Benchmarking Deep Learning Infrastructures by means of Tensorflow and Containers

Steinbuch Centre for Computing, Karlsruhe Institute of Technology

Adrian Grupp[1)], **Valentin Kozlov**[1)], Isabel Campos[2)], Mario David[3)], Jorge Gomes[3)], Álvaro López García[2)]

[1)] KIT (Germany) [2)] CSIC (Spain) [3)] LIP (Portugal)

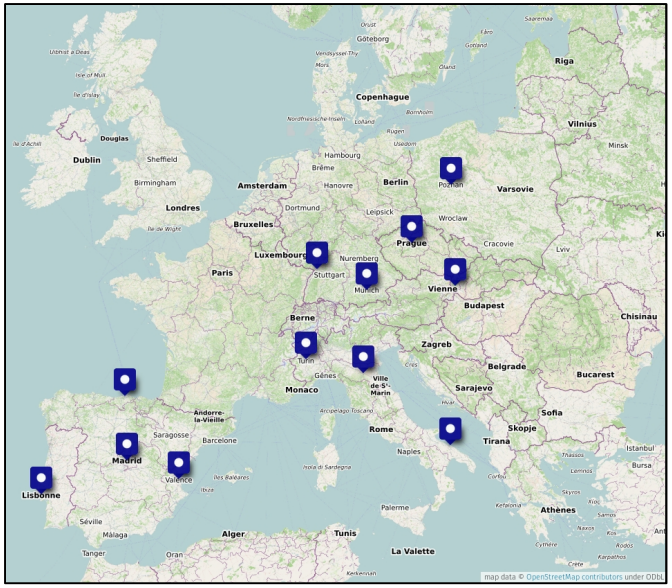
Scalable Data Analytics
in Scientific Computing Workshop

Frankfurt, 20-June-2019

DEEP Hybrid DataCloud: context

Designing and **E**nabling **E**-Infrastructures for intensive data **P**rocessing in a Hybrid DataCloud

- Started as a spin-off project (together with XDC) from INDIGO-DataCloud technologies
- H2020 project, EINFRA-21 call
- Runs November 1st 2017 – April 2020
- March 27, 2019 : mid-term Review (Luxembourg)
- 9 academic partners + 1 industrial partner: CSIC, LIP, INFN, PSNC, KIT, UPV, CESNET, IISAS, HMGU; Atos



Ease and lower the entry barrier for scientists
in using intensive computing techniques, e.g. deep learning, to exploit very large data sources

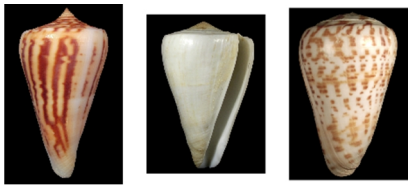
- Build ready to use modules and offer them through an open catalog or marketplace (Docker images)
- Implement common software development techniques also for scientist's applications (DevOps)
- Transparent execution on e-Infrastructures (pilot testbed, orchestration)

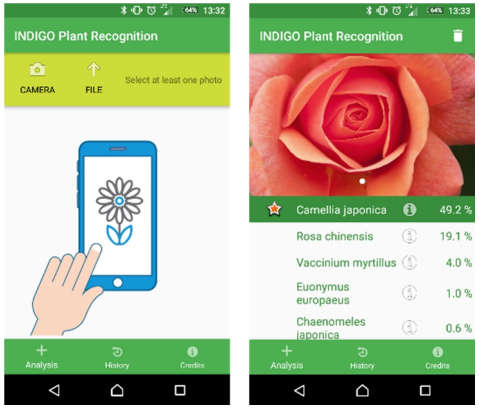
DEEP pilot use-cases

Plant classification → Image Classification

For training and testing image classifiers (CNNs; TensorFlow). From this model the following services are derived:

- Plants
- Conus marine snails
- Seeds
- Phytoplankton

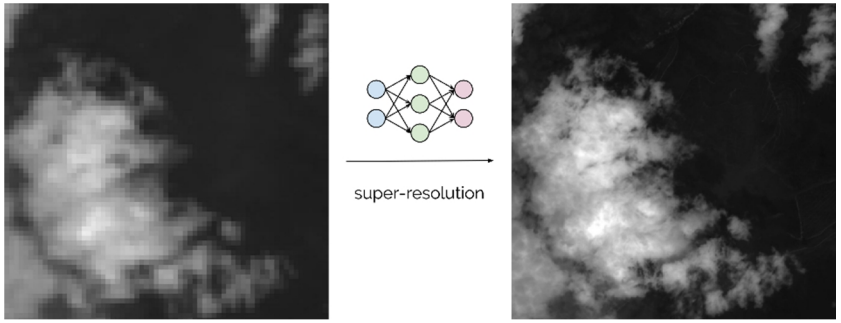




Satellite Imagery

Explore application of Machine Learning for satellite imagery (e.g. remote object detection, terrain segmentation, meteorological prediction).

Currently being developed is super-resolution service to upscale low resolution bands to high resolution with deep learning (e.g. DSen2; TensorFlow)



DEEP pilot use-cases

Retinopathy

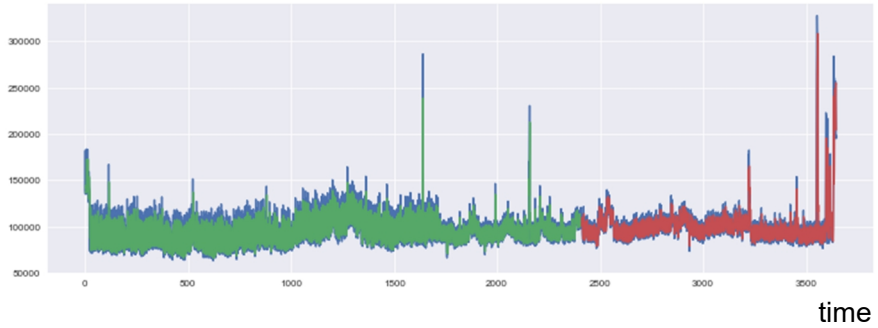
Diabetic retinopathy is a fast-growing cause of blindness worldwide. The use-case focuses on a deep learning approach (CNNs; TensorFlow) to automated classification of retinopathy based on color fundus retinal photography images (DR=0 (=healthy) .. 4 (blind)).

DR=0

DR=4

Massive Online Data Streams: Online analysis of data streams

Intrusion detection systems: provide an architecture able to analyze massive on-line data streams, also with historical records, in order to generate alerts in real-time. Based on proactive time-series prediction adopting artificial neural networks (e.g. LSTM, GRU; TensorFlow).



dataset, prediction (train), prediction (test)

Motivation

- To check our (hw + sw) implementations and various **infrastructures** in general
- To have a **reference** point

Need a tool:

- Optimized for best performance
- Scalable across multiple GPUs
- Scalable across different nodes (distributed training)
- Not affected by other pre-installed libraries
- Not affected by the OS (Linux) version
- Easily deployable
- Open source, good support, …

Our approach:

TensorFlow Benchmarks code packed in Docker containers and run in a multi-user environment by either udocker or singularity container tools

TensorFlow Benchmarks

- Publicly available from TensorFlow
  (https://github.com/tensorflow/benchmarks)
- Highly optimized CNN implementations:
  AlexNet, ResNet, VGG, and InceptionV3
- Already implemented functionality for a multi-GPU and distributed training (e.g. via TF parameter server, Horovod)
- Performance metrics:
  images/second, accuracy, loss
- Official results from the developers
  (https://tensorflow.org/guide/performance/benchmarks)

Image Source:
https://www.wikiwand.com/de/TensorFlow

Container tools

udocker

Run applications encapsulated in docker containers.

- without system administrators intervention
- without additional system software
- as normal user
- in interactive or batch systems
- Supports several execution modes via *PRoot*, *Fakechroot*, *runC* or *singularity*
- GPU access

Open source: https://github.com/indigo-dc/udocker

Computer Physics Communications **232**, 84–97 (2018). DOI: 10.1016/j.cpc.2018.05.021

singularity

A container solution created by necessity for scientific application driven workloads.

- installed by a system admin
- runs containers as a normal user
- *blocks privilege escalation within the container*
- interactive or batch system mode
- uses its own image format
- supports Docker
- utilizes User NameSpaces
- GPU access

https://sylabs.io PLoS ONE (2017). DOI: 10.1371/journal.pone.0177459

Methodology

Datasets:

- Synthetic Data: random generated
- ImageNet: size 144 GB
- CIFAR-10: size 341 MB, small resolution (32x32)

Models:

- AlexNet, ResNet50, ResNet56

TF Benchmark has to match TensorFlow version (1.10)

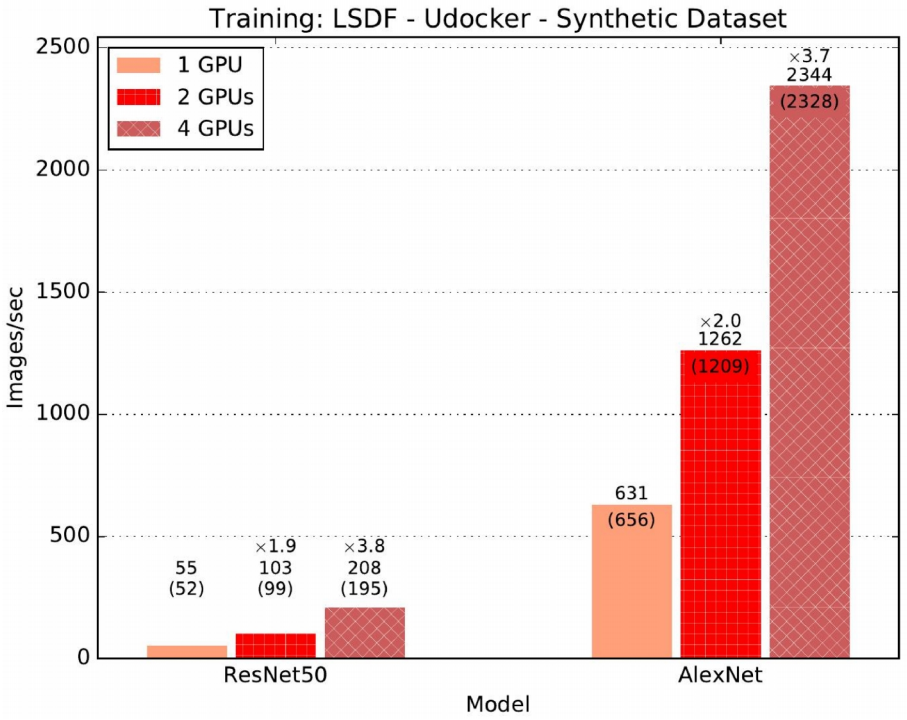
Own scripts:

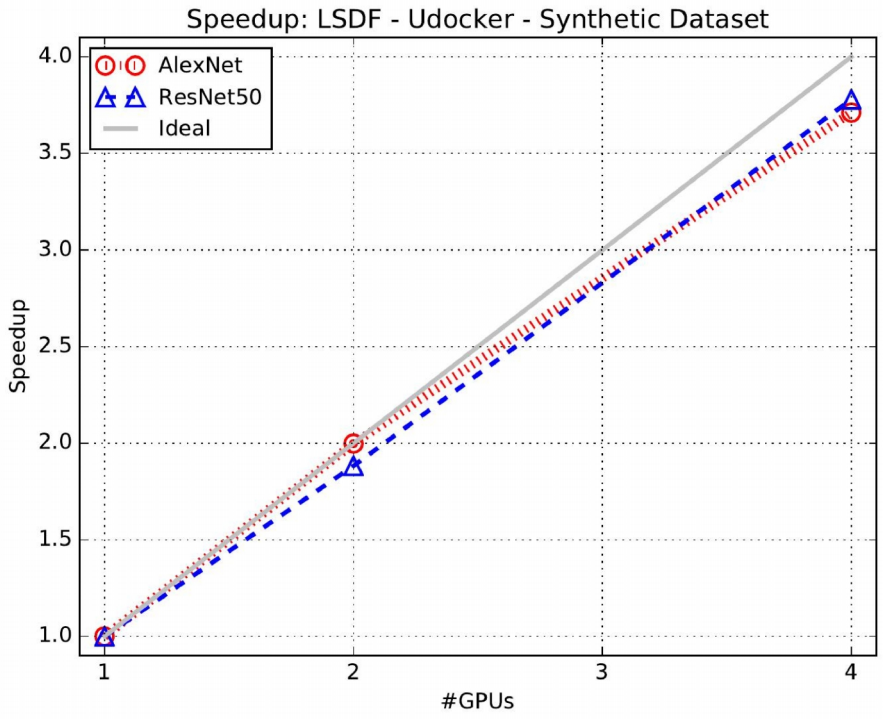
- Job submission, Result parsing & plotting

Two machines:

Setup	CPU	RAM	Nvidia GPU (driver version)	Storage	Batch system
LSDF-GPU	2 × Intel Xeon E5-2630 v3	128 GB	4 × Tesla K80, 12 GB (410.72)	NFS mount	none
ForHLR-II	4 × Intel Xeon E7-4830 v3	1 TB	4 × GTX980 Ti, 6 GB (418.56)	Lustre	Slurm

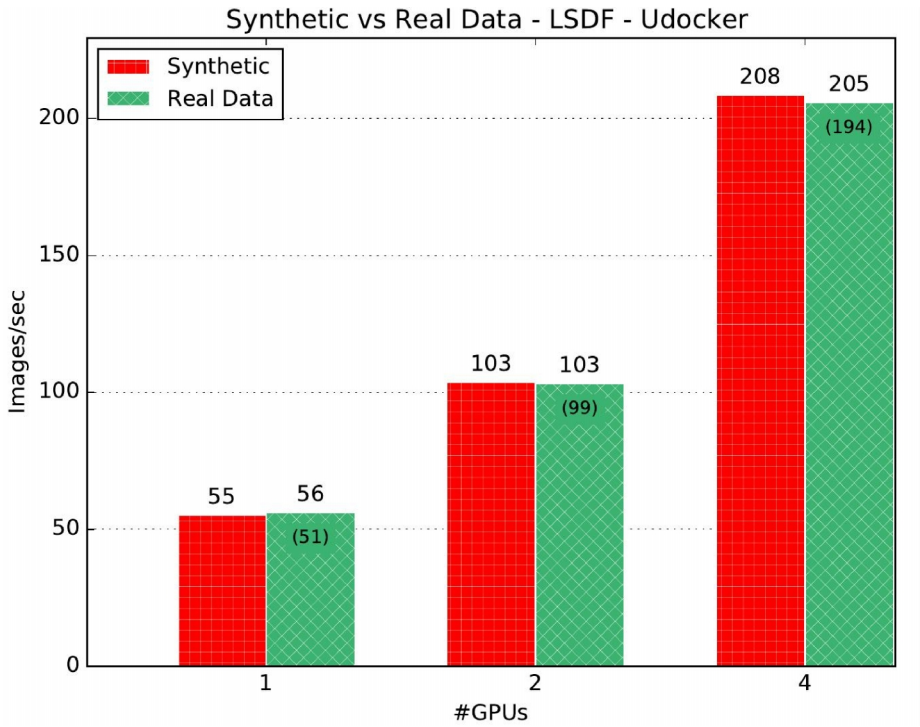
Results: Synthetic Data



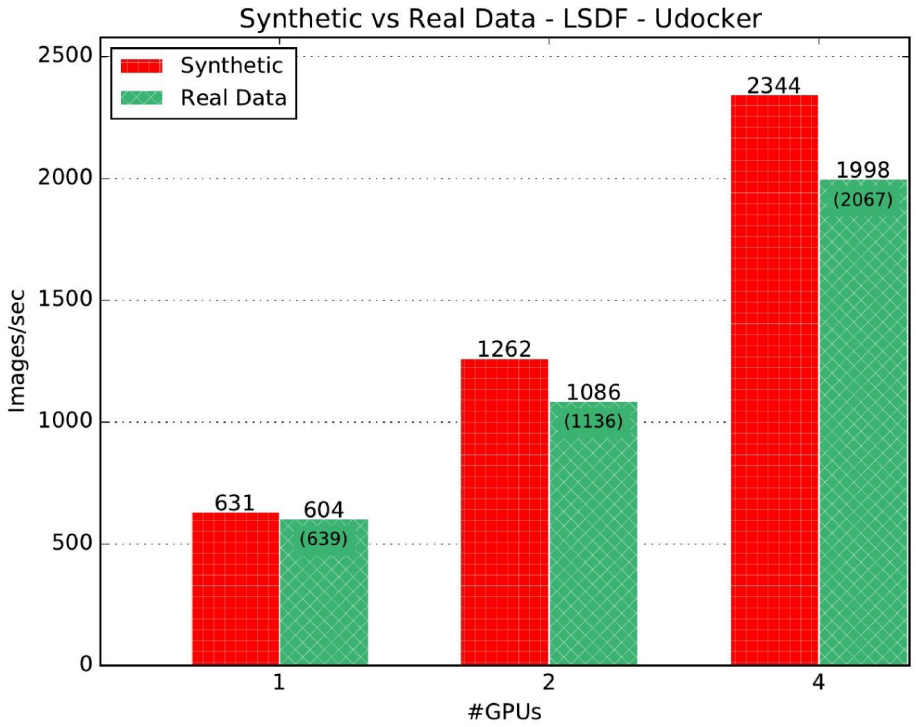


Neural network	Batch size per GPU	Optimizer	variable_update	local_parameter_device
ResNet50	64	sgd	parameter_server	cpu
AlexNet	512	sgd	parameter_server	cpu

Results: Synthetic vs. Real (ImageNet)



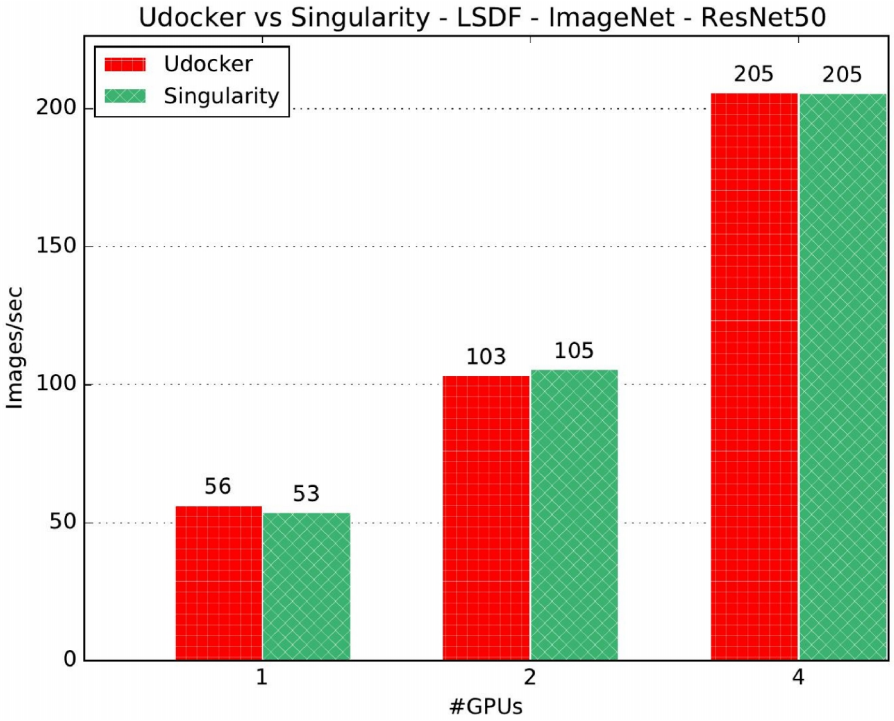
ResNet50



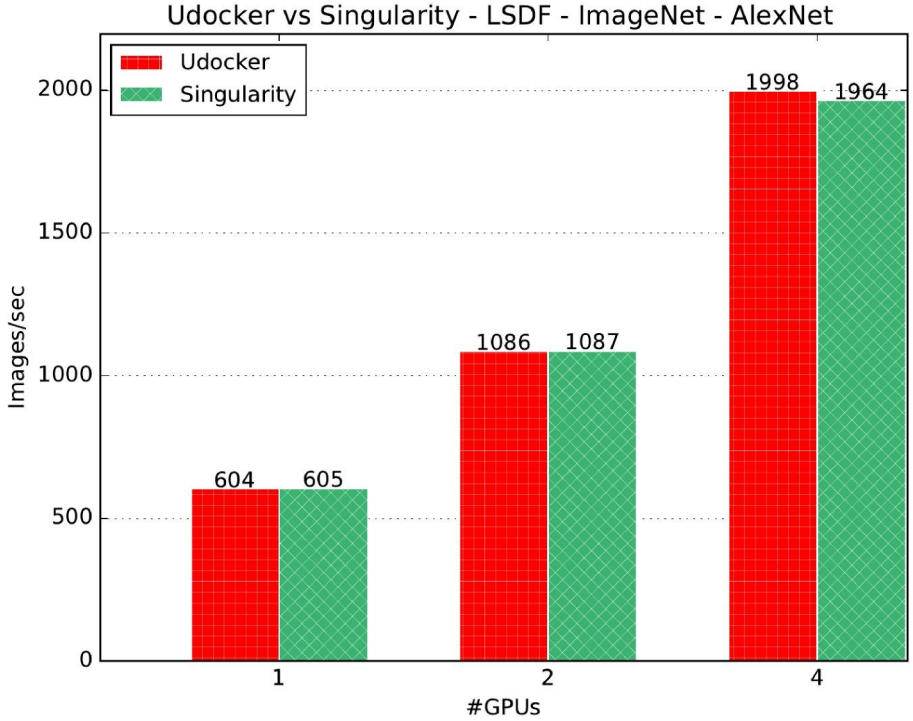
AlexNet

Compare Synthetic Data with ImageNet dataset, udocker

Results: udocker vs. singularity



ResNet50



AlexNet

Comparison for ImageNet: both tools perform equally well

Results: Overview of multi-GPU (short) runs

	# GPUs	Official TF, K80 img/s	LSDF-GPU, K80 udocker	ForHLR-II, GTX 980 Ti udocker
ResNet50	1	51	56	118
	2	99 (x 1.9)	103 (x 1.8)	225 (x 1.9)
	4	194 (x 3.8)	205 (x 3.7)	426 (x 3.6)
AlexNet	1	639	604	1314
	2	1136 (x 1.8)	1086 (x 1.8)	2320 (x 1.8)
	4	2067 (x 3.2)	1998 (x 3.3)	4006 (x 3.0)

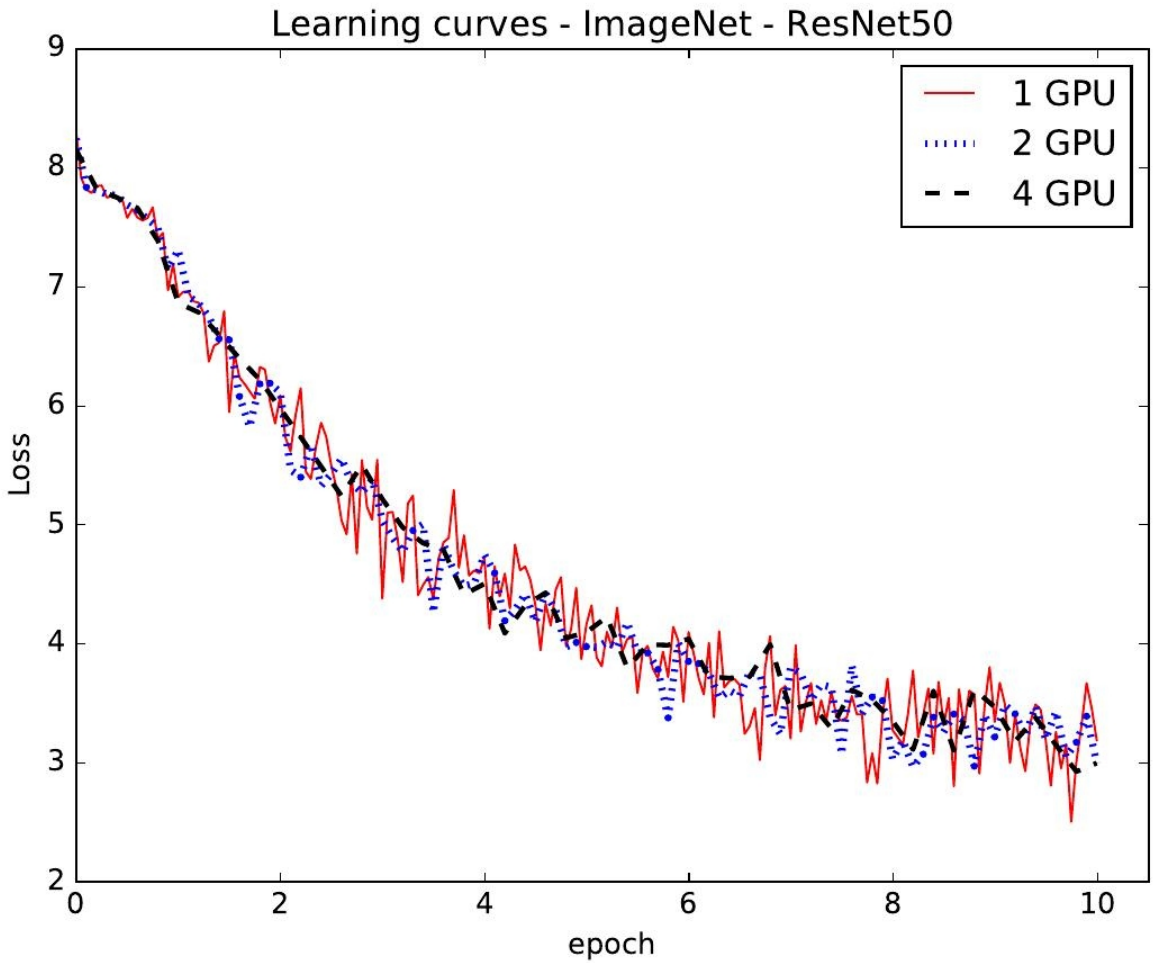
Comparison for ImageNet, udocker

Results: long training, ResNet50

# GPUs	LSDF-GPU 5 epochs				ForHLR-II 10 epochs			
	Accuracy		Training time hh:mm		Accuracy		Training time hh:mm	
	top_1	top_5			top_1	top_5		
1	0.36	0.63	33:45		0.53	0.78	30:23	
2	0.37	0.64	16:46	x 2.0	0.53	0.79	15:50	x 1.9
4	0.35	0.62	8:34	x 3.9	0.51	0.76	8:16	x 3.7

Comparison for ResNet50, ImageNet, udocker

Results: long training, loss evolution



ResNet50, ImageNet, udocker, ForHLR-II

Results: Distributed training with Horovod

Horovod works through MPI (and Infiniband on ForHLR-II) (https://github.com/horovod/horovod)

	TF	# GPUs	Data	ForHLR-II GTX 980 Ti, udocker	
1 node / max 4 GPU, std. TensorFlow	1.10	1	ImageNet	118	
	1.10	2	ImageNet	225	(x 1.9)
	1.10	4	ImageNet	426	(x 3.6)
Horovod, 2 nodes	1.10	2	synthetic	252	(x 2.1)
Horovod, 12 nodes	1.10	12	synthetic	1413	(x 12.0)
Horovod, 2 nodes	1.12	4	synthetic	492	(x 4.2)
Horovod, 2 nodes	1.12	8	synthetic	974	(x 8.25)
Horovod, 6 nodes	1.12	24	ImageNet	2437	(x 20.65 *))

preliminary

*) Batch size 32 instead of 64

OpenMPI 4.0 + horovod 0.16 → Docker image. ResNet50 via udocker on ForHLR-II

Summary & Outlook

- Practical application of using TF benchmarks packed in a Docker image as a tool to evaluate deep learning infrastructures
  - Best practices
  - Isolation
  - Portability
- udocker (and singularity) are used in a multi-user environment to run containers
- no overhead due to usage of containers is obtained in the multi-GPU case
- First results for distributed case (Horovod)

Outlook:

- More neural network architectures to test
- Systematic study on distributed training
- Integration with DEEPaaS API (https://deepaas.readthedocs.io)
- Try out other DL-oriented benchmarks (e.g. DawnBench, MLPerf, DeepBench)

backup

Parameters: Horovod case

job_horovod.sh:

```
OPTIONS="--batch_size=64 --optimizer=sgd \
--variable_update=horovod --display_every=200 --num_gpus=1 \
--model=resnet50 --data_name=imagenet --num_batches=100 \
--train_dir=/benchmarks/run_output/output \
--log_dir=/benchmarks/run_output/log \
--benchmark_log_dir=/benchmarks/run_output/log \
--benchmark_test_id=190517_1155 --data_dir=/benchmarks/data"

SCRIPT="python /benchmarks/tf_cnn_benchmarks.py $OPTIONS"

MPI_PARAMS="-bind-to none -map-by slot -x NCCL_DEBUG=INFO \
-x LD_LIBRARY_PATH -x PATH -x HOROVOD_MPI_THREADS_DISABLE=1 \
-mca pmix_server_usock_connections 1 -mca pml ob1 -mca btl ^openib"

mpirun $MPI_PARAMS \
udocker run --hostenv --hostauth --user=$USER -v ${HOST_DATA}:${UDOCKER_DATA} \
-v ${HOST_RUN}:${UDOCKER_RUN} ${CONTAINER} $SCRIPT
```

```
$> module load mpi/openmpi/4.0
$> sbatch -p visu --job-name=tfbench -nodes=6 --ntasks-per-node=4 \
 --time=8:00:00 --output=job.out --error=job.err job_horovod.sh
```