Working on the Irene-amd machine

Table of Content

1. Connexion

```
ssh login@irene-amd-fr.ccc.cea.fr
```

2. Introduction

On-line users manual: ■https://www-tgcc.ccc.cea.fr/docs/irene.info.pdf (you will need a TGCC login and password)

- Partition name : Rome
- CPUs: 2x64 AMD Rome@2.6Ghz (AVX2)
- Core/Node: 128
- Nodes: 2292
- Total core: 293376
- RAM/Node: 256GB
- RAM/core : 2GB

- When submitting a job through ccc_msub or ccc_mprun, you must specify -m work, -m scratch, -m store, or combine them like in -m work,scratch. This constraint has the advantage that your jobs won't be suspended if a file system you don't need becomes unavailable. This is done in all jobs in libIGCM.
- The default time limit for a job submission is 2 hours (7 200 s).
- Irene post-processing nodes : xlarge are free and useful for post-processing operations.
    - Fat nodes for computation requiring a lot of shared memory (coming from irene.info)
    - Partition name: *xlarge*
    - CPUs: 4x28-cores Intel Skylake@2.1GHz
    - GPUs: 1x Nvidia Pascal P100
    - Cores/Node: 112
    - Nodes: 5
    - Total cores: 560
    - RAM/Node: 3TB
    - RAM/Core: 5.3GB
    - IO: 2 HDD de 1 TB + 1 SSD 1600 GB/NVMe

3. Job manager commands

- `ccc_msub job` -> submit a job
- `ccc_mdel ID` -> kill the job with the specified ID number
- `ccc_mstat -u login` -> display all jobs submitted by login, add `-f` to see full job name
- `ccc_mpp` -> display all jobs submitted on the machine. `ccc_mpp -n` to avoid colors.
- `ccc_mpp -u $(whoami)` ->display your jobs.

4. Suggested environment

4.1. General environment

Before working on Irene you need to prepare your environment. We propose you 2 files which you can copy from the home igcmg. The first one called **bashrc** will source the second called **bashrc_irene**. Copy both files to your home, rename them by adding a dot as prefix. You can add personal settings in your .bashrc_irene. Do as follow:

```
cp ~igcmg/MachineEnvironment/irene/bashrc ~/.bashrc
cp ~igcmg/MachineEnvironment/irene/bashrc_irene ~/.bashrc_irene
```

The .bashrc will source your own .bashrc_irene which must be in your home.

After re-connexion or source of .bashrc_irene, check your loaded modules:

```
>> module list
Currently Loaded Modulefiles:
1) ccc/1.0                     4) c/gnu/8.3.0           7) licsrv/intel      10) fortran/intel/20.0.0    13) feature/mkl/si
2) flavor/buildcompiler/gcc/8  5) fortran/gnu/8.3.0     8) c++/intel/20.0.0  11) feature/mkl/lp64        14) feature/mkl/ve
3) c++/gnu/8.3.0               6) gnu/8.3.0             9) c/intel/20.0.0    12) feature/mkl/sequential  15) mkl/20.0.0
```

The modules are specified in the file ~igcmg/MachineEnvironment/irene/env_atlas_irene which is sourced in bashrc_irene. In this environment we are loading and using module for data post-treatment. The environment need for the compilation and the execution of simulation will be load by libIGCM (it will be done with the file ~igcmg/MachineEnvironment/irene/env_irene).

Create ~/.forward file in your main home containing only one line with your email address to receive emails from libIGCM.

5. How the storage project is set by libIGCM

When you use libIGCM it is recommended to dedicate one modipsl/libIGCM to one project allocation. By default, the output folders IGCM_OUT will be created in the directories $CCCSCRATCHDIR, $CCCWORKDIR and $CCCSTOREDIR corresponding to the project used in the main job. It is important that the same project is used in the post-processing jobs in libIGCM.

For gencmip6 project, you have to set a subproject for computing and gencmip6 is forced for all directories.

If you need to use another project for the computing than the storage, it is possible to set the variable **DataProject** in config.card UserChoices section, for example DataProject=gen6328, read more [here](). This project will be used for all output directories for the computing job and post-processing jobs even if they have another project for computing in the headers. The variable **DataProject** can also be used if you work with different project allocations in the same modipsl. Only exception (harmless) is the first RUN_DIR folder which is always created in the $CCCSCRATCHDIR corresponding to the dfldatadir loaded in when submitting main job. When the job resubmits itself, the RUN_DIR will be in the same project space as the rest of the output. # Example of ins_job

```
> ./ins_job
...
Wait for the next question ...
Hit Enter or give project ID (default is gen0000), possible projects are gen1111 gen2222 ... or other xxxcmip6 : aaacmip6
gen0000 (RETURN)
ProjectID is gen0000 at Irene
Hit Enter or give TYPE OF NODE required for post-processing (default is "xlarge"), possible types of nodes are "skylake" o
(RETURN)
ProjectNode for post-processing is xlarge at Irene
Hit Enter or give NUMBER OF CORES required for post-processing (default is "8")
possible numbers of cores are "1" to "112" for xlarge :
(RETURN)
ProjectCore for post-processing is 8
Wait for the next question ...
Hit Enter or give project ID (default is gen0000), possible projects are gen1111 gen2222 ... or other xxxcmip6 : aaacmip6
(RETURN)
PostID is gen0000 at Irene on xlarge for post-processing
```

6. File systems used on Irene

A figure to illustrate Irene filesystems is available [here]()

7. Irene job headers

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

```
#####################
## IRENE   TGCC/CEA ##
#####################
#MSUB -r MY-SIMULATION
#MSUB -o Script_Output_MY-SIMULATION.000001
#MSUB -e Script_Output_MY-SIMULATION.000001
```

```
#MSUB -eo
#MSUB -n 976
#MSUB -x
#MSUB -T 86400
#MSUB -A dekcmip6
#MSUB -q rome
#MSUB -m store,work,scratch
```

The detail is as follows:

Control	Keyword	Argument	Example	Comments
Job name	`-r`	string	`#MSUB -r Job_MY-SIMULATION`	The string should not contain the underscore (_) character.
Standard output file name	`-o`	string	`#MSUB -o Script_Output_MY-SIMULATION.000001`	
Error output file name	`-e`	string	`#MSUB -e Script_Output_MY-SIMULATION.000001`	If both `-o` and `-e` names are the same, the two outputs will be merged. The `-eo` option of the example is as a matter of fact redundant with that.
Number of MPI tasks allocated	`-n`	integer	`#MSUB -n 976`	This is for the SPMD case; in the MPMD case, the integer is the number of cores.
Node exclusivity	`-x`	none	`#MSUB -x`	
Wall-time (maximum time allowed for execution)	`-T`	integer	`#MSUB -T 86400`	Time is expressed in number of seconds.
Project allocation	`-A`	string	`#MSUB -A dekcmip6`	
Partition used	`-q`	string	`#MSUB -q rome`	Choice is amongst `rome`
Visible spaces	`-m`	(comma-separated) string(s)	`#MSUB -m store,work,scratch`	Specific to TGCC. The job can only "see" spaces specified on this line. Possible spaces are amongst `store` and/or `work` and/or `scratch`.
Job priority	`-U`	string	`#MSUB -U high`	Possible arguments are `high`, `medium` or `low`. (This command does not appear by default on jobs generated by libIGCM.)

8. Use of specific options to increase computing performances

Following options have to be used with caution. Please contact the person in charge of the configuration you use before launching production runs with these options.

8.1. Increase the number of cores per MPI/OpenMP process

Because of the architecture of Irene-amd processor, you could be limited by the memory available per core as well as the memory bandwidth. In order to use more cores per MPI process (or OpenMP thread), you can specify the factor you want to increase this number of cores in config.card by using **DEP** keyword as follows :

```
ATM= (gcm.e, lmdz.x, 71MPI, 8OMP, 2DEP)
SRF= ("" ,"" )
SBG= ("" ,"" )
OCE= (opa, opa.xx, 360MPI, 2DEP)
ICE= ("" ,"" )
MBG= ("" ,"" )
CPL= ("", "" )
IOS= (xios_server.exe, xios.x, 12MPI, 2DEP)
```

In this example, the number of cores will be increased by a factor 2 for each component.

8.2. Use of dedicated nodes for XIOS servers

Some preliminary tests (bad performances) drive us to investigate the use of dedicated nodes for XIOS servers. To activate this functionality, you have to modify your config.card by using "DEDICATED" keyword as follows :

```
ATM= (gcm.e, lmdz.x, 71MPI, 8OMP, 2DEP)
SRF= ("" ,"" )
SBG= ("" ,"" )
OCE= (opa, opa.xx, 360MPI, 2DEP)
ICE= ("" ,"" )
MBG= ("" ,"" )
CPL= ("", "" )
IOS= (xios_server.exe, xios.x, 12MPI,2DEP, DEDICATED)
```

Note that this functionality is only available for IOS component.