Formation Calmip-ATOS

EOS

Fev 2018

Cyril.Mazauric@atos.net

Your business technologists. Powering progress

Présentation AGENDA

- Présentation du supercalculateur EOS
- Batch scheduler : SLURM
- Les modules
- La compilation
- Quelques outils

Présentation préparation

ssh –X <login>@eos
ou
ssh –X <login>@192.168.12.1

cp /tmpdir/renon/TP_FORMATION_EOS.tar .

tar –xf TP_FORMATION_EOS.tar

cd TP_FORMATION_EOS

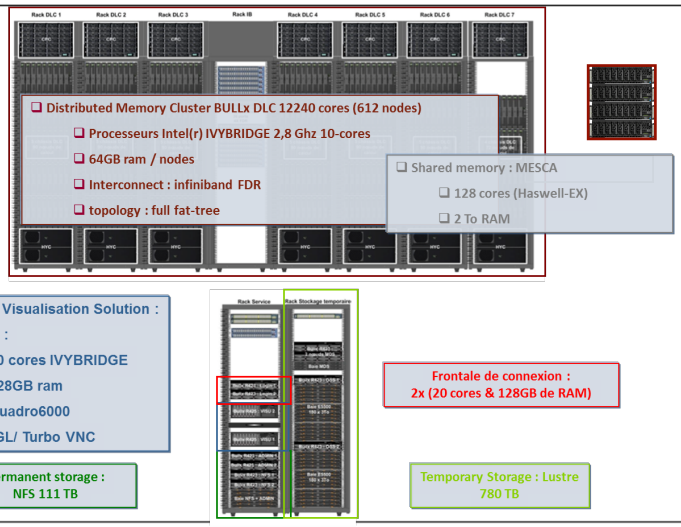
tar –xzf TP_FORMATION.tar.gz

WIFI
pass :

Présentation d'EOS

Fev 2018

EOS (2014 – 2017)



Appel d'Offre brique Stockage : Système de Stockage « ATLAS »

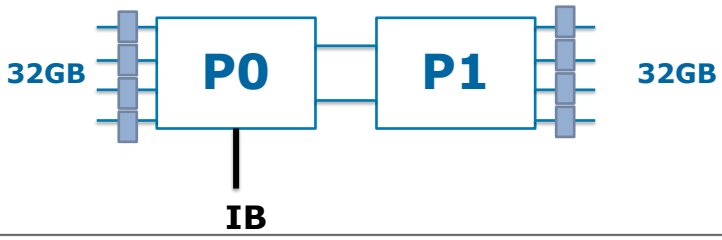
❑ Système de Stockage « ATLAS » : Solution DDN intégrée par ATOS (Bull) (5 années de maintenance incluses)

Espace A
Appliance DDN GPFS
2 contrôleurs GS12k
16 Go/s
Tiroirs : 48*8To NL-SAS+SSD
10 Tiroirs : 3 840 To (bruts)
3 Po utiles (RAID6 8+2)

Espace B
Stockage Objet
Solution DDN : WOS (Web Object Storage)
8 * Tiroirs WOS
Tiroirs : 96*8To SMR
6 144To (bruts)
4 Po utiles (suivant policy)

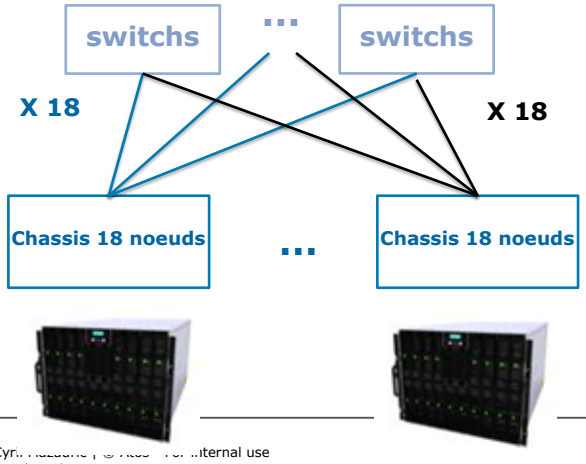
EOS caractéristiques

- 612 nœuds IvyBridge bi-sockets : 2 x 10 cœurs par nœuds => 12240 cœurs de calculs
- Hyperthreading activé : chaque nœud possède donc 2 x 10 x 2 cœurs
  - l'HT est conseillé pour les jobs hybride MPI/OpenMP
- les processeurs IVB sont cadencés à 2,8 GHz (Turbo -> 3,1 GHz)
- 64 GB de mémoire par nœuds (4 x 8 GB à 1866 MHz x 2) soit une bande passante mémoire de 96 GB/s



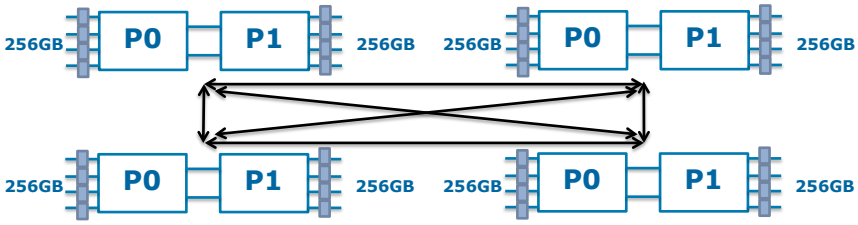
EOS caractéristiques

- Infiniband FDR soit une bande passante pingpong de 5,9 GB/s
- Topologie full fat tree



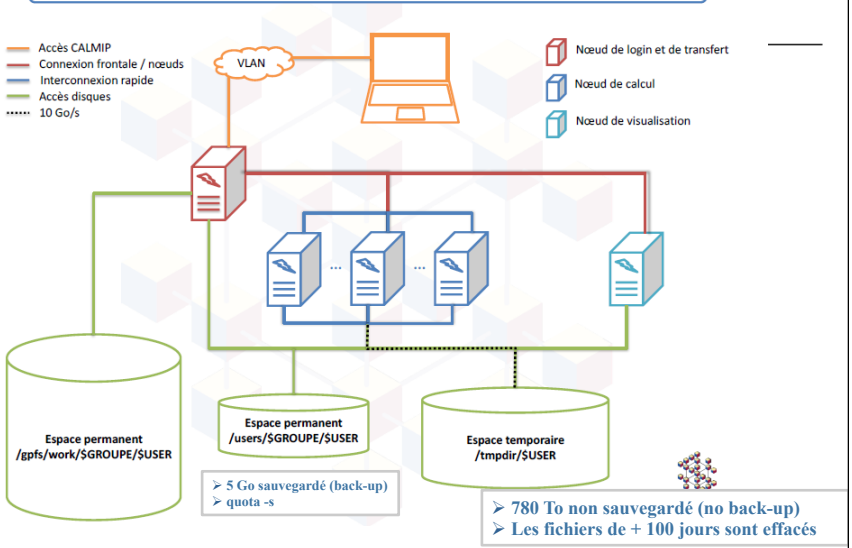
MESCA caractéristiques

- 1 nœud Haswell-EX 8 sockets : 8 x 16 cœurs => 128 cœurs de calculs
- Les processeurs HSW sont cadencés à 2,2 GHz
- 2 TB de mémoire (8 x 256 GB à 1066 MHz)



- Sur EOS, vous ne pourrez utiliser que 64 cœurs maximum et 1024 GB de mémoire maximum.
- Ce nœud sera donc partagé entre plusieurs utilisateurs.

Système EOS : accès aux espaces fichiers

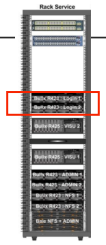


Le batch scheduler : slurm

Fev 2018

SLURM PRINCIPE I

Principe de connexion

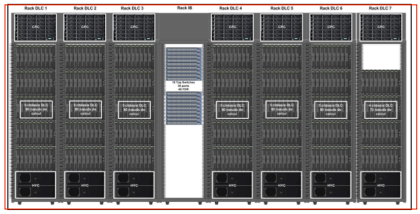


- Connexion sur la frontale

  >> ssh -X {username}@eos.calmip.univ-toulouse.fr

- Lancement des calculs => Mode différé avec le gestionnaire de batch SLURM

>> sbatch mon_job



Page web associée :
http://www.calmip.univ-toulouse.fr/spip/spip.php?rubrique96

slurm

principe II

calmip

- SLURM permet de réserver les nœuds dont vous avez besoin mais aussi peut vous aider à placer vos processus de calcul.

- Les partitions d'EOS:
  - Partition exclusive : les nœuds sont alloués de manière exclusive
  - Partition shared : les nœuds sont partagés (utilisation des cpuset) [seul 6 nœuds sont intégrés à la partition shared]
  - Partition mesca : 1 nœud SMP de type mesca. Ce nœud est partagé.

- Routage des jobs:
  - Tout job qui demande plus de 10 tâches est routé vers la partition exclusive, moins de 10 tâches vers la partition shared

- Remarque :
  - Pour réserver un nœud partagé (shared ou mesca) il faut spécifier la quantité de mémoire voulue (option –mem).
  - Pour reserver une partie de la machine mesca, ajouter --qos=mesca

Prise en main d'EOS

Organisation des files d'attente SLURM

QOS	Nombre de cores	Nombre de nœuds	Walltime	Jobs/ user	Ram/ core	Partition	Remarque
mono	< 10	1	400h	3 max	-	shared	Non exclusif - HT
noeud	20	1	300h	3 max	3gb	exclusive	Exclusif - HT
noeud9	40 à 180	2 à 9	200h	2 max	3gb	exclusive	Exclusif - HT
noeud18	200 à 360	10 à 18	150h	2 mas	3gb	exclusive	Exclusif - HT
noeud36	380 à 720	19 à 36	100h	1 max	3gb	exclusive	Exclusif - HT
noeud72	740 à 1440	37 à 72	48h	1max	3gb	exclusive	Exclusif - HT
noeud90	1460 à 1800	73 à 90	36h	1 max	3gb	exclusive	Exclusif - HT
mesca	1 à 64	1	100h	1 max	16gb(1 To max.)	mesca	Non exclusif - HT
visu	20	1	2h	1 max	6gb	visu	Non exclusif - HT

slurm commandes

- **sinfo** : pour visualiser la disponibilité des nœuds d'EOS

```
exclusiv*    up   infinite      1 drain* eoscomp431
exclusiv*    up   infinite    143  alloc eoscomp[5-8,15-16,18-35,108-125,144-
exclusiv*    up   infinite    459   idle eoscomp[0-4,9-14,17,36-107,126-143,1]
```

- **squeue –u bull**: pour visualiser l'ensemble des jobs soumis par le user bull

```
 JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
 84313      exclusiv     bash       bull  R       1:57:40      1       occigen305
```

- **scancel 11111** : pour supprimer un job de la file d'attente slurm.

slurm commandes

- **scontrol show jobid=46529** : pour visualiser l'ensemble des caractéristiques d'un job

```
JobId=46529 Name=MnCo2O4
   UserId=lionel(739) GroupId=p1252(564)
   Priority=10030 Account=p1252 QOS=noeud9
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=0 Restarts=0 BatchFlag=1 ExitCode=0:0
   RunTime=3-02:26:32 TimeLimit=3-08:00:00 TimeMin=N/A
   SubmitTime=2015-01-09T11:18:48 EligibleTime=2015-01-09T11:18:48
   StartTime=2015-01-09T11:58:03 EndTime=2015-01-12T19:58:03
   PreemptTime=None SuspendTime=None SecsPreSuspend=0
   Partition=exclusive AllocNode:Sid=eoslogin1:10851
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=eoscomp[18-22]
   BatchHost=eoscomp18
   NumNodes=5 NumCPUs=100 CPUs/Task=1 ReqS:C:T=*:*:*
   MinCPUsNode=20 MinMemoryNode=0 MinTmpDiskNode=0
   Features=(null) Gres=(null) Reservation=(null)
   Shared=0 Contiguous=0 Licenses=(null) Network=(null)
   Command=/tmpdir/lionel/Mn11Co1O16/Mn11Co1O16_VASP/script_vasp
   WorkDir=/tmpdir/lionel/Mn11Co1O16/Mn11Co1O16_VASP
```

slurm script de soumission

- Pour soumettre un job il faut construire un script de lancement (voir à la racine de vos comptes un exemple de script)

sbatch <arguments> script

- Les arguments du script peuvent être donnés en entête du script (Priorité aux arguments du sbatch)

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks=20
#SBATCH --cpu-per-tasks=4
#SBATCH --ntasks-per-node=5
#SBATCH --threads-per-core=1
#SBATCH -J CALMIP
#SBATCH --time=01:00:00
```


slurm script de soumission

- Pour soumettre un job il faut construire un script de lancement (voir à la racine de vos comptes un exemple de script)

sbatch <arguments> script

- Les arguments du script peuvent être donnés en entête du script (Priorité aux arguments du sbatch)

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks=20
#SBATCH --cpu-per-tasks=4
#SBATCH --ntasks-per-node=5
#SBATCH --threads-per-core=1
#SBATCH -J CALMIP
#SBATCH --time=01:00:00
#SBATCH --qos=mesca
```

slurm script de soumission

- Pour soumettre un job il faut construire un script de lancement (voir à la racine de vos comptes un exemple de script)

sbatch <arguments> script

- Les arguments du script peuvent être donnés en entête du script (Priorité aux arguments du sbatch)

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks=20
#SBATCH --cpu-per-tasks=4
#SBATCH --ntasks-per-node=5
#SBATCH --threads-per-core=1
#SBATCH -J CALMIP
#SBATCH --time=01:00:00
#SBATCH --qos=mesca
#SBATCH --mem=10000
```

Unité MB

slurm variables d'environnement

VARIABLE	DESCRIPTION
SLURM_NODELIST	Liste des nœuds réservés eoscomp[1-6,9]
SLURM_SUBMIT_DIR	Répertoire de lancement
SLURM_NNODES	Nombre de nœuds réservés
SLURM_NTASKS_PER_NODE	Nombre de tâches MPI par nœud
SLURM_JOBID	ID du job slurm
SLURM_NTASKS	Nombre de tâches MPI

slurm un exemple

```
# entête

module purge
module load intel intelmpi

workdir=$SLURM_SUBMIT_DIR/DIR_$SLURM_NTASKS_$SLURM_JOBID
mkdir $workdir
cd $workdir

for i in $( nodeset –e $SLURM_NODELIST ) ; do
for j in {1..20}; do echo $i >> machinefile ; done
done

#lancement
```


slurm lancement

- deux façons de lancer votre job MPI:
  - **srun**
    - srun et intelmpi

      export I_MPI_PMI_LIBRARY=/usr/lib64/libpmi.so

      srun –n <tasks> ./binaire
    - srun et bullxmpi/openmpi

      srun –resv-ports –n <tasks> ./binaire
  - **mpirun/mpiexec/mpiexec.hydra .....**
    - Il faudra désactiver le placement slurm si on laisse la bibliothèque MPI le faire elle-même.

      sbatch –cpu_bind=none

slurm srun sans utiliser un script

- On peut aussi soumettre un job à partir du répertoire dans lequel se trouve votre binaire et vos inputs sans utiliser de script slurm

module load env_compilo + env_mpi

srun –N 1 –n 2 ./binaire

- **SLURM propage votre environnement**

slurm exercices

Créer un script slurm qui utilise la commande srun et le lancer sur deux nœuds (4 ppn).
hostname est une commande linux

```
#!/bin/bash
#SBATCH --nodes=?
#SBATCH --ntasks=?
#SBATCH --ntasks-per-node=?
#SBATCH --threads-per-core=1
#SBATCH -J My_Name
#SBATCH --time=00:10:00

srun ?? hostname
```

slurm exercices

Créer un script slurm qui utilise la commande srun et le lancer sur deux nœuds (4 ppn).

```
#!/bin/bash
#SBATCH --nodes=2
#SBATCH --ntasks=8
#SBATCH --ntasks-per-node=4
#SBATCH --threads-per-core=1
#SBATCH –J My_Name
#SBATCH --time=00:10:00

srun –n $SLURM_NTASKS hostname
ou
srun  hostname
```


slurm exercices

Même exercice mais directement en utilisant srun sans écrire de script

slurm **exercices**

calmip

Même exercice mais directement en utilisant srun
sans écrire de script

```
srun –N 2 –n 8 --ntasks-per-node=4 hostname
ou
srun –N 2 –n 8 hostname
```


Les modules

calmip

Fev 2018

modules commandes

- L'environnement module permet de charger un environnement et de le supprimer proprement

COMMANDE	DESCRIPTION
module av	Liste des modules disponibles
module li	Liste des modules chargés dans votre environnement
module load mod_name	Chargé le module dans votre environnement
module unload mod_name	Supprimer le module de votre environnement
module display mod_name	Permet de voir le fichier de configuration du module
module use <directory>	Permet d'ajouter vos propres modules
module sw lib1 lib2	Permet de charger une lib à la place d'une autre

modules eos

- Ce qui est chargé par défaut sur EOS :

```
[bull@eoslogin1 ~]$ module li
Currently Loaded Modulefiles:
  1) oscar-modules/1.0.3   2) intel/14.0.2.144      3) intelmpi/4.1.3.049
```

- Ce qui est à votre disposition sur EOS :

```
------------------------------------------------- /opt/modules/modulefiles -------------------------------
bullxde/3.1                          bullxmpi_mlx/bullxmpi_mlx-1.2.9.2      oscar-modules/1.0.3(default)
bullxmpi/bullxmpi-1.2.9.2            bullxmpi_mlx_mt/bullxmpi_mlx_mt-1.2.9.2

------------------------------------------------- /usr/local/modules/modulefiles -------------------------
ansys/15.07                          hypre/2.9.0bbullxmpimkl                mxm/3.4.3082
ansys/16                             intel/11.1                             ncl/6.2.0
ansys/16.2                           intel/11.1.064                         nco/4.4.4
ansys/17.0                           intel/12.1.5                           netcdf/4.3.2
ansys/17.2                           intel/13.1.1                           openbugs/3.2.3
bullxmpi/bullxmpi-1.2.8.3            intel/14.0.2.144(default)              openmpi/1.8.2
cdo/1.6.4                            intel/15.0.090                         paraview/4.1.0
chdb/bullxmpi/0.65                   intel/16.1.0                           paraview/4.4.0(default)
chdb/bullxmpi/0.66                   intel/16.1.1                           paraview/pvbatch-4.4.0
chdb/intelmpi/0.65                   intel/16.1.3                           pnetcdf/1.5.0_intelmpi
chdb/intelmpi/0.66                   intel/2011.sp4                         python/2.7.6
clang/3.7.1                          intel/mkl.11.0.5                       R/3.1.1
ffmpeg/25062014                      intel-fc-9/9.1                         tcltk/8.6.3
fftw/3.3.4_openmp                    intelmpi/3.2.2.006                     totalview/2016.05.01
gcc/4.6.0                            intelmpi/4.1.3.049(default)            totalview/2016.07.10
gcc/4.9.2                            intelmpi/5.0.1.035                     totalview/8.13
gcc/5.3.0                            intelmpi/5.1.1.109                     totalview/8.14.1
geris/260614                         intelmpi/5.1.2.150                     totalview/8.15.0
git/2.3.0                            intelmpi/5.1.3.210                     totalview/8.15.3
gsl/1.16_intel                       itac/9.0.1                             vtune/2013.17
hdf5/1.8.13                          java/jdk1.8.0_25                       vtune/2015.0
hdf5/1.8.13_bullxmpi                 julia/0.4.6                            vtune/2016.1
hdf5/1.8.14_intelmpi                 meep/1.2.1                             vtune/2016.1.1
hdf5/1.8.15                          mpiprof_bullxmpi/bullxmpi1.2.7.1_compilo14 vtune/2016.1.3
hypre/2.9.0bbullxmpi                 mpiprof_intelmpi/intelmpi4_compilo14
[bull@eoslogin1 ~]$
```

modules bullxmpi 1.2.9.1

- La nouvelle bullxmpi est maintenant à disposition sur EOS
  - bullxmpi : OpenMPI 1.6.5 tunée par bull (interface btl)
  - bullxmpi_mlx : la bullxmpi compilée avec le support des outils mellanox (mxm)
  - bullxmpi_mlx_mt : la bullxmpi compilée avec le support des outils mellanox (mxm) et le support du thread multiple (threads capables de lancer des comm MPI et de faire des I/O)
- La bullxmpi_mlx vous permettra de mesurer les meilleures performances. Plus particulièrement dans le cas d'utilisation intensives de communications point à point.

modules exercise

Quelle bibliothèque MPI est chargée par défaut ?

Veuillez changer cette bibliothèque par une autre ?

```
module li
module load
module unload
module sw
```

Vérifier la bibliothèque MPI chargée avec la commande which mpirun

La compilation

Fev 2018

compilation compilateurs

- Le système de calcul EOS utilise le système d'exploitation linux. Ainsi deux environnements de compilation sont disponibles :
  - GNU : gcc, g++,gfortran
  - Intel(r) : icc, icpc, ifort *(En général plus performant)*

- Librairie MPI (Message Passing Interface)
  - Intel MPI (r) : mpiifort,mpiicc,mpiicpc, mpif90, mpicxx et mpicc
  - BullxMPI (basée sur OpenMPI) : mpif90 et mpicc, mpiCC

mpiifort,mpiicc, mpif90,mpicc sont des wrappers. Ces wrappers permettent à la fois de compiler les sources et de faire l'édition des liens pour la librairie MPI sous-jacente.

Vérifier que les modules intel et intelmpi sont chargé : module li
Tester mpiicc –show pour visualiser ce que fait le wrapper
Tester mpicc –show pour visualiser ce que fait le wrapper

http://www.calmip.univ-toulouse.fr/spip/spip.php?article471
http://www.calmip.univ-toulouse.fr/spip/spip.php?article447

compilation options debug

- Fortran (Intel)
  - *-O0* Disables all optimizations
  - *-warn all* Enables all warning messages.
  - *-warn errors* Force warnings to be reported as errors.
  - *-check all* Enables all check options (runtime)
  - *-fpe0* This setting provides full IEEE support (runtime).
    -fpe3 All floating-point exceptions are disabled
  - -fltconsistency Enables improved foating-point consistency.
    Default OFF
  - -mp1 Improves foating-point precision and consistency.
    Default: OFF: The compiler provides good accuracy and run-time performance at the expense of less consistent floating-point results.
  - -fp-model <keyword> Controls the semantics of floating-point calculations
    Default: -fp-model fast=1 The compiler uses more aggressive optimizations on floating-point calculations.

compilation compilateurs

- Les options de debug les plus courantes
  - Compiler avec -traceback –g
  - -check bounds : Performs run-time checks on whether array subscript and substring references are within declared bounds.

[no]bounds Determines whether checking occurs for array subscript and character substring expressions.

[no]uninit Determines whether checking occurs for uninitialized variables.
all Enables all check options.

compilation compilateurs

- 3 niveaux d'optimisation
  - *-O1 Optimize for code size*
  - *-O2 Default option enables optimizations for speed, including global code scheduling, predication, and speculation*
  - *-O3 Advanced optimization. Enables O2 optimizations plus more aggressive*
- Meta option
  - *-fast ⇔ -ipo -O3 -no-prec-div -static -xHost*
    - *Don't use -fast, this option can change between two compilers version, if you*
      - *use -static option all library must have a static version.*

compilation compilateurs

- Specialisation de la compilation pour votre processeur

- Tells the compiler to generate optimized code specialized for the Intel processor that executes your program : -x<code>
  - *-xAVX: May generate Intel(R) Advanced Vector Extensions.* ***Instructions supported by EOS***
  - *-xCORE-AVX2 : May generate Intel(R) Advanced Vector Extensions 2.* ***Instructions supported only on MESCA node.***

compilation exercices

15

calmip

module li : avez-vous un compilateur à disposition ?
cd TP_FORMATION/portage
make clean
make TEST FLAGS=" "
srun -N1 –n 1 ./portage_test

- Ce test plante, pourquoi ?
- Utilisez certaines options de compilation pour débuguer ce test

man ifort
- -traceback
- –g
- -check bounds ??

compilation exercices

16

calmip

module li : avez-vous un compilateur à disposition ?
cd TP_FORMATION/portage
make clean
make TEST FLAGS=" "
srun -N1 –n 1 ./portage_test

```
Time for test 1 :  3.0000001E-04 s

 A(M,M)=  1.7115440250050047E-002
forrtl: severe (174): SIGSEGV, segmentation fault occurred
Image              PC                Routine            Line        Source
portage_test       00000000004712C9  Unknown               Unknown  Unknown
portage_test       000000000046FB9E  Unknown               Unknown  Unknown
portage_test       0000000000441592  Unknown               Unknown  Unknown
portage_test       00000000004261A3  Unknown               Unknown  Unknown
portage_test       00000000004050BB  Unknown               Unknown  Unknown
libpthread.so.0    00002B7438723710  Unknown               Unknown  Unknown
portage_test       0000000000403644  Unknown               Unknown  Unknown
portage_test       0000000000402D76  Unknown               Unknown  Unknown
libc.so.6          00002B743894FD1D  Unknown               Unknown  Unknown
portage_test       0000000000402C69  Unknown               Unknown  Unknown
```

compilation exercices

module li : avez-vous un compilateur à disposition ?
cd TP_FORMATION/portage
make clean
make TEST **FLAGS=" –g –traceback "**
srun -N1 - ./portage_test

```
 Time for test 1 :  9.3999999E-03 s

 A(M,M)=  1.7115440250500 47E-002
forrtl: severe (174): SIGSEGV, segmentation fault occurred
Image              PC                Routine            Line        Source
portage_test       0000000000470229  Unknown               Unknown  Unknown
portage_test       000000000046EAFE  Unknown               Unknown  Unknown
portage_test       00000000004404F2  Unknown               Unknown  Unknown
portage_test       0000000000425103  Unknown               Unknown  Unknown
portage_test       000000000040416B  Unknown               Unknown  Unknown
libpthread.so.0    00002B7A9DBEE710  Unknown               Unknown  Unknown
portage_test       0000000000403C8B  test1_                   193  portage.f90
portage_test       0000000000403EE6  MAIN__                   299  portage.f90
portage_test       0000000000402D76  Unknown               Unknown  Unknown
libc.so.6          00002B7A9E01ED1D  Unknown               Unknown  Unknown
portage_test       0000000000402C69  Unknown               Unknown  Unknown
```


compilation exercices

module li : avez-vous un compilateur à disposition ?
cd TP_FORMATION/portage
make clean
make TEST **FLAGS=" –check bounds "**
srun -N1 - ./portage_test

```
 Time for test 1 :  1.5200000E-02 s

forrtl: severe (408): fort: (2): Subscript #2 of the array A has value 101 which
is greater than the upper bound of 100

Image              PC                Routine            Line        Source
portage_test       0000000000403C32  Unknown               Unknown  Unknown
portage_test       0000000000402D76  Unknown               Unknown  Unknown
libc.so.6          00002AB605ACCD1D  Unknown               Unknown  Unknown
portage_test       0000000000402C69  Unknown               Unknown  Unknown
```

compilation exercices

module li : avez-vous un compilateur à disposition ?
cd TP_FORMATION/portage
make clean
make TEST **FLAGS=" –g –traceback –check bounds "**
srun -N1 - ./portage_test

```
 Time for test 1 :  1.9200001E-02 s

forrtl: severe (408): fort: (2): Subscript #2 of the array A has value 101 which is
greater than the upper bound of 100

Image              PC                Routine            Line        Source
portage_test       0000000004043FD  test1_                191  portage.f90
portage_test       0000000000404E9A  MAIN__               299  portage.f90
portage_test       0000000000402D76  Unknown           Unknown  Unknown
libc.so.6          00002B3DA27D9D1D  Unknown           Unknown  Unknown
portage_test       0000000000402C69  Unknown           Unknown  Unknown
```


compilation exercices

module li : avez-vous un compilateur à disposition ?
cd TP_FORMATION/povray/povray31/source/unix
make clean
make useicc2 CF=" "
sbatch script.slurm

```
#!/bin/bash
#SBATCH -J Povray
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --time=00:10:00

./povbench2 ../../scenes/advanced/chess2.pov
```

compilation exercices

55

module li : avez-vous un compilateur à disposition ?
cd TP_FORMATION/povray/povray31/source/unix
make clean
make useicc2 CF=" "
sbatch script.slurm

60 sec

```
#!/bin/bash
#SBATCH -J Povray
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --time=00:10:00

ldd ./povray
echo "==============="
./povbench2 ../../scenes/advanced/chess2.pov
```


compilation exercices

56

module li : avez-vous un compilateur à disposition ?
cd TP_FORMATION/povray/povray31/source/unix
make clean
make useicc2 CF=" **–fast** "
sbatch script.slurm

```
#!/bin/bash
#SBATCH -J Povray
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --time=00:10:00

ldd ./povray
echo "==============="
./povbench2 ../../scenes/advanced/chess2.pov
```

compilation exercices

module li : avez-vous un compilateur à disposition ?
cd TP_FORMATION/povray/povray31/source/unix
make clean
make useicc2 CF=" **–fast** "
sbatch script.slurm

54 sec

```
#!/bin/bash
#SBATCH -J Povray
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --time=00:10:00

ldd ./povray
echo "==============="
./povbench2 ../../scenes/advanced/chess2.pov
```


compilation timing

- Le temps d'exécution est mesuré directement par l'application elle-même
- Vous pouvez aussi, utiliser les commandes suivantes pour mesurer le temps d'exécution
  - *time mpirun ….*
  - */usr/bin/time mpirun ….*

compilation exercices

51

cd TP_FORMATION/NPB3.2.1/NPB3.2-MPI
Mettre à jour le fichier config/make.def mpif90 => mpiifort
make clean
make BT CLASS=B NPROCS=64
export I_MPI_PMI_LIBRARY=/usr/lib64/libpmi.so
srun -N 4 -n 64 –time=00:10:00 bin/bt.B.64

compilation exercices

52

make clean
Mettre à jour le fichier config/make.def mpif90 => mpiifort
make BT CLASS=B NPROCS=64
export I_MPI_PMI_LIBRARY=/usr/lib64/libpmi.so
srun -N 4 -n 64 bin/bt.B.64

Time in seconds =	8.38
Total processes =	64
Compiled procs =	64
Mop/s total =	83789.94
Mop/s/process =	1309.22
Operation type =	floating point
Verification =	SUCCESSFUL

8.38 sec

Environnement de développement d'EOS

Librairies scientifiques

➢ BLAS, LAPACK, SCALAPACK, etc ... : Intel® Math Kernel Libray (MKL)

$$\|u\|$$
$$u \leftarrow \alpha Av + \beta u$$
$$C \leftarrow \alpha AB + \beta C$$
$$Au = b$$
$$Au = \lambda u$$
$$A = LU$$

$$\alpha, \beta, \lambda \in IR; u, v, b \in IR^n; A \in IR^{n \times m}, B \in IR^{m \times n}, A \in IR^{n \times n}$$

```
>> module show intel/14.0.2.144
prepend-path        INCLUDE /usr/local/intel/composer_xe_2013_sp1.2.144/mkl/include
```

➢ Le module Intel inclut les compilateurs et la MKL

Page web associée :
http://www.calmip.univ-toulouse.fr/spip/spip.php?article414

Outils

Fev 2018

tools top

- Afin de vérifier le placement et l'avancé de votre application, le plus simple est d'utiliser la commande top
  - 1 : permet d'afficher la liste des cœurs de calcul et leur utilisations (cpu, wait, etc.)
  - fj : pour ajouter une colonne avec le numeros des cœurs
  - H : pour afficher les threads

```
Tasks: 1000 total,   4 running, 921 sleeping,   0 stopped,  75 zombie
Cpu(s):  5.8%us,  9.6%sy,  0.0%ni, 84.6%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  131936084k total, 88493288k used, 43442796k free,    69328k buffers
Swap:        0k total,        0k used,        0k free, 65946488k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+   P COMMAND
  313 rmartin   20   0 15980  996  700 R 100.0  0.0  41141:19  5 pmi_proxy
  390 rmartin   20   0 15824  964  700 R 100.0  0.0  41141:14 12 pmi_proxy
32313 brissaud  20   0  422m 405m  960 R 100.0  0.3 127:38.07 10 xseismic_CPML_2
 4332 guiltat   20   0  590m  38m  10m S  2.6  0.0   1:17.59  2 vmd_LINUXAMD64
  276 root      39  19     0    0    0 S  1.0  0.0   3468:17  0 kipmi0
20542 bull      20   0 13800 1952  920 R  0.7  0.0   0:00.10  0 top
20551 marsale   20   0  140m 3692 2268 S  0.7  0.0   0:00.02  2 vim
 2148 rpc       20   0 19136  928  520 S  0.3  0.0  22:03.31  0 rpcbind
 4043 root      20   0     0    0    0 S  0.3  0.0 234:30.16  6 ptlrpcd_1
 4048 root      20   0     0    0    0 S  0.3  0.0 187:23.67  0 ptlrpcd_6
 6863 hneau     20   0 13940 2068 1008 S  0.3  0.0  16:59.61  8 top
12438 vijayc    20   0  105m 2224  936 S  0.3  0.0   0:01.30 11 watch
18952 root      20   0     0    0    0 S  0.3  0.0   0:05.60  0 flush-lustre-2
```


top exercices

cd TP_FORMATION/NPB3.2.1/NPB3.2-MPI
make BT CLASS=**D** NPROCS=**25**
export I_MPI_PMI_LIBRARY=/usr/lib64/libpmi.so
srun –N 2 –n 25 bin/bt.**D**.25
Ouvrez un second terminal.
Connectez-vous aux nœuds réservés et essayez la commande top

20 mn

top –u ..
- 1
- fj

calmip quelques scripts

- Ajouter à la fin de votre script slurm:

- jobinfo $SLURM_JOBID
- Donne des informations sur votre job

- infoincidentjob
- En cas de plantage vous donne des informations sur la raison de celui-ci

tools gprof

- Comment ?
  - Compiler et linker votre code avec -p et –g
  - Faites le tourner sur EOS
  - A la fin de l'exécution vous obtenez des fichiers gmon.out.*
  - Puis utiliser la commande suivante:

    gprof a.out gmon.out.*

- Un exemple de ce que vous pouvez obtenir

```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 26.21      9.10     9.10 67697001     0.00     0.00  binvcrhs_
 17.93     15.32     6.22      603     0.01     0.02  y_solve_cell_
 11.79     19.41     4.09      603     0.01     0.01  z_solve_cell_
 11.60     23.43     4.03 67697001     0.00     0.00  matmul_sub_
  9.86     26.85     3.42      202     0.02     0.02  compute_rhs_
  7.78     29.55     2.70      603     0.00     0.01  x_solve_cell_
  3.20     30.66     1.11 67697001     0.00     0.00  matvec_sub_
```

tools mpiprof

- MPIPROF est un profiler mpi léger, son impact sur les performances est minimal
- Commande à lancer dans votre script slurm

module load <intel_compiler>
module load <mpi_lib>

puis

mpiprof_bullxmpi/bullxmpi1.2.7.1_compilo14

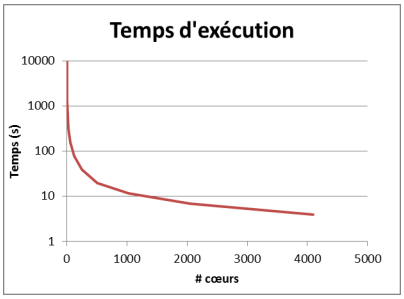
ou

mpiprof_intelmpi/intelmpi4_compilo14

tools mpiprof

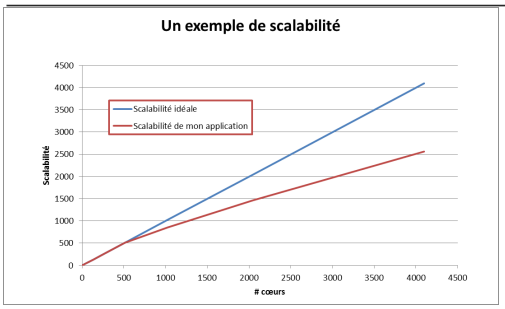
```
MPI Communications Statistics:
==============================

Total Parallel time : 7281.543846 s
Total Communication time : 551.369430 s
Ratio :   7.57 %

Accumulated data of all processes:
==================================
                        Min [R]          Max [R]          Average      Stand dev.

    Parallel time:     910.185775 [7]    910.239525 [0]        910.192981       0.017595
(  0.00 %)
Communication time:      21.771790 [0]    150.005858 [7]         68.921179      58.516123
( 84.90 %)
.....
MPI_Allreduce: 303.233854 s ( 55.00 %)
          number          16730 [0]        16730 [0]          16730.00
          time (s)      4.767301 [0]     106.838756 [6]         37.904232
          size (b)       135232 [0]       135232 [0]         135232.00
```

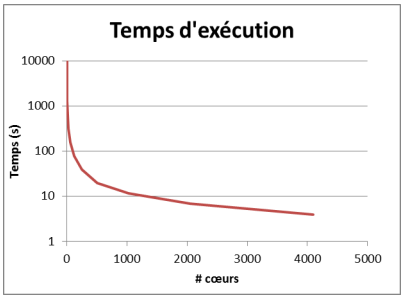
tools Scalabilité



- Le temps d'exécution de mon application décroit lorsque j'augmente le nombre de cœurs.
- Mais mon application est-elle réellement efficace ?

tools Scalabilité





- Il faut calculer la scalabilité de mon code (c1 x t1) / tn
- La scalabilité idéale est atteinte si en doublant le nombre de cœur je divise le temps par deux.

- Ici au-delà de 500 cœurs, la scalabilité décroit et au-delà de 2000 cœurs je consomme du temps CPU pour rien. Et le temps CPU c'est ce que je paye !

tools vtune

- Analyser la performance de votre code : Vtune [amplifierXE]

Vtune [amplifierXE] est un outil d'analyse de Performance de code. Il permet de déterminer facilement les régions chaudes du codes (c'est-à-dire les régions les plus consommatrices en tant de calculs]. Il permet de déterminer le profil d'exécution d'un code multihreadé (OpenMP). Vtune s'utilise à travers une interface graphique : amplxe-gui

- Utiliser Vtune sur Eos

module load vtune/2015.0

http://www.calmip.univ-toulouse.fr/spip/spip.php?article450

tools visualisation graphique a distance

- Pourquoi la visualisation à distance ?

Les logiciels scientifiques génèrent souvent de grosses quantités de données, qu'il sera nécessaire de post-traiter, par exemple en utilisant des logiciels de visualisation. Ces logiciels demandent des ressources non négligeables (carte graphique, mémoire) pas toujours disponibles dans les laboratoires. Il est par ailleurs laborieux de rapatrier systématiquement en local de gros fichiers de données.

- La visualisation à distance permet de résoudre ces difficultés :

Les fichiers n'ont pas besoin d'être transférés
Les ressources graphiques nécessaires se trouvent à Calmip, il n'y a pas besoin de grosse station graphique au laboratoire pour visualiser les données, même en 3D, dans de bonnes conditions

- Prérequis

Le logiciel turboVnc doit être installé sur votre poste de travail. Vous n'avez pas besoin de l'intégralité du paquetage, seule la partie client (vncviewer) est utile
Si vous ne souhaitez pas l'installer, il est possible d'utiliser notre déploiement java web start (attention ! URL provisoire, vous devez ajouter l'URL https://lipm-grpware.toulouse.inra.... à votre liste d'applications autorisées par java)

http://www.calmip.univ-toulouse.fr/spip/spip.php?article463

tools totalview

- Le débogueur totalview

totalview est parfaitement intégré à l'outil de batch slurm, il est simple de déboguer des codes parallèles en les faisant tourner sur le nœuds de calcul. Il possède une interface graphique, qui sera exécutée sur la frontale.

- Utiliser totalview sur eos

Prérequis
Rappel : se connecter avec serveur X
Votre code doit être compilé avec les switches -g et (de préférence) -O0

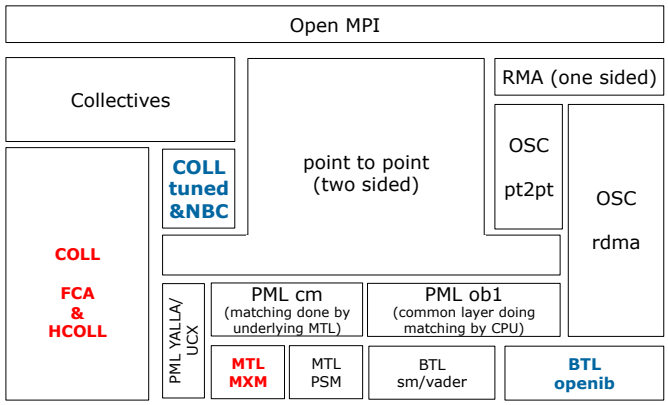
Charger totalview

Module load totalview

Amélioration des performances

Fev 2018

bullxmpi OpenMPI Modular component Architecture



bullxmpi
bullxmpi_mlx

bullxmpi OpenMPI tuning

- Quelles sont les variables que l'on peut modifier ?
  - ompi_info –all : liste l'ensemble des variables mca disponibles
  - /opt/mellanox/mxm/bin/mxm_dump_config –f : configuration de mxm
- Comment les utiliser ?
  - export OMPI_MCA_<nom de la var>=<valeur>
  - export MXM_<nom>=<valeur>

bullxmpi OpenMPI tuning

- Communications ud (unrielable datagram)
  export MXM_TLS=ud,shm,self
- Communications rc (rielable connected)
  export MXM_TLS=rc,shm,self
- Par défaut MXM fait son propre choix entre rx et ud.

- Pour tuner les opérations collectives :

  export OMPI_MCA_coll_tuned_use_dynamic_rules=1
  export OMPI_MCA_coll_tuned_**bcast**_algorithm=<1 à 5>

- et des milliers d'autres ….

slurm Outil placement

- CALMIP met à votre disposition un outil qui permet de placer vos taches MPI et/ou vos threads OpenMP automatiquement

https://www.calmip.univ-toulouse.fr/spip/spip.php?article502

- En argument pour placement on passe d'abord le nombre de processus MPI, puis le nombre de trhreads par processus MPI. Dans le cas ci-dessous on génère par défaut l'argument '—cpu_bind' qui sera utilisé par la commande 'srun' dans votre script SLURM:

```
[@eoslogin1 ]$ placement 4 5
--cpu_bind=mask_cpu:0x1f,0x7c00,0x3e0,0xf8000
```

- Pour l'utiliser dans votre script SLURM vous pouvez taper la commande suivante:

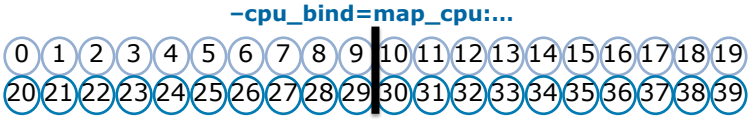
```
srun $(placement 4 5) ./Mon_application_MPI
```

slurm contrôler le placement des processus

- Afin d'améliorer les performances de votre application, il est conseillé de jouer sur le placement de vos processus MPI et tâches OpenMP
- Pour vous aider, à placer vos tâches vous pouvez utiliser l'option cpu_map de srun.

slurm srun sans utiliser les cœurs logiques

- pur MPI, on va placer les tâches mpi de manière très précise avec **–cpu_bind=map_cpu:…**



slurm srun sans utiliser les cœurs logiques

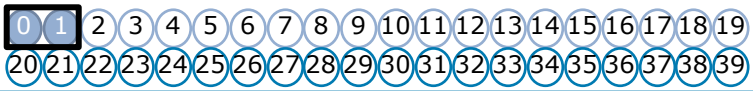
- pur MPI, on va placer les tâches mpi de manière très précise avec –cpu_bind=map_cpu:…

```
if [ $OMP_NUM_THREADS == 1 ] ; then
  time srun -K1 --resv-ports --cpu_bind=map_cpu:0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19  -n
$mpi_tasks  $LOADMODULE
```

slurm srun sans utiliser les cœurs logiques

- OMP_NUM_THREADS=2 et PPN=10

1 1 en binaire = 3 en Hexadécimal

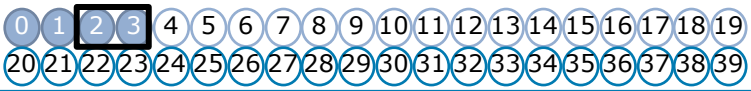



```
if [ $OMP_NUM_THREADS == 1 ] ; then
  time srun -K1 --resv-ports --cpu_bind=map_cpu:0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19  -n
$mpi_tasks  $LOADMODULE

elif [ $OMP_NUM_THREADS == 2 ] ; then
  time srun -K1 --resv-ports  -c $OMP_NUM_THREADS  --cpu_bind=mask_cpu:0x3,0xc,
0x30,0xc0,0x300,0xc00,0x3000,0xc000,0x30000,0xc0000 -n $mpi_tasks  $LOADMODULE

elif [ $OMP_NUM_THREADS == 20 ] ; then
```

slurm srun sans utiliser les cœurs logiques

- OMP_NUM_THREADS=2 et PPN=10

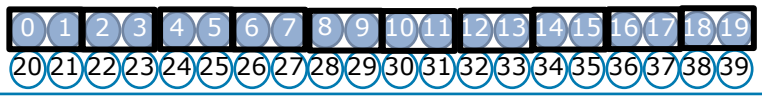
0 0 1 1 → 1100 en binaire = C en Hexadécimal




```
if [ $OMP_NUM_THREADS == 1 ] ; then
  time srun -K1 --resv-ports --cpu_bind=map_cpu:0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19  -n
$mpi_tasks  $LOADMODULE

elif [ $OMP_NUM_THREADS == 2 ] ; then
  time srun -K1 --resv-ports  -c $OMP_NUM_THREADS  --cpu_bind=mask_cpu:0x3,0xc,
0x30,0xc0,0x300,0xc00,0x3000,0xc000,0x30000,0xc0000 -n $mpi_tasks  $LOADMODULE

elif [ $OMP_NUM_THREADS == 20 ] ; then
```

slurm srun sans utiliser les cœurs logiques

- OMP_NUM_THREADS=2 et PPN=10




```
if [ $OMP_NUM_THREADS == 1 ] ; then
  time srun -K1 --resv-ports --cpu_bind=map_cpu:0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19  -n
$mpi_tasks  $LOADMODULE

elif [ $OMP_NUM_THREADS == 2 ] ; then
  time srun -K1 --resv-ports  -c $OMP_NUM_THREADS  --cpu_bind=mask_cpu:0x3,0xc,
0x30,0xc0,0x300,0xc00,0x3000,0xc000,0x30000,0xc0000 -n $mpi_tasks  $LOADMODULE

elif [ $OMP_NUM_THREADS == 20 ] ; then
```

slurm srun en utilisant les cœurs logiques

- pur MPI …. Est-ce vraiment efficace ?

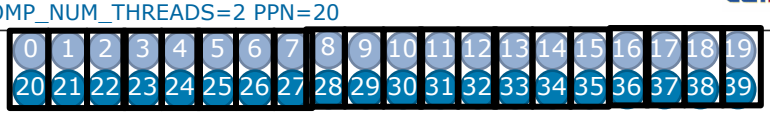
slurm srun en utilisant les cœurs logiques

- pur MPI …. Est-ce vraiment efficace ?


```
if [ $OMP_NUM_THREADS == 1 ] ; then
  time srun -K1 --resv-ports --cpu_bind=map_cpu:
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36
,37,38,39  -n $mpi_tasks  $LOADMODULE
```

slurm srun en utilisant les cœurs logiques

- OMP_NUM_THREADS=2 PPN=20




```
if [ $OMP_NUM_THREADS == 1 ] ; then
  time srun -K1 --resv-ports --cpu_bind=map_cpu:
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39
-n $mpi_tasks  $LOADMODULE
elif [ $OMP_NUM_THREADS == 2 ] ; then
  time srun -K1 --resv-ports  -c $OMP_NUM_THREADS  --cpu_bind=mask_cpu:
0x100001,0x200002,0x400004,0x800008,0x1000010,0x2000020,0x4000040,0x8000080,0x10000100,
0x20000200,0x40000400,0x80000800,0x100001000,0x200002000,0x400004000,0x800008000,0x100
0010000,0x2000020000,0x4000040000,0x8000080000 -n $mpi_tasks  $LOADMODULE
```

Thanks

For more information please contact:
T+ 33 1 98765432
F+ 33 1 88888888
M+ 33 6 44445678
firstname.lastname@atos.net

dd-mm-yyyy