HPC SysAdmin Meeting'11

environment modules

… o cómo administrar entornos de usuario dinámicamente

Daniel Masó (XRQTC, UdG) i Carles Acosta (XRQTC, UAB)

¿De qué vamos a hablar?

- Problemática, motivación de uso y concepto
- Instalación
- Uso de modules
- Estructura y orden
- Múltiples arquitecturas

En un escenario habitual:

- Cuando un usuario ejecuta en un shell, normalmente se inicializa el entorno para las diferentes aplicaciones que necesita.
- Cada usuario tiene diferentes archivos en su home donde guarda esa información (.bashrc, p.e.).
- Aumento de la complejidad de los archivos de inicialización
- = Δ de la dificultad por parte del administrador para trazar problemas y aplicar cambios
- = Sistemas muy heterogéneos
- La intención de Modules es simplificar este proceso.

Motivaciones

Furlani, JL. The Design of The Modules Package, ***1991****.*

- Evitar problemas de mantenimiento del entorno a los usuarios.
- Simplificar la diseminación de la información y la documentación de nuevo software.
- Simplificar el cambio continuo de entorno en una misma sesión
- Disminuir la dependencia en los servidores cuando las aplicaciones de estos servidores no son necesarias.
- Solventar las dificultades asociadas entre los cambios de diferentes versiones de aplicaciones.

¿Qué son los 'environment modules'?

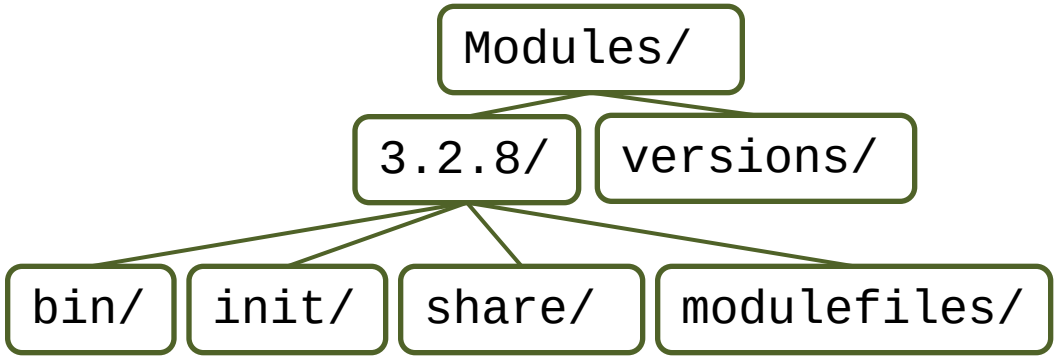
- Conjunto de scripts que nos permiten modificar fácilmente el entorno de usuario.
- Los diferentes módulos cargan el entorno para una aplicación concreta.
- Soporta las shells más populares: bash, csh, ksh, tcsh, etc. y lenguajes como Perl.

http://modules.sourceforge.net/
http://sourceforge.net/projects/modules/files/Modules/

Última versión: 3.2.8 (04/10/10)

- Dependencia con Tcl/Tclx (8.4)
- Debe instalarse en un directorio compartido (NFS, GlusterFS,…)
  Por ejemplo `/opt/`
- ```
  $tar –xvf modules-3.2.8.tar.gz
  $cd modules-3.2.8/
  $./configure --prefix=/opt/--with-tcl-lib=/usr/lib64 --
  with-tcl-inc=/usr/include && make && make install
  ```

En /opt se crea la estructura siguiente


  ```

- Debe inicializarse con el login de los usuarios según la shell que se usa
  `$source /opt/Modules/3.2.8/init/bash`

  Especifica variables como el MODULESHOME o MODULEPATH.

```
$module av

---------- /opt/Modules/versions ---------
3.2.8

----/opt/Modules/3.2.8/modulefiles -------
dot  module-cvs  module-info modules  null  use.own
```

- Comandos básicos:

`$module avail (o av)` muestra los módulos que tenemos disponibles

`$module load modulefile (o add)` carga un módulo concreto, por ejemplo, module load TORQUE

`$module unload modulefile (o rm)` quita un módulo cargado

`$module list` lista los módulos cargados por el usuario

`$module show modulefile (o display)` muestra la información de un módulo determinado

`$module whatis modulefile` muestra la información que contiene el module-whatis del fichero modulefile

`$module update` recarga todos los módulos cargados

`$module purge` elimina todos los módulos cargados

`$module help` muestra información de ayuda del módulo

Creando entornos

- Una vez instalado, es fácil definir los entornos, consiste básicamente en crear los diferentes ficheros en modulefiles

- Los modulefiles están escritos en Tcl (Tool Command Language) y son interpretados por el modulecmd que se ejecuta cada vez que invocamos module

```
/opt/Modules/3.2.8/bin/modulecmd
/opt/Modules/3.2.8/modulefiles
```

- Ejemplo: un módulo para el sistema de colas

```
/usr/local/torque
bin/
```

```
$ vi /opt/modules-3.2.8/Modules/3.2.8/modulefiles/TORQUE
```

```
#%Module
##
## Torque/PBS modulefile
##
##
proc ModulesHelp { } {
        global version modroot

        puts stderr "\tTORQUE Carga el entorno para el gestor
de colas"
}

module-whatis   "Entorno para Torque/OpenPBS"

prepend-path     PATH /usr/local/torque/bin


$module av
$module load TORQUE
$module list
$module whatis TORQUE
$module help TORQUE
```

- Comandos básicos en los ficheros modulefiles

`module-whatis` información que aparecerá al usar module whatis modulefile

`prepend-path/append-path` añadir directorios al PATH, MANPATH o LD_LIBRARY_PATH

`remove-path` eliminar directorios del PATH, MANPATH o LD_LIBRARY_PATH

`setenv` añadir o cambiar variables de entornos

`prereq` listar los módulos que son requisito para un módulo en concreto

`conflict` especificar que un módulo crea un conflicto con otro

`set-alias` establecer un alias

`unset-alias` quitar un alias

`set puts switch`, etc.

- Comandos básicos en los ficheros modulefiles

`module-whatis` información que aparecerá al usar module whatis modulefile

Se trata simplemente de ofrecer información al usuario

módulo g09_b.01-EM64T:

```
module-whatis   "Entorno para Gaussian09 Rev. B.01 (EM64T)“
```

```
$module whatis g09_b.01-EM64T
g09_B.01-EM64T    : Entorno para Gaussian09 Rev. B.01 (EM64T)
```

- Comandos básicos en los ficheros modulefiles

`prepend-path/append-path` directorios al PATH, MANPATH o LD_LIBRARY_PATH

`remove-path` eliminar directorios del PATH, MANPATH o LD_LIBRARY_PATH

Una de las claves por las que usar modules

módulo molpro2010.01_EM64T:

```
append-path    PATH /opt/MOLPRO/molpro2010_em64t/bin
append-path    LD_LIBRARY_PATH /opt/MOLPRO/molpro2010_em64t/lib
```

```
$module load molpro2010.1_EM64T
$which molpro
/opt/MOLPRO/molpro2010_em64t/bin/molpro
$echo $PATH
/opt/MOLPRO/molpro2010_em64t/bin:/usr/local/bin:/bin:/usr/bin…
$echo $LD_LIBRARY_PATH
/opt/MOLPRO/molpro2010_em64t/lib/
```

- Comandos básicos en los ficheros modulefiles

setenv añadir o cambiar variables de entornos

módulo g09_b.01-EM64T:

```
set             g09root /opt/GAUSSIAN/g09b.01_em64t/
set             GAUSS_SCRDIR /scratch
set             gr      $g09root
set             gman    $g09root/bsd
set             GAUSS_EXEDIR $gr/g09/bsd:$gr/g09/private:$gr/g09
set             GV_DIR  /QFsoft/GAUSSIAN/

setenv          g09root $g09root
setenv          GAUSS_EXEDIR $GAUSS_EXEDIR
setenv          GAUSS_ARCHDIR $gr/g09/arch
setenv          GMAIN $GAUSS_EXEDIR
setenv          GAUSS_SCRDIR $GAUSS_SCRDIR
setenv          G09BASIS $gr/basis
```

```
$module load g09_b.01-EM64T
$echo $G09BASIS
/opt/GAUSSIAN/g09b.01_em64t/basis
```

- Comandos básicos en los ficheros modulefiles

`prereq` listar los módulos que son requisito para un módulo en concreto

`conflict` especificar que un módulo crea un conflicto con otro

módulo mvapich2-64/pgi8.0/1.2p1:

```
prereq pgi64/8.0-3
conflict mvapich2-64/pgi8.0/1.0.6p1
```

```
$module load mvapich2-64/pgi8.0/1.2p1
mvapich2/1.7a_pgi8.0(20):ERROR:151: Module 'mvapich2/1.7a_pgi8.0'
depends on one of the module(s) 'pgi64/8.0-3'
…

$module load mvapich2-64/pgi8.0/1.0.6p1
$module load pgi64/8.0-3
$module load mvapich2-64/pgi8.0/1.2p1
mvapich2-64/pgi8.0/1.2p1(26):ERROR:150: Module
'mvapich2-64/pgi8.0/1.2p1' conflicts with the currently loaded
module(s)'mvapich2-64/pgi8.0/1.0.6p1‘
…
```

- Comandos básicos en los ficheros modulefiles

`set-alias` establecer un alias

`unset-alias` quitar un alias

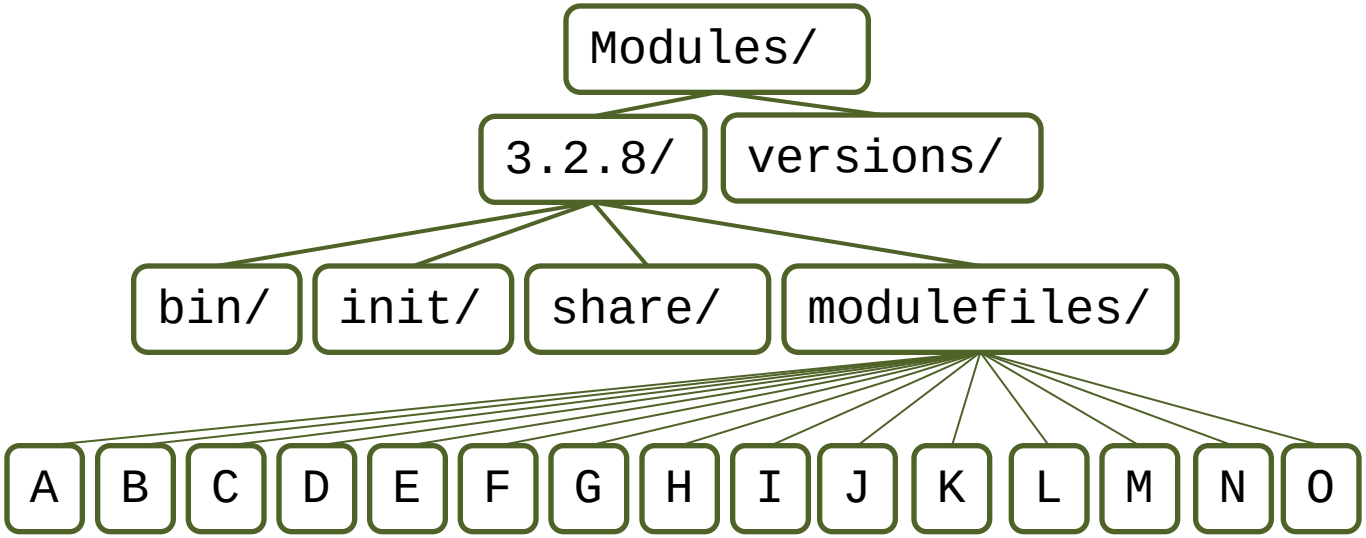
Muy interesante para simplificar por ejemplo las líneas de comandos.
Así, para cálculos MPI podríamos usar:

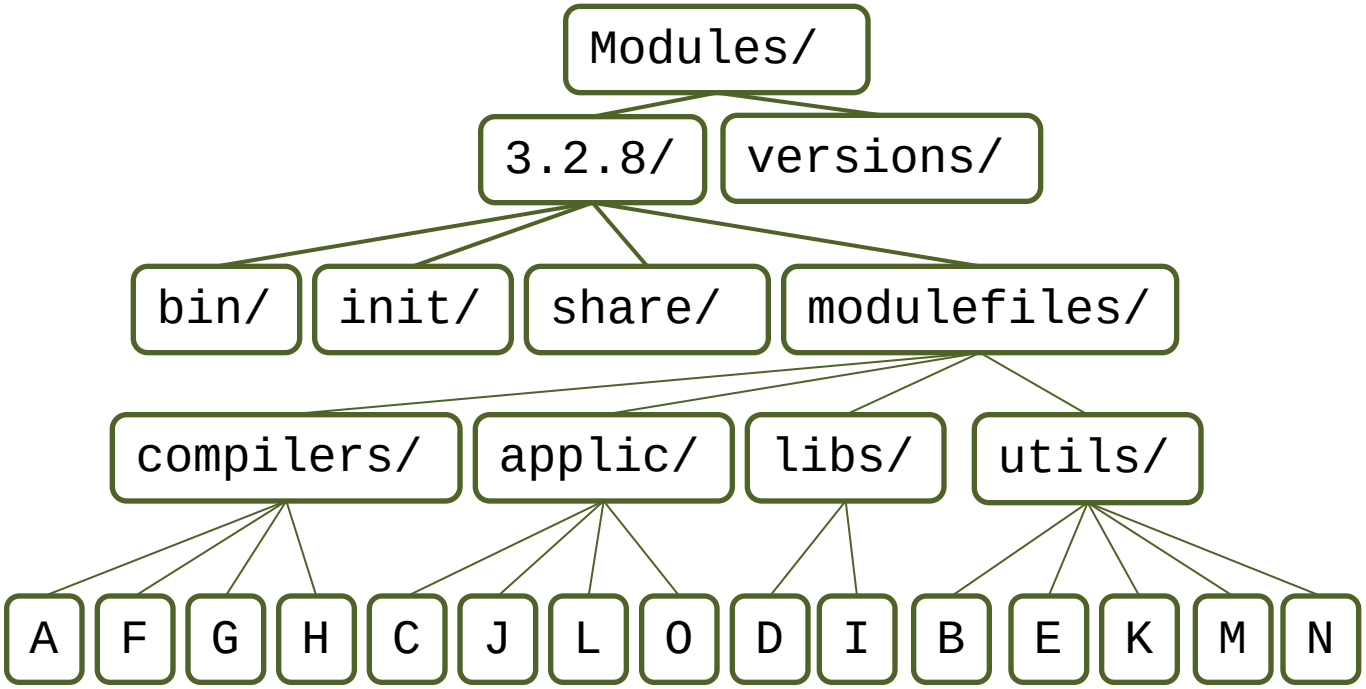
*OpenMPI

```
set-alias MPIRUN “mpirun –np \$nprocs –mca bt self,opeinb \$1”
```

*MPICH

```
set-alias MPIRUN “mpirun –np \$nprocs –nolocal –machinefile
$TMPDIR/machines \$1”
```


Modules/
3.2.8/
versions/
bin/
init/
share/
modulefiles/
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O




```
Modules/3.2.8/init/.modulespath

/opt/modules-3.2.8/Modules/3.2.8/modulefiles/compilers
/opt/modules-3.2.8/Modules/3.2.8/modulefiles/applic
/opt/modules-3.2.8/Modules/3.2.8/modulefiles/libs
/opt/modules-3.2.8/Modules/3.2.8/modulefiles/utils
```

Sin modulespath

```
$module av

---------- /opt/modules-3.2.8/Modules/versions ---------
3.2.8
----/opt/modules-3.2.8/Modules/3.2.8/modulefiles -------
A   B   C   D   E   F   G   H   I   J   K   L   M   N   O

$module load D
```

modulespath

```
$module av

---------- /opt/modules-3.2.8/Modules/versions ---------
3.2.8
----/opt/modules-3.2.8/Modules/3.2.8/modulefiles/compilers ---
A   F   G   H
----/opt/modules-3.2.8/Modules/3.2.8/modulefiles/applic ---
C   J   L   O
----/opt/modules-3.2.8/Modules/3.2.8/modulefiles/libs ---
D   I
----/opt/modules-3.2.8/Modules/3.2.8/modulefiles/utils ---
B   E   K   M   N

$module load D
```

Ejemplo

- Queremos compilar e instalar el programa VASP5.2 con IFC11.1 y OpenMPI 1.4.3
- Tenemos instalado IFC11 en /opt. Primero creamos el módulo de IFC11.1:

`Modules/3.2.8/modulefiles/compilers/IFC11.1`

```
#%Module
##
proc ModulesHelp { } {
       puts stderr "\tCarga variables de entorno de Intel(R) Compiler Suite"
       puts stderr "\n\tincluye Intel C++, Intel Fortran, Intel MKL, Intel
IPP\n"

}

module-whatis   “Modulo para variables de entorno ICS. 64 bits"

prepend-path    PATH                    /opt/IFC/ics-11.1.072/bin/intel64
prepend-path    LD_LIBRARY_PATH         /opt/IFC/ics-11.1.072/lib/intel64
prepend-path    MANPATH                 /opt/IFC/ics-11.1.072/man
append-path     INTEL_LICENSE_FILE      /opt/IFC/ics-
11.1.072/NCOM_L_CMP_NDM4-G8ZZH8B6.lic
prepend-path    LD_LIBRARY_PATH         /opt/IFC/ics-11.1.072/mkl/lib/em64t
prepend-path    LIBRARY_PATH            /opt/IFC/ics-11.1.072/mkl/lib/em64t
prepend-path    CPATH                   /opt/IFC/ics-11.1.072/mkl/include
```

```
$module load IFC11.1
```

- Ahora pasamos a compilar OpenMPI e instalarlo…

```
$cd … configure … make … make install
/opt/openmpi-1.4.3-ics
```

- Creamos también un módulo para openmpi (openmpi-1.4.3-ifc11.1):

```
Modules/3.2.8/modulefiles/MPI/openmpi-1.4.3-ifc11.1
```

```
append-path PATH /opt/openmpi-1.4.3-ics/bin
append-path LD_LIBRARY_PATH /opt/openmpi-1.4.3-ics/lib
set-alias MPIRUN “mpirun –np \$nprocs –mca bt self,opeinb \$1”

module load IFC11.1
conflict mpich2
```

```
$module load openmpi-1.4.3-ifc11.1
$module list
Currently Loaded Modulefiles:
  1) TORQUE  2) modules  3) IFC11.1  4) openmpi-1.4.3-ifc11.1
```

- Compilamos e instalamos el VASP…

```
$cd … vi makefile … make –f makefile
/opt/vasp-5.2-ompi-ifc11.1
```

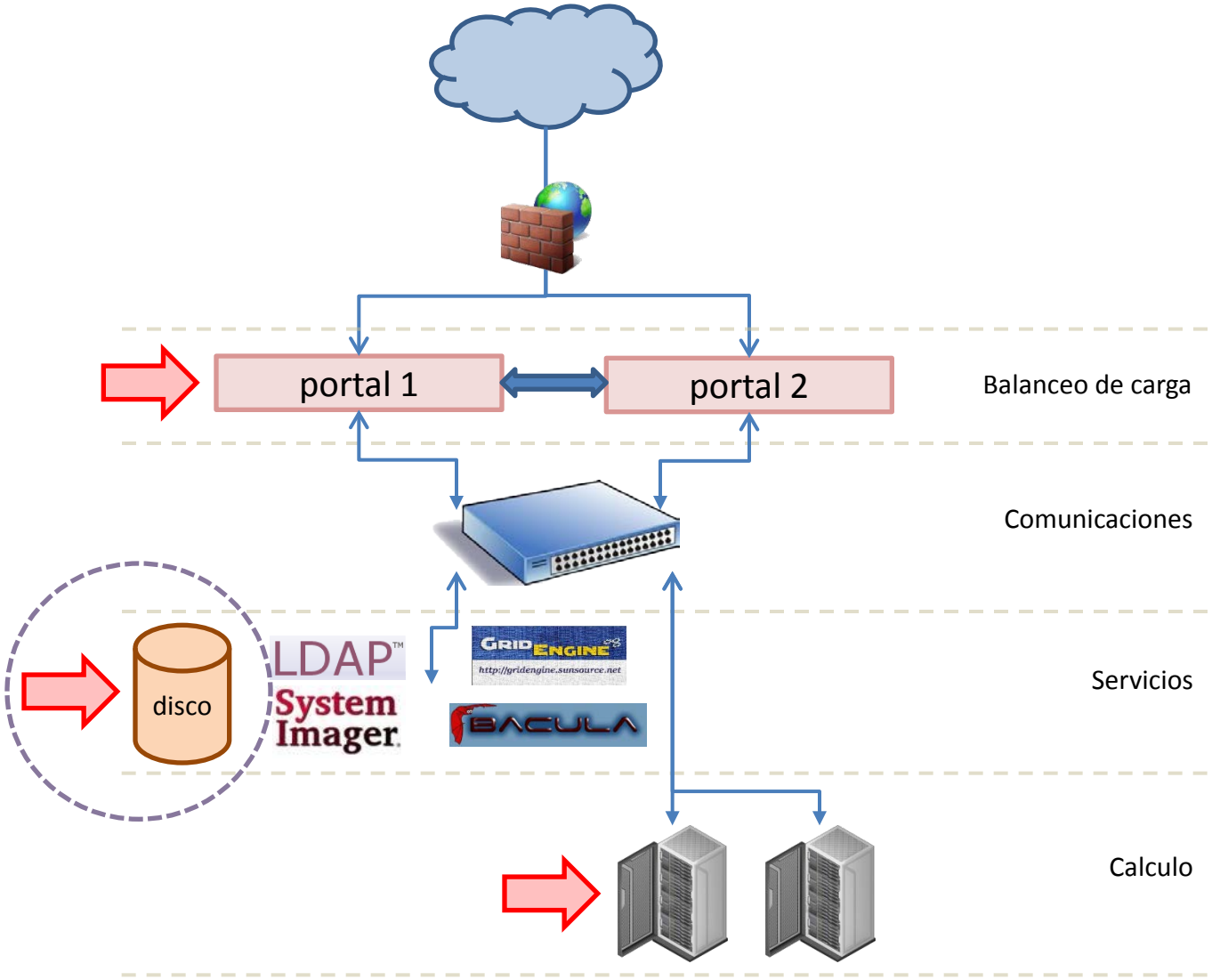
- Creamos el módulo (vasp5.2-ifc11.1-ompi)

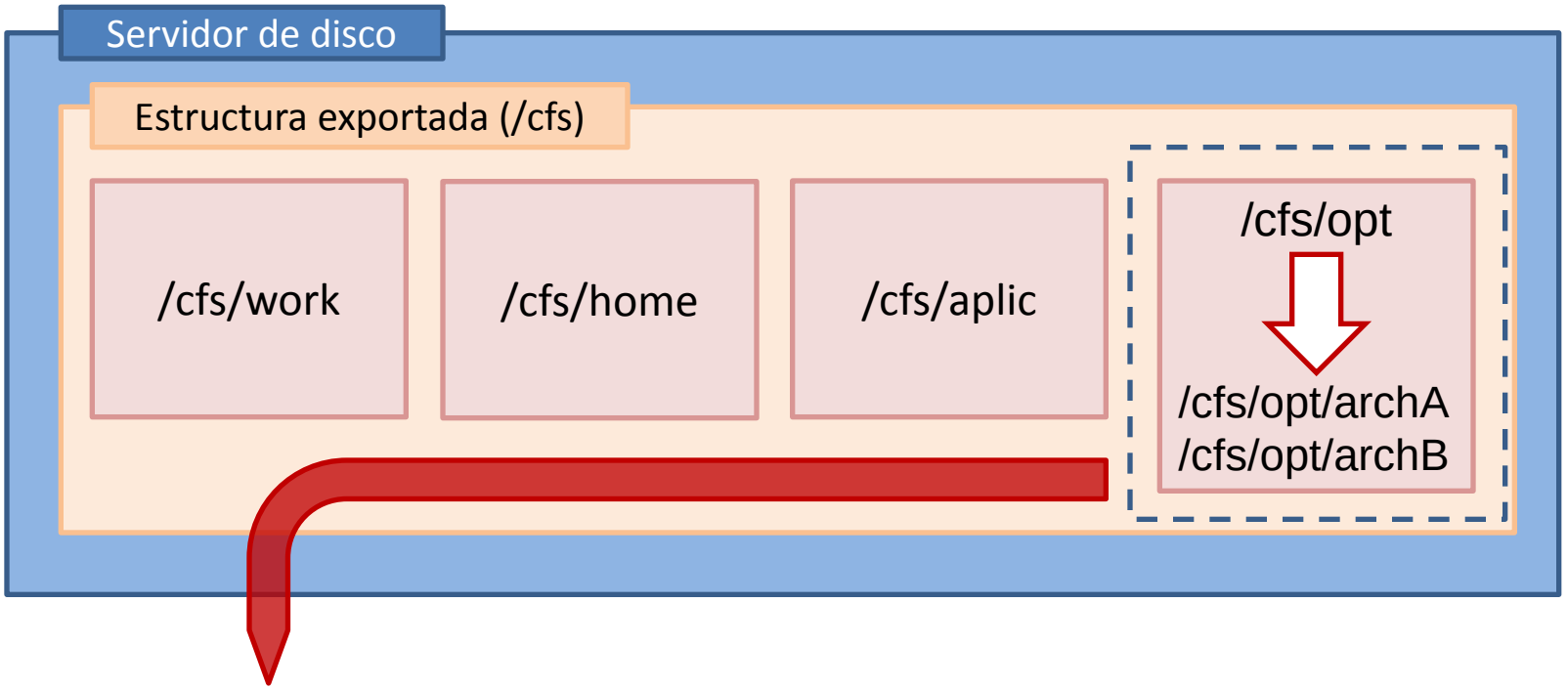
```
Modules/3.2.8/modulefiles/applic/vasp5.2-ifc11.1-ompi
```

```
prepend-path  PATH /opt/vasp-5.2-ompi-ifc11.1/vasp.5.2/

module load openmpi-1.4.3-ifc11.1
conflict vasp4
```

- El usuario sólo necesita cargar el módulo vasp5.2 y tener así cargado todo el entorno tanto de ejecutables del programa como librerías del compilador e interfaz MPI


portal 1
portal 2
Balanceo de carga
Comunicaciones
disco
LDAP
System Imager
GRID ENGINE
http://gridengine.sunsource.net
BACULA
Servicios
Calculo



/cfs/opt/archA/Modules/3.2.8/modulefiles

/cfs/opt/archB/Modules/3.2.8/modulefiles

- Configuramos algunos scripts para adaptar el sistema a nuestras diferentes arquitecturas (clusters)

```
……
FILESYSTEMS="home aplic opt work"
CLUSTER=archA

do_start() {
        echo "Running ldconfig";
        ldconfig;
        sleep 1;
        echo "Running depmod -a";
        depmod -a;
        sleep 1;
        for fs in $FILESYSTEMS
          do
             if [ -x /cfs/$fs ] ; then
             HAVE_MP=1
             else
             echo "Mount point: $fs doesn't exist creating..."
             mkdir /cfs/$fs;
             fi
             sleep 1;
             echo "Mounting GlusterFS : /cfs/$fs";
             /sbin/glusterfs -f /etc/glusterfs/glusterfs-client-${fs}.vol /cfs/$fs;
        done
        sleep 2
        mount -o bind /cfs/aplic/$CLUSTER /aplic
        mount -o bind /cfs/opt/$CLUSTER /opt
        mount -o bind /cfs/home /users
        mount -o bind /cfs/work /work
}
…..
```

Script para montar GlusterFS en nodos de càlculo de arquitectura *archA*

Para arquitectura *archB*, solo hay que substituir la variable de CLUSTER.

- Así podemos obtener, por ejemplo, una estructura parecida a…

nodo1archA:~ # module av

```
------------------------------------------------------------ /opt/Modules/modulefiles/Libs ------------------------------------------------------------
fftw3/3.2.2_gcc-4.3.2          intel_mkl/11.0.074          mkl/10.0                    petsc/3.0.0-p12_intel10.1_complex

---------------------------------------------------------- /opt/Modules/modulefiles/Compilers ----------------------------------------------------------
intel/10.1              intel_compiler_suite/11.0.074 intel_compiler_suite/11.1.072 pgi/8.0-6

--------------------------------------------------------- /opt/Modules/modulefiles/Applications ---------------------------------------------------------
GlobalArrays/ga-4-1-1_intel-10.1_ofed-1.5.1_blcr-8.2          gamess/2010r2_intel11.1_mkl11.1_ompi-1.4.2

------------------------------------------------------------ /opt/Modules/modulefiles/MPI ------------------------------------------------------------
Intel-MPI/4.0.028                    OpenMPI/1.4.1_intel11.1_icc_tcp          OpenMPI/1.4.2_intel11.1_ib_ofed1.5.2

nodo1archA :~ #
```

nodo1archB:~ # module av

```
------------------------------------------------------------ /opt/Modules/modulefiles/Libs ------------------------------------------------------------
gsl/1.14_intel10.1          lapack/3.2.2_intel_11.0.074    petsc/3.0.0-p12_intel10.1    slepc/3.0.0-p12_intel10.1

---------------------------------------------------------- /opt/Modules/modulefiles/Compilers ----------------------------------------------------------
intel/9.1               intel_compiler_suite/11.1.059 pgi/10.2

--------------------------------------------------------- /opt/Modules/modulefiles/Applications ---------------------------------------------------------
crystal/09                                  siesta/3.0-rc2_serial
gamess/2010r2_intel11.1_mkl11.1             siesta/smeagol_ompi

------------------------------------------------------------ /opt/Modules/modulefiles/MPI ------------------------------------------------------------
OFED/1.5.1                    OpenMPI/1.4.1_intel11.1_tcp(default)    OpenMPI/1.4.2_pgi-10.2_ofed-1.5.1_blcr-8.2

nodo1archB :~ #
```

Modules y ficheros en el portal de acceso

Script (module_archA) ubicado en /usr/local/bin del portal de acceso

```
#!/bin/bash
CLUSTER=archA
MODULEPATH=/cfs/opt/$CLUSTER/Modules/modulefiles
dirs="Compilers Libs Applications Base"

case "$1" in
   avail)
   echo "=============================================================="
   echo "     Moduls que es poden trobar al cluster $CLUSTER       "
   echo "=============================================================="
   for i in $dirs
     do
      printf "========================= %s ========================= \n" $i
      cd $MODULEPATH/$i;
      ls $(find . -type f | sed -e "s/.\///")
     done
   ;;
…
…….
…..
echo "=============================================================="
   echo " Aquesta comanda nomes mostra les els moduls del cluster $CLUSTER"
   echo " Els moduls que es mostren, sols es poden carregar en els nodes de"
   echo " calcul d'aquest cluster."
   echo "=============================================================="
   echo " Comandes disponibles :
         + display|show         modulefile
         + avail                modulefile
         + whatis               modulefile
        "
   echo "=============================================================="
   ;;
esac
```

- Si ejecutamos el script sin ninguna opcion:

```
portal:/# module_archA
============================================================
 Aquesta comanda nomes mostra les els moduls del cluster archA
 Els moduls que es mostren, sols es poden carregar en els nodes de
 calcul d'aquest cluster.
============================================================
 Comandes disponibles :
        + display|show          modulefile
        + avail                 modulefile
        + whatis                modulefile

============================================================
```

- De qué software dispongo en esta arquitectura/cluster?

```
portal:~ # module_archA avail

------------------------------------------------------------ /opt/Modules/modulefiles/Libs -------------------------------------------------------
fftw3/3.2.2_gcc-4.3.2           intel_mkl/11.0.074              mkl/10.0                        petsc/3.0.0-p12_intel10.1_complex

------------------------------------------------------------ /opt/Modules/modulefiles/Compilers ---------------------------------------------------
intel/10.1                  intel_compiler_suite/11.0.074 intel_compiler_suite/11.1.072 pgi/8.0-6

------------------------------------------------------------ /opt/Modules/modulefiles/Applications ------------------------------------------------
GlobalArrays/ga-4-1-1_intel-10.1_ofed-1.5.1_blcr-8.2            gamess/2010r2_intel11.1_mkl11.1_ompi-1.4.2

------------------------------------------------------------ /opt/Modules/modulefiles/MPI ---------------------------------------------------------
Intel-MPI/4.0.028                       OpenMPI/1.4.1_intel11.1_icc_tcp             OpenMPI/1.4.2_intel11.1_ib_ofed1.5.2

nodo1archA :~ #
```

- Enviar un job en cola. Modules+job+SGE

  - Localizamos el modulo que nos interesa cargar para que nuestro job funcione.

    Como? Mediante 'module_archA avail' en el nodo portal

  - Añadimos el comando para que, una vez el script entre en el nodo de cálculo, cargue el modulo que queremos

    Como?

```
#! /bin/bash

module load adf

$ADFBIN/quild << eor
title La_C82

XC
GGA  BLYP
DISPERSION
end

SCF
ITERATIONS 99
CONVERGE 1e-06 1e-05
diis ok=0.01
MIXING 0.10 0.15
end
.....
...............
.....
atoms
C     -0.73174628   -1.23281025    3.78349708
C      0.73174628   -1.23281025    3.78349708
C      1.49061102    0.00000000    3.73902676
C      0.73174628    1.23281025    3.78349708
C     -0.73174628    1.23281025    3.78349708
```

- Enviar un job en cola. Modules+job+SGE
  - Lanzamos el script al sistema de colas mediante qsub

Tipo de cola a utilizar *Nombre del script*

qsub -q llarga3d@@grupo_archA …… script.job

Dónde lo quiero lanzar

HPC SysAdmin Meeting'11

environment modules

… o cómo administrar entornos de usuario dinámicamente

Daniel Masó (XRQTC, UdG) i Carles Acosta (XRQTC, UAB)