ABINIT
How to work on a supercomputer: Environment and job submission

Example of the *Cobalt* supercomputer at TGCC (french computing center)

Content

First step: log on to the supercompter.
In a terminal window, enter : **ssh -Y cobalt.ccc.cea.fr**

At any time, you can type the **machine.info** *command to access to the computer user's guide.*

1.1. The "module" command

The *Cobalt* supercomputer use Environment Modules. The Environment Modules package provides for the dynamic modification of a user's environment via modulefiles.

To familiarize yourself with this environment, you should first list the available modules with the command:

module avail

The module that are used now in your environment are given by the command: ***module list***
First, lets clean your session by unloading all modules in memory:

module purge

Then, to use the ABINIT module, you just need to use the simple command:

module load abinit/8.12.0-beta

All software mandatory for ABINIT execution (dependendies) will be automatically loaded. You will have automatically abinit in your path (try it by typing *abinit --version*!).

1.2. The batch script

On a supercomputer you cannot execute your ABINIT calculation directly by a simple command like *abinit< files > log* or *mpirun -n 200 abinit< files > log* (for a parallel calculation). Job submissions and resources allocation are managed by a specific environment. Special commands prefixed by **ccc_** are provided to execute these operations (this is specific to the TGCC computing center).

First, you need to write a script which will define a set of directives about your job and how to execute it. Then submit your job on a given batch queue. Here is a typical script which you can write in a file called e.g. job.ls:

```
#!/bin/bash
#MSUB -r MyJob                      # Name of the job (to be changed)
#MSUB -n 50                         # Number of MPI processes to use
#MSUB -c 1                          # Number of tasks per process to use
#MSUB -T 60                         # Time limit in seconds
#MSUB -o ls_%I.o                    # Standard output. %I is the job id
#MSUB -e ls_%I.e                    # Error output.    %I is the job id
#MSUB -A dam00000                   # For the ABINIT school only
#MSUB -E '--reservation=ABINIT01'   # For the ABINIT school only
set -x                              # list the commands during execution

export OMP_NUM_THREADS=1            # Same number as in the "MSUB -c" line above

ls > output                         # Execute job ls
```

Note that the first 7 lines are commented with one #, so that they are not a part of the shell script. However the supercomputer system read them and understand them.
The first 6 lines define the name of the task, the number of processors to use, the time limit of the job, the standard output and error files.
Note: the number of processors used is equal to the number of processes times the number of tasks per process.

The 8th and 9th lines are specific to the ABINIT school:

- dam00000 is the project number of the school,
- ABINIT01 is the name of the set of CPU cores reserved for the school (change it to ABINIT02, ABINIT03... on next days).

The rest of the file contains usual script commands in the bash langage. To execute this file, just launch:

ccc_msub job.ls

Various tools provides you information about your job. Just try them:

- **ccc_mpp** provides you information about the running of your job and gives you the BatchID of your job.
  Note: *job.ls* run too fast to allow to see any information.
- **ccc_mpeek** is useful to have information about the job.
- **ccc_mdel** can be used to kill it during execution.

After the job is executed, you will find two new files in the current directory corresponding to the standard and error output files. You can have a look to these two files to see if the job executed correctly.

1.3. Handling of files

Working on a supercomputer is qualitatively not different from working on a desktop computer. However, quantitatively, it is completely different: As several processors are working at the same time, they might write a huge number of files on the working directory. For his reason, it is all the more important to think a little bit about a correct handling of files. In particular some *filesystems* are dedicated to supercomputing and other are not.

In particular, you must not execute ABINIT and produce results on your HOME, you must do it on the $CCCSCRATCHDIR or $CCCWORKDIR which are fast I/O file systems.

Let's adopt a clean way to handle files and work in the $CCCWORKDIR file system.

1.4. Trying ABINIT

Here is a typical script; you can copy it in a file called e.g. *job.abinit*:

```
#!/bin/bash
#MSUB -r MyJob                    # Name of the job (to be changed)
#MSUB -n 8                        # Number of MPI processes to use
#MSUB -c 1                        # Number of tasks per process to use
#MSUB -T 60                       # Time limit in seconds
#MSUB -o ls_%I.o                  # Standard output. %I is the job id
#MSUB -e ls_%I.e                  # Error output.    %I is the job id
#MSUB -A dam00000                 # For the ABINIT school only
#MSUB -E '--reservation=ABINIT01' # For the ABINIT school only
set -x                            # list the commands during execution

export OMP_NUM_THREADS=1          # Same number as in the "MSUB -c" line above

module load abinit/8.12.0-beta    # Load ABINIT module (beta version fot this school)

cd $CCCWORKDIR/abinit-8.10.2/tests/tutorial/Input  # Enter the tutorial directory (optional line)
mkdir -p Work ; cd Work                            # Working directory for the tutorial (optional line)

ccc_mprun  abinit < ../tbase3_x.files > log        # Execute ABINIT in parallel
```

Read the script and try to understand all the steps...
Then create an new file named job.abinit.

To use this script, you just have to launch the following command:

- **ccc_msub job.abinit**

The job will execute ABINIT in parallel.

Have nice tutorials !

For more information about Cobalt environnement, type ***machine.info*** in a terminal...
