

## General information

### Modules

Based on the requirements by the DEEP-EST applications several software tools have already been installed on the DEEP system/DEEP-ER SDV. To see which modules are available use

```
module avail
```

For most tools and compilers there are several versions available, you can list the different versions of a specific tool with `module avail ?tool name?`, for example

```
-bash-4.1$ module avail intel

----- /usr/local/modulefiles/COMPILER -----
intel/13.0.4      intel/15.0      intel/16.0      intel/16.3      intel/17.1      intel/18.0
intel/14.0.3      intel/15.2.164  intel/16.2(default)  intel/17.0      intel/17.2      intel/18.1
```

You can load a specific version or just use the default with for example

```
module load intel
```

The modules which are currently loaded can be listed with the command `?module list?`. To unload a module just type `?module unload?` followed by the name of the module.

### Information on new software stack and lmod

Since the second half of 2018 a new software stack managed via Easybuild is available on the system (more information on Easybuild is available [?here](#).

**Currently, this stack is not the default one, but must be actively enabled by the users with this command:**

```
enable_new_stack
```

The software made available on the new stack via Easybuild is arranged hierarchically, with only some core modules made available to the user at login. Typically, these core modules include the compilers, or some other standalone tools (i.e. JUBE). Loading such core modules via `module load` allows the list of available modules to increase, including the modules that are installed with the fore-mentioned core module as a dependency.

A typical workflow would be the following. After enabling the new stack, the user would be in this situation:

```
$ module avail

----- /usr/local/software/haswell/Stages/2018a/modules/all/Compilers -----
Autotools/20170619  CMake/3.9.6      CMake/3.11.1      Doxygen/1.8.14    LLVM/5.0.1      Perl/5.26.1      Python/2.7.
Bazel/0.11.1        CMake/3.10.3     CMake/3.12.2 (D)   GMP/6.1.2         PAPI/5.6.0      PostgreSQL/10.3  Python/3.6.
----- /usr/local/software/haswell/Stages/2018a/UI/Compilers -----
Intel/2018.2.199-GCC-5.5.0

----- /usr/local/software/haswell/Stages/2018a/UI/Tools -----
Advisor/2018_update3  EasyBuild/3.6.1  JUBE/2.2.1      Java/1.8.0_162    VTune/2018_update3  intel-para/2018a-mt

Where:
D: Default Module
```

Loading the Intel compiler (Intel/2018.2.199-GCC-5.5.0 module), allows the list of available tools to be expanded:

```
module load Intel/2018.2.199-GCC-5.5.0
module avail

----- /usr/local/software/haswell/Stages/2018a/modules/all/Compiler/mpi -----
ParaStationMPI/5.2.1-1-mt  ParaStationMPI/5.2.1-1 (D)
```

```

----- /usr/local/software/haswell/Stages/2018a/modules/all/Compiler/in
mkl-dnn/0.13

----- /usr/local/software/haswell/Stages/2018a/modules/all/Compi
Autotools/20170619      CMake/3.9.6      CMake/3.11.1      Doxygen/1.8.14      LLVM/5.0.1      Perl/5.26.1      Python/2.7.
Bazel/0.11.1          CMake/3.10.3     CMake/3.12.2 (D)   GMP/6.1.2          PAPI/5.6.0      PostgreSQL/10.3   Python/3.6.

----- /usr/local/software/haswell/Stages/2018a/UI/Com
Intel/2018.2.199-GCC-5.5.0 (L)

----- /usr/local/software/haswell/Stages/2018a/UI/T
Advisor/2018_update3    EasyBuild/3.6.1    JUBE/2.2.1      Java/1.8.0_162    VTune/2018_update3    intel-para/2018a-mt

Where:
L:  Module is loaded
D:  Default Module

```

After further loading an MPI library, the user can then access all parallel tools that require MPI:

```

module load ParaStationMPI/5.2.1-1
module avail

----- /usr/local/software/haswell/Stages/2018a/modules/all/MPI/intel/2018.2
Boost/1.66.0-Python-2.7.14      Extrae/3.5.2      FTI/1.2      HDF5/1.8.20      PETSc/3.9.0_int8      R/3.4.3
Boost/1.66.0-Python-3.6.5 (D)   FFTW/3.3.7      GSL/2.4      PETSc/3.9.0_complex      PETSc/3.9.0      (D)      SIONlib/1.7.

----- /usr/local/software/haswell/Stages/2018a/modules/all/Compiler/mpi/
ParaStationMPI/5.2.1-1-mt      ParaStationMPI/5.2.1-1 (L,D)

----- /usr/local/software/haswell/Stages/2018a/modules/all/Compiler/in
mkl-dnn/0.13

----- /usr/local/software/haswell/Stages/2018a/modules/all/Compi
Autotools/20170619      CMake/3.9.6      CMake/3.11.1      Doxygen/1.8.14      LLVM/5.0.1      Perl/5.26.1      Python/2.7.
Bazel/0.11.1          CMake/3.10.3     CMake/3.12.2 (D)   GMP/6.1.2          PAPI/5.6.0      PostgreSQL/10.3   Python/3.6.

----- /usr/local/software/haswell/Stages/2018a/UI/Com
Intel/2018.2.199-GCC-5.5.0 (L)

----- /usr/local/software/haswell/Stages/2018a/UI/T
Advisor/2018_update3    EasyBuild/3.6.1    JUBE/2.2.1      Java/1.8.0_162    VTune/2018_update3    intel-para/2018a-mt

Where:
L:  Module is loaded
D:  Default Module

```

It can be seen from the previous examples that with the new software stack, the command `module avail` does not allow the user to visualize all the modules actually installed on the system, but rather the ones made available by the system with the modules previously loaded up to that point. In order to explore the full hierarchy of modules, the command

```
module spider
```

must be used. After enabling the new stack on a new shell, it's possible to search for a certain module (Extrae, for instance) with the following instructions:

```

$ ml spider Extrae

-----

Extrae: Extrae/3.5.2

```

**Description:**

Extræe is the core instrumentation package developed by the Performance Tools group at BSC. Extræe is capable of instrumenting applications based on MPI, OpenMP, pthreads, CUDA1, OpenCL1, and StarSsl using different instrumentation approaches. The information gathered by Extræe typically includes timestamped events of runtime calls, performance counters and source code references. Besides, Extræe provides its own API to allow the user to manually instrument his or her application.

You will need to load all module(s) on any one of the lines below before the "Extræe/3.5.2" module is available to load

```
Intel/2018.2.199-GCC-5.5.0 ParaStationMPI/5.2.1-1
```

**Help:****Description**

=====

Extræe is the core instrumentation package developed by the Performance Tools group at BSC. Extræe is capable of instrumenting applications based on MPI, OpenMP, pthreads, CUDA1, OpenCL1, and StarSsl using different instrumentation approaches. The information gathered by Extræe typically includes timestamped events of runtime calls, performance counters and source code references. Besides, Extræe provides its own API to allow the user to manually instrument his or her application.

**More information**

=====

- Homepage: <http://www.bsc.es/computer-sciences/performance-tools>
- Site contact: [sc@fz-juelich.de](mailto:sc@fz-juelich.de)

The description provided by the module `spider` command includes also a list of modules that need to be loaded, in order to be able to load the Extræe module.

The module `spider` command is part of a new implementation of the generic module. Together with this added functionality, `lmod` provides also a wrapper around the `module` command, namely the `ml` command. This wrapper has some user-friendly features:

- `ml` alone lists all the currently loaded modules;
- `ml +module` (or `ml module`) allows to load a specific module (e.g. `ml Intel` loads the default version of the Intel module);
- `ml -module` unloads the previously loaded module (e.g. `ml -Intel` removes the currently loaded version of the Intel module);
- the `ml` wrapper can be used instead of `module` for all the subcommands of `module` (e.g. `ml avail` or `ml av`, or `ml restore`, etc.)

More information on `lmod` can be found [?here](#).

## Compiler (relevant for old stack only!)

There are several compilers available, but as it is highly recommended to use the Intel compiler on the KNC system it might be best to also use it on the DEEP system.

Installed compilers:

- Intel compiler: `module load intel`
- GNU compiler: `module load gcc`
- PGI Compiler: `module load pgi`

## Profiling and analysis tools (relevant for old stack only!)

- Intel VTune: `module load VTune`
- Intel Advisor: `module load Advisor`
- Intel Inspector: `module load Inspector`
- ScoreP: `module load scorep`
- Darshan: `module load darshan`
- Extræe: `module load UNITE extræe`

- Scalasca: module load UNITE scalasca

## **MPI programs**

It is recommended to use the [ParaStation?](#) MPI installation.