *Définition du degré de continuité [Raj02]*
- *« Stream based I/O »*
  - *Protocole intégré dans PVFS*
  - *Une unique requête encapsule plusieurs appels I/O*
  - *Surcoût engendré faible, mais spécifique Portabilité ?*



## Techniques d'optimisation
## Agrégation des requêtes (5)

- Enseignements :
  - « Data sieving » une approche validée en tenant compte
    - Du degré de continuité des données
    - De la correspondance avec la vue physique
  - Approche collective
    - Minimise les transferts disques ↔ mémoire
    - Coûts parfois important (échanges de messages, points de synchronisation, ...)
  - « Stream based I/O » trop corrélée à l'architecture

⇒ Vers un « service » intermédiaire de haut niveau

## Techniques d'optimisation
## Cache (1)

- *Exploitation des espaces de stockage locaux*
  - *Limite les phénomènes de saturation I/O*
  - *Concept éprouvé*
- *Nombreux algorithmes et techniques*
  - *Lecture avancée, écriture retardée ...*
- *Cache client vs cache global vs cache hiérarchique*

*multiples niveaux de « buffer » cache au sein d'une architecture*
*Répercussion d'un défaut de page sur tous les niveaux*
*«Contrôler» plus finement les caches ? [Vilayannur03]*



## Techniques d'optimisation
## Cache (2)

- Technique de cache « basique » et I/O //
  - Systèmes de fichiers // fournissent les couches basiques
  - Mais, peuvent se révéler inefficace
    - Exemple : Lecture avancée et masque double ►
  - Bibliothèques I/O // s'appuient sur les systèmes de fichiers ⇒ Généralement pas de gestion de cache
- Interaction entre l'interface de haut niveau et les gestionnaires de caches ⇒ Impératif (masque)?



## Techniques d'optimisation
## Asynchronisme (1)

- *But : « Eliminer » totalement les latences I/O en tirant partie de processus d'arrière plan.*

*Maximiser le recouvrement I/O / Calcul*

- *Technique intégrée de manière croissante dans plusieurs solutions*
  - *Au sein même des unités de stockage*
  - *Systèmes de fichiers (PVFS 2, Lustre, ...)*
  - *Bibliothèques (MTIO, MercutIO, ROMIO, ...)*

*Peu d'applications développées pour exploiter ce potentiel*



## Techniques d'optimisation
## Asynchronisme (2)

- *2 politiques : explicite vs implicite*
  - *Explicite*
    - *Mise en oeuvre « controlée »*
    - *↗ la charge de travail pour le développeur*
  - *Implicite*
    - *Transparence pour le développeur*
    - *Comportement néfaste dans certains cas (mise en concurrence des tâches)*

*Implémentation judicieuse du sous-module de gestion des I/O en prenant compte du degré de continuité*



## Techniques d'optimisation
## Cache et Asynchronisme

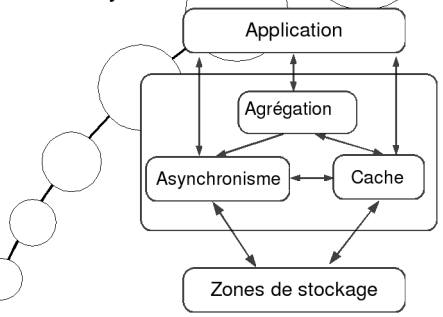
- Enseignements :
  - Cache
    - Prendre en compte les informations fournies par les abstractions de haut niveau
    - Paramétrable avec politique variable (choix, ...)
  - Asynchronisme
    - Fournir des routines explicites ⇒ contrôle « fin »
    - Déclenchement transparent en arrière plan si susceptible d'apporter un gain

⇒ Un service intermédiaire proposant ces fonctionnalités
Confirmation : faut il un intergiciel spécifique aux I/O // ?



## Techniques d'optimisation
## Bilan

- Un embryon d'architecture



## Plan

- Introduction
- Caractérisation des comportements « I/O »
- Techniques d'optimisation
  - Agrégation des requêtes
  - Cache et asynchronisme

ϖ Un embryon d'entité unique

- MPI I/O au sein de la pile I/O
- Bilan et perspectives



## Plan

- Introduction
- Caractérisation des comportements « I/O »
- Techniques d'optimisation
- MPI I/O au sein de la pile I/O
  - Les travaux actuels
  - MPI I/O, Modèle et implantation
- Bilan et perspectives



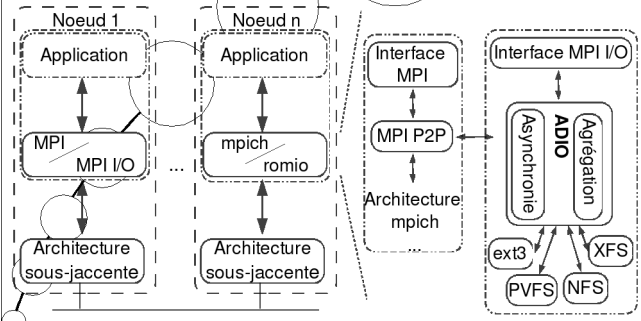
## MPI I/O au sein de la pile I/O
### Travaux actuels

- *Analyse de la pile I/O*
  - *micro-contrôlleur, peu exploitable, environnement fermé*
  - *Agrégation des unités de stockage via réseau : validée*
  - *Exploitation des unités de stockage*
    - *Systèmes de fichiers // : NFSp, PVFS, Lustre, PFS...*
    - *Longuement analysés*
    - *Accès distant : READ 2 [Cos03], Quadrics Direct I/O*
    - *Efficace oui mais localisation ? Solution potentielle ?*
  - *Bibliothèques de haut niveau : MPI I/O est un standard*
- *Nombreux travaux d'optimisations internes à la machinerie MPI*



## MPI I/O au sein de la pile I/O
### Modèles et implantation

- MPI I/O est construit en s'appuyant sur MPI





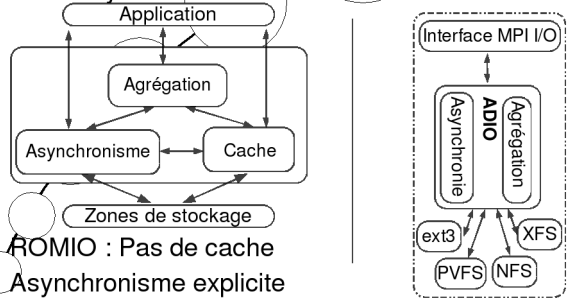

## MPI I/O au sein de la pile I/O
### Modèles et implantation (1)

- *Plusieurs implantations MPI I/O*
- *MercutIO [Rajaram02]*
  - *Totalement asynchrone*
  - *Portable mais depuis peu propriétaire (ChaMPIon)*
- *Implantations spécifiques : GPFS, HPSS*
- *ROMIO*
  - *La + déployée (mpich, lam/mpi, ....)*
  - *Base de travail*



## MPI I/O au sein de la pile I/O
### Modèles et implantation (2)

- « Embryon d'architecture » vs ROMIO




- ROMIO : Pas de cache
- Asynchronisme explicite

⇒ Compléter l'ADIO

## Plan

- Introduction
- Caractérisation des comportements « I/O »
- Techniques d'optimisation
- MPI I/O au sein de la pile I/O
  - les travaux actuels - Longuement abordée, Direct I/O
  - MPI I/O, Modèle et implantation - Validé, ROMIO
- Bilan et perspectives



## Bilan et perspectives (1)

- Systèmes de fichiers intègrent beaucoup de fonctionnalités
  - Ajouter un/des modules(s) spécifique(s) pour la gestion des I/O parallèles augmente leur complexité (mise en oeuvre, maintien, coût ...)

*Faut il un unique système complet pour la gestion des données durant l'exécution d'une d'application HPC sur une grappe ?*

*Le rapatriement efficace des données sur des zones de stockage fiable : une autre problématique ? (gxfer, gridFTP, « HPSS movers » ...)*



## Bilan et perspectives (2)

- Vers une solution plus souple et moins lourde

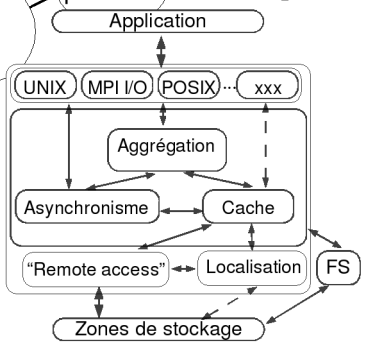
Portable, utilise l'ADIO

Client / Serveur ?
Distribué ?
⇒ Compromis : hierarchie

Accès distants et localisation ?

Architecture ouverte

Read 2 (« myrinet »)
Direct I/O (« Elan kernel »)






## Bilan et perspectives (3)

- *Meilleure gestion des accès sur les données*
  *Besoins matérielles ↘ / Coût ↘*
- *Oui, mais les « méta-informations » ?*
  - *Problématique interne ou externe à celle des I/O //.*
  - *Des techniques pour diminuer le facteur de réactivité des méta-opérations*
    - *Matérielle : parallèlisation, équilibrage*
    - *Logicielle : « Ex-Hash » pour les répertoires (GPFS), exploitation de BD, création de protocole dédié ?*



## Questions ?

- « A supercomputer is a device for converting a CPU bound problem into an I/O bound problem »
  K. Batcher

## Plus d'infos...

- http://www-id.imag.fr/~lebre/