Performance Analysis of Large Scale Distributed DL: A Case Study

Anara Kozhokanova

aiXcelerate 2021: Part II – Machine Learning (ML)

Outline

- Distributed DL Case Study: ResNet-50 on CLAIX
- Application Profiling/Tracing
- Performance Analysis
- Study Outcome

Case Study: ResNet-50 on CLAIX

HPC.NRW

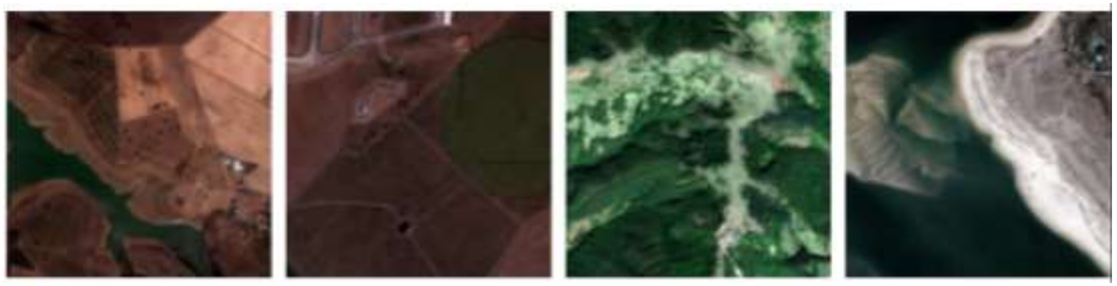
- Cluster overview: **CLAIX**, IT Center, RWTH Aachen University
  - 48 dual 24-core Intel Xeon Platinum 8160 ‘Skylake’ compute nodes,
    - Each node: 2 NVidia Tesla V100-16GB coupled with NVlink 2.0 (25GB/s-2links)

- Environment overview:
  - CentOS Linux 7.9.2009, kernel 3.10.0
  - GCC/8.2.0, CUDA/10.0, CuDNN/7.4, OpenMPI 3.1.3
  - Conda 2.9.2 virtual environment:
    - NCCL 2.5.7.1, **Horovod** 0.18.2, **TensorFlow-GPU** 1.15.0, Python 3.7, mpi4py 3.0.3, Scikit-learn 0.23.2, Keras 2.2.4, HDF5 1.10.61.15.4

Case Study: Application Overview

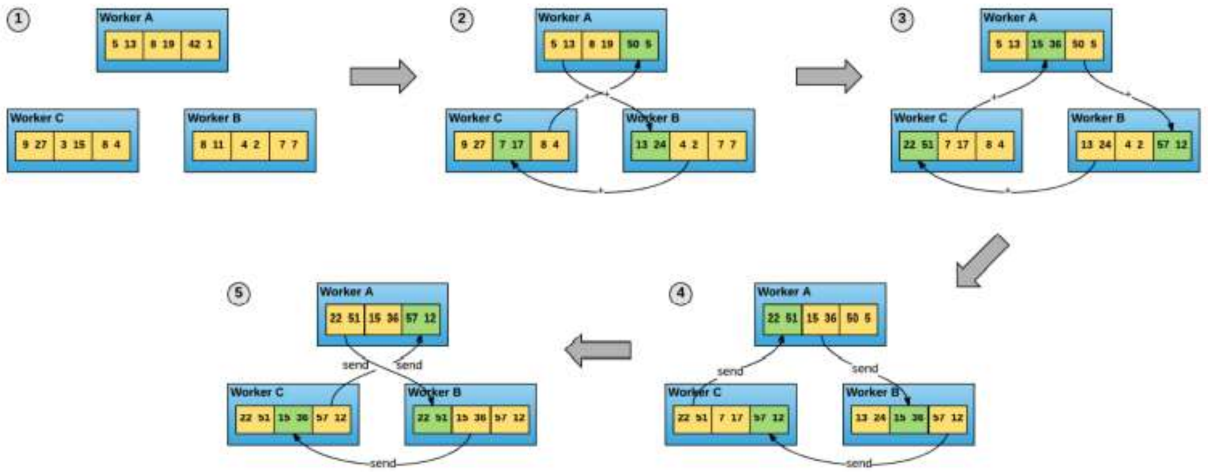
HPC.NRW

- Title: Remote Sensing Big Data Classification with High Performance Distributed Deep Learning
- Project: Performance Optimization and Productivity (POP)
- Code: Python, MPI, CUDA
- NN architecture: ResNet-50 (CNN)
- Epochs: 100
- Batch size: 64
- Total trainable parameters: 23,915,115
- Data:
  - 590326 patches from Sentinel-2 tiles, each patch annotated with 1-12 labels (total 43 labels)
  - 12 spectral bands: ranging 120x120px, 60x60px, 20x20px. BigEarthNet tiles (http://bigearth.net)
  - Training data: 60%, Validation: 20%, Testing: 20%
  - Total size: 245GB (before preprocessing)
  - Data preprocessing: upsampled up to 120x120px, random flipping/rotation, mix-up



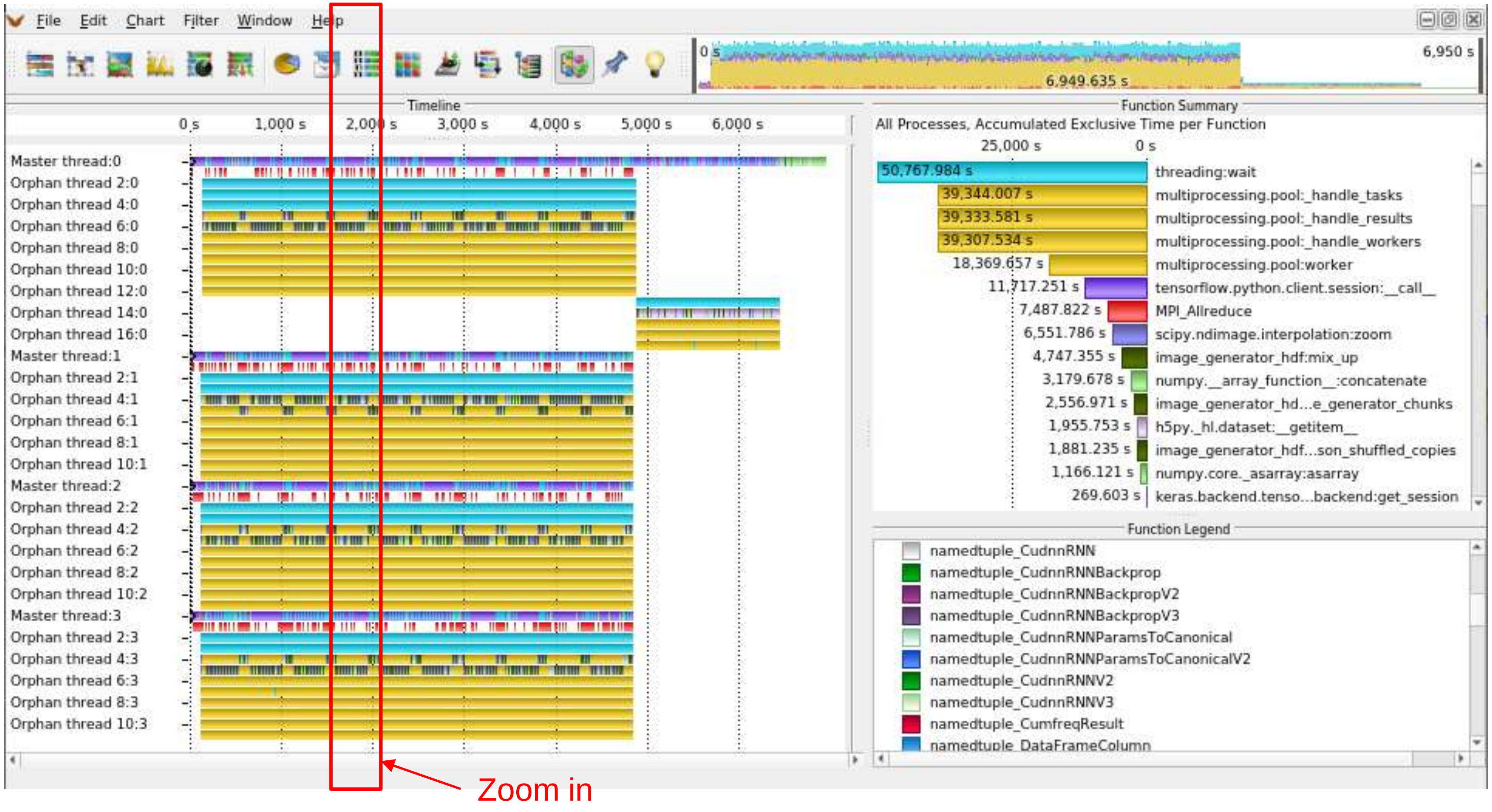
Horovod Overview

- Horovod – open source synchronous data-parallel framework for distributed learning
- Supports Keras, TensorFlow, PyTorch, MXNet
- NCCL, MPI
- Other topologies supported: Hierarchical allreduce, Tree-based allreduce, etc
- Bandwidth bound -> at large scale becomes latency bound



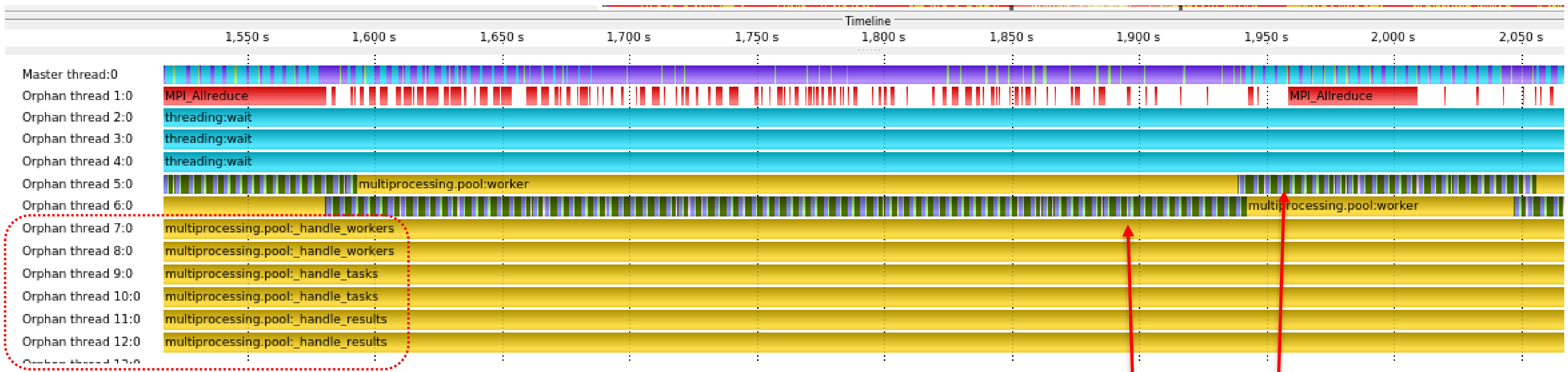
Application Profiling/Tracing

– Score-P 6.0 with Score-P Python Binding 3.0, Vampir 9.8.0, Cube 4.6



- GPUs: 4
- Iterations: 10
- Without CUDA
  - Some overhead when enabling CUDA profiling/tracing
  - reduce iterations (epochs) and/or data size

Application Profiling/Tracing

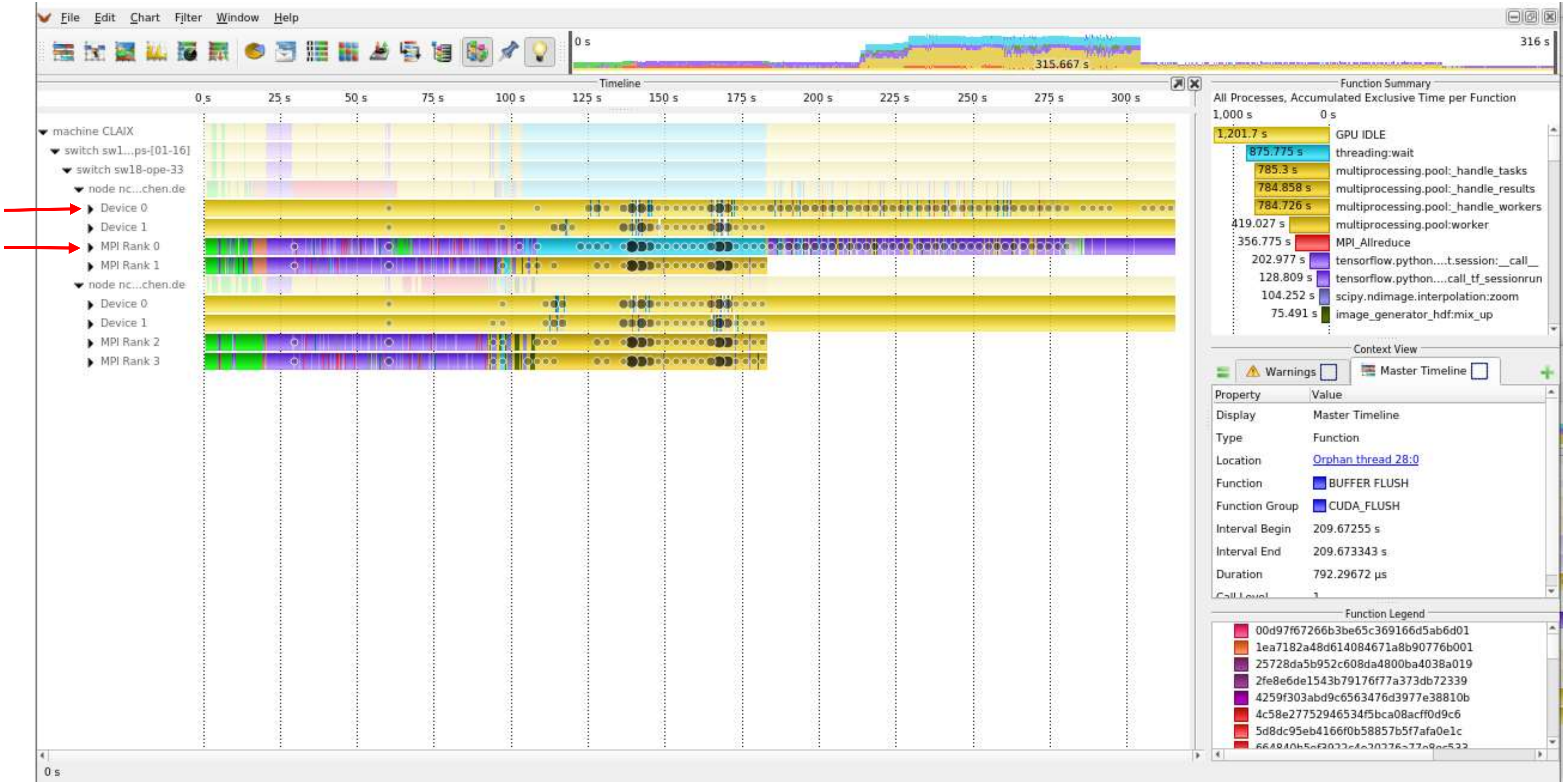


- Each rank spawns 2 workers threads for training and validation operations:
  - Each worker spawns:
    - _handle_workers
    - _hande_tasks
    - _handle_results

```
history = model.fit_generator(
    generator = image_generator_chunks(file = new_file, ind = X_train, batch_size = BATCH_SIZE, rank = hvd.rank(), n_channels = num_channels, arr_norm=arr_norm),
    validation_steps=VAL_STEPS // hvd.size(),
    validation_data = image_generator_chunks(file = new_file, ind = X_train, batch_size = BATCH_SIZE, rank = hvd.rank(), n_channels = num_channels, arr_norm=arr_norm),
    steps_per_epoch = STEPS // hvd.size(),
    epochs = EPOCHS,
    callbacks=callbacks,
    initial_epoch = initial_epoch )
```

Application Profiling/Tracing

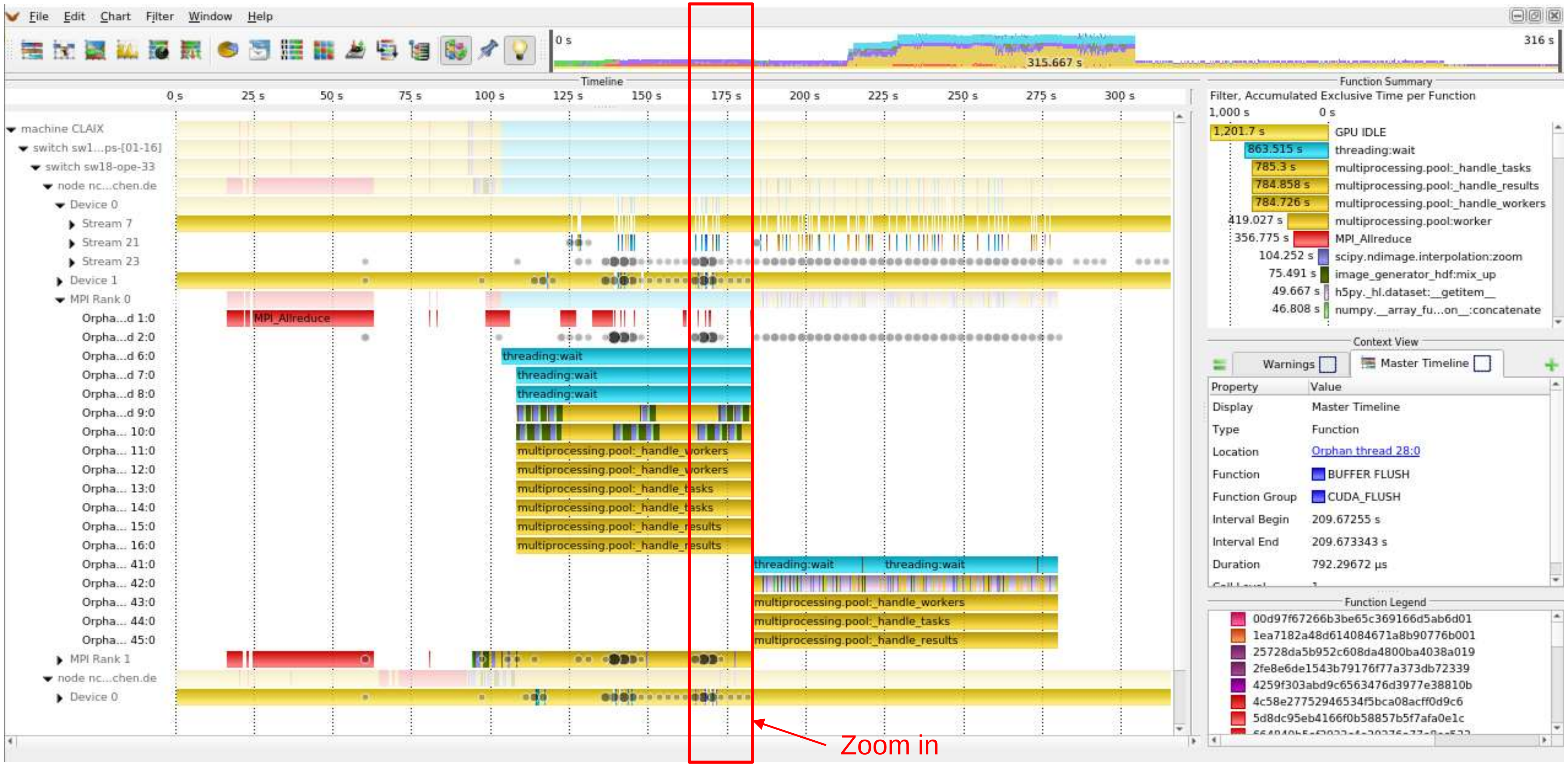
HPC.NRW



- GPUs: 4
- Epochs:2
- Batch size: 256
- Batches per epoch: 12
- Data reduction:
  - train: 0.025%
  - validate:0.010%
  - test: 0.010%
- With CUDA
  - SCOREP_CUDA_ENABLE =runtime,kernel,pure_idle, memcpy,idle,references,flushatexit
- With filtering

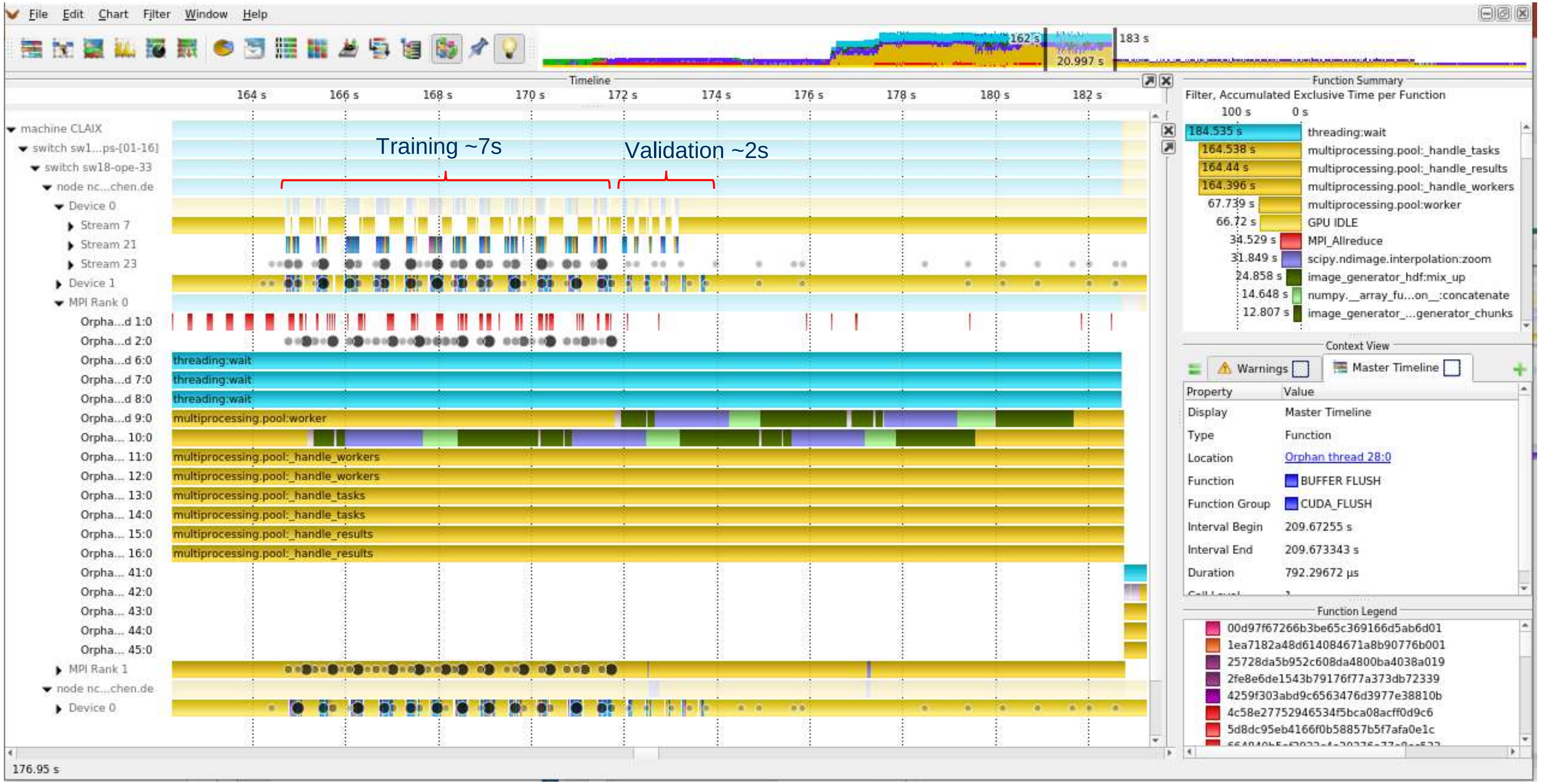
Application Profiling/Tracing

HPC.NRW



INNOVATION THROUGH COOPERATION.

Application Profiling/Tracing

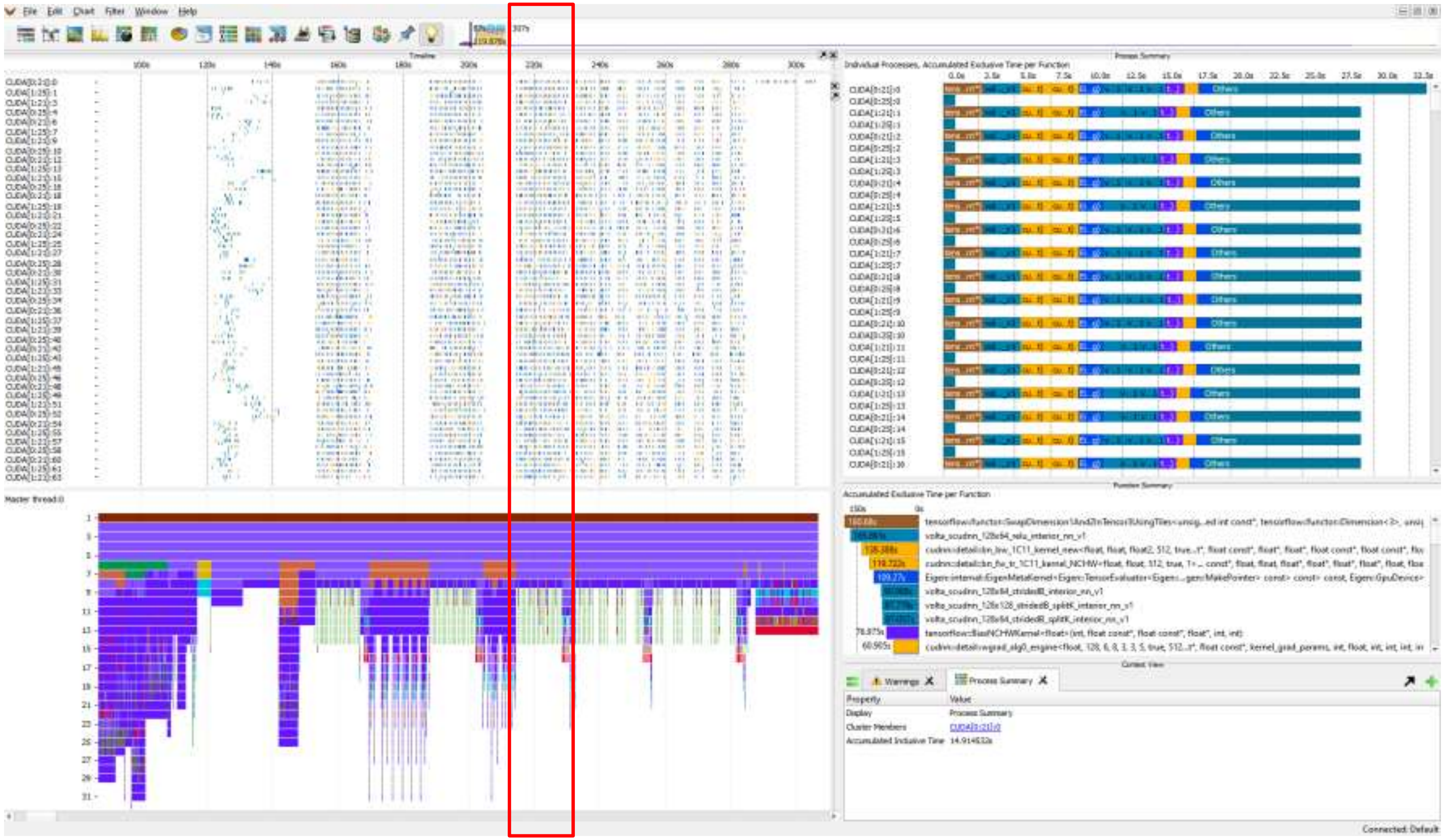


TF reports:

```
12/12 [==============================] - 9s 768ms/step
```

- Observation:
  - Total epoch time: 9 sec
  - 12 batches/epoch
  - GPU is idle about half of the time in an epoch!

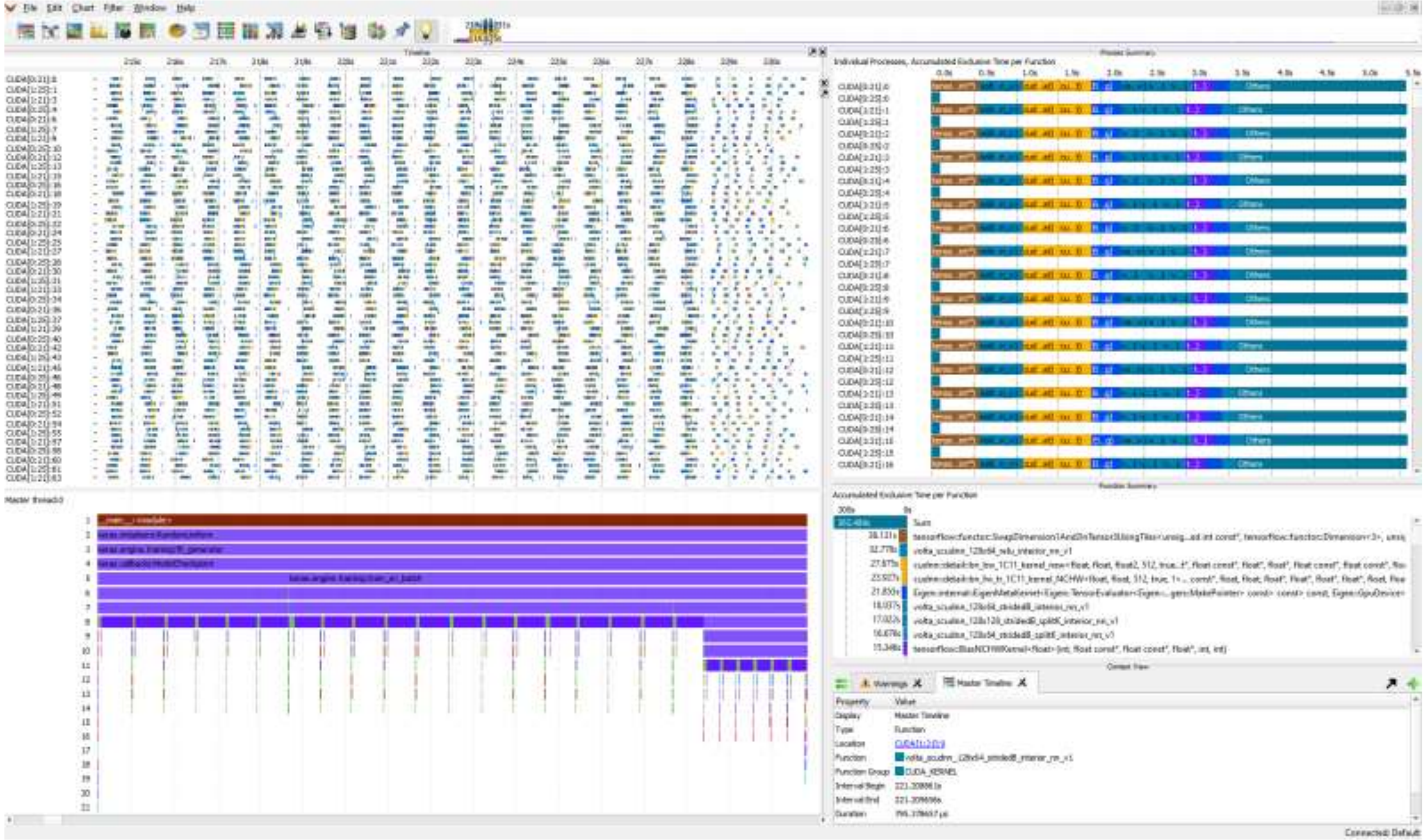
CUDA Kernels View



- **GPUs:** 64
- **Epochs:** 5
- **Data size:** original (except testing phase)
- **Focus of analysis**: 3rd epoch

GPU Performance Metrics

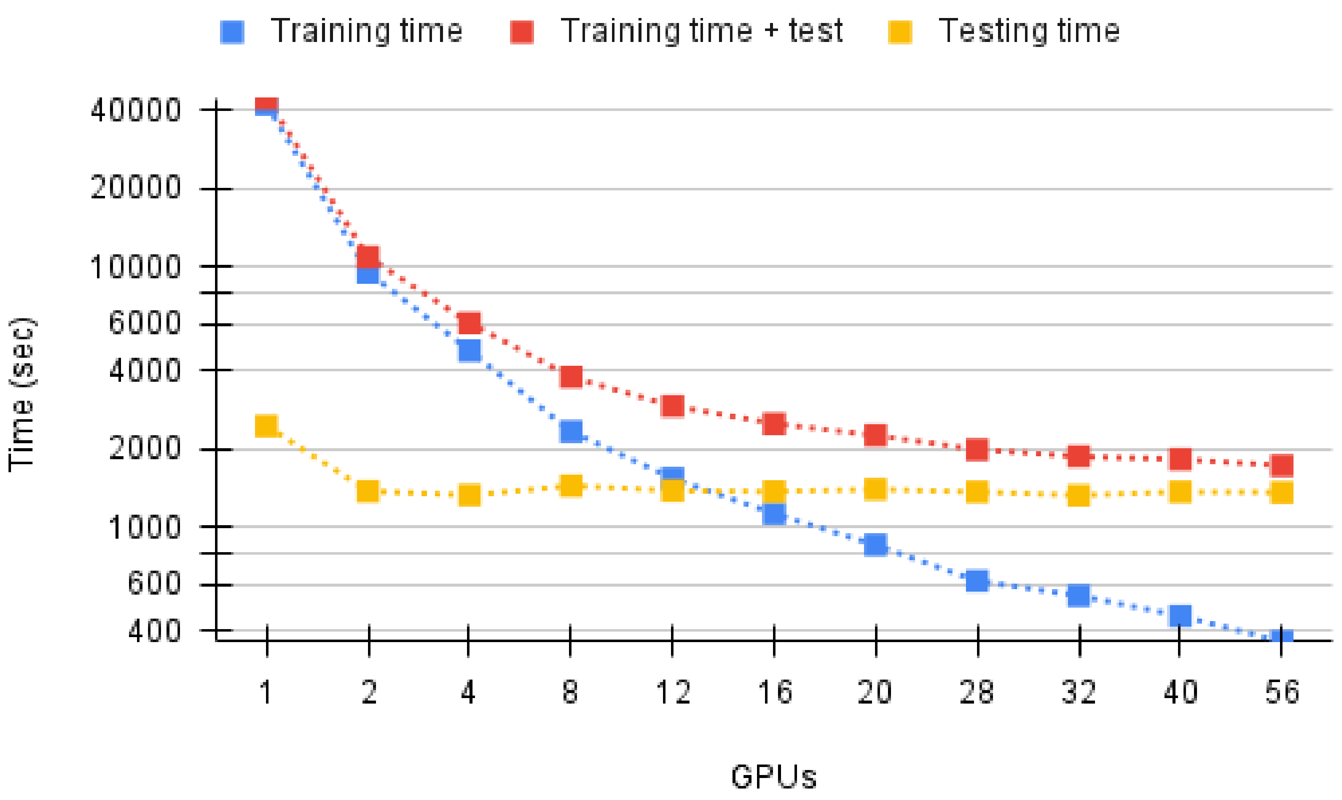
HPC.NRW



POP Metrics (multiplicative model)

# GPUs	Parallel Efficiency	Load Balance	Communication Efficiency
64 GPU	0.34	~0.99	0.33
48 GPU	0.27	~0.99	0.26

Application Scaling: Strong Scaling



Strong Scaling: batch size is constant

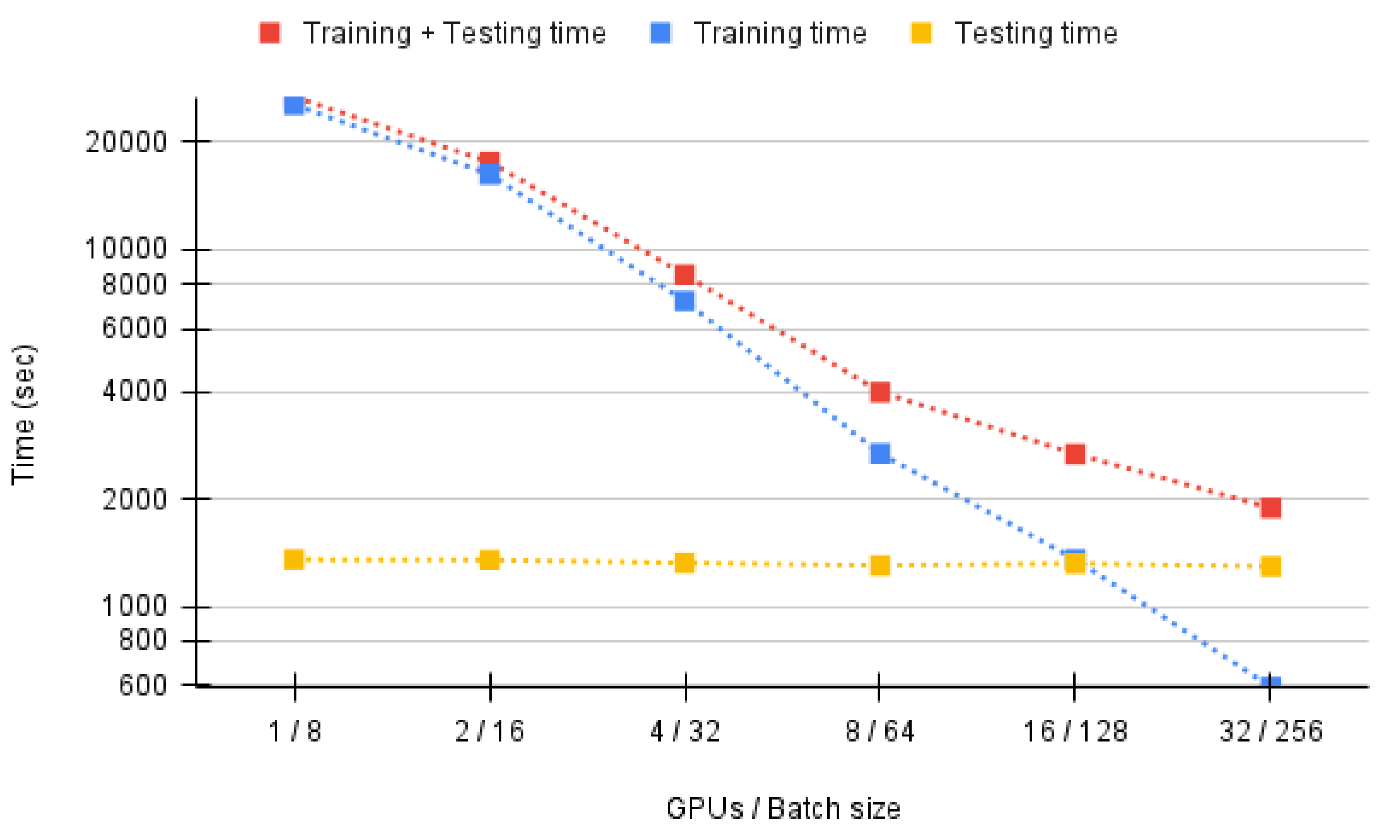
- Batch size: 256 (max per GPU)
- Epochs: 10

Observation:

- Total scaling is bound by sequential testing phase
- Training phase exhibits superlinear speedup

Application Scaling: Weak Scaling

HPC.NRW



Weak scaling: batch size is increased with more workers

- Batch size is limited by GPUs memory:
  - Max batch size: 256, fills up 16 GB
- Bigger batch size means less communication overhead and more samples are processed per each epoch -> faster runtime
- Batch size is a tradeoff between GPU memory utilization and model's convergence

Observation:

- Superlinear speedup in the training phase
- Testing time bounds the total scalability

Recommendations

HPC.NRW

- Parallelize the sequential testing part or move it to a separate single-node job
- Add workers to image generator to supply preprocessed batched data to GPU for reducing GPU idle time
- Online data preprocessing can starve GPUs
  - Preprocess data offline
- Horovod's Tensor Fusions:
  - Tensor Fusion – combining all the tensors that are ready to be reduced at certain time into **one reduction operation** and batch them in small **allreduce** operations. Enabled by default with 64MB buffer size. Test with bigger buffer size.
- Use GPU nodes that are on one leaf switch to avoid network contention
- Use TensorFlow configuration that supports **AVX**
- AMP: Automatic Mixed Precision: FP16, TF32 (starting from TF 1.15.2). Nvidia GPUs with compute capability 7.0=< (Volta, Turing, Ampere) support mixed precision

Conclusion

HPC.NRW

- Working with ML/DL applications can be a challenge:
  - time-consuming and hard to debug
  - building proper environment with different libraries:
    - virtual environment: Python virtual environment, Conda
    - containers: Docker, Singularity, etc
  - Don't be discouraged! Once the issues are addressed, everything will work.
- **Availability** of the GPU nodes is crucial (currently GPU nodes are fully used):
  - Test small cases locally first. If works, submit to Slurm
- Profiling applications with big datasets: reduce data, iterations, Scorep's CUDA features in order to obtain a readable profile
- Know your tools, frameworks, system well for achieving good performance
- Many performance analysis tools exist, use those that suit your needs: framework's toolkit (Horovod Timeline, TensorFlow's TensorBoard), Nvidia Nsight Systems, etc

- IT Center HelpDesk offers support in various technical subjects and will help with resolving your issues

Thank you for attention
Questions?