Mémento IDRIS

Guide pour Linux, module, SLURM & IDRIS

Voir aussi FAQ IDRIS : http://www.idris.fr/faqs/

Accéder à JEAN ZAY - HPE SGI 8600

Connexion à partir d'un terminal
ssh *login*@jean-zay.idris.fr

`PuTTY`	client SSH pour Windows
`Xming/XQuartzX11`	serveur X11 pour Windows/Mac

Transfert de fichiers entre JEAN ZAY et votre machine locale
sftp *login*@jean-zay.idris.fr destination
sftp> put *localSource JZDestination*
sftp> get JZSource localDestination

scp localSource *login*@jean-zay.idris.fr:*JZDestination*
scp *login*@jean-zay.idris.fr:JZSource localDestination

Pour des transferts de gros fichiers (plus de 2GB)
bbftp -e 'put localSource *JZDestination*' -u login -s jean-zay.idris.fr
bbftp -e 'get JZSource localDestination' -u login -s jean-zay.idris.fr

`FileZilla`	Client SFTP pour Linux, Windows, Mac
`WinSCP`	Client SFTP/SCP pour Windows
`BBFTP In2p3`	Client BBFTP pour Linux, Mac

Gestion des fichiers Linux

`man` *command*	Affiche le manuel de la commande
`pwd`	Détaille le chemin du répertoire courant
`cd` *dir*	Change le répertoire courant à *dir*
`~ (tilde)`	Répertoire *home*
`. (point)`	Répertoire courant
`.. (double point)`	Répertoire parent
`ls`	Liste les fichiers dans le répertoire courant
`ls -lh`	Liste les fichiers en format long
`ls -a`	Ajoute à la liste les fichiers cachés
`rm` *file*	Efface le fichier *file*
`mkdir` *dir*	Crée un répertoire vide *dir*
`rm -r` *dir*	Efface le répertoire dir et **tout son contenu**
`cp` *file1 file2*	Copie *file1* dans *file2*
`cp` *file dir*	Copie le fichier *file* dans le répertoire *dir*
`mv` *file1 file2*	Renomme le fichier *file1* en *file2*
`mv` *file dir*	Déplace le fichier *file* dans le répertoire *dir*
`wget` *URL*	Charge un fichier d'une *URL* sur Internet Seulement accessible à partir d'une frontale
`git clone` *URL*	Clone un depôt git - Seulement accessible à partir d'une machine frontale
`unzip` *file*`.zip` `zip -r` *file*`.zip` *dir*	Extrait / Compresse un fichier ZIP
`tar -xzf` *ftgz* `tar -czf` *ftgz dir*	Extrait / Compresse un gz tarball (*ftgz*) compressé
`chmod`	Change les permissions *read/write/execute*
`setfacl` *ACLOPT dir*	Définit un *Access Control List* pour *dir*
`which` *cmd*	Affiche le chemin complet d'une commande
`whereis` *cmd*	Localise le binaire, la source et le manuel
`du -hdl` *dir*	Indique la taille des répertoires courant et enfants
`du --inodes -dl` *dir*	Indique la quantité d'*inodes* des répertoires courant et enfants
`find`	Trouve des fichiers dans un répertoire
`tree`	Affiche un arbre des répertoires enfants

Informations système

`hostname`	Donne le nom de la machine ou du nœud
`date`	Affiche l'heure et la date du système
`top`	Liste l'utilisation processeur et mémoire
`htop`	Affiche une version augmentée de *top*
`watch nvidia-smi`	Donne les compteurs d'utilisation des GPU

Affichage et édition de fichier

`cat` *file*	Affiche la totalité du fichier texte
`more` *file*	Permet le défilement du fichier texte
`less` *file*	Affiche une version optimisée de *more*
`head` *file*`/tail` *file*	Affiche le début et la fin du fichier
`emacs`	Editeur de texte personnalisable
`vim`	Editeur de texte minimaliste
`nano`	Editeur de texte simple

Alias et variables système

`alias`	Remplace dynamiquement une commande avec une autre
`env`	Liste les variables système d'environnement
`export` *var*`=`*val*	Crée une variable d'environnement *$var* avec la valeur *val*
`echo`	Affiche une valeur à l'écran
`echo $`*var*	Affiche la variable d'environnement *var*

Redirection des entrées/sorties

`$(`*command*`)`	Lance d'abord la commande dans $() , puis donne la sortie au reste de la commande. Identique à `` `command` `` .
`<`	Redirection vers l'entrée standard
`>`	Redirection vers la sortie standard
`2>`	Redirection vers les erreurs standards
`2>&1`	Redirection vers les erreurs et la sortie
cmd1 `|` *cmd2*	Commande *pipe*; redirige la sortie de commande *cmd1* vers l'entrée de *cmd2*
`tee` *file*	Lit l'entrée standard, et l'écrit dans la sortie standard et dans *file*, ex: *echo "0"	tee file.txt*
`eval` *args*	Exécute *args* comme une commande *shell*

Filtres

`wc`	Compte des mots
`grep`	Filtre par *expression régulière*
`sort`	Trie l'entrée
`uniq`	Filtre les lignes dupliquées
`cut`	Coupe certains champs ou colonnes
`sed`	*Stream editor* - recherche et remplacement
`awk`	Pour un filtrage complexe en script

Gestion des modules

`module avail` *firstLetterName*	Donne tous les modules dont le nom commence par ...
`module show` *name*	Affiche les informations du module
`module load` *name*	Charge le module
`module list`	Liste les modules chargés
`module unload` *name*	Ôte le module de l'environnement
`module purge`	Efface tous les modules chargés

SLURM

`squeue -lu $USER`	Liste vos jobs dans la queue
`squeue -u $USER --start`	Liste les jobs en court d'exécution et l'heure de démarrage estimée
`scontrol show job` *jobid*	Donne les infos de gestion d'un job
`scontrol show node` *nodeid*	Donne les infos de gestion d'un nœud
`scancel` *jobid*	Annule le job *jobid*
`sinfo`	Affiche les infos système
`sacct`	Afiche les informations de compte
`sbatch` *filename*	Soumet un script *batch*
`salloc`	Obtient une allocation interactive
`srun` *execfile*	Obtient une allocation et exécute une application

Directives SBATCH

`--job-name=`*name*	Nom du job
`--account=`*my_project@dev*	Compte à utiliser. Le champ *dev* peut être *cpu* ou *gpu*
`--partition=`*name*	Partition du système à utiliser
`--qos=`*name*	*Quality of service*
`--time=`*min*	Temps limite; soit au format *min* soit au format *dd-hh:mm:ss*
`--nodes=`*count*	Nombre de nœuds
`--ntasks-per-node=`*count*	Nombre de processus par nœud
`--ntasks=`*count*	Nombre total de processus
`--cpus-per-task=`*count*	Cœurs CPU par processus
`--gres=gpu:`*count*	Nombre de GPU (par nœud)
`--hint=nomultithread`	Désactive l'*hyperthreading*
`--exclusive`	Alloue des nœuds en mode exclusif
`--output=`*file*	Sortie standard; par défaut *slurm-jobid.out* si rien n'est spécifié
`--error=`*file*	Ecrit les erreurs standards dans *file*
`--array=`*arrayspec*	Soumet une collection de jobs similaires
`--dependency=`*state*`:`*job*	Dépendance entre jobs pour l'exécution en multi-étapes ou en cascade. ex pour *state* : *afterok*
`--mail-user=`*email*	Envoie par e-mail les alertes
`--mail-type=`*type*	Type des alertes e-mail: BEGIN, END, FAIL, REQUEUE, ALL
`-C, --constraint=`*feature*	Demande un nœud avec une contrainte spécifique

Variables d'environnement SLURM

`$SLURM_JOBID`	Job ID - identifiant
`$SLURM_JOB_NODELIST`	Noms des nœuds alloués au job
`$SLURM_NNODES`	Nombre de nœuds alloués au job
`$SLURM_JOB_CPUS_PER_NODE`	Cœurs CPU par nœud alloué au job
`$SLURM_NTASKS`	Nombre de tâches allouées au job
`$SLURM_SUBMIT_DIR`	Répertoire de soumission du job
`$SLURMD_NODENAME`	Nom du nœud exécutant le script du job
`$SLURM_PROCID`	Le rang MPI du processus courant
`$SLURM_ARRAY_JOB_ID`	Job ID maître dans le Job Array
`$SLURM_ARRAY_TASK_ID`	Task ID dans le Job Array
`$SLURM_ARRAY_TASK_COUNT`	Nombre de tâches du Job Array
`$SLURM_ARRAY_TASK_MIN`	Min. Task ID dans le Job Array
`$SLURM_ARRAY_TASK_MAX`	Max. Task ID dans le Job Array

Commandes IDRIS

`idrdoc`	Affiche des informations *système* utiles
`idrquota`	Affiche les quotas disques, *-m* | *-w* | *-s* *(home/work/store)*
`idracct`	Indique la consommation d'heure CPU et GPU
`idrproj`	Affiche les projets et change le projet par défaut
`idrenv`	Affiche la liste des variables d'environnement IDRIS
`idrjup`	Lance Jupyter Notebook sur le nœud courant
`idrlab`	Lance Jupyter Lab sur le nœud courant
`idr_compuse`	Vérifie l'état de consommation du projet

Commandes utiles

`time ./my_exe`	Retourne le temps d'éxécution d'un script
`eval $(idrenv -d projet)`	Force immédiatement le changemet du projet actif
`ssh nodeid nvidia-smi -l 1`	Affiche à partir d'une machine frontale, l'utilisation GPU pour un nœud spécifique.
`ssh nodeid` `[nodeid ~]$ htop`	Affiche l'utilisation processeur et mémoire d'un nœud spécifique
`sacct -A acc@gpu -S MMDDYY -E MMDDYY --format Elapsed,Submit,NNodes,JobID,NCPUS,NTasks,AllocGRES,User`	Affiche l'historique des jobs pour un compte entre la date de début (-S) et la date de fin (-E)
`srun --pty --sbatch_options bash`	Permet l'obtention d'un terminal sur un nœud de calcul (pour par exemple lancer Jupyter Notebook ou Jupyter Lab)

Outils (disponibles à partir de *module load*)

`idrmem`	Affiche la consommation maximale en mémoires virtuelle et physique d'un code MPI
`vtune`	Outil de profilage Intel
`map`	Outil de profilage Arm
`ddt`	Outil de débogage Arm
`scalasca`	Outil de profilage MPI et OpenMP

Exemples de scripts Sbatch

MPI/OpenMP mixés en job parallèle

```
#!/bin/bash
#SBATCH --job-name=Hybride                # nom du job
#SBATCH --ntasks=8            # Nombre de processus MPI
#SBATCH --cpus-per-task=10  # nombre de threads OpenMP
#SBATCH --hint=nomultithread     # pas d'hyperthreading
#SBATCH --time=00:10:00         # Temps d'exécution max
#SBATCH --output=Hybride%j.out       # fichier de sortie
#SBATCH --error=Hybride%j.out         # fichier d'erreur

# on se place dans le répertoire de soumission
cd ${SLURM_SUBMIT_DIR}

# nettoyage des modules charges en interactif et
herites par defaut
module purge

# chargement des modules
module load intel-all/19.0.4

# echo des commandes lancées
set -x

# nombre de threads OpenMP
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
# binding OpenMP
export OMP_PLACES=cores

# exécution du code
srun ./exec_mpi_omp
```

Job Multi-GPU Multi-nœuds

```
#!/bin/bash
#SBATCH -job-name=multi_gpu_multicore     # nom du job
#SBATCH --partition=gpu_p13     # partition GPU choisie
#SBATCH --ntasks=8    # nbr de tache MPI  (= nbr de GPU)
#SBATCH --ntasks-per-node=4    # nbr de tache par nœud
#SBATCH --gres=gpu:4              # nbr de GPU par nœud
#SBATCH -cpus-per-task=10       # nbr de CPU par tache
#SBATCH --hint=nomultithread    # pas de hyperthreading
#SBATCH --time=00:10:00         # temps d'execution max
#SBATCH --output=multi_gpu_mpi%j.out    # fichier sortie
#SBATCH --error=multi_gpu_mpi%j.out  # fichier d'erreur

# nettoyage des modules charges en interactif et
herites par defaut
module purge

# chargement des modules
module load ...

# echo des commandes lancees
set -x

# execution du code
srun ./executable_multi_gpu_mpi_cuda-aware
```

Job Multi-GPU Mono-nœud pour l'IA

```
#!/bin/bash

#SBATCH --job-name="GTSRB Full conv."     # nom du job
#SBATCH --ntasks=4    # nombre de tâche (= nbr de GPU)
#SBATCH --gres=gpu:4          # nombre de GPU par noeud
#SBATCH --cpus-per-task=10  # nbr de cœurs à réserver
#SBATCH --hint=nomultithread         # cœurs physiques
#SBATCH --time=03:00:00       # temps exécution maximum
#SBATCH --output="_batch/G.out"      # fichier de sortie
#SBATCH --error="_batch/G.err"        # fichier d'erreur
#SBATCH --mail-user=mail@domaine
#SBATCH --mail-type=ALL

MODULE_ENV="tensorflow-gpu/py3/2.2.0"
RUN_DIR="$WORK/fidle/GTSRB"
RUN_SCRIPT="./run/full_convolutions.py"

# ---- Welcome...
echo '----------------------------------------'
echo "Start : $0"
echo '----------------------------------------'
echo "Job id         : $SLURM_JOB_ID"
echo "Job name       : $SLURM_JOB_NAME"
echo "Job node list  : $SLURM_JOB_NODELIST"
echo '----------------------------------------'
echo "Script         : $RUN_SCRIPT"
echo "Run in         : $RUN_DIR"
echo "With env.      : $MODULE_ENV"
echo '---------    --------------------------'
# ---- Module

module purge
module load "$MODULE_ENV"

# ---- Run it...
#
cd "$RUN_DIR"
srun python "$RUN_SCRIPT"
```

Espaces disques

`$HOME`	Répertoire d'accueil destiné aux fichiers de configuration, 3Go / 150k inodes, sauvegardé
`$WORK`	Destiné aux codes sources et aux données d'entrée/sortie, 5To/500k inodes, sauvegardé
`$SCRATCH`	Disque SSD, 2.2Po partagé par tous les utilisateurs, nettoyage régulier, non sauvegardé
`$JOBSCRATCH`	Disque SSD, répertoire éphémère le temps de l'exécution d'un job, non sauvegardé
`$STORE`	Disque d'archivage à long terme, 50To / 100k inodes, non sauvegardé

Disques partagés par tous les membres d'un projet

`$ALL_CCFRWORK`	Disque *WORK* commun
`$ALL_CCFRSCRATCH`	Disque *SCRATCH* commun
`$ALL_CCFRSTORE`	Disque *STORE* commun

Disque public

`$DSDIR`	Pour les bases de données publiques en IA

Partitions

`cpu_p1`	1528 nœuds CPU 40-cœurs (par défaut pour CPU)

`gpu_p13`
`v100-32g`
`v100-16g`
	par défaut pour les jobs GPU
	261 nœuds 4-GPU(32Go) 40-cœurs
	351 nœuds 4-GPU(16Go) 40-cœurs

`gpu_p2`
`gpu_p2l`
`gpu_p2s`
	Réservée à l'IA, 31 nœuds 8-GPU(32Go) 24-cœurs
	Sous-partition 11 nœuds avec 768 Go de RAM CPU
	Sous-partition 20 nœuds avec 384 Go de RAM CPU

`prepost`	4 nœuds de traitement pre/post – limitée à 20h – non facturée en heure
`visu`	5 nœuds de visualisation – limitée à 1h - non facturée en heure

Quality of Service

`qos_cpu-t3`	QoS CPU (défaut)	20h, 512 nœuds par job, 1200 nœuds par user
`qos_cpu-t4`	QoS travaux longs CPU	100h, 4 nœuds par job, 32 nœuds par *user*, 128 par qos
`qos_cpu-dev`	QoS Dev. CPU	2h, 128 nœuds par job, 128 nœuds par user, 1200 par qos
`qos_gpu-t3`	QoS GPU (défaut)	20h, 512 GPU par job, 1024 GPU par user
`qos_gpu-t4`	QoS travaux longs GPU	100h, 16 GPU par job, 128 GPU par *user*, 512 GPU par qos
`qos_gpu-dev`	QoS Dev. GPU	2h, 32 GPU par job, 32 GPU par *user*, 512 GPU par qos