System usage

The DEEP-EST Data Analytics Module (DAM) can be used through the SLURM based batch system that is also used for (most of) the Software Development Vehicles (SDV). You can request DAM nodes (`dp-dam[01-16]`) with an interactive session like this:

```
srun -N 4 --tasks-per-node 2 -p dp-dam --time=1:0:0 --pty /bin/bash -i
[kreutz1@dp-dam01 ~]$ srun -n 8 hostname
dp-dam01
dp-dam01
dp-dam02
dp-dam02
dp-dam03
dp-dam03
dp-dam04
dp-dam04
```

When using a batch script, you have to adapt the partition option within your script: `--partition=dp-dam`

Using Cuda

To compile and run Cuda applications on the Nvidia V100 cards included in the DAM nodes, it is necessary to load the CUDA module:

```
[deamicis1@deepv ~]$ ml CUDA
[deamicis1@deepv ~]$ ml

Currently Loaded Modules:
 1) GCCcore/.8.3.0 (H)   2) binutils/.2.32 (H)   3) nvidia/.418.40.04 (H,g)   4) CUDA/10.1.105 (g)

 Where:
  g:  built for GPU
  H:             Hidden Module
```

Using FPGAs

Each node is equipped with a Stratix 10 FPGA. For getting started using OpenCL with the FPGAs you can find some hints as well as the slides and exercises from the Intel FPGA workshop held at JSC

```
/usr/local/fpga
```

It is recommended to do the first steps in an interactive session on a DAM node. To set up and check the FPGA environment, do the following:

```
source /usr/local/fpga/FPGA_init.sh
lspci | grep -i 'accel'
aocl list-devices
aoc -list-boards
# -- optional for doing the exercises:
# export CL_CONTEXT_EMULATOR_DEVICE_INTELFPGA=1
```

You can copy and untar the lab into your home directory to do the exercises step by step. The exercises use the emulator device instead of the actual FPGA device due to the long compilation time for the FPGAs. For using the FPGA device you will have to compile your OpenCL kernels using `-board=pac_s10_dc` option:

```
# compile for the emulator
aoc -march=emulator -fast-emulator kernel-file.cl

# compile for the FPGA device
aoc -board=pac_s10_dc kernel-file.cl
```

In addition, you will have to adapt the OpenCL host file to select the correct platform ("Intel® FPGA SDK for OpenCL™" or "Intel® FPGA Emulation Platform for OpenCL™ (preview)"). **Attention:** Compiling kernels for the FPGA device (instead of the emulator) might take several hours.

Although eclipse is available on the DAM nodes, compiling and running the example applications might not work out, so you have to fall back to the command line as described in the exercise manual and by using the provided `simple_compile.sh` scripts.

Filesystems and local storage

The home filesystem on the DEEP-EST Cluster Module is provided via GPFS/NFS and hence the same as on (most of) the remaining compute nodes. The local storage system of the DAM running BeeGFS is available at

```
/work
```

The file servers are reachable through the 40 GbE interface of the DAM nodes.

This is NOT the same storage being used on the DEEP-ER SDV system. Both, the DEEP-EST prototype system and the DEEP-ER SDV have their own local storage.

It's possible to access the local storage of the DEEP-ER SDV (`/sdv-work`), but you have to keep in mind that the file servers of that storage can just be accessed through 1 GbE ! Hence, it should not be used for performance relevant applications since it is much slower than the DEEP-EST local storages mounted to `/work`.

There is node local storage available for the DEEP-EST DAM node (2 x 1.5 TB NVMe SSD), but configuration is to be done for those devices.

Multi-node Jobs

Currently, multi-node MPI jobs are possible on the DAM only by modifying the environment in the following way:

```
$ ml Intel ParaStationMPI
$ env LD_LIBRARY_PATH=/opt/parastation/lib64:/opt/extoll/x86_64/lib/:${LD_LIBRARY_PATH} PSP_TCP=0 PSP_OPENIB=0 PSP_EXTOLL=
Hello World from processor dp-dam02, rank 1 out of 2
Hello World from processor dp-dam01, rank 0 out of 2
```

Attention: This is a temporary workaround.