Utilisation de Béluga pour LAMMPS

Antoine Rincent, 2019

Connexion au superordinateur

Il faut d'abord et avant tout avoir un compte chez ComputeCanada, vers ce lien : https://ccdb.computecanada.ca/

Une fois le compte acquit, il est question de se connecter au superordinateur, pour se faire il est recommandé d'utiliser MobaXterm, en version gratuite et portable (puisque plus performante).

Une fois MobaXterm lancé, pour se connecter au superordinateur, utiliser l'appel suivant :

ssh [username]@beluga.calculcquebec.ca

Il faut ensuite mettre son mot de passe de calcul canada, le logiciel permet de s'en souvenir pour qu'à de futures connexions, en insérant la ligne ssh[…], le mot de passe soit mis automatiquement.

Utilisation de LAMMPS

NB : Les fonctions présentées ici sont pour le superordinateur Béluga, dans le cas où le lecteur cherche à suivre ce guide sur un superordinateur différent, il est possible que les fonctions et leur utilisation varient.

Il faut avoir le module de LAMMPS de disponible, pour se faire, la fonction suivante peut être utilisée afin de connaître la liste de modules installés :

module list

Dans la plupart des cas, il ne sera pas préinstallé, il faudrait alors vérifier ceux disponibles à l'aide de :

module avail

Par la suite, il est possible d'installer le module désiré, avec :

Module add lammps-[…]

NB : Il est possible d'utiliser load plutôt que add. De plus, il est possible d'appuyer sur tab suite à avoir écrit « lammps », pour autocomplete le nom du module désiré.

```
[anrina@beluga3 ~]$ module list

Currently Loaded Modules:
  1) nixpkgs/16.09   (S)      3) gcccore/.7.3.0  (H)   5) ifort/.2018.3.222 (H)   7) openmpi/3.1.2 (m)
  2) imkl/2018.3.222 (math)   4) icc/.2018.3.222 (H)   6) intel/2018.3      (t)   8) StdEnv/2018.3 (S)

  Where:
   S:     Module is Sticky, requires --force to unload or purge
   m:     MPI implementations / Implémentations MPI
   math:  Mathematical libraries / Bibliothèques mathématiques
   t:     Tools for development / Outils de développement
   H:               Hidden Module



[anrina@beluga3 ~]$ module load lammps-user-intel
[anrina@beluga3 ~]$ module list

Currently Loaded Modules:
  1) nixpkgs/16.09     (S)      6) intel/2018.3  (t)   11) eigen/3.3.2  (math)  16) libxc/4.2.3                (chem)
  2) imkl/2018.3.222   (math)   7) openmpi/3.1.2 (m)   12) voro++/0.4.6 (math)  17) quantumespresso/6.3         (chem)
  3) gcccore/.7.3.0    (H)      8) StdEnv/2018.3 (S)   13) vtk/6.3.0    (vis)   18) tbb/2018_U5                 (t)
  4) icc/.2018.3.222   (H)      9) python/2.7.14 (t)   14) kim/1.9.7    (chem)  19) lammps-user-intel/20180822 (chem)
  5) ifort/.2018.3.222 (H)     10) hdf5/1.10.3   (io)  15) netcdf/4.6.1 (io)

  Where:
   S:     Module is Sticky, requires --force to unload or purge
   m:     MPI implementations / Implémentations MPI
   math:  Mathematical libraries / Bibliothèques mathématiques
   io:    Input/output software / Logiciel d'écriture/lecture
   t:     Tools for development / Outils de développement
   vis:   Visualisation software / Logiciels de visualisation
   chem:  Chemistry libraries/apps / Logiciels de chimie
   H:               Hidden Module
```

Il faut ensuite déterminer le nom des exécutables LAMMPS à utiliser dans le futur. Pour se faire il est possible d'utiliser la fonction suivante :

module show [module]

```
[anrina@beluga3 ~]$ module show lammps-user-intel
-------------------------------------------------------------------------------------------------------------------------
   /cvmfs/soft.computecanada.ca/easybuild/modules/2017/avx512/MPI/intel2018.3/openmpi3.1/lammps-user-intel/20180822.lua:
-------------------------------------------------------------------------------------------------------------------------
help([[
Description
===========
LAMMPS (Large-scale Atomic/Molecular Massively Parallel Simulator) is
 a classical molecular dynamics simulation. LAMMPS has potentials for solid-state materials
 (metals, semiconductors) and soft matter (biomolecules, polymers) and coarse-grained or
 mesoscopic systems. It can be used to model atoms or, more generically, as a parallel
 particle simulator at the atomic, meso, or continuum scale. It can be coupled to various
 programs. The following packages are not included within this version:
 -GPU, -KOKKOS, -MANYBODY, -MSCG, -QEQ, -USER-ATC, -USER-QUIP


More information
================
 - Homepage: http://lammps.sandia.gov/
]])
whatis("Description:  LAMMPS (Large-scale Atomic/Molecular Massively Parallel Simulator) is
 a classical molecular dynamics simulation. LAMMPS has potentials for solid-state materials
 (metals, semiconductors) and soft matter (biomolecules, polymers) and coarse-grained or
 mesoscopic systems. It can be used to model atoms or, more generically, as a parallel
 particle simulator at the atomic, meso, or continuum scale. It can be coupled to various
 programs. The following packages are not included within this version:
 -GPU, -KOKKOS, -MANYBODY, -MSCG, -QEQ, -USER-ATC, -USER-QUIP")
whatis("Homepage: http://lammps.sandia.gov/")
conflict("lammps-user-intel")
depends_on("imkl/2018.3.222")
depends_on("python/2.7.14")
depends_on("hdf5/1.10.3")
depends_on("eigen/3.3.2")
depends_on("voro++/0.4.6")
depends_on("vtk/6.3.0")
depends_on("kim/1.9.7")
depends_on("netcdf/4.6.1")
depends_on("quantumespresso/6.3")
depends_on("tbb/2018_U5")
prepend_path("LIBRARY_PATH","/cvmfs/soft.computecanada.ca/easybuild/software/2017/avx512/MPI/intel2018.3/openmpi3.1/lammps-user
-intel/20180822/lib")
prepend_path("PATH","/cvmfs/soft.computecanada.ca/easybuild/software/2017/avx512/MPI/intel2018.3/openmpi3.1/lammps-user-intel/2
0180822/bin")
setenv("EBROOTLAMMPS","/cvmfs/soft.computecanada.ca/easybuild/software/2017/avx512/MPI/intel2018.3/openmpi3.1/lammps-user-intel
/20180822")
setenv("EBVERSIONLAMMPS","20180822")
setenv("EBDEVELLAMMPS","/cvmfs/soft.computecanada.ca/easybuild/software/2017/avx512/MPI/intel2018.3/openmpi3.1/lammps-user-inte
l/20180822/easybuild/avx512-MPI-intel2018.3-openmpi3.1-lammps-user-intel-20180822-easybuild-devel")
family("lammps")
```

Par la suite, il est possible de retrouver les exécutables (surlignés en vert) dans le répertoire présenté par « PATH » dans l'appel de module show :

```
[anrina@beluga3 ~]$ cd ..
[anrina@beluga3 home]$ cd ..
[anrina@beluga3 /]$ cd ..
[anrina@beluga3 /]$ cd cvmfs
[anrina@beluga3 cvmfs]$ cd soft.computecanada.ca/
[anrina@beluga3 soft.computecanada.ca]$ cd easybuild/
[anrina@beluga3 easybuild]$ cd software/
[anrina@beluga3 software]$ cd 2017
[anrina@beluga3 2017]$ cd avx512/
[anrina@beluga3 avx512]$ cd MPI
[anrina@beluga3 MPI]$ cd intel2018.3/
[anrina@beluga3 intel2018.3]$ cd openmpi3.1/
[anrina@beluga3 openmpi3.1]$ cd lammps-user-intel/
[anrina@beluga3 lammps-user-intel]$ cd 20180822/
[anrina@beluga3 20180822]$ ls
bench  bin  cmake  doc  easybuild  examples  lib  LICENSE  list-packages.txt  potentials  python  README  src  tools
[anrina@beluga3 20180822]$ cd bin
[anrina@beluga3 bin]$ ls
lmp  lmp_intel_cpu_openmpi
```

Il est maintenant possible d'appeler l'exécutable déterminé ainsi par le biais d'un fichier de soumission de tâche. Si des packages nécessaires aux calculs sont manquants (voir capture plus haute avec flèche rouge), il faudra compiler manuellement l'exécutable LAMMPS. Voir plus bas pour un guide à cet effet.

Compilation manuelle de l’exécutable LAMMPS

Il faut d’abord et avant tout, [télécharger un tarball de LAMMPS](), je recommande la dernière version stable pour des raisons de simplicité, dans ce cas-ci, 12Dec2018 :

Download a tarball

Select the code you want, click the "**Download Now**" button, and your browser should download a gzipped tar file. Unpack it with the following command, and see the README file to get started.

```
tar -xzvf file.tar.gz
```

There have been ~316,000 downloads of LAMMPS from Sept 2004 thru Dec 2018.

LAMMPS molecular dynamics package:

- (•) LAMMPS --- Stable version (12 Dec 2018) - Recent C++ version source + doc tarball, GPL license, ~150 Mb. Includes all bug fixes and new features described on this page, up to the date of the most recent stable release.
- ( ) LAMMPS --- Development version - Most current C++ version source + doc tarball, GPL license, ~150 Mb. Includes all bug fixes and new features described on this page.
- ( ) LAMMPS --- Documentation - Most current C++ version doc tarball, GPL license, ~21 Mb.
- ( ) LAMMPS 2001 --- older f90 version source tarball, GPL license, 1.1 Mb, last updated 17 Jan 2005
- ( ) LAMMPS 99 --- older f77 version source tarball, GPL license, 840 Kb

( ) No package

[Download Now]

Il faut ensuite l’uploader sur Béluga, ce qui se fait bien à partir de MobaXterm à l’aide d’un clic and drag vers le répertoire désiré. Une fois l’upload terminé, il est possible d’extraire les fichiers du tarball à l’aide de la fonction suivante :

tar -xvf [nom du répertoire].tar.gz

```
[2019-05-08 08:48.08]  ~
[arl33.Surface] ➤ ssh anrina@beluga.calculquebec.ca
Warning: Permanently added 'beluga.calculquebec.ca' (RSA) to the list of known hosts.
Last login: Mon May  6 21:10:27 2019 from 206.167.243.2
################################################################################

                                  Bienvenue sur Béluga / Welcome to Béluga

                                  Aide/Support:     support@calculcanada.ca
                                  Globus endpoint:  computecanada#beluga-dtn
                                  Documentation:    docs.calculcanada.ca

################################################################################

2019-04-18 - Étant donné les transferts massifs de données vers Béluga, le
  système de sauvegarde n'est pas nécessairement à jour à tous les jours.
2019-04-18 - Because of massive data transfers to Béluga, the backup system is
  not necessarily up to date everyday.
[anrina@beluga4 ~]$ ls
LAMMPS_Antoine  log.lammps  nearline  projects  scratch  subjob
[anrina@beluga4 ~]$ cd LAMMPS_Antoine
[anrina@beluga4 LAMMPS_Antoine]$ ls
Al99.eam.alloy  Al_Cp_3.log  Al_fcc_3.in  lammps  log.lammps  subjob.pbs
[anrina@beluga4 LAMMPS_Antoine]$ cd lammps
[anrina@beluga4 lammps]$ ls
bench  cmake  doc  examples  lammps-stable.tar.gz  lib  LICENSE  potentials  python  README  src  tools
[anrina@beluga4 lammps]$ tar -xvf lammps-stable.tar.gz
lammps-12Dec18/
lammps-12Dec18/LICENSE
lammps-12Dec18/README
lammps-12Dec18/bench/
lammps-12Dec18/bench/log.6Oct16.chain.fixed.icc.1
lammps-12Dec18/bench/log.6Oct16.chute.scaled.icc.4
lammps-12Dec18/bench/log.6Oct16.eam.scaled.icc.4
lammps-12Dec18/bench/in.eam
lammps-12Dec18/bench/in.chain.scaled
lammps-12Dec18/bench/in.rhodo
lammps-12Dec18/bench/FERMI/
lammps-12Dec18/bench/FERMI/in.eam
lammps-12Dec18/bench/FERMI/in.rhodo
lammps-12Dec18/bench/FERMI/in.lj.titan
lammps-12Dec18/bench/FERMI/in.lj
lammps-12Dec18/bench/FERMI/in.eam.titan
lammps-12Dec18/bench/FERMI/in.rhodo.scaled.titan
lammps-12Dec18/bench/FERMI/README
lammps-12Dec18/bench/FERMI/in.rhodo.split.titan
```

```
lammps-12Dec18/tools/eam_generate/
lammps-12Dec18/tools/eam_generate/Al_Zhou.c
lammps-12Dec18/tools/eam_generate/Cu_Mishin1.c
lammps-12Dec18/tools/eam_generate/W_Zhou.c
lammps-12Dec18/tools/eam_generate/Cu_Zhou.c
lammps-12Dec18/tools/eam_generate/README
lammps-12Dec18/tools/ipp/
lammps-12Dec18/tools/ipp/template_lammps.input
lammps-12Dec18/tools/ipp/template_createAtoms.input
lammps-12Dec18/tools/ipp/fill_parameters_lammps.input
lammps-12Dec18/tools/ipp/README
lammps-12Dec18/tools/ipp/fill_parameters_createAtoms.input
lammps-12Dec18/tools/ipp/ipp
lammps-12Dec18/tools/kate/
lammps-12Dec18/tools/kate/README.txt
lammps-12Dec18/tools/kate/lammps.xml
lammps-12Dec18/tools/doxygen/
lammps-12Dec18/tools/doxygen/Developer.dox.lammps
lammps-12Dec18/tools/doxygen/Doxyfile.lammps
lammps-12Dec18/tools/doxygen/README
lammps-12Dec18/tools/doxygen/doxygen.sh
lammps-12Dec18/tools/emacs/
lammps-12Dec18/tools/emacs/README.md
lammps-12Dec18/tools/emacs/lammps-mode.el
[anrina@beluga4 lammps]$
```

Ceci prendra un certain temps, mais une fois l'extraction terminée il sera possible de compiler l'exécutable, pour se faire, se rendre dans le répertoire ainsi extrait, puis dans le répertoire du code source s'y retrouvant (src) :

```
[anrina@beluga1 lammps]$ cd 12Dec18/
[anrina@beluga1 12Dec18]$ ls
bench  cmake  doc  examples  lib  LICENSE  potentials  python  README  src  tools
[anrina@beluga1 12Dec18]$ cd src
[anrina@beluga1 src]$
```

Il faut ensuite faire l'installation des packages désirés, dans mon cas, les packages suivant seront installés : BODY, MANYBODY, MISC, OPT, USER-INTEL, USER-MEAMC, USER-OMP, il est possible de retrouver la liste des packages ainsi que leur utilité dans la documentation de LAMMPS. Il n'y a pas vraiment de désavantages à installer plus de packages que nécessaire, cependant ceci pourrait encourir des complexités causées par l'installation de ces derniers, et augmenter le temps nécessaire à la compilation de l'exécutable désiré. Il est possible que ces installations de package prennent un certain temps. Pour se faire, il faut utiliser l'une des fonctions suivantes :

make yes-[Package voulu] yes-[Package voulu]

make yes-all no-[Package indésirable] no-[Package indésirable]

```
[anrina@beluga1 src]$ make yes-BODY yes-MANYBODY yes-MISC yes-OPT yes-USER-INTEL yes-USER-MEAMC yes-USER-OMP
Installing package BODY
Installing package MANYBODY
Installing package MISC
Installing package OPT
Installing package USER-INTEL
Installing package USER-MEAMC
Installing package USER-OMP
[anrina@beluga1 src]$
```

Il faut ensuite lancer la compilation de l'exécutable à l'aide du compilateur de notre choix, ceux-ci se retrouvent dans le sous-répertoire /MAKE/OPTIONS/, en raison de l'utilisation voulue sur des processeurs Intel avec OpenMP, le compilateur intel_cpu_openmpi est utilisé. Accessoirement, la compilation avec les autres compilateurs a été testée, mais ils ont tous eu un problème, confirmant ce choix d'une manière assez informelle. Le nom du compilateur se retrouve dans /OPTIONS/, sans le « Makefile. ». Pour exécuter la compilation, il faut utiliser cet appel de make :

make [nom du compilateur]

```
[anrina@beluga1 src]$ make intel_cpu_openmpi
make[1]: Entering directory '/home/anrina/LAMMPS_Antoine/lammps/12Dec18/src'
Gathering installed package information (may take a little while)
make[1]: Leaving directory '/home/anrina/LAMMPS_Antoine/lammps/12Dec18/src'
Compiling LAMMPS for machine intel_cpu_openmpi
make[1]: Entering directory '/home/anrina/LAMMPS_Antoine/lammps/12Dec18/src/Obj_intel_cpu_openmpi'
cc -O -o fastdep.exe ../DEPEND/fastdep.c
```

```
d_chunk.o pair_yukawa.o compute_temp_ramp.o npair_half_nsq_newtoff.o improper_deprecated.o npair_half_multi_newton.o compute_temp
_partial.o atom_vec_atomic.o npair_half_nsq_newton.o nstencil_full_bin_3d.o npair_half_bin_newtoff_ghost_omp.o compute_cluster_at
om.o dihedral_hybrid.o atom_vec_ellipsoid.o npair_half_bin_atomonly_newton.o dump_atom.o compute_orientorder_atom.o pair_adp.o re
gion_block.o ntopo_bond_template.o npair_half_respa_nsq_newton_omp.o dump_cfg.o      -ltbbmalloc -lmkl_intel_ilp64 -lmkl_sequenti
al -lmkl_core -o ../lmp_intel_cpu_openmpi
size ../lmp_intel_cpu_openmpi
   text    data     bss     dec     hex filename
11194191         437216   17320 11648727         b1bed7 ../lmp_intel_cpu_openmpi
make[1]: Leaving directory '/home/anrina/LAMMPS_Antoine/lammps/12Dec18/src/Obj_intel_cpu_openmpi'
[anrina@beluga1 src]$
```

Cette étape prendra relativement longtemps (±20 minutes), suite à quoi il sera possible d'utiliser l'exécutable ainsi créé. Par défaut, son nom est lmp_intel_cpu_openmpi, afin de le vérifier il est possible de faire la fonction ls et de chercher l'exécutable (à l'aide de la couleur verte), ceci peut prendre un moment en raison du nombre important de fichiers s'y . Dans l'exemple du fichier d'appel plus bas, le chemin est le même qu'ici, mais au lieu d'être dans le dossier /12Dec18/, il est dans le dossier /1/.

```
[anrina@beluga1 src]$ ls
```

```
compute_erotate_sphere_atom.h    LATTE                          pair_gauss_omp.h
compute_erotate_sphere.cpp       lattice.cpp                    pair_gw.cpp
compute_erotate_sphere.h         lattice.h                      pair_gw.h
compute_fragment_atom.cpp        library.cpp                    pair_gw_zbl.cpp
compute_fragment_atom.h          library.h                      pair_gw_zbl.h
compute_global_atom.cpp          lmpinstalledpkgs.h             pair.h
compute_global_atom.h            lmp_intel_cpu_openmpi          pair_hybrid.cpp
compute_group_group.cpp          lmptype.h                      pair_hybrid.h
compute_group_group.h            lmpwindows.h                   pair_hybrid_overlay.cpp
compute_gyration_chunk.cpp       main.cpp                       pair_hybrid_overlay.h
compute_gyration_chunk.h         MAKE                           pair_lcbop.cpp
compute_gyration.cpp             Makefile                       pair_lcbop.h
compute_gyration.h               Makefile.list                  pair_lj96_cut.cpp
```

Il est possible d'obtenir de l'aide sur l'exécutable créé et en vérifier les packages installés à l'aide de l'appel suivant :

./[nom de l'exécutable] -help

```
[anrina@beluga1 src]$ ./lmp_intel_cpu_openmpi -help
```

```
Large-scale Atomic/Molecular Massively Parallel Simulator - 12 Dec 2018

Usage example: ./lmp_intel_cpu_openmpi -var t 300 -echo screen -in in.alloy

List of command line options supported by this LAMMPS executable:

-echo none/screen/log/both  : echoing of input script (-e)
-help                       : print this help message (-h)
-in filename                : read input from file, not stdin (-i)
-kokkos on/off ...          : turn KOKKOS mode on or off (-k)
-log none/filename          : where to send log output (-l)
-mpicolor color             : which exe in a multi-exe mpirun cmd (-m)
-nocite                     : disable writing log.cite file (-nc)
-package style ...          : invoke package command (-pk)
-partition size1 size2 ...  : assign partition sizes (-p)
-plog basename              : basename for partition logs (-pl)
-pscreen basename           : basename for partition screens (-ps)
-restart2data rfile dfile ... : convert restart to data file (-r2data)
-restart2dump rfile dgroup dstyle dfile ...
                            : convert restart to dump file (-r2dump)
-reorder topology-specs     : processor reordering (-r)
-screen none/filename       : where to send screen output (-sc)
-suffix gpu/intel/opt/omp   : style suffix to apply (-sf)
-var varname value          : set index style variable (-v)

Active compile time flags:

-DLAMMPS_GZIP

Installed packages:

BODY MANYBODY MISC OPT USER-INTEL USER-MEAMC USER-OMP
```

Pour en sortir, il suffit de taper la touche q du clavier (il est possible de naviguer le document avec la molette de la souris).

NB : en compilant manuellement l'exécutable, il n'est pas nécessaire de charger des modules tel que décrit plus haut, puisque l'exécutable remplace le module désiré, et les modules complémentaires nécessaires sont chargés et mis en place lors de la compilation de l'exécutable.

Fichier de soumission de tâche

Voici un exemple de fichier de soumission de tâche pour l'appel de LAMMPS, ce dernier est expliqué ligne par ligne plus bas. Celui-ci doit correspondre aux standards Slurm voici un bon guide. L'extension de ce fichier n'est pas importante, dans cet exemple il a été nommé newjob.input.

```
#!/bin/bash
#
#SBATCH --job-name=80_250
#SBATCH --output=80_250.slurmoutput
#
#SBATCH --ntasks=80
#SBATCH --nodes=4-8
#SBATCH --cpus-per-task=1
#SBATCH --time=23:50:00
#SBATCH --mem-per-cpu=2048
#
#SBATCH --mail-type=ALL
#SBATCH --mail-user=antoine.rincent@gmail.com

module load intel/2018.3 openmpi/3.1.2 lammps-special-intel/20180822

export OMP_NUM_THREADS=1

srun --mpi=openmpi lmp_intel_cpu_openmpi < Al_80_250.in > Al_80_250.log
srun --mpi=openmpi lmp_intel_cpu_openmpi < Al_80_260.in > Al_80_260.log
```

1. Sert d'en-tête au fichier, à recopier sans différences.
3. –job-name donne le nom (dans ce cas-ci « 80_250») au projet, ceci est facultatif dans le cas de Béluga, mais ce n'est pas forcément le cas en fonction du superordinateur utilisé.
4. --output sert à déterminer le nom du fichier d'output désiré. Ce dernier permet d'enregistrer les détails d'un problème si jamais il y en a un qui devait survenir. Si le programme s'exécute sans soucis, le fichier créé sera vide.
6. --ntasks détermine le nombre de tâches dédiées pour exécuter les calculs.
7. --nodes indique le nombre de nœuds désirés pour exécuter les calculs. Si un seul chiffre est indiqué, c'est le nombre de nœuds qui seront posés, dans un appel comme plus haut, il est demandé entre 4 et 8 nœuds, la raison derrière ce choix est expliquée plus bas dans la section sur l'appel optimal.
8. --cpus-per-task détermine le nombre de processeurs à utiliser par tâche.
9. --time détermine le temps d'exécution attendu du script. Si le script n'est pas terminé une fois ce délai écoulé, il sera arrêté de force. Ce délai doit être d'au moins 5 minutes (00 :05 :00) pour les tests et d'au moins 1 heure (01 :00 :00) pour les calculs, en durant moins de 7 jours au maximum. Il est recommandé pour maximiser la priorité d'exécution de mettre un temps très près mais inférieur à 24h, puisqu'entre 1h et 24h, le scheduler ne fait pas de différence.
10. --mem-per-cpu détermine la mémoire allouée à chaque processeur, en Mo, dans ce cas-ci, 2048Mo, équivalent à 2Gb seront alloués à chacun des 80 processeurs.
12. --mail-type sert à déterminer quelles informations envoyer par courriel, avec ALL, un courriel sera envoyé lorsque la tâche sera entamée, finie ou arrêtée pour quelque raison que ce soit, y compris un bug.
13. --mail-user détermine à quelle adresse courriel les informations de --mail-type seront envoyées.

15. Sers à appeler le module utilisé pour cet exemple.
17. Cette ligne permet de remplacer la variable d'environnement du nombre de processeurs fournis à LAMMPS automatiquement à la même valeur que celle demandée dans la ligne 7
19. Cette ligne représente la version Béluga de l'appel de LAMMPS dans la console Windows, celle-ci suit la logique suivante :
    srun --mpi=[type d'MPI à utiliser] [exécutable LAMMPS sans l'extension] < [fichier d'input avec l'extension] > [fichier d'output avec l'extension]
    Il est important que le nom du fichier d'output soit le même que dans l'appel de LAMMPS, dans cet exemple-ci, le fichier input de LAMMPS possède la ligne suivante : log An_80_250.log, on note qu'ici l'option append n'aura aucun impact, qu'elle soit présente ou non, si un fichier du même nom est présent dans le répertoire de travail, il sera effacé et remplacé par le nouveau.
    NB : Il est possible de spécifier le filepath vers l'exécutable compilé manuellement en mettant comme exécutable, par exemple :
    /home/anrina/LAMMPS_Antoine/lammps/1/src/lmp_cpu_intel_openmpi
20. Comme la précédente, cette ligne lancera LAMMPS avec un nouveau fichier d'input et un fichier d'output différent. Il est à noter que pour se faire, ce dernier sera lancé suite à l'exécution de la ligne 19. S'il est plutôt désiré que plusieurs simulations LAMMPS aient lieu en même temps afin d'accélérer le processus, il faudra écrire un fichier de soumission pour chaque simulation LAMMPS, et les lancer indépendamment les unes des autres, manuellement.

Pour appeler le script de soumission, il faut ensuite faire l'appel suivant à partir du répertoire où il se retrouve et où se retrouvent les fichiers pour LAMMPS, où le fichier d'output sera créé :

sbatch [fichier d'appel avec son extension]

S'il a été créé dans un environnement Windows, il est important d'utiliser depuis Béluga la commande suivante afin d'enlever les sauts de ligne de Windows pour que le fichier soit adéquatement lu :
dos2unix [fichier d'appel avec son extension]

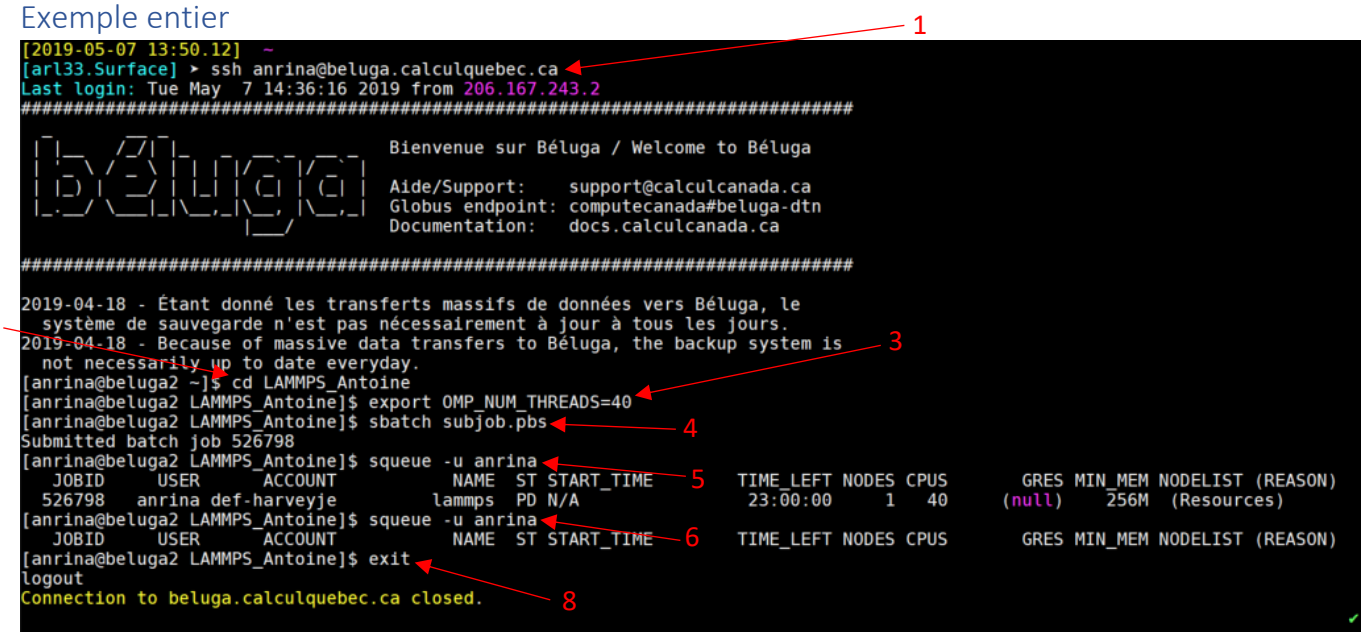
Voir plus bas pour l'appel optimal et l'envoi de travaux à Béluga pour minimiser le temps d'exécution.

NB : Il est capital de ne mettre à aucun endroit des espaces, notez que le titre et noms dans l'exemple plus haut comportent des ____ plutôt que des espaces. Autrement les options mises ne seront pas lues par le scheduler, et des valeurs par défaut seront assignées.

Dans le cas de LAMMPS, il n'est pas nécessaire d'allouer plus d'un processeur par tâche, puisque ce sont des ressources alors inutilisées. Si c'était cependant ce qui était voulu, il faudrait, dans le fichier d'input de LAMMPS, mettre comme option à la ligne package omp [nombre] le même nombre qu'à --cpus-per-task, de même pour OMP_NUM_THREADS, dans ce cas-ci, il y a package omp 1 dans le fichier d'input LAMMPS et --cpus-per-task=1 ainsi que OMP_NUM_THREADS=1 dans le fichier de soumission, autrement il serait possible de faire package omp 5 dans le fichier d'input LAMMPS et --cpus-per-task=5 et OMP_NUM_THREADS=5 dans le fichier de soumission.

Pour plus de documentation sur les fichiers de soumission, ainsi que certains cas limites (par exemple (--ntasks * --cpus-per-task) < (--nodes)), se référer à la documentation des fichiers de soumission SLURM.

Exemple entier



En ordre des étapes ayant eu lieu

1. Connexion à Béluga au travers du compte Calcul Canada
2. Ouverture du répertoire dans lequel se retrouvent le fichier d'envoi et les fichiers de LAMMPS
3. Changement de la variable d'environnement OMP_NUM_THREADS au nombre de cœurs utilisés, dans ce cas-ci, 40
4. Envoi du travail à effectuer via Béluga à l'aide de sbatch
5. Utilisation de squeue -u [utilisateur] pour vérifier que la tâche à effectuer est bien envoyée au superordinateur
    a. On note qu'il y a un JOBID, associé à mon nom d'utilisateur (anrina), le START_TIME est N/A et TIME_LEFT est égal au temps fourni dans subjob.pbs, car la tâche ne s'est pas encore entamée.
6. Réutilisation de squeue -u [utilisateur] pour confirmer que la tâche à effectuer est terminée (après un certain temps d'attente)
    a. On note qu'il n'y a aucun JOBID qui est imprimé, indiquant que je n'ai aucune tâche en attente ou en cours, donc que la tâche s'est accomplie.
7. Sauvegarde des fichiers de sortie à l'aide de l'explorateur de fichiers présents dans l'interface de MobaXterm (pas visible dans cette capture d'écran)
8. Fermeture de la session avec la commande « exit »

Exécution optimale du script

Afin de tester l'exécution optimale du script LAMMPS dans Béluga, j'ai testé toutes les combinaisons possibles d'options lors de l'appel, afin de comparer les performances résultantes. Le « plan d'expériences » se retrouve plus bas, en bleu les tests utilisent OpenMP, en vert les tests utilisent simplement MPI sans OpenMP.

Plan d'expériences

Test #	Suffix OMP	OMP_NUM_THREADS	Package OMP __	--mpi _	--ntasks	--cpus-per-task	--mem-per-cpu	Error si applicable
1	Avec	1	1	openmpi	1	1	1024	Fini
2	Avec	1	1	openmpi	1	20	1024	Fini
3	Avec	1	1	openmpi	20	1	1024	Fini
4	Avec	1	1	openmpi	20	20	1024	Fini
5	Avec	1	20	openmpi	1	1	1024	Fini
6	Avec	1	20	openmpi	1	20	1024	Fini
7	Avec	1	20	openmpi	20	1	1024	Fini
8	Avec	1	20	openmpi	20	20	1024	Fini
9	Avec	20	1	openmpi	1	1	1024	Fini
10	Avec	20	1	openmpi	1	20	1024	Fini
11	Avec	20	1	openmpi	20	1	1024	Fini
12	Avec	20	1	openmpi	20	20	1024	Fini
13	Avec	20	20	openmpi	1	1	1024	Fini
14	Avec	20	20	openmpi	1	20	1024	Fini
15	Avec	20	20	openmpi	20	1	1024	Fini
16	Avec	20	20	openmpi	20	20	1024	Fini
17	Sans	#N/A	#N/A	none	1	1	1024	Fini
18	Sans	#N/A	#N/A	none	1	20	1024	Fini
19	Sans	#N/A	#N/A	none	20	1	1024	Fini
20	Sans	#N/A	#N/A	none	20	20	1024	Fini

Plus bas se retrouvent les résultats de ces tests, roulés sur le module de LAMMPS présent sur Béluga, en comparaison d'un test similaire roulé sur mon ordinateur personnel, avec 12 threads, afin d'être comparable à ceux de Béluga (20 cœurs à 2.4GHz pour Béluga contre 12 cœurs à 3.6GHz ≈ 18 cœurs à 2.4GHz). Il se retrouve aussi les résultats en utilisant l'exécutable compilé manuellement avec Make, dont la démarche est décrite plus haut.

Il est possible de remarquer qu'il vaut au mieux donner --ntasks=20 ainsi qu'un nombre de --cpus-per-task (cpp) au choix, en mettre plus résultera avec une exécution plus rapide, mais moins prioritaire pour le scheduler. Je recommanderais de mettre 1 cpp pour des tâches rapides qui ne requièrent pas trop de mémoire, ou un nombre plus élevé si les tâches sont lourdes en calculs et en mémoire.

On observe aussi que MPI est moins performant qu'OpenMP, il est donc meilleur de mettre --mpi=openmp dans l'appel, avec package OMP 1 dans le fichier LAMMPS, au risque de grandement ralentir l'exécution du script, avec OMP_NUM_THREADS=1, bien que ce dernier n'ait pas un grand impact.

Enfin, on voit que tous les tests donnent le même lattice constant à la fin du script, ce qui démontre la précision de LAMMPS, la seule exception est le test sur ma propre machine, qui roulait sur une version différente de LAMMPS, expliquant la différence de résultat.

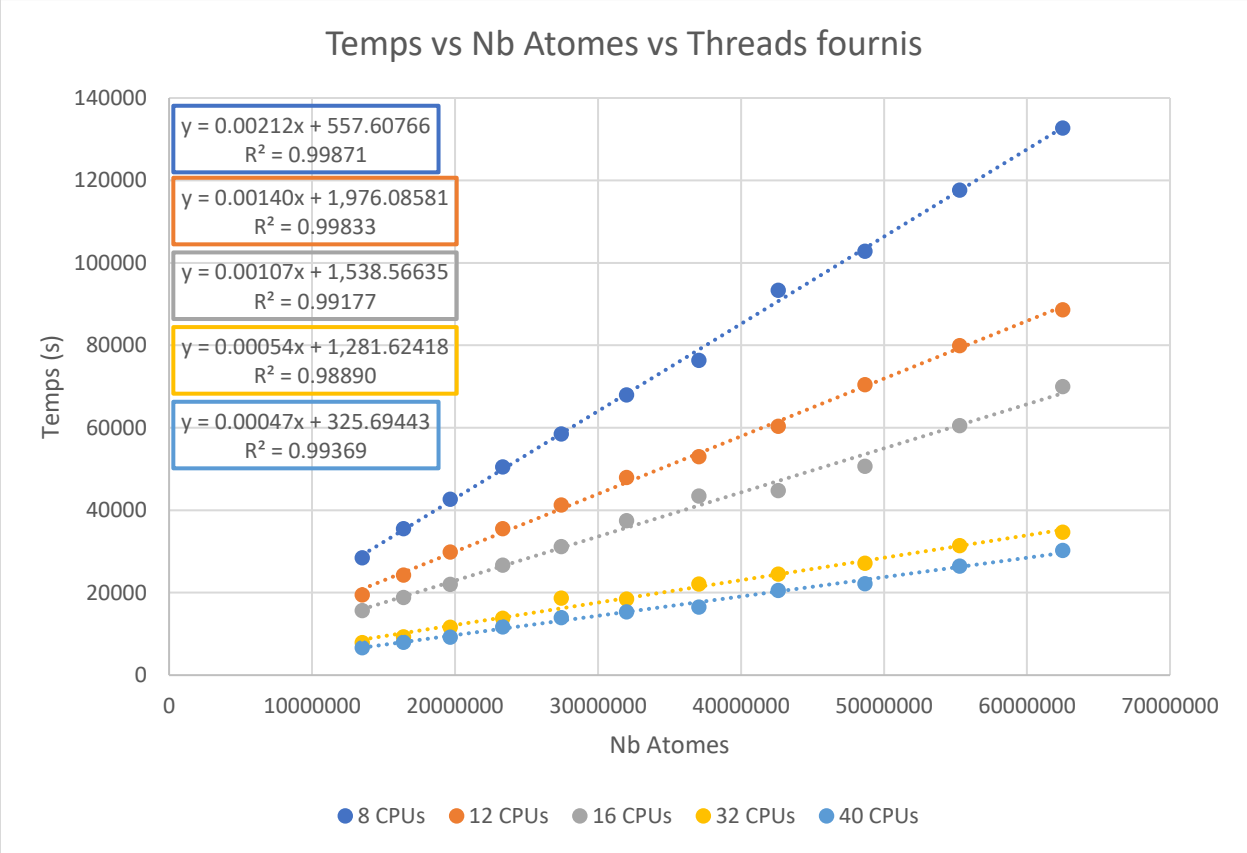
Résultats des tests

Test	Repetition	Iterations	MPI allocation (Mbytes)	CPU Usage	Timesteps/s	hh	mm	ss	Tps Sim (s)	Pair	Neigh	Comm	Output	Modify	Other	Lattice Cste	Commentaires
Tour	40	10000	263.60	996.0%	15.104	0	13	43	823	80.10	0.00	2.49	0.01	15.99	1.40	4.0482789	12 threads
Make	40	10000	12.15	99.7%	51.908	0	4	0	240	87.08	0.00	8.62	0.00	3.65	0.65	4.0489921	Comme test 3 mais compilé manuellement
1	40	10000	139.20	99.7%	3.894	0	52	47	3167	95.62	0.00	0.44	0.00	3.51	0.42	4.0489921	
2	40	10000	139.20	99.8%	3.084	1	6	40	4000	96.53	0.00	0.36	0.00	2.78	0.33	4.0489921	
3	40	10000	12.15	99.6%	51.481	0	4	1	241	86.55	0.00	9.17	0.01	3.74	0.54	4.0489921	exécuté immédiatement
4	40	10000	12.15	99.5%	51.127	0	4	2	242	86.24	0.00	8.20	0.01	5.01	0.55	4.0489921	
5	40	10000	354.70	80.8%	1.621	2	13	11	7991	77.30	0.00	0.24	0.01	22.27	0.18	4.0489921	
6	40	10000	354.70	1985.0%	32.252	0	6	37	397	77.95	0.00	4.44	0.02	14.08	3.51	4.0489921	Appel similaire à tour
7	40	10000	34.49	51.2%	0.317	9	57	58	35878	33.36	0.00	11.77	0.04	45.94	8.88	4.0489921	Effectué sur 2 nœuds
8	40	10000	34.60	1939.8%	146.582	0	3	14	194	41.49	0.00	21.20	0.03	26.88	10.40	4.0489921	
9	40	10000	139.20	99.7%	3.047	1	7	27	4047	96.20	0.00	0.39	0.00	3.07	0.34	4.0489921	
10	40	10000	139.20	99.7%	3.094	1	6	27	3987	96.58	0.00	0.35	0.00	2.74	0.32	4.0489921	
11	40	10000	12.15	99.6%	51.020	0	4	2	242	85.91	0.00	9.78	0.00	3.77	0.54	4.0489921	exécuté immédiatement
12	40	10000	12.15	99.8%	52.682	0	3	56	236	87.54	0.00	8.08	0.01	3.79	0.58	4.0489921	
13	40	10000	354.70	81.3%	1.644	2	10	44	7844	78.58	0.00	0.24	0.00	21.00	0.18	4.0489921	
14	40	10000	354.70	1913.8%	20.590	0	9	45	585	73.50	0.00	3.89	0.03	19.38	3.20	4.0489921	
15	40	10000	34.60	51.1%	0.320	13	32	3	48723	33.31	0.00	11.87	0.04	45.99	8.80	4.0489921	Effectué sur 2 nœuds
16	40	10000	34.60	1921.8%	101.972	0	2	25	145	36.98	0.00	20.26	0.05	28.48	14.22	4.0489921	
17	40	10000	139.10	99.6%	2.237	1	29	42	5382	96.75	0.00	0.32	0.00	2.42	0.51	4.0489921	
18	40	10000	139.10	99.7%	2.303	1	27	22	5242	97.35	0.00	0.24	0.00	1.99	0.42	4.0489921	
19	40	10000	12.15	99.6%	37.861	0	5	19	319	88.27	0.00	8.15	0.00	2.87	0.71	4.0489921	exécuté immédiatement
20	40	10000	12.15	99.8%	40.897	0	4	55	295	90.65	0.00	6.43	0.03	2.39	0.51	4.0489921	

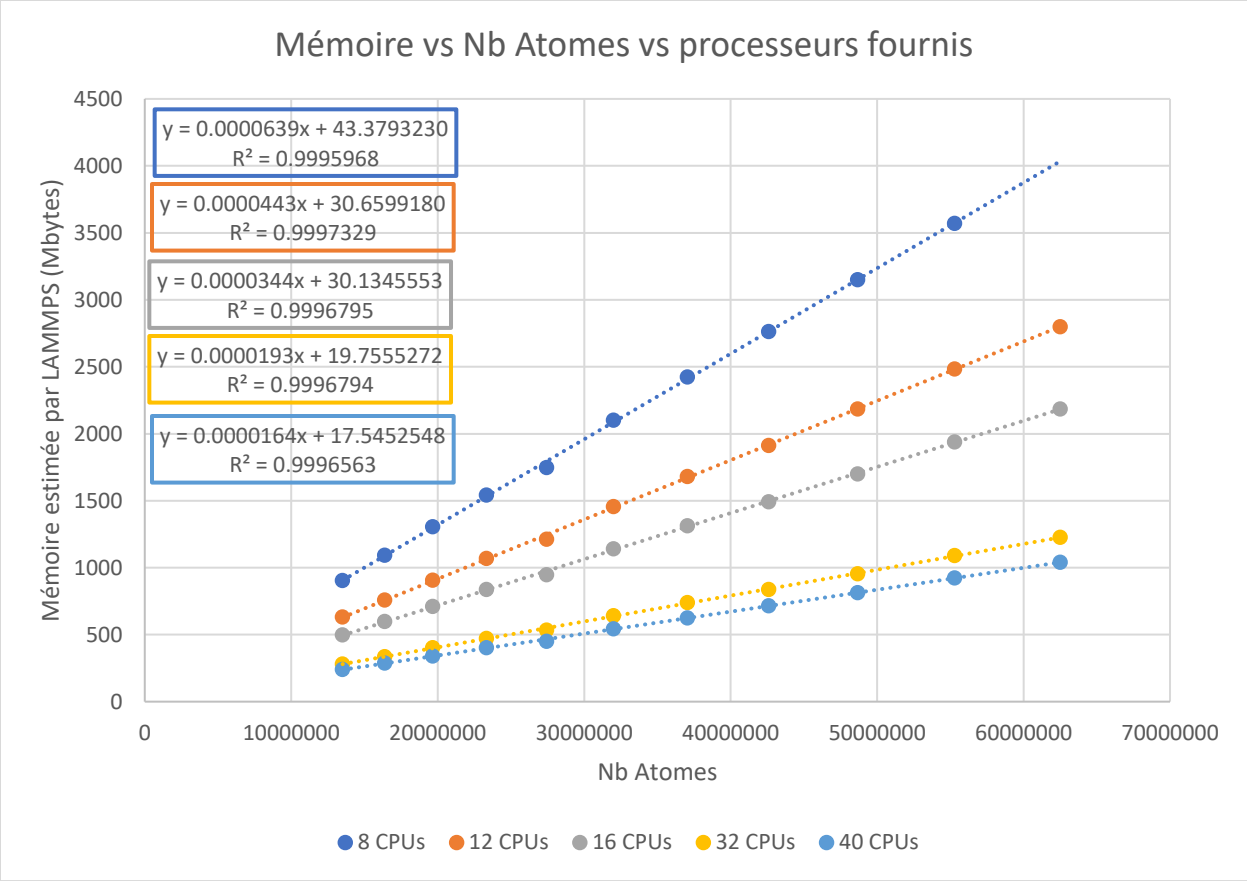
Utilisation optimale des processeurs et de leur répartition sur des nœuds

Afin d'étudier la parallélisation et la synergie de LAMMPS avec Béluga, j'ai effectué le même test avec un nombre croissant d'atomes, sur un nombre croissant de processeurs, pour voir en quoi accroître le nombre de processeurs aide à accélérer les calculs de LAMMPS.

Impact du nombre de processeurs



On remarque que plus le nombre de processeurs augmente, plus les simulations sont courtes, ce qui n'est pas vraiment étonnant pour un logiciel aussi parallélisé que LAMMPS. On remarque un petit plateau vers 40, qui n'est pas le nombre optimal de processeurs pour des simulations (puisque 32 serait moins demandant pour des temps de simulations comparables), cependant en augmentant le nombre de processeurs totaux, on réduit la mémoire nécessaire par processeur pour la simulation, ce qui peut s'avérer très utile pour des simulations de mailles importantes.



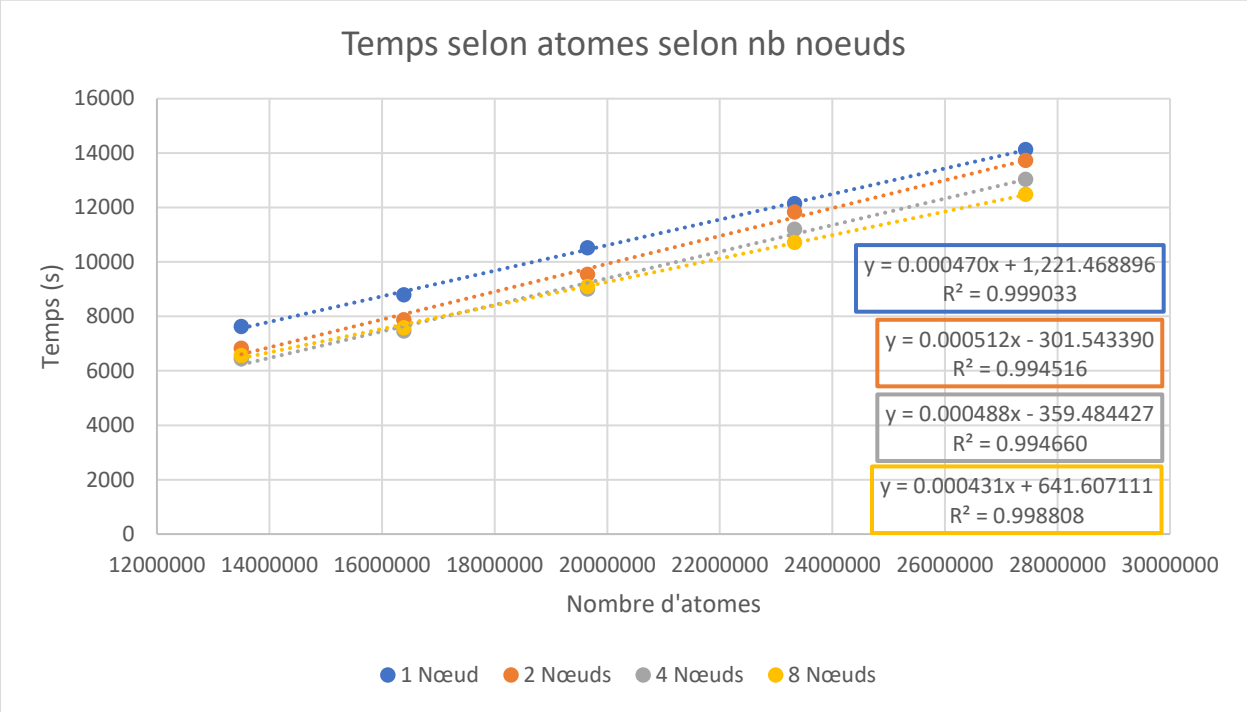
Il est possible de remarquer que l'évolution en fonction du nombre de processeurs alloués évolue de la même manière que dans le cas du temps de simulation. Ceci peut être expliqué par le fait que LAMMPS est très bien parallélisé, et que dans les deux cas, les ressources sont bien réparties parmi les différents processeurs alloués.

Enfin, le nombre de processeurs n'a pas eu d'impact sur les calculs ayant lieu. Ceci n'est pas visible dans le présent document, mais plutôt dans le document Excel compilant les résultats des tests effectués, pour des raisons de concision. Pour un nombre d'atomes donné, la valeur du lattice constant suite aux 10 000 itérations de la rampe NPT était toujours constante, de même pour le nombre d'étapes nécessaire à sa minimisation. Ceci démontre entre l'efficacité de la parallélisation de LAMMPS.

Un test a été effectué avec 80 processeurs, résultant en un temps de simulation de 4h21min, contre 8h24min pour la même avec 40 processeurs, indiquant que le plateau entre 32 et 40 est simplement puisqu'à ce nombre de processeurs, la différence de 8 processeurs n'est pas particulièrement notable à ce nombre. En dépit de la plus grande rapidité d'exécution offerte par une simulation à 80 processeurs, je recommande de se contenter de 40 processeurs à moins que la simulation nécessite une très grande mémoire ou un temps de calcul excessivement long, autrement le temps d'attente pour avoir accès aux 80 processeurs sera supérieur aux gains de temps de calcul encouru par cette augmentation du nombre de processeurs.

Impact du nombre de nœuds

Contrairement au nombre de processeurs, le nombre de nœuds possède un impact surprenant sur les temps de calcul.



La théorie voudrait qu'en ayant tous les processeurs sur un même nœud, les calculs soient plus rapides, car la communication entre les processeurs se ferait plus facilement. Cependant comme on peut le voir sur le graphique plus haut, avec un nombre plus élevé de nœuds, le temps de calcul se réduit, jusqu'à atteindre un plateau entre 4 et 8 nœuds.

À mon avis, il possible d'expliquer ceci par le fait que les nœuds sont connectés entre eux par une connexion infiniband de très grande vitesse, légèrement supérieure à la vitesse au sein d'un même nœud. De plus, selon LAMMPS, le CPU usage à 1 nœud était de 99.1%, contre 99.4% à 2 nœuds et 99.6% à 4 et 8 nœuds, expliquant aussi le gain de performance. Je ne pourrais cependant pas dire la cause de cette différence d'utilisation du processeur.

Pour cette raison, je recommande dans les options du fichier de soumission (voir plus haut dans la section appropriée) de réclamer au minimum 4 nœuds, et un maximum raisonnable (entre 8 et 20 par exemple), afin de minimiser l'équilibre temps de calcul et temps d'attente.

Comme précédemment, le nombre de nœuds n'a eu aucun impact sur les résultats (lattice constant suite aux nombreuses itérations et nombre d'étapes de minimisation nécessaires à la minimisation de cette dernière), indiquant encore une fois que LAMMPS est bien parallélisé, puisque le nombre de nœuds et de processeurs n'a visiblement aucun impact sur les résultats pour une version de LAMMPS et un nombre d'atomes donné.

Il est également bon de noter que le nombre de nœuds n'a aucun impact sur la mémoire nécessaire, chaque simulation demandait une quantité égale de mémoire pour un nombre d'atomes donné.

Fonctions utiles à connaître pour l'interface UNIX

Pour la modification de fichiers .txt

Il est possible de modifier des fichiers .txt avec la fonction vi [fichier.extension], puis en cliquant la touche i du clavier, pour modifier le contenu du fichier, suite à quoi pour sortir du mode édition il faut appuyer sur Esc (échap), pour fermer il faut taper :q! (incluant le :) ou pour sauvegarder et fermer il faut taper :wq (encore un fois avec :).

Commandes en interface par ligne

Si l'on veut tuer une tâche ayant lieu (visible avec squeue -u [nom]), il est possible de noter son JOBID puis utiliser la fonction scancel [JOBID], ce qui y mettra fin.

De même, si une tâche a lieu DANS la console, par exemple lors de l'extraction d'un tarball ou de la compilation d'un exécutable, il est possible de faire ctrl + c (comme pour copier dans Windows), ce qui force l'arrêt de la tâche en cours dans la console.

S'il s'agit de supprimer un répertoire il est possible d'utiliser rm -rf [répertoire] afin de forcer la suppression d'un dossier et la totalité de son contenu.

Pour changer des variables d'environnement en dehors d'un script d'envoi, il faut suivre la même nomenclature, c'est-à-dire export [variable d'environnement]=[valeur] .

Ressources utiles supplémentaires

FAQ de computecanada

Explications sur l'exécution de tâches de computecanada

Wiki de calculquebec

Documentation de Béluga de Calcul Canada