Lmod: Nuevo sistema de Environment Modules

Como decíamos en apartados anteriores, TeideHPC es un entorno compartido por cientos de usuarios, por lo que no es lo mismo la forma de proporcionar múltiples aplicaciones en sus múltiples versiones a multitud de usuarios en un entorno de computación distribuida que en un ordenador personal o en tu propio servidor.

La herramienta para gestionar estos entornos se llama ***Environment Modules*** y and ***Lmod*** es una implementación de éste.

Lmod es un sistema de módulos basado en Lua que maneja fácilmente MODULEPATH usando una estructura jerárquica llamada **Hierarchical module naming scheme** lo que proporciona una forma conveniente de cambiar dinámicamente el entorno de los usuarios a través de archivos de módulos. Esto incluye agregar o eliminar fácilmente directorios a la variable de entorno PATH.

- https://lmod.readthedocs.io/en/latest/

Nomenclatura Jerárquica de módulos.

Nomenclatura plana (Flat) vs Nomenclatura jerárquica

El esquema de nomenclatura de módulo predeterminado utilizado hasta ahora es un ejemplo de esquema de nomenclatura de módulo "plano" normal, que se caracteriza por:

- todos los archivos del módulo están directamente disponibles para cargar;
- cada nombre de módulo identifica de forma única una instalación particular;

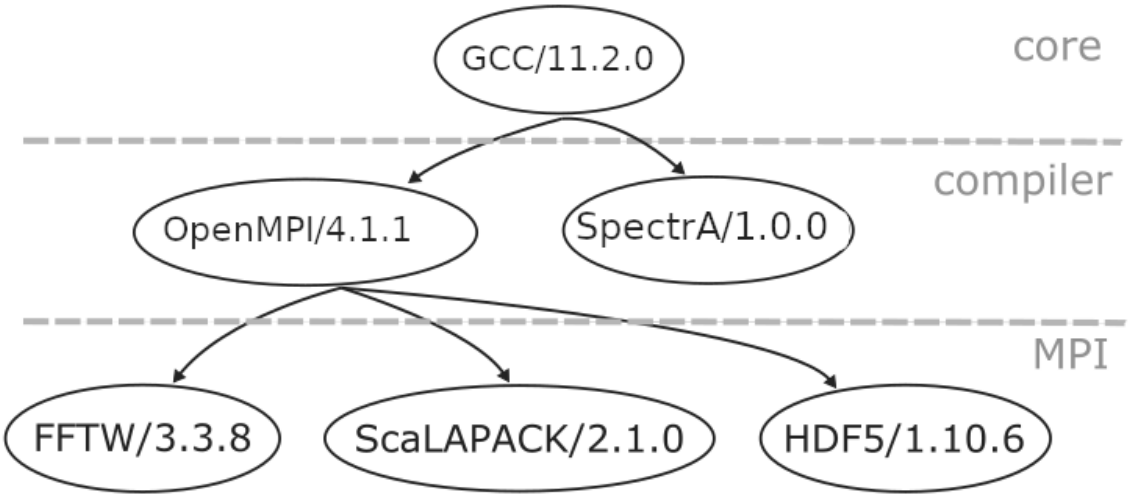
```
$ module av

---------------------------------------------- /opt/envhpc/modulefiles/.rhel6 ----------------------------------------------
admixtools/7.0.2/gcc            hdf5/1.8.16/icc                 perl/5.34.0/gcc
admixture/1.3.0                 hdf5-parallel/1.12.1/intelmpi   picard/2.25.6
angsd/0.935/gcc                 hdf5-parallel/1.8.13/intelmpi   platypus/0.8.1/gcc
....
```

Por el contrario, un **esquema de módulos jerárquicos consta de una jerarquía de archivos de módulos**.

La jerarquía de módulos típica tiene 3 niveles:

- **CORE**: donde se encuentran los archivos del módulo para el software que se instaló utilizando el *sistema;*
- **COMPILER**: donde los archivos de módulo para el software que se instaló usando un determinado *compilador solo;*
- **MPI**: que alberga archivos de módulos para software que se instaló utilizando (al menos) un determinado *compilador y componente MPI;*

En la siguiente imágen se puede ver una jerarquía de módulos de 3 niveles:



En este ejemplo el nivel CORE solo incluye un único módulo GCC/11.2.0, mientras que el nivel COMPILADOR incluye dos módulos: OpenMPI/4.1.1 y SpectrA/1.0.0. En el nivel MPI, hay tres módulos disponibles: uno para FFTW, uno para ScaLAPACK y otro para HDF5.

Como notará, en **cada nivel seleccionamos el módulo de la capa que estamos ingresando**. En el primer nivel primer seleccionamos nuestro compilador. Cuando estamos en el nivel de compilador seleccionamos nuestra implementación MPI, y dentro del nivel MPI seleccionamos nuestro software.

Algunos módulos en el nivel superior de la jerarquía actúan como un **"gateway"** a los módulos en el siguiente nivel inferior, es decir, **para que los módulos adicionales estén disponibles para cargar, primero se debe cargar uno de estos módulos del gateway.** En el caso de la imagen anterior, para realizar el load OpenMPI/4.1.1 primero es necesario precargar el nivel de core GCC/11.2.0 y en el caso de querer usar ScaLAPACK/2.1.0 se necesita haber precargado GCC/11.2.0 y OpenMPI/4.1.1.

Las características de esta jerarquía de módulos son:

- no todos módulos están directamente disponibles para cargar;
- algunos módulos sirven como puerta de entrada (gateway) a más módulos;
- para acceder a algunas instalaciones de software, primero deberá cargar uno o más módulos "*gateway*" para poder utilizarlos;
- resolución de conflictos "automática" mediante la recarga de módulos

Probablemente pueda pensar en otras formas de organizar los archivos de módulos en un árbol de módulos jerárquicos, pero aquí nos ceñiremos este estándar jerárquico ***CORE / COMPILER / MPI*** bastante extendido.

Cómo usar Lmod y comprender el esquema jerárquico?

En secciones posteriores verá cómo se utiliza *Lmod* junto con el programador de tareas Slurm pero para familiarizarse con esta herramienta, puede iniciar sesión y ejecutar los siguientes comandos de uso más común para entender cómo funciona así como entender la jerarquía de módulos.

```
$module spider modulename           -- buscar un module
```

```
$module avail (o av) (modulename)    -- muestra modulos disponibles
```

```
$module load modulename             -- carga un módulo / Compilar / aplicación específico
```

```
$module unload modulename (or rm)    -- elimina un módulo cargado
```

```
$module purge                       -- elimina módulos cargados
```

```
$module list                        -- lista módulos cargados
```

```
$module show modulename             -- muestra la info del módulo
```

```
$module (--raw) show modulename      -- muestra el contenido de un módulo
```

```
$module update modulename           -- recarga un módulo
```

```
$module reset                       -- volver a un conjunto inicial de módulos
```

Si hay muchos módulos en un sistema, puede ser difícil ver qué módulos están disponibles para cargar. Lmod proporciona el comando de descripción general para proporcionar una lista concisa. Por ejemplo

```
$module overview                -- show an overview with all available modules
```

Para más información puede consultar la documentación oficial de esta herramienta.. https://lmod.readthedocs.io/en/latest/010_user.html

¿Cómo funciona el esquema jerárquico?

Buscar software a nivel CORE

Por defecto, si no hay módulos cargados, *lmod* realiza la búsqueda en el nivel CORE, pudiento ver el software que depende del sistema, como el compilador y otras aplicaciones.

```
$ module av
```

```
------------------------------------ /opt/envhpc/modulefiles/.rhel7-eb/all/Core -------------------------------------
   ADMIXTURE/1.3.0                  M4/1.4.18                      gettext/0.21
   Bison/3.5.3                      M4/1.4.19              (D)     gompi/2021b
   Bison/3.7.1                      Miniconda2/4.7.10              iimpi/2021b
   Bison/3.7.6              (D)     Miniconda3/4.9.2               imkl/2021.4.0
   FastQC/0.11.8-Java-1.8           OpenSSL/1.1                    intel-compilers/2021.4.0
   FastQC/0.11.9-Java-11    (D)     PLINK/1.07-x86_64              intel/2021b
   GATK/3.8-0-Java-1.8.0_281        PLINK/1.9b_6.21-x86_64         ncurses/6.2
   GCC/10.2.0                       PLINK/2.00a2.3_x86_64    (D)   picard/2.25.5-Java-13
   GCC/11.2.0               (D)     SHAPEIT/2.r837.GLIBCv2.12      pkg-config/0.29.2
   GCCcore/10.2.0                   Trimmomatic/0.39-Java-1.8      popt/1.18
   GCCcore/11.2.0           (D)     binutils/2.35                  snpEff/4.3t-Java-1.8
   GCTA/1.94.0beta                  binutils/2.36.1                tbl2asn/20200302-linux64
   Java/1.8.0_281           (1.8)   binutils/2.37          (D)     zlib/1.2.11
   Java/11.0.2              (11)    flex/2.6.4
   Java/13.0.2              (D:13)  foss/2021b

  Where:
   Aliases:  Aliases exist: foo/1.2.3 (1.2) means that "module load foo/1.2" will load foo/1.2.3
   D:        Default Module

If the avail list is too long consider trying:

"module --default avail" or "ml -d av" to just list the default modules.
"module overview" or "ml ov" to display the number of modules for each name.

Use "module spider" to find all possible modules and extensions.
Use "module keyword key1 key2 ..." to search for all possible modules matching any of the
"keys".
```

Buscar software a nivel COMPILADOR

Después de cargar un compilador, por ejemplo, cualquier versión de GCC, es posible ver la pila de software compilada para esta versión del compilador.

```
$ module load GCCcore/11.2.0
$ module av
```

```
-------------------------- /opt/envhpc/modulefiles/.rhel7-eb/all/Compiler/GCCcore/11.2.0
---------------------------
   Autoconf/2.69                  PMIx/4.1.0                    hypothesis/6.14.6
   Autoconf/2.71          (D)     PROJ/8.1.0                    intltool/0.51.0
   Automake/1.16.2                Pango/1.48.8                  jbigkit/2.1
   Automake/1.16.4        (D)     Perl/5.32.1                   libGLU/9.0.2
   Autotools/20200321             Perl/5.34.0            (D)    libarchive/3.4.3
   Autotools/20210726     (D)     ProbABEL/0.5.0                libarchive/3.5.1    (D)
   BioPerl/1.7.8-Perl-5.34.0      PyOpenGL/3.1.5                libcerf/1.17
   BioPerl/1.7.8          (D)     Python/2.7.18-bare            libdrm/2.4.107
   Bison/3.7.1                    Python/2.7.18                 libevent/2.1.12
   Bison/3.7.6            (D)     Python/3.8.6                  libfabric/1.13.2
   Brotli/1.0.9                   Python/3.9.6-bare             libffi/3.4.2
   CMake/3.18.4                   Python/3.9.6           (D)    libgd/2.3.1
...
------------------------------------ /opt/envhpc/modulefiles/.rhel7-eb/all/Core ------------------------------------
   Bison/3.5.3                    M4/1.4.19                     gompi/2021b
   Bison/3.7.1                    Miniconda2/4.7.10             iimpi/2021b
   Bison/3.7.6                    Miniconda3/4.9.2              imkl/2021.4.0
   FastQC/0.11.8-Java-1.8         OpenSSL/1.1                   intel-compilers/2021.4.0
   FastQC/0.11.9-Java-11  (D)     PLINK/1.07-x86_64             intel/2021b
   GATK/3.8-0-Java-1.8.0_281      PLINK/1.9b_6.21-x86_64        ncurses/6.2
   GCC/10.2.0                     PLINK/2.00a2.3_x86_64  (D)    picard/2.25.5-Java-13
   GCC/11.2.0             (D)     SHAPEIT/2.r837.GLIBCv2.12     pkg-config/0.29.2
   GCCcore/10.2.0                 Trimmomatic/0.39-Java-1.8     popt/1.18
   GCCcore/11.2.0         (L,D)   binutils/2.35                 snpEff/4.3t-Java-1.8
   GCTA/1.94.0beta                binutils/2.36.1               tbl2asn/20200302-linux64
   Java/1.8.0_281         (1.8)   binutils/2.37                 zlib/1.2.11
   Java/11.0.2            (11)    flex/2.6.4
   Java/13.0.2            (D:13)  foss/2021b
```

Software de búsqueda con spider

Lmod tiene una herramienta muy útil para esto, el comando **spider** el cual proporciona la información necesaria para cargar un módulo.

- El software depende del nivel de CORE / COMPILER

```
$ module spider cmalt
```

```
  cMALT: cMALT/0.3.8-Java-1.8
----------------------------------------------------------------------------------------------------------------
    Description:
      The Genome Analysis Toolkit or GATK is a software package developed at the Broad
```

```
Institute to analyse
     ....
   You will need to load all module(s) on any one of the lines below before the "cMALT/0.3.8-
Java-1.8" module is available to load.

     GCCcore/11.2.0

   Help:
     Description
     ===========
     The Genome Analysis Toolkit or GATK is a software package developed at the Broad
Institute
   ...
```

Para cargar el módulo, deberá cargar todos los módulos enumerados

```
$ module load GCCcore/11.2.0 cMALT/0.3.8-Java-1.8
```

- Software depende de CORE / COMPILADOR / MPI

```
$ module spider eigen/3.4
```

```
  Eigen: Eigen/3.4.0
----------------------------------------------------------------------------------------------------
   Description:
     Eigen is a C++ template library for linear algebra: matrices, vectors, numerical solvers,
and related
     algorithms.


   You will need to load all module(s) on any one of the lines below before the "Eigen/3.4.0"
module is available to load.

     GCC/11.2.0  OpenMPI/4.1.1
     GCCcore/11.2.0

   Help:
   ...
```

Para cargar el módulo, deberá cargar todos los módulos enumerados en orden ***CORE / COMPILADOR / MPI / SOFTWARE***

```
$ module load GCCcore/11.2.0 GCC/11.2.0 OpenMPI/4.1.1 Eigen/3.4.0
```

- Otra opción es cargar **gompi/2021b**.

gompi significa *GNU Open MPI* y como pueden ver, ahora aparece todo el software compilado bajo GCC y OpenMPI.

```
$ module load gompi/2021b
$ module av
```

```
-------------------------/opt/envhpc/modulefiles/.rhel7-eb/all/MPI/GCC/11.2.0/OpenMPI/4.1.1
-------------------------
   ATLAS/0.9.9                    Infernal/1.1.4                    SciPy-bundle/2021.10
   AdmixTools/7.0.2               Kraken2/2.1.1                     XML-LibXML/2.0132-
Perl-5.32.1
   Armadillo/10.5.3               LAMPLD/1.0                        XML-LibXML/2.0132-
Perl-5.34.0
   Arrow/6.0.0                    MAFFT/7.487-with-extensions       arpack-ng/3.8.0
   BLAST+/2.11.0                  MitoZ/2.3-Python-3.9.6            arrow-R/6.0.0.2-R-4.1.2
   Beast/2.5.1                    MrBayes/3.2.7                     beagle-lib/3.0.2
   Biopython/1.79-Python-3.9.6    Ninja/1.10.1                      git/2.33.1                (D)
   ....
---------------------------- /opt/envhpc/modulefiles/.rhel7-eb/all/Compiler/GCC/11.2.0
------------------------------
   BCFtools/1.10.2        GCTA/1.94.0beta          (D)    SAMtools/1.14
   BCFtools/1.12   (D)    GEOS/3.9.1                      Shapely/1.8.0
   BEDTools/2.30.0        GSL/2.7                         SpectrA/0.9.0
   BLIS/0.8.1             HTSlib/1.14                     SpectrA/1.0.0             (D)
   BWA/0.7.17             LAPACK/3.9.1                    VCFtools/0.1.16
   BamTools/2.5.2         LAPACK/3.10.1            (D)    XML-LibXML/2.0132-Perl-5.32.1
   ....
--------------------------- /opt/envhpc/modulefiles/.rhel7-eb/all/Compiler/GCCcore/11.2.0
---------------------------
   Autoconf/2.69                  PMIx/4.1.0                (L)    hypothesis/6.14.6
   Autoconf/2.71           (D)    PROJ/8.1.0                       intltool/0.51.0
   Automake/1.16.2                Pango/1.48.8                     jbigkit/2.1
   Automake/1.16.4         (D)    Perl/5.32.1                      libGLU/9.0.2
   Autotools/20200321             Perl/5.34.0               (D)    libarchive/3.4.3
   Autotools/20210726      (D)    ProbABEL/0.5.0                   libarchive/3.5.1    (D)
   ....
------------------------------------ /opt/envhpc/modulefiles/.rhel7-eb/all/Core -------------------------------------
   ADMIXTURE/1.3.0                M4/1.4.18                 gettext/0.21
   Bison/3.5.3                    M4/1.4.19                 gompi/2021b          (L)
   Bison/3.7.1                    Miniconda2/4.7.10         iimpi/2021b
   Bison/3.7.6                    Miniconda3/4.9.2          imkl/2021.4.0
```

Significado de algunos módulos importantes

- **GCC/11.2.0**:

GCC/11.2.0 define el módulo que agrupa todo el CORE de GCC.

- **GCCCore/11.2.0**:

GCCCore/11.2.0 define el CORE de GCC y todo el software compilado con él.

- **foss/2021b**

El módulo foss consiste en su totalidad en el término común *FOOS*, que es la abreviatura de *Free and Open source software*.

Este módulo consta de los siguientes paquetes y se encarga de precargarlos.

- binutils
- Compilador GNU GCC (C), g++ (C++) and gfortran (Fortran)
- Librería Open MPI
- Librería FlexiBLAS con OpenBLAS + LAPACK como backend
- Librería ScaLAPACK
- Librería FFTW (Fourier transform (DFT))

```
$ module load foss/2021b
$ module list
Currently Loaded Modules:
  1) GCCcore/11.2.0   5) numactl/2.0.14      9) hwloc/2.5.0      13) libfabric/1.13.2  17) FlexiBLAS/3.0.4
  2) zlib/1.2.11      6) XZ/5.2.5           10) OpenSSL/1.1      14) PMIx/4.1.0        18) FFTW/3.3.10
  3) binutils/2.37    7) libxml2/2.9.10     11) libevent/2.1.12  15) OpenMPI/4.1.1     19) ScaLAPACK/2.1.0-fb
  4) GCC/11.2.0       8) libpciaccess/0.16  12) UCX/1.11.2       16) OpenBLAS/0.3.18   20) foss/2021b
```

- **gompi/2021b**:

El módulo *gompi* precarga todo el software compilado con GNU GCC y OpenMPI

```
$ module load gompi/2021b
$ module list
Currently Loaded Modules:
  1) GCCcore/11.2.0   5) numactl/2.0.14      9) hwloc/2.5.0      13) libfabric/1.13.2
  2) zlib/1.2.11      6) XZ/5.2.5           10) OpenSSL/1.1      14) PMIx/4.1.0
  3) binutils/2.37    7) libxml2/2.9.10     11) libevent/2.1.12  15) OpenMPI/4.1.1
  4) GCC/11.2.0       8) libpciaccess/0.16  12) UCX/1.11.2       16) gompi/2021b
```

- **intel/2021b**:

Este módulo común precarga los compiladores y librerías de Intel.

- Compiladores Intel C/C++/Fortran (icc, icpc and ifort)
- binutils
- GCC, el cual sirve como base de los compiladores de intel.
- Librería Intel MPI
- Librería Intel Math Kernel Library (MKL) para funcionalidades de BLAS/LAPACK/FFT

```
$ module load intel/2021b
$ module list
Currently Loaded Modules:
  1) GCCcore/11.2.0   3) binutils/2.37              5) numactl/2.0.14   7) impi/2021.4.0    9) imkl-
FFTW/2021.4.0
  2) zlib/1.2.11      4) intel-compilers/2021.4.0   6) UCX/1.11.2       8) imkl/2021.4.0   10) intel/
2021b
```