UTILISATION DU CLUSTER MYRIA

https://services.criann.fr/services/calcul/cluster-myria/utilisation/

Patrick Bousquet-Mélou
Patrick.Bousquet-Melou@criann.fr
Version 1.6

Février 2021

CRIANN

Document

Versions

Version	Date	Auteur	Commentaire
1.1	07/03/2019	PBM	
1.2	03/10/2019	PBM	
1.3	06/02/2020	PBM	Directive --mem (au lieu de --mem-per-cpu) suite aux changements de Slurm
1.4	23/07/2020	PBM	Directive —share de Slurm supprimée (n'existe plus)
1.5	05/08/2020	PBM	Débogage avec compilateur Intel : options précisées pour C et FORTRAN
1.6	12/02/2021	PBM	Mises à jour d'URL

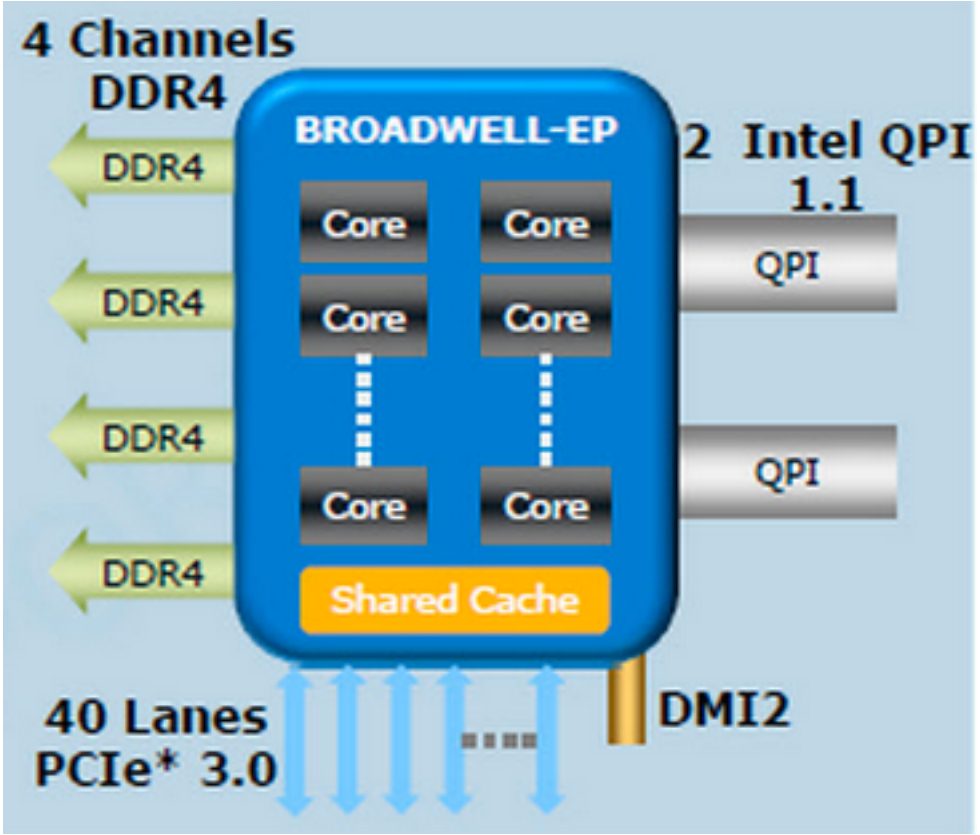
Sommaire

Introduction : notions d'architecture

Processeurs (puces) / cœur

Architectures

- Exemple de la puce Intel «Broadwell EP»
  - 14 **cœurs** (unités de traitement ou «cores»)
  - 3 niveaux de cache de mémoire
    - Cache L1 (intégré au cœur) : 64 Ko (32 Ko octets données + 32 Ko instructions)
    - Cache L2 dédié (1 pour chaque cœur) : 256 Ko
    - Cache L3 partagé (1 pour l'ensemble des cœurs) : 35 Mo
- En général, un cœur exécute une seule tâche (ou processus) d'une application parallèle (multi-tâches)



Puce Intel «Broadwell EP»

Nœud de calcul

Architectures

- Un nœud de calcul est une machine individuelle (serveur qui s'apparente à un PC : 1 carte mère, 1 système d'exploitation)
  - Ses processeurs, et cœurs, accèdent à sa mémoire (dite partagée)
- Exemple : nœud de calcul «Brodwell»
  - 2 puces 14-cœurs Intel «Broadwell EP»
  - 128 Go de mémoire DDR4
- Une application utilisant un parallélisme de type «multithreads» (*parallélisme à mémoire partagée*, avec OpenMP notamment) n'accède aux ressources matérielles (cœurs, mémoire) que d'un seul nœud de calcul
  - Au CRIANN :
    - 28 cœurs et 128 Go de mémoire avec les serveurs standards (bi-processeurs)
    - 256 cœurs et 3,9 To de mémoire avec le serveur SMP

Exemple : Nœud de calcul dx360-M3 (ancienne solution IBM iDataPlex)

Cluster (grappe)

Architectures

- Un cluster (grappe) est constitué d'un ensemble de N nœuds de calcul interconnectés par un réseau
- Exemple : cluster ATOS BULL Myria

  https://services.criann.fr/services/calcul/cluster-myria/utilisation/
- Une application utilisant un *parallélisme à mémoire distribuée* (avec la librairie MPI notamment) peut accéder aux ressources (cœurs, mémoire) de plusieurs nœuds de calcul
  - Exemple : 1 calcul MPI peut utiliser 4200 cœurs avec 4 Go de mémoire par cœur sur le calculateur Myria
  - Les échanges de données entre cœurs répartis dans plusieurs nœuds se font par des communications à travers le réseau d'interconnexion (technologie réseau Intel Omni-Path sur Myria)

Grappe de calcul ATOS BULL Myria

Configuration matérielle et logicielle

Description matérielle

Configuration ATOS BULL Myria

- **11144 cœurs de calcul**
  - 10504 Xeon, 640 KNL
- Puissance théorique de 419 TFlops Xeon, 327 TFlops GPU et 27 TFlops Xeon Phi KNL
- 53 To de mémoire DDR4
- 2,5 Po de disques rapides partagés
  - 25 Go/s de débit agrégé en écriture
  - 20 Go/s de débit agrégé en lecture
- Réseau Intel Omni-Path à faible latence et haut débit (100 GBit/s)
- Connexion sur le réseau SYVIK : 4 X 10 GBit/s et 1 x 40 GBit/s

Grappe de calcul ATOS BULL Myria

Description matérielle (suite)

Serveurs

- **366 nœuds de calcul bi-processeurs Broadwell** (28 cœurs à 2,4 GHz, 128 Go de RAM DDR4 à 2400 MHz), dont :
  - 12 nœuds dotés chacun de 20 To de disques internes
  - 1 nœud doté de 20 To de disques internes et d'1 To de mémoire
  - 12 nœuds dotés chacun de 2 cartes **GPU Kepler K80** (4 unités de traitement GPU par nœud, 12 Go de mémoire embarquée par unité)
  - 8 nœuds dotés chacun de 2 cartes **GPU Pascal P100** (2 unités de traitement GPU par nœud, 12 Go de mémoire embarquée par unité)
  - 1 nœud doté de 3 cartes GPU Pascal P100
- **5 nœuds de calcul bi-processeurs SkyLake** (32 cœurs à 2,1 GHz, 196 Go de RAM DDR4 à 2666 MHz) dotés chacun de 4 cartes **GPU Volta V100-SXM2-32GB** interconnectées par NVLink2
- **10 nœuds de calcul dotés chacun d'un Xeon Phi KNL** 7210 (64 cœurs à 1,3 GHz et 96 Go de RAM DDR4 à 2133 MHz par nœud, 16 Go de mémoire rapide MCDRAM embarquée dans le processeur KNL)
- **1 serveur SMP** avec 256 cœurs Haswell à 2,2 GHz et 4 To de RAM DDR4 à 2133 MHz
- **2 serveurs de visualisation** (bi-processeurs Broadwell) dotés chacun d'une carte GPU K80 et de 256 Go de RAM DDR4 à 2400 MHz

Description matérielle (fin)

Serveurs

- 5 serveurs de connexion (bi-processeurs Broadwell) dotés chacun de 256 Go de RAM DDR4 à 2400 MHz

Connexion et environnement

Myria

- La connexion s'effectue en ssh vers *myria.criann.fr* (ssh login@myria.criann.fr)

- L'utilisateur a un espace de travail personnel dans /home
  - Par défaut un quota disque utilisateur est positionné à 50 Go dans /home
  - La commande *mmlsquota* fournit la place que l'utilisateur occupe dans la partition /home

```
login@Myria-1:~:$ mmlsquota gpfs1:home
                         Block Limits
Filesystem Fileset    type             KB      quota      limit   in_doubt    grace ...
gpfs1      home       USR       507696128  524288000  534298624          0     none ...
```

- Les personnalisations (variables d'environnement, alias) se font avec un fichier *~/.bash_profile* à créer.

Transferts

Entre Myria et l'extérieur

- Pour le transfert de quantités significatives de données entre Myria et l'extérieur, le serveur *myria-transfert.criann.fr* est à privilégier car il est connecté au réseau SYVIK par une interface 40 Gbit/s
  - Cette machine donne accès au même répertoire /home que les frontales. Les identifiants de connexion (ssh, scp, sftp) sur myria-transfert.criann.fr sont les mêmes que sur myria.criann.fr.
  - Les transferts entre Myria et les ressources de l'IDRIS ne sont autorisées que par myria-transfert.criann.fr.

Description logicielle

Environnement de développement

- Documentation
  - https://services.criann.fr/services/calcul/cluster-myria/utilisation/

- Les applications et librairies sont accessibles par un environnement de modules (voir commande «module avail»). Notamment :
  - **Compilateurs** Intel 2016 et 2017, Gnu 4.8.5 : Fortran, C, C++
    - support OpenMP
  - Librairies Intel **MPI** 2016 et 2017, Open MPI 2.0.1
    - support MPI-3 complet
  - Librairies mathématiques
    - Intel **MKL** : BLAS, LAPACK, ScaLAPACK
    - **FFTW** 3.3.5
  - Librairies de formats de données
    - **HDF5** 1.8.18
    - **NetCDF** 4.1.3

La commande «module»

Gestion d'environnement d'applications et de librairies

- https://services.criann.fr/services/calcul/cluster-myria/utilisation/modules/
- Commandes
  - Taper «module help» pour l'ensemble des options de la commande *module*
    - «module **avail»** : lister les modules disponibles
    - «module **avail** *nom_appli*» : lister les modules disponibles pour l'application indiquée
    - «module **help** *nom_appli/version*» : afficher l'aide en ligne (si disponible) pour l'application indiquée
    - «module **load** *nom_appli/version*» : charger l'environnement de l'application indiquée
    - «module **unload** *nom_appli/version*» : décharger l'environnement de l'application indiquée
    - «module **list»** : lister les environnements chargés

- Cas des librairies
  - **Le chargement d'un module de librairie permet de compiler sans appliquer d'options «-I ...» ni «-L...» relative à la librairie en question dans un Makefile, si le compilateur Intel est utilisé**. Le même Makefile peut alors être employé sur plusieurs centres de calcul ayant mis en place un module pour la librairie en question.

  - Pour la structure générale d'un fichier Makefile, voir **/soft/makefiles**.

La commande «module» (fin)

Exemples

- login@Myria-1:~: **module avail castem**

  ---------- /soft/Modules/Myria_modules/cfd_fem_meshing ----------

  castem/2016

- login@Myria-1:~: **module load castem/2016**

  load castem/2016 environment

- login@Myria-1:~ **module help openfoam/1612+**

  ---------- Module Specific Help for 'openfoam/1612+' ----------

  OpenFOAM v1612+ : batch script model is "job_OpenFOAM-v1612+.sl" (/soft/slurm/criann_modeles_scripts)

- login@Myria-1:~ **module avail fftw**

  ---------- /soft/Modules/Myria_modules/libraries ----------

  fftw/3.3.5/impi fftw/3.3.5/serial fftw/3.3.5/serial-threaded

Logiciels mutualisés

Applications scientifiques disponibles

- **Liste à jour** : https://services.criann.fr/services/calcul/cluster-myria/utilisation/
- Chimie et dynamique moléculaire
  - GAUSSIAN, Jaguar 8.0 et 9.4 (Schrödinger)
  - GAMESS 2013
  - NAMD 2.9 (patch Plumed) et 2.12, GROMACS 5.1.4 (patch Plumed), DL_POLY_CLASSIC 1.9
- Mécanique des fluides
  - FDS 5.5.0 (incendies)
  - OpenFOAM : versions officielles : 2.3.0, 3.0.1, 1606+ et 1612+, développement : 3.0.x, 4-x-4.0
  - Telemac v7p1r1
- Mécanique / éléments finis
  - Cast3M 2016, Code_Aster 13.4
  - FreeFem++ 3.51
- Mathématiques
  - R 3.3.2, 3.5.1 et 3.5.2
- Pré-/post-traitements
  - Mailleurs : Gmsh 2.16.0, Neper 3.0.1, Salome 7.8.0
  - Visualisation : Paraview 4.0.1 et 5.2.0, Visit 2.12.1

Librairies disponibles : mathématiques

Environnement de développement

Librairie	Fonctionnalité	URL
FFTW 3.3.5	Librairie parallèle pour le calcul de transformées de Fourier discrètes, sur des données de type réel ou complexe, dans une ou plusieurs dimensions	http://www.fftw.org/
GSL 2.3	Gnu Scientific Library : librairie mathématique pour programmes C et C++	https://www.gnu.org/software/gsl/
MATHEVAL	Evaluation d'expressions symboliques	https://www.gnu.org/software/libmatheval/

Librairie	Fonctionnalité	URL

MKL	Librairie mathématique d'Intel
	
	- BLAS
	  - BLAS 1 : opérations vecteur - vecteur
	  - BLAS 2 : opérations matrice - vecteur
	  - BLAS 3 : opérations matrice - matrice
	  - versions «sparse» de BLAS 1 - 3 : extension aux vevteurs et matrices creux
	
	- LAPACK (Linear Algebra PACKage)
	  - Résolution de systèmes linéaires d'inconnues réelles ou complexes, pour différents types de matrices (générale, bande, triangulaire, triangulaire bande, tridiagonale, symétrique définie positive, hermitienne définie positive, etc.
	
	- ScaLAPACK (Scalable Linear Algebra PACKage)
	  - Version parallèle (MPI) de LAPACK
		https://software.intel.com/en-us/intel-mkl/documentation

MODULEF99	Librairie modulaire d'éléments finis	https://www.rocq.inria.fr/modulef/francais.html
MUMPS	Solveur direct parallèle pour matrices creuses	http://mumps.enseeiht.fr/

Librairie	Fonctionnalité	URL
PETSc	Librairie parallèle pour la gestion de vecteurs et de matrices creuses, la résolution de systèmes linéaires correspondants (liés à des équations aux dérivées partielles) avec des solveurs directs ou itératifs	https://www.mcs.anl.gov/petsc/
SLEPc	Calcul de valeurs et de vecteurs propres de matrices creuses de grande taille (se fonde sur PETSc)	http://slepc.upv.es/
SuiteSparse 4.5.4	Algorithmes de résolution de matrices creuses	http://faculty.cse.tamu.edu/davis/suitesparse.html

Librairies disponibles : maillages

Environnement de développement

Librairie	Fonctionnalité	URL
METIS 5.1.0	Librairie séquentielle pour le partitionnement de graphes et de maillages, et la re -numérotation de matrices creuses	http://glaros.dtc.umn.edu/gkhome/metis/metis/overview
PARMETIS 4.0.3	Librairie parallèle pour le partitionnement de graphes et de maillages, et la re -numérotation de matrices creuses	http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview
SCOTCH 6.0.0	Librairie séquentielle pour le partitionnement de graphes et de maillages, et la re -numérotation par blocs de matrices creuses	http://www.labri.fr/perso/pelegrin/scotch/scotch_fr.html
MMG3D4 4.0.1	Remaillage tétraédrique	https://hal.inria.fr/hal-00681813 https://gforge.inria.fr/frs/?group_id=4307

Librairies disponibles : formats de données

Environnement de développement

Librairie	Fonctionnalité	URL
HDF5 1.8.18	Ecriture et lecture performantes de grands volumes de données scientifiques, dans un format indépendant des architectures matérielles	https://portal.hdfgroup.org/display/HDF5/HDF5
NETCDF 4.1.3 et 4.4	Création, accès ou partage de données scientifiques organisées en tableaux, dans un format indépendant des architectures matérielles	https://www.unidata.ucar.edu/software/netcdf/
MED 3.0.7	Format de données de maillages	http://www.code-aster.org/spip.php?article98

- Fonctions de compression de HDF5 (dans sa version séquentielle)
  - Compression GZIP (supportée sur Myria)
    - Exemples C et FORTRAN sur https://portal.hdfgroup.org/display/HDF5/Examples+by+API
    - Facteur de compression parfois important (exemple : taille de fichier de 5 Mo => 64 Ko)

Environnement de compilation

Compilateurs (CPU)

Environnement de compilation

- Les compilateurs Intel (version 2016 2017 ou 2019) sont conseillés
  - FORTRAN 77 / 90 / 95 : ifort
  - C : icc
  - C++ : icpc
  - Documentation :
    - https://software.intel.com/en-us/intel-fortran-compiler-17.0-user-and-reference-guide
    - https://software.intel.com/en-us/intel-cplusplus-compiler-17.0-user-and-reference-guide
    - manpages : «man ifort», «man icc», «man icpc»
  - Des modèles élémentaires de fichiers Makefile et un README se trouvent dans le répertoire :
    - /soft/makefiles/SERIALCODES

- Les compilateurs Gnu sont aussi disponibles
  - FORTRAN : gfortran (version 4.8.5)
  - C : gcc (version 4.8.5)
  - C++ : g++ (version 4.8.5)
  - manpages : «man gfortran», «man gcc», «man g++»

Commandes de compilation (CPU)

Environnement de compilation

Type de code / librairie MPI utilisée	Activation de l'environnement	Compilateur	Commande de compilation FORTRAN / C / C++
Séquentiel (ou OpenMP)	-	Intel (conseillé)	ifort / icc / icpc
		Gnu	gfortran / gcc / g++
MPI / Intel MPI	-	Intel (conseillé)	mpiifort / mpiicc / mpiicpc
		Gnu	mpif90 / mpicc / mpicxx
MPI / Open MPI	module load openmpi/2.0.1	Intel (conseillé)	mpif90 / mpicc / mpicxx
	module load openmpi-gnu/2.0.1	Gnu	mpif90 / mpicc / mpicxx

Compilation des codes accélérés sur GPU

Environnement de compilation

- Environnement logiciel et soumission des calculs
  - https://services.criann.fr/services/calcul/cluster-myria/utilisation/gpgpu/

  - Outils de programmation (voir lien précédent)

    APIs : CUDA, OpenCL, CUDA FORTRAN

Codes parallèles MPI

Environnement de compilation

- Des modèles élémentaires de fichiers Makefile et un README se trouvent dans le répertoire :

  /soft/makefiles/MPICODES

- Les deux librairies Intel MPI et Open MPI sont disponibles
  - Elles supportent complètement la norme MPI-3

- Lors du portage d'un code MPI sur Myria, il est conseillé de tester Intel MPI (l'environnement de sa version 2017 est pré - chargé), puis Open MPI si nécessaire

- Si à l'issue de ses tests, un utilisateur souhaite activer de manière permanente l'environnement de la librairie Open MPI, il doit inclure la ligne suivante dans son fichier ~/.bash_profile (à créer)

```
login@Myria-1:~: cat .bash_profile
# Librairie Open MPI / compilateur Intel
module load mpi/openmpi/2.0.1
```

Options de compilation Intel : optimisation

Optimisation

- Les options suivantes sont utiles pour l'optimisation de performance des applications en production

Option d'optimisation	Description
-O2 -xCORE-AVX2	Optimisation de base, avec instructions AVX2 (optimisation pour Broadwell)
-O3 -xCORE-AVX2 -fp-model precise	Optimisation avancée, avec instructions AVX2 et renforcement de la précision
-O3 -xCORE-AVX2	Optimisation avancée, avec instructions AVX2
-qopenmp	Activation des directives OpenMP

Options de compilation Intel : optimisation (fin)

Optimisation

- D'autres options peuvent parfois améliorer la performance ; leur validité sur un code donné doit être vérifiée :

Option d'optimisation	Description
-ip	Interprocedural optimization (autorise l'«inlining» de sous-programmes présents dans un même fichier source)
-ipo	Interprocedural optimization (autorise l'«inlining» de sous-programmes présents dans des fichiers sources séparés)
-no-prec-div	Optimisation du calcul de divisions de nombres réels
-O3 -xCORE-AVX2 -ipo -no-prec-div	- Optimisation avancée, - Instructions AVX2 (optimisation pour Broadwell), - Optimisations inter-procédurales, - Optimisation du calcul de divisions de nombres réels

Options de compilation Intel : autres options utiles

Compilation

Option	Description
-cpp	Interprétation du preprocessing C (#define, #ifdef, etc.) dans les codes sources FORTRAN
-I /home/group/login/inc	Répertoire des fichiers inclus (insérés dans des sources avec «include 'fichier.h')
-module /home/group/login	Répertoire des fichiers .mod produits par la compilation de modules FORTRAN 90 (par défaut, ces fichiers sont produits dans le répertoire courant)
-p	Option de profilage (compilation et édition de lien). Après exécution, un profil s'obtient avec gprof : **gprof ./code.exe gmon.out > prof.out**

Options de compilation Intel : débogage

Débogage

- Les options suivantes sont conseillées, la première à tester en premier :

  -O2 -g -traceback

  -O2 -g -traceback -check-pointers=rw (C) / -O2 -g -traceback -check bounds (FORTRAN)

  -O0 -g -traceback -check-pointers=rw (C) / -O0 -g -traceback -check bounds (FORTRAN)

- «-traceback» permet d'obtenir la pile d'exécution jusqu'à l'exception rencontrée :

```
login@Myria-1:~: ./code.sh
forrtl: severe (41): insufficient virtual memory
Image              PC                Routine            Line        Source
code.exe           000000000047177D  Unknown               Unknown  Unknown
code.exe           0000000000470285  Unknown               Unknown  Unknown
code.exe           0000000000447359  Unknown               Unknown  Unknown
code.exe           00000000004340FF  Unknown               Unknown  Unknown
code.exe           00000000004103AE  Unknown               Unknown  Unknown
code.exe           0000000000403293  alloc_c_                 7  alloc_c.f90
code.exe           0000000000402FE6  MAIN__                  18  main_program.f90
code.exe           0000000000402ACC  Unknown               Unknown  Unknown
libc.so.6          00000038A0C1D994  Unknown               Unknown  Unknown
code.exe           00000000004029D9  Unknown               Unknown  Unknown
```

Options de compilation Intel : débogage (fin)

Débogage

- Détection à l'exécution des débordements d'indices de tableaux hors des limites déclarées (faire attention au ralentissement de l'exécution d'un calcul)
  - C : « -check-pointers=rw »
  - FORTRAN : « -check bounds »

- Autres options :
  - Détection des variables locales utilisées mais non initialisées :
    - C : « -check=uninit »
    - FORTRAN : « -check uninit »
  - Détection de toutes «floating point exceptions» (NaN, division par zéro, etc.)
    - C : « -fp-trap=all »
    - FORTRAN : « -fpe0 »

Outils de base pour le profilage de code

Profilage de code : gprof

Répartition du temps par sous-programme

- Outil gprof
  - Compilation et édition de lien :
    - Compilateur Intel : option «-pg» ou «-p»
    - Compilateur Gnu : option «-pg»
  - Après exécution du code :
    - Un fichier «gmon.out» est généré
    - Obtention du profil : **gprof /home/group/login/bin/code.exe gmon.out > prof.out**
  - Attention au caractère intrusif éventuel :
    - Exécuter un même cas avec et sans profilage pour s'assurer de la cohérence des performances
    - Si gprof est trop intrusif, contacter le support CRIANN : support@criann.fr

Profilage de code : gprof (suite)

Répartition du temps par sous-programme

- Exemple
  - «Flat profile» : répartition du temps par sous-programme

```
login@Myria-1:~: more prof.out
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  s/call   s/call  name
 43.66     11.37    11.37  2309348    0.00     0.00  sg_ffty_
 17.32     15.88     4.51  1571610    0.00     0.00  sg_fftpx_
 10.87     18.71     2.83    27286    0.00     0.00  sg_fftrisc_
  6.03     20.28     1.57        2    0.79     0.79  symkpt_
  5.72     21.77     1.49    40266    0.00     0.00  projbd_
  4.42     22.92     1.15   273780    0.00     0.00  mkffkg3_
  1.88     23.41     0.49    29655    0.00     0.00  opernl4a_
  1.69     23.85     0.44      414    0.00     0.05  cgwf_
  1.08     24.13     0.28    52387    0.00     0.00  indfftrisc_
  1.00     24.39     0.26    25101    0.00     0.00  opernl4b_
  0.77     24.59     0.20   100665    0.00     0.00  dotprod_g_
  0.61     24.75     0.16    20133    0.00     0.00  precon_
  0.50     24.88     0.13      437    0.00     0.00  pw_orthon_
```

Profilage de code : gprof (fin)

Graphe d'appel

- «Call graph» : renseigne sur la répartition des appels

```
granularity: each sample hit covers 2 byte(s) for 0.04% of 26.04 seconds

index % time    self  children    called     name
                0.00   26.04       1/1          main [2]
[1]     100.0   0.00   26.04       1        MAIN__ [1]
                0.00   24.43       1/1          driver_ [3]
                0.00    1.60       1/1          m_ab6_invars_mp_ab6_invars_load_ [19]
                0.00    0.00       1/1          chkinp_ [98]
-----------------------------------------------------------
                                                 <spontaneous>
[2]     100.0   0.00   26.04                 main [2]
                0.00   26.04       1/1          MAIN__ [1]
-----------------------------------------------------------
                0.00   24.43       1/1          MAIN__ [1]
[3]      93.8   0.00   24.43       1        driver_ [3]
                0.00   24.43       1/1          gstateimg_ [5]
                0.00    0.00   10/10259         status_ [121]
```

- fonction mère
- fonction courante
- fonction fille

Profilage de codes MPI

Approche simple avec Intel MPI

- Emploi de variables d'environnement d'Intel MPI
  - Ne s'applique pas aux codes hybrides MPI+OpenMP

```
#SBATCH --exclusive

#SBATCH -J "profile"
#SBATCH --output profile.o%J
#SBATCH --error profile.e%J
#SBATCH --partition 2tcourt
#SBATCH --time 00:20:00
#SBATCH --ntasks 112
#SBATCH --mem-per-cpu 3000

module load compilers/intel/2017
module load mpi/intelmpi/2017

export I_MPI_STATS=ipm
export I_MPI_STATS_FILE=mpistats.log

srun ./a.out > a.log
```

Profilage de codes MPI (suite)

Approche simple avec Intel MPI (suite)

- Première partie du rapport (fichier «mpistats.log»)

```
Intel(R) MPI Library Version 2017 Update 1

Summary MPI Statistics
Stats format: region
Stats scope : full

################################################################################
#
# command : /gpfs1/home/group/login/bin/mpicode.exe (completed)
# host    : my217/x86_64_Linux              mpi_tasks : 112 on 4 nodes
# start   : 04/28/17/09:46:26               wallclock : 244.342915 sec
# stop    : 04/28/17/09:50:30               %comm     : 25.41
# gbytes  : 0.00000e+00 total               gflop/sec : NA
#
################################################################################
# region  : *   [ntasks] = 112
#
#                         [total]       <avg>         min           max
# entries                 112           1             1             1
# wallclock               15576.1       243.377       242.947       244.343
# user                    15441.1       241.267       240.122       241.849
# system                  57.0291       0.89108       0.537019      1.72746
# mpi                     3957.55       61.8367       36.7655       99.774
# %comm                                 25.4077       15.1193       41.0673
# gflop/sec               NA            NA            NA            NA
# gbytes                  0             0             0             0
```

Profilage de codes MPI (fin)

Approche simple avec Intel MPI (fin)

- Deuxième partie du rapport (fichier «mpistats.log»)

```
#
#                            [time]       [calls]      <%mpi>      <%wall>
# MPI_Waitall                1665.18      1.17067e+08  42.08       10.69
# MPI_Allreduce              1472.26      2.8962e+07   37.20       9.45
# MPI_Recv                   228.282      2.0976e+07   5.77        1.47
# MPI_Isend                  183.943      2.82238e+08  4.65        1.18
# MPI_Bcast                  140.29       464960       3.54        0.90
# MPI_Irecv                  85.176       2.82238e+08  2.15        0.55
# MPI_Barrier                68.1029      2496         1.72        0.44
# MPI_Init                   63.2853      64           1.60        0.41
# MPI_Scan                   25.7659      451200       0.65        0.17
# MPI_Wtime                  15.3316      1.78266e+08  0.39        0.10
# MPI_Send                   9.93186      2.0976e+07   0.25        0.06
# MPI_Finalize               0.00045228   64           0.00        0.00
# MPI_Get_processor_name     3.24249e-05  64           0.00        0.00
# MPI_Comm_rank              1.50204e-05  64           0.00        0.00
# MPI_Type_size              8.82149e-06  128          0.00        0.00
# MPI_Comm_size              5.96046e-06  64           0.00        0.00
# MPI_TOTAL                  3957.55      9.31642e+08  100.00      25.41
################################################################################
```

Soumission des travaux

Modèles de script de soumission

Fichiers dans le répertoire /soft/slurm/criann_modeles_scripts

Domaine	Type de travail	Modèles de script
Général	Code séquentiel	job_serial.sl
	Code OpenMP	job_OpenMP.sl
	Code parallèle MPI compilé avec Intel MPI	job_MPI.sl
	Code parallèle hybride MPI/OpenMP	job_MPI_OpenMP.sl
	Code parallèle hybride MPI/OpenMP/GPU	job_MPI_OpenMP_GPU.sl

Domaine	Type de travail	Modèles de script
	Code parallèle hybride MPI/OpenMP sur Xeon Phi KNL	job_KNL.sl
Mécanique des fluides	Code Fluent	job_Fluent-<vers>.sl
	Code OpenFOAM	job_OpenFOAM-<vers>.sl
	Code FDS 5.5.0	job_FDS-5.5.0_serial.sl job_FDS-5.5.0_parallel.sl
Chimie quantique	Code Gaussian 03	job_Gaussian03.sl
	Code Scrhödinger Jaguar	job_Jaguar-<vers>.sl
	Code Gamess 2013	job_Gamess2013.sl

Domaine	Type de travail	Modèles de script
Dynamique moléculaire	Code DL_POLY classic 1.9	job_DLPOLY_class_1.9.sl
	Code NAMD 2.9	job_NAMD-2.9_plumed_MPI-SMP-GPU.sl
	Code NAMD 2.12	job_NAMD-2.12_MPI.sl job_NAMD-2.12_MP-SMP-GPUI.sl
	Code GROMACS 5.1.4 (simple précision)	job_Gromacs-5.1.4-sp-plumed_CPU.sl job_Gromacs-5.1.4-sp-plumed_GPU.sl

Calcul séquentiel

Modèle de script /soft/slurm/criann_modeles_scripts/job_serial.sl

```
#!/bin/bash
# Slurm submission script, serial job
# CRIANN v 1.00 - Jan 2017
# support@criann.fr

# Allow shared resources for intranode job

# Job name
#SBATCH -J ‘’serial.run”
# Batch output file
#SBATCH --output serial.o%J
# Batch error file
#SBATCH --error serial.e%J

# Partition (submission class)
#SBATCH --partition 2tcourt

# Job time (hh:mm:ss)
#SBATCH --time 12:00:00

# ------------------------------
# Job maximum memory (MB)
#SBATCH --mem 3000
# ------------------------------
#SBATCH --mail-type ALL
# User e-mail address
##SBATCH --mail-user firstname.name@address.fr

# Compiler environment
module load compilers/intel/2017

# Copy input data and go to working directory
cp *.in $LOCAL_WORK_DIR

cd $LOCAL_WORK_DIR
echo Working directory : $PWD

# Serial code execution
/home/group/login/bin/serialcode.exe > serialcode.log

# Move output data to target directory
mkdir $SLURM_SUBMIT_DIR/$SLURM_JOB_ID
mv *.log *.dat $SLURM_SUBMIT_DIR/$SLURM_JOB_ID
```

Calcul OpenMP

Modèle de script /soft/slurm/criann_modeles_scripts/job_OpenMP.sl

```
#!/bin/bash
# Slurm submission script, OpenMP job
# CRIANN v 1.00 - Jan 2017
# support@criann.fr
# Allow shared resources for intranode job

# Job name
#SBATCH -J “openmp.run”
# Batch output file
#SBATCH --output openmp.o%J
# Batch error file
#SBATCH --error openmp.e%J

# Partition (submission class)
#SBATCH --partition 2tcourt
# Job time (hh:mm:ss)
#SBATCH --time 12:00:00

# ------------------------------
# OpenMP threads number
#SBATCH --cpus-per-task 8
# Job maximum memory (MB)
#SBATCH --mem 3000
# ------------------------------
#SBATCH --mail-type ALL
# User e-mail address
##SBATCH --mail-user firstname.name@address.fr

# Compiler environment
module load compilers/intel/2017

# Copy input data and go to working directory
cp *.in $LOCAL_WORK_DIR

cd $LOCAL_WORK_DIR
echo Working directory : $PWD

# Serial code execution
/home/group/login/bin/openmpcode.exe > openmpcode.log

# Move output data to target directory
mkdir $SLURM_SUBMIT_DIR/$SLURM_JOB_ID
mv *.log *.dat $SLURM_SUBMIT_DIR/$SLURM_JOB_ID
```

Calcul parallèle MPI

Modèle de script /soft/slurm/criann_modeles_scripts/job_MPI.sl

```
#!/bin/bash
# Slurm submission script, MPI job
# CRIANN v 2.00 - Feb 2020
# support@criann.fr
# Not shared resources
#SBATCH --exclusive
# Job name
#SBATCH -J “mpi.run”
# Batch output file
#SBATCH --output mpi.run.o%J
# Batch error file
#SBATCH --error mpi.run.e%J

# Partition (submission class)
#SBATCH --partition 2tcourt
# Job time (hh:mm:ss)
#SBATCH --time 12:00:00

# ------------------------------
# MPI tasks number
#SBATCH --ntasks 280
# Maximum memory per compute node (MB)
#SBATCH --mem 120000
# ------------------------------
```

```
# MPI tasks per compute node
# (on Myria if a MPI task needs > 4300MB of memory)
##SBATCH --ntasks-per-node 14

#SBATCH --mail-type ALL
# User e-mail address
##SBATCH --mail-user firstname.name@address.fr

# Compiler / MPI environments
# ---------------------------------------
module load compilers/intel/2017
module load mpi/intelmpi/2017
# ---------------------------------------
# Copy input data and go to working directory
cp *.in $LOCAL_WORK_DIR
cd $LOCAL_WORK_DIR
echo Working directory : $PWD

# MPI code execution
srun /home/group/login/bin/mpicode.exe > mpicode.log

# Move output data to target directory
mkdir $SLURM_SUBMIT_DIR/$SLURM_JOB_ID
mv *.log *.dat $SLURM_SUBMIT_DIR/$SLURM_JOB_ID
```

Calcul parallèle MPI : commande srun «multi-programmes»

Couplage d'applications

- Dans le cadre de couplages d'applications, des commandes «mpirun» en mode MPMD (Multi Program Multiple Data) sont usitées avec d'autres environnements de batch que Slurm :

  mpirun -n 56 ./a.out : -n 28 ./b.out

- Avec Slurm, la commande équivalente est la suivante (à la fin du script précédent)

  srun **--multi-prog** myrun.conf

  où myrun.conf est un fichier texte contenant :

```
0-55    ./a.out
56-83   ./b.out
```

  le nombre total de processus parallèles MPI étant spécifié par «#SBATCH --ntasks 84»

- Pour information, les coupleurs de référence suivants sont disponibles sur Myria
  - OpenPALM 4.2.3 (http://www.cerfacs.fr/globc/PALM_WEB/)
  - OASIS3-MCT 3.0 (https://portal.enes.org/oasis)

Calcul parallèle hybride MPI/OpenMP

Modèle de script /soft/slurm/criann_modeles_scripts/job_MPI_OpenMP.sl

```
#!/bin/bash
# Slurm submission script, MPI/OpenMP job
# CRIANN v 2.00 - Feb 2020
# support@criann.fr
# Not shared resources
#SBATCH --exclusive
# Job name
#SBATCH -J “mpi-openmp.run”
# Batch output file
#SBATCH --output mpi-openmp.run.o%J
# Batch error file
#SBATCH --error mpi-openmp.run.e%J
# Partition (submission class)
#SBATCH --partition 2tcourt
# Job time (hh:mm:ss)
#SBATCH --time 12:00:00
# ------------------------------
# MPI tasks number
#SBATCH --ntasks 280
# CPUs per MPI task
# (OMP_NUM_THREADS is set to the same value)
#SBATCH --cpus-per-task 2
# Maximum memory per compute node (MB)
#SBATCH --mem 120000
# ------------------------------
```

```
# MPI tasks per compute node
# (on Myria if a MPI task needs > 4300MB of memory)
##SBATCH --ntasks-per-node 14

#SBATCH --mail-type ALL
# User e-mail address
##SBATCH --mail-user firstname.name@address.fr

# Compiler / MPI environments
# ---------------------------------------
module load compilers/intel/2017
module load mpi/intelmpi/2017
# ---------------------------------------
# Copy input data and go to working directory
cp *.in $LOCAL_WORK_DIR
cd $LOCAL_WORK_DIR
echo Working directory : $PWD

# MPI/OpenMP code execution
srun /home/group/login/bin/mpicode.exe > mpicode.log

# Move output data to target directory
mkdir $SLURM_SUBMIT_DIR/$SLURM_JOB_ID
mv *.log *.dat $SLURM_SUBMIT_DIR/$SLURM_JOB_ID
```

Code accéléré sur GPU

Directives

- Pour exécuter un code accéléré sur GPU, qu'il soit séquentiel, OpenMP, MPI, MPI / OpenMP, il suffit d'adapter le modèle de script de soumission correspondant (parmi ceux décrits précédemment : job_serial.ll, job_OpenMP.ll, job_MPI.ll, job_MPI_OpenMP.ll) :
  - Ajouter les lignes suivantes :

```
# GPU architecture and number
# ------------------------------------
# Partition (submission class)
##SBATCH --partition gpu_k80
#SBATCH --partition gpu_p100

# GPUs per compute node
#      gpu:4 (maximum) for gpu_k80
#      gpu:2 (maximum) or gpu_p100
##SBATCH --gres gpu:4
#SBATCH --gres gpu:2
# ------------------------------------
```

Code accéléré sur GPU (fin)

Environnement

- Si le code applique CUDA ou OpenCL ajouter les lignes suivantes avant la ligne d'exécution du code :

```
module load cuda/8.0
```

- Si le code applique CUDA FORTRAN, ajouter les lignes suivantes avant la ligne de commande d'exécution du code

```
# PGI CUDA Fortran
module load pgi/16.10
```

- Dans le cas particulier d'une application **mono-GPU**, ne pas spécifier «#SBATCH --exclusive»

```
#!/bin/bash

#SBATCH --partition gpu_p100
#SBATCH --gres gpu:1
```

- Slurm assure le non partage des unités GPU d'un même nœud de calcul (en gérant lui-même la variable d'environnement $CUDA_VISIBLE_DEVICES)

Exemple : code MPI+OpenMP / CUDA

Modèle de script /soft/slurm/criann_modeles_scripts/job_MPI_OpenMP_GPU.sl

```
#!/bin/bash
# Slurm submission script, MPI/OpenMP/GPU job
# CRIANN v 2.00 - Feb 2020
# support@criann.fr
# Not shared resources
#SBATCH --exclusive
# Job name
#SBATCH -J “mpi-opemp-gpu.run”
# Batch output file
#SBATCH --output mpi-openmp-gpu.run.o%J
# Batch error file
#SBATCH --error mpi-openmp-gpu.run.e%J
# Job time (hh:mm:ss)
#SBATCH --time 12:00:00
# GPUs architecture and number
# ---------------------------
# Partition : gpu_k80 / gpu_p100 / gpu_v100
#SBATCH --partition gpu_p100

# GPUs per compute node
#   gpu:4 (maximum) for gpu_k80
#   gpu:2 (maximum) for gpu_p100
#   gpu:4 (maximum) for gpu_v100
##SBATCH --gres gpu:4
#SBATCH --gres gpu:2
# Compute nodes number
#SBATCH --nodes 4
# MPI tasks per compute node
#SBATCH --ntasks-per-node 2
# CPUs per MPI task (= OMP_NUM_THREADS)
#SBATCH --cpus-per-task 2
# Maximum memory per compute node (MB)
#SBATCH --mem 120000
# ------------------------------
#SBATCH --mail-type ALL
##SBATCH --mail-user firstname.name@address.fr
# Compiler / MPI environments
# ---------------------------------------
module load compilers/intel/2017
module load mpi/intelmpi/2017
module load cuda/10.0
# ---------------------------------------
# Copy input data and go to working directory
cp *.in $LOCAL_WORK_DIR
cd $LOCAL_WORK_DIR ; echo Working directory : $PWD
# MPI/OpenMP code execution
srun /home/group/login/bin/mpicode.exe > mpicode.log
# Move output data to target directory
mkdir $SLURM_SUBMIT_DIR/$SLURM_JOB_ID
mv *.log *.dat $SLURM_SUBMIT_DIR/$SLURM_JOB_ID
```

Application OpenFOAM

Modèle de script /soft/slurm/criann_modeles_scripts/job_OpenFOAM-<vers>.sl

```
#!/bin/bash
# Slurm submission script, OpenFOAM v1612+ parallel job
# CRIANN v 2.00 - Feb 2020
# support@criann.fr
# Not shared resources
#SBATCH --exclusive
# Job name
#SBATCH -J “foam.run”
# Batch output file
#SBATCH --output foam.run.o%J
# Batch error file
#SBATCH --error foam.run.e%J

# Partition (submission class)
#SBATCH --partition 2tcourt
# Job time (hh:mm:ss)
#SBATCH --time 12:00:00

# ------------------------------
# OpenFOAM parallel processes
#SBATCH --ntasks 140
# Maximum memory per compute node (MB)
#SBATCH --mem 120000
# ------------------------------
```

```
# MPI tasks per compute node
# (on Myria if a MPI task needs > 4300MB of memory)
##SBATCH --ntasks-per-node 14

#SBATCH --mail-type ALL
# User e-mail address
##SBATCH --mail-user firstname.name@address.fr

# OpenFOAM v1612+ environment
# ----------------------------------------
module load openfoam/1612+
# ----------------------------------------
# Copy input data and go to working directory
cp -r *FOAM_CASE_INPUT/* $LOCAL_WORK_DIR

cd $LOCAL_WORK_DIR
echo Working directory : $PWD

# OpenFOAM solver parallel execution
srun interFoam -parallel > openfoam.log

# Move output data to target directory
mkdir $SLURM_SUBMIT_DIR/$SLURM_JOB_ID
mv * $SLURM_SUBMIT_DIR/$SLURM_JOB_ID
```

Application OpenFOAM (suite)

Personnalisation

- Différentes versions d'OpenFOAM sont mutualisées (dans /soft)

- Un utilisateur peut ajouter un module optionnel dans son espace personnel

- Exemple (module cfMesh)

```
login@Myria-1:~/VISU/cfMesh-v1.1.1:$ ls
Allwclean  Allwmake  executables  meshLibrary  python  README tutorials  userGuide  utilities

login@Myria-1:~/VISU/cfMesh-v1.1.1:$ module load openfoam/3.0.1
load mpi/intelmpi/2017 environment
openfoam/3.0.1 environment

login@Myria-1:~/VISU/cfMesh-v1.1.1:$ env | grep FOAM_INST_DIR
FOAM_INST_DIR=/gpfs2/soft/OpenFOAM-3.0.1

login@Myria-1:~/VISU/cfMesh-v1.1.1:$ env | grep WM_PROJECT_USER_DIR
WM_PROJECT_USER_DIR=/home/group/login/OpenFOAM/login-3.0.1

login@Myria-1:~/VISU/cfMesh-v1.1.1: nohup ./Allwmake > Allwmake.LOG < /dev/null &
```

Application OpenFOAM (fin)

Personnalisation

- Une fois la compilation de cfMesh effectuée, ce module est alors installé dans un répertoire du /home de l'utilisateur : *$WM_PROJECT_USER_DIR/platforms*

```
login@Myria-1:~:$ module load openfoam/3.0.1
load mpi/intelmpi/2017 environment
openfoam/3.0.1 environment

login@Myria-1:~:$ ls /home/group/login/OpenFOAM/login-3.0.1/platforms/linux64GccDPInt32Opt/
bin  lib

login@Myria-1:~:$ type cartesianMesh
cartesianMesh is /home/group/login/OpenFOAM/login-3.0.1/platforms/linux64GccDPInt32Opt/bin/cartesianMesh

login@Myria-1:~:$ type fireFoam
fireFoam is /soft/OpenFOAM-3.0.1/OpenFOAM-3.0.1/platforms/linux64GccDPInt32Opt/bin/fireFoam
```

Les partitions (classes de soumission)

Travaux de debug

Classe	Description	Conseils
debug	• temps ≤ 30 minutes • les travaux accèdent à 363 nœuds de calcul «Broadwell» • 1 travail est limité à 5 nœuds (140 cœurs et 585 Go de mémoire / 28 cœurs et 1 To de mémoire)	L'un des 363 serveurs associé à cette partition est équipé d'un To de mémoire

Les partitions (classes de soumission) (suite)

Travaux sur architecture standard

(*) L'horaire du week-end (partition 2tcourt) commence le vendredi soir à 17H et se termine le dimanche soir à 20H

Classe	Description	Conseils
2tcourt	• temps ≤ 12 heures • les travaux accèdent à 332 nœuds de calcul «Broadwell» • 1 travail est limité en semaine à 75 nœuds (2100 cœurs et 9600 Go de mémoire) • 1 travail est limité en week-end (*) à 150 nœuds (4200 cœurs et 19200 Go de mémoire)	• Mémoire par cœur ≤ 4 Go dans le cas MPI
tcourt	• temps ≤ 24 heures • les travaux accèdent à 322 nœuds de calcul «Broadwell» • 1 travail est limité à 75 nœuds (2100 cœurs, 9600 Go de mémoire)	Mémoire par cœur ≤ 4 Go dans le cas MPI

Classe	Description	Conseils
court	• 24 heures < temps ≤ 48 heures • les travaux accèdent à 100 nœuds de calcul «Broadwell» • 1 travail est limité à 20 nœuds (560 cœurs, 2560 Go de mémoire)	Mémoire par cœur ≤ 4 Go dans le cas MPI
long	• 48 heures < temps ≤ 100 heures • les travaux accèdent à 50 nœuds de calcul «Broadwell» • 1 travail est limité à 10 nœuds (280 cœurs, 1280 Go de mémoire)	• Mémoire par cœur ≤ 4 Go dans le cas MPI
tlong	• 100 heures < temps ≤ 300 heures • les travaux accèdent à 10 nœuds de calcul «Broadwell» • 1 travail est limité à 2 nœuds (56 cœurs, 256 Go de mémoire)	• Mémoire par cœur ≤ 4 Go dans le cas MPI
tcourt_intra	• temps ≤ 24 heures • les travaux accèdent à 134 nœuds de calcul «Broadwell» • 1 travail est limité à 1 nœud (28 cœurs, 128 Go de mémoire)	

Les partitions (classes de soumission) (fin)

Travaux sur architectures spécifiques

Classe	Description
disque	• temps ≤ 300 heures • les travaux accèdent à 12 nœuds de calcul «Broadwell» (20 To de disques par nœud) • 1 travail est limité à 1 nœud (28 cœurs) • 12 nœuds ont 128 Go de mémoire • 1 nœud a 1 To de mémoire
smplarge	• temps ≤ 3 heures • 1 travail peut accéder à la totalité du serveur SMP «Haswell» (256 cœurs, 3,99 To de mémoire)
smplong	• temps ≤ 72 heures • 1 travail est limité à la moitié du serveur SMP «Haswell» (128 cœurs, 2 To de mémoire)

Classe	Description
knl	• temps ≤ 300 heures • les travaux accèdent à 10 nœuds de calcul «Xeon Phi KNL» • 1 travail est limité à 4 nœuds (256 cœurs x 4 threads, 384 Go de mémoire)
gpu_k80	• temps ≤ 48 heures • les travaux accèdent à 9 nœuds de calcul «Broadwell» (avec GPUs Kepler K80) • 1 travail est limité à 8 nœuds (224 cœurs, 1024 Go de mémoire)
gpu_p100	• temps ≤ 48 heures • les travaux accèdent à 9 nœuds de calcul «Broadwell» (avec GPUs Pascal P100) • 1 travail est limité à 8 nœuds (224 cœurs, 1024 Go de mémoire)

Classe	Description
gpu_v100	• temps ≤ 48 heures • les travaux accèdent à 5 nœuds de calcul «SkyLaKe» (avec GPUs Volta V100) • 1 travail est limité à 5 nœuds (160 cœurs, 900 Go de mémoire)
gpu_all	• temps ≤ 48 heures • les travaux accèdent à 18 nœuds de calcul «Broadwell» (avec GPUs K80 ou P100) • 1 travail est limité à 1 nœud (28 cœurs, 128 Go de mémoire)
gpu_court	• temps ≤ 4 heures • les travaux accèdent à 4 nœuds de calcul «Broadwell» (avec GPUs K80) • 1 travail est limité à 1 nœud (28 cœurs, 128 Go de mémoire)

Classe	Description
visu	• temps ≤ 12 heures • les travaux accèdent à 2 nœuds «Broadwell» à 256 Go de RAM chacun • 1 travail est limité à 4 cœurs et 62,5 Go (64000 Mo) de mémoire

Commandes du batch

Slurm

- Les commandes les plus utiles sont les suivantes

Action	Commande
Visualisation de la charge du cluster	slurmtop -f -
Caractéristiques des partitions	sinfo
Lancement d'un travail	sbatch *script_sounission.sl*
Visualisation de la file d'attente des travaux	squeue
Visualisation de la file de ses propres travaux	squeue -u *login*

Action	Commande
Obtention exhaustive des caractéristiques d'un travail : ressources CPU, mémoire et temps demandé, nœuds de calcul utilisés par un travail en cours d'exécution, date de début d'exécution d'un travail en cours, etc.	scontrol show job *job_id*
Prévision d'horaire de passage d'un travail	squeue --start --job *job_id*
Prévision d'horaire de passage de ses propres travaux	squeue -u *login* --start
Arrêt d'un travail	scancel *job_id*

Commandes du batch (fin)

Slurm

- Le tableau précédent fait mention de l'identifiant d'un travail «*job_id*»
- Cet identifiant est indiqué en première colonne de la **sortie de la commande squeue** :

```
login@Myria-1:~:$ squeue -u login
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
             44617    disque jobname1    login PD       0:00      1 (Priority)
             43454    disque jobname2    login  R 2-01:55:54      1 my371
```

Statut d'un travail soumis

https://slurm.schedmd.com/squeue.html

```
login@Myria-1:~:$ squeue -u login
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
             44715  smpcourt jobname1    login PD       0:00      1 (QOSMaxCpuPerUserLimit)
             44717    2tcourt jobname2    login PD       0:00      1 (Priority)
             44713      long  jobname3    login  R 2-01:55:54      1 my371
```

- Statut (colonne «ST»)
  - PD : pending (en attente de disponibilité de la ressource)
  - R : running
  - CG : completing
  - CD : completed

- Raison d'une mise en attente (colonne «NODELIST(REASON)»)
  - (Resources) : en attente de disponibilité de la ressource
  - (Priority) : un ou plusieurs travaux sont plus prioritaires dans la partition considérée
  - (Dependency) : dépendance entre travaux (voir «travaux multi-étapes» ci-après)
  - (QOSMaxCpuPerUserLimit) : la QOS (Qualité Of Service) de la partition considérée a atteint son maximum de ressources utilisables simultanément

Mode interactif

Mise au point

- Les calculs interactifs se font sur les frontales de Myria, pour de la mise au point ou des pré- ou post-traitements consommant peu de ressources.
  - Ne pas dépasser 4 threads pour les codes OpenMP et 4 processus pour les codes MPI

- Calcul MPI interactif (commande pouvant être précédée de «module load mpi/openmpi/2.0.1» si besoin ; sinon, Intel MPI 2017 s'applique) :

```
login@Myria-1:~: mpirun -np 4 ./mpicode.exe >& mpicode.log &
```

Visualisation graphique

Pré - ou post - traitements

- Documentation : https://services.criann.fr/services/calcul/cluster-myria/utilisation/visu/

- Ressource
  - 2 serveurs dotés chacun de 28 cœurs, d'une carte graphique (NVIDIA K80) et de 256 Go de mémoire
  - Limitation par travail : 4 cœurs, 62,5 Go de mémoire et 12 heures de temps (partition «visu»)

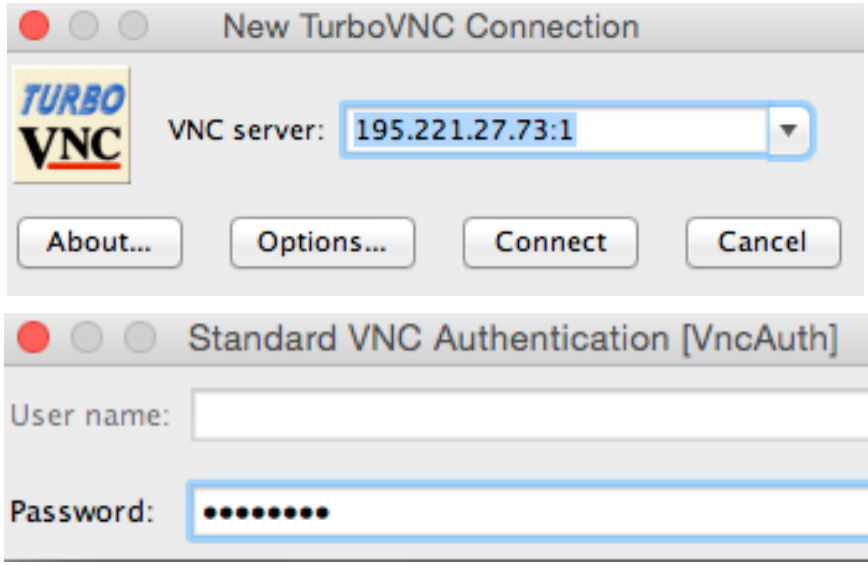
- Soumission en batch : commande «*startvisu*»
  - Par défaut : 4 cœurs, 62,5 Go de mémoire et 4 heures
  - Extension du temps : «*startvisu --time 8:00:00*» par exemple (limite : 12 heures)
  - Spécification de l'un des deux serveurs de visualisation (rarement utile)
    - «*startvisu -w visu1*» ou «*startvisu -w visu2*»
  - Spécification d'un identifiant de réservation (lors de formations) : «startvisu -r <reserv_id>»

Visualisation graphique (suite)

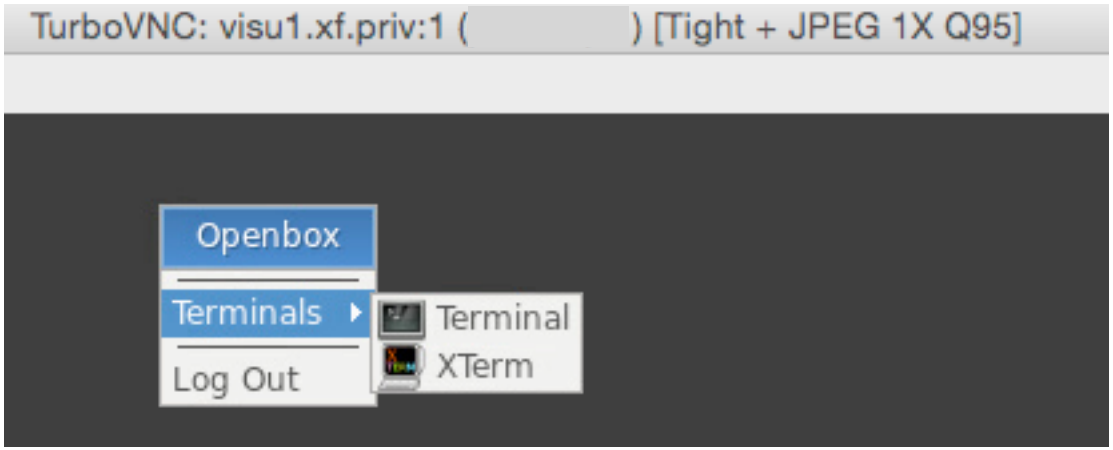
Exemple

```
login@Myria-2:~:$ startvisu

Submitted batch job 46442

Your job ID is 46442, waiting for Slurm to
launch it...

You have to follow the instructions which will
be printed to your terminal to connect to your
VNC server.

Waiting for the VNC session to start....
You can now launch your TurboVNC viewer on your
computer

       Address: 195.221.27.73:1 (visu1)

       Password controller: 23119135 (you can
control the session)
       Password viewer:     11871211 (you can
only view the session)


You may have multiple controller/viewer
sessions.

Eventually, you can terminate your session by 2
ways:
   - in your desktop, you can log out
   - type the following command 'scancel 46442'
```

Client VNC Sur poste local



Une fois connecté : ouvrir un terminal



Visualisation graphique (fin)

Exemple (fin)

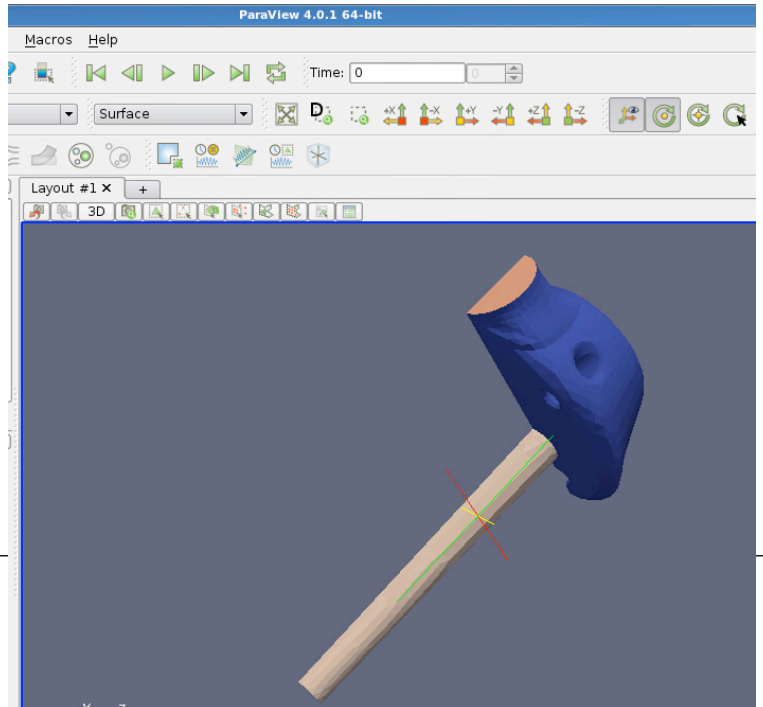
TurboVNC: visu1.xf.priv:1 ( ) [Tight + JPEG 1X Q95]

Ctrl Alt

@visu1:~/VISU/demo/sawOctree

File Edit View Search Terminal Help

```
bash-4.2$ source /etc/bashrc
         @visu1 ~]$ cd VISU/demo/
bunnyOctree/ sawOctree/
         @visu1 ~]$ cd VISU/demo/sawOctree/
         @visu1 sawOctree]$ module load paraview/4.0.1_Serial
load Paraview 4.0.1 Serial environment
On visualization server, don't forget to use "vglrun paraview"...
         @visu1 sawOctree]$ vglrun paraview &
```



Bonnes pratiques de soumission

Généralités

Slurm

- Respecter l'utilisation de la ligne suivante figurant dans les modèles de script d'exécution, la partition /dlocal du système de fichiers n'étant pas soumise aux quotas utilisateurs :

```
cd $LOCAL_WORK_DIR
```

- Séparer en plusieurs travaux les étapes de calcul (pré-traitement, solveur, post-traitement) nécessitant des quantités de ressources différentes
  - => Se référer au paragraphe «Soumission de travaux multi-étapes»

Lecture du rapport de consommation

Slurm

- Prêter attention au rapport de consommation de ressources (**mémoire** notamment), fourni dans le fichier de sortie standard du batch (nommé <job_name>.o<num> par défaut)

```
########################################
   Comptabilité du calcul 45460
########################################
JobID              45460.2
JobIDRaw           45460.2
JobName            IMB-MPI1_impi
Partition
MaxVMSize          1020872K
MaxVMSizeNode      my155
MaxVMSizeTask      59
AveVMSize          37516.29K
MaxRSS             618500K
```

Grandeur	Signification
MaxVMSize	Quantité de mémoire virtuelle (allouée) par le processus parallèle ayant consommé le plus de mémoire virtuelle (Ko)
MaxRSS	Quantité de mémoire résidente (utilisée en RAM) par le processus parallèle (tâche MPI) ayant consommé le plus de mémoire résidente (Ko)

- Slurm contrôle la mémoire résidente (MaxRSS), en fonction de la demande
  - «#SBATCH --mem» : mémoire demandée pas nœud de calcul

  - **Si MaxRSS ⩽ 4300MB, mais que les nœuds de calcul ont été dépeuplés («#SBATCH --ntasks-per-node» < 28), ne plus les dépeupler pour les soumissions ultérieures de travaux de même type**

Rapport synthétique de consommation

Slurm

- L'ajout de la commande «sacct» suivante, en dernière ligne d'un script de soumission, est utile

```
#!/bin/bash

# Job name
#SBATCH --J job1
# ...
#SBATCH --ntasks 140
# ...
srun ./code1.exe

srun ./code2.exe

sacct --format=AllocCPUs,AveCPU,MaxRSS,MaxVMSize,JobName -j $SLURM_JOB_ID
```

- Le fichier de sortie de Slurm contient alors

```
 AllocCPUS     AveCPU     MaxRSS  MaxVMSize     JobName
---------- ---------- ---------- ----------  ----------
       140                                        job1
       140   00:02:12     19028K    353168K  code1.exe
       140   00:18:38   1204832K   1627044K  code2.exe
```

Suivi de consommation d'un travail en cours

Slurm

- La commande sstat est utile

```
login@Myria-2:~:$ squeue -u $USER
             JOBID PARTITION     NAME     USER  ST       TIME  NODES  NODELIST(REASON)
             45880   2tcourt     job3    login  R       5:05      2   my[141-142]

login@Myria-2:~: sstat --format='AveCPU,MaxRSS,MaxVMSize' -j 45880
   AveCPU     MaxRSS    MaxVMSize
 ---------- ---------- ----------
 00:03.000    39928K    444816K
```

Gestion des ressources

Slurm

- L'ajustement des demandes aux besoins permet de ne pas monopoliser inutilement une quantité de mémoire ou un nombre de processeurs surestimé(e) pour un calcul

- Réaliser des tests d'accélération parallèle des codes MPI
  - Détermination du nombre de processus pertinent pour une taille donnée de problème (taille de maillage par exemple)
  - Si le nombre de processus est surestimé, l'utilisateur :
    - gaspille son quota d'heures.CPU
    - prive inutilement les autres utilisateurs d'une certaine quantité de ressources

- L'utilisation d'une durée inférieure ou égale à 24 heures réduit le temps passé en file d'attente (étant donné la répartition actuelle des partitions)
  - Pour des calcul d'une durée supérieure à 24 heures, soumettre plusieurs travaux successifs d'une durée inférieure ou égale à 24 heures, un travail utilisant des fichiers de reprise générés par le travail précédent

Exécution de travaux : aspects avancés

Travaux multi - étapes : exemple

Usage avancé de Slurm

- Exemple
  1. Partition de maillage : travail séquentiel avec beaucoup de mémoire
  2. Solveur : travail MPI multi-nœud avec peu de mémoire par processus parallèle
  3. Archivage et compression des fichiers de résultats : travail séquentiel

```
#!/bin/bash

# Step 1 : preprocessing

#SBATCH -J "prepare"
#SBATCH --output prepare.o%J
#SBATCH --error prepare.e%J
#SBATCH --partition tcourt_intra
#SBATCH --time 01:30:00
#SBATCH --mem 10000

cp *.in $LOCAL_WORK_DIR
cd $LOCAL_WORK_DIR

/home/group/login/bin/preprocess > pre.log
```

```
#!/bin/bash

# Step 2 : solver

#SBATCH --exclusive
#SBATCH -J "solver"
#SBATCH --output solver.o%J
#SBATCH --error solver.e%J
#SBATCH --partition 2tcourt
#SBATCH --time 12:00:00
#SBATCH --ntasks 280
#SBATCH --mem-per-cpu 3000

mv /dlocal/run/$PREV_JOB_ID/* $LOCAL_WORK_DIR
cd $LOCAL_WORK_DIR

srun /home/group/login/bin/solver > solver.log
```

Travaux multi - étapes : exemple (fin)

Usage avancé de Slurm

```
#!/bin/bash

# Step 3 : archiving data

#SBATCH -J "post"
#SBATCH --output post.o%J
#SBATCH --error post.e%J
#SBATCH --partition tcourt_intra
#SBATCH --time 00:30:00
#SBATCH --mem 3000

mv /dlocal/run/$PREV_JOB_ID/* $LOCAL_WORK_DIR
cd $LOCAL_WORK_DIR

tar cvfz RESULTS_DIR.tgz RESULTS_DIR

# Move output data to target directory
mkdir $SLURM_SUBMIT_DIR/$SLURM_JOB_ID
mv RESULTS_DIR.tgz $SLURM_SUBMIT_DIR/$SLURM_JOB_ID
```

Travaux multi-étapes : exécution

Soumission de chaque étape

```
login@Myria-2:~/SLURM/MULTI-ETAPES:$ sbatch prepare.sl
Submitted batch job 45184

login@Myria-2:~/SLURM/MULTI-ETAPES:$ PREV_JOB_ID=45184 sbatch --dependency afterok:45184 solver.sl
Submitted batch job 45185

login@Myria-2:~/SLURM/MULTI-ETAPES:$ squeue -u login
             JOBID PARTITION     NAME   USER   ST       TIME  NODES  NODELIST(REASON)
             45185   2tcourt   solver  login   PD       0:00      2 (Dependency)
             45184 tcourt_in  prepare  login   R        0:36      1  my141

login@Myria-2:~/SLURM/MULTI-ETAPES:$ PREV_JOB_ID=45185 sbatch --dependency afterok:45185 post.sl
Submitted batch job 45187

login@Myria-2:~/SLURM/MULTI-ETAPES:$ squeue -u login
             JOBID PARTITION     NAME     USER  ST       TIME  NODES NODELIST(REASON)
             45185   2tcourt   solver    login  PD       0:00      2 (Dependency)
             45187 tcourt_in     post    login  PD       0:00      1 (Dependency)
             45184 tcourt_in  prepare    login   R       1:02      1 my141
```

Travaux multi-étapes : exécution (fin)

Automatisation

- La seule exécution du script Shell qui suit soumet chacune des étapes

```
#!/bin/bash

set –x

# Job 1
JOB1_ID=$(sbatch prepare.sl | awk '{print $NF}')

# Job 2
JOB2_ID=$(PREV_JOB_ID=$JOB1_ID sbatch ––dependency afterok:$JOB1_ID solver.sl | awk '{print $NF}')

# Job 3
PREV_JOB_ID=$JOB2_ID sbatch ––dependency afterok:$JOB2_ID post.sl

squeue –u $USER
```

- Remarque
  - Avec l'ajout de l'option « **--kill-on-invalid-dep=yes** », dans les commandes sbatch des étapes «Job 2» et «Job 3» précédentes, Slurm tue un travail soumis à dépendance si le travail précédent a échoué

Dépendances entre travaux

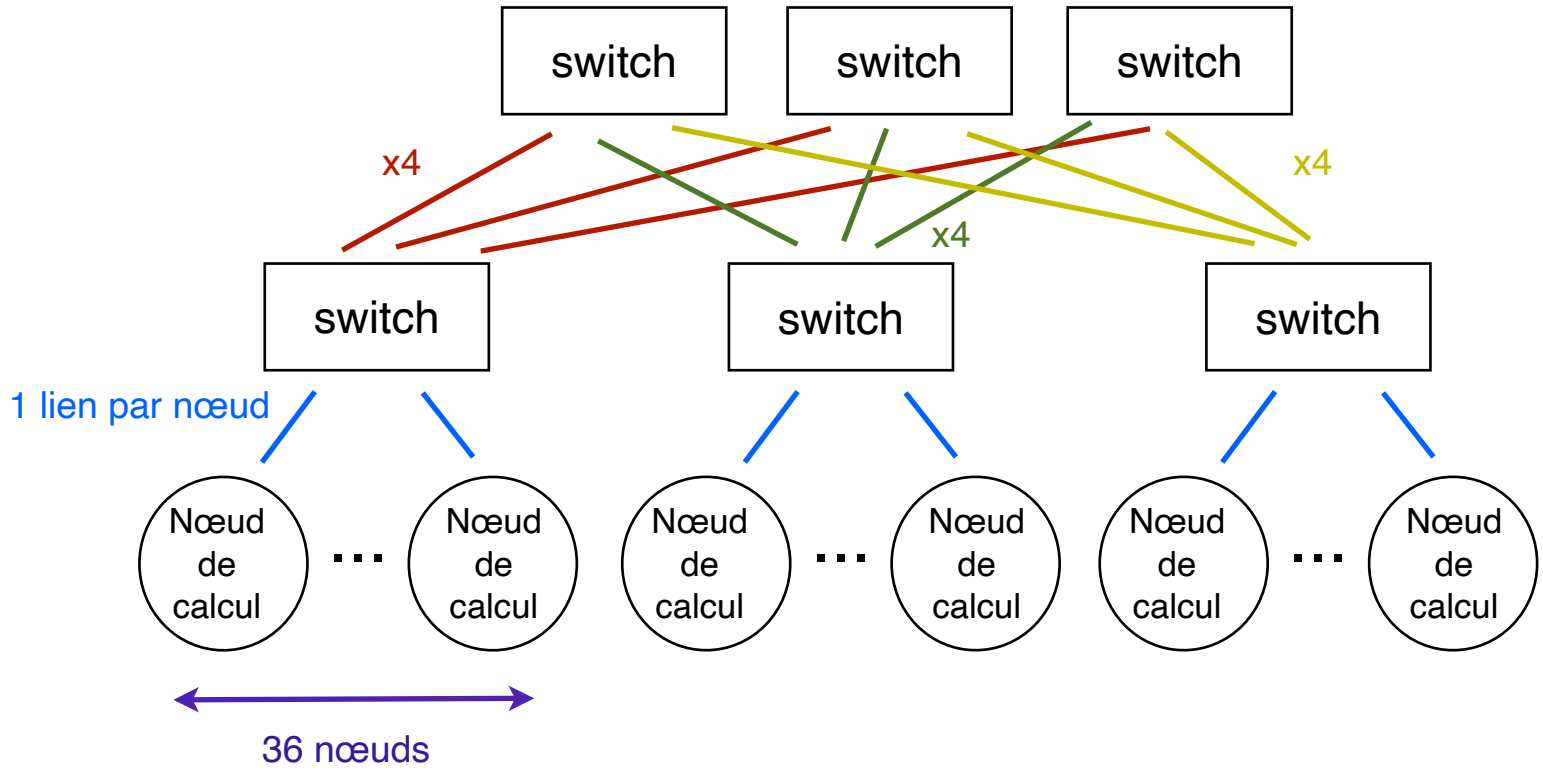
Autres possibilités (voir https://slurm.schedmd.com/sbatch.html)

- L'option --dependency (ou -d) de la commande sbatch admet notamment les sous-options suivantes
  - after:job_id
    - This job can begin execution after the specified job has begun execution.
  - afterany:job_id
    - This job can begin execution after the specified job has terminated.
  - afternotok:job_id
    - This job can begin execution after the specified job has terminated in some failed state (non-zero exit code, node failure, timed out, etc).
  - **afterok:job_id**
    - This job can begin execution after the specified job has successfully executed (ran to completion with an exit code of zero).
  - **singleton**
    - This job can begin execution after any previously launched jobs *sharing the same job name* and user have terminated.

Prise en compte de la topologie réseau

Réseau Omni-Path de Myria

- Myria possède un réseau Omni-Path en arbre (*fat tree*) avec un facteur de blocage de 1 à 3



- Des groupes de 36 nœuds (28 cœurs Broadwell par nœud) partagent un même switch
  - **Un travail de 1008 cœurs peut être exécuté dans un seul switch**

Prise en compte de la topologie réseau (fin)

Soumission «intra-switch» (nombre de cœurs inférieur ou égal à 1008)

- Commande *sbatch job_MPI.sl*
  - Si la distribution du calcul sur plusieurs switches réduit son temps d'attente, une telle distribution est appliquée

- Commande *sbatch --switches=**1** job_MPI.sl*
  - L'usage de nœuds de calcul répartis dans **un seul switch** est forcé (même si cela allonge le temps d'attente, quelle que soit la durée de cet allongement)

- Commande *sbatch --switches=1@**2:00:00** job_MPI.sl*
  - Si l'usage de nœuds de calcul répartis dans un seul switch est possible avec un temps d'attente inférieur ou égal à **2 heures**, cet usage est appliqué
  - Au-delà de 2 heures de temps d'attente, si une répartition sur plusieurs switches réduit le délai, une telle répartition est appliquée

- (source : https://slurm.schedmd.com/sbatch.html)

Placement des processus

Fonctions de Slurm (option de srun)

- Un placement spécifique des processus sur les cœurs peut optimiser la performance d'exécution dans le cas où les nœuds de calcul sont partiellement dépeuplés
- Exemple 1
  - Pour une application qui met en jeu une bande passante mémoire importante, l'utilisation d'un cœur sur deux, par puce, augmente la bande passante mémoire disponible par processus. Plusieurs dizaines de pourcentage de gain de rapidité peuvent être obtenus.
  - La partition «2tcourt» faisant partie de celles qui sont déclarées sur les nœuds de calcul à 28 cœurs (2 puces Broadwell de 14 cœurs chacune), le script suivant place et attache les processus de manière optimale si l'utilisateur souhaite utiliser un cœur sur deux.

```
#!/bin/bash
#SBATCH --exclusive
#SBATCH --partition 2tcourt
#SBATCH --nodes 4
#SBATCH --ntasks-per-node 14
# In each compute node : CPU cores 0 - 13 : first chip, CPU cores 14 - 27 : second chip
# => 7 processes in chip 1, 7 processes in chip 2
srun --cpu_bind=map_cpu:0,1,2,3,4,5,6,15,16,17,18,19,20,21 ./a.out
```

Placement des processus (fin)

Fonctions de Slurm (option de srun)

- Exemple 2
  - Une application doit être exécutée avec 128 processus MPI (raisons possibles : algorithme, licence limitant le nombre de processus, réalisation d'un benchmark)
    - L'utilisateur peut souhaiter exécuter le calcul avec un nombre de processus homogène par nœud : 16 processus par nœud (Broadwell) à 28 cœurs

```
#!/bin/bash
#SBATCH --exclusive
#SBATCH --partition 2tcourt
#SBATCH --nodes 8
#SBATCH --ntasks-per-node 16
# In each compute node : CPU cores 0 - 13 : first chip, CPU cores 14 - 27 : second chip
# => 8 processes in chip 1, 8 processes in chip 2
srun --cpu_bind=map_cpu:0,1,2,3,4,5,6,7,15,16,17,18,19,20,21,22 ./a.out
```

Centre Régional Informatique et d'Applications Numériques de Normandie