Formation utilisateurs HPC@LR

Gestion des ressources pour les exécutions

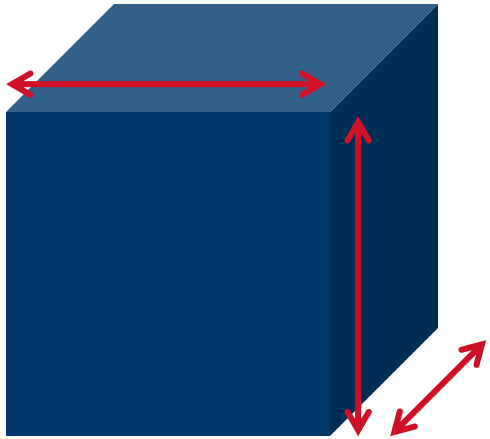
Marc Mendez-Bermond – Expert Solutions HPC

Au programme

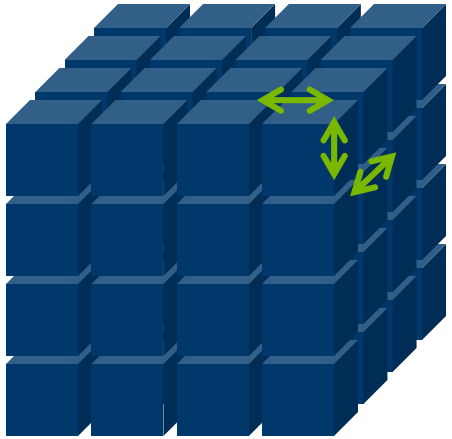
- **Parallélisme et paradigmes de programmation**
  - OpenMP, MPI et OpenMP/MPI
  - Traitement par lots
- **Gestion des ressources**
  - SLURM et ses concepts
  - Scripts de lancement
- **Spécificités Intel MPI et Intel TrueScale**
  - Environnement

Parallélisme et paradigmes de programmation

Principes du calcul parallèle



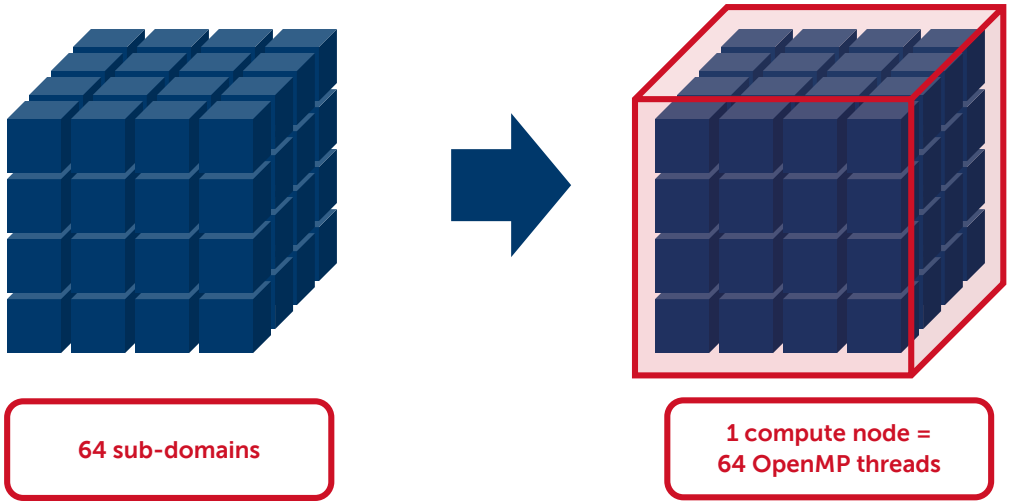
1 problem : size N
Tseq. = X * Y * Z + seq global



N problems : size 1
T// = x * y * z + seq. global

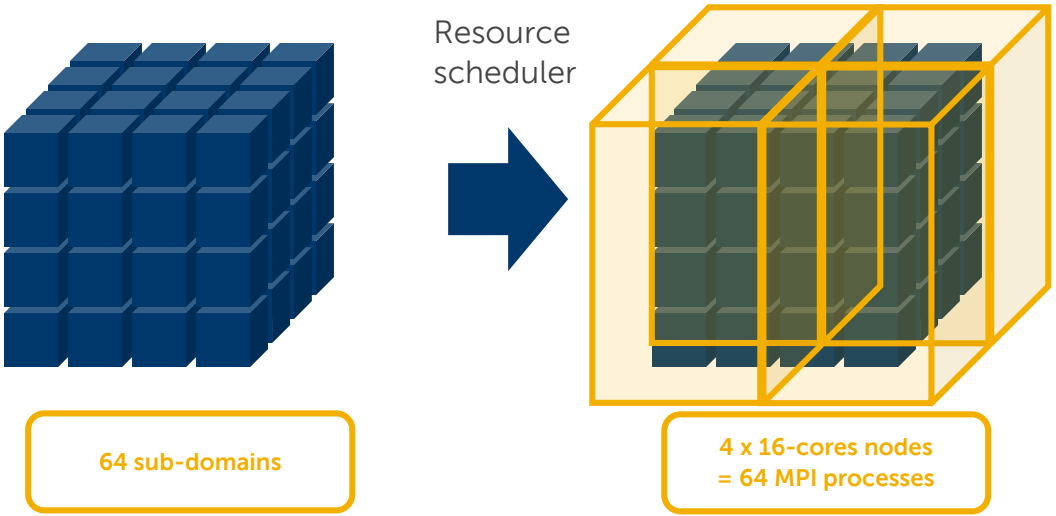
- Décomposition du domaine en sous-domaines adaptés à la physique du phénomène
- L'algorithme numérique implémentera un motif de communications pour coordonner les calculs réalisés pour chacun des sous-domaines
- Multiplication de la puissance de calcul et de la mémoire disponibles au prix d'un effort sur les algorithmes et la gestion de l'infrastructure

Programmation de calculs parallèles : **OpenMP**



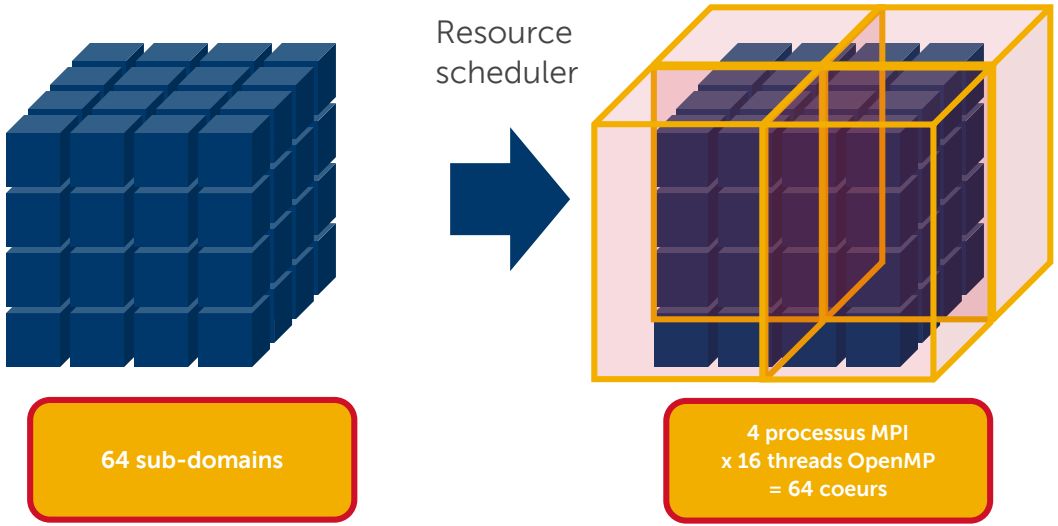
- OpenMP : parallélisation par **multithreading**
- Usage : directives simples insérées dans le code source pour indiquer les sections parallèles, synchronisations, données partagées et autres réglages
- **(Aujourd'hui) aucun support pour les communications inter-nœuds**

Programmation de calculs parallèles : **MPI**



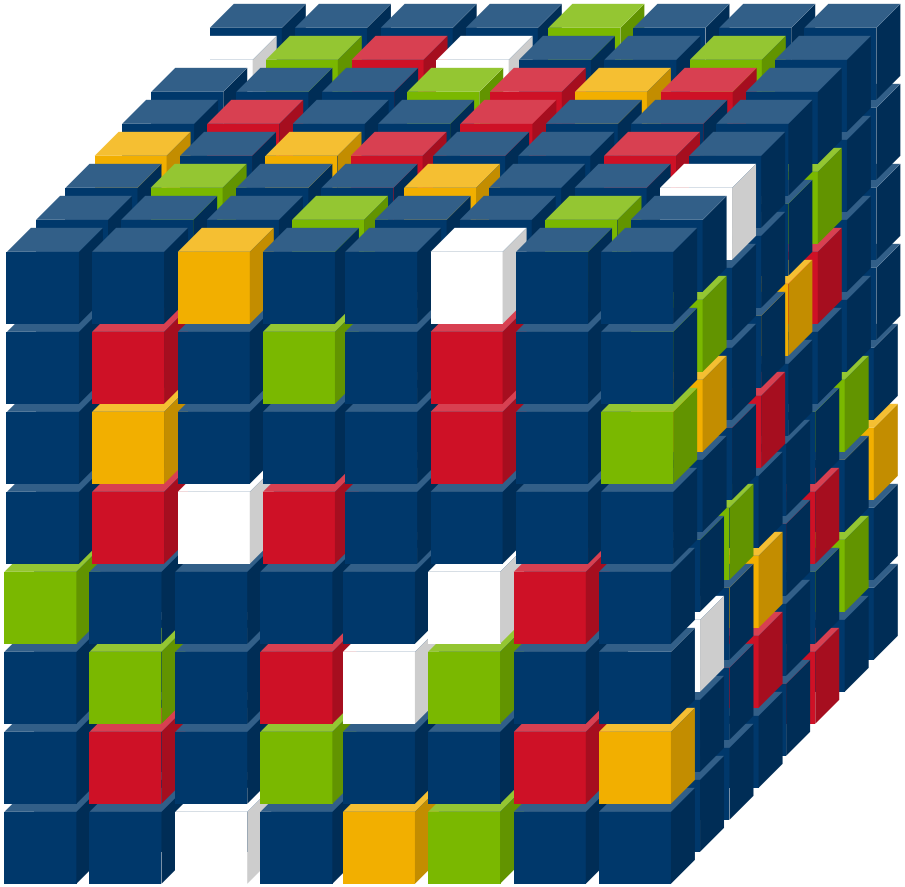
- MPI : bibliothèque d'envoi de messages entre **tâches MPI**
- Usage : programmation spécifique, totalement explicite (contrôle total = grands risques de problèmes dus au séquencement des communications)
- **Communications intra et inter-nœuds pour 1 à 100000 nœuds**
- Réseau d'interconnexion InfiniBand largement souhaité : faible latence et haut débit

Programmation de calculs parallèles : **MPI/OpenMP**

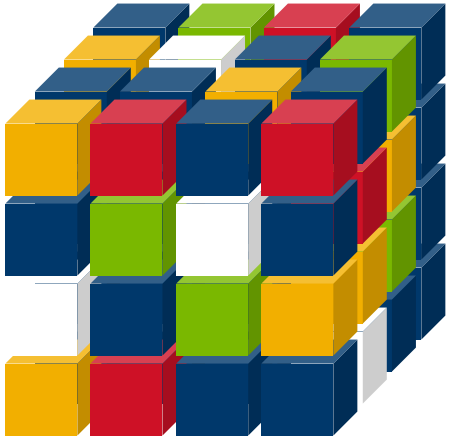


- Mode hybride avec :
  - OpenMP en intra-nœud pour distribuer des **threads** sur les cœurs de calcul
  - MPI pour les communications inter-nœuds entre **tâches MPI**
- **Généralement un paradigme très performant, évolutif et très efficace énergétiquement**

Traitement parallèle par lot : grilles de calcul/séquentiel



Batch processing



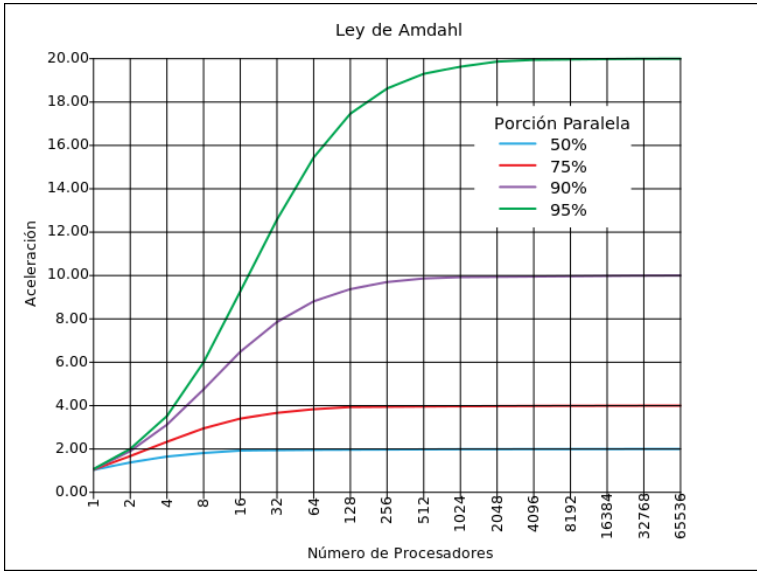
X problems with size=1 run on a grid computer with N cores
=
Acceleration ~ N
Computers may be geographically distributed !

Facteurs de performances

- Le parallélisme permet d'accroitre les performances dans les limites de la loi d'Amdhal :

$$R = \frac{1}{(1-s) + \frac{s}{N}}$$

- Cette formule met en relation le taux de parallélisme de l'application (s, en fraction du temps d'exécution) avec le nombre de processeurs disponibles pour une exécution parallèle (N).



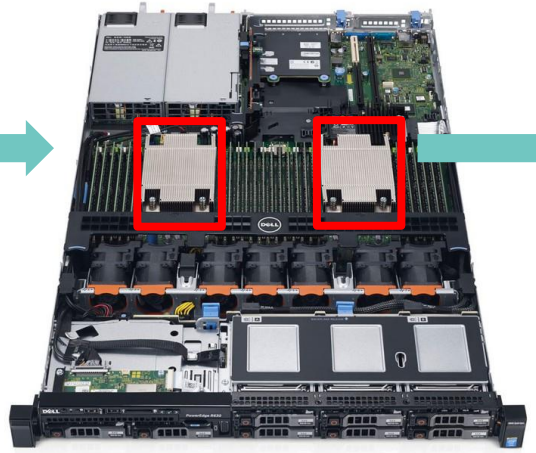
Gestion de ressources

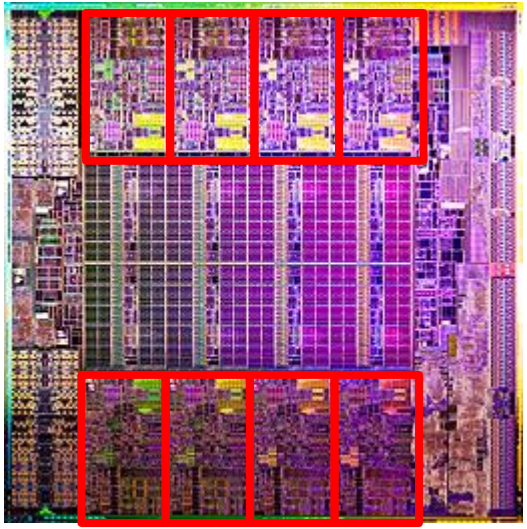
Gestion de ressources et SLURM

- Un **ordonnanceur** organise l'allocation des ressources disponibles du supercalculateur.
- Les **travaux** de l'ensemble des utilisateurs sont orchestrés en fonction :
  - Des ressources demandées (nombre de cœurs/nœuds, mémoire, ...) – ajustées par l'utilisateur
  - De la politique de priorités du supercalculateur – ajustée par l'administrateur
- L'**ordonnanceur** est un outil qui organisera l'exécution, la collecte des informations de l'exécution, les fichiers de sortie et les éventuelles défaillances.
- SLURM est l'ordonnanceur installé sur Thau
  - SLURM (*Simple Linux Utility for Resource Management*) est un pur développement libre démarré par le LLNL pour répondre aux besoins des supercalculateurs dotés de CPU multi-cœurs.

Hiérarchie des ressources des grappes de calcul

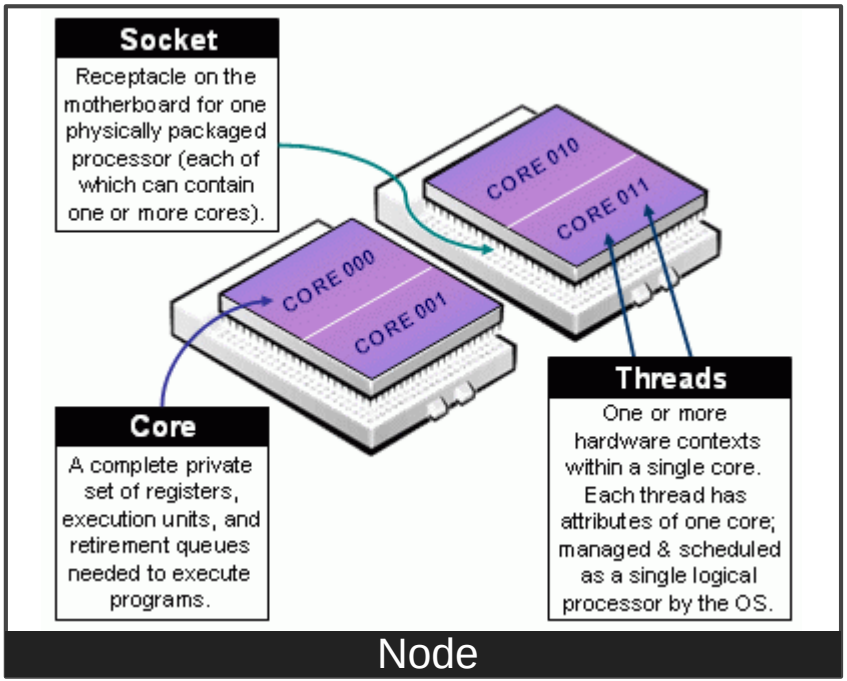
La **grappe** est composée de **nœuds de calcul**	Chaque **nœud de calcul** dispose d'1 ou plusieurs **processeurs**	Les **processeurs** sont composés de plusieurs **cœurs** de calcul





SLURM : Multi-core/Multi-thread

Il est nécessaire de comprendre les correspondances entre le vocabulaire SLURM et celui des systèmes et applications !



Le **node** correspond à un nœud de calcul ou serveur.

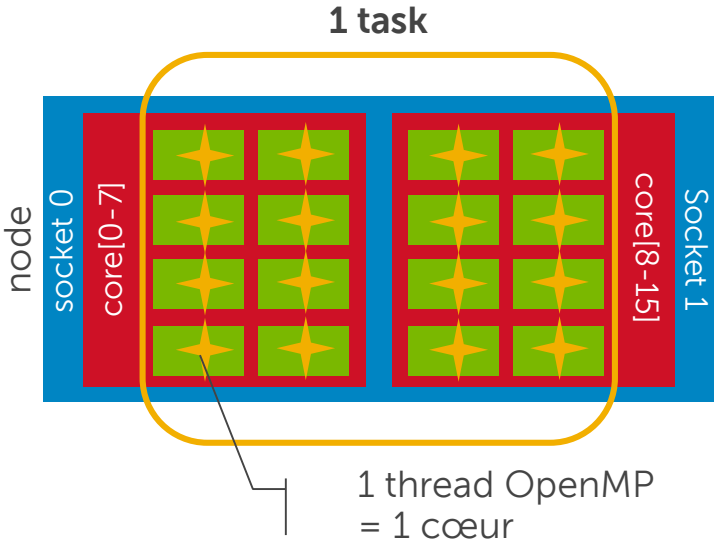
Le **socket** est un arrangement physique pour les cœurs de processeur qui partagent certaines ressources : cache L3 et canaux mémoire en particulier.

Le **cœur** est l'unité élémentaire de calcul qui est le grain le plus fin d'allocation. Il est parfois aussi appelé processeur ou CPU du fait d'un lexique remontant aux processeurs mono-cœurs.

Le **thread** SLURM correspond aux contextes matériels proposés par la technologie Intel Hyper-Threading. Vous pouvez généralement assumer qu'un thread est équivalent à un cœur.

Exécution OpenMP

- Exécution OpenMP sur 1 nœud et N threads
- SLURM requiert les paramètres suivants :
  - 1 nœud
  - 1 tâche par nœud
  - N cœurs par tâche (N doit être inférieur ou égal au nombre de cœurs disponibles sur 1 nœud !!!)



N = 16

```
$ srun -N 1 --ntasks-per-node=1 --cpus-per-task=[N] ./openmp.x
```

Exécution batch OpenMP

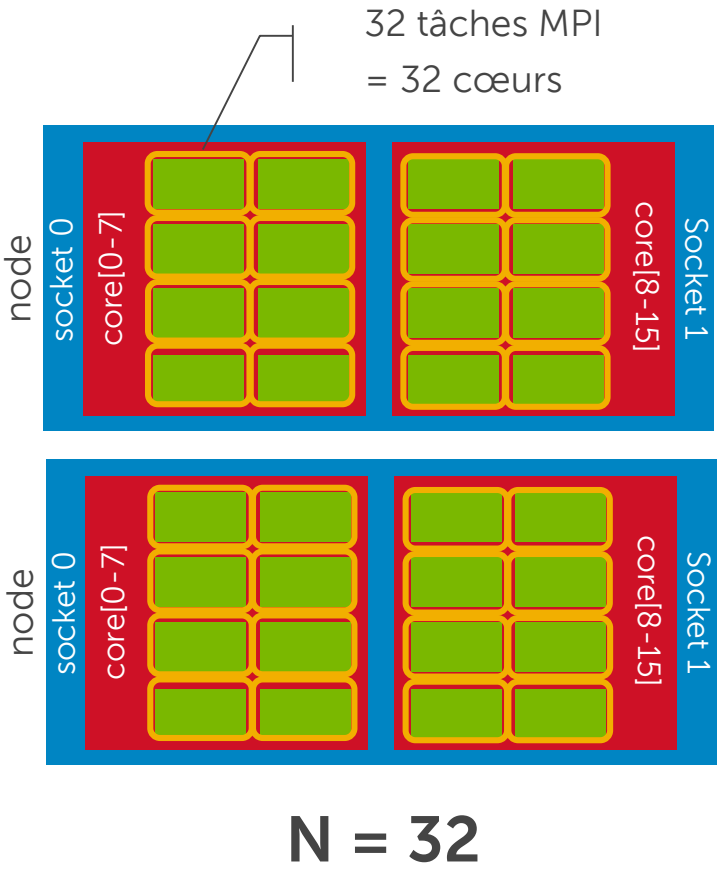
Intel C/C++/Fortran

```
#!/bin/bash
#SBATCH --job-name openmp
#SBATCH --output openmp-%j.out
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=16

cd $SLURM_SUBMIT_DIR

# OpenMP runtime settings
export OMP_NUM_THREADS=$SLURM_CPUS_ON_NODE

./openmp.x
```

Exécution MPI (simple)

- Exécution MPI avec N tâches
- SLURM requiert les paramètres suivants :
  - Le nombre de tâches

```
$ srun –n [N] ./mpi.x
```



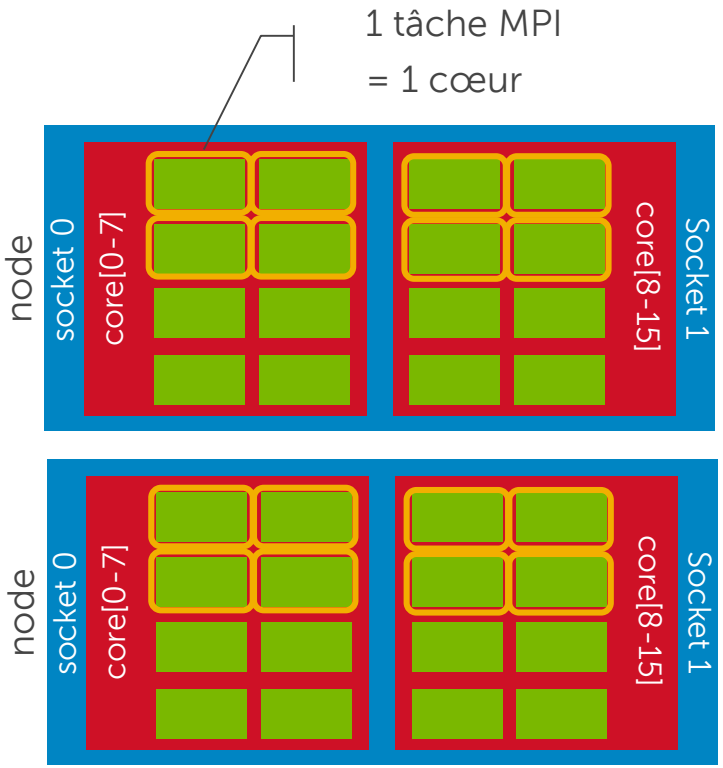
Exécution batch MPI (simple toujours)

Intel MPI

```
#!/bin/bash
#SBATCH --job-name mpi-simple1
#SBATCH --output mpi-simple1-%j.out
#SBATCH -n 32

cd $SLURM_SUBMIT_DIR
srun ./mpi.x
```

Exécution MPI (moins simple)

- Exécution MPI avec N tâches en « dépeuplant » de 50%
- SLURM requiert les paramètres suivants :
  - Le nombre de nœuds : *nodes*
  - Le nombre de tâches par nœud : *tpn*
  - Avec N = *nodes* * *tpn*



N = 16

```
$ srun –N [nodes] –tasks-per-node [tpn] ./mpi.x
```

Exécution batch MPI – placement logique

Intel MPI

```
#!/bin/bash
#SBATCH --job-name mpi-simple2
#SBATCH --output mpi-simple2-%j.out
#SBATCH -n 16
#SBATCH --ntasks-per-node 8

cd $SLURM_SUBMIT_DIR
srun ./mpi.x
```

Exécution batch MPI – placement physique

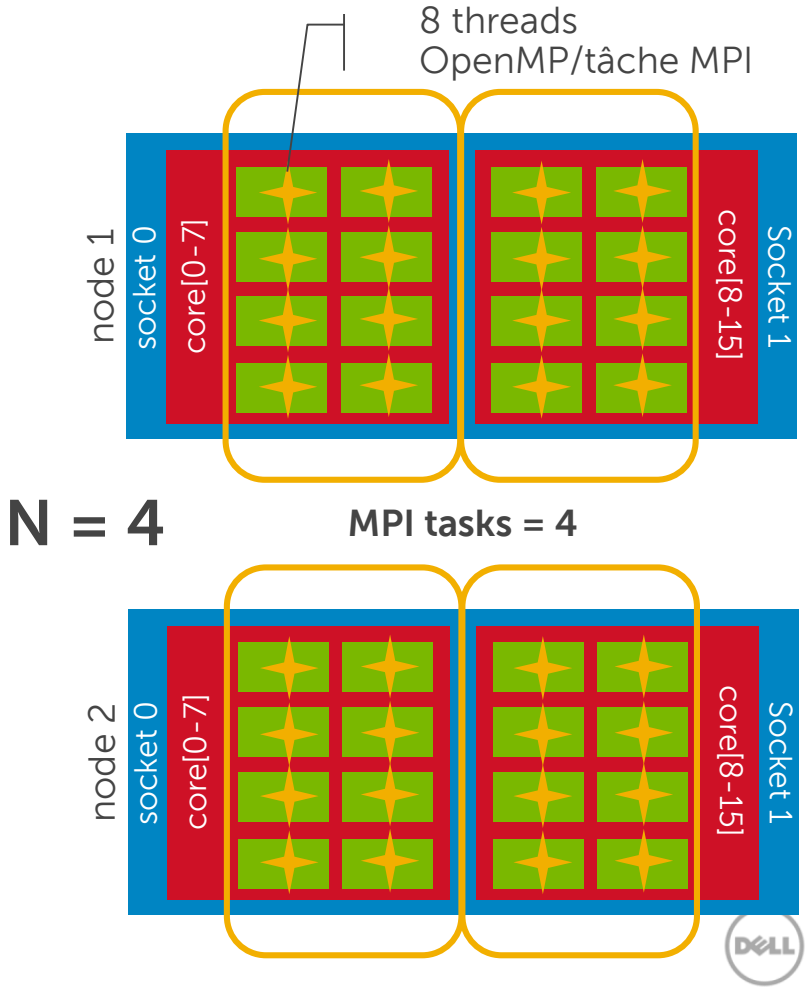
Intel MPI

```
#!/bin/bash
#SBATCH --job-name mpi-simple3
#SBATCH --output mpi-simple3-%j.out
#SBATCH -n 16
#SBATCH --nsockets-per-node 2
#SBATCH --ncores-per-socket 4

cd $SLURM_SUBMIT_DIR
srun ./mpi.x
```

Exécution OpenMP/MPI

- Exécution MPI/OpenMP
  - 2 niveaux hiérarchiques pour le parallélisme
    - Le nombre de tâches MPI à distribuer sur les nœuds
    - Le nombre de threads OpenMP par tâche MPI
- SLURM requiert les paramètres suivants :
  - Le nombre de tâches
  - Le nombre de processeurs/cœurs par tâche

```
$ srun –n [N] --ntasks-per-node 2 \
        --ncpus-per-task 8 ./openmp-mpi.x
```



Exécution batch OpenMP/MPI

Intel MPI

```
#!/bin/bash
#SBATCH --job-name openmp-mpi
#SBATCH --output openmp-mpi-%j.out
#SBATCH -n 4
#SBATCH --ntasks-per-node=2
#SBATCH --cpus-per-task=8

echo "Running on: $SLURM_NODELIST"
echo "SLURM_NTASKS=$SLURM_NTASKS"
echo "SLURM_NTASKS_PER_NODE=$SLURM_NTASKS_PER_NODE"
echo "SLURM_CPUS_PER_TASK=$SLURM_CPUS_PER_TASK"
echo "SLURM_NNODES=$SLURM_NNODES"
echo "SLURM_CPUS_ON_NODE=$SLURM_CPUS_ON_NODE"
```

```
# OpenMP settings
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

cd $SLURM_SUBMIT_DIR
srun ./mpi.x
```

Intel MPI et Intel TrueScale

Environnement spécifique (provisoire)

- La bibliothèque Intel MPI doit disposer d'un environnement spécifique pour l'interconnexion Intel TrueScale.
- Le cluster thau sera prochainement configuré par défaut pour Intel TrueScale. D'ici là, voici la marche à suivre :

```
$ cat ${HOME}/etc/tmi.conf
# TMI provider configuration
psm 1.0 libtmip_psm.so " "

$ export TMI_CONFIG=${HOME}/etc/tmi.conf

$ export I_MPI_FABRICS=tmi

$ export I_MPI_PMI_LIBRARY=<slurmpath>/lib64/libpmi.so

$ srun –n 64 mpi.x
```

Script complet Intel MPI / Intel TrueScale

```
#!/bin/bash
#SBATCH --job-name mpi-simple
#SBATCH --output mpi-simple-%j.out
#SBATCH -n 64

module purge
module load slurm
module load intel-mpi
module list

echo "Running on: $SLURM_NODELIST"
echo "SLURM_NTASKS=$SLURM_NTASKS"
echo
"SLURM_NTASKS_PER_NODE=$SLURM_NTASKS_PER_NODE"
echo "SLURM_CPUS_PER_TASK=$SLURM_CPUS_PER_TASK"
echo "SLURM_NNODES=$SLURM_NNODES"
echo "SLURM_CPUS_ON_NODE=$SLURM_CPUS_ON_NODE"

SLURM_LIB=$( echo $LD_LIBRARY_PATH | sed -e
's/:/\n/g' | grep slurm | grep "lib64$" )
if [ -z "${SLURM_LIB}" ]; then
        echo "SLURM_LIB cannot be set from
LD_LIBRARY_PATH, trying from PATH"
        SLURM_LIB=$( which sinfo | sed -e
's$bin/sinfo$lib64$' )
        if [ -z "${SLURM_LIB}" ]; then
                echo "SLURM_LIB cannot be set
from PATH, aborting. Check your SLURM
environment."
                exit 255
        fi
fi

echo "SLURM_LIB set to ${SLURM_LIB}"
export I_MPI_PMI_LIBRARY=$SLURM_LIB/libpmi.so
export I_MPI_DEBUG=1
export I_MPI_FABRICS=tmi
export TMI_CONFIG=${HOME}/etc/tmi.conf

cd $SLURM_SUBMIT_DIR
srun ./mpi.x
```

Merci !

marc_mendez_bermond@dell.com