Table of Contents

Information on software stack and using modules **2**
General information 2
List of available modules 4
Using Cuda 4
Testing oneAPI 4
MPI programs - compilation with ParaStation MPI 4
Using the old software stacks (for SDV Hardware) **5**
Information relative to the legacy stack only 5
Compilers 5
Profiling and analysis tools 5

Information on software stack and using modules

General information

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). It provides a `module` command for loading software environments. To see which modules are available use

```
module avail
```

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 logging in, the user would be in this situation:

```
$ module avail

------------------------------------------------------------------------------------------------ /usr/local/software/sk
  Autotools/20180311                        CubeWriter/4.4.2          Keras/2.2.4-GPU-Python-3.6.8        Ninja/1.9.0
  Bazel/0.20.0                              Doxygen/1.8.15            LLVM/8.0.0                          PAPI/5.7.0
  CMake/3.14.0                              GEOS/3.7.1-Python-3.6.8   MPFR/4.0.2                          Perl/5.28.1
  Clang/8.0.0-GCC-8.3.0-CUDA-10.1.105 (g)   GMP/6.1.2                 Mesa/19.0.1                         PostgreSQL/11.
  CubeGUI/4.4.3                             Graphviz/2.40.1           Meson/0.50.0-Python-3.6.8           PyTorch/1.1.0-
  CubeLib/4.4.3                             HDF5/1.10.5-serial        NCCL/2.4.6-1-CUDA-10.1.105   (g)    Python/2.7.16

------------------------------------------------------------------------------------------------------- /usr/local/
  GCC/8.3.0    Intel/2019.3.199-GCC-8.3.0    PGI/19.3-GCC-8.3.0

------------------------------------------------------------------------------------------------------------------ /usr/loca
  Advisor/2019_update3        EasyBuild/3.8.1        Inspector/2019_update3    Java/1.8         VTune/2019_update3
  CUDA/10.1.105        (g)    EasyBuild/3.9.1 (D)    JUBE/2.2.2                UCX/1.6.1 (D)    cuDNN/7.5.1.10-CUDA-10.1.1

-------------------------------------------------------------------------------------------------------------------------
  extoll    openmpi/1.6.1_extoll

 Where:
  g:  built for GPU
  D:  Default Module

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

It is recommended to use the Intel compiler and tool chain on the DEEP-EST system unless there are good reasons to move to the GCC or PGI compilers. Loading the Intel compiler (Intel/2019.3.199-GCC-8.3.0), allows the list of available tools to be expanded:

```
$ module load Intel/2019.3.199-GCC-8.3.0
$ module avail

------------------------------------------------------------------------------------ /usr/local/software/skylake/Sta
  IntelMPI/2019.3.199    ParaStationMPI/5.2.2-1-mt    ParaStationMPI/5.2.2-1 (D)

------------------------------------------------------------------------------------ /usr/local/software/skylake/S
  GSL/2.5    HDF5/1.10.5-serial (D)

------------------------------------------------------------------------------------------------ /usr/local/software/sk
  Autotools/20180311                        CubeWriter/4.4.2          Keras/2.2.4-GPU-Python-3.6.8        Ninja/1.9.0
  Bazel/0.20.0                              Doxygen/1.8.15            LLVM/8.0.0                          PAPI/5.7.0
  CMake/3.14.0                              GEOS/3.7.1-Python-3.6.8   MPFR/4.0.2                          Perl/5.28.1
  Clang/8.0.0-GCC-8.3.0-CUDA-10.1.105 (g)   GMP/6.1.2                 Mesa/19.0.1                         PostgreSQL/11.
  CubeGUI/4.4.3                             Graphviz/2.40.1           Meson/0.50.0-Python-3.6.8           PyTorch/1.1.0-
```

```
  CubeLib/4.4.3                              HDF5/1.10.5-serial         NCCL/2.4.6-1-CUDA-10.1.105   (g)    Python/2.7.16

------------------------------------------------------------------------------------------------------------------- /usr/local/
  GCC/8.3.0    Intel/2019.3.199-GCC-8.3.0 (L)    PGI/19.3-GCC-8.3.0

------------------------------------------------------------------------------------------------------------------- /usr/loca
  Advisor/2019_update3        EasyBuild/3.8.1        Inspector/2019_update3    Java/1.8         VTune/2019_update3
  CUDA/10.1.105        (g)    EasyBuild/3.9.1 (D)    JUBE/2.2.2                UCX/1.6.1 (D)    cuDNN/7.5.1.10-CUDA-10.1.1

------------------------------------------------------------------------------------------------------------------------------
  extoll    openmpi/1.6.1_extoll

 Where:
  g:  built for GPU
  L:  Module is loaded
  D:  Default Module
```

To see which modules are currently loaded, use the `module list` command.

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 logging in to a new shell, it's possible to search for a certain module (Extrae, for instance) with the following instructions:

```
$ module spider Extrae

------------------------------------------------------------------------------------------------------------------------------
 Extrae:
------------------------------------------------------------------------------------------------------------------------------
   Description:
     Extrae is the core instrumentation package developed by the Performance Tools group at BSC. Extrae is capable of inst
     applications based on MPI, OpenMP, pthreads, CUDA1, OpenCL1, and StarSs1 using different instrumentation approaches.
     information gathered by Extrae typically includes timestamped events of runtime calls, performance counters and sourc
     references. Besides, Extrae provides its own API to allow the user to manually instrument his or her application.

    Versions:
       Extrae/3.7.0
       Extrae/3.7.1
```

To get information on how to load a certain module including a list of modules that need to be loaded, in order to be able to load a certain Extrae version, you have to use the `module spider` command with that version:

```
$ module spider Extrae/3.7.0

------------------------------------------------------------------------------------------------------------------------------
 Extrae: Extrae/3.7.0
------------------------------------------------------------------------------------------------------------------------------
   Description:
     Extrae is the core instrumentation package developed by the Performance Tools group at BSC. Extrae is capable of inst
     applications based on MPI, OpenMP, pthreads, CUDA1, OpenCL1, and StarSs1 using different instrumentation approaches.
     information gathered by Extrae typically includes timestamped events of runtime calls, performance counters and sourc
     references. Besides, Extrae 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 "Extrae/3.7.0" module is available to load
```

```
    Intel/2019.3.199-GCC-8.3.0  ParaStationMPI/5.2.2-1
    Intel/2019.3.199-GCC-8.3.0  ParaStationMPI/5.2.2-1-mt

  Help:

    Description
    ===========
    Extrae is the core instrumentation package developed by the Performance Tools
     group at BSC. Extrae is capable of instrumenting applications based on MPI, OpenMP, pthreads,
     CUDA1, OpenCL1, and StarSs1 using different instrumentation approaches. The information gathered
     by Extrae typically includes timestamped events of runtime calls, performance counters and source
     code references. Besides, Extrae 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
```

The `module spider` command is part of a new implementation of the old `module` command, namely `lmod`, which is provided with Easybuild. Together with this added functionality, lmod provides also a wrapper around the `module` command: 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.

List of available modules

A list of the modules currently available on the Easybuild stack of the DEEP System can be found ?here. In addition to the commpiler toolchains (Intel, GCC, PGI) there are several tools and packages available for different fields of applications including machine and deep learning algorithms.

Using Cuda

For using Cuda you have to load the correspondant module which will set up the correct CUDA environment (including the nvidia driver):

```
module load CUDA
```

For using Cuda with ParaStationMPI, see also here.

Testing oneAPI

Intel's oneAPI programming model can be tested on the prototype system. Currently the beta 5 version is installed in `/usr/local/intel/oneapi/inteloneapi`. There is no module avaible, please use instead

```
source /usr/local/intel/oneapi/inteloneapi/setvars.sh
```

See the oneAPI ?programming guide to get started.

Currently, only the CPUs can be used as a device, but support for Stratix 10 (DAM) and Nvidia GPUs (DAM, ESB) are expected to become available in the next month. Intel HD integrated GPUs and Arria10 FPGAs can be tested in the ?Intel DevCloud.

MPI programs - compilation with ParaStation MPI

It is recommended to use the ParaStation MPI installation. When compiling software using the ParaStation MPI compiler wrappers, by default the Intel compiler will be used:

```
$ mpicc --version
icc (ICC) 19.0.3.199 20190206
Copyright (C) 1985-2019 Intel Corporation.  All rights reserved.
```

In order to use GCC rather than the Intel compiler, use one of the following:

```
$ mpicc -cc=gcc --version
gcc (GCC) 8.3.0
Copyright (C) 2018 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

$ MPICH_CC=gcc mpicc --version
gcc (GCC) 8.3.0
Copyright (C) 2018 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Using the old software stacks (for SDV Hardware)

As of July 2019, a new software stack compiled for Skylake is loaded by default on the DEEP-EST system.

The old Haswell stack (necessary to use the deeper-sdv nodes) can be enabled with the command:

```
enable_old_stack
```

If the old stack is needed in a batch script, the alias given above does not work. To circumvent this issue, please add the following lines to your script before calling the `enable_old_stack` command:

```
shopt -s expand_aliases
. /usr/local/software/configs/modules.sh
```

The legacy stack installed before the transition to Easybuild is still available on the system with the command:

`enable_legacy_stack`

As with the old Easybuild stack, the lines reported above must be included in your batch script to run the `enable_legacy_stack` command in a batch job.

Information relative to the legacy stack only

Compilers

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

- Intel VTune: module load VTune
- Intel Advisor: module load Advisor
- Intel Inspector: module load Inspector

- ScoreP: module load scorep
- Darshan: module load darshan
- Extrae: module load UNITE extrae
- Scalasca: module load UNITE scalasca