Expériences d'entrées-sorties sur machines massivement parallèles

Philippe.Wautelet@idris.fr

CNRS** - **IDRIS

Ecole « Optimisation »
Maison de la Simulation / 7 au 11 octobre 2013

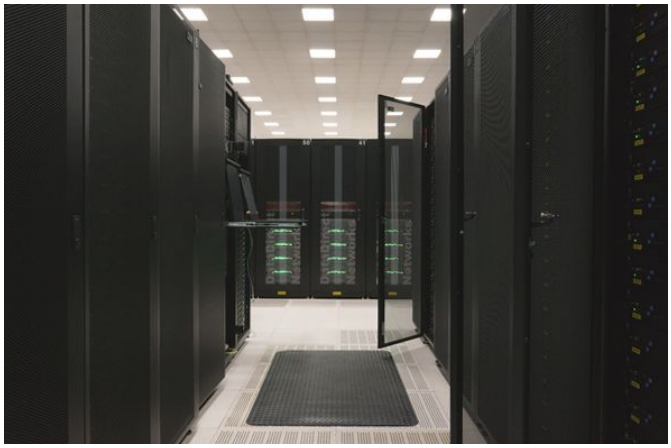
Sommaire

Tests de performances

Configuration IDRIS

Turing : IBM Blue Gene/Q

Ada : IBM x3750

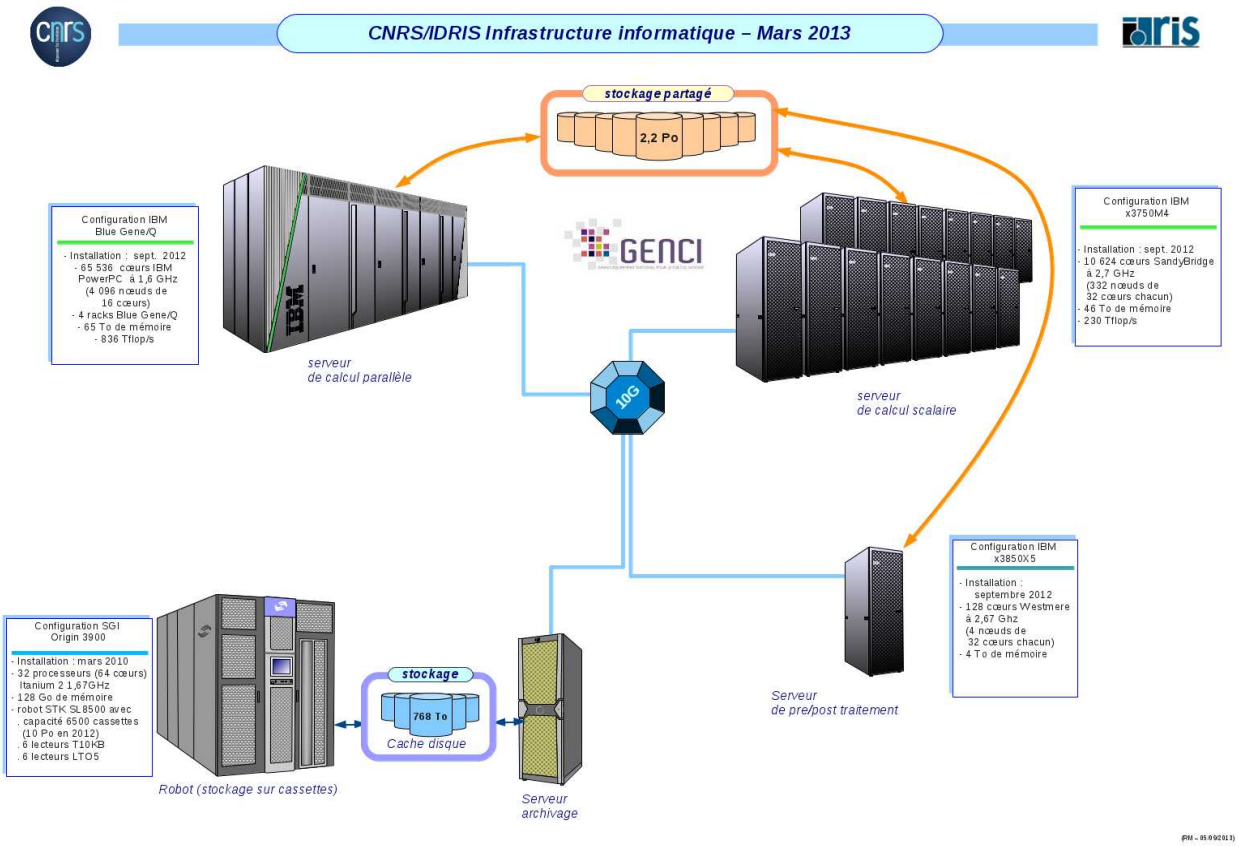


Configuration IDRIS

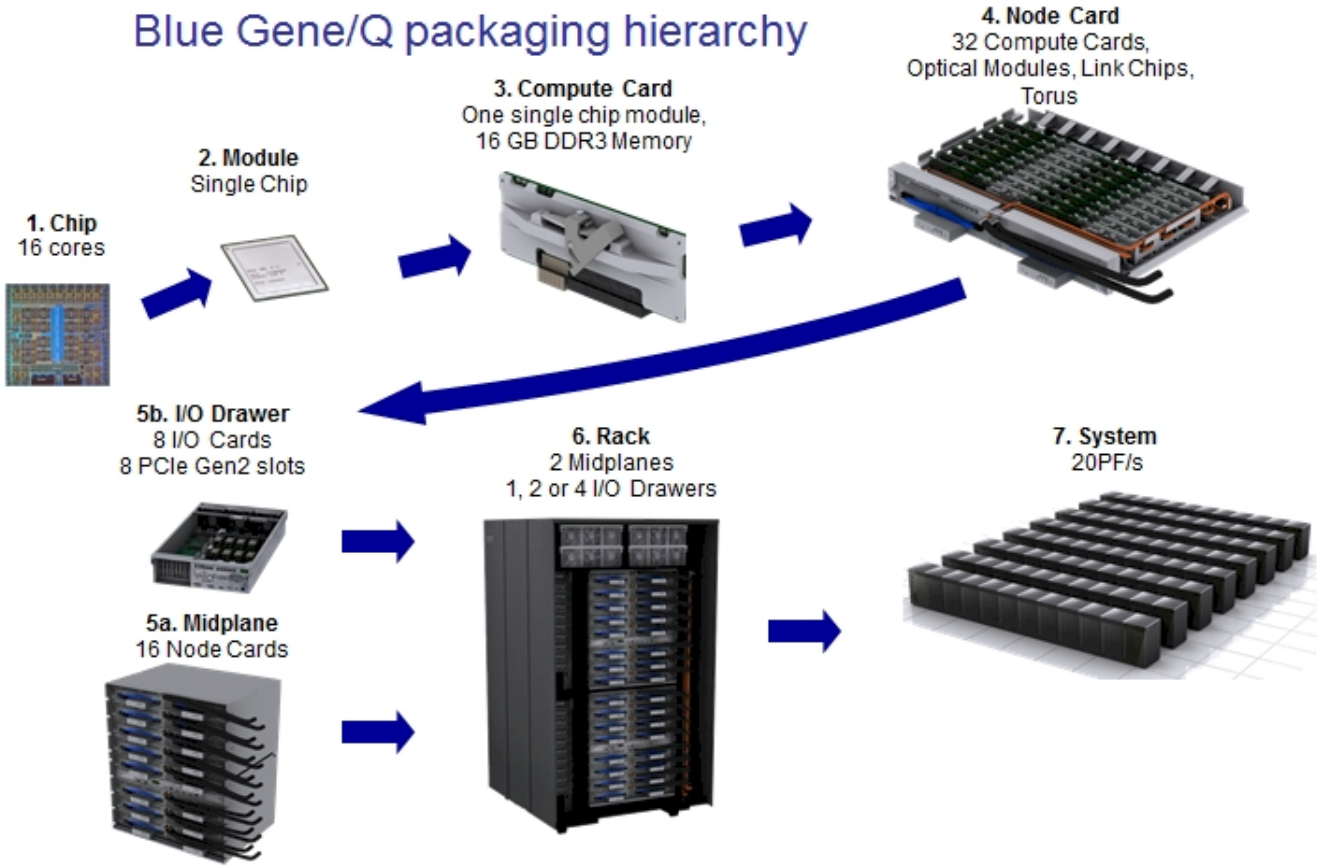
Chiffres importants

- **Turing : 4 *racks* Blue Gene/Q** :
  - 4.096 nœuds
  - 65.536 cœurs
  - 262.144 *threads*
  - 64 Tio
  - 839 Tflop/s
  - 424 kW (106 kW/*rack*)
- **Ada : 15 *racks* IBM x3750M4** :
  - 332 nœuds de calcul et 4 nœuds pour pré et post-traitement
  - 10.624 cœurs Intel SandyBridge à 2,7 GHz
  - 46 Tio
  - 230 Tflop/s
  - 366 kW
- 2,2 Po utiles et 50 Gio/s sur disques partagés entre BG/Q et Intel
- 790 kW pour la configuration complète (hors climatisation)

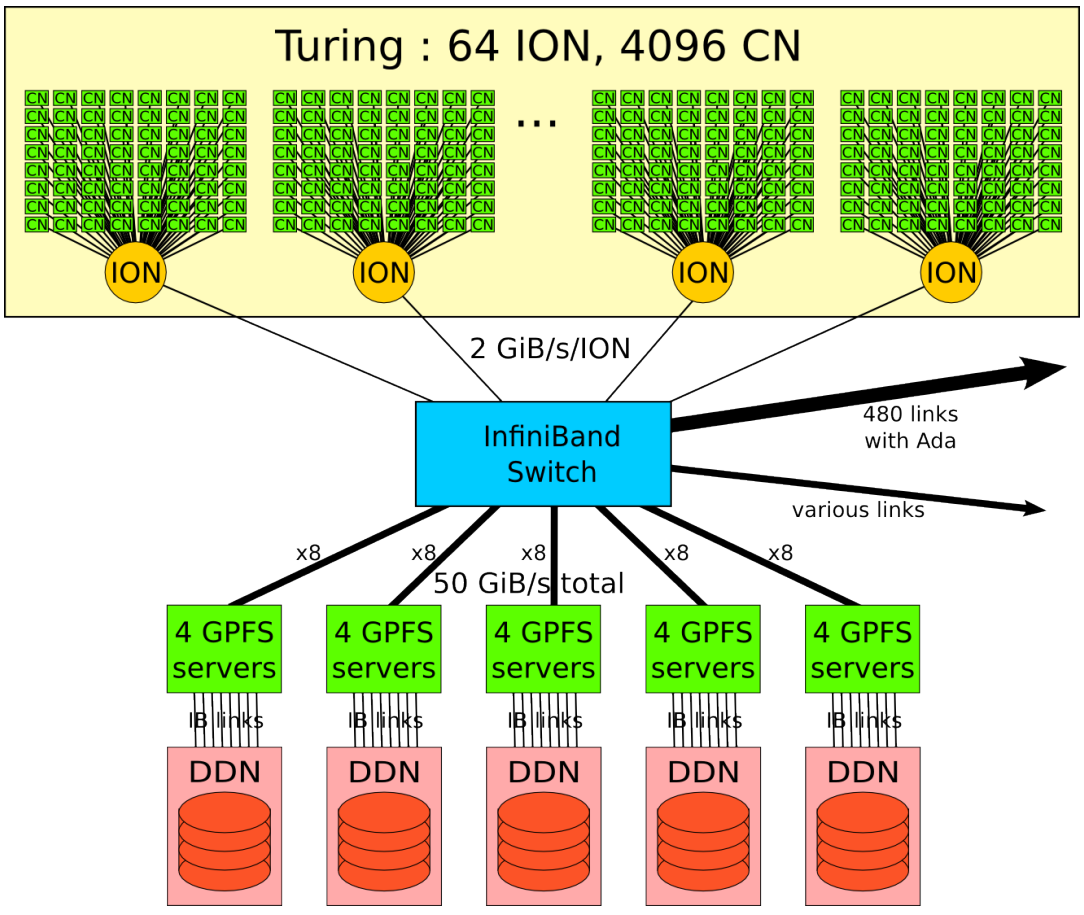
Configuration IDRIS



Architecture Blue Gene/Q



Entrées/sorties Blue Gene/Q IDRIS



Entrées/sorties IDRIS

Côté clients BG/Q (nœuds d'I/O)

- 1 I/O node tous les 64 compute nodes => 16/rack
- les ION gèrent tous les I/O (transparent côté CN)
- ION sont points de sortie des CN
- ION fournissent sockets aux CN
- ION : jusqu'à 3 Gio/s

Entrées/sorties IDRIS

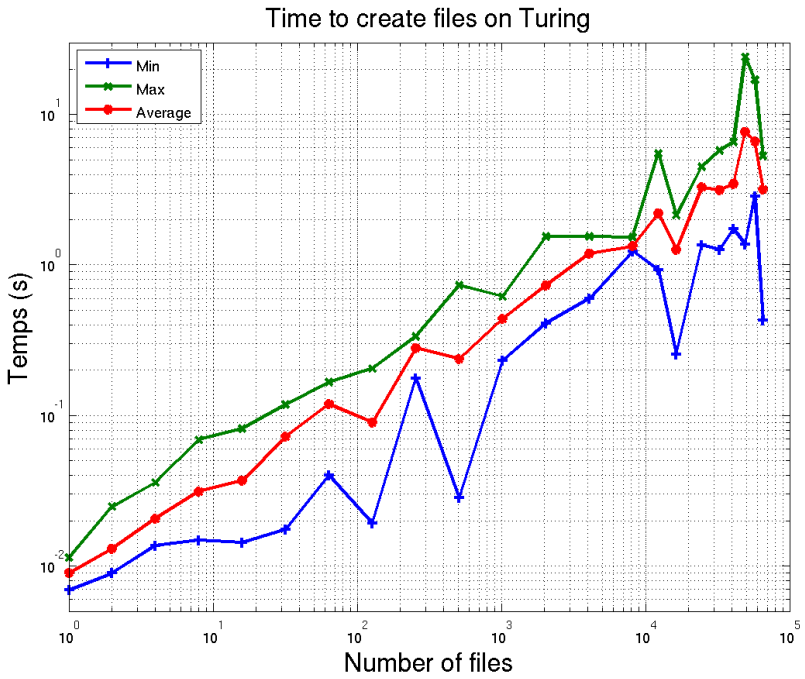
Côté serveurs I/O

- 20 serveurs GPFS
- 5 couplets DDN SFA10K (4 serveurs GPFS/baie) à 10 Gio/s/baie
- 2800 disques SATA 1 To (560/baie) en RAID6 8+2
- 2,2 Po utiles
- 1 Po pour le WORKDIR et 1 Po pour le TMPDIR partagés entre Ada et Turing
- Taille de bloc (WORKDIR et TMPDIR) : 4 Mio
- Plus petite écriture : taille_bloc/32 (4 Mio / 32 = 128 kio)

Fichiers séparés

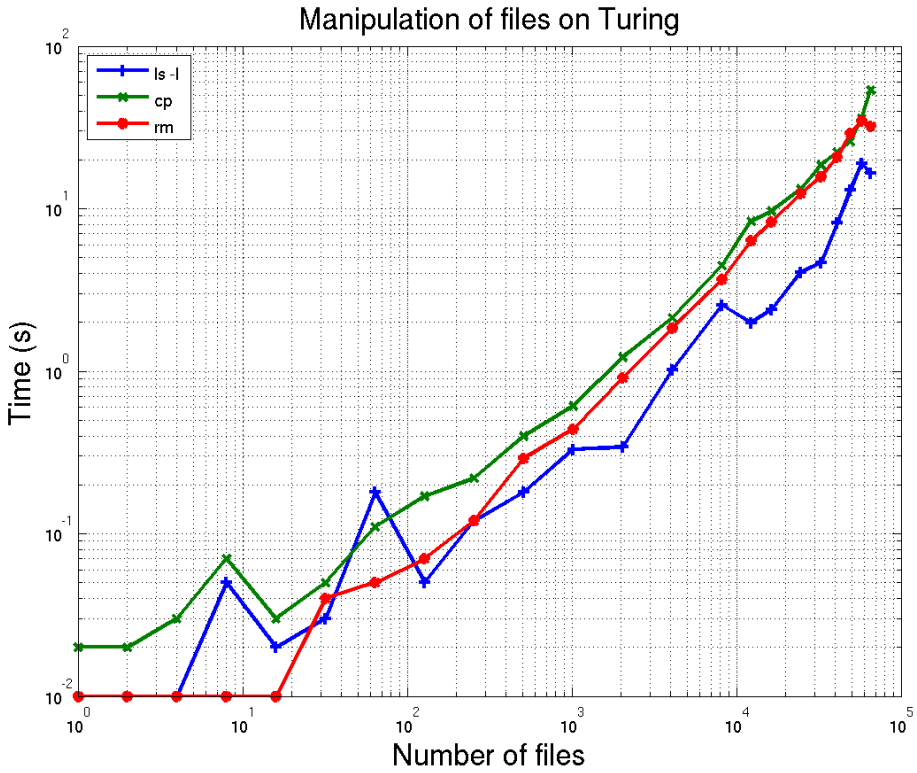
Temps de création d'un fichier par processus

Chaque processus crée un nouveau fichier et le ferme. Le temps mesuré correspond au processus le plus lent.



Fichiers séparés

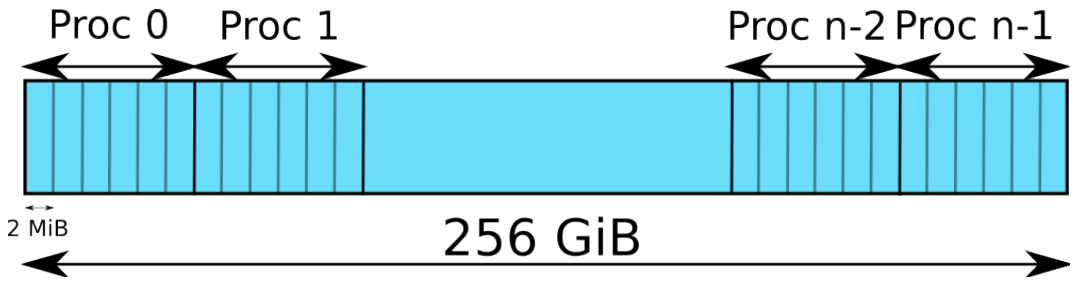
Temps de manipulation d'une série de fichiers



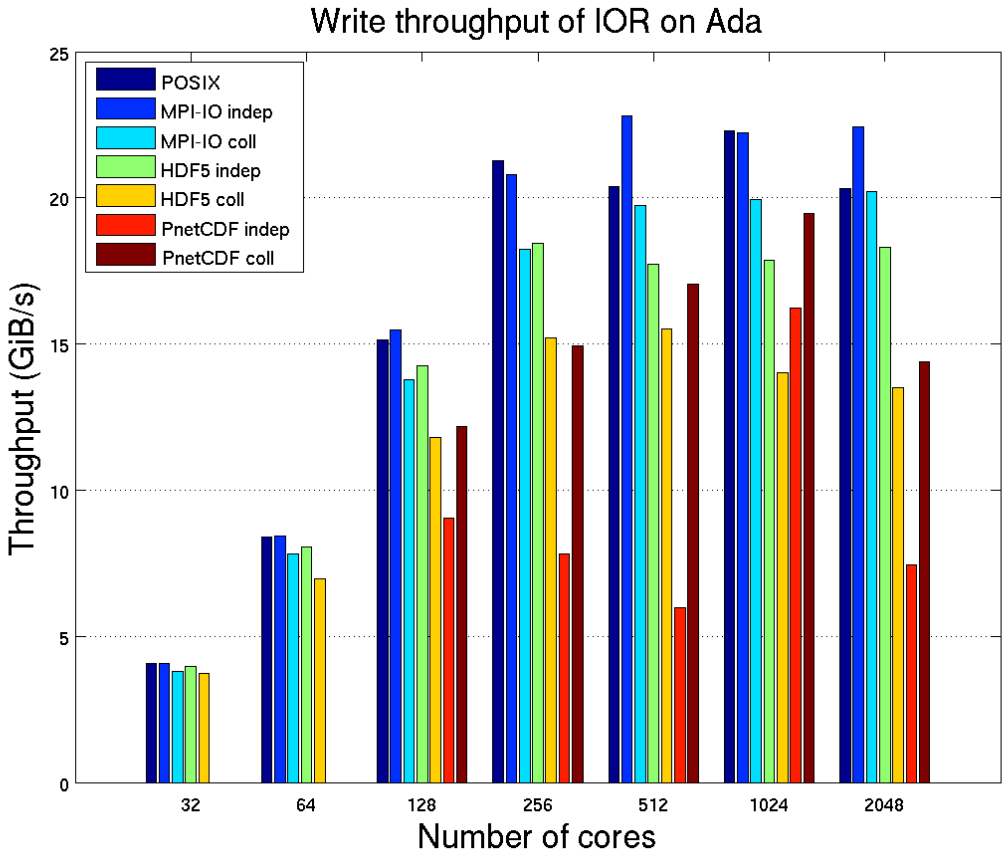
IOR

Présentation du benchmark IOR

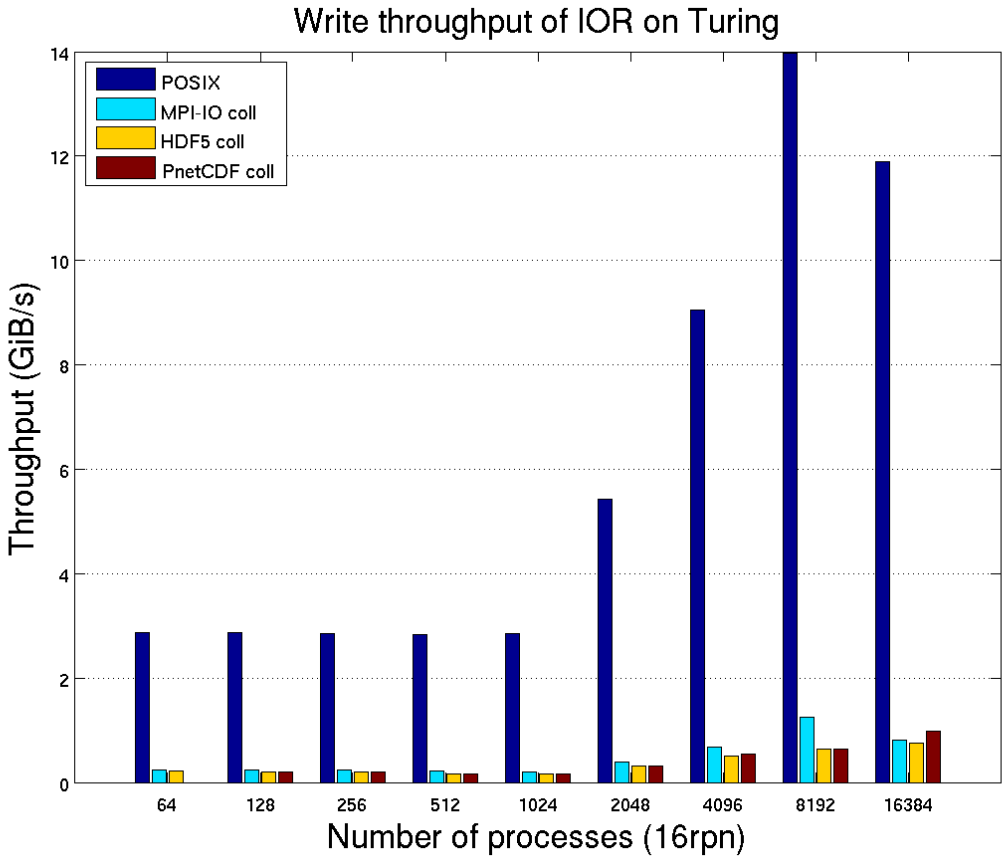
- IOR est un benchmark très utilisé pour les mesures de performances de systèmes de fichiers parallèles
- Supporte les fichiers POSIX séparés, MPI-I/O, HDF5 et Parallel-NetCDF
- Mode collectif ou non
- Chaque processus écrit son morceau de fichier de taille *blocksize* en l'écrivant par bloc de taille *xfersize*
- Tailles choisies ici : taille totale : 256 GiB, $xfersize = 4MiB$ (correspondant à la taille d'un bloc GPFS)
- *Hints* MPI-I/O (nécessaires pour bonnes performances) :
  - sur Ada : *romio_ds_write=disable*
  - sur Turing : valeurs par défaut (certains hints plantent)



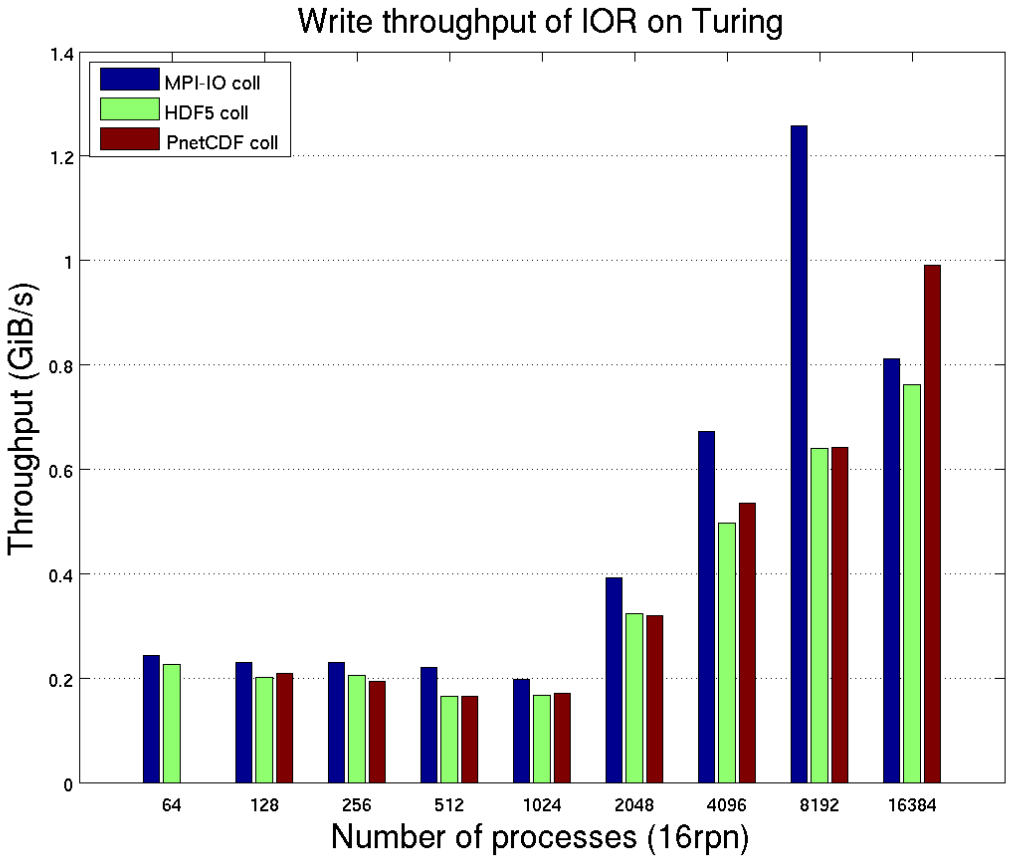
IOR



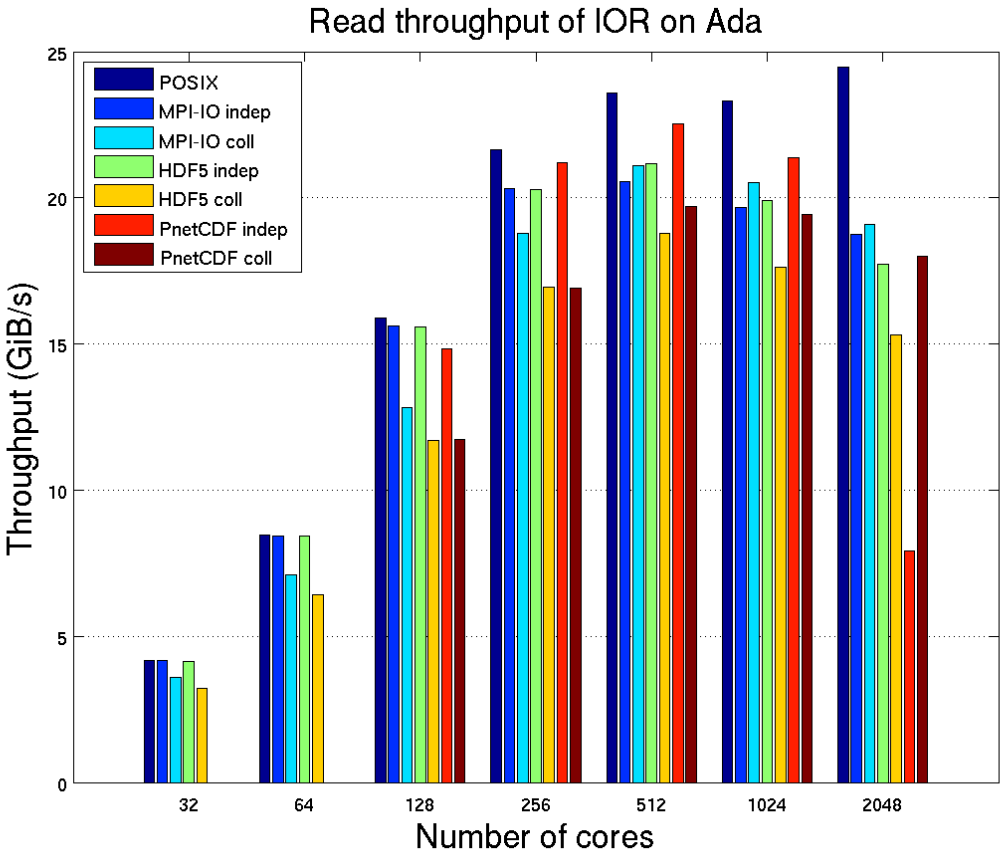
IOR


Write throughput of IOR on Turing
POSIX
MPI-IO coll
HDF5 coll
PnetCDF coll
Throughput (GiB/s)
0
2
4
6
8
10
12
14
64
128
256
512
1024
2048
4096
8192
16384
Number of processes (16rpn)

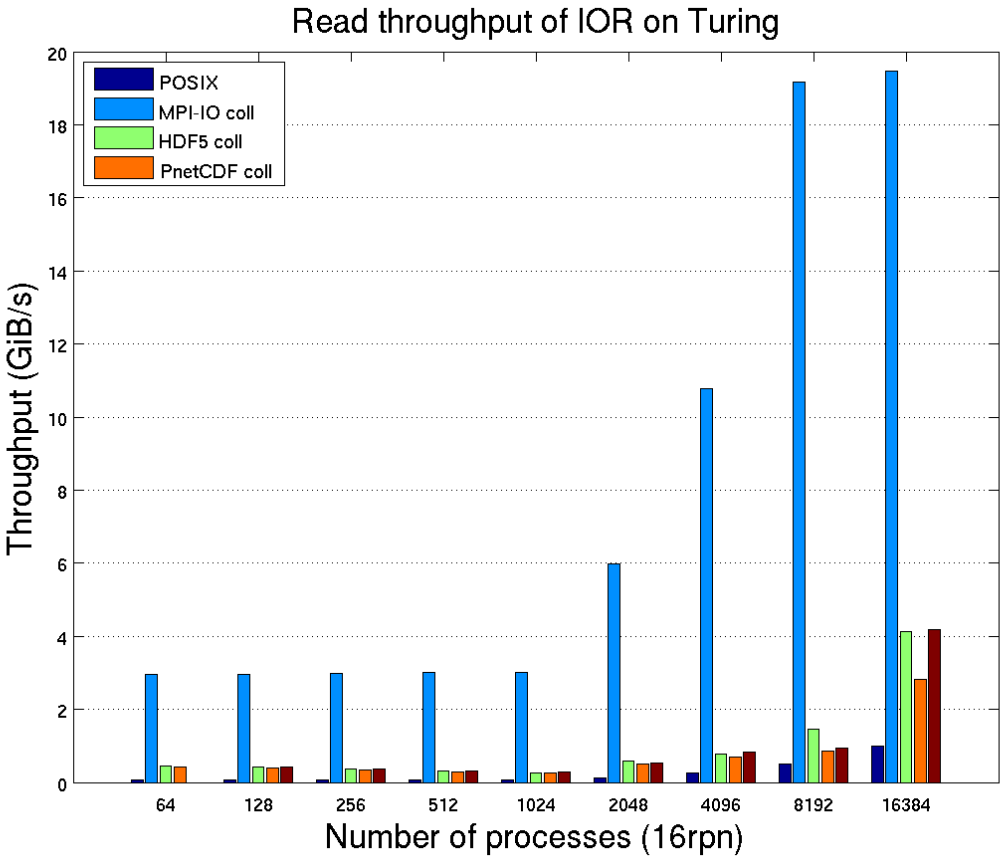
IOR


Write throughput of IOR on Turing
MPI-IO coll
HDF5 coll
PnetCDF coll
Throughput (GiB/s)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
64
128
256
512
1024
2048
4096
8192
16384
Number of processes (16rpn)

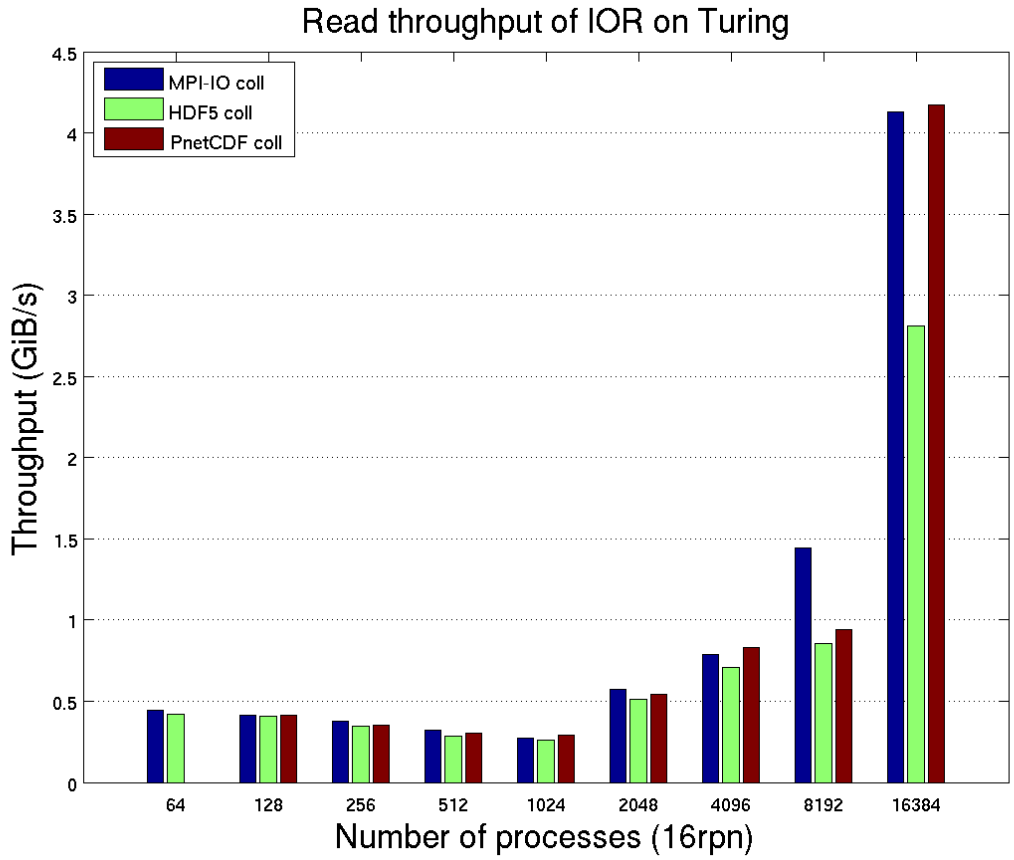
IOR


Read throughput of IOR on Ada
POSIX
MPI-IO indep
MPI-IO coll
HDF5 indep
HDF5 coll
PnetCDF indep
PnetCDF coll
Throughput (GiB/s)
0
5
10
15
20
25
32
64
128
256
512
1024
2048
Number of cores

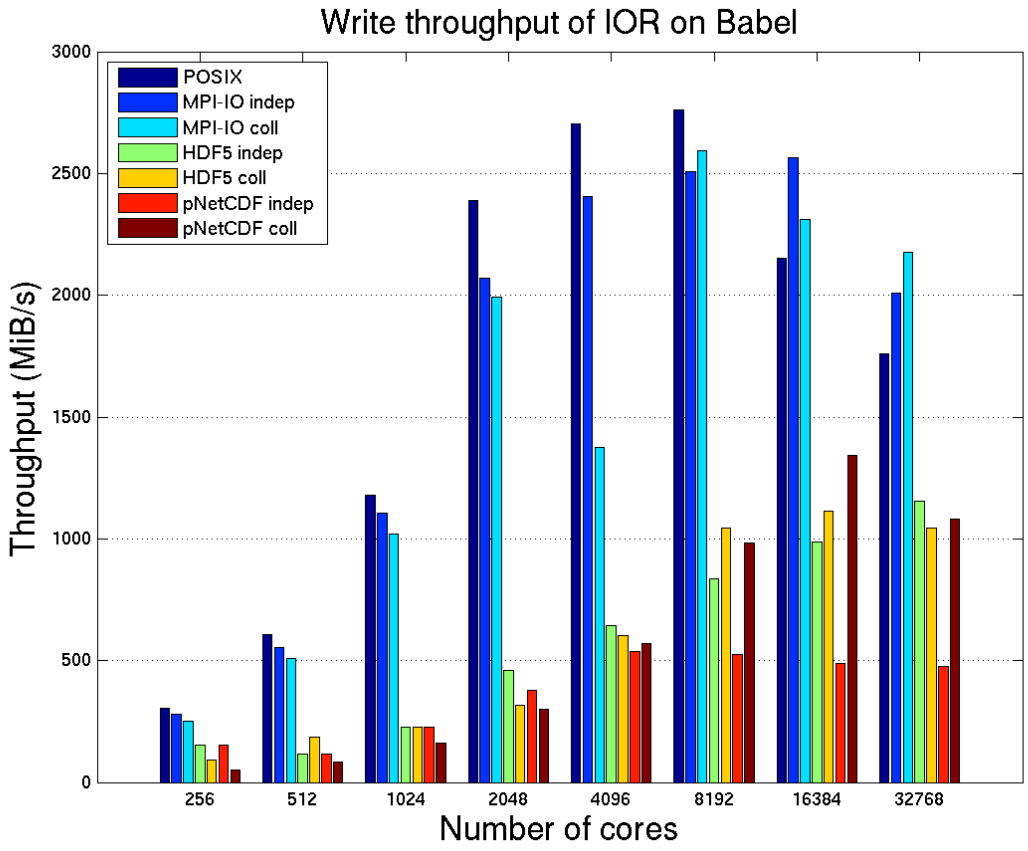
IOR


Read throughput of IOR on Turing
POSIX
MPI-IO coll
HDF5 coll
PnetCDF coll
Throughput (GiB/s)
0
2
4
6
8
10
12
14
16
18
20
64
128
256
512
1024
2048
4096
8192
16384
Number of processes (16rpn)

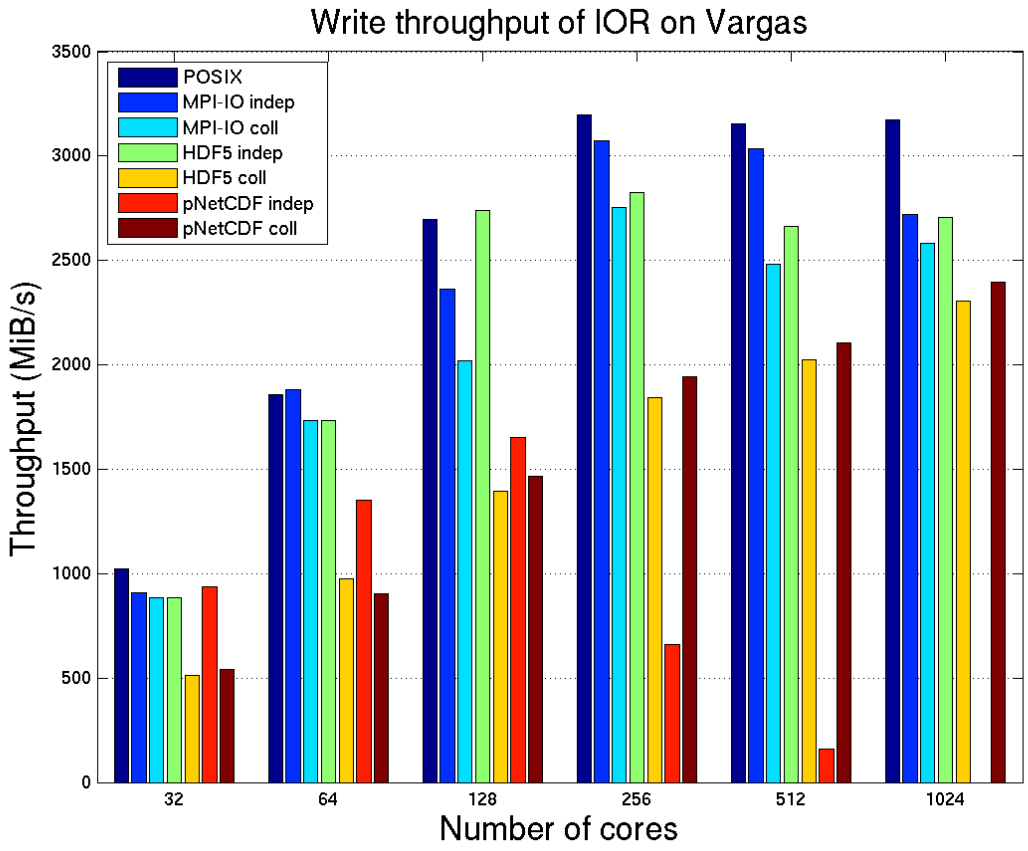
IOR


Read throughput of IOR on Turing
MPI-IO coll
HDF5 coll
PnetCDF coll
Throughput (GiB/s)
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
64
128
256
512
1024
2048
4096
8192
16384
Number of processes (16rpn)

IOR


Write throughput of IOR on Babel
POSIX
MPI-IO indep
MPI-IO coll
HDF5 indep
HDF5 coll
pNetCDF indep
pNetCDF coll
Throughput (MiB/s)
0
500
1000
1500
2000
2500
3000
256
512
1024
2048
4096
8192
16384
32768
Number of cores

IOR



IOR et hints MPI-I/O

Hints sur Ada : exemple

Tests écriture IOR sur Ada avec 128 processus et en mode collectif.

Hint	MPI-I/O	HDF5	pNetCDF
romio_cb_write = automatic romio_ds_write = automatic	9893	1081	9592
romio_cb_write = enable romio_ds_write = automatic	429	459	465
romio_cb_write = automatic romio_ds_write = disable	**10824**	**8295**	**9772**

Les valeurs sont en MiB/s. romio_cb_write = automatic et romio_ds_write = automatic par défaut.

Commentaires tests bas niveau IOR

- *POSIX* : approche la plus performante dans (presque) tous les cas. Les débits maximum sont atteints dès 1 rack Blue Gene/Q ou 8 nœuds (256 processus) d'Ada.
- *MPI-I/O* : très performant. Presque aussi bon que POSIX sauf sur Blue Gene/Q (impossible de positionner le hint romio_cb_write/read à automatic ou disable sans plantage).
- *HDF5* : légèrement moins bon que MPI-I/O.
- *Parallel-NetCDF* : proche d'HDF5 en mode collectif (très légèrement plus rapide). Assez mauvais en mode individuel : mauvaise extensibilité.
- Les *hints* MPI-I/O sont cruciaux pour obtenir de bonnes performances. Les valeurs par défaut ne sont pas toujours adaptées à ce *benchmark* (situation différente avec le code RAMSES).

RAMSES

Présentation

- RAMSES est une application de calcul d'astrophysique développée au départ par Romain Teyssier (CEA) sous licence CeCILL.
- Résout les équations d'Euler en présence d'auto-gravité et de refroidissement traités comme termes sources dans les équations du moment et de l'énergie.
- Raffinement adaptatif de maillage (méthode AMR) avec équilibrage de charge et défragmentation de la mémoire dynamiques
- Méthode multi-grilles et gradient conjugué pour l'équation de Poisson
- Solveurs de Riemann (Lax-Friedrich, HLLC, exact) pour la dynamique des gaz adiabatiques
- Dynamique de particules (sans collisions) pour la matière noire et les étoiles
- Formation des étoiles, supernovae...
- Code utilisant MPI optimisé par l'IDRIS (entre autres) et tournant régulièrement sur plusieurs dizaines de milliers de processus

RAMSES est disponible sur le site
http://www.itp.uzh.ch/ teyssier/ramses/RAMSES.html

RAMSES et les I/O

Entrées dans RAMSES (version originale)

RAMSES peut soit commencer une nouvelle simulation, soit repartir d'une solution précédente (*restart*)

- Dans tous les cas, lecture d'un fichier texte de type *namelist* Fortran contenant les paramètres de la simulation par tous les processus
- Pour une nouvelle simulation, lecture de quelques fichiers d'entrées par le processus 0 qui *broadcaste* vers les autres et lecture par tous les processus de quelques données dans un fichier d'entrées.
- Pour un *restart*, chaque processus lit son ou ses fichiers (entre 1 et 4 selon les options)

Sorties dans RAMSES (version originale)

- Les sorties sont effectuées au choix tous les *n* pas temps et/ou à certains instants dans le temps physique de la simulation
- Chaque processus écrit son/ses fichiers
- Selon les options, entre 1 et 4 fichiers sont écrits par processus
- 2 petits fichiers sont égalements écrits par le premier processus

Processus spécialisés dans les I/O

Processus spécialisés : entrées

- Dans tous les cas, lecture d'un fichier texte de type *namelist* Fortran contenant les paramètres de la simulation par tous les processus
- Pour une nouvelle simulation, pas de changements : lecture de quelques fichiers d'entrées par le processus 0 qui *broadcaste* vers les autres et lecture par tous les processus de quelques données dans un fichier d'entrées.
- Pour un *restart*, les processus sont groupés et seul un processus par groupe lit ses fichiers à un moment donné (passage de jeton jusqu'à ce que tous les processus aient lu leurs données)

Processus spécialisés : sorties

- 2 types de processus : processus de calcul et processus d'I/O dans 2 communicateurs MPI distincts
- Ecritures remplacées par des envois aux processus d'I/O
- Les processus d'I/O peuvent écrire dans un espace disque qui leur est local
- Une fois toutes les données envoyées, les processus de calcul reprennent leurs opérations
- Les processus d'I/O peuvent pendant ce temps transférer les données locales vers un espace partagé

RAMSES et les I/O

Bibliothèques parallèles

L'utilisation des bibliothèques d'entrées-sorties parallèles suivantes a été intégrée dans RAMSES :

- MPI-I/O
- HDF5
- Parallel-NetCDF

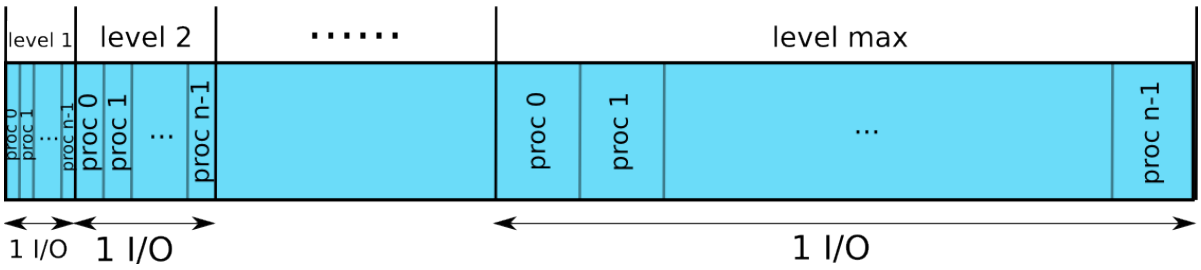
Caractéristiques

- Selon les options, de 1 à 4 gros fichiers partagés entre tous les processus par sortie (chaque processus écrit sa partie)
- Toutes les I/O se font via les interfaces collectives (car les plus performantes dans les tests RAMSES)
- Actuellement, fichiers dépendant du nombre de processus (pas possible de faire un *restart* avec un nombre différent de processus)
- Certaines données sont encore redondantes entre les différents fichiers

Structure des fichiers (utilisés dans les tests)

Fichier AMR

- Un ensemble de paramètres identiques sur tous les processus (14 entiers, 22 doubles, un petit tableau d'entiers (10*nlevelmax) et une chaîne de caractères)
- Un ensemble de variables différentes sur chaque processus (8 entiers par processus et 3 structures de données (ncpu*nlevelmax entiers par processus)). Les valeurs sur les différents processus d'une même variable sont disposées les unes derrières les autres dans le fichier.
- Un ensemble de grosses structures de données (8 structures contenant entre 1 et 8 valeurs entières ou doubles par maille) pour un total de 156 octets * nombre de mailles. Elles sont structurées de la façon suivante :



Remarques : les grilles (ou niveaux) sont de tailles croissantes (facteur 8 pour les grilles pleines) et certains processus peuvent ne pas contenir de valeurs pour certains niveaux (dépendants du raffinement).

Structure des fichiers (utilisés dans les tests)

Fichier hydro

- Un ensemble de paramètres identiques sur tous les processus (5 entiers, 1 double)
- Un ensemble de variables différentes sur chaque processus (2 structures de données (ncpu*nlevelmax et 100*nlevelmax entiers par processus)). Les valeurs sur les différents processus d'une même variable sont disposées les unes derrières les autres dans le fichier.
- Un ensemble de grosses structures de données (5 structures contenant 8 valeurs double précision par maille) pour un total de 320 octets * nombre de mailles. Elles sont structurées de la même façon que dans AMR sauf que les valeurs sont groupées par 8 (donc 8 fois plus de valeurs par niveau).

Fichier AMR vs hydro

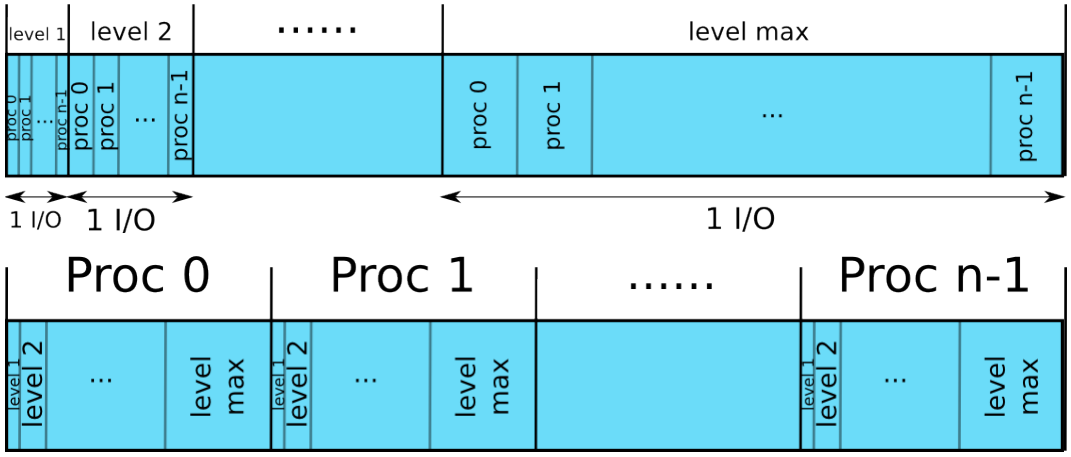
- Le fichier AMR est plus complexe et contient plus de variables différentes
- Les 2 premières sections sont beaucoup plus importantes dans le fichier AMR (mais restent relativement petites en taille absolue)
- Le fichier hydro est environ 2 fois plus gros (320 octets par maille au lieu de 156 pour la 3ème section)
- La granularité des écritures est nettement plus grosse dans le fichier hydro

Approches MPI-I/O et structure des données

Comparaison des approches

Tests MPI-I/O effectués sur le fichier hydro sur Blue Gene/P avec 2048 cœurs et cas test sedov3d 1024 (taille du fichier : 54,9 GiB)

	Groupes par niveau			Groupes par processus	
	write_at_all	write_at	write_ordered	write_at_all	write_at
Débit (MiB/s)	810	158	197	106	342



Hints MPI-I/O

Définition des hints MPI-I/O

- Les hints MPI-I/O sont des indications que l'ont peut fournir à la couche MPI-I/O pour optimiser les performances (ils influencent aussi les bibliothèques reposant sur MPI-I/O comme HDF5, netCDF-4 et Parallel-NetCDF)
- Chaque implémentation fournit sa liste de hints (parfois pas ou peu documentés)
- L'impact sur les performances peut être majeur
- Les hints peuvent être choisis lors de l'ouverture des fichiers, pour certains en cours d'utilisation des fichiers ou parfois par des variables d'environnement
- Chaque hint a une valeur par défaut

Hints MPI-I/O

Hints sur Turing (IBM Blue Gene/Q)

- Testés : romio_cb_read/romio_cb_write, romio_ds_read/romio_ds_write, romio_no_indep_rw et cb_nodes
- Non testés : cb_buffer_size, romio_cb_fr_types, romio_cb_fr_alignment, romio_cb_alltoall, romio_cb_pfr, romio_cb_ds_threshold, ind_rd_buffer_size, ind_wr_buffer_size

Hints sur Ada (IBM PEMPI, x86_64)

- Testés : romio_cb_read/romio_cb_write, romio_ds_read/romio_ds_write, romio_no_indep_rw, cb_config_list, cb_nodes et cb_buffer_size
- Non testés : romio_cb_fr_types, romio_cb_fr_alignment, romio_cb_alltoall, romio_cb_pfr, romio_cb_ds_threshold, ind_rd_buffer_size et ind_wr_buffer_size

Hints MPI-I/O

Hints sur Ada : exemple

Tests d'écriture HDF5 (sedov3d 1024^3 256 processus)

hint	*AMR*	*hydro*
romio_cb_write = automatic romio_ds_write = automatic	18.4s	15.1s
romio_cb_write = enable romio_ds_write = automatic	**16.9s**	**13.2s**
romio_cb_write = disable romio_ds_write = automatic	160.4s	77.3s
romio_cb_write = automatic romio_ds_write = enable	18.8s	15.9s
romio_cb_write = automatic romio_ds_write = disable	18.5s	16.0s

La première ligne correspond aux valeurs par défaut.

Hints MPI-I/O

Hints sur Blue Gene/Q : exemple

Tests de lecture MPI-IO (sedov3d 1024^3 4096 processus (1 par cœur))

hint	*AMR*	*hydro*
romio_cb_read = enable romio_ds_read = automatic	**37.4s**	**18.1s**
romio_cb_read = disable romio_ds_read = automatic	1297.7s	177.9s
romio_cb_read = automatic romio_ds_read = automatic	1128.4s	149.0s
romio_cb_read = enable romio_ds_read = disable	42.2s	20.2s
romio_cb_read = enable romio_ds_read = enable	40.5s	18.8s

La première ligne correspond aux valeurs par défaut.

Hints MPI-I/O et taille des fichiers

Choix des hints pour les tests

- Blue Gene/Q
  - En écriture : valeurs par défaut
  - En lecture : valeurs par défaut
- Ada
  - En écriture : romio_cb_write = enable
  - En lecture : romio_cb_read = disable

Taille des fichiers de sortie

Taille (en GiB) des fichiers de sortie HDF5 sur 1024 processus en fonction de la taille du problème étudié (tous les tests sont basés sur un sedov3d).

	128^3	256^3	512^3	1024^3	2048^3
AMR	0,213	0,707	3,814	25,860	191,8
hydro	0,299	1,292	7,646	52,848	393,2

Remarque : la taille des fichiers dépend également du nombre de processus à cause des variables et structures différentes sur chacun d'entre-eux.

Procédure de test

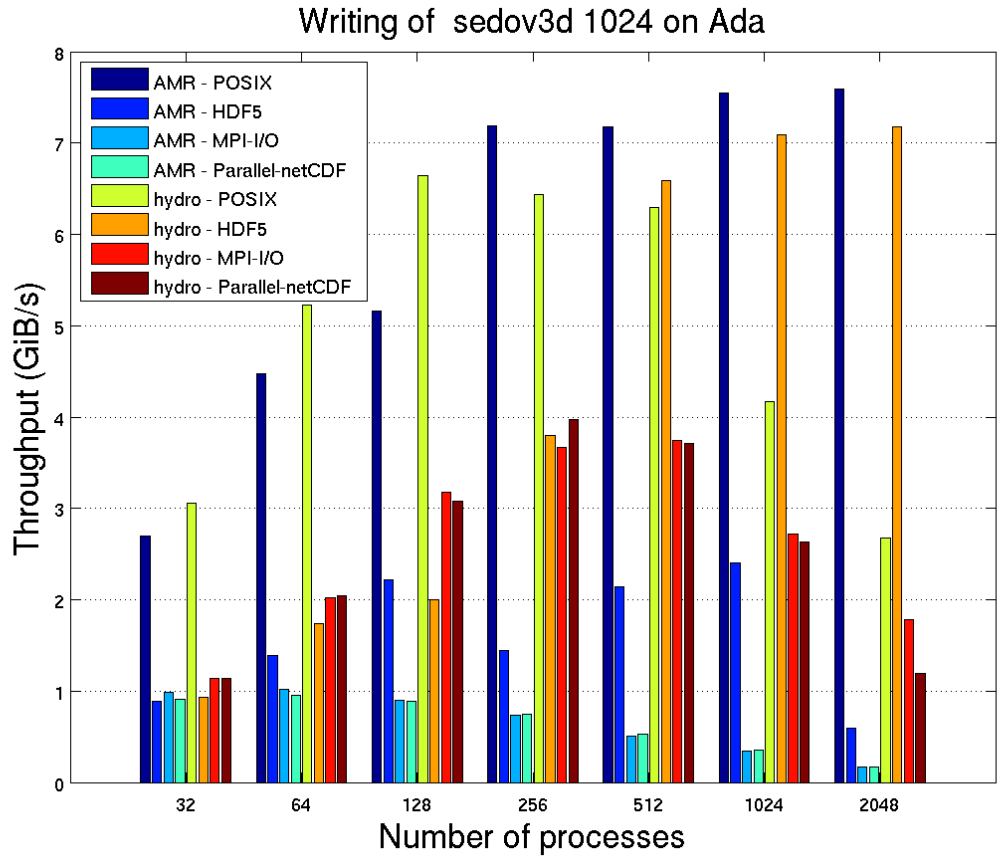
Procédure de test

- Cas test choisi : sedov3d (explosion dans une boîte)
- Taille du domaine : 1024^3 et 2048^3
- Nombre de processus : du plus petit au plus grand possible
- Pas de raffinement adaptatif de maillage
- Toutes les mesures effectuées au moins 5 fois (dans la mesure du possible)
- Runs non dédiés sur les machines (machines en production normale)
- **Attention :** l'algorithme d'écriture pour les fichiers HDF5 sur Ada et Turing est différent des autres approches avec un regroupement de plusieurs écritures (agrégation de plusieurs niveaux de maillage).

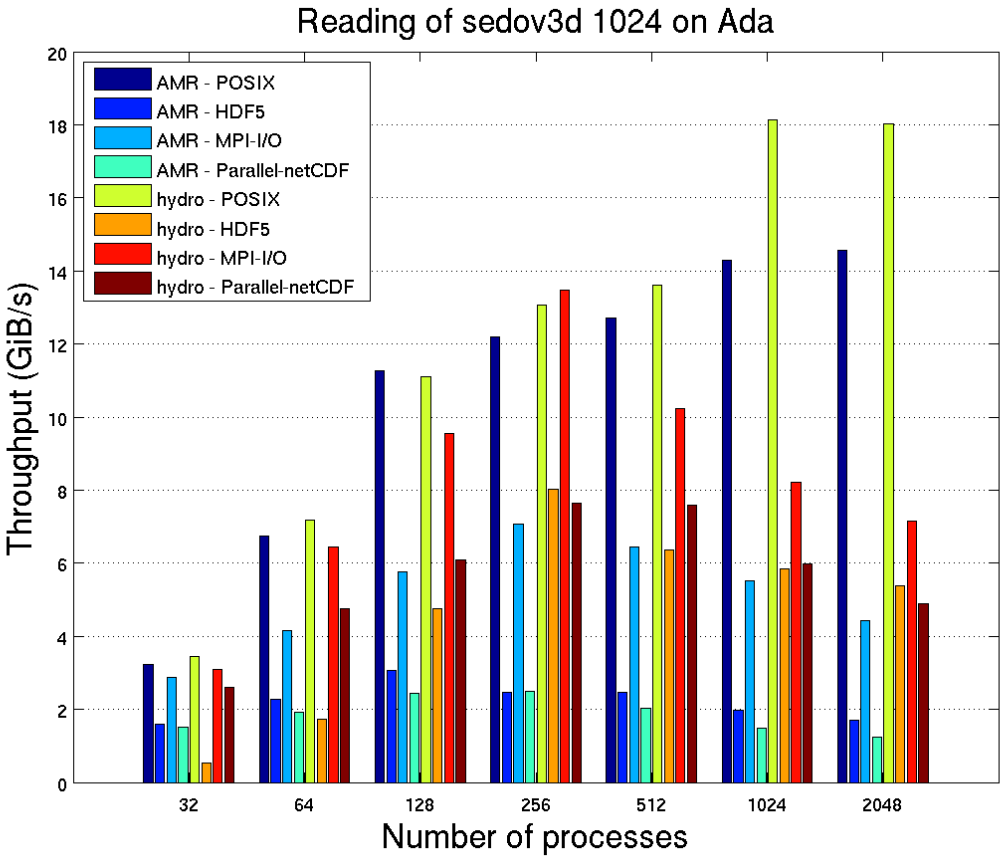
Dates des tests et version des bibliothèques d'I/O

Machine(s)	*Dates des tests*	*HDF5*	*PnetCDF*
Ada/Turing	Eté 2013	1.8.11	1.3.1
Babel/Vargas	Hiver 2010/2011	1.8.5	1.2.0
Curie	Novembre 2011	1.8.7	1.1.1
Inti	Eté 2012	1.8.9	1.1.1

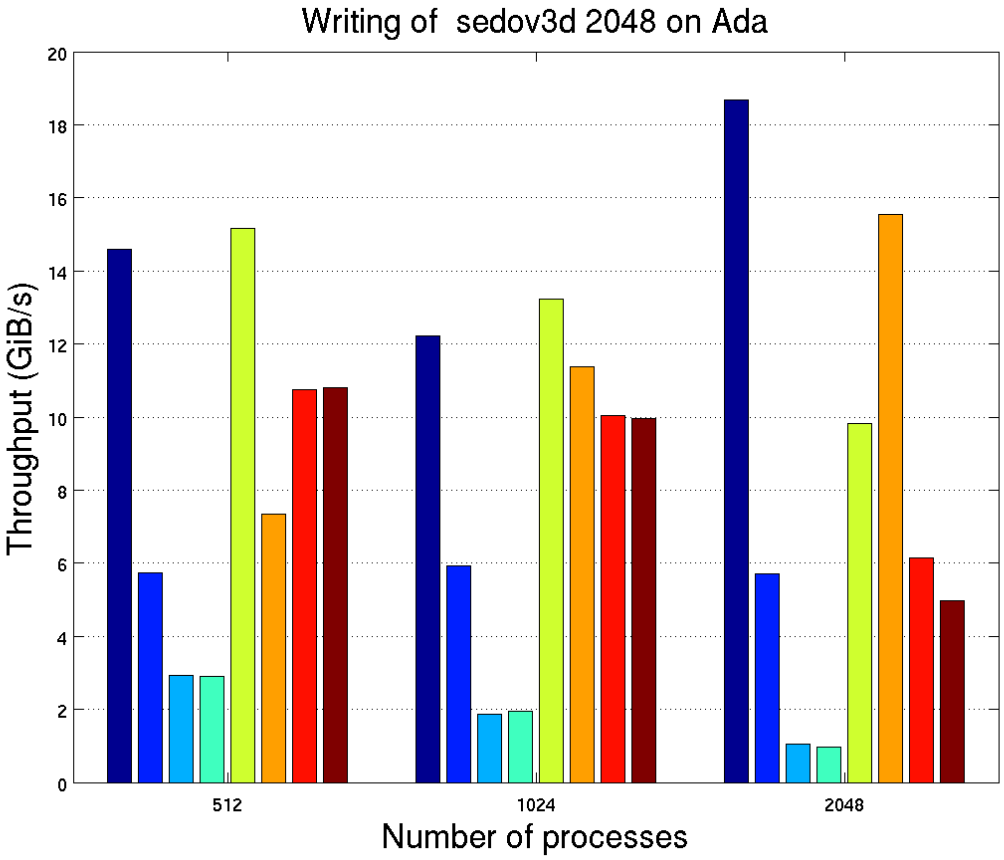
Ecriture Sedov3d 1024^3 sur Ada



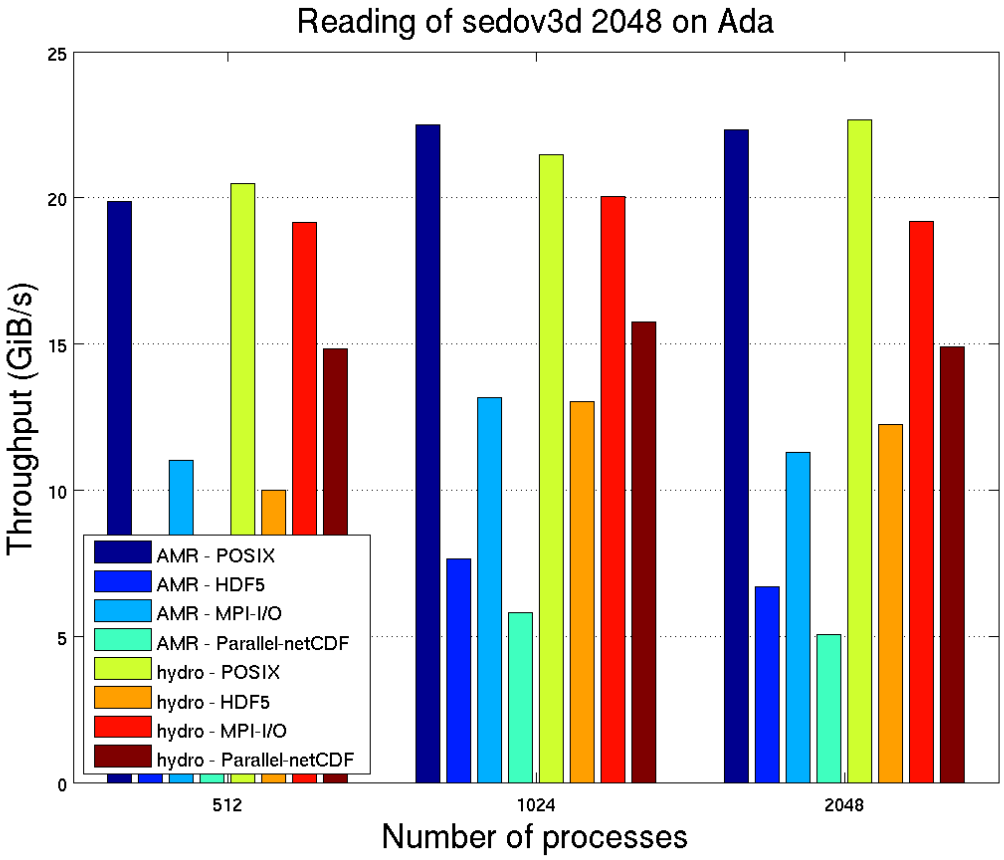
Lecture Sedov3d 1024^3 sur Ada



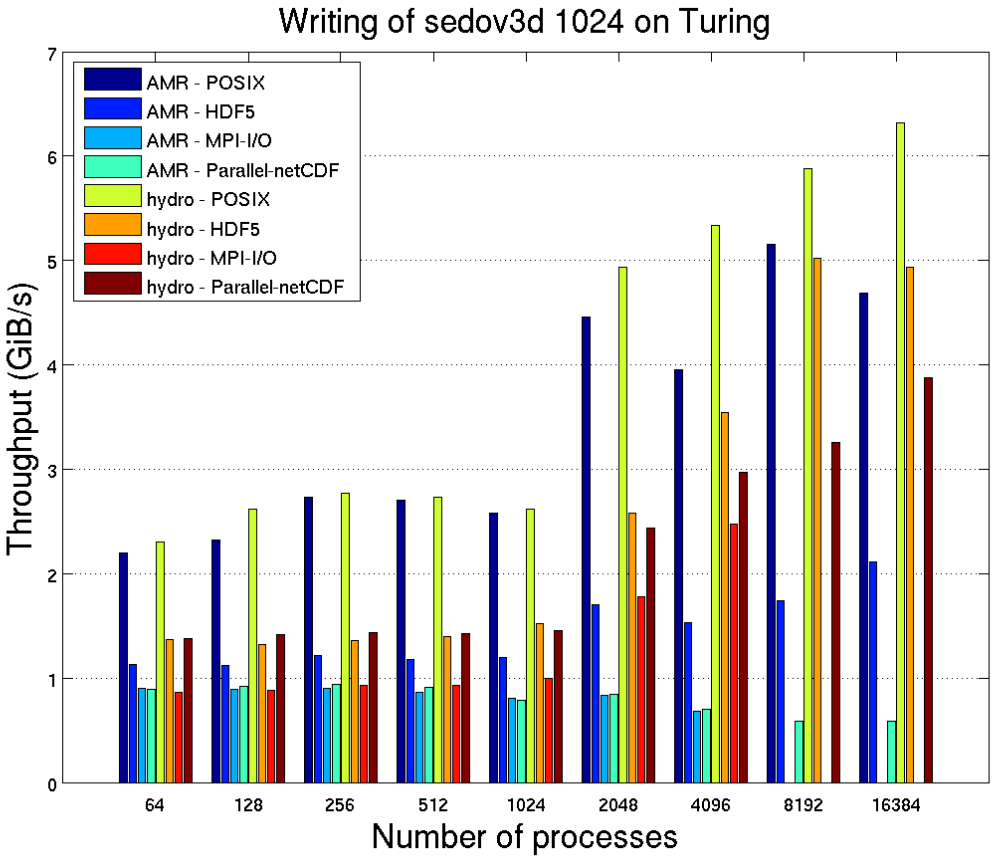
Ecriture Sedov3d 2048^3 sur Ada



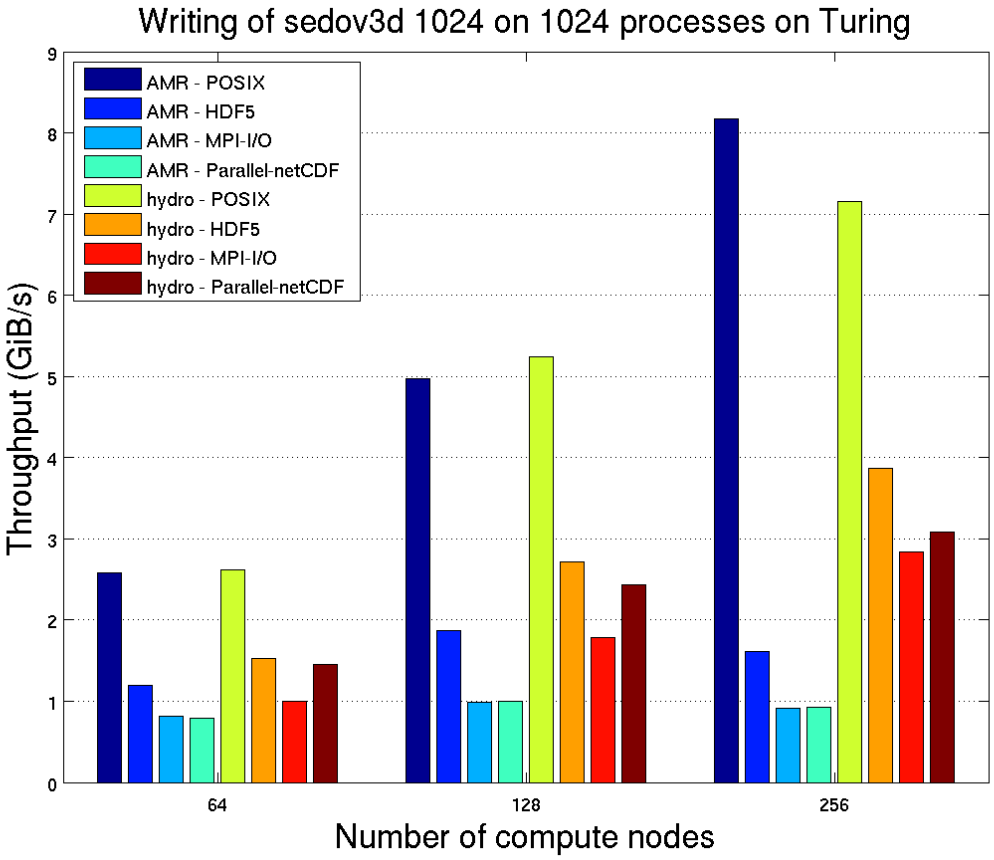
Lecture Sedov3d 2048^3 sur Ada



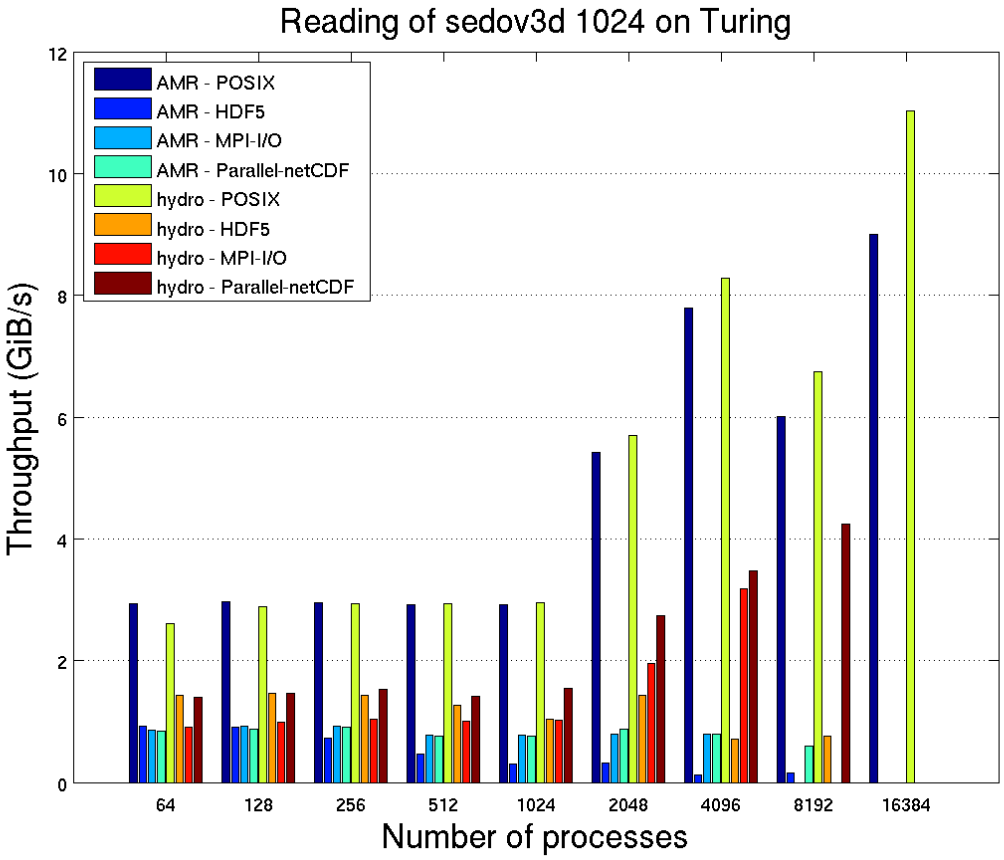
Ecriture Sedov3d 1024^3 sur Turing



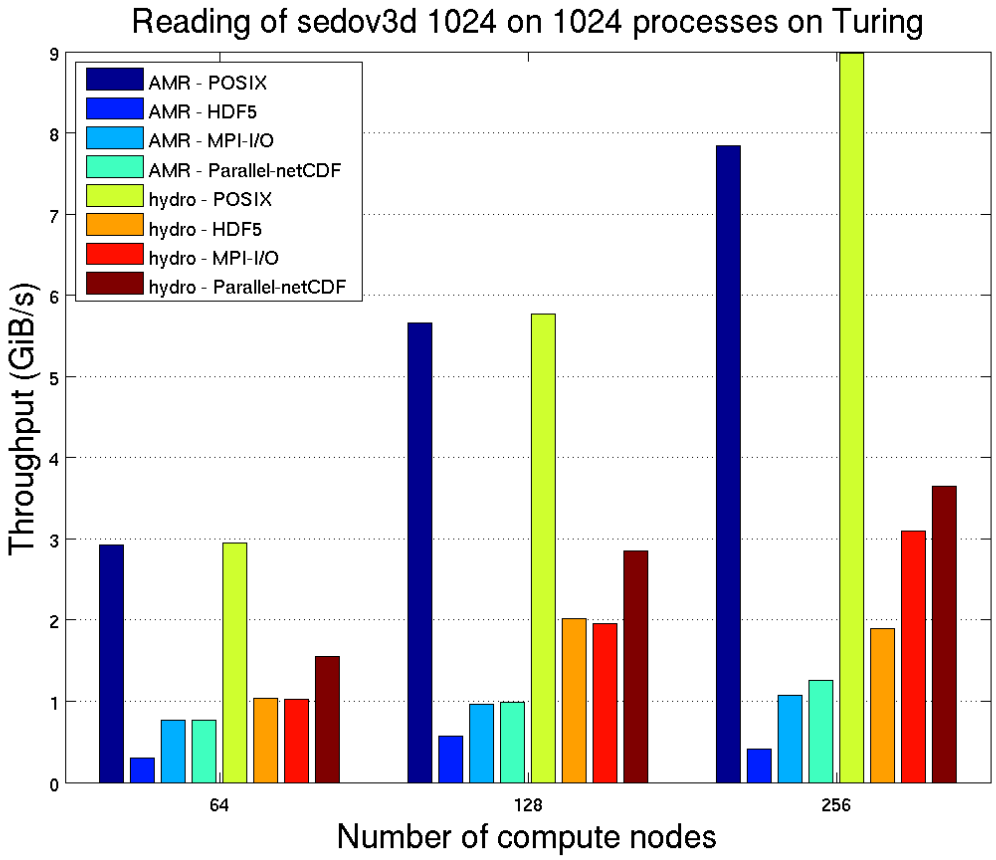
Ecriture Sedov3d 1024^3 sur Turing



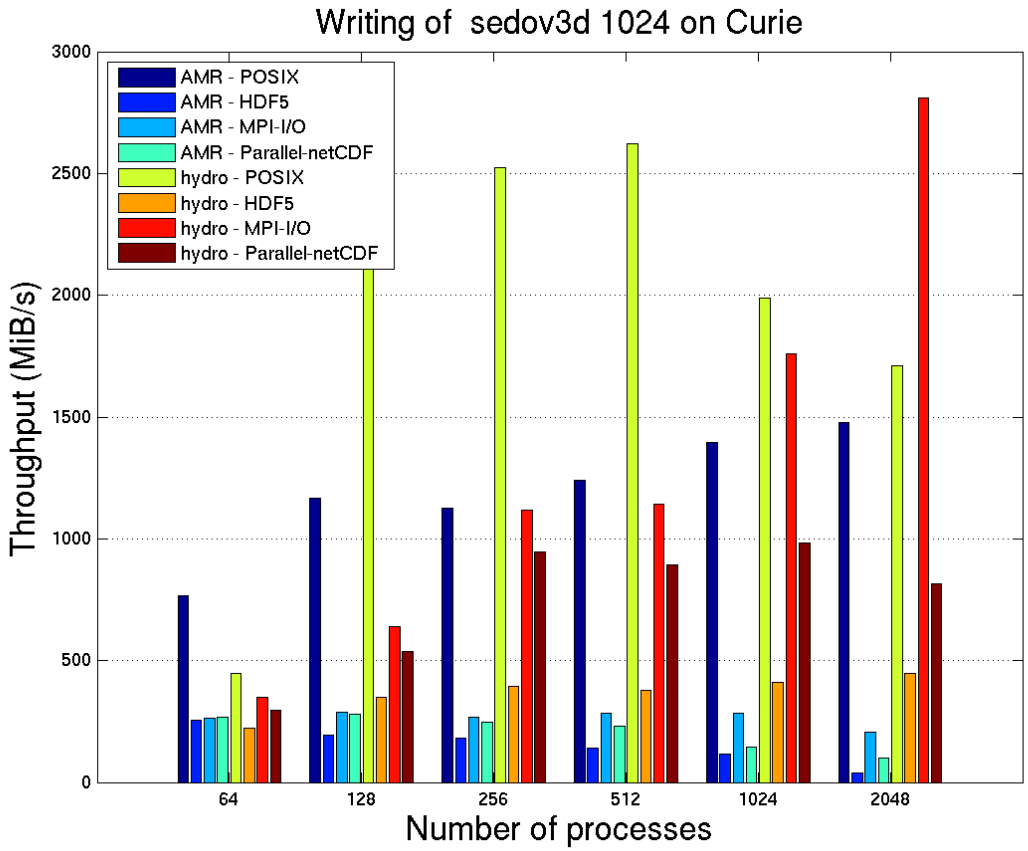
Lecture Sedov3d 1024^3 sur Turing



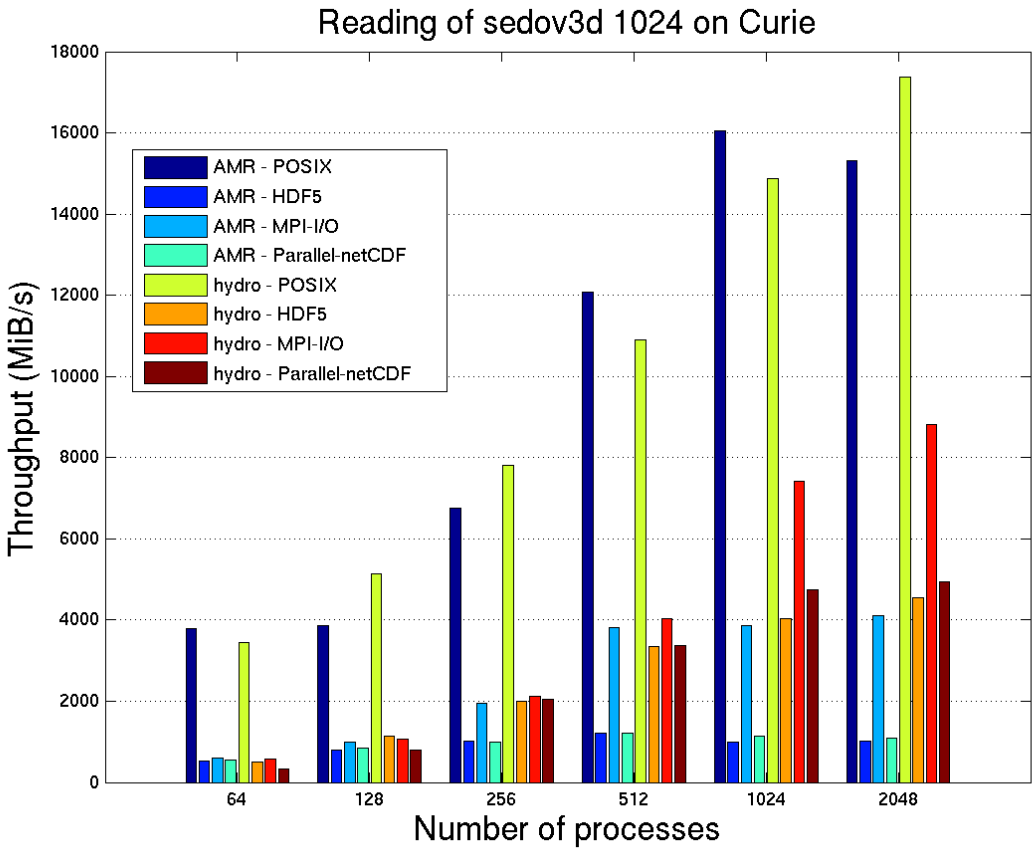
Lecture Sedov3d 1024^3 sur Turing



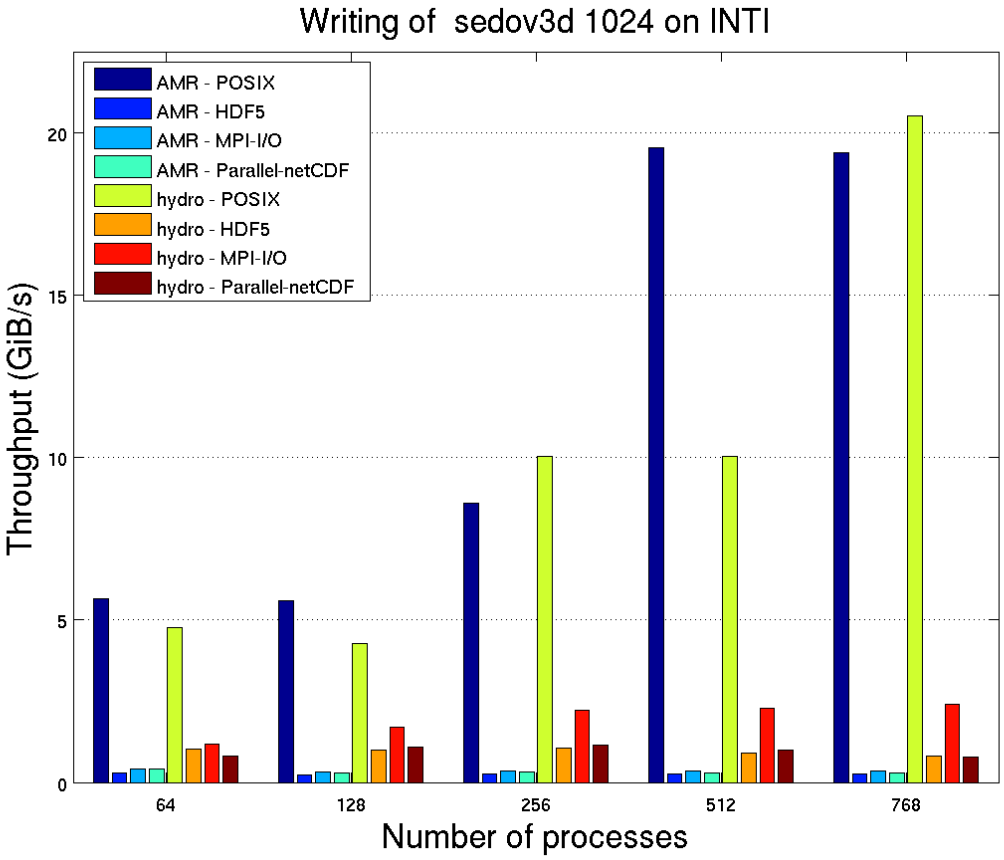
Ecriture Sedov3d 1024^3 sur Curie



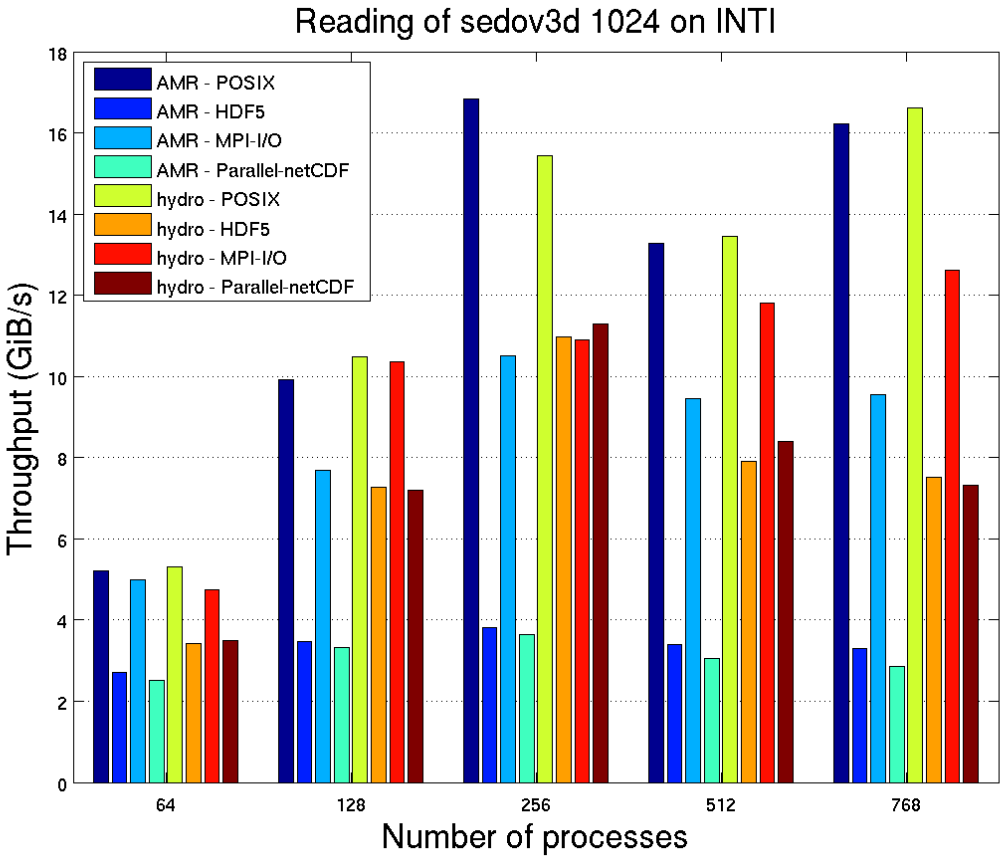
Lecture Sedov3d 1024^3 sur Curie



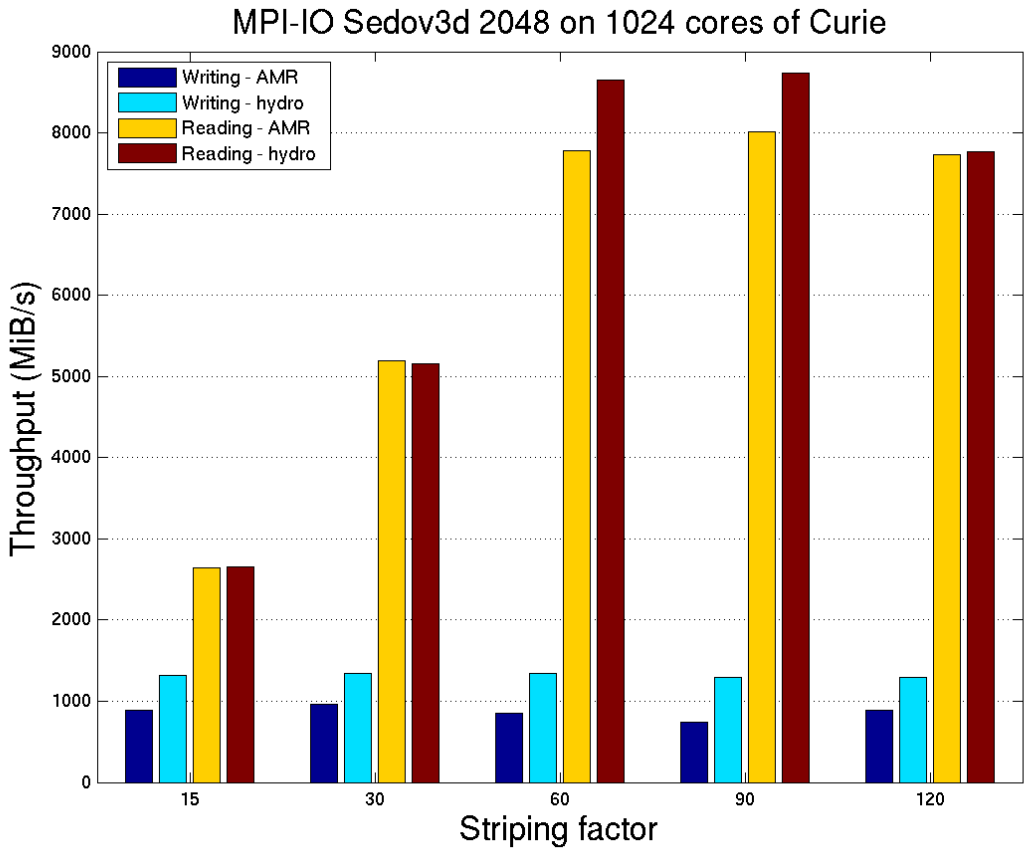
Ecriture Sedov3d 1024^3 sur INTI



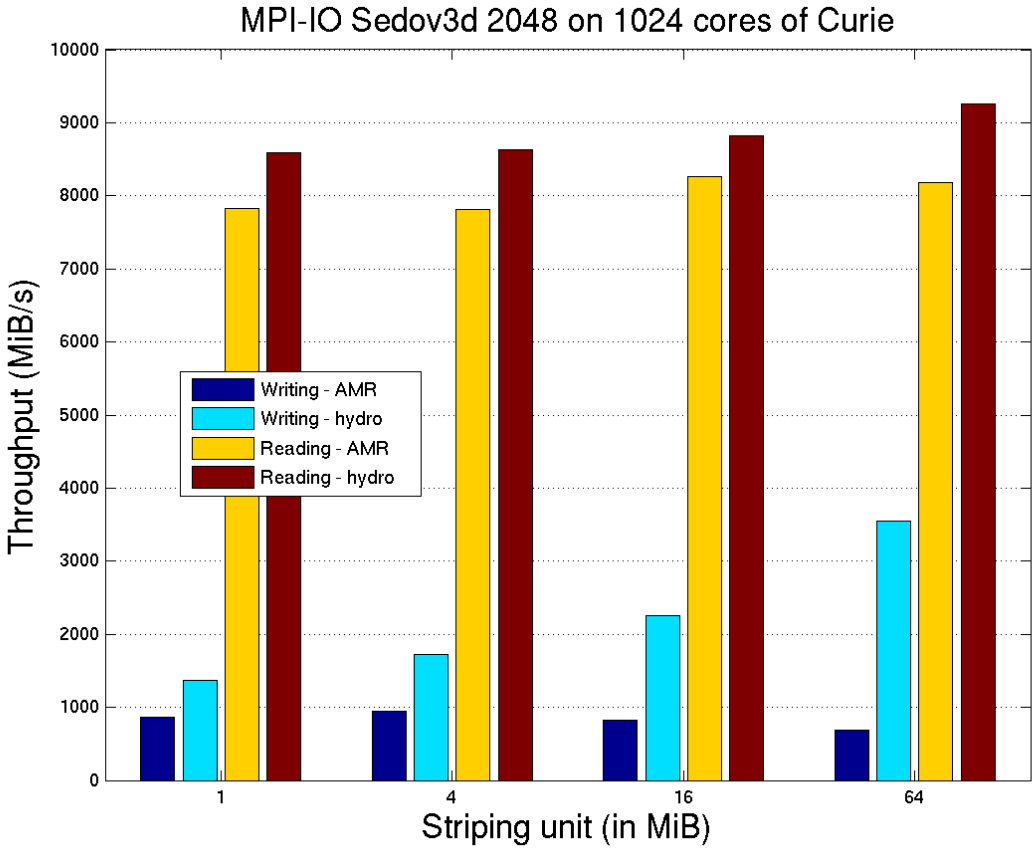
Lecture Sedov3d 1024^3 sur INTI



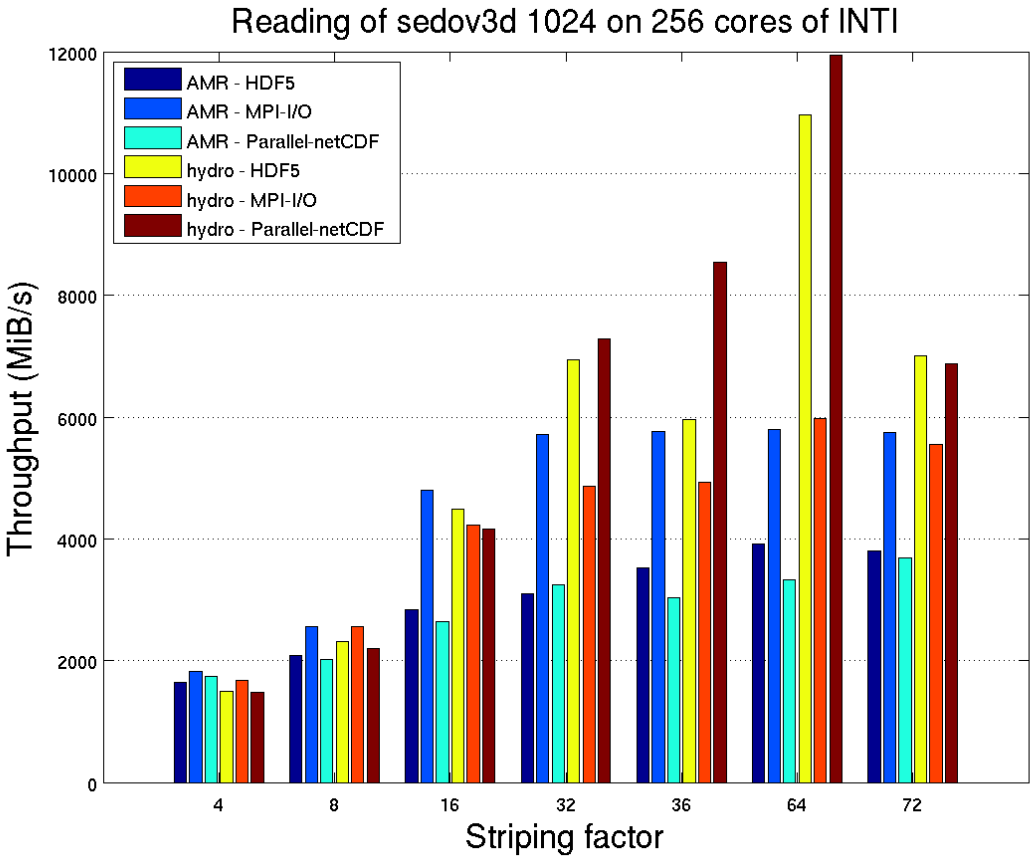
Sedov3d 2048^3 sur Curie : effet du striping_factor



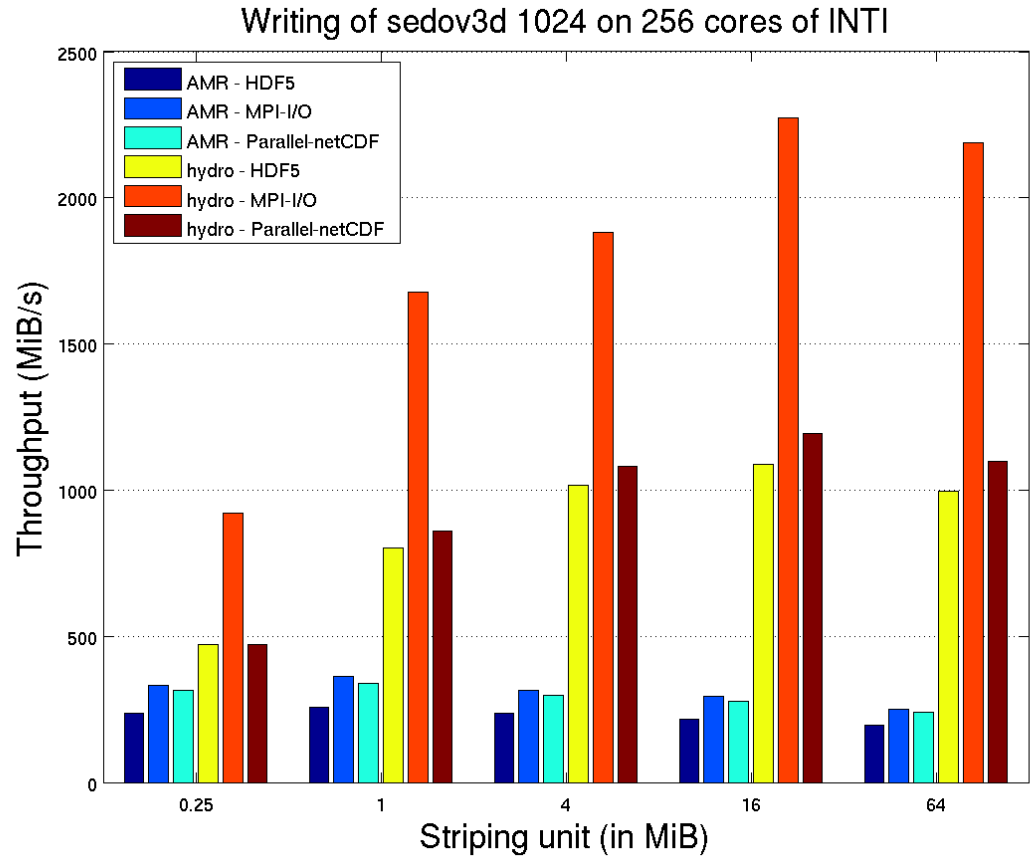
Sedov3d 2048^3 sur Curie : effet du striping_unit



Lecture Sedov3d 1024^3 sur INTI : effet du striping_factor



Ecriture Sedov3d 1024^3 sur INTI : effet du striping_unit



Conclusions RAMSES

Commentaires tests applicatifs RAMSES

- L'approche fichiers POSIX séparés est presque toujours nettement plus rapide. Avec un problème suffisamment gros et un nombre de processus élevé, le débit crête du système de fichiers peut être approché en lecture.
- Pour les 3 autres approches, HDF5 est généralement la plus performante en écriture, mais, attention, l'algorithme utilisé pour les écritures est plus optimisé (agrégation de plusieurs niveaux de maillage). En lecture, selon les cas, c'est l'approche MPI-IO ou Parallel-NetCDF qui l'emporte. HDF5 étant la plus lente.
- A taille de problème fixe, les écarts entre les différents paradigmes tendent à s'accroître avec le nombre de processus.
- Les 3 approches MPI-I/O, HDF5 et Parallel-NetCDF sont malgré tout assez proches (surtout que les tests n'ont pas été effectués en mode dédié).
- Jusqu'à 4096 processus, tous les fichiers ont pu être écrits et relus.
- Au-delà de 4096 processus, les fichiers MPI-IO n'ont pas pu être écrits. Et au-delà de 8192 processus, seuls les fichiers séparés n'ont pas posé de problèmes. Les fichiers Parallel-NetCDF et HDF5 ont été écrits, mais n'ont jamais pu être relus (ces fichiers ne sont donc peut-être pas valides).

Pistes pour améliorer les performances

Pistes

- Processus spécialisés (approche suivie par d'autres groupes)
- Structurer les données différemment
- Accès par blocs de taille minimale ou fixe (pour éviter les I/O trop petits par rapport à la taille d'un bloc du système de fichiers)
- Appels MPI-I/O non-bloquants
- Evolution des implémentations MPI-IO et des bibliothèques HDF5 et Parallel-NetCDF
- Autres bibliothèques d'I/O (SIONlib...)

Conclusions

Avertissement

Avertissement

Les conclusions sur les différentes approches d'I/O parallèles (POSIX, MPI-I/O, HDF5 et Parallel-NetCDF) présentées ici ne sont valables que :

- pour les applications utilisées (IOR et RAMSES),
- pour les architectures cibles testées (systèmes IBM x3750 Ada et Blue Gene/Q de l'IDRIS),
- avec les versions des applications, systèmes et bibliothèques à la date des tests réalisés (antérieurs à la rédaction de ce document).

De plus, de fortes variations de performance sont très fréquentes même lorsque les conditions (charge machine...) semblent stables.
Elles sont donc susceptibles de nettement différer dans d'autres conditions.

Conclusions et remarques finales

- Les besoins en entrées-sorties sont toujours croissants.
- L'approche fichiers séparés commence à montrer ses limites, même si elle reste la plus performante.
- Les paradigmes d'entrées-sorties parallèles tels que MPI-I/O, HDF5 et Parallel-NetCDF sont fonctionnels, mais souffrent encore de problèmes gênants.
- Ils permettent une gestion et une logistique simplifiée des données (fichiers en nombre plus limité, outils, portabilité garantie avec HDF5 et Parallel-NetCDF).
- Ces approches sont néanmoins complexes à mettre en œuvre (difficulté croissante de MPI-I/O à HDF5 en passant par Parallel-NetCDF).
- Les performances de celles-ci laissent à désirer. Atteindre un niveau raisonnable n'est pas trivial et nécessite un niveau d'expertise poussé.
- Des problèmes de robustesse et d'extensibilité existent sur un nombre de processus important (plusieurs milliers) avec ces paradigmes parallèles. Ce qui va à l'encontre de l'objectif de ces bibliothèques. Notons néanmoins les évolutions récentes très encourageantes.
- Un travail important de toute la communauté HPC (développeurs des bibliothèques et des applications, constructeurs et utilisateurs) est cependant encore nécessaire avant d'avoir des solutions performantes, robustes et extensibles sur les architectures massivement parallèles qui se généralisent dans tous les centres de calcul.

Documentation et liens

Documentation et liens

Pour en savoir plus

- Le site web de l'IDRIS : http://www.idris.fr
- Le site d'HDF5 : http://www.hdfgroup.org/HDF5/
- Le site de Parallel-NetCDF : http://trac.mcs.anl.gov/projects/parallel-netcdf
- Les implémentations MPI *open source* principales : OpenMPI http://www.open-mpi.org et MPICH http://www.mpich.org
- Le logiciel de benchmark IOR : http://sourceforge.net/projects/ior-sio/
- RAMSES : http://www.itp.uzh.ch/ teyssier/ramses/RAMSES.html. Attention, la version avec I/O parallèles n'est pas disponible sur ce site. Contactez l'auteur de cette présentation pour plus d'informations.
- Une liste de benchmarks est proposée sur ce site : http://www.mcs.anl.gov/ thakur/pio-benchmarks.html
- Le site de netCDF : http://www.unidata.ucar.edu/software/netcdf/index.html
- La bibliothèque ADIOS : http://adiosapi.org/index.php5 ?title=Main_Page
- La bibliothèque SIONlib : http://www.fz-juelich.de/jsc/sionlib/