Calcul en environnement Linux au Latmos et IPSL MesoCentre

I introduction

L'environnement Linux pour le calcul choisi au latmos en 2014 est la distribution Scientific Linux 6.x
Elle sera déployée petit à petit sur tous les postes fixes linux du laboratoire en remplacement des Mandriva et Mageia ainsi que sur les serveurs de calculs.
C'est aussi la distribution et l'environnement choisi au LOCEAN,à l'IPSL et au LMD Jussieu
ainsi que sur les clusters de calcul du mésocentre de l'IPSL (ClimServ et CICLAD) avec les mêmes outils et version.

Le guide est donc totalement applicable aussi bien au LATMOS que sur le Mésocentre de l'IPSL (CLIMSERV et CICLAD)

Ou faire des calculs sous linux au laboratoire ?

1) sur **son poste de travail** linux (principalement du travail interactif et graphique)
2) sur **les clusters du laboratoire** (foehn64 sur Jussieu et srv-calcul1 sur guyancourt)
   Possibilité de travail interactif et batch , plus grande capacité mémoire vive
   quelques dizaines de processeurs au maximum
3) sur les **clusters du mesocentre de l'IPSL**
   (plusieurs centaines de cœurs , Plusieurs centaines de TeraOctets)
   climserv http://climserv.ipsl.polytechnique.fr/
   ciclad http://ciclad-web.ipsl.jussieu.fr/
   Ce sont des moyens adaptés au travail sur d'importantes quantités de données qui bien souvent sont déjà stockés sur le mésocentre de l'IPSL donc qu'il n'est pas forcément nécessaire de dupliquer au laboratoire
   c'est aussi un bon endroit pour des tests de parallèlismes sur quelques dizaines de coeurs
4) le **centre de calcul de l'UPMC** (http://www.ics.upmc.fr/)
   il suffit d'être dans l'annuaire de l'UPMC pour obtenir un compte
   - Mesu Alpha SGI UV 2000 SMP 1024 coeurs /16T de ram (Grande mémoire partagée 2013)
   - Mesu Beta SGI ICE XA cluster de 84 nœuds (24 coeurs par noeud / 128Gb)
   accès aux deux via mesu.dsi.upmc.fr

   calcul parallèle sur un nombre plus important de coeur que sur le mesocentre de l'IPSL

5) les **centres de calculs nationaux** (plusieurs milliers de cœurs)
   IDRIS http://www.idris.fr/
   CCRT http://www-ccrt.cea.fr/fr/le_ccrt/actualites.htm
   TGCC http://www-hpc.cea.fr/fr/complexe/tgcc.htm

pour pouvoir travailler sur ces sites il faut que le projet fasse préalablement des demandes d'heure de calcul (il peut y avoir des conditions de nationalité pour les ouvertures de compte)

|II les logiciels disponibles

A la commande module

sur les serveurs un certain nombre d'outils sont disponibles en plusieurs version .
L'utilisateur peut choisir quel version parmi celle disponibles il désire utiliser. Ce choix se fait a travers la commande module

Syntaxe

```
module (avail [produit] | load produit[/version] | list | switch produit/version1 produit/version2 | display produit[/version] …)
```

- "avail" : liste les différents produits disponibles et leurs versions,
- "load" : charge le produit dans sa version par défaut (notée default), si aucune `version` n'est spécifiée,
- "list" : liste les différents produits chargés et leur version,
- "switch" : change la version d'un produit déjà chargé.
- "purge" : décharge tous les modules déjà chargés
- "display" : montre le contenu du fichier module

`produit` représente au choix :

- un compilateur,
- une bibliothèque,
- une application ou un utilitaire.

`version` représente les évolutions d'un même produit, elle peut être égale à :

- `default` : version par défaut; c'est celle qui est prise si vous ne spécifiez aucune version. Cette version est en général la plus adaptée,
- `numéro` : numéro de version complet du type X.Y.Z

Initialisation de la commande module

L'initialisation de `module` est faite implicitement dans les fichiers d'environnement général. Il n'y a donc rien à positionner pour avoir l'accès à la commande `module`.

Liste des produits disponibles

Vous cherchez un produit spécifique, une version particulière d'une bibliothèque ? L'avons-nous à votre disposition ? Pour répondre à cette question, la commande à exécuter est `module avail` :

```
$ module avail

---------------------------- /usr/share/Modules/modulefiles ----------------------------
dot         module-git  module-info modules     null        use.own

------------------------------ /etc/modulefiles/Compilers ------------------------------
gnu/4.4.7(default)   gnu/4.9.3     nagfor/5.3  nagfor/6.0(default)
pgi/2013(default) pgi/2011 intel/12.1.3.293(default)

------------------------------ /etc/modulefiles/Libraries ------------------------------
hdf5/1.8.10.patch1-gfortran netcdf4/4.2.1.1-pgf2013     openmpi/1.6.5-gfortran
hdf5/1.8.10.patch1-ifort    openmpi/1.4.5-gfortran      openmpi/1.6.5-ifort
hdf5/1.8.10.patch1-pgf2011  openmpi/1.4.5-ifort         openmpi/1.6.5-pgf2011
hdf5/1.8.10.patch1-pgf2013  openmpi/1.4.5-pgf2011       openmpi/1.6.5-pgf2013
netcdf4/4.2.1.1-gfortran    openmpi/1.4.5-pgf2011gcc    openmpi/1.6.5-pgfgcc
netcdf4/4.2.1.1-ifort       openmpi/1.4.5-pgf2013
netcdf4/4.2.1.1-pgf2011     openmpi/1.4.5-pgfgcc

------------------------------ /etc/modulefiles/Products -------------------------------
cdo/1.6.3(default)           idl/8.2                      python/2.7.6-canopy-1.3
ferret/6.7.2(default)        matlab/2010b.sp2             python/2.7-anaconda(default)
ferret/6.9                   matlab/2013b(default)        python/3.4-anaconda3
gmt/4.5.11(default)          ncl/6.1.2(default)           R/3.1.1
grads/2.0.2(default)         nco/4.3.7(default)           scilab/5.4.1(default)
idl/6.4(default)             python/2.7.3-epd7            scilab/5.5.1
```

exemples pour la commande module :

```
[weill@ciclad-ng ~]$ type pgf90
pgf90 est /opt/pgi-2013/linux86-64/2013/bin/pgf90
[weill@ciclad-ng ~]$ module load pgi/2011
[weill@ciclad-ng ~]$ type pgf90
pgf90 est /opt/pgi-2011/linux86-64/2011/bin/pgf90
[weill@ciclad-ng ~]$ type matlab
matlab est /opt/matlab-2013b/matlab
[weill@ciclad-ng ~]$ module load matlab/2010b.sp2
[weill@ciclad-ng ~]$ type matlab
matlab est /opt/matlab-2010b.sp2/matlab
[weill@ciclad-ng ~]$ module switch matlab/2013b
[weill@ciclad-ng ~]$ type matlab
matlab est /opt/matlab-2013b/matlab
[weill@ciclad-ng ~]$ module list
Currently Loaded Modulefiles:
  1) pgi/2011       2) matlab/2013b
[weill@ciclad-ng ~]$ module purge
[weill@ciclad-ng ~]$ type pgf90
-bash: type: pgf90 : non trouvé
[weill@ciclad-ng ~]$ module list
No Modulefiles Currently Loaded.
```

B python

sur foehn64 , srv-calcul1 et le mésocentre de l'ipsl (CICLAD , CLIMSERV)
il y a 5 versions de python disponibles

celle du system (/usr/bin/python python 2.6.6) que je déconseille dans le cadre d'une utilisation scientifique

```
weill@foehn64 ~> module avail python

----------------------/etc/modulefiles/Products-----------------------
python/2.7.3-epd7             python/2.7-anaconda(default)
python/2.7.6-canopy-1.3       python/3.4-anaconda3
```

Deux distributions viennent de la société enthought https://www.enthought.com/
la licence n'est pas libre et l'ajout de package est trop compliqué (voir impossible)
par contre elles possèdent beaucoup de packages par défaut. Nous les gardons sur les serveurs pour des raisons de compatibilité mais il n'y aura plus d'ajout de package
pour avoir la liste des packages pour ces deux distributions python :
pip freeze (115 package pour epd7 , 157 pour canopy)
aujourd'hui nous conseillons l'utilisation de la distribution python anaconda disponible avec python 2.7 ou bien python 3.4 (existe aussi pour windows et mac OS X)
https://store.continuum.io/cshop/anaconda/
Pour avoir la liste des packages installés avec anaconda sur nos linux :
conda list (pip freeze ne les donnes pas tous)

comment ajouter un package python dans anaconda quand on n'a pas les droits d'écriture dans la base

cette question, c'est toute la problématique de python lorsque l'on est sur une machine que l'on administre pas. Plutôt qu'installer l'intégralité de la distribution Anaconda dans sa home (environ 2 Go), il existe un mécanisme appelé Virtual Environment qui lui occupe au départ environ 24Mo et dans lequel vous allez pouvoir ajouter des packages

```
 ~> module load python/2.7-anaconda
 ~> conda create -n my_python --clone /opt/anaconda
```

Cela crée un répertoire .conda dans votre home qui fait au départ 24 Mo il grossira uniquement en fonction des packages que vous ajouterez à votre installation
pour l'utiliser vous devez tapper :
`source activate my_python` (a faire à chaque nouveau login)
Ensuite, pour installer des packages supplémentaires :
essayer en premier : `conda install « package name»` (option j'ai de la chance;-) nous prévenir on pourra l'ajouter dans la base
ensuite suivant les packages vous pouvez essayer :

- `pip install « package name»`
- `easy_install « package name»`
- `binstar search « package name»`

Pour décharger l'environnement virtuel :

```
source deactivate my_python
```

C compilateur fortran et C

plusieurs compilateurs sont installés sur les machines :
- gcc/gfortran 4.4.7 (celui du système) et gcc/gfortran 4.9.3
- ifort/icc 12.1.3.293
- pgf95/pgcc (2 versions : 2013 par défaut et 2011)
- nagfor (Fortran uniquement version 6.0 et 5.3)

Un certain nombre de bibliothèques classiques ont été installé pour chacun des compilateurs (hdf5 , netcdf3 , netcdf4.2, openmpi 1.4.5 et openmpi 1.6.5) ainsi qu'un fichier de module pour chacune d'entre elles

Pour les utiliser , on charge les modules du compilateur et des bibliothèques

exemple : compilateur ifort et librairie netcdf 4.2

```
[weill@srv-calcul1 ~]$ module load intel netcdf4/4.2.1.1-ifort
 [weill@srv-calcul1 ~]$ module list
Currently Loaded Modulefiles:
  1) python/2.7-anaconda     2) netcdf4/4.2.1.1-ifort   3)
intel/12.1.3.293
 [weill@srv-calcul1 ~]$ ifort sample.f90 -I/opt/netcdf42/ifort/include
-L/opt/netcdf42/ifort/lib -lnetcdff -lnetcdf
[weill@srv-calcul1 ~]$ ldd a.out
        linux-vdso.so.1 =>  (0x00007fffb63ff000)
        libnetcdff.so.5 => /opt/netcdf42/ifort/lib/libnetcdff.so.5
(0x00007f3ae28e8000)
        libnetcdf.so.7 => /opt/netcdf42/ifort/lib/libnetcdf.so.7
(0x00007f3ae2562000)
        libm.so.6 => /lib64/libm.so.6 (0x0000003483800000)
        libpthread.so.0 => /lib64/libpthread.so.0 (0x0000003483c00000)
        libc.so.6 => /lib64/libc.so.6 (0x0000003483400000)
        libgcc_s.so.1 => /lib64/libgcc_s.so.1 (0x0000003486000000)
        libdl.so.2 => /lib64/libdl.so.2 (0x0000003484000000)
        libhdf5_hl.so.7 => /opt/hdf518/ifort/lib/libhdf5_hl.so.7
(0x00007f3ae2309000)
        libhdf5.so.7 => /opt/hdf518/ifort/lib/libhdf5.so.7
(0x00007f3ae1d70000)
        libz.so.1 => /lib64/libz.so.1 (0x0000003484400000)
        librt.so.1 => /lib64/librt.so.1 (0x0000003484800000)
        libsz.so.2 => /usr/lib64/libsz.so.2 (0x00007f3ae1b5c000)
        libcurl.so.4 => /usr/lib64/libcurl.so.4 (0x0000003486800000)
        libifport.so.5 =>
/opt/intel/composer_xe_2011_sp1.9.293/compiler/lib/intel64/libifport.so.5
(0x00007f3ae1a26000)
[...]
 /opt/intel/composer_xe_2011_sp1.9.293/compiler/lib/intel64/libintlc.so.5
(0x00007f3ae0b58000)
        /lib64/ld-linux-x86-64.so.2 (0x0000003483000000)
```

ifort : si vous obtenez le message

```
ifort: error #10001: could not find directory in which g++ resides
```

essayer l'une des trois solutions

```
export LC_ALL=fr_FR.UTF-8 ou bien
export LC_MESSAGES=C ou bien
unset LC_MESSAGES
```

III cluster et gestion de batch

A introduction

Un cluster est un ensemble de machines avec trois rôles distincts

noeuds de tête (pour se connecter et compiler)

noeuds d'exécutions (les machines sur lesquelles seront exécutés les programmes)

gestionnaire de batch (qui affecte les noeuds d'exécutions aux programmes et réparti les ressources entre les différents utilisateurs). nous utilisons Torque+Maui comme gestionnaire de jobs

en fait, depuis le noeud de tête on va soumettre des jobs qui vont s'exécuter sur d'autres machines en réservant les ressources demandées pour un usage exclusif ... pendant un certain temps.

Les ressources sont le nombre de CPU et la Mémoire

B soumissions de jobs

la soumission de job se fait à travers la commande **qsub**
on ne peut pas soumettre de binaire mais que des scripts (shell, python, perl ...)
votre job démarre par défaut dans votre répertoire d'accueil donc il faudra sûrement vous déplacer dans le bon répertoire dans votre job
le job minimum c'est :

```
#!/bin/bash
my_binary
```

il faut rendre le job exécutable : chmod u+x myjob.sh
ensuite lancer le job
exemple :

```
[weill@srv-calcul1 ~]$ cat myjob.sh
#!/bin/bash
pwd
echo "hello"

[weill@srv-calcul1 ~]$ qsub myjob.sh
6987.pbsmaster6.latmos.ipsl.fr
[weill@srv-calcul1 ~]$
```

lorsque le job est fini vous récupérez deux fichiers

myjob.sh.e<<numero_du_job>> : il contient les erreurs lors de l'exécution de votre script. Si il n'y pas d'erreur c'est un fichier vide
[weill@srv-calcul1 ~]$ ls -l myjob.sh.e6987
-rw------- 1 weill latmos 0 20 févr. 19:03 myjob.sh.e6987

myjob.sh.o<<numero_du_job>> : il contient les messages écrit par votre script + des informations sur les ressources utilisés par le script

[weill@srv-calcul1 ~]$ cat myjob.sh.o6987
Running Host: srv-calcul3.latmos.ipsl.fr
/net/nfs/home/weill
hello
Epilogue Args:
Job ID: 6987.pbsmaster6.latmos.ipsl.fr
User ID: weill
Group ID: latmos
Job Name: myjob.sh
Session ID: 46420
Resource List: mem=3gb,neednodes=1,nodes=1,vmem=4gb,walltime=02:00:00
Resources Used: cput=00:00:00,mem=0kb,vmem=0kb,walltime=00:00:00
Queue Name: short
Running Host: srv-calcul3.latmos.ipsl.fr

C commandes de gestion de jobs

lancement de jobs : **qsub**
lister les jobs : **qstat, showq**
tuer un job : **qdel**
état du cluster : **check-cluster**
modifier un job : **qalter** (ne fonctionne que lorsqu'il n'est pas en cours d'exécution)

1 queues d'exécution

des queues (files) différentes en fonction de la durée de l'exécution d'un programme ont été définies

pour avoir la listes des queues existantes : **qstat -q**

```
[weill@ciclad-ng ~]$ qstat -q

server: cicladpbs6.private.ipsl.fr

Queue              Memory CPU Time Walltime Node  Run Que Lm  State
---------------- ------ -------- -------- ----  --- --- --  -----
short                --       --    02:00:00   --    0   0 --   E R
day                  --       --    24:00:00   --    0   0 --   E R
weeks2               --       --    360:00:0   --    3   0 --   E R
infini               --       --    860:00:0   --    4   0 --   E R
week                 --       --    168:00:0   --    9   0 --   E R
default              --       --       --      --    0   0 --   E R
std                  --       --    06:00:00   --    2   0 --   E R
h12                  --       --    12:00:00   --    0   0 --   E R
days3                --       --    72:00:00   --    0   0 --   E R
```

2 jobs dans tous ses états

A la soumission , un job se trouve dans un état appelé « ***Queued*** ».
ensuite si le job est elligible il passe en état « ***Idle*** »
si il y a assez de ressources pour l'exécuter il passe en état « ***Running*** »
En règle générale il se passe quelques secondes entre la soumission et l'exécution si le cluster n'est pas plein

3 les limites utilisateurs

celles ci dépendent des clusters sur lesquels vous calculez et peuvent êttre modifiées par les administrateurs au cours du temps :

limites par défaut au 01/05/2015

LATMOS GUYANCOURT : 40 processeurs dans le cluster
8 cpu par utilisateur dans toutes les queues sauf weeks2(4) et infini(2)
Jobs interactifs uniquement dans les queues =< 12H
Jobs parallèles uniquement dans les queues =< week
3Go de ram et 4Go de mémoire virtuelle par défaut

LATMOS JUSSIEU : 16 processeurs dans le cluster
4 cpu par utilisateur
Job interactif uniquement dans les queues < 12H
Jobs parallèles uniquement dans les queues =< week
3Go de ram et 4Go de mémoire virtuelle par défaut

CICLAD : 1200 processeurs dans le cluster
24 cpu par utilisateur
Jobs interactifs uniquement dans les queues =< 12H
Jobs parallèles uniquement dans les queues =< week
3Go de ram et 4Go de mémoire virtuelle par défaut

CLIMSERV : 912 processeurs dans le cluster

pas de limite sauf dans les queues
treedays (20)
week (10)
infini (5)
2Go de ram et 4Go de mémoire virtuelle par défaut

4 check-cluster

C'est une commande qui a été écrite localement qui permet de connaitre l'état des noeuds d'un cluster

```
[weill@ciclad-ng ~]$ check-cluster
NODE         STATE        FREE-CPU    FREE-MEM    LOAD  PROPERTY   PARTITION
ciclad5      Busy            0/8       4/31 Gb    6.24  [std]      std
ciclad6      Memory-Full     2/8       0/31 Gb    6.05  [std]      std
ciclad8      Running         3/8       2/31 Gb    5.14  [std]      std
ciclad9      Memory-Full     0/8       0/30 Gb    8.03  [std]      std
ciclad10     Running         5/8      19/31 Gb    4.64  [std]      std
ciclad16     Busy            0/32     26/126 Gb   32.24  [std]      std
ciclad17     Running        12/32    102/126 Gb   14.39  [std]      std
ciclad18     Running        29/32    105/126 Gb    5.68  [std]      std
ciclad23     Busy            0/64      2/252 Gb   64.09  [std]      std
ciclad24     Busy            0/64      2/252 Gb   64.00  [std]      std
ciclad14     Idle           32/32    126/126 Gb    0.41  [heppi]    std
ciclad15     Running         7/32     27/126 Gb   25.14  [heppi]    std
ciclad19     Idle           64/64    252/252 Gb    0.00  [heppi]    std
ciclad20     Busy            0/64      2/252 Gb   64.00  [heppi]    std
ciclad21     Busy            0/64      2/252 Gb   63.98  [heppi]    std
ciclad22     Idle           64/64    252/252 Gb    0.00  [heppi]    std
    44 Active Jobs     366 of  584 Processors Active (62.67%)
                        13 of   16 Nodes Active      (81.25%)
Total Jobs: 196   Active Jobs: 44   Idle Jobs: 0   Blocked Jobs: 152
[weill@ciclad-ng ~]$
```

5 commandes qstat et showq

qstat (sans options liste tous les jobs) : même ceux des autres utilisateurs
qstat -u « user » : liste uniquement les job de l'utilateur user
qstat -a : liste tous les job avec un petit peu plus d'information
qstat -r : liste tous les jobs en train d'être exécuté
qstat -n : donne les noeuds d'exécution des jobs
qstat -f « numéro du job » : donne toutes les informations sur le job en question
qstat -q : donne toutes les queues existantes sur le cluster

showq (sans options) : donnes tous les jobs
showq -i : donne tous les jobs en attente de ressources
showq -b : donne tous les jobs qui sont bloqués

[weill@srv-calcul1 ~]$ showq -b
JobName User Reason

7051 weill violates active HARD MAXJOB limit of 3 for class infini user (R: 1, U: 3)
7057 weill violates active HARD MAXJOB limit of 4 for class weeks2 user (R: 1, U: 4)
7059 weill violates active HARD MAXPROC limit of 8 for user weill (R: 6, U: 7)

6 commande qdel

qdel « numero du job » : détruire un job (quelque soit son état : running , idle ou en queue)

7 commande qsub

a les principaux paramètres

la commande minimum c'est : **qsub job.sh**

dans ce cas là on utilise les paramètres par défaut des systèmes

2H de CPU , 3 Go de ram (6H 2Go sur climserv)

spécifier la durée du job : 2 solutions
exemple pour 24H

```
qsub -q day job.sh
qsub -l walltime=24:00:00 job.sh
```

Avoir en sortie un seul fichier contenant à la fois les erreurs et les sorties
à la fin vous obtenez un seul fichier nommé job.sh.o« jobID » à la fin de l'execution de votre job

```
qsub -j oe job.sh
```

Pour pouvoir surveiller la sortie et les erreurs du job pendant l'exécution
le fichier de sortie se trouvera dans votre home sous le nom job.sh.o« JobID »
Attention à votre quota si le script est très verbeux

```
qsub -k oe -j oe job.sh
```

pour surveiller la sortie pendant l'exécution : **tail -f $HOME/job.sh.o« jobID »**

même si c'est un système de batch vous avez la possibilité de l'utiliser pour faire de l'interactif (-IV)et même du graphique (-IVX)

```
[weill@camelot ~]$ qsub -IVX
qsub: waiting for job 122427.merlinvirt-c.climserv to start
qsub: job 122427.merlinvirt-c.climserv ready

[weill@merlin8-c ~]$
```

par contre en interactif la durée est limitée à 12H

```
[weill@srv-calcul1 ~]$ qsub -q day -IVX
qsub: submit error (Queue does not allow interactive jobs MSG=interactive
job is not allowed for queue: user weill@srv-calcul1.latmos.ipsl.fr, queue
day)
```

le passage d'argument à votre script n'est pas autorisé par qsub, par contre il est possible de définir des variables via qsub que l'on peut réutiliser dans le script ($variable_name)

qsub -v variable_name="variable_value"

Il est possible de mettre les options de qsub sous forme de commentaire dans le script en commençant la ligne par #PBS

exemple :

```
#!/bin/bash
#PBS -q week
#PBS -j oe  -k oe
```

IDL Matlab en batch ...

Il est possible de lancer en batch via qsub de l'IDL , du matlab ... sous réserve que vous n'utilisiez pas le graphique sortie des graphique directement sous forme ps pdf jpg ...

Si votre programme vous dit qu'il n'a pas assez de mémoire pendant le job il est possible de lancer un jobs avec plus de mémoire

Spécification de la mémoire pour le job

qsub -l mem=3g,vmem=4g

Attention à ne pas trop surestimer la mémoire utilisé par le programme
la mémoire coute cher et limite le nombre de cpu utilisable par les autres

lancement de job parallèlle avec plusieurs CPU ou bien plusieurs noeuds (ciclad-climserv)

`qsub -l nodes=x:ppn=y`	je veux y processeurs sur x noeuds
`qsub -l nodes=1:ppn=4`	je veux un noeud avec 4 coeurs
`qsub -l nodes=1:ppn=2+nodes=3:ppn=4`	je veux 1 noeud avec 2 coeurs+3 noeuds avec 4 coeurs

lancement d'un job à une heure donnée

`qsub -a « time »`	
`qsub -a 2010`	lancer le job à 20H10
`qsub -a $(date -d '+2min' +%H%M)`	lancer le job 2 minutes après la commande

Dépendance de job (ne lancer le job que si le premier c'est bien executé)

```
qsub -W depend=afterok:jobid[:jobid...]
```

Pour en savoir encore plus

```
man qsub
```