Current state of running AI workloads on LUMI

Christian Schou Oxvig & René Løwe Jacobsen

LUMI User Support Team (LUST)

Danish e-infrastructure Consortium (DeiC)

DeiC

LUMI

June 1st, 2023

What are "AI workloads on LUMI"?

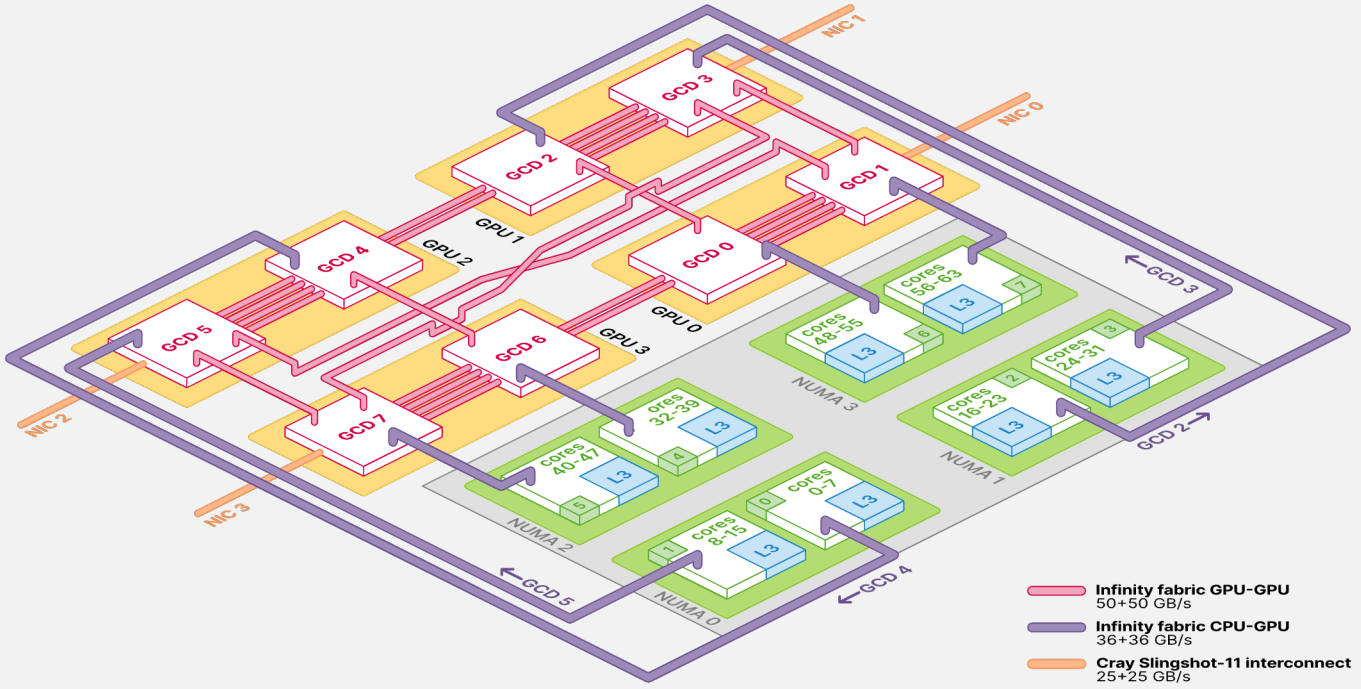
What is an "AI environment on LUMI"?

This talk:

Discussion of technical details related to the use of the LUMI-G GPUs for training deep learning models

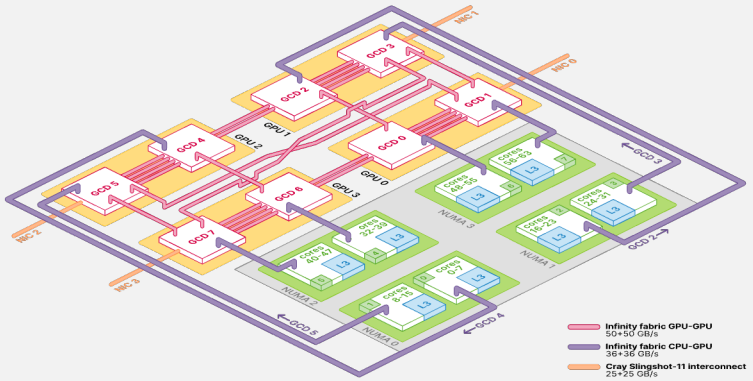
Node architecture

The LUMI-G node high level architecture



Using a single GPU node

- LUMI-G consists of 2560 nodes each with 4 AMD Instinct MI250X GPUs
- ROCm is the AMD equivalent of Nvidia's CUDA
  - ROCm is less mature than and not (yet) as feature rich as CUDA
  - ROCm support in popular deep learning frameworks is still immature/experimental/non-existing
  - For some applications the ROCm performance is inferior compared to CUDA
- Options for getting PyTorch/Tensorflow/JAX/etc. with ROCm support on LUMI
  - Compiling yourself is known to be notoriously difficult and sensitive to the ROCm version – questionable if this is ever going to be supported on LUMI
  - Installing as a pip package is discouraged since it may put too much stress on the Lustre file system
  - Using a Singularity/Apptainer container is likely going to be the recommended way on LUMI
    - The official AMD InfinityHub containers are outdated
    - The official AMD ROCm dockerhub containers are more up-to-date, but not very well documented
    - Building your own container on LUMI is (in general) not possible due to security concerns over enabling fakeroot / user namespaces
- Manual configuration and tuning is in general needed to make it run (and perform)
  - Setting environment variables
  - Correct bindings of CPU and GPU
    - https://docs.lumi-supercomputer.eu/runjobs/scheduled-jobs/distribution-binding/#gpu-binding
  - Sorting out hostlists
- Proof-of-concepts/examples of running PyTorch/Tensorflow on LUMI
  - https://docs.lumi-supercomputer.eu/software/packages/pytorch/
  - https://lumi-supercomputer.github.io/LUMI-EasyBuild-docs/p/PyTorch/
  - https://github.com/Lumi-supercomputer/ml-examples



Scaling to multiple GPU nodes

- Intra-node communication via RCCL (AMD equivalent of Nvidia's NCCL)
  - Supported via aws-ofi-rccl plugin provided by AMD
- Inter-node communication via Cray Slingshot 11 interconnect
  - Only supports Cray MPICH via libfabric/OFI (and Ethernet)
    - No (accelerated) OpenMPI/UCX (yet)
    - TCP/IP fallback (but that doesn't scale well)
- The way to go (most likely):
  - Use framework distribution mechanisms with RCCL(/NCCL) as backend, e.g. PyTorch DistributedDataParallel
  - Use the RCCL <--> libfabric integration provided by the aws-ofi-rccl plugin
- When using 3rd party distribution mechanisms (Horovod/DeepSpeed/Ray/...), you may need to use AMD ROCm forks and/or compile yourself against MPICH

What about the beginner and intermediate users?

- You may use cotainr on LUMI to easily create a Singularity/Apptainer container which is based on an official ROCm docker image and contains your conda/pip environment
- We are looking into ways to include the aws-ofi-rccl plugin
  - User installable via EasyBuild
  - Ship a LUMI/ROCm container (base) image that includes it (currently only done in the local CSC stack)
- We are looking into ways to provide default sane environment variables, slurm options, etc.