Working on the Jean Zay machine

Table of Content

Last Update 10/10/2019

1. Introduction

- On-line users manual: ■http://www.idris.fr/eng/jean-zay
- Jean-Zay computing nodes: the nodes of CPU partition have 40 cores each.
  - Intel Cascade Lake nodes for regular computation
  - Partition name: **cpu_p1**
  - CPUs: 2x20-cores Intel Cascade Lake 6248 @2.5GHz
  - Cores/Node: 40
  - Nodes: 1 528
  - Total cores: 61120
  - RAM/Node: 192GB
  - RAM/Core: 4.8GB
- Jean-Zay post-processing nodes : xlarge are free and useful for post-processing operations.
  - Fat nodes for computation requiring a lot of shared memory
  - Partition name: **prepost**
  - CPUs: 4x12-cores Intel Skylake 6132@3.2GHz
  - GPUs: 1x Nvidia V100
  - Cores/Node: 48
  - Nodes: 4
  - Total cores: 192
  - RAM/Node: 3TB
  - RAM/Core: 15.6GB

2. Job manager commands

- `sbatch job` -> submit a job
- `scancel ID` -> kill the job with the specified ID number
- `sacct -u login -S YYYY-MM-DD` -> display all jobs submitted by login, add `-f` to see full job name
- `squeue` -> display all jobs submitted on the machine.
- `squeue -u $(whoami)` ->display your jobs.

3. Suggested environment

3.1. General environment

Before working on Jean Zay you need to prepare your environment. This is important to do before compilation to ensure the use of same modules as done by libIGCM running environment. We propose you a bash_login file which you can copy from the work commun psl. Copy it to your home, rename it by adding a dot as prefix. You can add personal settings in your .bashrc_login. Do as follow:

```
cp $WORK/../../psl/commun/MachineEnvironment/jeanzay/bash_login ~/.bashrc
```

After re-connexion or source of .bash_login, check your loaded modules for intel, netcdf, mpi, hdf5 needed for the compilation:

```
module list
Currently Loaded Modulefiles:
 1) intel-compilers/19.0.4                5) netcdf/4.7.0/intel-19.0.4-mpi           9) ferret/7.2/gcc-9.1.0
 2) intel-mpi/19.0.4                      6) netcdf-fortran/4.4.5/intel-19.0.4-mpi  10) subversion/1.9.7/gcc-4.8.5
 3) intel-mkl/19.0.4                      7) nco/4.8.1/gcc-4.8.5                    11) cdo/1.9.7.1/intel-19.0.4
 4) hdf5/1.10.5/intel-19.0.4-mpi          8) ncview/2.1.7/intel-19.0.4-mpi
```

The modules are specified in the file $WORK/../../psl/commun/MachineEnvironment/jeanzay/env_jeanzay which is sourced in bash_login. The same file env_jeanzay is sourced in libIGCM.

Create ~/.forward file in your main home containing only one line with your email address to receive emails from libIGCM. -- actually doesn't work (2019/11/28)

4. Example of a job to start an executable in a Parallel environnement

4.1. MPI

Here is an example of a simple job to start an executable orchidee_ol (or gcm.e commented). The input files and the executable must be in the directory before starting the executable.

```
#!/bin/bash
#SBATCH --job-name=TravailMPI      # name of job
#SBATCH --ntasks=80                # total number of MPI processes
#SBATCH --ntasks-per-node=40       # number of MPI processes per node
# /!\ Caution, "multithread" in Slurm vocabulary refers to hyperthreading.
#SBATCH --hint=nomultithread       # 1 MPI process per physical core (no hyperthreading)
#SBATCH --time=00:10:00            # maximum execution time requested (HH:MM:SS)
#SBATCH --output=TravailMPI%j.out  # name of output file
#SBATCH --error=TravailMPI%j.out   # name of error file (here, in common with output)

# go into the submission directory
cd ${SLURM_SUBMIT_DIR}


# echo of launched commands
set -x

# code execution
srun ./orchidee_ol
#srun ./gcm.e
```

4.2. Hybrid MPI-OMP

```
#!/bin/bash
#SBATCH --job-name=Hybrid          # name of job
#SBATCH --ntasks=8             # name of the MPI process
#SBATCH --cpus-per-task=10     # number of OpenMP threads
# /!\ Caution, "multithread" in Slurm vocabulary refers to hyperthreading.
#SBATCH --hint=nomultithread   # 1 thread per physical core (no hyperthreading)
#SBATCH --time=00:10:00            # maximum execution time requested (HH:MM:SS)
#SBATCH --output=Hybride%j.out     # name of output file
#SBATCH --error=Hybride%j.out      # name of error file (here, common with the output file)

# go into the submission directory
cd ${SLURM_SUBMIT_DIR}


# echo of launched commands
set -x

# number of OpenMP threads
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
# OpenMP binding
export OMP_PLACES=cores

# code execution
srun ./lmdz.e
```

4.3. MPMD

5. JeanZay job headers

Here is an example of a job header as generated by libIGCM on the JeanZay machine:

```
#####################
## JEANZAY    IDRIS ##
#####################
#SBATCH --job-name=MY-SIMULATION
#SBATCH --output=Script_Output_MY-SIMULATION.000001
#SBATCH --error=Script_Output_MY-SIMULATION.000001
#SBATCH --ntasks=443
#SBATCH --cpus-per-task=8
#SBATCH --hint=nomultithread
#SBATCH --time=00:30:00
#SBATCH --account gzi@cpu
```

Details are as follows:

Control	Keyword	Argument	Example	Comments
Job name	`--job-name`	string	`#SBATCH --job-name=Job_MY-SIMULATION`	
Standard output file name	`--output`	string	`#SBATCH --ouput=Script_Output_MY-SIMULATION.000001`	
Error output file name	`--error`	string	`#SBATCH --error=Script_Output_MY-SIMULATION.000001`	
Number of MPI tasks	`--ntasks`	integer	`#SBATCH --ntasks=443`	
Number of OpenMP threads	`--cpus-per-task`	integer	`#SBATCH --cpus-per-task=8`	
To allocate one thread per physical core	`--hint`	`nomultithread`	`#SBATCH --hint=nomultithread`	"Multithread" does indeed refer to hyperthreading for Slurm.
Wall-time (maximum time allowed for execution)	`--time`	date HH:MM:SS	`#SBATCH --time=24:00:00`	
Account used	`--account`	string	`#SBATCH --account=myaccount@cpu`	