TARANTELLA
A FRAMEWORK FOR DISTRIBUTED DEEP LEARNING

Peter Labus, Ph.D.

Competence Center for High Performance Computing, Fraunhofer ITWM, Kaiserslautern.

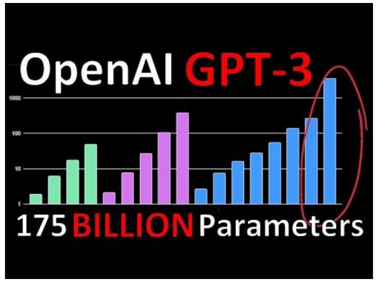
Fraunhofer Center Machine Learning.

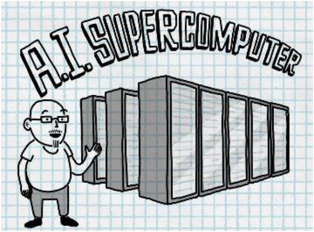
10th HPC Status Konferenz
Gauß-Allianz

October 1, 2020

Achievements in AI need exponentially growing computing power

Achievements in AI...



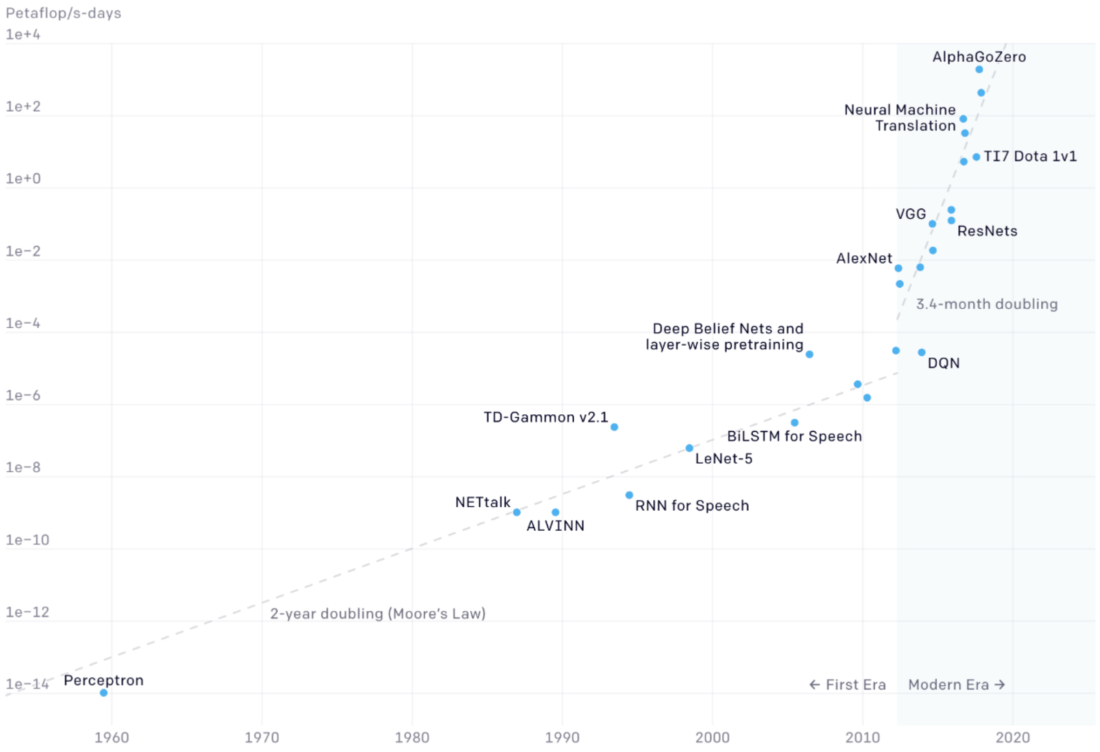


JULY 22, 2019 • 2 MINUTE READ

Microsoft Invests In and Partners with OpenAI to Support Us Building Beneficial AGI

Microsoft is investing $1 billion in OpenAI to support us building artificial general intelligence (AGI) with widely distributed economic benefits. We're

...need exponentially growing computing power



"Moore's Law of AI": FLOP/s-days double every 3.5 months

[https://openai.com/blog/ai-and-compute/]

Fraunhofer

Our distributed Deep Learning framework *Tarantella* has three objectives

distributed Deep Learning
with *Tarantella*

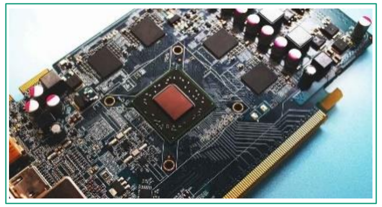
objectives

High usability without HPC expertise

- high-level user interface
- integrate well with existing tools (*TensorFlow2*)

Deep Learning without memory limits

- automatic use of pipelining & layer-parallelism



Good scalability on many HPC systems

- leverage highly optimized data parallel implementation based on GASPI
- vendor-independent solution

SPONSORED BY THE
Federal Ministry of Education and Research

ZIH
Center for Information Services & High Performance Computing

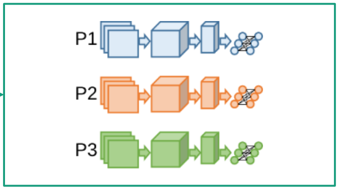
UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

Tarantella implements three parallelization strategies

Tarantella implements…

parallelization strategy		status

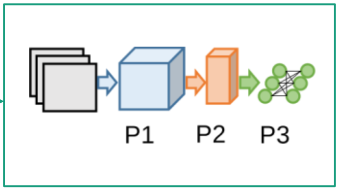
data parallelism



- **split data in micro-batches**
- easy to implement ✓
- best strong scalability ✓
- does not improve memory consumption ✗

release candidate

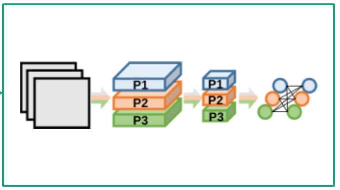
pipelining



- **split DNN in partitions**
- improves memory consumption ✓
- hard to achieve good load balancing ✗
- more difficult to implement ✗

functioning prototype

layer parallelism



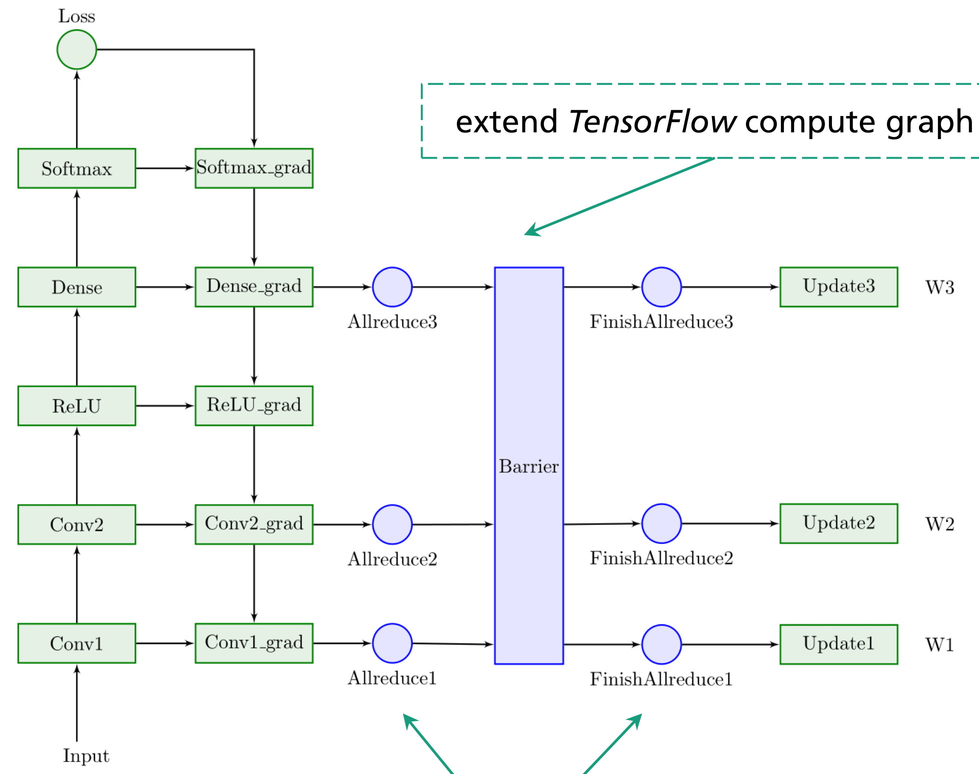
- **split layer in partitions**
- necessary for very large layers ✓
- can improve load balancing ✓
- hard & time-consuming to implement ✗

partial implementation

[Demystifying parallel and distributed deep learning, Ben-Nun et al.]

Tarantella's data parallelism overlaps *allreduces* with backpropagation

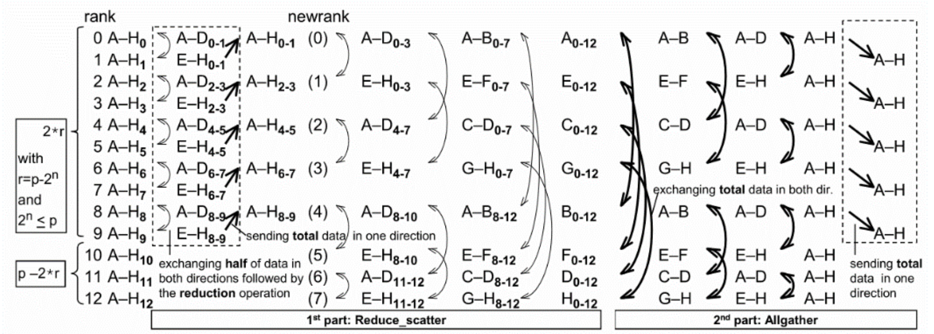
backpropagation in *Tarantella*



extend *TensorFlow* compute graph

non-blocking *allreduce* using GASPI & the C++ TF ops interface

allreduce



- *reduce scatter* & *allgather* with recursive halving / doubling
- bandwidth efficient algorithm
- interleave iterations of multiple *allreduces*
- communication thread triggers progress in background

Planned optimizations:

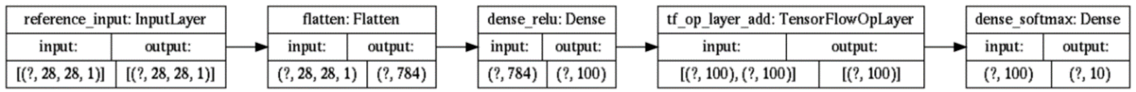
- fused gradient buffers
- *allreduce* algorithm for latency-bound case
- hierarchical *allreduce*

[Optimization of Collective Communication Operations in MPICH, R. Thakur et al.]

Fraunhofer

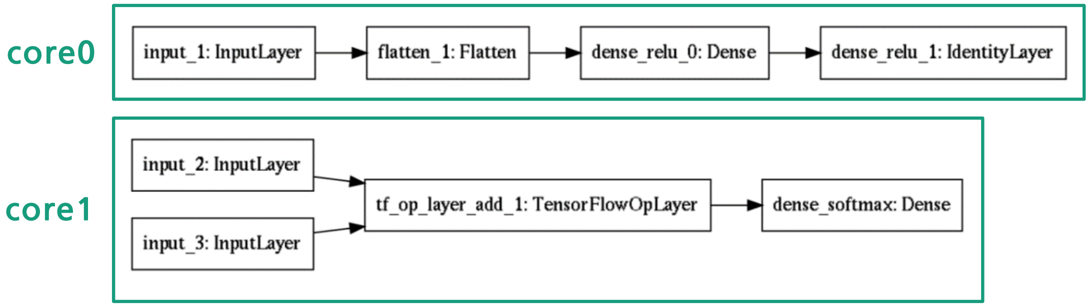
Tarantella's pipelining builds on Keras and GASPI

Init: Get Keras model from user



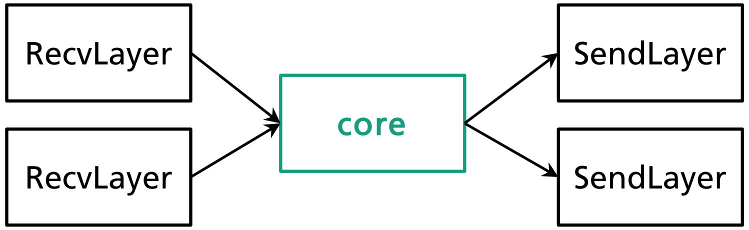
- support full Keras interface (compile, fit, evaluate, …)
- automatically manage dataset transformations / splitting

1. Step: Split user model into partitions



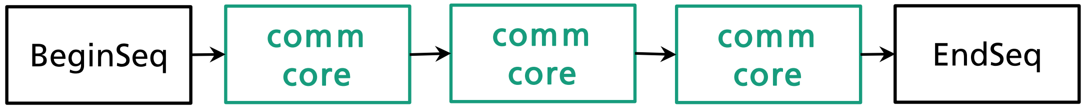
- **TODO:** auto-partitioning using performance model (ZiH)

2. Step: Add SendLayers & RecvLayers



- non-blocking *send* & blocking *recv* backend (GASPI & C++ TF ops)
- custom backward pass executes *recv*/*send*
- generate tags for each connection & micro-batch

3. Step: Replicate for all micro-batches & serialize



- share weights amongst all micro-batches
- fake losses / metrics / datasets
- generate datasets for micro-batches

Tarantella integrates well into existing *TensorFlow2 / Keras* models

TensorFlow2 / Keras model

```
import tensorflow as tf

# Create Keras model
model = tf.keras.Model(resnet50.get_model())

# Define optimizer with learning rate
sgd = tf.keras.optimizers.SGD(learning_rate=base_learning_rate)

# Build Keras model
model.compile(optimizer = sgd,
              loss = 'categorical_crossentropy',
              metrics = (['categorical_accuracy']))

# Load input data in mini-batches
train_dataset = tf.data.FixedLengthRecordDataset(filenames_train)
train_dataset = train_dataset.shuffle().repeat().batch(batch_size)
val_dataset = tf.data.FixedLengthRecordDataset(filenames_validation)

# Perform synchronous training
model.fit(train_dataset, nepochs, val_dataset)
```

Integration

- ***Tarantella* support is trivial to add**
- automatic distribution of datasets
- automatic partitioning of large models*
- advanced interface for pipelining for power-users*

Tarantella model

```
import tensorflow as tf

# Step 1: initialize the framework
import tarantella as tnt
tnt.init()

# Create Keras model
model = tf.keras.Model(resnet50.get_model())

# Step 2: wrap the model
model = tnt.TarantellaModel(model)

# Define optimizer with appropriate learning rate for large batch sizes
sgd = tf.keras.optimizers.SGD(learning_rate=base_learning_rate)

# Build Keras model
model.compile(optimizer = sgd,
              loss  ='categorical_crossentropy',
              metrics = (['categorical_accuracy']))

# Load input data (which will be read distributedly in micro-batches)
train_dataset = tf.data.FixedLengthRecordDataset(filenames_train)
train_dataset = train_dataset.shuffle().repeat().batch(batch_size)
val_dataset = tf.data.FixedLengthRecordDataset(filenames_validation)

# Perform distributed synchronous training
model.fit(train_dataset, nepochs, val_dataset)
```

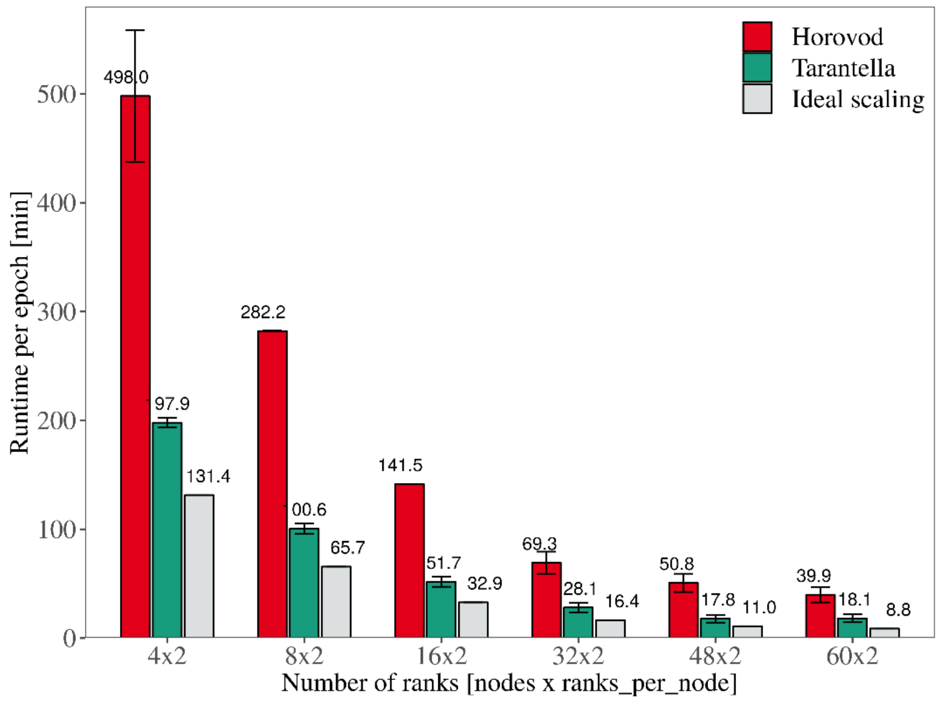
execute *Tarantella* with

```
tarantella_run -n 8 -ngpuspernode 4 -m machinefile \
./models/resnet50.py --batch-size=1024 -e 100
```

The first large scale benchmarks…

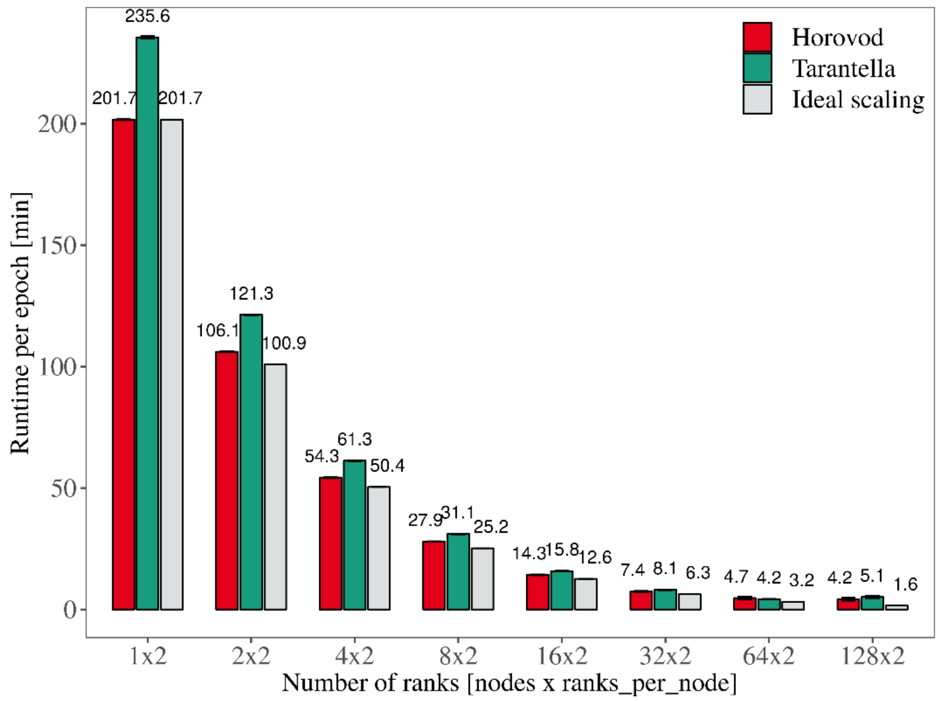
- image classification: ResNet50 on ImageNet
- TensorFlow 2.2, Horovod 0.20.2
- 10 epochs, micro-batch size = 256

SeisLab, ITWM Kaiserslautern



- 2 x Intel® Xeon® Gold 6148 CPU @ 2.40GHz (20 cores/40 hyperthreads)
- Mellanox ConnectX-5 Infiniband network with 100 Gbit/s
- OpenMPI 1.10.7

SuperMUC-NG, LRZ Munich

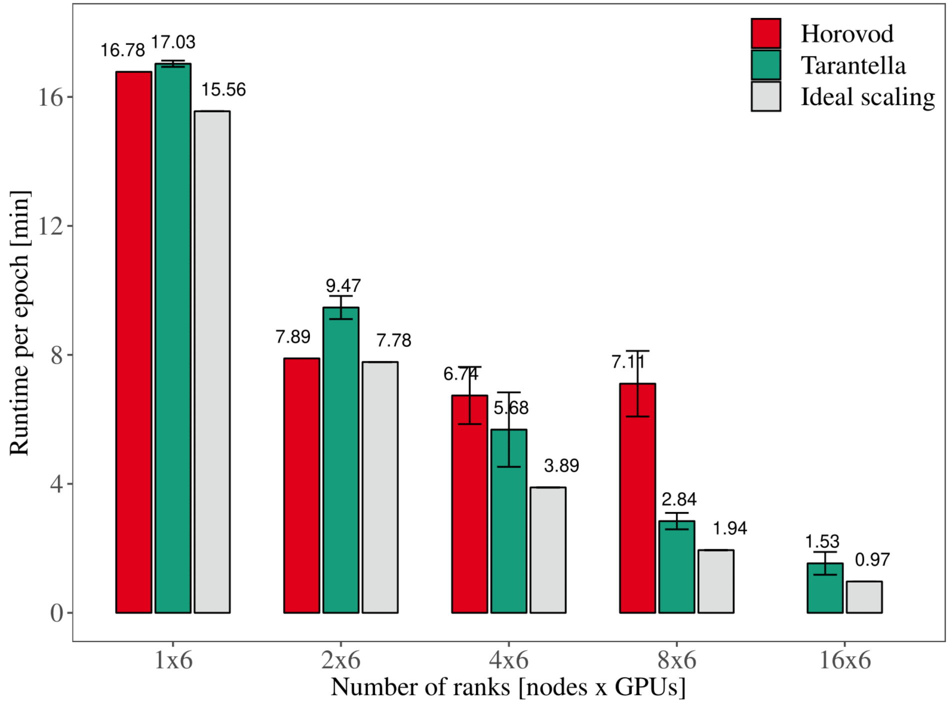


- 2 x Intel® Skylake® Xeon® Platinum 8174 CPU @3.10GHz, 3.90GHz boost (48 cores/node)
- Intel® OmniPath network with 100 Gbit/s
- Intel MPI 2019

…show promising results

- image classification: ResNet50 on ImageNet
- TensorFlow 2.2, Horovod 0.19.6
- 10 epochs, micro-batch size = 32

HPC-DA, ZiH Dresden



Results:

- compatible / better than state-of-the-art (*Horovod*)
- good *strong scalability*
- further optimizations to be exploited

Next steps:

- ***MLPerf*** benchmark suite evaluation
- full model parallelism **in active development**

- 2 x IBM Power9 CPU (2.80 GHz, 3.10 GHz boost, 22 cores)
- 6 x NVIDIA VOLTA V100 with 32GB HBM2
- Mellanox EDR Infiniband network with 100 Gbit/s
- NVLINK bandwidth 150 GB/s between GPUs and host
- openMPI 3.1.4 & NCCL 2.4.8

Distributed Deep Learning with *Tarantella*: summary & outlook

Distributed Deep Learning with *Tarantella*

1. …builds on and integrates with time-proven software
2. …leverages data & model parallelism for large scale Deep Learning without memory limits
3. …brings easy-to-use scalability to the Deep Learning community

***Tarantella* open source release date:**

16.11.2020

www.tarantella.org

SC20 Everywhere we are | more than hpc.