
LUX
PROVIDE
EXPLORING
THE FRONTIERS
OF DIGITAL INTELLIGENCE
MADE IN
LUXEMBOURG

LUX PROVIDE

MADE IN LUXEMBOURG

INFORMATION & COMMUNICATION TECHNOLOGIES LUXEMBOURG AWARDS 2022

ICT Business Partner of the Year

ISO 27001 CERTI TRUST

EuroHPC Joint Undertaking

European High Performance Computing Joint Undertaking

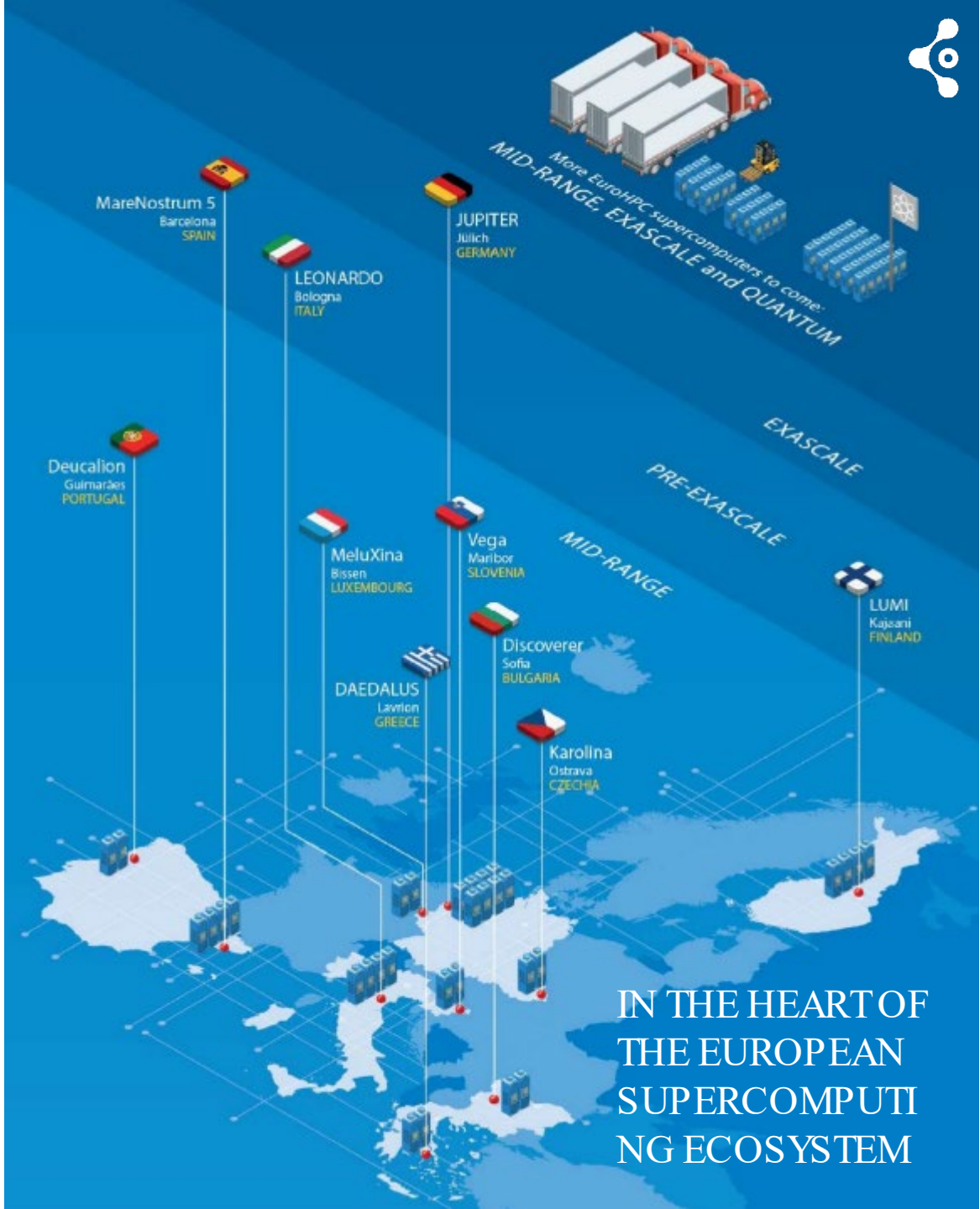
Over 8 billion Euro for cutting-edge supercomputing and quantum computing ecosystem in Europe

European Network of Competence Centers in HPC

United network of HPC actors in 33 European countries to foster adoption of HPC

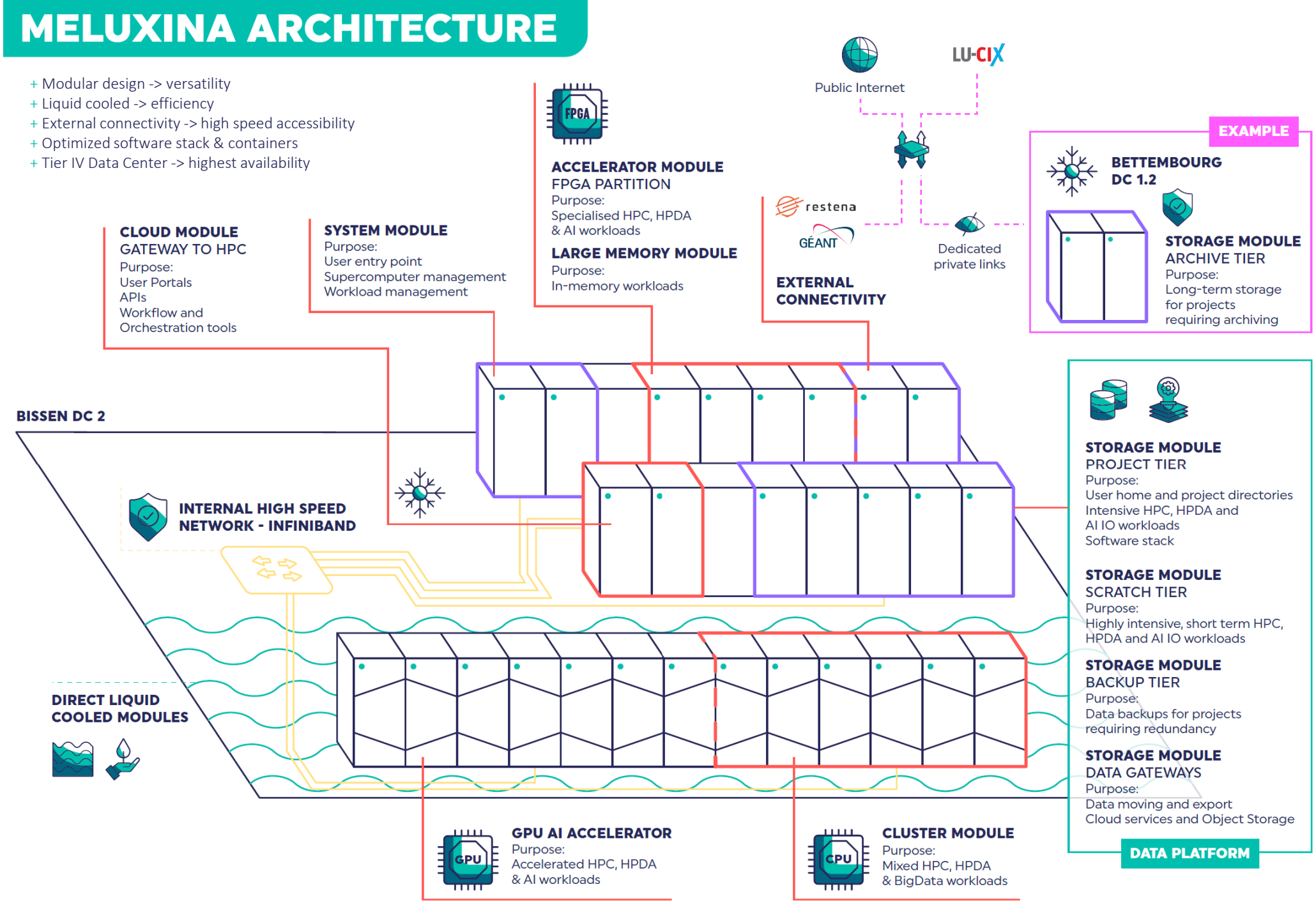
European Master for HPC

Pan-European Master's program by leading educational institutions focused on HPC

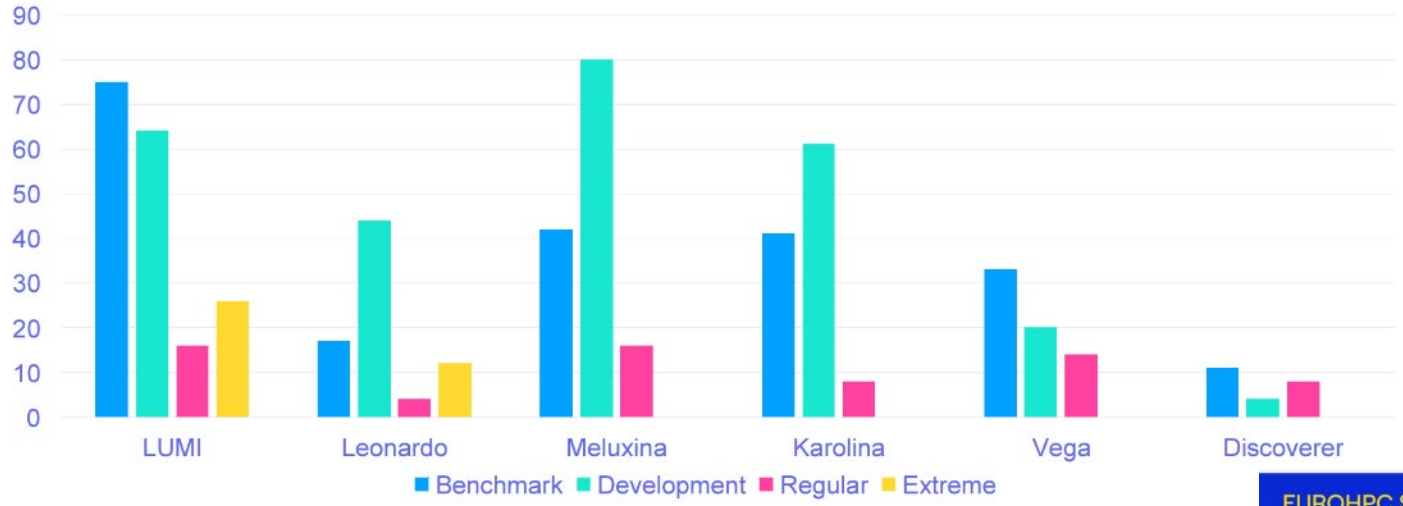


MELUXINA ARCHITECTURE

+ Modular design -> versatility
+ Liquid cooled -> efficiency
+ External connectivity -> high speed accessibility
+ Optimized software stack & containers
+ Tier IV Data Center -> highest availability



596 active projects distribution per system per type of EuroHPC JU access modes



Meluxina runs
25% of all EuroHPC JU projects

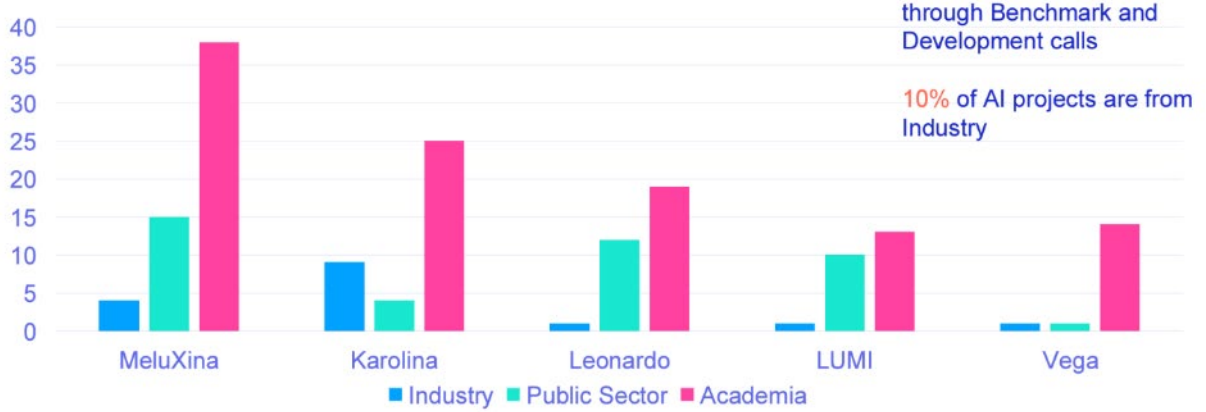
Meluxina runs
35% of all EuroHPC JU AI projects

EUROHPC SUMMIT 2024 ANTWERP 18-21 MARCH

167 active AI projects distribution per system per type of applicant

94% of projects were applied through Benchmark and Development calls

10% of AI projects are from Industry



(*) Leonardo was available for access since June 2023
(**) LUMI-G was on maintenances few times during the 2023 as it was newly installed

DeployAI

Mission

We will build an enhanced AIoD Platform with an EU-standard set of implementation-ready AI services and democratic access to cutting-edge AI technology.

Vision

We aim to foster business collaboration between the Industrial Community, Public Sector, and AI Product Developers to propel trustworthy innovation in the European market.

Partnerships

MELUXINA USER SOFTWARE ENVIRONMENT (MUSE)

Compilers, Languages & Performance Eng.

- AOCC
- GCC
- Intel
- NVIDIA HPC SDK
  - ✓ incl. PGI
- Support for various programming languages
  - ✓Python
  - ✓R
  - ✓Julia
  - ✓Go
  - ✓Rust
- Performance and debugging tools
  - ✓Intel
  - ✓NVIDIA
  - ✓ARM (Forge)
  - ✓Scalasca
  - ✓SCORE-P
  - ✓Extrae
  - ✓PAPI
  - ✓Valgrind
  - ✓GDB
  - ✓AMD-uProf
  - ✓ Nsight-Systems
  - ✓ Nsight-Compute
  - ✓ Vtune
  - ✓ gperf
  - ✓ extrap
  - ✓ Inspector
- Many build &support tools (Autotools, CMake, ..)

Parallelization tools, MPI suites & acceleration libraries

- OpenMPI
- Intel MPI
- ParaStationMPI
- NVHPC
- TBB
- PETSc
- KOKKOS
- cuBLAS
- cuFFT
- cuDNN
- NCCL
- TensorRT

Numerical &data libraries

- BLIS
- Intel MKL
- FFTW
- OpenBLAS
- ScalaPACK
- Boost
- Eigen
- ARPACK
- HDF5
- netCDF
- OpenCV
- CDO

Frameworks, runtime &platform tools

- PyTorch
- RAPIDSAI
- Torch Text+Vision
- TensorFlow + Hub
- Horovod
- Keras
- Theano
- Jupyter Lab
- Apache Spark
- Matlab Runtime
- dotNET Core + SDK
- Dakota
- Quantum Computing/AI
  - Cirq
  - QsimCirq
  - CuQuantum
  - Pennylane
  - Qiskit

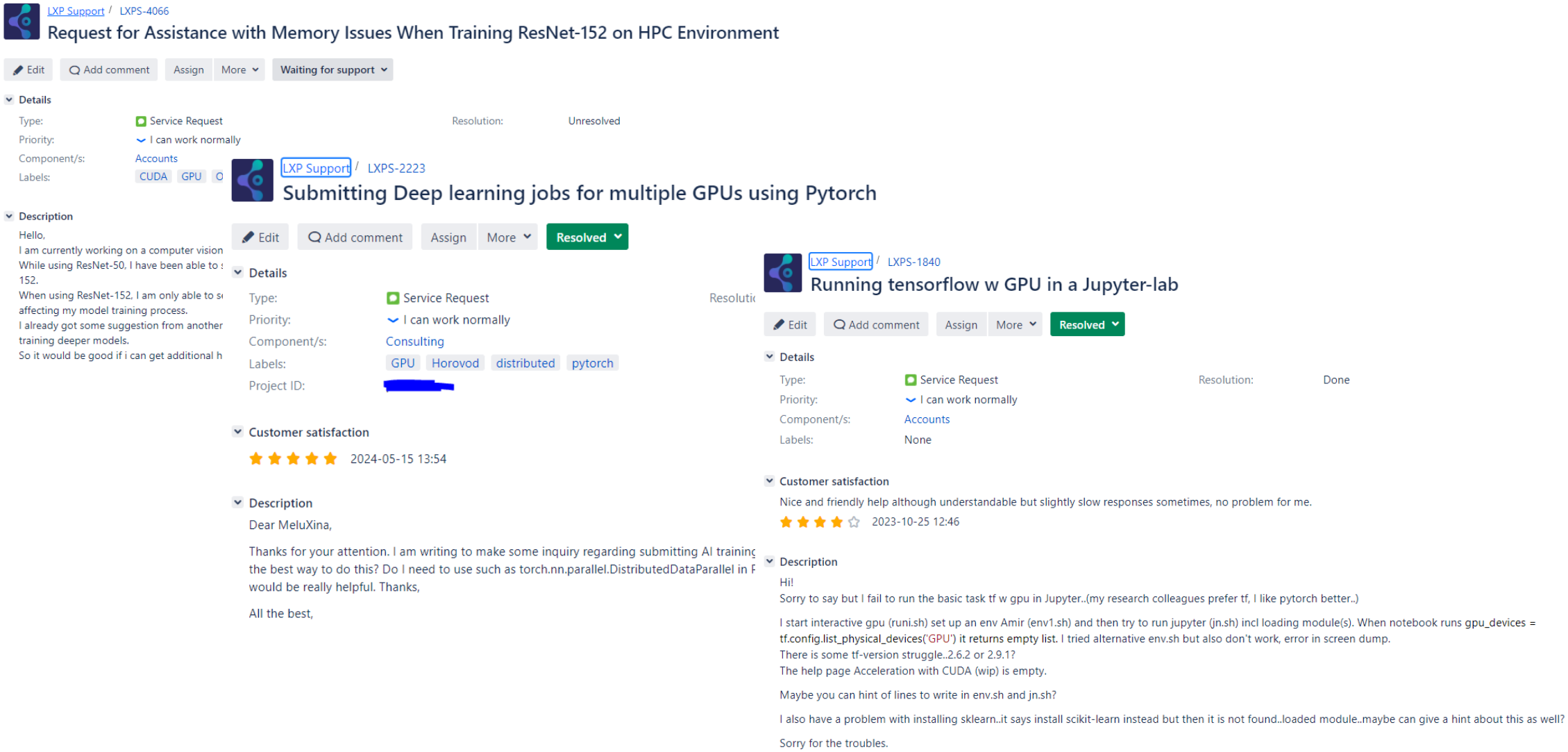
End user applications

- GROMACS
- OpenFOAM
- FOAM-Extend
- CP2K
- QuantumESPRESSO
- NAMD
- QMCPACK
- NWChem
- HOOMD-blue
- Freud-analysis
- DualSPHysics
- POV-Ray
- Blender
- MDAnalysis
- BioPython
- QUDA

- Visualisation
  - ✓ParaView
  - ✓VMD
  - ✓OVITO
  - ✓NCO
  - ✓ NCView

ISVs license can be implemented, we have 3 license servers dedicated for that

AI users



LXP Support / LXPS-4066

Request for Assistance with Memory Issues When Training ResNet-152 on HPC Environment

Edit | Add comment | Assign | More | Waiting for support

Details

Type: Service Request
Priority: I can work normally
Component/s: Accounts
Labels: CUDA GPU O
Resolution: Unresolved

Description

Hello,
I am currently working on a computer vision
While using ResNet-50, I have been able to s
152.
When using ResNet-152, I am only able to s
affecting my model training process.
I already got some suggestion from another
training deeper models.
So it would be good if i can get additional h

LXP Support / LXPS-2223

Submitting Deep learning jobs for multiple GPUs using Pytorch

Edit | Add comment | Assign | More | Resolved

Details

Type: Service Request
Priority: I can work normally
Component/s: Consulting
Labels: GPU Horovod distributed pytorch
Project ID: [redacted]
Resoluti

Customer satisfaction

2024-05-15 13:54

Description

Dear MeluXina,

Thanks for your attention. I am writing to make some inquiry regarding submitting AI training
the best way to do this? Do I need to use such as torch.nn.parallel.DistributedDataParallel in P
would be really helpful. Thanks,

All the best,

LXP Support / LXPS-1840

Running tensorflow w GPU in a Jupyter-lab

Edit | Add comment | Assign | More | Resolved

Details

Type: Service Request
Priority: I can work normally
Component/s: Accounts
Labels: None
Resolution: Done

Customer satisfaction

Nice and friendly help although understandable but slightly slow responses sometimes, no problem for me.
2023-10-25 12:46

Description

Hi!
Sorry to say but I fail to run the basic task tf w gpu in Jupyter..(my research colleagues prefer tf, I like pytorch better..)

I start interactive gpu (runi.sh) set up an env Amir (env1.sh) and then try to run jupyter (jn.sh) incl loading module(s). When notebook runs gpu_devices = tf.config.list_physical_devices('GPU') it returns empty list. I tried alternative env.sh but also don't work, error in screen dump.
There is some tf-version struggle..2.6.2 or 2.9.1?
The help page Acceleration with CUDA (wip) is empty.

Maybe you can hint of lines to write in env.sh and jn.sh?

I also have a problem with installing sklearn..it says install scikit-learn instead but then it is not found..loaded module..maybe can give a hint about this as well?

Sorry for the troubles.

AI tutorials

- Llama 3 inference using NVidia TensorRT and Triton
- Distributed training – Resnet50
- PyTorch inference of pre-trained model

LUX PROVIDE **MeluXina User Documentation** Search

MeluXina User Documentation
Welcome
MeluXina supercomputer
Gaining access
First steps
Application examples
HPC
HPDA
FPGA
How To
Use Llama 3 with NVIDIA TensorRT-LLM and Triton Inference Server
Distribute a Resnet50 training with PyTorch over several GPUs
Use VS Code to develop on Meluxina with ease
Use PyTorch to run the inference of a pre-trained mode
Containerization
Cloud
What's new
FAQ

Use Llama 3 with NVIDIA TensorRT-LLM and Triton Inference Server

Objective

This 30-minute tutorial strongly relies on the following technical article, i.e., Turbocharging Meta Llama 3 Performance with NVIDIA TensorRT-LLM and NVIDIA Triton Inference Server.

- The objective of this 30-minute tutorial is to show how to:
  - Start a Inference server such as the NVIDIA Triton Inference server on Meluxina
  - Use TensorRT-LLM to build TensorRT engines that contain state-of-the-art optimizations to perform inference efficiently on NVIDIA GPUs
  - Setup the Llama3 model as application case

Server-side (Meluxina)

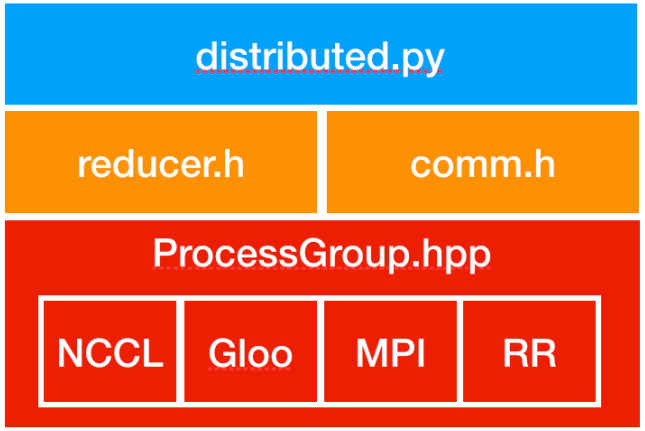
Setup

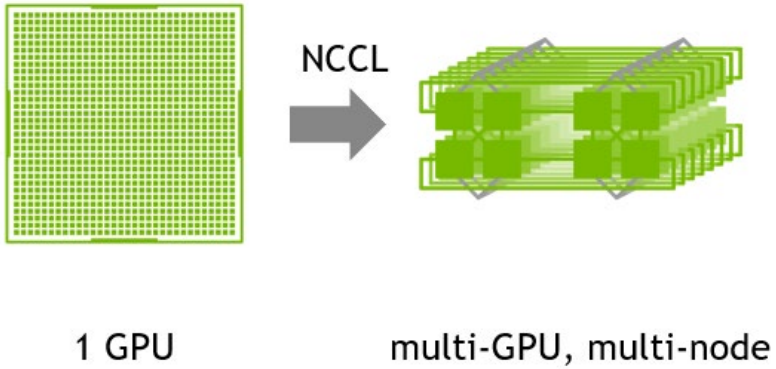
- Once connected to the machine, let us start from an empty directory: `mkdir Triton-30min && cd Triton-30min`
- We then take an interactive job on the `gpu` partition (see the below command)
- To avoid installing all dependencies required by both the NVIDIA TensorRT-LLM and the Triton

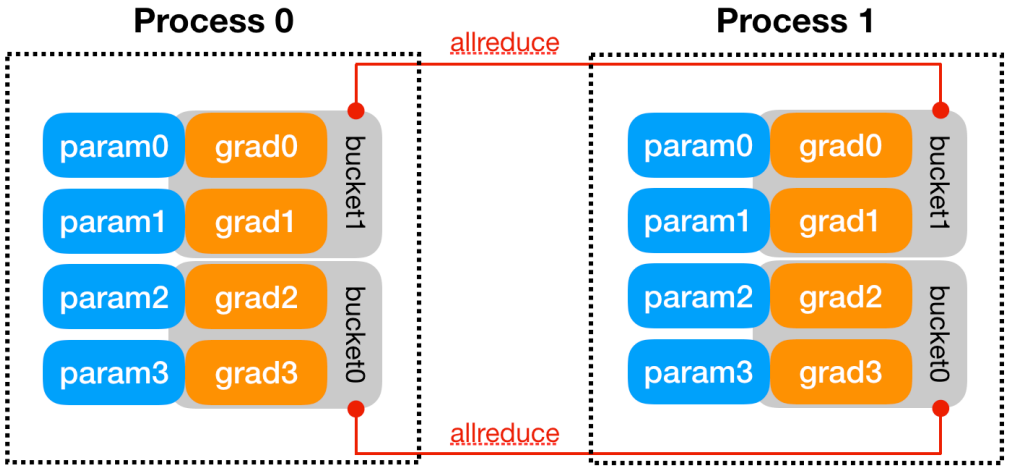
Table of contents
Objective
Server-side (Meluxina)
Setup
Using the TensorRT-LLM backend with Llama3
Using Llama3 with the Triton Inference Server
Prepare a slurm launcher script to start the Triton Inference Server
Changing the transaction policy
Retrieving the ssh command for port forwarding
Client-side (Local machine)
SSH forwarding
Small client in python

AI tutorials

Distributed training – Resnet50

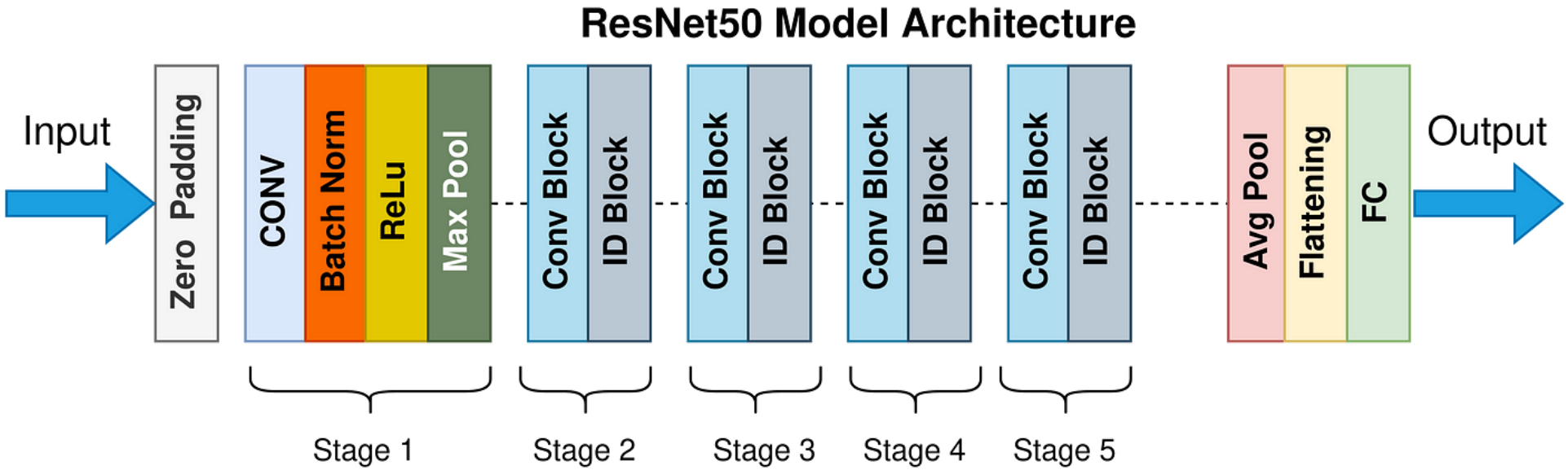




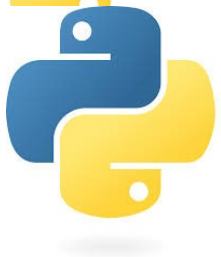


AI tutorials

Distributed training – Resnet50



get_port.py



resnet50_LXP.py

resnet50_LXP.py

AI tutorials

Distributed training – Resnet50

resnet50_LXP.py

```
#!/bin/bash -l
#SBATCH --job-name="restnetDistri"
#SBATCH --nodes=4
#SBATCH --ntasks-per-node=1
…
```

SLURM directives

```
module load env/staging/2023.1
module load PyTorch/2.1.2-foss-2023a-CUDA-12.1.1
module load torchvision/
…
```

Loading software

```
nodes=($(scontrol show hostnames $SLURM_JOB_NODELIST))
nodes_array=($nodes)
head_node=${nodes_array[0]}
head_node_ip=$(srun --nodes=1 --ntasks=1 -w "$head_node" hostname --ip-address)
…
```

Setting up distributed architecture

```
export NCCL_SOCKET_IFNAME=ib0
export NCCL_ASYNC_ERROR_HANDLING=1
export OMP_NUM_THREADS=8
…
```

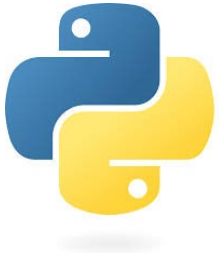
Enabling optimized communications

```
CUDA_VISIBLE_DEVICES="0,1,2,3" srun --cpus-per-task=8 --wait=60 --ntasks-per-node=1 --kill-on-bad-exit=1 torchrun --max_restarts 3 --nnodes ${SLURM_NNODES} --nproc_per_node ${NGPUS_PER_NODE} --rdzv_id 10000 --rdzv_backend c10d --rdzv_endpoint $endpoint --log_dir ${PWD}/log_torch resnet50_LXP.py 4 1 32
```

Run

AI tutorials

Distributed training – Resnet50



resnet50_LXP.py

```
def main(save_every: int,\
         total_epochs: int,\
             batch_size: int,\
                 snapshot_path: str = os.path.join(os.getcwd(), 'snapshot.pt')):
    world_size, rank = ddp_setup()
    batch_size_per_gpu = int(batch_size / 4 )
    dataset, model, optimizer = load_train_objs(total_epochs, batch_size_per_gpu, world_s
    train_data = prepare_dataloader(dataset, batch_size)
    trainer = Trainer(model, train_data, optimizer, save_every, snapshot_path)
    trainer.train(total_epochs)
    print('Will now destroy the process group\n')
    destroy_process_group()
    print('Process group destroyed')
    import sys
    # https://github.com/pytorch/pytorch/issues/76287
    sys.exit(0)
```

Setting up Distributed Data Parallel

Load training dataset

Prepare data loader w.r.t batch size

Instantiate trainer with model, data and optimizer

Launch distributed training

Troubleshooting (recent example from a real user)

- User reporting that his workload (face recognition software) is taking 2x longer than expected
- First investigations point to an issue with how user is launching the software with SLURM.
- Constant communication with end user
- Further investigations show that with minimal changes to code (1 line), and optimized pinning/binding, the user could get 8.8x performance improvement.

In the face_detector_deepsparse.py file, I modified again on line 34 the following code

with self._compiled_model = Engine(model=onnx_filepath, batch_size=batch_size)

self._compiled_model = Engine(model=onnx_filepath, batch_size=batch_size*4,num_cores=256,scheduler="multi_stream", num_streams=8)

I basically increase the batch size by 4, increase the num_cores to match the hyperthreaded cpu cores and fix the number of concurrent streams to match the numa domains, 8

and then with the following command to run it
NM_BIND_THREADS_TO_CORES=1 numactl --physcpubind=0-255 python3 pipeline_manager.py --mp4_dir ../mp4_25fps/ --saving_dir ../results/

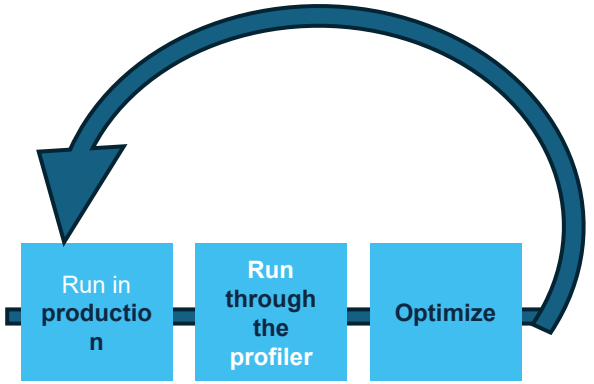
I manage to get it from **8min24 sec to 57 secs** on my 4 videos dataset.

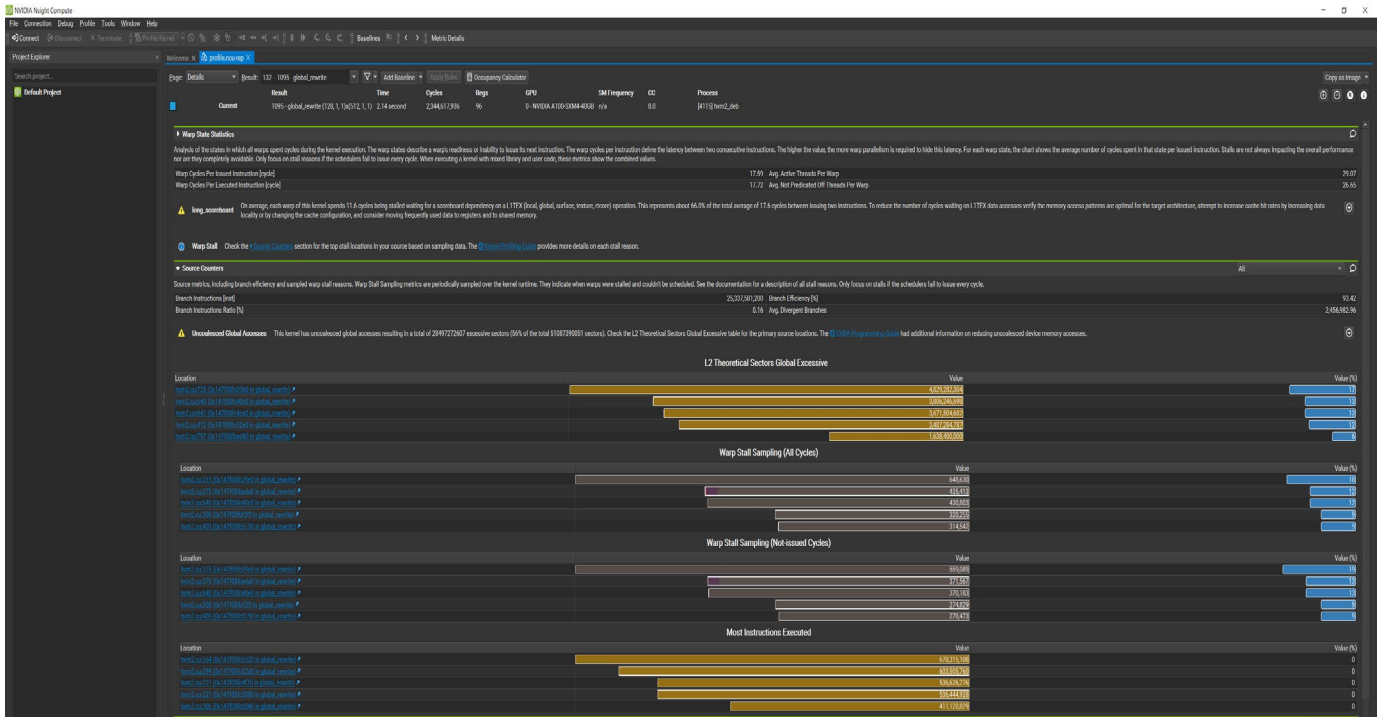
Extract from the ticket

Profiling

- Module load Linaro-forge
  - For DDT and MAP
- Module load NVHPC
  - For Nsight-systems and Nsight-compute

https://docs.lxp.lu/hpc/profiling/





Module	CYCLES_NOT_IN_HALT ▼	RETIRED_INST	IPC	CPI
ufs_weather_model	4855467000000	208528000000	1.90	0.53
libiomp5.so	2442263000000	894847000000	0.78	1.29
libpsm3-fi.so	759563000000	553557000000	2.05	0.49
libmpi.so.12.0.0	497695000000	837529000000	1.68	0.59
libpthread-2.28.so	192781000000	421877000000	2.19	0.46
libmlx5.so.1.23.40.0	53111000000	110212000000	2.08	0.48
libnetcdff.so.7.1.0	14603000000	26336000000	1.80	0.55
libc-2.28.so	3765000000	2803000000	0.74	1.34
vdso64.so	922000000	627000000	0.68	1.47
ld-2.28.so	463000000	311000000	0.67	1.49
libmpifort.so.12.0.0	295000000	175000000	0.59	1.69
libAMDMpitAgent.so	290000000	125000000	0.43	2.32
libstdc++.so.6.0.29	18000000	5000000	0.28	3.60
libucm.so.0.0.0	4000000	0	0.00	0.00
libibverbs.so.1.14.40.0	1000000	0	0.00	0.00

AI-oriented profilers: Scalene and NVidia DLProf

1. Links to the tools
   - **Scalene**: https://github.com/plasma-umass/scalene
   - **NVidia DLProf**: https://docs.nvidia.com/deeplearning/frameworks/dlprof-user-guide/index.html

2. Installing the tools

```
pip3 install scalene
pip3 install nvidia-pyindex
pip3 install nvidia-dlprof
pip3 install nvidia-dlprofviewer
```

AI-oriented profilers: Scalene

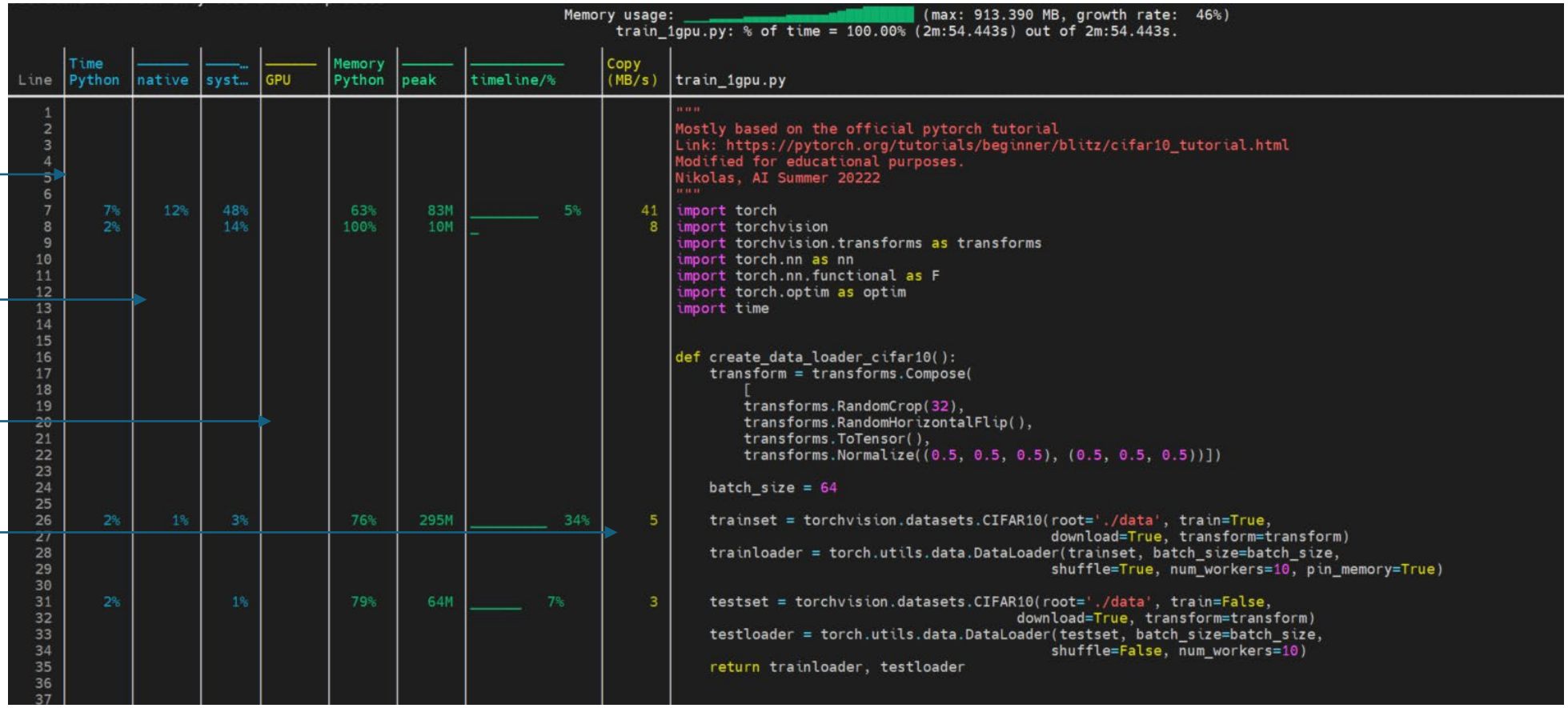
scalene python3 train_1gpu.py

Time spent in Python interpreter

Time spent in native code

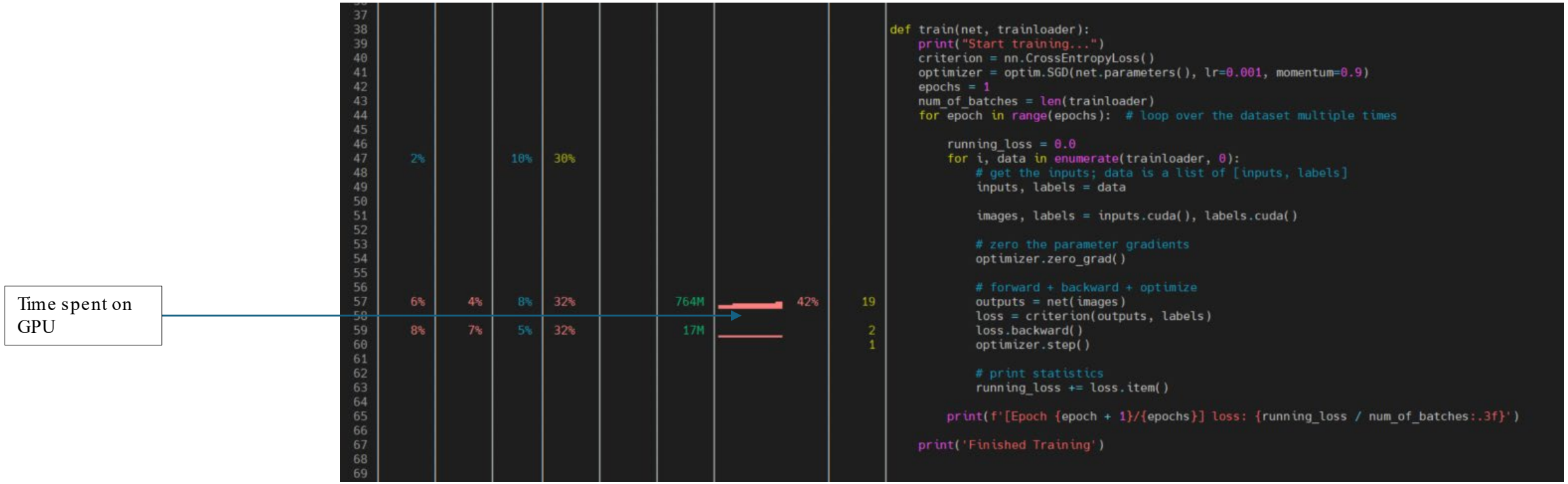
Time spent on GPU

Data movement

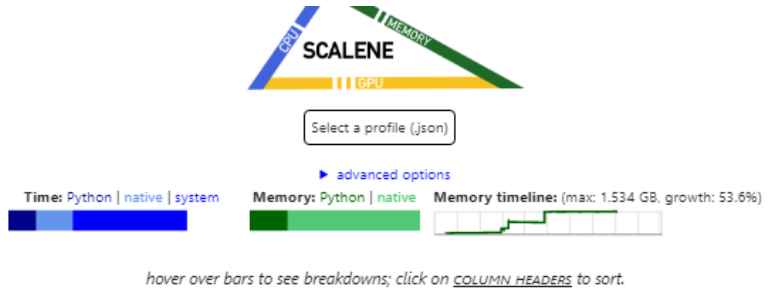


AI-oriented profilers: Scalene

Zooming on GPU usage

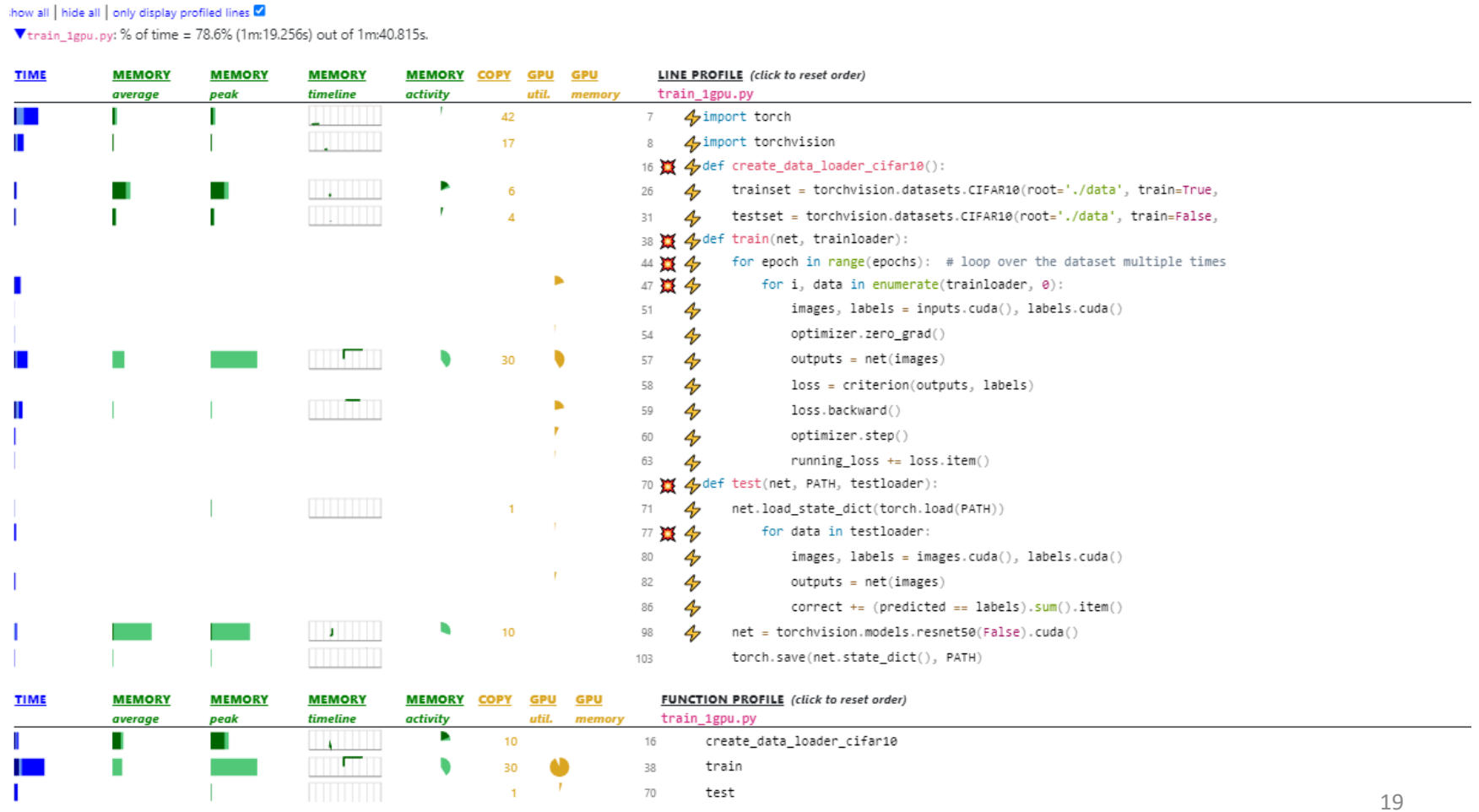


AI-oriented profilers: Scalene



scalene –json --outfile profile_scalene.json python3 train_1gpu.py

Using the scalene GUI (http://plasma-mass.org/scalene-gui/), You can view the .json profile and ask ChatGPT for some optimizations ☺



AI-oriented profilers: NVidia DLProf

Profiling with Nvidia DLProf requires some light modifications to the source code

1. Import the library

   import nvidia_dlprof_pytorch_nvtx as nvtx

2. Modify the main function
   add the following initialization code :

   nvtx.init(enable_function_stack=True)

   Wrap the train function with this code:

   with torch.autograd.profiler.emit_nvtx():

```
92  if __name__ == '__main__':
93      start = time.time()
94
95      import torchvision
96
97      PATH = './cifar_net.pth'
98      trainloader, testloader = create_data_loader_cifar10()
99      net = torchvision.models.resnet50(False).cuda()
100     nvtx.init(enable_function_stack=True)
101     start_train = time.time()
102     with torch.autograd.profiler.emit_nvtx():
103       train(net, trainloader)
104     end_train = time.time()
105     # save
106     torch.save(net.state_dict(), PATH)
107     # test
108     test(net, PATH, testloader)
109
110     end = time.time()
111     seconds = (end - start)
112     seconds_train = (end_train - start_train)
113     print(f"Total elapsed time: {seconds:.2f} seconds, \
```

AI-oriented profilers: NVidia DLProf

Once you have done the modifications, just run:

dlprof python3 train_1gpu.py

This will create a couple of files, two sqlite files and one nsys-rep.

```
[fbongiovanni@login02 pytorch-ddp]$ ls -lah *.sqlite
-rw-r-----. 1 fbongiovanni lxp 101M Mar 22 15:15 dlprof_dldb.sqlite
-rw-r--r--. 1 fbongiovanni lxp 650M Mar 22 15:12 nsys_profile.sqlite
[fbongiovanni@login02 pytorch-ddp]$ ls -lah *.nsys-rep
-rw-rw-r--. 1 fbongiovanni lxp 54M Mar 22 15:11 nsys_profile.nsys-rep
[fbongiovanni@login02 pytorch-ddp]$
```

AI-oriented profilers: NVidia DLProf

Once you have run the application with dlprof, you can use the dlprofviewer to view the reports.

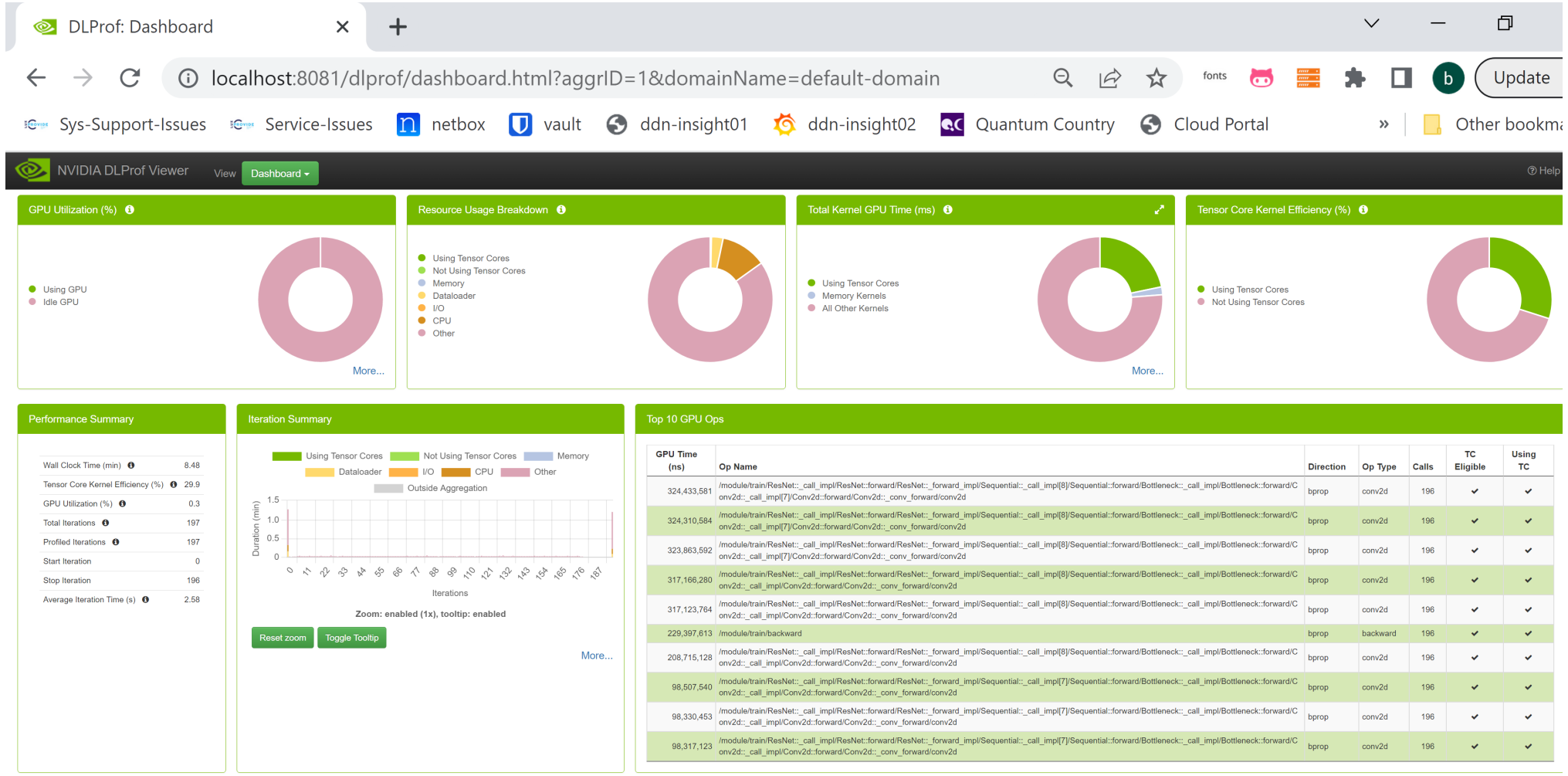
dlprofviewer -b 0.0.0.0 -p 8081 dlprof_dldb.sqlite

```
[fbongiovanni@mel2101 pytorch-ddp]$ dlprofviewer -b 0.0.0.0 -p 8081 dlprof_dldb.sqlite
[dlprofviewer-09:39:28 PM UTC] dlprofviewer running at http://mel2101:8081
```

This will launch a local web server on the node itself. Using a simple SSH tunnel, you can browse it locally. **Note: make sure to bind on 0.0.0.0 because by default it will launch it with localhost on the node itself.**

ssh –NL 8081:mel2101.meluxina.lxp.lu:8081 fbongiovanni@login.lxp.lu

AI-oriented profilers: NVidia DLProf



AI-oriented profilers: NVidia DLProf

DLProf also generates some good advice that you may want to take into account.

System Config

GPU Count	4
GPU Name(s)	NVIDIA A100-SXM4-40GB NVIDIA A100-SXM4-40GB NVIDIA A100-SXM4-40GB NVIDIA A100-SXM4-40GB
CPU Model	AMD EPYC 7452 32-Core Processor
GPU Driver Version	515.65.01
Framework	PyTorch 1.12.0
CUDA Version	11.7
NSys Version	2022.4.1.21-0db2c85
DLProf CLI Version	v1.8.0
DLProf DB Version	1.8.0
DLProf Viewer Version	1.8.0

Expert Systems

	Problem	Recommendation
»	2269 ops were eligible to use tensor cores but none are using FP16	Try enabling AMP (Automatic Mixed Precision). For more information: https://developer.nvidia.com/automatic-mixed-precision
	The following GPU device IDs are not being used: 1 2 3	Make sure to pass the necessary options to docker, the framework, and the model to use all available GPUs
	12.1% of the aggregated run was spent in the dataloader while not simultaneously running on the GPU	Focus on reducing time spent in the training data input process. This could be time spent in file reading, preprocessing and augmentation or file transfer. Set num_workers > 0 in the dataloader to enable asynchronous data loading. Consider using NVIDIA DALI, a library that is a high performance alternative to built-in data loaders and data iterators. Learn more here: https://developer.nvidia.com/DALI
»	The aggregated iteration range of 0 to 196 contains a lot of variation	Try limiting the iteration range to a steady range by rerunning with the --database option and setting --iter_start=27 --iter_stop=83
	Convolution operations were detected but torch.backends.cudnn.benchmark was not enabled.	Try setting torch.backends.cudnn.benchmark = True in your network. For best performance, the input shapes should be relatively stable.
	GPU Memory is underutilized: Only 12% of GPU Memory is used	Try increasing batch size by 4x to increase data throughput

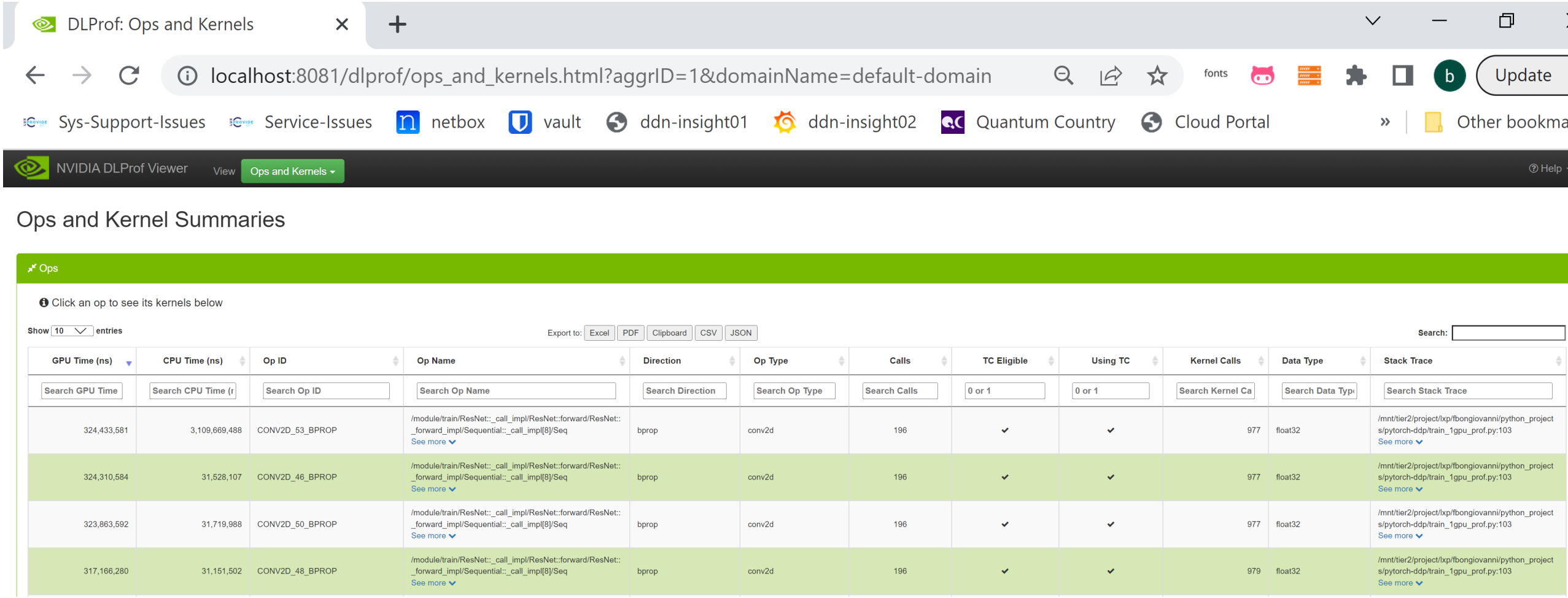
Guidance

Understanding GPU utilization and timing details of the operations is the first step in profiling your model.

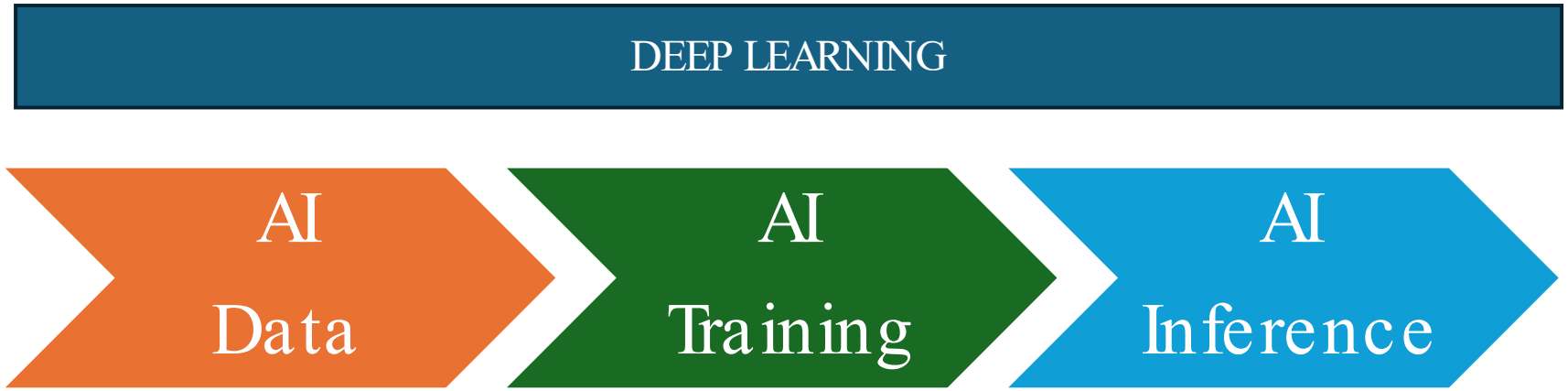
- To learn more about Tensor cores and Mixed Precision training, visit this site:https://developer.nvidia.com/tensor_cores
- You will find resources on how to train networks with mixed precision and make full use of Tensor cores for Tensorflow models here: https://docs.nvidia.com/deeplearning/sdk/mixed-precision-training/index.html#training_tensorflow
- Note that if there are multiple kernels being observed on single op, these are likely performing data transposes to prepare the data for efficient use by tensorcores. Such transposes themselves would not use tensor cores.

AI-oriented profilers: NVidia DLProf

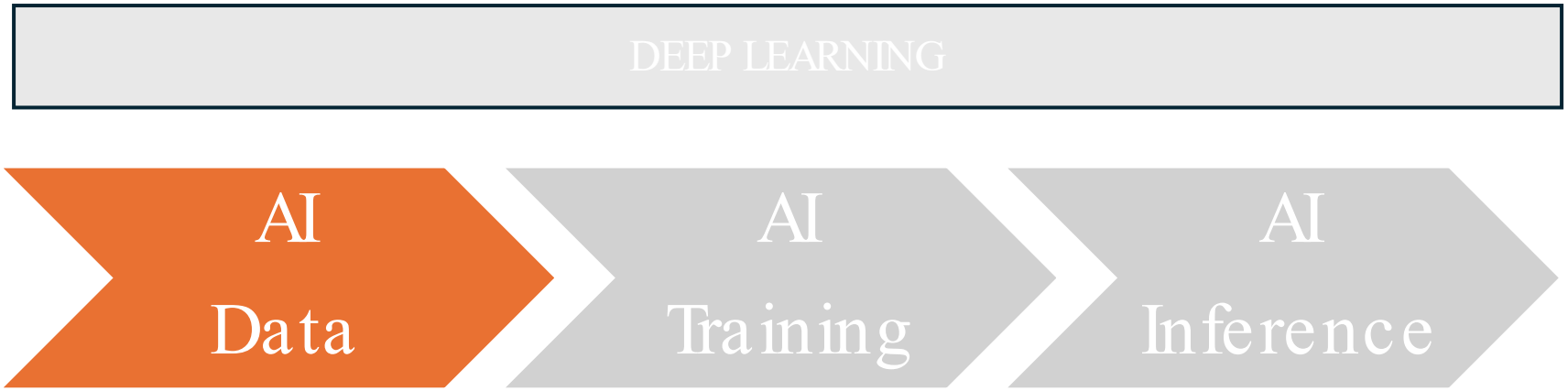
DLProf gives you a very detailed view on the ops in your model, how many times there were called…



Further optimizations

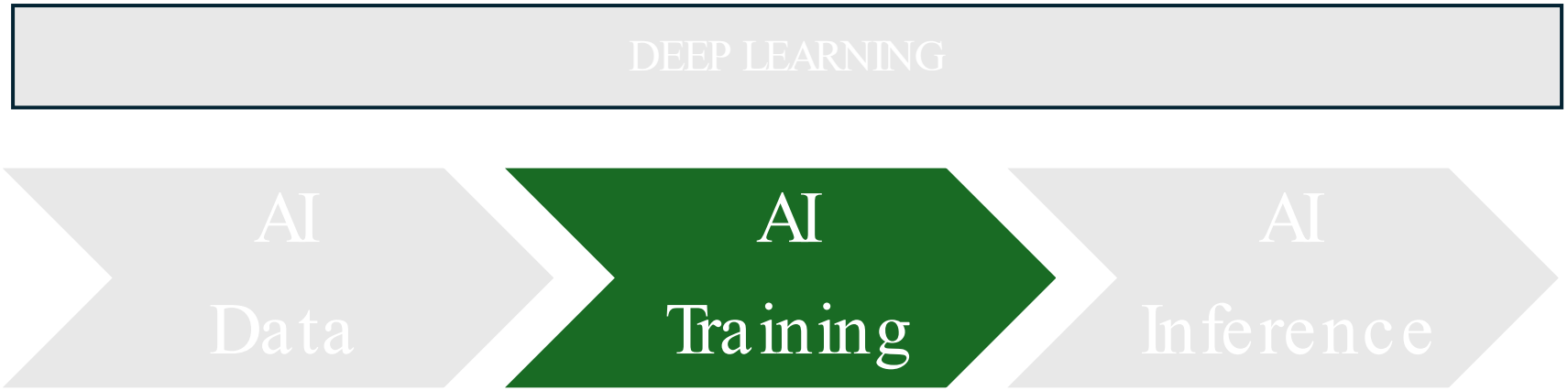


Further optimizations



- Lustre-friendliness
  - Avoid lots of small files
  - PyTorch + HDF5
    - https://github.com/mvsjober/pytorch-hdf5
    - https://blade6570.github.io/soumyatripathy/hdf5_blog.html
- Use of local disk on GPU nodes (each GPU node has one 1.92TB SSD disk)
- Use of Scratch tier (>400 GB/s in read/write throughput)

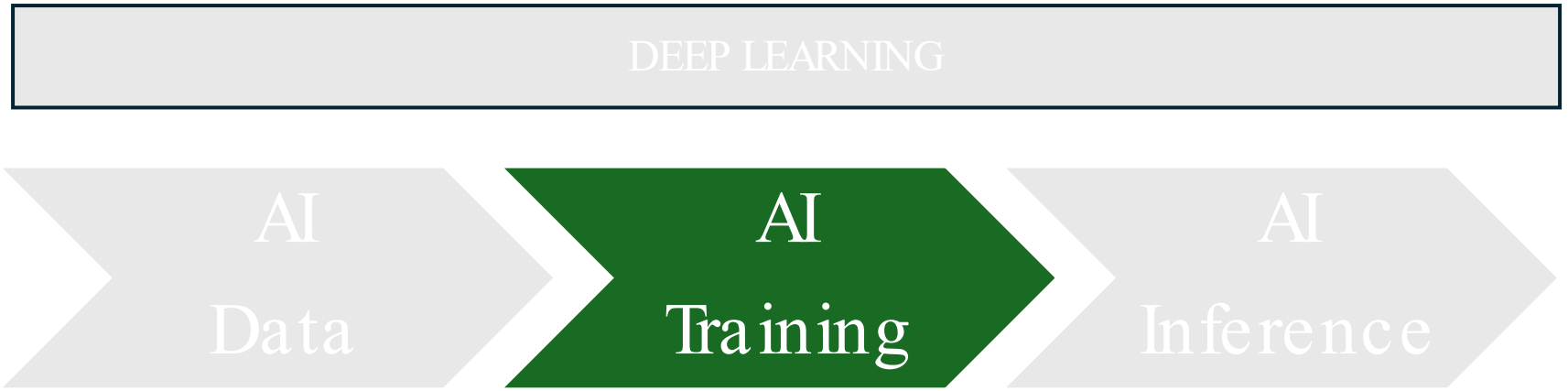
Further optimizations



TRANSFOMER model

- Embedding
- Attention
- Multi-Layer Perceptrons (MLPs)
- Unembedding

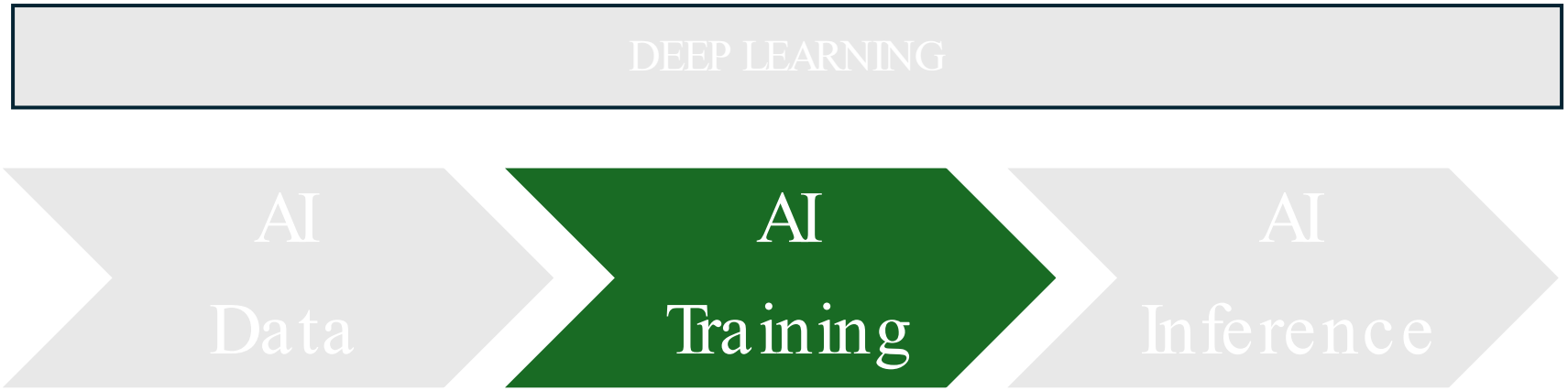
Further optimizations



TRANSFOMER model

- Embedding
- Attention → FlashAttention v2/v3 - https://github.com/Dao-AILab/flash-attention
- Multi-Layer Perceptrons (MLPs)
- Unembedding

Further optimizations

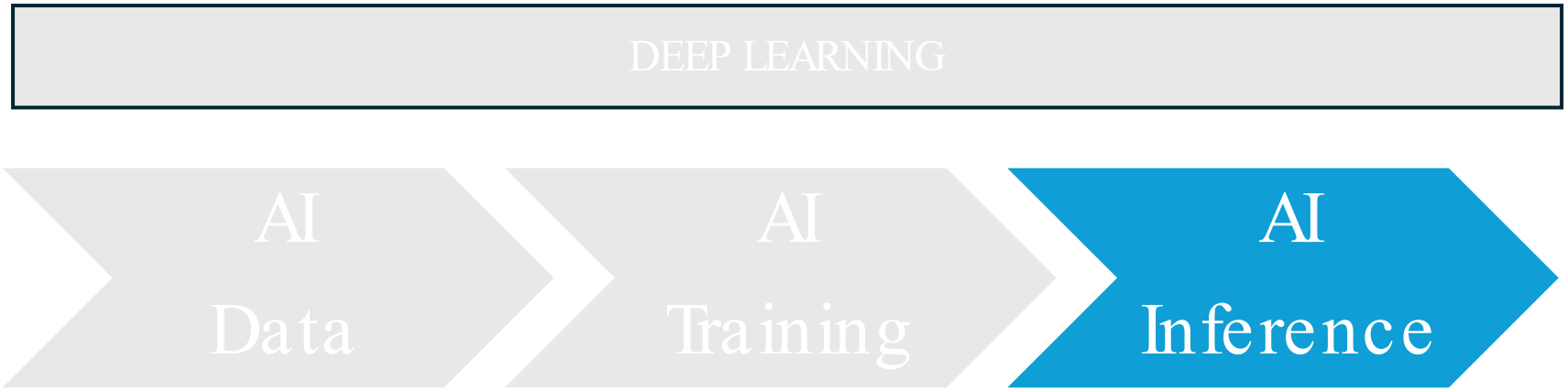


TRANSFOMER model

- Embedding
- Attention
- Multi-Layer Perceptrons (MLPs)
- Unembedding

- DaceML - https://daceml.readthedocs.io/en/latest/
- Thunder - https://github.com/Lightning-AI/lightning-thunder
- DeepSpeed - https://www.deepspeed.ai/
- ColossalAI - https://github.com/hpcaitech/ColossalAI

Further optimizations

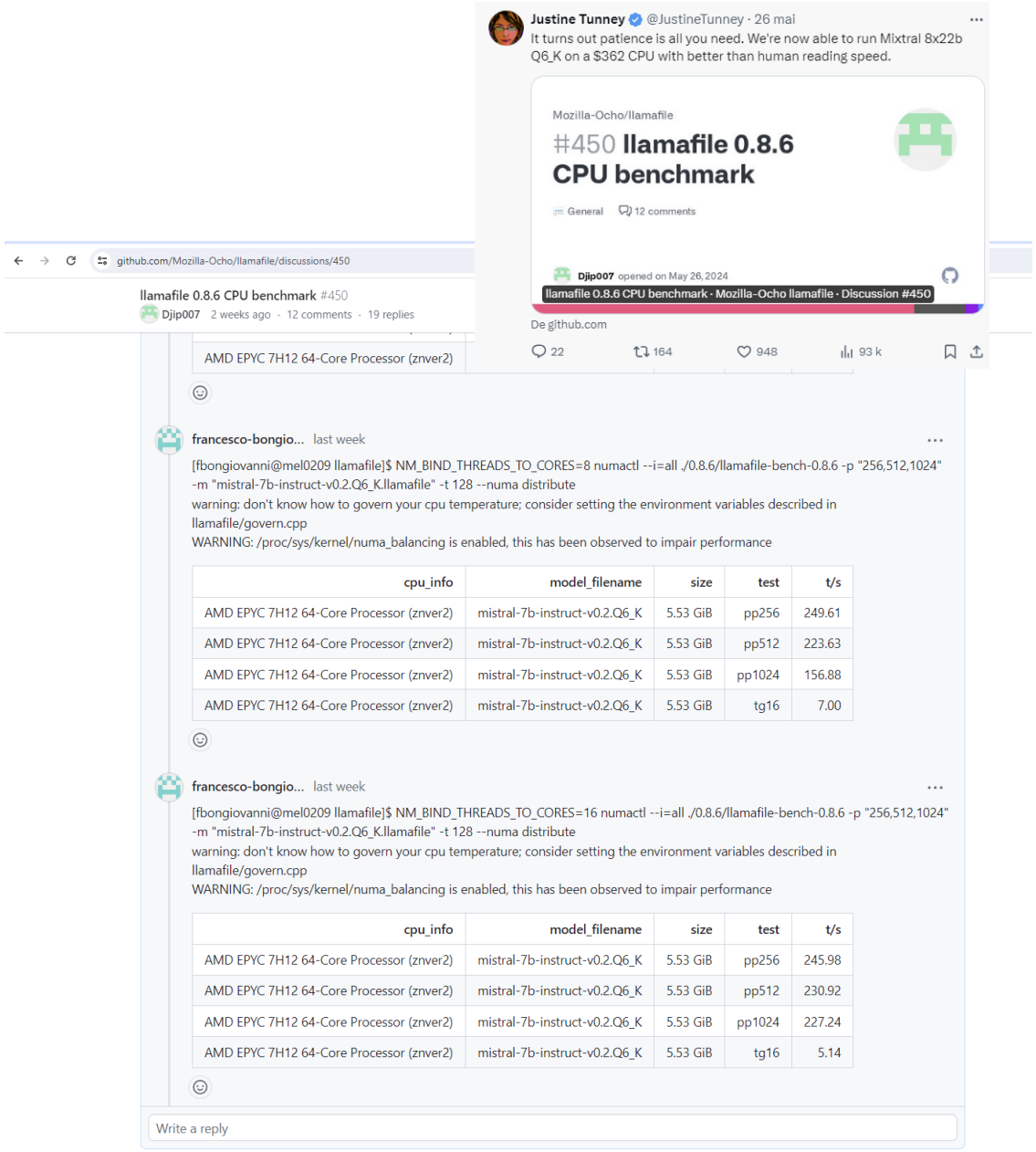


Possible optimizations

- Quantization
- Pruning
- Knowledge Distillation
- Specialized hardware → FPGAs

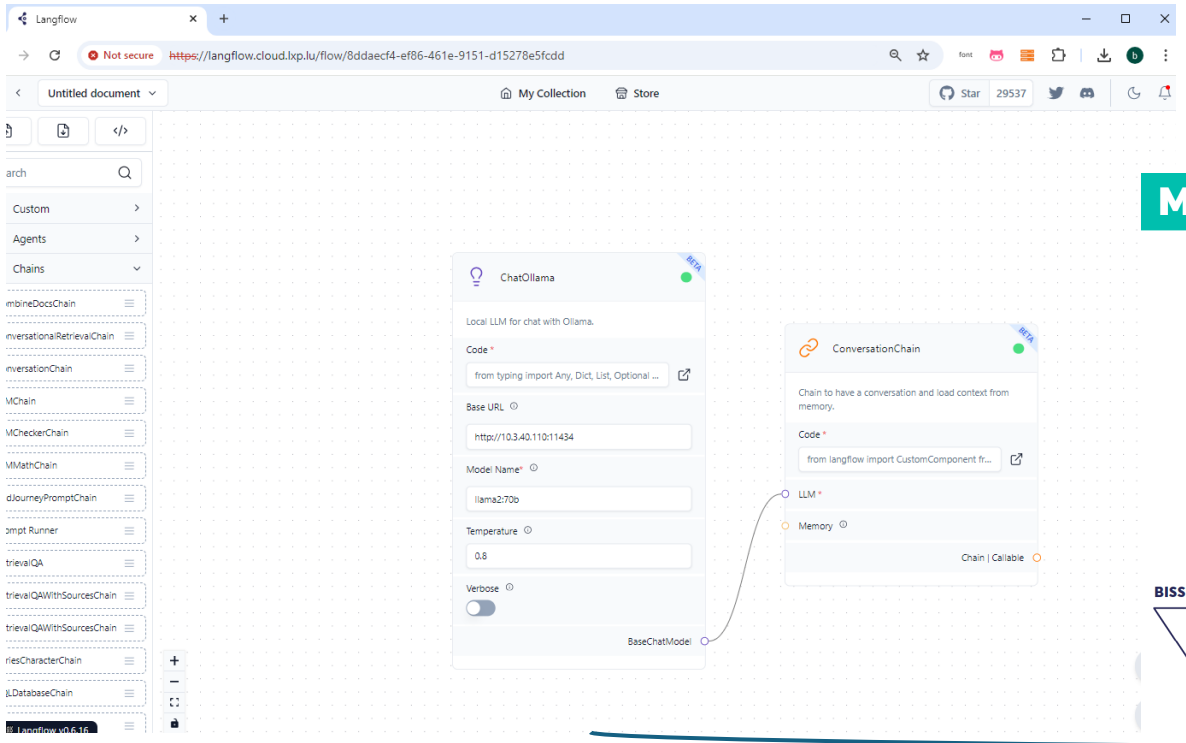
Current investigations around AI

- AI inference on CPUs
- AI inference on FPGAs



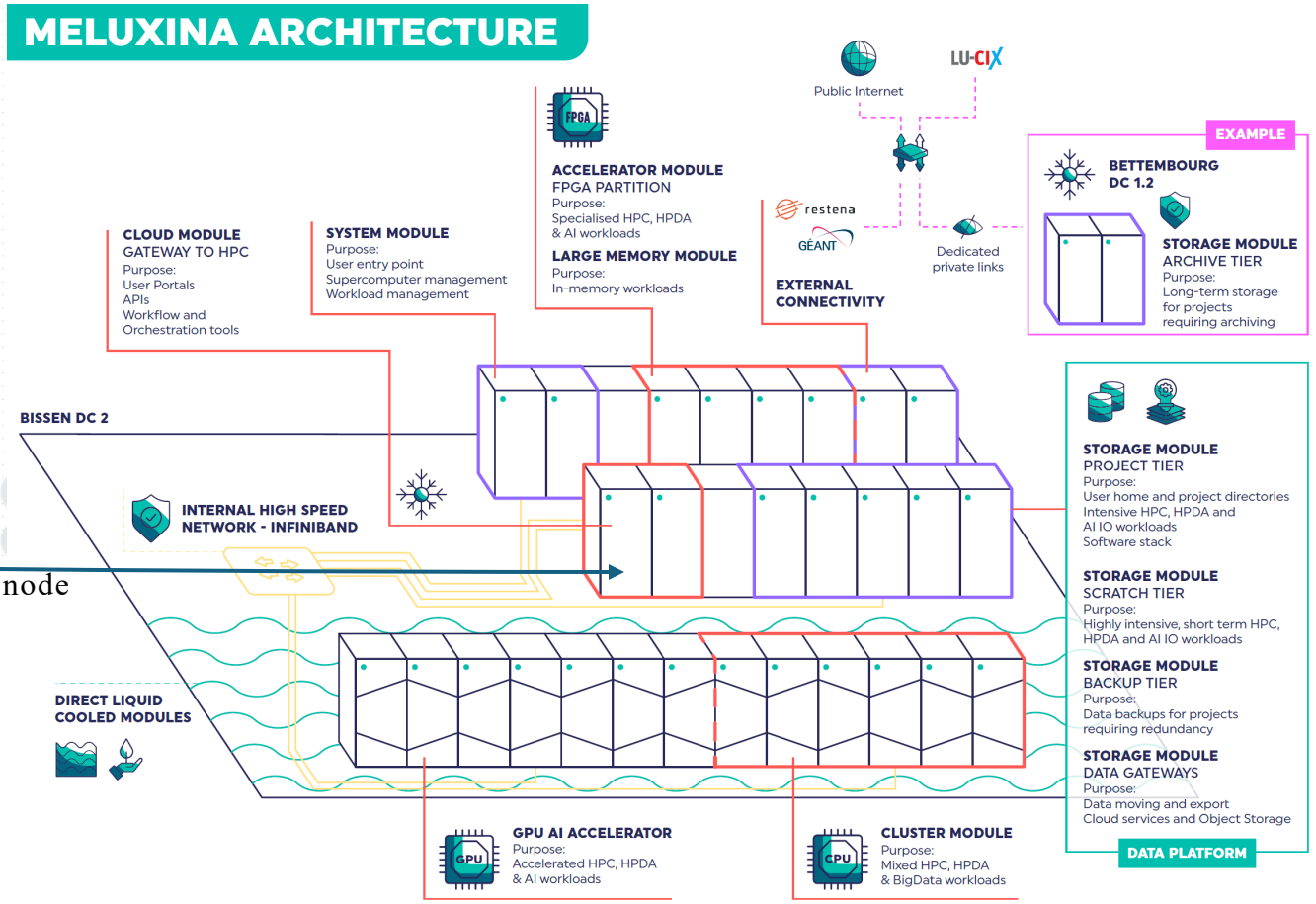
llamafile lets you distribute and run LLMs with a single file.

Current investigations around AI



Deployed on cloud node

langflow



Current investigations around AI

```
[fbongiovanni@mel2110 fbongiovanni]$ export OLLAMA_HOST=0.0.0.0:11434
[fbongiovanni@mel2110 fbongiovanni]$
```

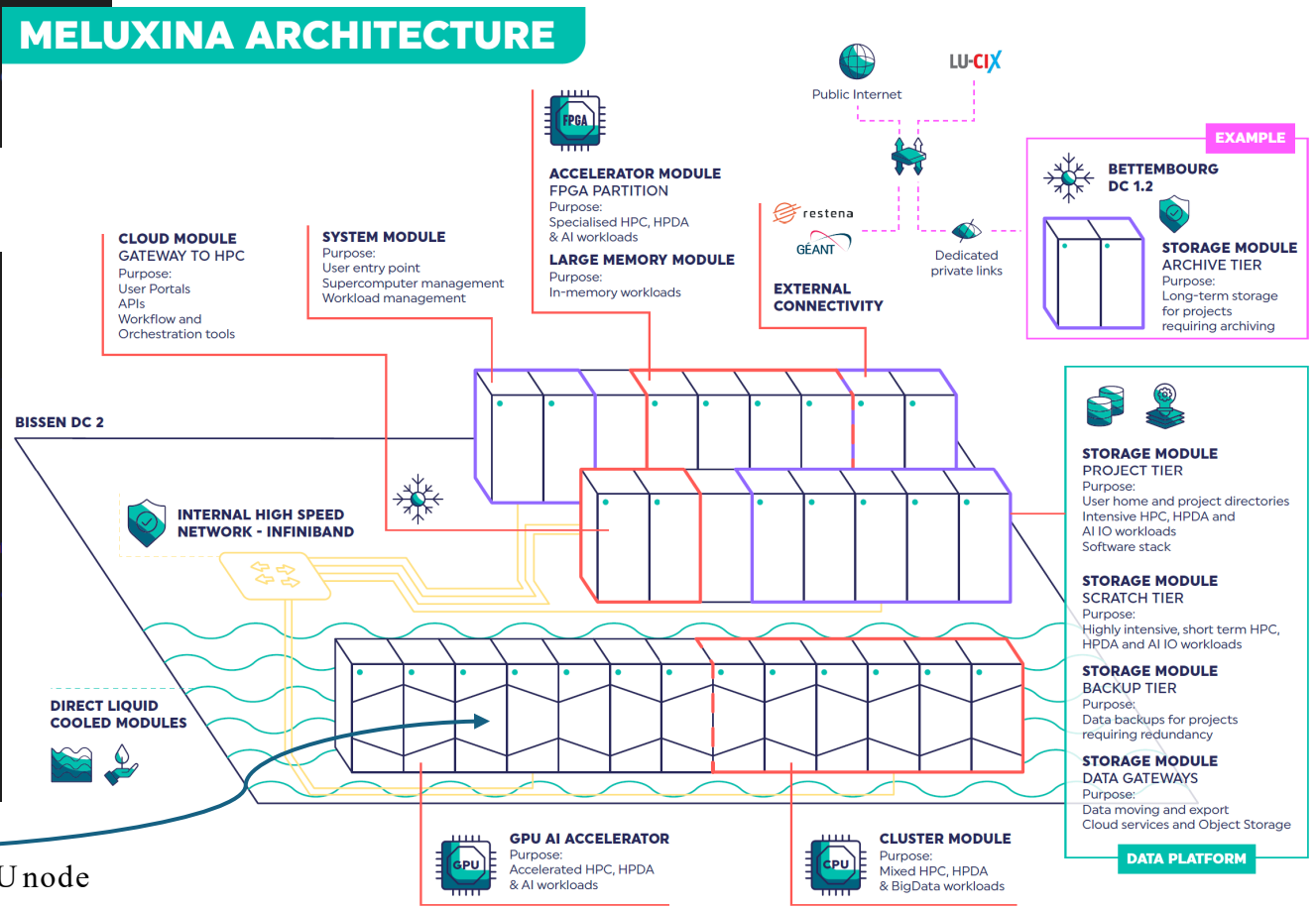
Execute Ollama server

```
[fbongiovanni@mel2110 fbongiovanni]$ ollama serve &
[1] 13889
[fbongiovanni@mel2110 fbongiovanni]$ time=2024-09-24T23:33:31.624+02:00 level=INFO source=images.g
10 msg="total blobs: 9"
time=2024-09-24T23:33:31.690+02:00 level=INFO source=images.go:717 msg="total unused blobs removed
"
time=2024-09-24T23:33:31.695+02:00 level=INFO source=routes.go:1021 msg="Listening on [::
sion 0.1.28)"
time=2024-09-24T23:33:31.697+02:00 level=INFO source=payload_common.go:107 msg="Extractin
braries..."
```

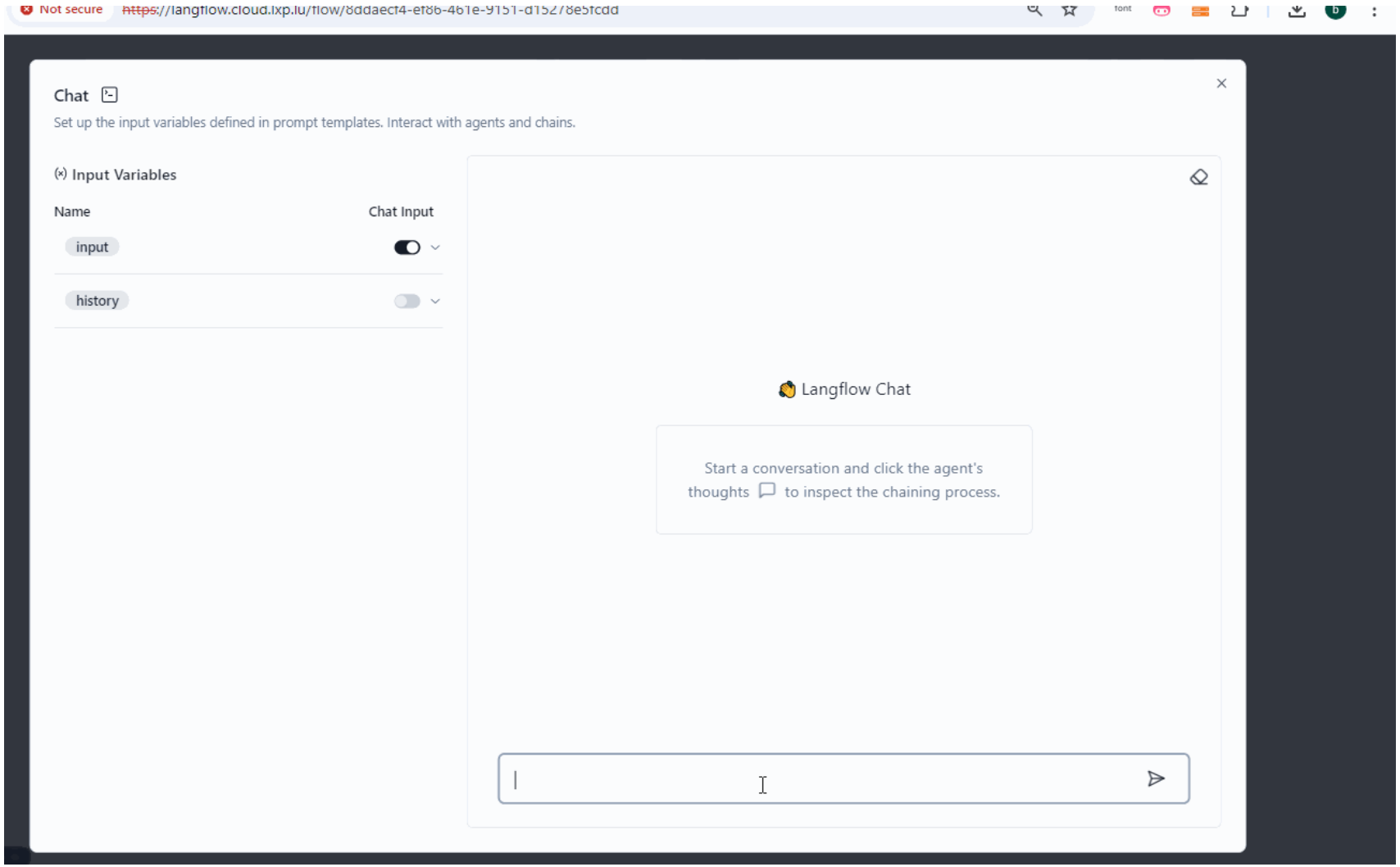
```
[fbongiovanni@mel2110 fbongiovanni]$ ollama run llama2:70b
[GIN] 2024/09/24 - 23:34:13 | 200 |    3.714155ms |       127.0.0.1 | HEAD     "/"
[GIN] 2024/09/24 - 23:34:13 | 200 |   79.534428ms |       127.0.0.1 | POST     "/api/show"
[GIN] 2024/09/24 - 23:34:13 | 200 |    3.505146ms |       127.0.0.1 | POST     "/api/show"
time=2024-09-24T23:34:14.144+02:00 level=INFO source=cpu_common.go:11 msg="CPU has AVX2"
time=2024-09-24T23:34:14.154+02:00 level=INFO source=gpu.go:146 msg="CUDA Compute Capabilit
: 8.0"
time=2024-09-24T23:34:14.154+02:00 level=INFO source=cpu_common.go:11 msg="CPU has AVX2"
time=2024-09-24T23:34:14.154+02:00 level=INFO source=gpu.go:146 msg="CUDA Compute Capabilit
: 8.0"
time=2024-09-24T23:34:14.154+02:00 level=INFO source=cpu_common.go:11 msg="CPU has AVX2"
loading library /tmp/ollama1892780583/cuda_v11/libext_server.so
time=2024-09-24T23:34:14.159+02:00 level=INFO source=dyn_ext_server.go:90 msg="Loading Dyna
rver: /tmp/ollama1892780583/cuda_v11/libext_server.so"
time=2024-09-24T23:34:14.160+02:00 level=INFO source=dyn_ext_server.go:150 msg="Initializin
rver"
ggml_init_cublas: GGML_CUDA_FORCE_MMQ:   yes
ggml_init_cublas: CUDA_USE_TENSOR_CORES: no
ggml_init_cublas: found 4 CUDA devices:
  Device 0: NVIDIA A100-SXM4-40GB, compute capability 8.0, VMM: yes
  Device 1: NVIDIA A100-SXM4-40GB, compute capability 8.0, VMM: yes
  Device 2: NVIDIA A100-SXM4-40GB, compute capability 8.0, VMM: yes
  Device 3: NVIDIA A100-SXM4-40GB, compute capability 8.0, VMM: yes
```

Run Llama2:70b model

Running on GPU node



Current investigations around AI




LUX
PROVIDE
EXPLORING
THE FRONTIERS
OF DIGITAL INTELLIGENCE
MADE IN
LUXEMBOURG