

# Les logiciels de chimie à l'IDRIS



Fabien Leydier  
28 juin 2013



## Objectifs de la présentation

- ▶ Donner une vision d'ensemble des logiciels de chimie
  - Nombreuses possibilités offertes par TOUS les logiciels
  - Utilisation par habitude...
  - Ici, pas de jugement subjectif !
- ▶ Comment bien (mieux ?) utiliser les logiciels de chimie
  - Ils sont TOUS parallèles (plus ou moins...)
  - Ils comportent souvent des options spécifiques (parfois « oubliées » car peu ou mal documentées...)
  - La plupart donnent des informations sur la parallélisation ! (bien souvent négligées...)
- ▶ Calculer à l'IDRIS
  - Comprendre ce qui se passe sur les machines
  - Construire un script de soumission adapté
- ▶ Présentation non exhaustive (malheureusement)
  - Mais quelques conseils pratiques !



# Sommaire

- ▶ Présentation de l'IDRIS
- ▶ Présentation des comités thématiques
- ▶ Parallélisme : de la Machine à la Chimie
- ▶ Les logiciels de chimie à l'IDRIS
  - Description des possibilités (propriétés, utilisation, etc.)
  - Comment calculer à l'IDRIS
  - Utilisation des logiciels

## Présentation de l'IDRIS

# L'IDRIS en deux mots

- ▶ *Institut de Développement et des Ressources en Informatique Scientifique*
- ▶ Unité propre de service du CNRS (UPS 851), fondée fin 1993
- ▶ Centre de calcul équipé de supercalculateurs parmi les plus puissants du moment
  - Centre d'excellence dans le domaine du Calcul de Haute Performance (HPC)
  - Puissance pétaflopique depuis janvier 2013
  - Équipements financés par GENCI depuis 2012
    - *Grand Équipement National de Calcul Intensif*
    - Coordonne la gestion et l'attribution des heures sur les machines



## Calculer à l'IDRIS

### *Comment obtenir des heures de calcul ?*

- ▶ [www.edari.fr](http://www.edari.fr)
  - **2 sessions d'attribution par an (octobre – avril)**
    - Constitution d'un dossier décrivant le projet scientifique
    - Analyse des demandes par les Comités thématiques
  - **Demande au « fil de l'eau »**
    - Pour compléter la demande en cours d'année
    - Limitée à 10 % de l'allocation initiale
  - **Accès préparatoire**
    - Compte de test, en vue éventuellement d'une demande ultérieure
    - Allocation forfaitaire : 15000 heures sur Ada, 50000 heures sur Turing



# PRESENTATION DES CT

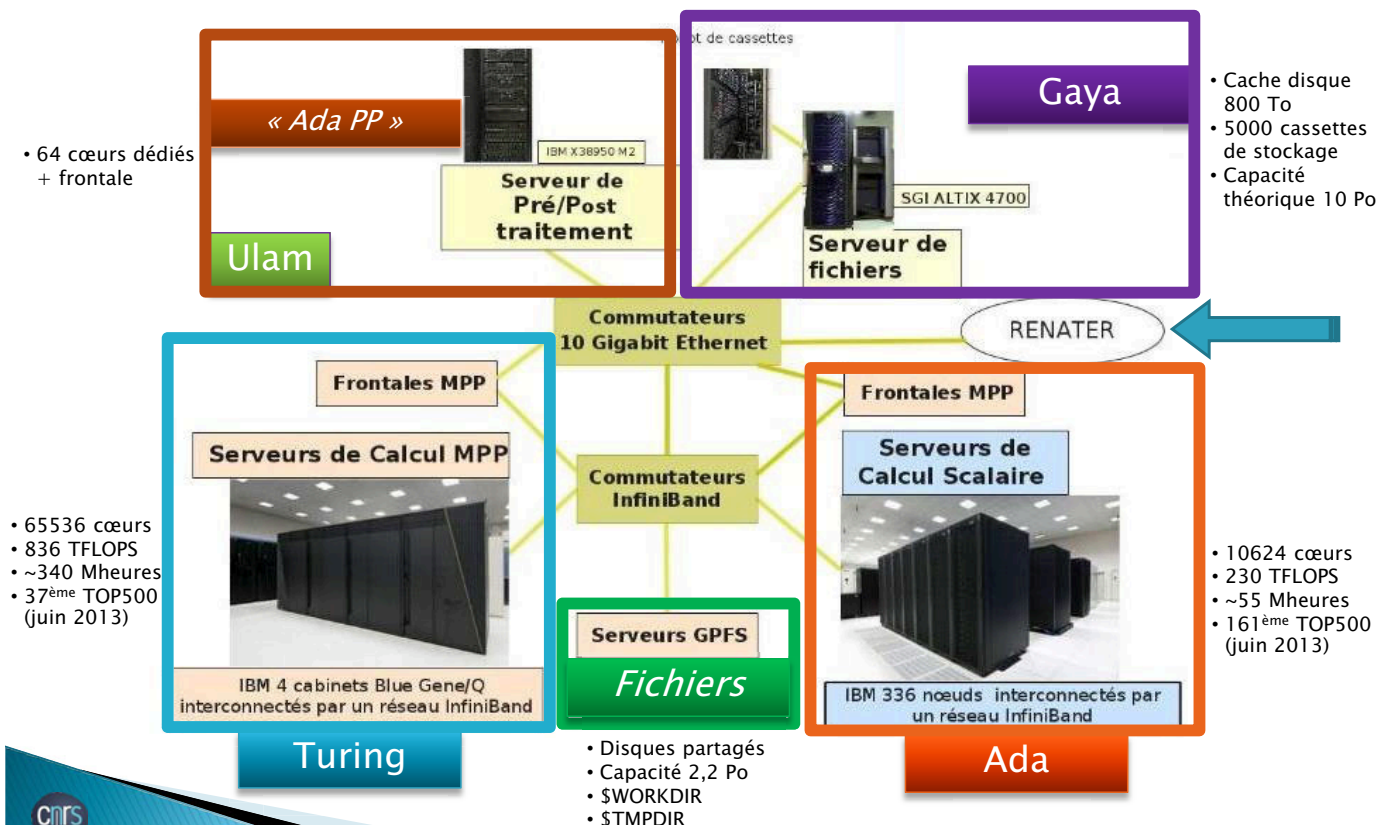
- ▶ **Comités Thématiques liés à la chimie**
  - **CT 7 : Systèmes moléculaires organisés et biologie**
    - Yves-Henri Sanejouand
      - [yves-henri.sanejouand@univ-nantes.fr](mailto:yves-henri.sanejouand@univ-nantes.fr)
  - **CT 8 : Chimie quantique et modélisation moléculaire**
    - Marie-Bernadette Lepetit
      - [marie-bernadette.lepetit@grenoble.cnrs.fr](mailto:marie-bernadette.lepetit@grenoble.cnrs.fr)
  - **CT 9 : Physique, chimie et propriétés des matériaux**
    - Alain Pasturel
      - [alain.pasturel@grenoble.cnrs.fr](mailto:alain.pasturel@grenoble.cnrs.fr)



## Le Parallélisme : De la Machine à la Chimie



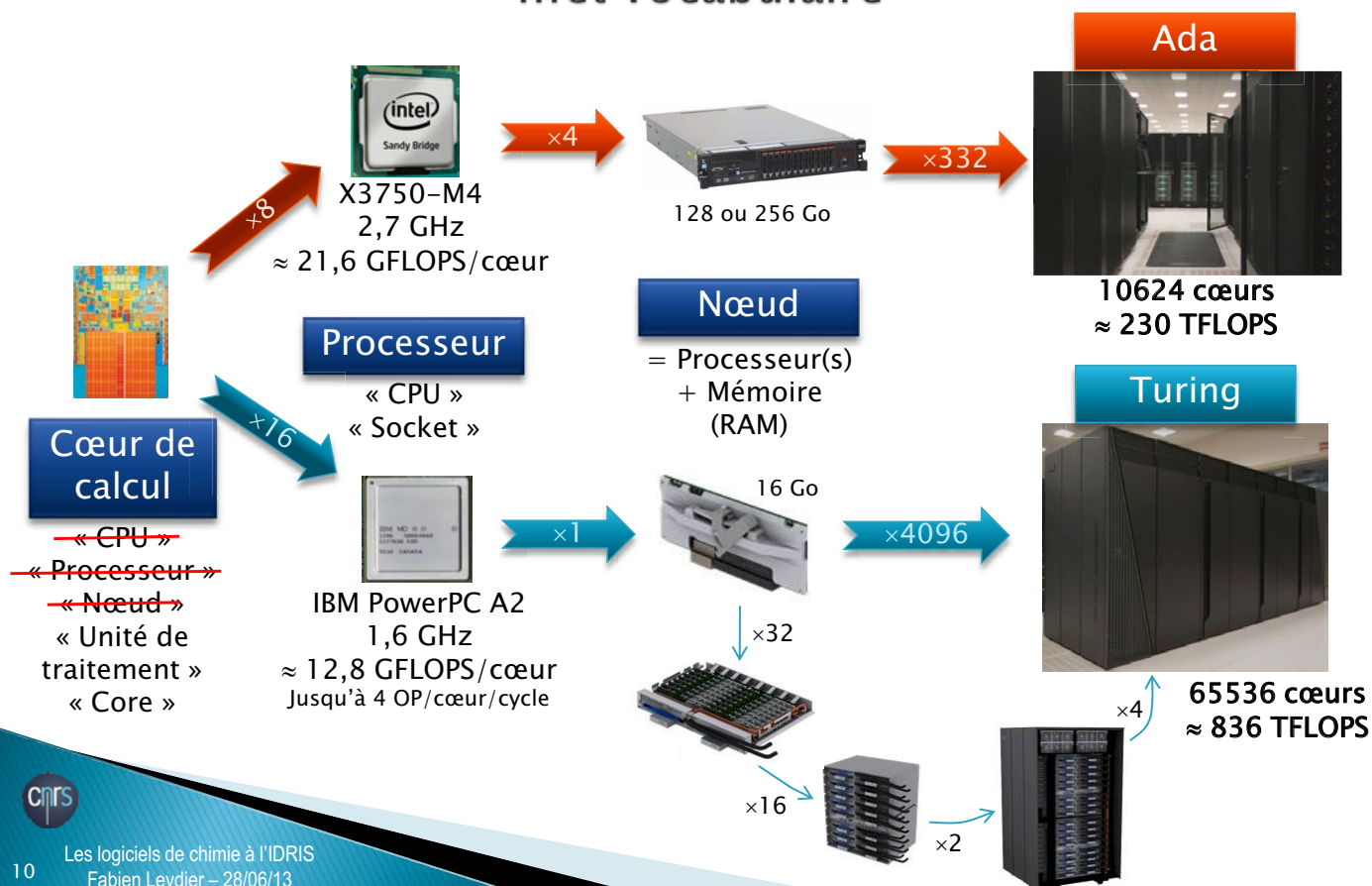
# Machines de l'IDRIS



Les logiciels de chimie à l'IDRIS  
Fabien Leydier – 28/06/13

9

## Architecture des machines de calcul... ...et vocabulaire



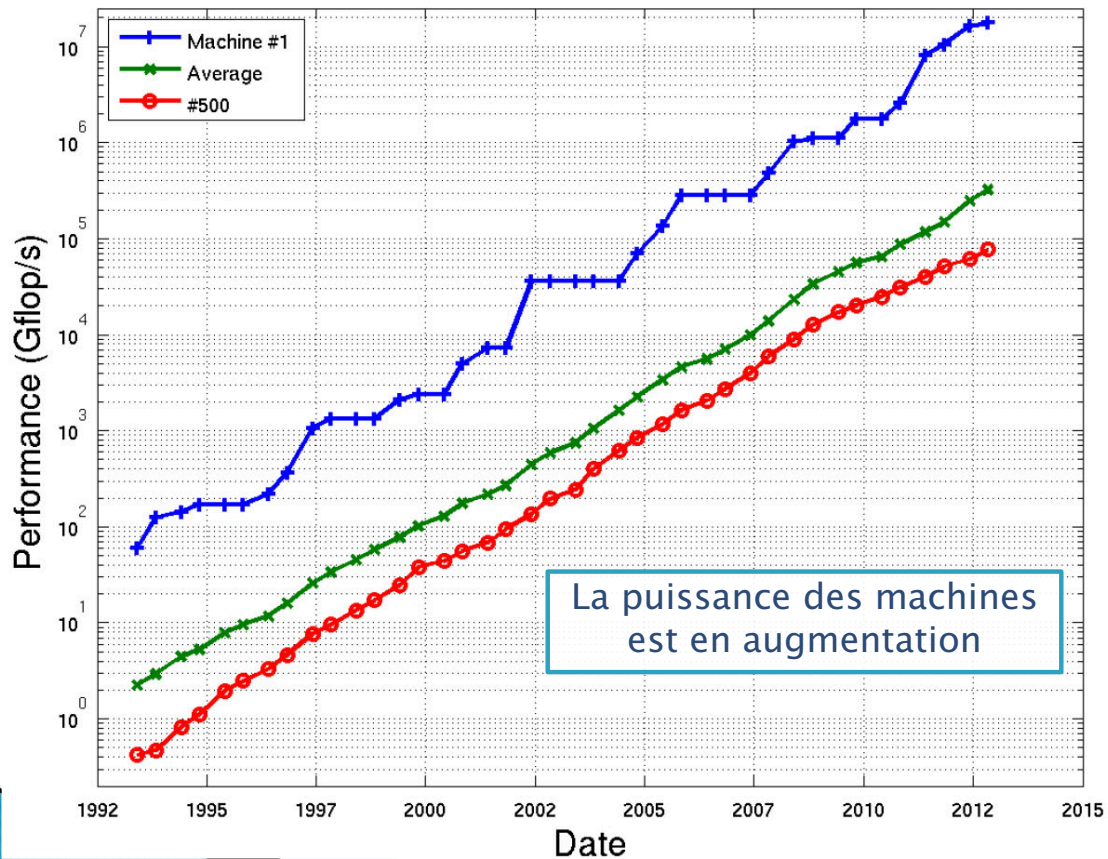
Les logiciels de chimie à l'IDRIS  
Fabien Leydier – 28/06/13

10



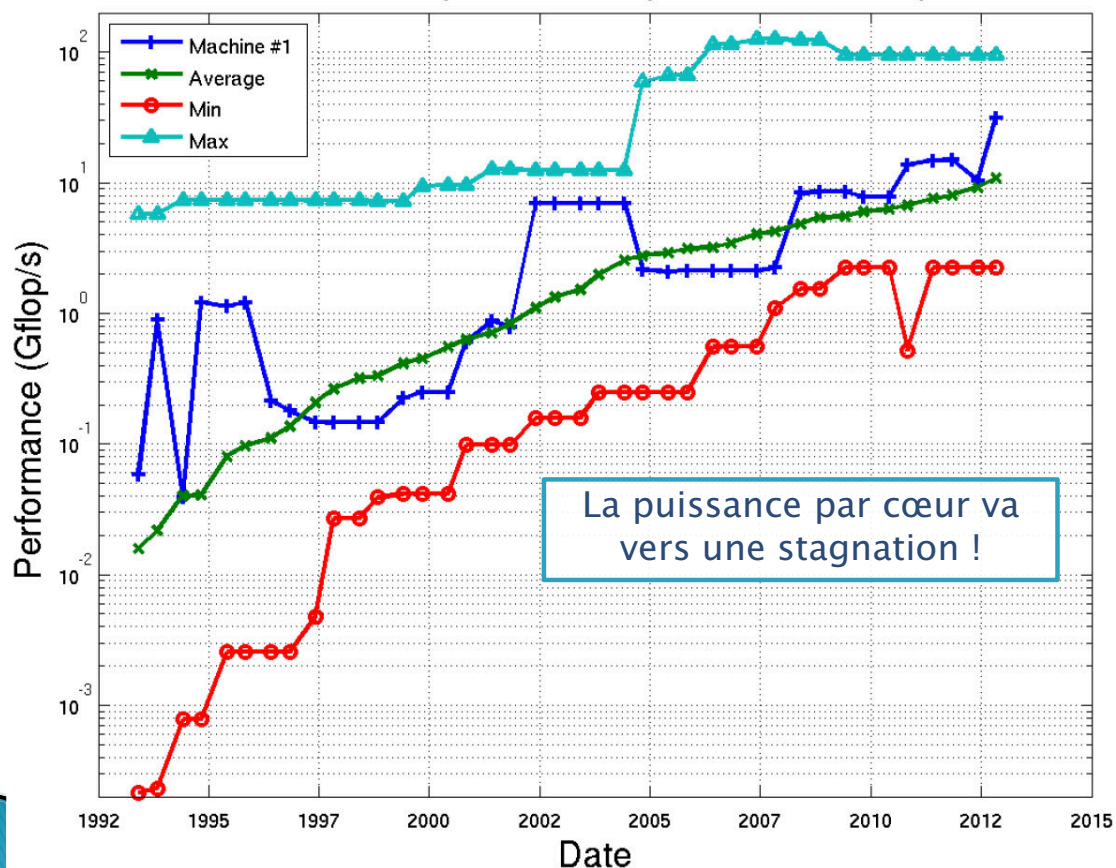
# Le parallélisme : origine et problématique

Evolution of the performance in the Top500



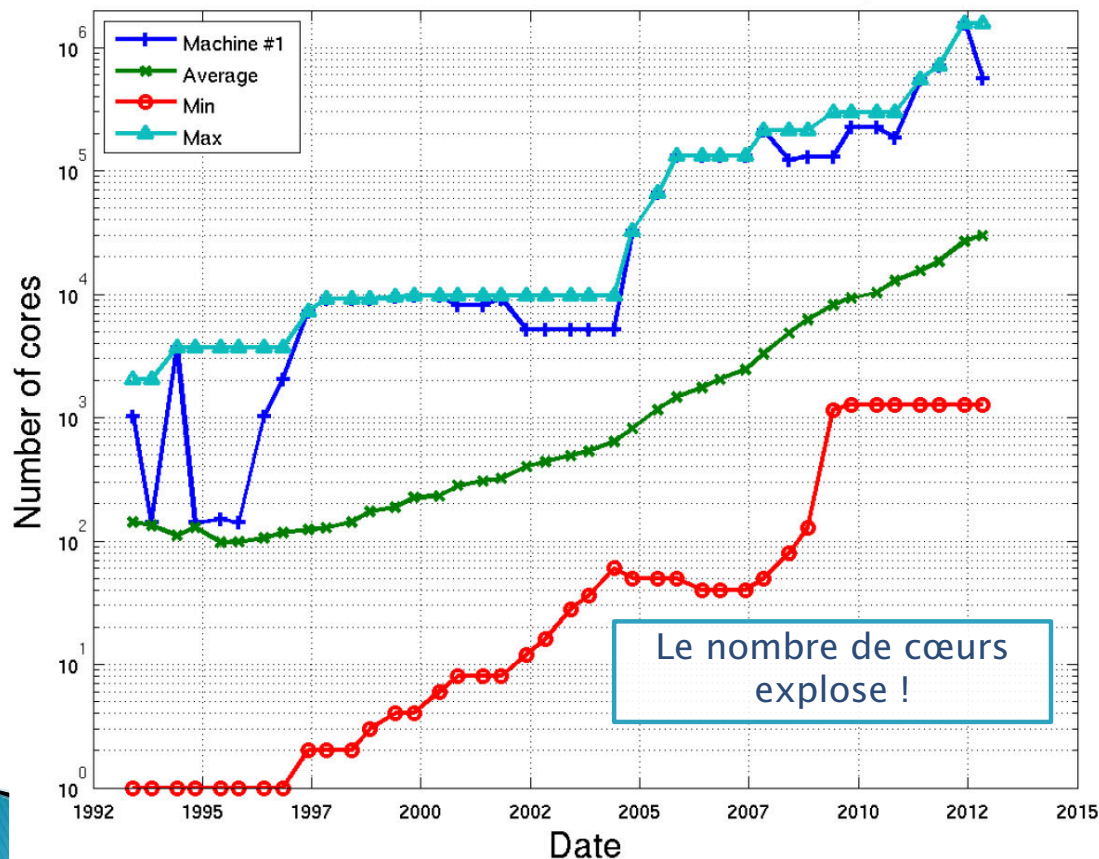
# Le parallélisme : origine et problématique

Evolution of the performance per core in the Top500



# Le parallélisme : origine et problématique

Evolution of the number of cores in the Top500



# Le parallélisme : origine et problématique

## ► Evolution des processeurs

- La puissance par cœur évolue faiblement (plafonnement de la fréquence, etc.)  
⇒ **Solution : augmenter le nombre de cœurs**

## ► Mémoire

- Augmentation rapide du nombre de cœurs  
⇒ **La mémoire par cœur stagne, voire diminue**

## ► Codes séquentiels (pas seulement la chimie...)

- Il ne suffit plus d'attendre l'évolution de la puissance des cœurs pour calculer plus vite
- Paralléliser un code séquentiel ?
  - Non pensé pour le parallélisme à l'origine
    - Demande une réécriture en grande partie (voire complète)
    - Demande énormément de ressources (temps, compétences)
    - Aucune garantie absolue sur les performances parallèles

# Parallélisme dans la chimie quantique

Découpage du système	Fonction d'onde et densité $e^-$	Fonctions mathématiques
<ul style="list-style-type: none"> <li>• Multiples images (NEB, ...)</li> <li>• Points k</li> </ul>	<ul style="list-style-type: none"> <li>• Coefficients de la base de projection (vecteurs d'onde, gaussiennes, etc.)</li> <li>• Bandes, calcul de la densité</li> </ul>	<ul style="list-style-type: none"> <li>• Fonctions (FFT, etc.)</li> <li>• Solveurs de valeurs propres</li> <li>• Manipulation de matrices</li> </ul>

- **Découpage du système**
  - Calcul d'éléments unitaires pour un calcul
  - Répartition des calculs unitaires en parallèle = très efficace
- **Fonction d'onde**
  - Cœur du calcul
  - Nombre de données à traiter parfois énorme
- **Fonctions mathématiques**
  - Reposent souvent sur des bibliothèques spécialisées
  - Génèrent beaucoup de communications (car beaucoup de données à traiter)

# Parallélisme dans la chimie quantique

Découpage du système	Fonction d'onde et densité $e^-$	Fonctions mathématiques
<ul style="list-style-type: none"> <li>• Multiples images (NEB, ...)</li> <li>• Points k</li> </ul>	<ul style="list-style-type: none"> <li>• Coefficients de la base de projection (vecteurs d'onde, gaussiennes, etc.)</li> <li>• Bandes, calcul de la densité</li> </ul>	<ul style="list-style-type: none"> <li>• Fonctions (FFT, etc.)</li> <li>• Solveurs de valeurs propres</li> <li>• Manipulation de matrices</li> </ul>

## Complexité du parallélisme

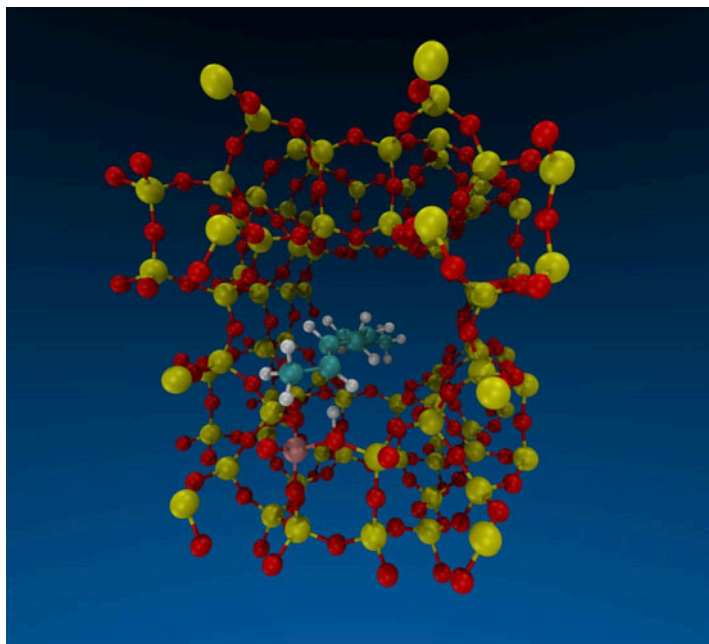
- **Parallélisme de plus en plus complexe à optimiser**
  - Algorithmes de symétrie
    - Réduction de la taille effective du système
    - But : aller vers des calculs de fonctions intrinsèquement parallèles
  - « Fonction d'onde et densité » → « Fonctions mathématiques »



# Que peut m'apporter le parallélisme ?

## ► Durée d'une simulation plus importante

H-MOR + 2,6-hexdiène (311 atomes), PBE, 30 Ry,  $\Gamma$ , dynamique BO @ 600K,  $\Delta t = 0,7$  fs, 20h de calcul, CPMD

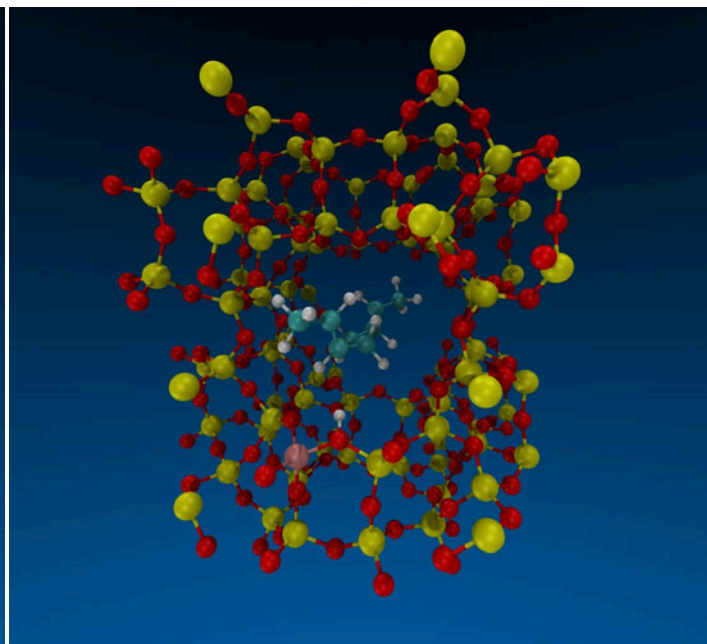


4 cœurs  
16 pas

33,8 jours

Temps de  
restitution  
pour 648 pas

0,8 jour



512 cœurs  
648 pas



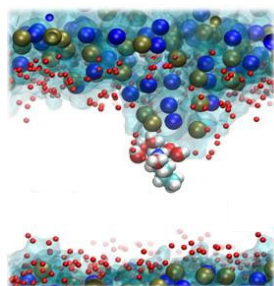
Les logiciels de chimie à l'IDRIS  
Fabien Leydier – 28/06/13

17

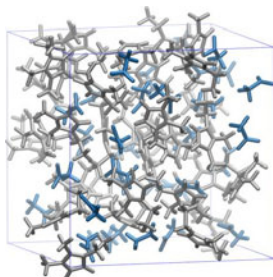
# Que peut m'apporter le parallélisme ?

## ► Traitement d'un système de taille importante

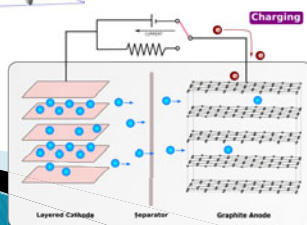
- Simulations « Grands challenges » réalisées à l'IDRIS



- Étude de transferts d'acides aminés au travers de membranes biologiques (lysines hydratées)
- 15000 atomes
- Code GROMACS sur Ada
- D. Bonhenry, F. Dehez, M. Tarek



- Simulation d'un liquide ionique
- 900 - 10000 atomes
- Codes CP2K et CPMD sur Ada
- N. Sieffert



- Modélisation des matériaux d'anodes pour batteries ion-lithium
- 208 - 625 atomes
- Code BigDFT sur Turing
- S. Krishnan, D. Caliste, T. Deutsch, L. Genovese, P. Pochet

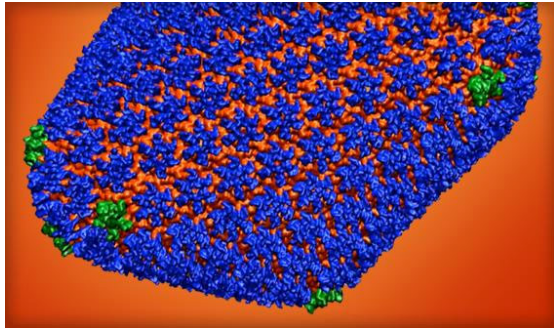


Les logiciels de chimie à l'IDRIS  
Fabien Leydier – 28/06/13

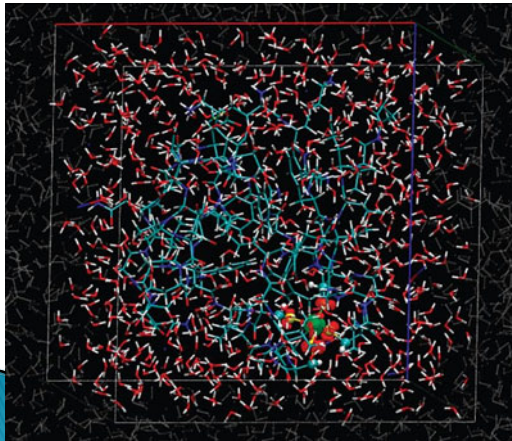
18

# Que peut m'apporter le parallélisme ?

## ► Traitement d'un système de taille importante



- Capsule entière du virus VIH, assemblage de plus de 1300 protéines modélisées à l'échelle atomique
- 64,4 millions d'atomes
- Code NAMD, 128000 cœurs sur Blue Waters (National Center for Supercomputing Applications, Université de l'Illinois), Cray XE, 11,5 PFLOPS
- K. Schulten, J. R. Perilla, P. Zhang *et al.*, *Nature*, 497, 2013, 643.



- Protéine de rubredoxine hydratée (B3LYP + base triple- $\zeta$  polarisée)
- 2825 atomes (26,7 nm<sup>3</sup>)
- Code CP2K, 8196 cœurs sur Jaguar (Oak Ridge National Laboratory), Cray XT5, 1,75 PFLOPS
- M. Guidon, J. Hutter, J. VandeVondele, *J. Chem. Theory, Comput.*, 5, 2009, 3010.



Les logiciels de chimie à l'IDRIS  
Fabien Leydier – 28/06/13

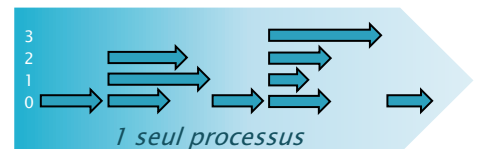
19

## Types de parallélisation

### ► OpenMP

#### ◦ Open Multi-Processing : « multi-tâches »

- Mémoire partagée sur un nœud
  - Programme répartissant le calcul entre des *threads* (ou tâches, processus légers, etc.)
    - 1 *thread* : portion du calcul attribuée à 1 cœur
    - Chaque *thread* a accès à toute la mémoire allouée au programme
  - 1 seul processus lancé
    - Programmation voisine d'un code séquentiel
    - Un programme peut être « facilement » parallélisable
      - Voire même automatiquement (compilateurs)
- Programme exécutable en intra-nœud seulement
  - Echanges directement dans la mémoire du nœud
  - Extensibilité limitée au nombre de cœurs composant un nœud



Les logiciels de chimie à l'IDRIS  
Fabien Leydier – 28/06/13

20

# Types de parallélisation

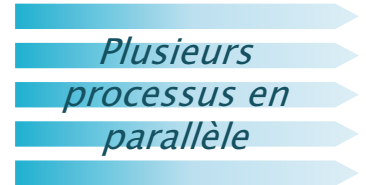
## ► MPI

### ◦ Message-Passing Interface : « Passage de messages »

- Echange d'informations entre des processus MPI (ou « tâches MPI »)

- Plusieurs processus lancés simultanément

- Processus MPI : code exécuté sur 1 cœur + mémoire associée
- Nécessite une programmation explicite, très spécifique
- Chaque processus ne fait pas forcément le même calcul



- Programme exécutable en intra- et extra-nœud

- Processus intra-nœud

- Echanges dans la mémoire du nœud



- Processus extra-nœud

- Echanges à travers un réseau très haute performance



Les logiciels de chimie à l'IDRIS  
Fabien Leydier – 28/06/13


21

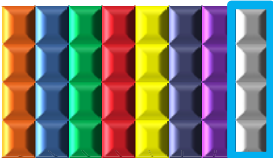
# Types de parallélisation

## ► Hybride MPI-OpenMP

### ◦ Combiner les deux concepts

MPI :  1 processus / cœur, plusieurs processus simultanés

OpenMP :  1 thread / cœur  
1 seul processus

Hybride :  1 processus MPI = plusieurs threads OpenMP  
= plusieurs cœurs

Processus MPI multiples

But : obtenir une parallélisation plus efficace, adaptée à l'architecture des machines

→ Avantage : extensibilité potentiellement accrue

→ Inconvénient : programmation hybride délicate



Les logiciels de chimie à l'IDRIS  
Fabien Leydier – 28/06/13

22



# Les limites du parallélisme

## ► Pourquoi limiter le nombre de cœurs ?

- Lois d'Amdahl et de Gustavson-Barsis
  - L'extensibilité n'est pas infinie par définition
    - Les communications, c'est-à-dire les échanges de données, deviennent non négligeables devant le temps consacré aux calculs
    - Efficacité parallèle moindre
    - Le temps de restitution ne diminue pas de façon linéaire...et peut même augmenter !
- Une fois le système découpé en ses plus petits éléments, il n'y a plus rien à partager
  - Des cœurs n'ont aucune charge de calcul
- OpenMP limité au nombre de cœurs d'un nœud

## ► Alors pourquoi en prendre plusieurs ?

- Temps de restitution diminué
- Calculer de plus gros systèmes



# Les limites du parallélisme

## ► Pourquoi ne pas augmenter la taille du système ?

- *Chemically irrelevant*
  - Parfois impossible sur un système moléculaire donné
  - Peu d'intérêt à obtenir  $n$  fois le même résultat
- Le temps de restitution sera plus important
  - Avec plus de cœurs, la « facture » augmente d'autant plus
- Certains algorithmes sont optimisés pour la symétrie
  - Activer la symétrie revient (presque) à étudier le système initial
  - Désactiver la symétrie revient à ne pas utiliser ces fonctions d'optimisation

## ► Alors pourquoi cette idée ?

- « Mieux » utiliser les ressources (loi de Gustavson-Barsis)
- Par exemple, sonder plus de configurations lors d'une dynamique



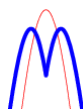
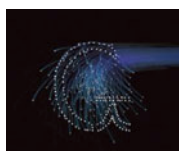


# Conclusion

- ▶ **La parallélisation est essentielle aujourd'hui**
  - Architecture des machines
  - Taille des simulations de plus en plus importante
- ▶ **L'extensibilité est la clé pour les performances**
  - Dépend de la programmation du code en lui-même
  - Dépend de la taille et de la complexité du système étudié
  - Il faut parfois accepter un surcoût en heures
    - Pour obtenir un résultat plus rapidement
    - Pour que son système tienne dans la mémoire de la machine



## Les logiciels à l'IDRIS



ADF

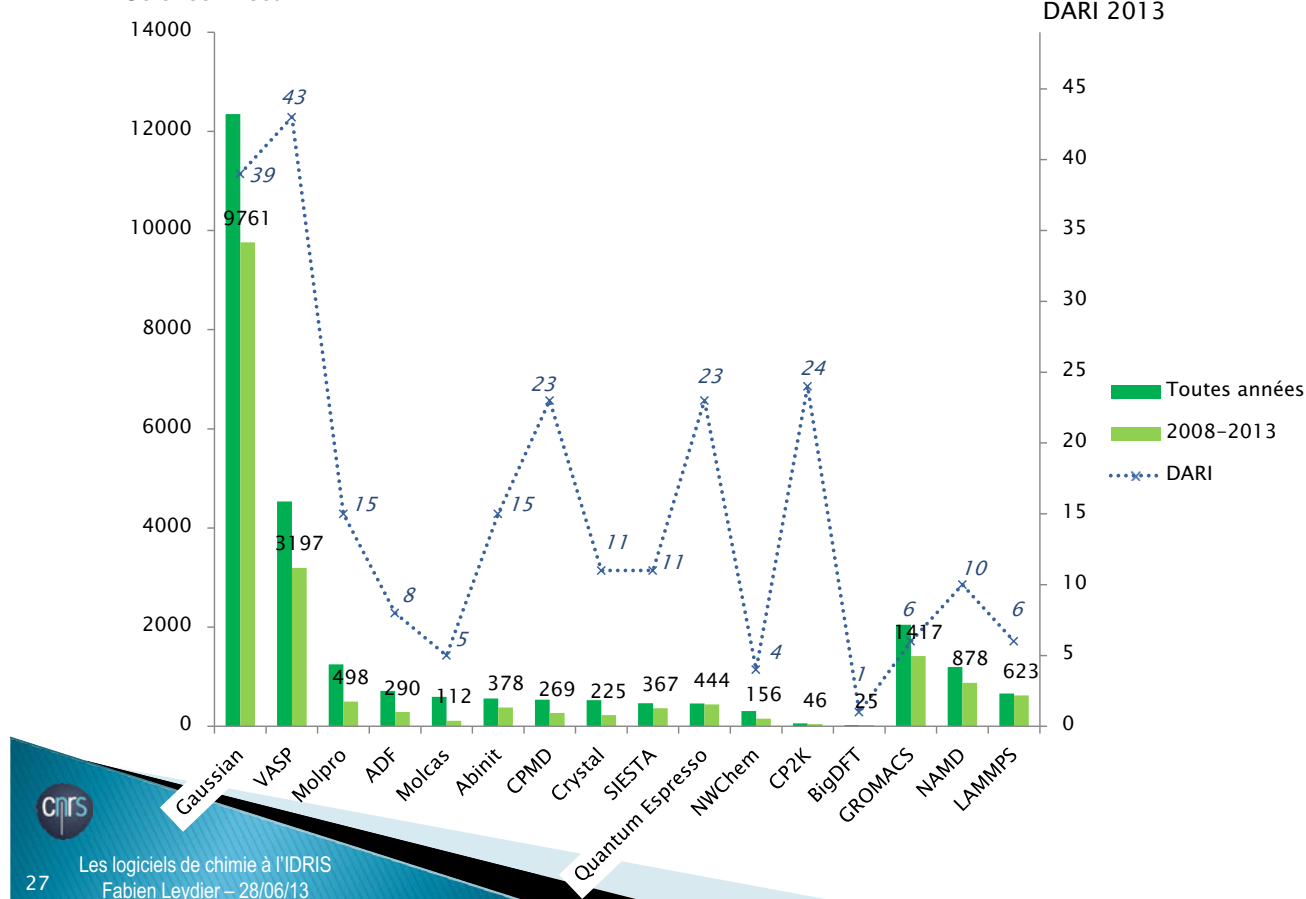


- ▶ 16 logiciels disponibles

# Popularité des logiciels

Nombre de publications  
ScienceDirect

Nombre de dossiers  
DARI 2013



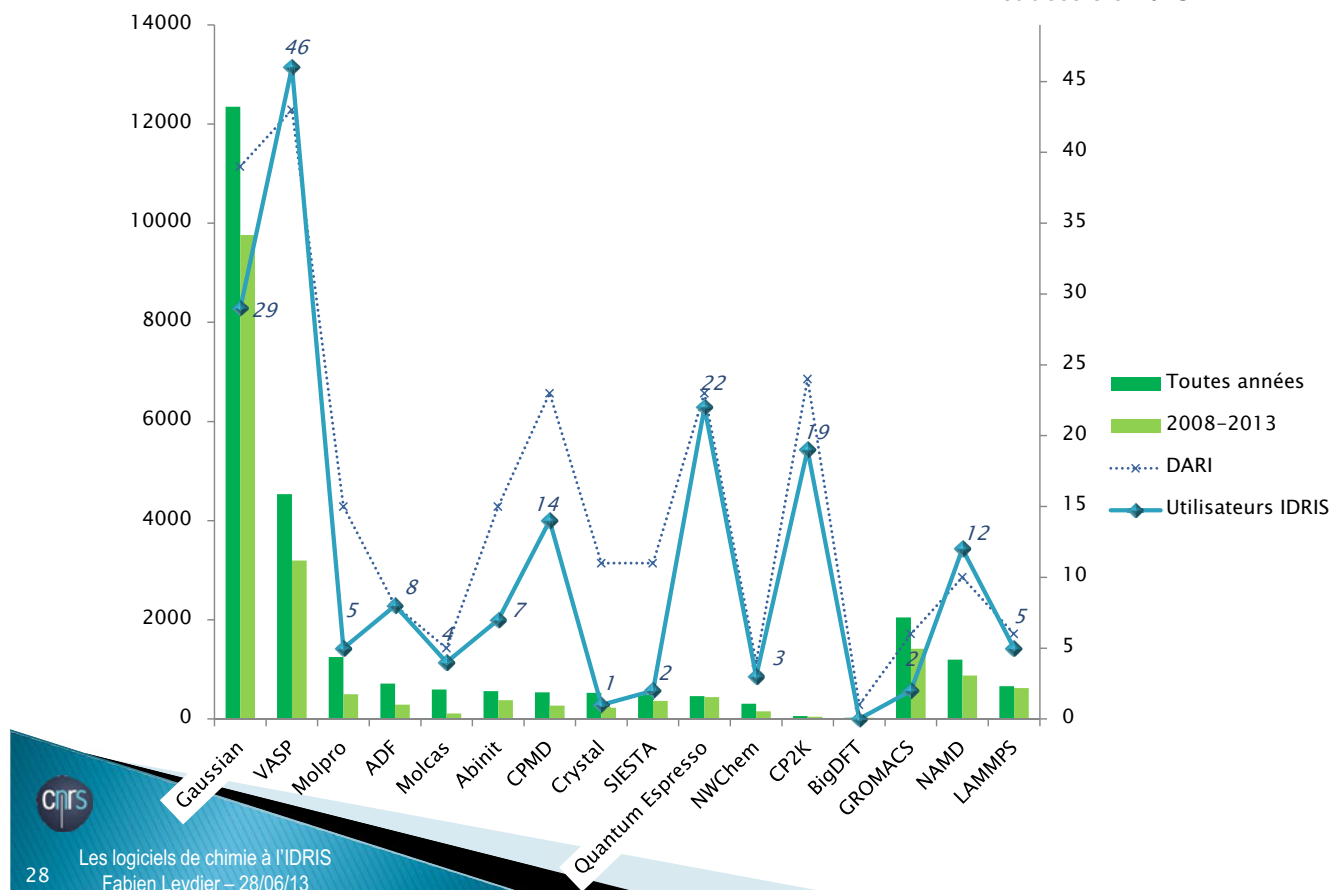
27

Les logiciels de chimie à l'IDRIS  
Fabien Leydier – 28/06/13

# Popularité des logiciels

Nombre de publications  
ScienceDirect

Nombre d'utilisateurs  
et dossiers 2013



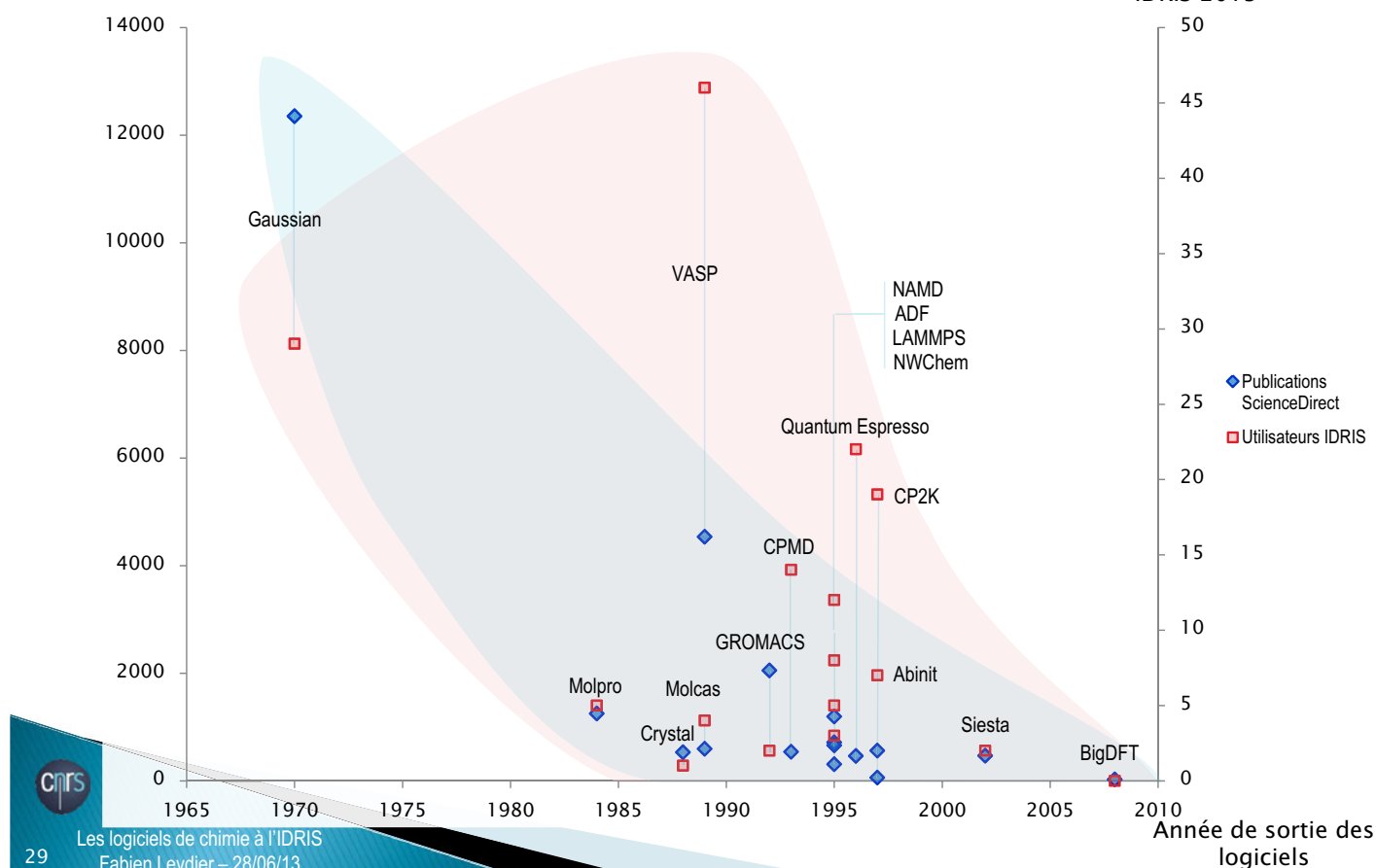
28

Les logiciels de chimie à l'IDRIS  
Fabien Leydier – 28/06/13

# Popularité des logiciels

Nombre de publications  
ScienceDirect

Nombre d'utilisateurs  
IDRIS 2013



## Utilisateurs, Licences, Prix

Accès  
 ■ = Libre  
 ■ = A demander  
 ■ = Licence exigée

Logiciels	Nombre de dossiers	Nombre d'utilisateurs		Prix	Type de licence à l'IDRIS	Type de fichiers fournis	Accès à l'IDRIS
		Ada	Turing				
Gaussian	39	29	–	28750\$ + 575\$	Site / Version / 20 ans	Source	ACL
VASP	43	46	–	1000€ / groupe	Maintenance / Version	Source	ACL
Molpro	15	5	–	10000£	Site / Version	Source	Libre
ADF	8	8	–	6400€	Site / Version / 1 an	Bin	Libre
Molcas	5	4	–	50000SEK (5900€)	Site / Version / 1 an	Source	Libre
Abinit	15	7	–	0	GNU GPL	Source	Libre
CPMD	23	11	3	0	Site	Source	Libre
Crystal	11	1	0	4800€	Site / Version	*.o	ACL
SIESTA	11	2	–	0	Site	Source	ACL
Quantum Espresso	23	20	2	0	GNU GPL	Source	Libre
NWChem	4	3	–	0	ECL	Source	Libre
CP2K	24	15	4	0	GNU GPL	Source	Libre
BigDFT	1	–	–	0	GNU GPL	Source	Libre
GROMACS	6	2	0	0	GNU GPL	Source	Libre
NAMD	10	10	2	0	GNU GPL	Source	Libre
LAMMPS	6	3	2	0	GNU GPL	Source	Libre

# Importance de l'installation

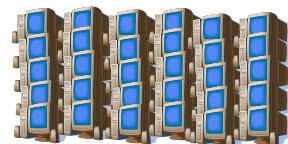
## ► Sur un PC de laboratoire

- Environnement testé par les développeurs
- Installation « classique »
  - Méthode d'installation livrée avec le logiciel conçue pour un environnement standard



## ► Sur un supercalculateur

- Environnement spécifique
  - Chemins des dossiers non standard, dossiers partagés, bibliothèques spécifiques, etc.
- Installation « non classique »
  - Si prévue par les développeurs : installation facilitée, options de compilation à tester
  - Si non prévue : peut devenir laborieux, options de compilation à adapter et à tester
- Exemple illustratif



Quantum Espresso v5.0.1	Optimisations désactivées, bibliothèques internes	Options par défaut, bibliothèques internes	Optimisé, bibliothèques externes	Sur-optimisé, bibliothèques externes
Options de compilation	-O0	-O2	-O2, -xAVX, mkl	-O3, -xAVX, -parallel, mkl
Temps elapsed (MOR, PBE, Γ, 10 pas)	3 h 3 min	20 min 28 s	6 min 6 s	9 min 0 s

(32×8 cœurs)



Les logiciels de chimie à l'IDRIS  
Fabien Leydier – 28/06/13

31

# Formalisme et parallélisation

■ = Installé à l'IDRIS  
■ = Pas de GPU à l'IDRIS  
■ = En cours d'étude

Logiciels	Nombre d'utilisateurs		Prix	Formalisme de calcul		Type de parallélisation			
	Ada	Turing		Chimie Quantique (DFT, ...)	Système de particules (champs de forces, biochimie, etc)	OpenMP	MPI	MPI + OpenMP	GPU
Gaussian	29	–	28750\$ +575\$	✓		★ ★ ★ ★ ✓	★ ★ ★ ★	★ ★ ★ ★	■
VASP	46	–	1000€ / groupe	✓			✓		■
Molpro	5	–	10000£	✓			✓		■
ADF	8	–	6400€	✓			✓		■
Molcas	4	–	50000SEK (5900€)	✓			✓		■
Abinit	7	–	0	✓		✓	✓	■	✓
CPMD	11	3	0	✓		✓	✓	✓	■
Crystal	1	0	4800€	✓			✓		■
SIESTA	2	–	0	✓			✓		■
Quantum Espresso	20	2	0	✓		✓	✓	✓	✓
NWChem	3	–	0	✓			✓		■
CP2K	15	4	0	✓		✓	✓	✓	■
BigDFT	–	–	0	✓		✓	✓	✓	✓
GROMACS	2	0	0		✓		✓		✓
NAMD	10	2	0		✓	✓	✓	■	✓
LAMMPS	3	2	0		✓		✓		✓

31

Fabien Leydier – 28/06/13



# Propriétés et spécificités – chimie quantique

Logiciel	Dimension des systèmes	Base de projection	Fonctionnelles	Nombre de possibilités
Gaussian	0, 1, 2, 3D	GTO	~46 (dont hybrides)	★★★★★
VASP	3D	PW	~11 (dont hybrides)	★★★★★
Abinit	3D	PW	25 (+ LibXC)	★★★★★
Quantum Espresso	3D	PW	~25 + hybrides	★★★★★
CPMD	0, 1, 2, 3D	PW	~19 (dont hybrides)	★★★★★
ADF	0, 1, 2, 3D	STO	~28 (dont hybrides)	★★★★★
Molpro	0D	GTO	~50 (dont hybrides)	★★★★★
CP2K	0, 1, 2, 3D	GTO+PW	~25 (+LibXC)	★★★★★
SIESTA	3D	NAO	~8	★★★★★
Molcas	0D	GTO	~33 (dont hybrides)	★★★★★
Crystal	0, 1, 2, 3D	GTO	~19 (dont hybrides)	★★★★★
NWChem	0, 1, 2, 3D	GTO	~82 (dont hybrides)	★★★★★
BigDFT	0, 1, 2, 3D	ondelettes	25 (+ LibXC)	★★★★★

Nombre Hybrides 3D fonctionnelles

33

Les logiciels de chimie à l'IDRIS  
Fabien Leydier – 28/06/13

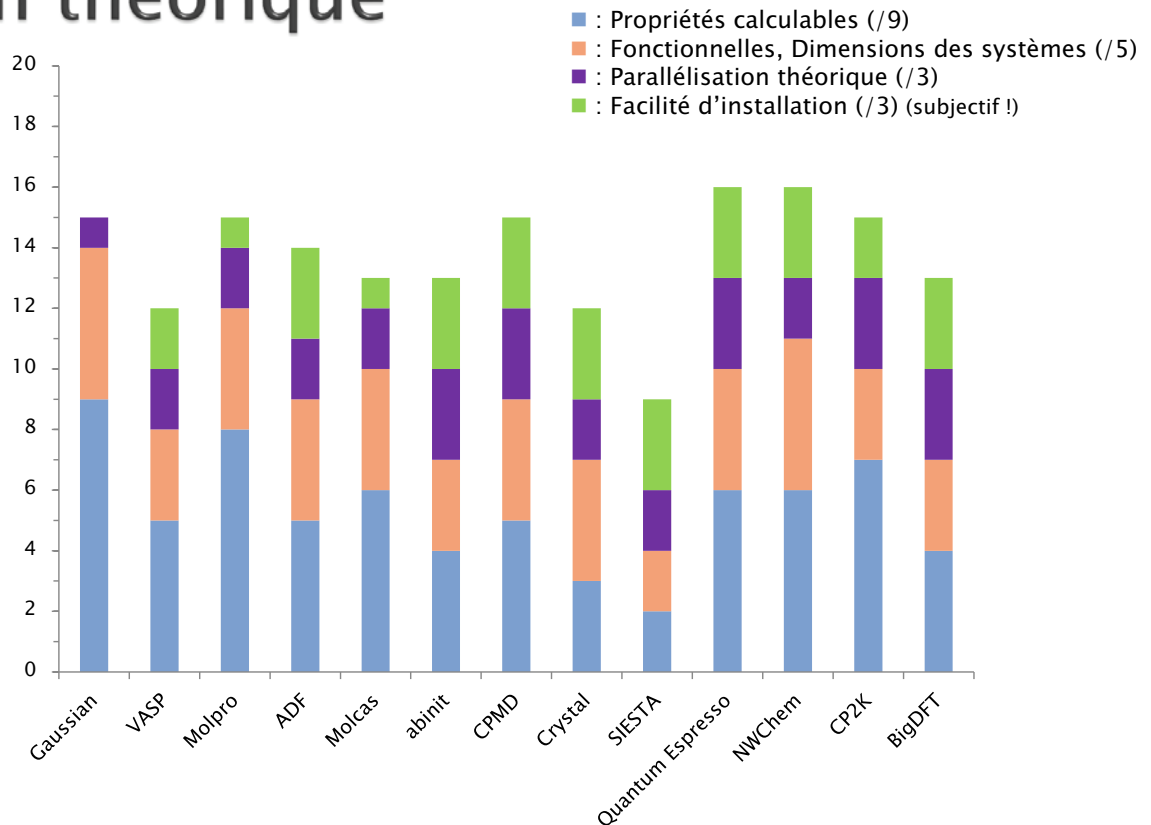
# Propriétés et spécificités – chimie quantique

Logiciel	Vibrations	TDDFT	Dynamique	QM/MM	MP2	RMN	Etat de transition	Autres
Gaussian	+ Anharm.			ONIOM				Magnétisme, solvant, dichroïsme, UV/Vis
VASP								
Abinit								
Quantum Espresso								EPR, spectres X
CPMD								
ADF								UV/Vis, solvant, Mosbauer, spectres X, magnétisme, EPR
Molpro	+ Anharm.							MCSCF, transitions e <sup>-</sup>
CP2K								EPR
SIESTA								
Molcas								MCSCF, solvant
Crystal	+ Anharm.							
NWChem				ONIOM				solvant, MCSCF, transitions e <sup>-</sup>
BigDFT								XANES

34

Les logiciels de chimie à l'IDRIS  
Fabien Leydier – 28/06/13

# Bilan théorique



- Globalement, tous les logiciels se valent en théorie
  - Mais certains restent spécifiques

## Logiciels orientés biochimie et champs de force

- Paramètres critiques pour les performances
  - Un système typique est composé de milliers (voire millions !) de particules
    - Décomposition du système étudié
    - Répartition sur les cœurs de calcul
  - Implémentation de méthodes de calcul
    - Particule Mesh Ewald, etc.

# Retour d'expériences

Logiciel	Conservation de l'énergie	Champs de force	Pas de temps	Performances	Autre
NAMD	Très bon	Implémentation très rigoureuse	Nombreuses possibilités	Bonne extensibilité, équilibrage de charge automatique	
GROMACS	Moins rigoureux	Portage de certains champs avec concessions sur la précision		Extensibilité moindre, équilibrage de charge automatique	Performance des calculs PME à étudier au préalable de l'étude, mécanisme pratique de <i>restart</i>
LAMMPS		Grande variété de champs de force disponible	Nombreuses possibilités		Plus lent que les autres logiciels pour les systèmes biologiques (facteur 2 à 6)

## Conclusion

- ▶ **Tous les logiciels de chimie sont parallèles...**
  - ...mais à des degrés différents
  - Les logiciels gratuits sont les plus parallèles !
    - Ce sont surtout les plus récents
- ▶ **Tous ne permettent pas les mêmes types de calculs**
  - Théories implémentées
  - Propriétés « exclusives »

# Comment calculer à l'IDRIS



39

Les logiciels de chimie à l'IDRIS  
Fabien Leydier – 28/06/13

## Comment calculer à l'IDRIS

- Les espaces disques
- Le système de classe
- Le script de soumission
  - Script mono- et multi-étapes
- Commandes de bilan/comptabilité

► Documentation plus détaillée disponible sur le site de l'IDRIS ([www.idris.fr](http://www.idris.fr))



40

Les logiciels de chimie à l'IDRIS  
Fabien Leydier – 28/06/13



# Les espaces disques

## ► 2 types de machines

- Serveur de fichiers GPFS
  - Espace disque des machines de calcul
  - Découpé en 3 sections pour chaque login du projet
    - \$HOME : petit espace, sauvegardé tous les jours
    - \$WORKDIR : espace de travail, pas de sauvegarde
    - \$TMPDIR : espace temporaire, effacé après chaque calcul
- Machine d'archivage (Gaya)
  - Stockage sur cartouches magnétiques
  - Archivage à long terme

## ► Demande d'espace

- Lors de la constitution du dossier DARI
- <https://extranet.idris.fr/>
  - Demande par machine
  - Volume + inodes pour chaque machine de calcul et pour Gaya



# Les espaces disques

## ► Serveur de fichier GPFS

- \$HOME
  - Modèles de scripts, exécutables, etc.
- \$WORKDIR : fichiers d'entrée et de sortie des calculs
  - On peut y lancer des calculs
    - Visibilité directe pendant l'exécution
      - 1 dossier pour 1 calcul permet de mieux s'y retrouver
  - Soumis à quota
    - Faire le ménage en fin de calcul
  - Performances identiques au \$TMPDIR
- \$TMPDIR : espace temporaire par calcul
  - Utile si les fichiers générés sont très (très) volumineux
  - Supprimé en fin de calcul
    - Pas de ménage à faire en fin de calcul
    - Attention à prévoir la recopie sur \$WORKDIR !
  - Conservé entre les étapes d'un *job* multi-étapes
    - Utile pour être sûr de recopier ses fichiers en fin de calcul
    - Limite de temps, plantage, etc.
  - Pas de visibilité directe si  $N_{\text{cœurs}} \leq 32$  sans *job* multi-étapes
    - Sinon, chemin à récupérer pendant l'exécution (**echo \$TMPDIR**)



# Les espaces disques

## ► Machine d'archivage (Gaya)

- Stockage sur cartouches magnétiques
  - Conserver des résultats à long terme
    - Réutilisés localement pour des calculs ultérieurs
    - Fichiers trop volumineux pour le \$WORKDIR
    - Sauvegarde sélective de résultats du \$WORKDIR
  - Accès « lent » : pas adapté aux transferts répétitifs
    - Pas de copie systématique
- Accès aux fichiers
  - Pas d'accès direct pour un calcul
    - Copier les fichiers avant ou après l'exécution du calcul (ou manuellement), via les commandes spécifiques :
      - **mput** : écrire sur Gaya
      - **mget** : copie de Gaya vers le répertoire courant
  - Durée de vie des fichiers
    - 1 an par défaut
    - Allongement via la commande **mfret** une fois par an
    - Email envoyé avant expiration des fichiers



# Les espaces disques

## ► \$WORKDIR partagé entre Ada et Turing

## ► Commandes pratiques

- Quotas
  - Sur chaque machine :
    - **quota\_u** : quota pour le \$HOME
    - **quota\_u -w** : quota pour le \$WORKDIR  
(rappel : pas de quota sur \$TMPDIR)
- Rappel : espace utilisé par un répertoire
  - **du -sh nom\_du\_répertoire**



# Le système de classes

## ► Répartition des *jobs* par LoadLeveler

- Type de *job* (séquentiel, OpenMP, MPI/hybride)
- Nombre de cœurs ou nœuds de calcul
- Mémoire
- Temps d'exécution

## ► Sur Ada

- Temps maximum par *job* :  $100\text{ h} \leq 32\text{ cœurs}$ ,  $20\text{ h} \leq 2048\text{ cœurs}$
- *Jobs* OpenMP et grosse mémoire sur des nœuds dédiés (38 nœuds)

## ► Sur Turing

- Temps maximum par *job* : 20 h (jusqu'à toute la machine, 4096 nœuds)
- Réservation minimale aujourd'hui : 64 nœuds (= 1024 cœurs)

## ► Commande : **news class**



Les logiciels de chimie à l'IDRIS  
Fabien Leydier – 28/06/13

45

# Le système de classes

## ► Commandes

- **llsubmit script** : lancer son script de soumission
- **llcancel numéro\_du\_job** : annuler l'exécution
- **llq** : pour voir toute la file d'attente
- **llq -u login** : pour voir sa propre file d'attente
  - Pour personnaliser l'affichage : `man llq` (`llq -f %...`)
- **llq -l numéro\_du\_job** : TOUTES les informations connues sur le *job* (temps, dossiers, etc.)



Les logiciels de chimie à l'IDRIS  
Fabien Leydier – 28/06/13

46

# Le système de classes

## ► exemple

Nœud maître d'exécution

```
$ llq
```

Id	Owner	Submitted	ST	PRI	Class	Running On
ada338.222911.2	rgmt002	4/22 15:06	R	100	mt8t4	ada305-id
ada338.228902.0	rspe001	4/26 11:08	R	100	mt8t4	ada316-id
ada338.228957.0	rspe001	4/26 11:48	R	100	mt8t4	ada320-id
ada338.229237.0	rspe001	4/26 14:55	R	100	mt8t4	ada306-id
ada338.229277.0	rspe001	4/26 15:19	R	100	mt8t4	ada308-id
ada338.229375.0	rspe001	4/26 16:14	R	100	mt8t4	ada312-id
ada338.229376.0	rspe001	4/26 16:15	R	100	mt8t4	ada307-id
ada338.228083.0	rwvq004	4/25 18:17	R	100	t4L	ada309-id
ada338.228084.0	rwvq004	4/25 18:17	R	100	t4L	ada311-id
ada338.228085.0	rwvq004	4/25 18:17	R	100	t4L	ada314-id
ada338.228086.0	rwvq004	4/25 18:17	R	100	t4L	ada315-id
ada338.228087.0	rwvq004	4/25 18:17	R	100	t4L	ada310-id
ada338.228088.0	rwvq004	4/25 18:17	R	100	t4L	ada317-id
ada338.228089.0	rwvq004	4/25 18:17	R	100	t4L	ada318-id
ada338.228090.0	rwvq004	4/25 18:17	R	100	t4L	ada305-id
ada338.228091.0	rwvq004	4/25 18:17	R	100	t4L	ada313-id
ada338.228092.0	rwvq004	4/25 18:17	R	100	t4L	ada316-id
ada338.232472.1	regi904	4/29 16:08	P	100	archive	ada338
ada338.231397.0	rjjv002	4/28 19:32	R	100	c8t4L	ada320-id
ada338.231310.0	ranq323	4/28 19:32	R	100	c16t4	ada306-id
ada338.231311.0	rspe001	4/28 19:32	R	100	mt8t4	ada308-id
ada338.231312.0	rspe001	4/28 19:32	R	100	c128t3	ada002-id
ada338.231313.0	rspe001	4/28 19:32	R	100	c256t3	ada013-id
ada338.231314.0	rspe001	4/28 19:32	R	100	c1024t3	ada039-id
ada338.231315.0	rspe001	4/28 19:32	R	100	c512t3	ada144-id
ada338.231316.0	rspe001	4/28 19:32	R	100	mt16t4	ada312-id
ada338.231317.0	rspe001	4/28 19:32	R	100	c32t3	ada191-id

Classe OpenMP : **mt**  
 Classe MPI/hybride : **c**  
 Nombre de cœurs max. de la classe : **1,2,3,4**  
 Temps max. de la classe : **t0,1,2,3,4**  
 Nœuds mémoire large (pour MPI) : **L**



Les logiciels de chimie à l'IDRIS  
 Fabien Leydier - 28/06/13

47

## Le script de soumission

```
##@ job_name =
##@ job_type =
##@ output =
##@ error =
##@ total_tasks =
##@ parallel_threads =
##@ wall_clock_limit =
##@ queue

module load chimie

set -x

poe chimie.exe input.in > out
```

### ► Directives LoadLeveler

- Commencent par **#@**
  - ~~#@, # @, # #@, # # @, @, #~~
- Se terminent par « **#@ queue** »
- Obligatoires / Fortement conseillées :
  - **#@ job\_name**
  - **#@ job\_type**
  - **#@ total\_tasks** et/ou **parallel\_threads**
  - **#@ bg\_size** (pour Turing)
  - **#@ wall\_clock\_limit**
  - **#@ output**
  - **#@ error**
- Optionnelles :
  - **#@ notify\_user** : envoi de mails
  - **#@ notification** : type de mail
  - **#@ class** : pour les classes spéciales
  - **#@ as\_limit** : demande mémoire par processus
  - **ATTENTION !** Ada ≠ Vargas : **data\_limit** et **stack\_limit** ne vous donneront pas votre demande, et risquent de la limiter
  - Documentation : <http://www.idris.fr/ada/ada-doc-ibm-intel.html>

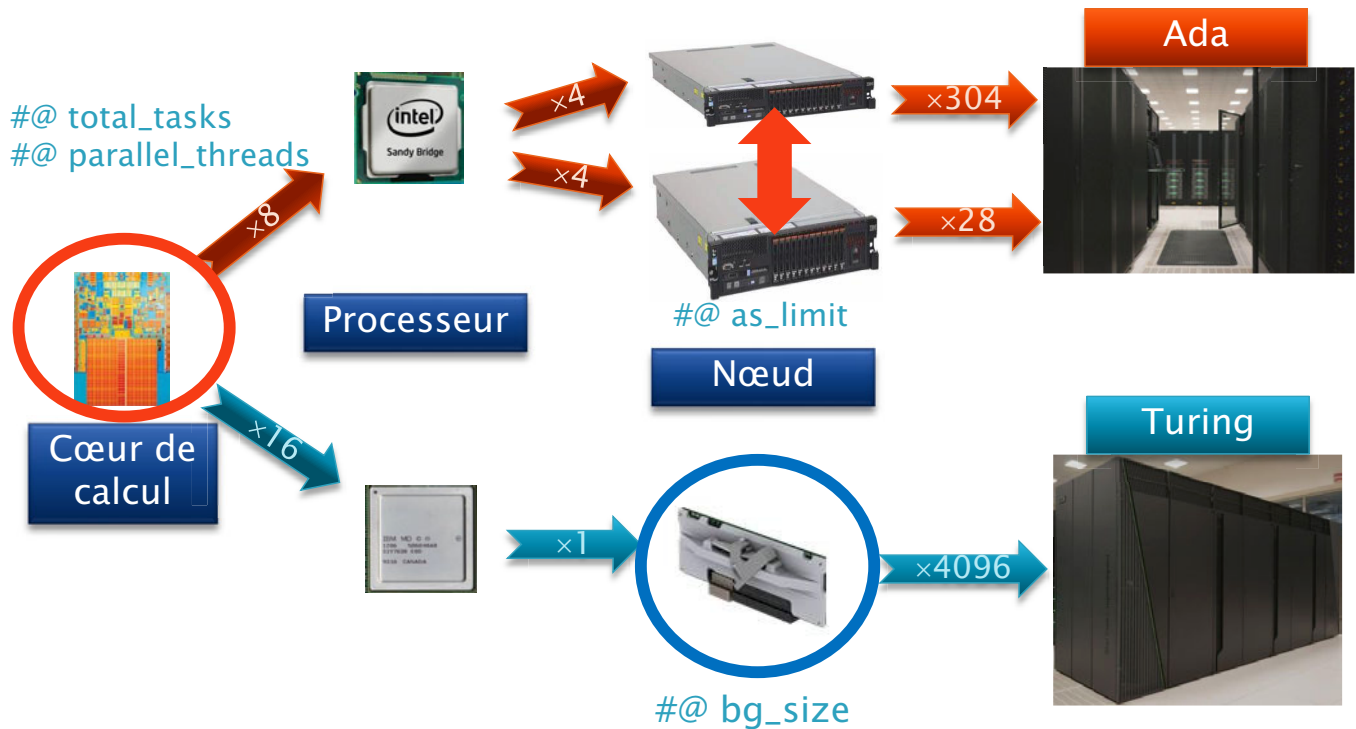


Les logiciels de chimie à l'IDRIS  
 Fabien Leydier - 28/06/13

48



# Ce que je réserve pour mon calcul



## Le script de soumission

### ► Classes spéciales

- Classe archive
  - Transferts de fichiers Gaya (mfput, mfget), « manipulation » de fichiers (cp, cat, >, ...)
  - `#@ class = archive`
  - Non facturée, calculs interdits
- Classe compilation
  - Exécution séquentielle, 20 h max
  - `#@ class = compil`
- Classe pré/post traitement
  - Traitement des données
  - `# @ requirements = (Feature == "prepost")`
  - Non facturée, calculs interdits, durée très limitée

# Le script de soumission : OpenMP

## ► Ada

- `#@ job_type = serial`
- Réserve des cœurs
  - `#@ parallel_threads = Ncœurs`
  - Décompte du temps (« facturation »)
    - $\leq 32 \text{ cœurs} : t \times N_{\text{cœurs}}$
- Réserve de la mémoire
  - Par défaut (pas de directive)
    - $7,0 \text{ Go} \times N_{\text{cœurs}}$  pour le programme
    - Utilisation des nœuds « large mémoire »
  - `#@ as_limit = Mem`
    - Au maximum :  $7,0 \times N_{\text{cœurs}}$  pour le programme (ex : `# @ as_limit = 28.0GB` pour 4 *threads*)
    - Directive non nécessaire : le maximum est déjà attribué par défaut



# Le script de soumission : OpenMP

## ► Turing

- Pas impossible, mais...
  - C'est utilisable a priori
    - D'après le découpage, 1 nœud = 16 cœurs + 16 Go de mémoire
    - Le programme s'exécute en intra-nœud : jusqu'à 64 *threads*
  - Réserve minimale
    - `#@ bg_size ≥ 64 ... nœuds`
    - Le programme ne va s'exécuter que sur 1 nœud : 63 nœuds « perdus » !
  - Performances
    - Processeurs
      - Turing : 1 nœud  $\approx 204,1 \text{ GFLOPS}$
      - Ada : 1 nœud  $\approx 691,2 \text{ GFLOPS}$
    - Mémoire
      - Turing :  $16 \text{ Go/nœud} \rightarrow 256 \text{ Mo} - 1 \text{ Go/cœur}$
      - Ada :  $128 - 256 \text{ Go/nœud} \rightarrow 4 - 8 \text{ Go/cœur}$
- En OpenMP, Turing n'est pas adapté par rapport à Ada



# Le script de soumission : MPI

## ► Ada

- `#@ job_type = parallel`
- Réserve des cœurs
  - `#@ total_tasks = Ncœurs`
  - Décompte du temps (« facturation »)
    - $\leq 32$  cœurs :  $t \times N_{\text{cœurs}}$
    - $> 32$  cœurs :  $t \times N_{\text{nœuds}} \times 32$  (ex : 33 cœurs demandés = 64 cœurs décomptés !)
- Réserve de la mémoire
  - Par défaut (pas de directive)
    - 3,5 Go/cœur = 3,5 Go/processus MPI
  - `#@ as_limit = Mem`
    - Utilisation des nœuds « larges mémoire » : max 32 cœurs
    - 7,0 Go/cœur = 7,0 Go/processus MPI

# Le script de soumission : MPI

## ► Turing

- `#@ job_type = BLUEGENE`
- Réserve des nœuds
  - `#@ bg_size = Nnœuds`
    - $N_{\text{nœuds}} \geq 64$  ( $\Rightarrow$  1024 cœurs = de 64 à 4096 processus MPI !)
    - Réserve par puissance de 2 (64, 128, 256, ..., 4096)
  - Décompte du temps (« facturation »)
    - $t \times N_{\text{nœuds}} \times 16$
- Répartition MPI
  - `--np NMPI` : nombre total de processus MPI (jusqu'à 4 par cœur)
- Réserve de la mémoire
  - Selon l'agencement des processus (`--ranks-per-node=`)
    - Minimum : 64 processus/nœud  $\rightarrow$  256 Mo/processus MPI
    - Maximum : 1 processus/nœud  $\rightarrow$  16 Go/processus MPI

# Le script de soumission : Hybride

## ► Ada

- `#@ job_type = parallel` ← *Élément de base : processus MPI*
- Réserve des cœurs
  - `#@ total_tasks = Nprocessus MPI`
  - `#@ parallel_thread = Nthreads/processus`
  - $N_{\text{cœurs}} = N_{\text{processus MPI}} \times N_{\text{threads/processus}}$
  - Décompte du temps (« facturation »)
    - $\leq 32$  cœurs :  $t \times N_{\text{cœurs}}$
    - $> 32$  cœurs :  $t \times N_{\text{nœuds}} \times 32$  (ex : 33 cœurs demandés = 64 cœurs décomptés !)
- Réserve de la mémoire
  - Par défaut (pas de directive)
    - $3,5 \text{ Go/cœur} = 3,5 \text{ Go} \times N_{\text{threads/processus}} / \text{processus MPI}$
  - `#@ as_limit = Mem`
    - Au maximum :  $7,0 \times N_{\text{cœurs}} / \text{processus MPI}$  (ex : `#@ as_limit = 28,0GB` pour 4 *threads/processus MPI*)
    - Limité à 32 cœurs ( $N_{\text{cœurs}}$  au total)



# Le script de soumission : Hybride

## ► Turing

- `#@ job_type = BLUEGENE`
- Réserve des nœuds
  - `#@ bg_size = Nnœuds`
    - $N_{\text{nœuds}} \geq 64$  ( $\Rightarrow 1024$  cœurs = de 64 à 4096 processus MPI !)
    - Réserve par puissance de 2 (64, 128, 256, ..., 4096)
  - Décompte du temps (« facturation »)
    - $t \times N_{\text{nœuds}} \times 16$
- Répartition MPI
  - `--np NMPI` : nombre total de processus MPI
- Réserve de la mémoire
  - Selon l'agencement des processus (`--ranks-per-node=` )
    - Minimum : 64 processus/nœud  $\rightarrow 256 \text{ Mo/processus MPI}$
    - Maximum : 1 processus/nœud  $\rightarrow 16 \text{ Go/processus MPI}$
- Répartition OpenMP
  - `--envs "OMP_NUM_THREADS=NOMP"` : nombre de *threads* par processus MPI
- Bilan de répartition
  - $\text{Bg\_size} \times \text{ranks-per-node} = \text{np}$
  - $\text{Ranks-per-node} \times \text{OMP\_NUM\_THREADS} \leq 64$
  - Pas de mémoire supplémentaire apportée par des *threads* OpenMP, contrairement à Ada



# Le script de soumission

## ► Lignes de commandes

```
##@ job_name =  
##@ job_type =  
##@ output =  
##@ error =  
##@ total_tasks =  
##@ parallel_threads =  
##@ wall_clock_limit =  
##@ queue
```

```
module load chimie  
  
set -x  
  
poe chimie.exe input.in > out
```

### ◦ Programme Modules

- Charge l'environnement du logiciel (*paths*, correctifs, variables d'environnement)
- Commandes :
  - `module load logiciel[/version]`
  - `module avail [logiciel]`
  - `module display logiciel[/version]`
  - `module unload logiciel[/version]`
  - `module switch logiciel logiciel/version`

### ◦ `set -x`

- Donne le retour des commandes passées
  - Très utile en cas de problème
  - Peut être verbeux (module, etc.)
    - A placer après `module` par exemple



# Le script de soumission

## ► Lignes de commandes

```
##@ job_name =  
##@ job_type =  
##@ output =  
##@ error =  
##@ total_tasks =  
##@ parallel_threads =  
##@ wall_clock_limit =  
##@ queue
```

```
module load chimie  
  
set -x  
  
poe chimie.exe input.in > out
```

### ◦ Lancement de l'exécutable

- OpenMP : lancement direct
  - Ex : `g09 input`
- MPI et hybride : lancement avec `poe`
  - Ex : `poe MPPCrystal`
- Pour certains logiciels : « `< input` »
  - Ex : `poe pw.x < input`
- Sur Turing : lancement avec `runjob`

### ◦ Redirection de la sortie standard (écran)

- Par défaut : tout est copié dans le fichier indiqué par `##@ output`
- Indiquer « `> fichier` » en fin de ligne d'exécutable pour découpler sortie du programme et sortie du `job`
  - Conseil : `##@ output` et `##@ error` vers un même fichier, et « `> fichier` » pour une sortie du calcul indépendante





# Le script de soumission

## ► Exemple Gaussian sur Ada avec \$TMPDIR

```
#@ job_name = moncalculGaussian
#@ job_type = serial
#@ output = $(job_name).$(jobid)
#@ error = $(job_name).$(jobid)
#@ parallel_threads = 32
#@ wall_clock_limit = 100:00:00
#@ queue

module load gaussian/g09_C01

set -x

cd $TMPDIR
cp $LOADL_STEP_INITDIR/input.in .
cp $LOADL_STEP_INITDIR/input.chk .

g09 input.in > output

rm *.chk

mkdir $LOADL_STEP_INITDIR/resultat
cp * $LOADL_STEP_INITDIR/resultat
```

## ► Exemple VASP sur Ada avec \$WORKDIR

```
#@ job_name = moncalculVASP
#@ job_type = parallel
#@ output = $(job_name).$(jobid)
#@ error = $(job_name).$(jobid)
#@ total_tasks = 128
#@ wall_clock_limit = 20:00:00
#@ queue

module load vasp

set -x

poe vasp
```



59

Les logiciels de chimie à l'IDRIS  
Fabien Leydier – 28/06/13

# Le script de soumission

## ► Exemple CPMD sur Turing avec \$WORKDIR

```
#@ job_name = moncalculCPMD
#@ job_type = BLUEGENE
#@ output = $(job_name).$(jobid)
#@ error = $(job_name).$(jobid)
#@ bg_size = 512
#@ wall_clock_limit = 20:00:00
#@ queue

module load cpmd

set -x

runjob --ranks-per-node 4 --envs "OMP_NUM_THREADS=16" --np 2048 : $CPMD_EXEDIR/cpmd.x ./input > out
```



60

Les logiciels de chimie à l'IDRIS  
Fabien Leydier – 28/06/13

# Script multi-étapes

```
#===== Directives globales =====
#@ job_name = multi-steps-calcul
#@ output = $(job_name).$(step_name).$(jobid)
#@ error = $(output)

#===== Directives pour étape 1 =====
#@ step_name = calcul
#@ job_type = serial
#@ parallel_threads = 8
#@ wall_clock_limit = 20:00:00
#@ queue

#===== Directives pour étape 2 =====
#@ step_name = copie
#@ dependency = (calcul >= 0)
#@ class = archive
#@ queue

case ${LOADL_STEP_NAME} in

#== Etape 1 =====
calcul )

set -ex
module load gaussian
cd $TMPDIR
cp ${LOADL_STEP_INITDIR}/* .
g09 input > resultat
;;

#== Etape 2 =====
copie )

set -ex
cd $TMPDIR
mkdir ${LOADL_STEP_INITDIR}/Output
cp * ${LOADL_STEP_INITDIR}/Output
;;

esac
```

- ▶ Permet de lancer l'équivalent de plusieurs scripts à la suite
  - Utilisation du TMPDIR (rémanent)
  - Sauver les résultats obtenus après la limite de temps de l'étape calcul
  - Dépendance entre les étapes
- ▶ S'écrit comme des scripts « éclatés »
- ▶ Lignes en # : commentaires



# Script multi-étapes

```
#===== Directives globales =====
#@ job_name = multi-steps-calcul
#@ output = $(job_name).$(step_name).$(jobid)
#@ error = $(output)

#===== Directives pour étape 1 =====
#@ step_name = calcul
#@ job_type = serial
#@ parallel_threads = 8
#@ wall_clock_limit = 20:00:00
#@ queue

#===== Directives pour étape 2 =====
#@ step_name = copie
#@ dependency = (calcul >= 0)
#@ class = archive
#@ queue

case ${LOADL_STEP_NAME} in

#== Etape 1 =====
calcul )

set -ex
module load gaussian
cd $TMPDIR
cp ${LOADL_STEP_INITDIR}/* .
g09 input > resultat
;;

#== Etape 2 =====
copie )

set -ex
cd $TMPDIR
mkdir ${LOADL_STEP_INITDIR}/Output
cp * ${LOADL_STEP_INITDIR}/Output
;;

esac
```

- ▶ Découpé en deux parties
  - Directives LoadLeveler
  - Commandes
  - Encadrées par des commandes obligatoires : case et esac



# Script multi-étapes

```
#===== Directives globales =====
#@ job_name = multi-steps-calcul
#@ output = $(job_name).$(step_name).$(jobid)
#@ error = $(output)

#===== Directives pour étape 1 =====
#@ step_name = calcul
#@ job_type = serial
#@ parallel_threads = 8
#@ wall_clock_limit = 20:00:00
#@ queue

#===== Directives pour étape 2 =====
#@ step_name = copie
#@ dependency = (calcul >= 0)
#@ class = archive
#@ queue

case ${LOADL_STEP_NAME} in

#== Etape 1 =====
calcul)

set -ex
module load gaussian
cd $TMPDIR
cp ${LOADL_STEP_INITDIR}/* .
g09 input > resultat
;;

#== Etape 2 =====
copie)

set -ex
cd $TMPDIR
mkdir ${LOADL_STEP_INITDIR}/Output
cp * ${LOADL_STEP_INITDIR}/Output
;;

esac
```

## ► 1<sup>er</sup> paragraphe

- Directives communes à toutes les étapes
- Chaque étape possède une sortie distincte



Les logiciels de chimie à l'IDRIS  
Fabien Leydier – 28/06/13

63

# Script multi-étapes

```
#===== Directives globales =====
#@ job_name = multi-steps-calcul
#@ output = $(job_name).$(step_name).$(jobid)
#@ error = $(output)

#===== Directives pour étape 1 =====
#@ step_name = calcul
#@ job_type = serial
#@ parallel_threads = 8
#@ wall_clock_limit = 20:00:00
#@ queue

#===== Directives pour étape 2 =====
#@ step_name = copie
#@ dependency = (calcul >= 0)
#@ class = archive
#@ queue

case ${LOADL_STEP_NAME} in

#== Etape 1 =====
calcul)

set -ex
module load gaussian
cd $TMPDIR
cp ${LOADL_STEP_INITDIR}/* .
g09 input > resultat
;;

#== Etape 2 =====
copie)

set -ex
cd $TMPDIR
mkdir ${LOADL_STEP_INITDIR}/Output
cp * ${LOADL_STEP_INITDIR}/Output
;;

esac
```

## ► 1<sup>re</sup> étape :

### Lancement du calcul

- **#@ step\_name**
  - Doit correspondre dans les directives et les commandes
  - Eviter les caractères spéciaux (-#@ : non acceptés)
- **set -ex**
  - Donne l'écho des commandes
  - En cas d'erreur sur l'une des commandes, l'ensemble de l'étape est considérée comme erronée
    - A placer en tête de l'étape
- Les directives se terminent par « **#@ queue** »
- Les commandes se terminent par « **;;** »



Les logiciels de chimie à l'IDRIS  
Fabien Leydier – 28/06/13

64

# Script multi-étapes

```
#===== Directives globales =====
#@ job_name = multi-steps-calcul
#@ output = $(job_name).$(step_name).$(jobid)
#@ error = $(output)
```

```
#===== Directives pour étape 1 =====
#@ step_name = calcul
#@ job_type = serial
#@ parallel_threads = 8
#@ wall_clock_limit = 20:00:00
#@ queue
```

```
#===== Directives pour étape 2 =====
#@ step_name = copie
#@ dependency = (calcul >= 0)
#@ class = archive
#@ queue
```

```
case ${LOADL_STEP_NAME} in
```

```
== Etape 1 =====
calcul )
```

```
set -ex
module load gaussian
cd $TMPDIR
cp ${LOADL_STEP_INITDIR}/* .
g09 input > resultat
;;
```

```
== Etape 2 =====
copie )
```

```
set -ex
cd $TMPDIR
mkdir ${LOADL_STEP_INITDIR}/Output
cp * ${LOADL_STEP_INITDIR}/Output
;;
```

```
esac
```

## 2<sup>e</sup> étape :

### Copie des résultats

#### • #@ dependency

- Gère la condition de lancement de l'étape 2
- Utilise le code de retour dans la condition
  - == 0 : si l'étape 1 s'est bien déroulée
  - > 0 : si l'étape 1 a eu un problème
  - >= 0 : peu importe l'état de l'étape 1
- L'ordre d'exécution est à gérer soi-même



Les logiciels de chimie à l'IDRIS  
Fabien Leydier - 28/06/13

65

# Script multi-étapes : exemple TMPDIR

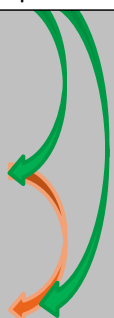
```
# @ job_name = multi-steps-vasp
#@ output = $(job_name).$(step_name).$(jobid)
#@ error = $(output)
```

```
# @ step_name = copy
#@ class = archive
#@ queue
```

```
# @ step_name = vasp
#@ dependency = (copy == 0)
#@ job_type = parallel
#@ total_tasks = 128
#@ wall_clock_limit = 10:00:00
#@ queue
```

```
# @ step_name = fin
#@ dependency = (copy == 0) && (vasp >= 0)
#@ class = archive
#@ queue
```

dépendances



```
case ${LOADL_STEP_NAME} in
```

```
copy )
set -ex
cd $TMPDIR
cp ${LOADL_STEP_INITDIR}/INCAR .
cp ${LOADL_STEP_INITDIR}/POSCAR .
cp ${LOADL_STEP_INITDIR}/POTCAR .
cp ${LOADL_STEP_INITDIR}/KPOINTS .
cp ${LOADL_STEP_INITDIR}/WAVECAR .
;;
```

```
vasp )
set -ex
module load vasp
cd $TMPDIR
echo "calcul lancé" > etat
poe vasp
;;
```

```
fin )
set -ex
cd $TMPDIR
rm WAVECAR
rm CHG*
cp * ${LOADL_STEP_INITDIR}
cd ${LOADL_STEP_INITDIR}
echo "job terminé" > etat
;;
```

```
esac
```



Les logiciels de chimie à l'IDRIS  
Fabien Leydier - 28/06/13

66

# Script multi-étapes : exemple WORKDIR

```
# @ job_name = multi-steps-vasp
# @ output = $(job_name).$(step_name).$(jobid)
# @ error = $(output)

# @ step_name = copy
# @ class = archive
# @ queue

# @ step_name = vasp
# @ dependency = (copy == 0)
# @ job_type = parallel
# @ total_tasks = 128
# @ wall_clock_limit = 10:00:00
# @ queue

# @ step_name = fin
# @ dependency = (copy == 0) && (vasp >= 0)
# @ class = archive
# @ queue
```

dépendances



```
case ${LOADL_STEP_NAME} in

copy )
set -ex
mfget WAVECAR.calcul_01-06-13 WAVCAR
;;

vasp )
set -ex
module load vasp
poe vasp
;;

fin )
set -ex
mfput WAVCAR WAVCAR.calcul-28-06-13
;;

esac
```



67

Les logiciels de chimie à l'IDRIS  
Fabien Leydier - 28/06/13

## Script multi-étapes : remarques

- ▶ Ne pas lancer plus d'une étape de calcul par *job*
  - Monopolisation des ressources en court-circuitant la file d'attente
  - Utiliser plus de cœurs pour diminuer le temps de restitution
  - Utiliser des *jobs* en cascade :

Script1

```
# @ job_name = multi-steps-vasp1
# @ output = $(job_name).$(step_name).$(jobid)
# @ error = $(output)

# @ step_name = vasp
# @ job_type = parallel
# @ total_tasks = 128
# @ wall_clock_limit = 20:00:00
# @ queue

# @ step_name = copy
# @ dependency = (vasp >= 0)
# @ job_type = archive
# @ queue

# @ step_name = submit
# @ dependency = (vasp > 0)&&(copy == 0)
# @ job_type = serial
# @ queue

case ${LOADL_STEP_NAME} in

vasp )
set -ex
module load vasp
cd $TMPDIR
cp ${LOADL_STEP_INITDIR}/INCAR .
cp ${LOADL_STEP_INITDIR}/POSCAR .
cp ${LOADL_STEP_INITDIR}/POTCAR .
cp ${LOADL_STEP_INITDIR}/KPOINTS .
poe vasp
;;

copy )
set -ex
cd $TMPDIR
cp * ${LOADL_STEP_INITDIR}
;;

submit )
set -ex
lsubmit Script2
esac
```

Script2

```
# @ job_name = multi-steps-vasp2
# @ output = $(job_name).$(step_name).$(jobid)
# @ error = $(output)

# @ step_name = vasp
# @ job_type = parallel
# @ total_tasks = 128
# @ wall_clock_limit = 20:00:00
# @ queue

# @ step_name = copy
# @ dependency = (vasp >= 0)
# @ job_type = archive
# @ queue

case ${LOADL_STEP_NAME} in

vasp )
set -ex
module load vasp
cd $TMPDIR
cp ${LOADL_STEP_INITDIR}/INCAR .
cp ${LOADL_STEP_INITDIR}/POSCAR POSCAR.0
cp ${LOADL_STEP_INITDIR}/POTCAR .
cp ${LOADL_STEP_INITDIR}/KPOINTS .
cp ${LOADL_STEP_INITDIR}/CONTCAR POSCAR
cp ${LOADL_STEP_INITDIR}/WAVECAR .
cp ${LOADL_STEP_INITDIR}/CHG* .
poe vasp
;;

copy )
set -ex
cd ${LOADL_STEP_INITDIR}
mv OUTCAR OUTCAR.1
mv OSZICAR OSZICAR.1
mv XDATCAR XDATCAR.1
cd $TMPDIR
cp * ${LOADL_STEP_INITDIR}
;;

esac
```



68

Les logiciels de chimie à l'IDRIS  
Fabien Leydier - 28/06/13



# Commandes de bilan/comptabilité

## ► Commande jar

- Edite un bilan de ses *jobs* par période
  - Par défaut, édite le bilan sur le mois en cours
    - `jar -d numéro_de_mois` : choix du mois
    - `jar -e date-date` : choix d'une période
  - Données du bilan :
    - noms et numéros des *jobs*
    - nombre de cœurs
    - temps elapsed et CPU (s)
    - efficacité :  $\text{rapport}(\text{temps CPU}) / (\text{temps elapsed} \times N_{\text{cœurs}})$ 
      - Donne une information sur l'efficacité parallèle de chaque *job*

## ► Récapitulatif des heures consommées

- Commande cpt
  - par login, pour tout le groupe
  - % de l'attribution restant
- Extranet de l'IDRIS
  - <https://extranet.idris.fr/>



Les logiciels de chimie à l'IDRIS  
Fabien Leydier – 28/06/13

69

# Utilisation des logiciels : Aspects pratiques, parallélisme et performances



Les logiciels de chimie à l'IDRIS  
Fabien Leydier – 28/06/13

70

# Gaussian

## ► Fichier d'entrée Gaussian

### ◦ %NProcShared=N<sub>cœurs</sub>

- Si %NProcShared > N<sub>cœurs</sub> : message : « **Inconsistency in NProc: NTCur= 32 but real number of threads= 4** », ou lancement d'un nombre supérieur de *threads* (performances **fortement** dégradées, plantage)
- Si %NProcShared < N<sub>cœurs</sub> : on se limite nous-mêmes, et facture =  $t \times N_{\text{cœurs}}$
- %NProc est obsolète

**%NProcShared=32**

**Will use up to 32 processors via shared memory.**

- Exemples de sorties si %NProcShared n'est pas adapté au calcul :

**PrsmSu: requested number of processors reduced to: 10 ShMem 1 Linda.**

**CalDSu: requested number of processors reduced to: 28 ShMem 1 Linda.**

**GetJJB would need an additional 45990731 words of memory to use all 32 processors.**

**DoSDTr: NPSUse= 13**

**JobTyp=1 Pass 1: l= 1 to 5 NPSUse= 1 ParTrn=F ParDer=F DoDerP=T.**

⇒ **difficile d'évaluer le nombre de cœurs maximum sans tester avec son système**

### ◦ %Mem=## GB

- Attribution dynamique de la mémoire pour le calcul
- Par défaut : 256 MB !
- La valeur de #@ as\_limit est trop élevée, mettre environ  $(as\_limit - 1.0 \text{ GB} * (N_{\text{cœurs}} + 1))$

### ◦ #P *fonctionnelle type\_de\_calcul etc.*

- Donne des informations supplémentaires en sortie (entrée dans les Links, convergence des cycles SCF, etc.)



Les logiciels de chimie à l'IDRIS  
Fabien Leydier – 28/06/13

71

# VASP

## ► Aucune commande spéciale pour lancer en parallèle

## ► Obtenir de meilleures performances

### ◦ Fichier INCAR

- Mots-clés à préciser  
LPLANE = .TRUE.  
NPAR = valeur optimale  
LSCALU = .FALSE.  
NSIM = 4  
[KPAR = valeur souhaitée] → version 5.3.3
- Valeurs testées, à consulter sur notre site web

**Exemple : H-MOR, 298 atomes, PBE, 400eV,  $\Gamma$ , 2 pas, version 5.3.3, 64 cœurs :**

- NPAR=1 :  $t_{\text{elapsed}} = 812 \text{ s}$
- NPAR=8 :  $t_{\text{elapsed}} = 477 \text{ s}$   $\div 1,9!$
- NPAR=64 :  $t_{\text{elapsed}} = 425 \text{ s}$   $\div 1,6$
- Version gamma :  $t_{\text{elapsed}} = 269 \text{ s}$  (NPAR=4)

### ◦ Messages d'erreur typiques

- “ **internal ERROR RSPHER:running out of buffer** ”
  - NPAR trop faible
  - Le calcul reste en machine ! → contrôler son bon déroulement au tout début
- Messages de **WARNING** en début de fichier OUTCAR
  - NPAR n'obéit pas à une règle précise (testé)
  - Pour la parallélisation sur les points k, vérifier les résultats car toujours en développement

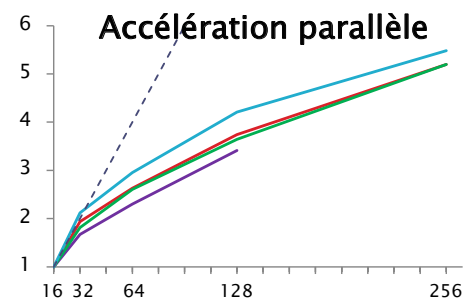
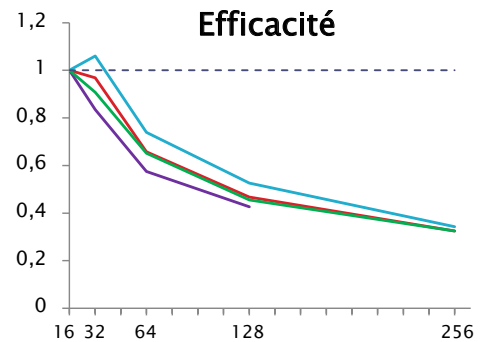
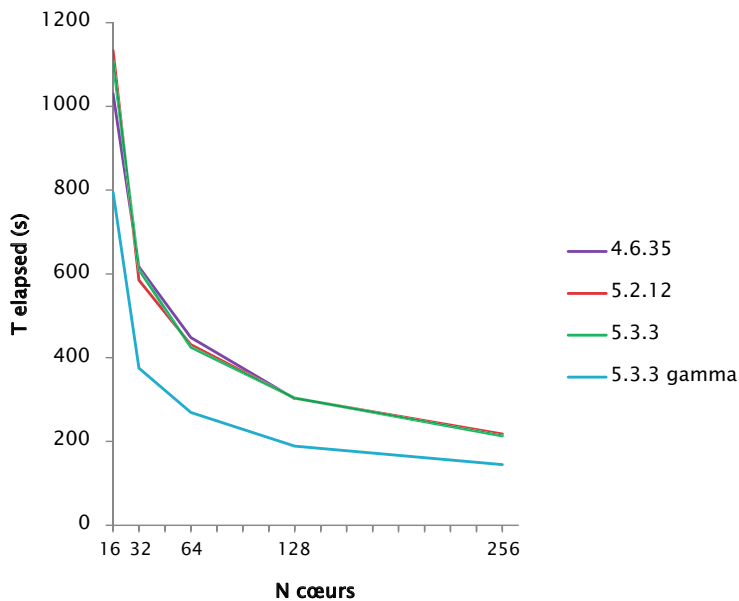


Les logiciels de chimie à l'IDRIS  
Fabien Leydier – 28/06/13

72

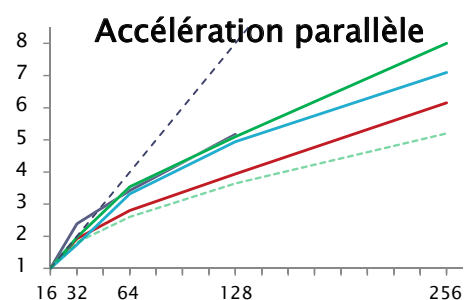
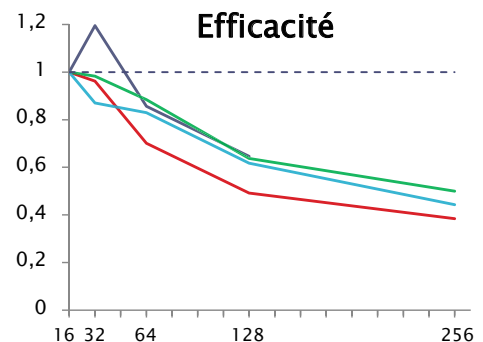
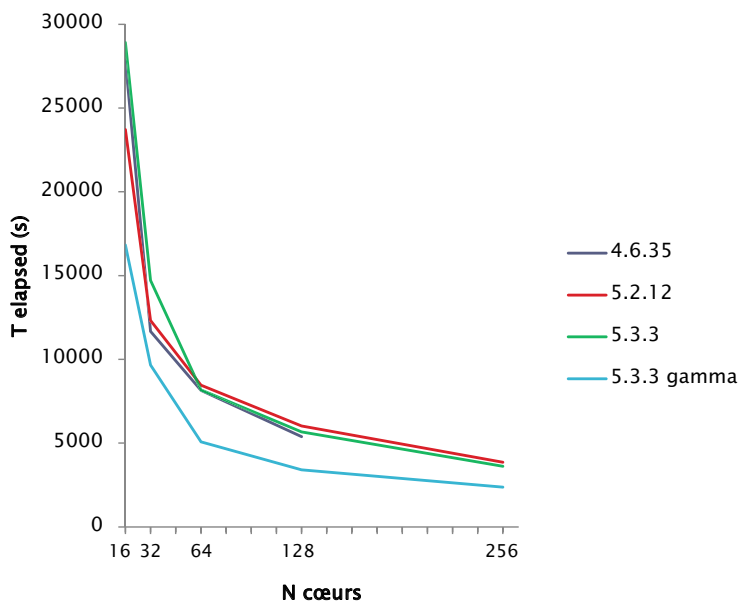
## ▶ Performances

- Exemple : H-MOR, 298 atomes, PBE, 400 eV,  $\Gamma$ , avec/sans dispersion, 2 pas



## ▶ Performances

- Exemple : H-MOR, 298 atomes, PBE, 400 eV,  $\Gamma$ , avec/sans dispersion, **100 pas**



# Crystal

## ► Versions du programme (MPI)

### ◦ PCrystal

- Version dite « à mémoire répliquée » : toutes les matrices sont copiées pour chaque processus
- Génère toutes les données sur le disque
  - Critique pour les performances, tous les fichiers sont lus/écrits en même temps
  - Si  $N_{MPI}$  élevé, le *file system* peut même être saturé (surtout sur Turing)

⇒ Pour les systèmes de petite/moyenne taille uniquement, surtout si beaucoup de symétrie

### ◦ MPPCrystal

- Version dite « à mémoire distribuée » : les matrices sont réparties sur les processus
- Beaucoup moins d'accès disque que PCrystal
  - Mais plus de communications MPI
- Certaines propriétés ne sont pas accessibles (IR notamment)

⇒ Pour les systèmes de taille importante uniquement



# Crystal

## ► Fichiers d'entrée Pcrystal

- Noms des fichiers imposés
  - « INPUT » pour le fichier principal
  - fort.## pour les fichiers écrits/lus (fonction d'onde, etc.)
    - Consulter le manuel pour la description de tous les numéros de fichiers
- Pas de commandes spéciales pour la parallélisation
- Messages d'information sur les tampons mémoire :

```
WARNING **** GENBUF **** COULOMB BIPO BUFFER TOO SMALL - TO AVOID I/O SET BIPOSIZE = 17315175
WARNING **** EXCBUF **** EXCH. BIPO BUFFER TOO SMALL - TO AVOID I/O SET EXCHSIZE = 7652340
```

- Tampons trop petits ⇒ génère encore plus d'accès disque !
  - Indiquer les valeurs dans le fichier principal ( BIPOSIZE ↴ 17400000 ↴ EXCHSIZE ↴ 7652340 ↴ )

## ► Fichiers d'entrée MPPCrystal

- Noms de fichiers identiques à PCrystal
- Fichier principal
  - Ajouter la commande « MPP »
    - Sinon lance une version approchée de Pcrystal
- Messages d'information sur la parallélisation (points k) :

```
WARNING **** MPP **** ALL K POINTS DONE BY ALL PROCESSORS
INFORMATION **** MPP **** This is because either:
INFORMATION **** MPP **** a) There are many k point relative to the number of processors or
INFORMATION **** MPP **** b) Each k point would have to run on a prime number of processors
```

- Pas assez de cœurs ou problème de répartition des points k

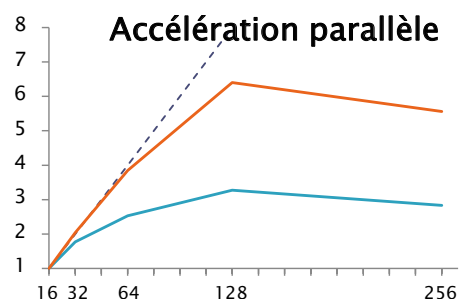
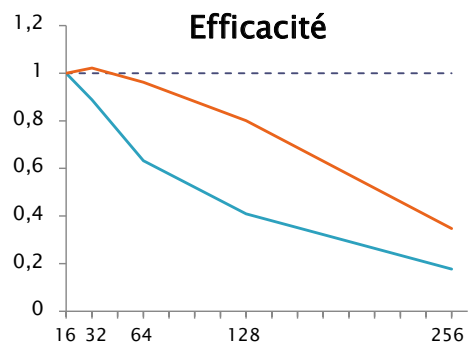
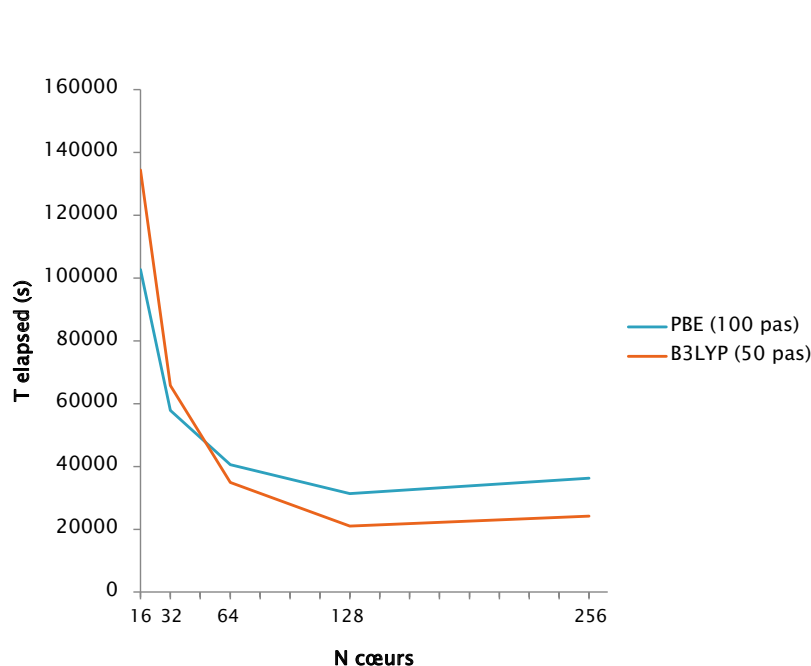
## ◦ Information sur les tampons mémoire identiques à Pcrystal



# Crystal

## ► Performances

- Exemple : H-MOR, 298 atomes,  $\Gamma$ , MPPCrystal



⇒ La charge par cœur augmente

# Quantum Espresso

## ► Aucune commande spéciale pour lancer en parallèle

## ► Aspect performances

- Options de l'exécutable pw.x, gérant l'affectation des cœurs
  - `-nimage n` : décompose en  $n$  images le système (état de transition, etc.).
  - `-ndiag n` : diagonalisation des matrices Hamiltoniennes (la valeur au carré ne doit pas dépasser le nombre de cœurs).
  - `-npool n` : partage des points  $k$  sur les cœurs (pour le point gamma, laisser par défaut ou mettre 1).
  - `-ntg n` : groupe pour les calculs de FFT.
  - Par défaut : toutes ces valeurs valent 1.
- Version 4.3+ : valeurs censées s'adapter (mais à tester), mais informations sur la parallélisation moins détaillées



# Quantum Espresso

## ▶ Aspect performances

- Exemples de sorties, **version 4.2** : MOR (144 atomes), PBE, 30 Ry, **pur MPI**

16 cœurs :

Proc/ Pool	planes (dense)	cols (grid)	G	planes (smooth)	cols (grid)	G	columns (wavefct)	G
1	4	798	26462	4	798	26462	200	3312
2	4	798	26462	4	798	26462	200	3308
3	3	798	26462	3	798	26462	200	3308
4	3	798	26462	3	798	26462	200	3308
5	3	798	26462	3	798	26462	200	3308
6	3	801	26462	3	801	26462	199	3307
7	3	798	26462	3	798	26462	200	3308
8	3	798	26462	3	798	26462	200	3308
9	3	798	26462	3	798	26462	200	3308
10	3	798	26462	3	798	26462	200	3308
11	3	798	26462	3	798	26462	200	3308
12	3	798	26462	3	798	26462	200	3308
13	3	798	26462	3	798	26462	198	3306
14	3	802	26462	3	802	26462	198	3306
15	3	802	26462	3	802	26462	198	3306
16	3	802	26462	3	802	26462	198	3306
tot	50	12783	423391	50	12783	423391	3191	52923

## Déséquilibre de charge

**64 cœurs :**

48	1	200	6612	1	200	6612	50	826
49	1	200	6612	1	200	6612	50	826
50	1	200	6612	1	200	6612	50	826
51	0	200	6612	0	200	6612	50	826
52	0	200	6612	0	200	6612	50	826
53	0	200	6612	0	200	6612	50	826
54	0	200	6612	0	200	6612	50	826
55	0	200	6612	0	200	6612	50	826
56	0	200	6612	0	200	6612	50	826
57	0	200	6612	0	200	6612	50	826
58	0	200	6612	0	200	6612	50	826
59	0	200	6612	0	200	6612	50	826
60	0	200	6612	0	200	6612	50	822
61	0	200	6612	0	200	6612	50	822
62	0	198	6610	0	198	6610	50	822
63	0	198	6610	0	198	6610	50	822
64	0	198	6610	0	198	6610	50	822
tot	50	12783	423391	50	12783	423391	3191	52923

Des cœurs ne participent pas directement au calcul!

Des cœurs ne  
participent pas  
directement au calcul!

⇒ **Solution : utiliser des *threads* OpenMP**

- Exemple de sortie, Version 4.3+ : informations sur le parallélisme moins détaillées

### Parallelization info

sticks:	dense	smooth	PW	G-vecs:	dense	smooth	PW
Min	198	198	48		6610	6610	822
Max	200	200	52		6620	6620	830
Sum	12783	12783	3191		423391	423391	52923
Tot	6392	6392	1596				

## Restitution en 114 pas

- 64 cœurs : 114 min  
(modification d'algorithme ?)

## CPMD

- ▶ **Aucune commande spéciale pour lancer en parallèle**

- Répartition des cœurs automatique pour le calcul de la fonction d'onde
  - Extensibilité fortement dépendante du système étudié

## ▶ Aspect performances

- Fichier d'entrée

- MEMORY BIG

- Sur Ada : permet de stocker des facteurs de structure du *cutoff* de la densité en mémoire (évite de les recalculer à chaque besoin)
- Sur Turing : attention à la mémoire disponible

- TASKGROUPS↓ n

- Globalement, partage des cœurs pour traiter les bandes en parallèle
- Améliore l'extensibilité lors de l'utilisation de beaucoup de tâches MPI

- Ondes planes

- Même problématique que Quantum Espresso

- Utiliser des *threads* OpenMP pour améliorer l'extensibilité

NCPU	NGW	NHG	PLANES	GXRAYS	HXRAYS	ORBITALS	Z-PLANES
0	893	6621	2	42	166	12	1
1	893	6621	2	42	166	13	1
2	897	6631	2	42	166	12	1
3	897	6627	2	42	166	12	1
4	898	6633	2	44	166	13	1
5	898	6627	2	44	166	12	1
6	900	6629	2	44	166	13	1
7	898	6625	2	44	166	12	1
8	896	6625	2	44	166	12	1
9	896	6625	2	44	166	13	1

## ► Aucune commande spéciale pour lancer en parallèle

- Par défaut : parallélisation sur les points k
  - Pour obtenir les performances les meilleures,  $N_{\text{cœurs}} \propto \text{nb points k}$

**mpi\_setup1 : WARNING –**

Your number of k-points ( 60) will not distribute correctly  
with the current number of processors ( 32).

You will leave some empty.

**ACTION:** you can reduce number of processors to 30 without losing speed.

## ► Obtenir de meilleures performances

- Parallélisation « KGB »
  - Points k (K), vecteurs d'onde (G) et bandes (B)
    - Mot-clé : `paral_kgb`
  - Non automatique : il faut déterminer le nombre de cœurs attribués pour chaque type
    - Abinit peut tester des configurations :
      - 1°) Lancer le calcul sur 1 cœur, avec `paral_kgb=-N_cœur`, sans autre option (très rapide, même réalisable en amont sur sa machine)
      - 2°) Récupérer la meilleure configuration dans le fichier de sortie
      - 3°) Lancer le calcul avec la meilleure configuration : `npkpt, npband, npfft`

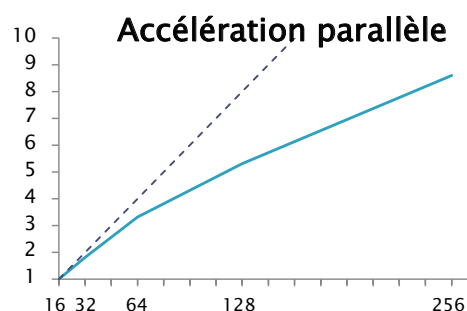
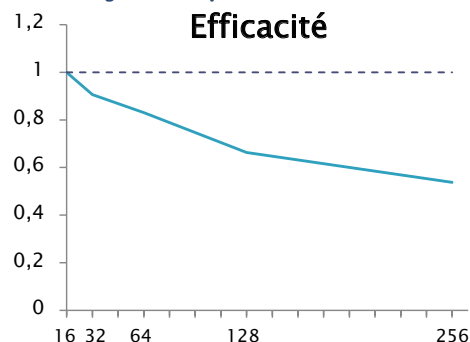
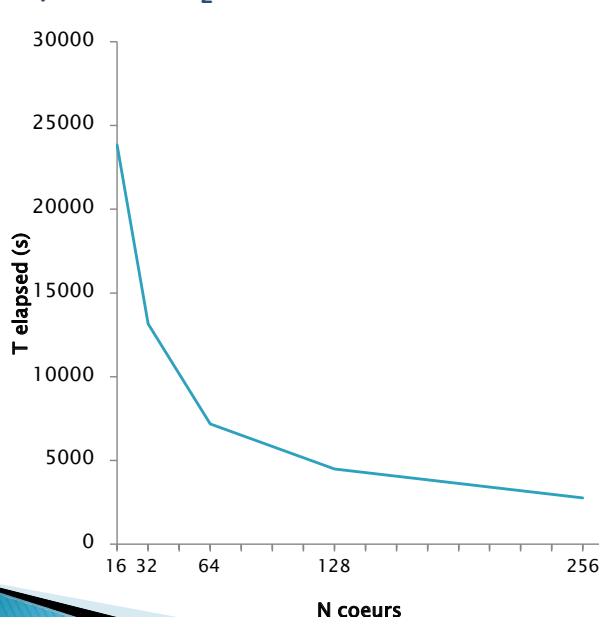
# CP2K

## ► Aucune commande spéciale pour lancer en parallèle

- Aucun contrôle sur la répartition pour la plupart des calculs (automatique)
- Pas d'information donnée sur la répartition effective

## ► Performances

- Exemple : 512 H<sub>2</sub>O, base DZVP, Pade, 280 eV, 50 pas de dynamique moléculaire



# Molcas

## ► Aucune commande spéciale pour lancer en parallèle

- Attention : ne pas lancer avec poe
  - L'exécutable **molcas** est en réalité un script
  - **poe** a déjà été intégré à l'environnement par défaut

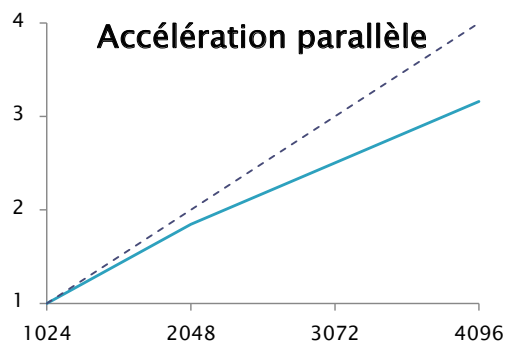
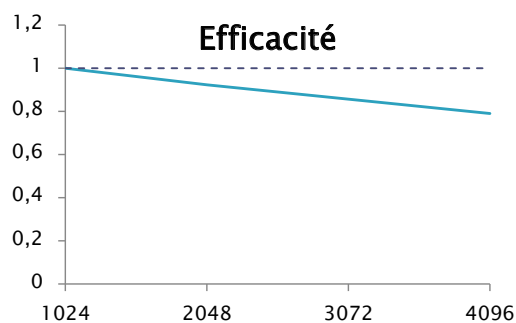
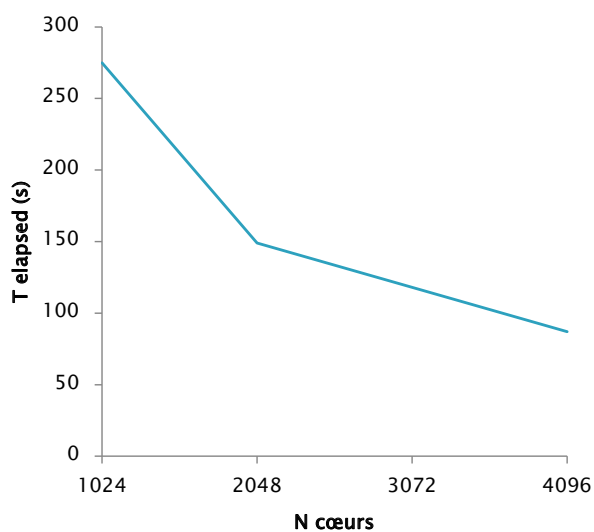
## ► Aspect performances

- Se référer à la commande **jar** pour l'efficacité
- Faire des tests d'extensibilité

# BigDFT

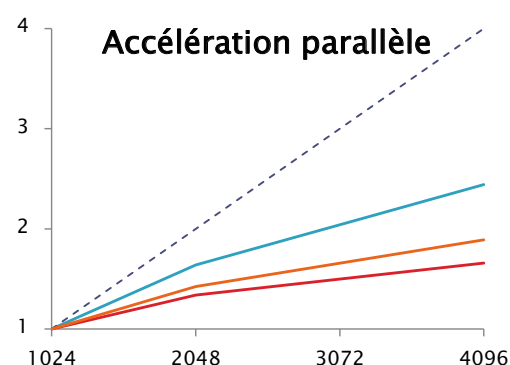
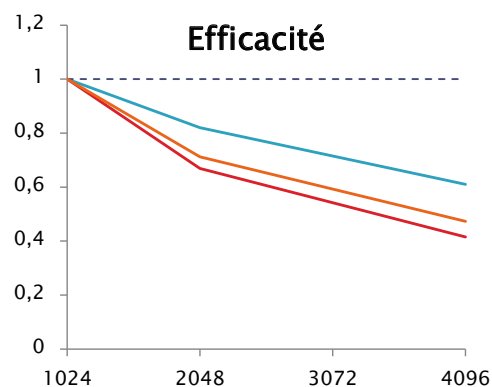
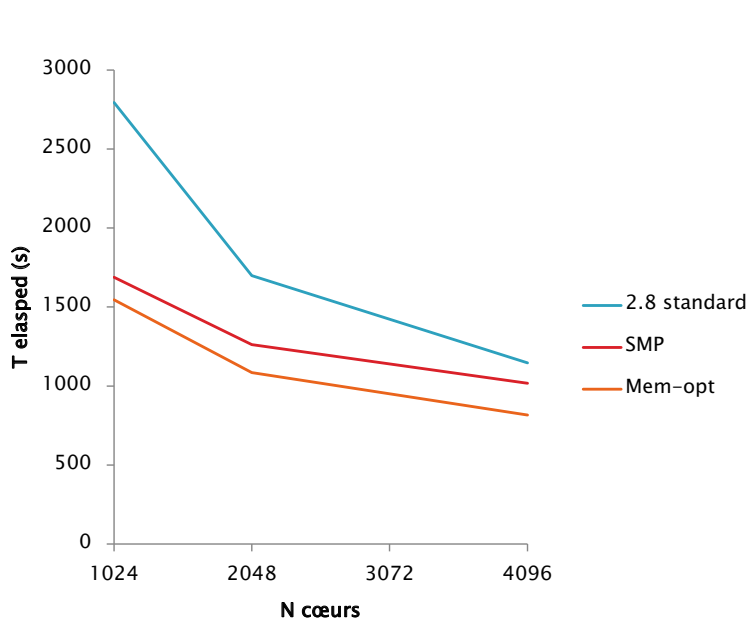
## ► Performances

- Exemple : 512 H<sub>2</sub>O, PBE, formalisme ondelettes, benchmark Turing



## Performances

- Exemple** : Canal bactérien (récepteur nicotinique humain) + propofol lié au canal, 600000 particules, 50000 pas de dynamique, benchmark Turing



## Aspect pratique : le « restart »

Logiciel	Fonction d'onde	Géométrie	Vibrations	Dynamique	Etat de transition	RMN	Autre	Type de restart
Gaussian	■	■	■	■	■	■		■ = Prévu par le logiciel ■ = Non prévu mais faisable ■ = Non prévu
VASP	■	■	■	■	■			
Abinit	■	■	■	■	■			
Quantum Espresso	■	■	■	■	■	■		
CPMD	■	■	■	■	■	■	TDDFT	
ADF	■	■	■	■	■	■		
Molpro	■	■	■	■	■	■	Intégrales	
CP2K	■	■	■	■	■	■		
SIESTA	■	■	■	■	■	■		
Molcas	■	■	■	■	■	■	Intégrales	
Crystal	■	■	■	■	■	■		
NWChem	■	■	■	■	■	■		
BigDFT	■	■	■	■	■	■		

# Aspect pratique : le « *restart* »

## ► 2 types de « *restart* »

- Mots-clés dans le fichier d'entrée
  - Nécessite d'éditer le fichier d'entrée pour continuer le calcul (ou créer un fichier dédié)
  - Syntaxe à respecter
  - Cascade de *jobs* peu pratique
- Fichiers de sortie à donner en entrée
  - Nécessite de connaître le nom des fichiers concernés (voir au besoin les manuels)
  - Utile lorsque le « *restart* » n'est pas implémenté (ex : dynamique, optimisation de géométrie)
  - Cascade de *jobs* facile à mettre en place (si aucun traitement autre que le nom des fichiers)
- Parfois les deux types sont nécessaires

## Bilan de la journée

### ► Calculer de la chimie avec du parallélisme

- C'est possible et même « obligatoire » dans un centre national !
- Demande d'utiliser au mieux les logiciels
  - Utiliser les options adaptées
    - Elles ne sont pas déterminées automatiquement par les logiciels
    - Etudier les manuels, tutoriaux, etc.
- Impératif : adapter les besoins logiciels et matériels à son étude
  - Migrer vers les solutions actuelles, les plus parallèles
    - Requiert un temps d'adaptation, qui sera forcément payant sur du plus long terme
  - Éviter les systèmes trop « petits »
- En cas de problème et/ou de mauvaises performances, ne pas hésiter à demander les conseils de l'Assistance IDRIS !



# Remerciements

- ▶ Noël Jakse
- ▶ Marie–Bernadette Lepetit
- ▶ Alain Pasturel
- ▶ Yves–Henri Sanejouand
  
- ▶ Marc Baaden
- ▶ Benoist Laurent
  
- ▶ Jean–Marie Teuler
  
- ▶ Denis Girou
- ▶ Membres de l’IDRIS