Deploying Machine Learning at Petascale on Secure Large Scale HPC Production Systems with Containers.

5.Feb.2020| Atanas Atanasov, Fabio Baruffa, David Brayford, Sofia Vallecorsa, Walter Riviera

Legal Notices & Disclaimers

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer. No computer system can be absolutely secure.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit http://www.intel.com/performance.

Cost reduction scenarios described are intended as examples of how a given Intel-based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

Statements in this document that refer to Intel's plans and expectations for the quarter, the year, and the future, are forward-looking statements that involve a number of risks and uncertainties. A detailed discussion of the factors that could affect Intel's results and plans is included in Intel's SEC filings, including the annual report on Form 10-K.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

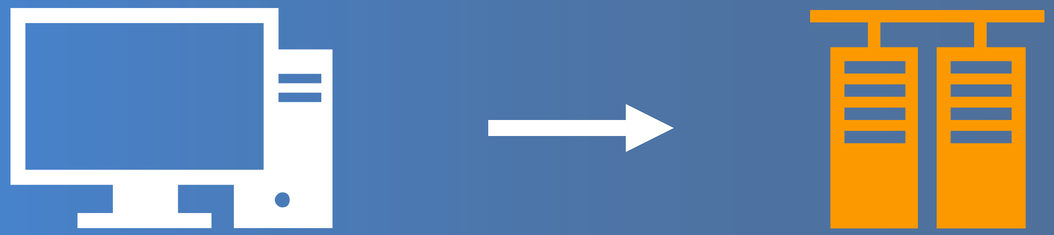
Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

Intel, the Intel logo, Pentium, Celeron, Atom, Core, Xeon, Movidius and others are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

High Performance AI (HPAI) in a **Container**

Transition AI algorithms from the
laptop to supercomputer
with minimal effort



“It just works”

HPAI =

Modeling & Simulation

- Equation based on model
- Computing driven
- Numerically intensive
- Creates simulations
- Monte Carlo
- Larger problems
- Iterative methods
- PDE

+

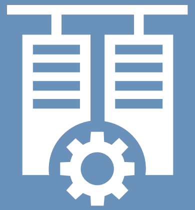
- Linear algebra
- Matrix operations
- Iterative methods
- Compute intensive
- Data transfer
- Predictive
- Probabilities
- Stencil codes
- Calculus
- Pattern recognition
- Graphs

Analytics

- Finds patterns
- Correlations in data
- Logic driven
- Creates inferences
- Knowledge discovery
- Graphs
- Data-driven science
- Predictions
- CNN
- RNN

Requirements for AI on HPC

Compute intensive hardware



Optimized AI frameworks
TensorFlow, PyTorch, Caffe

Optimized software
numerical libraries, Python

HPC specific software
distributed computing, workload manager

Method of deploying the AI software
in a simple, straight-forward and flexible way

Need to get to: “It just works”

Key Challenges

Package Management

Frameworks have conflicting dependencies

The frameworks & their dependencies need to be combined in a single module

Rapid update cycles

Provide a mechanism for users to build there own frameworks

Dynamic Programming Environment

Python dependencies

Each unique framework needs its own Python instance

Connecting to external servers

Build frameworks on systems without internet access

Intel Optimized Machine learning Frameworks

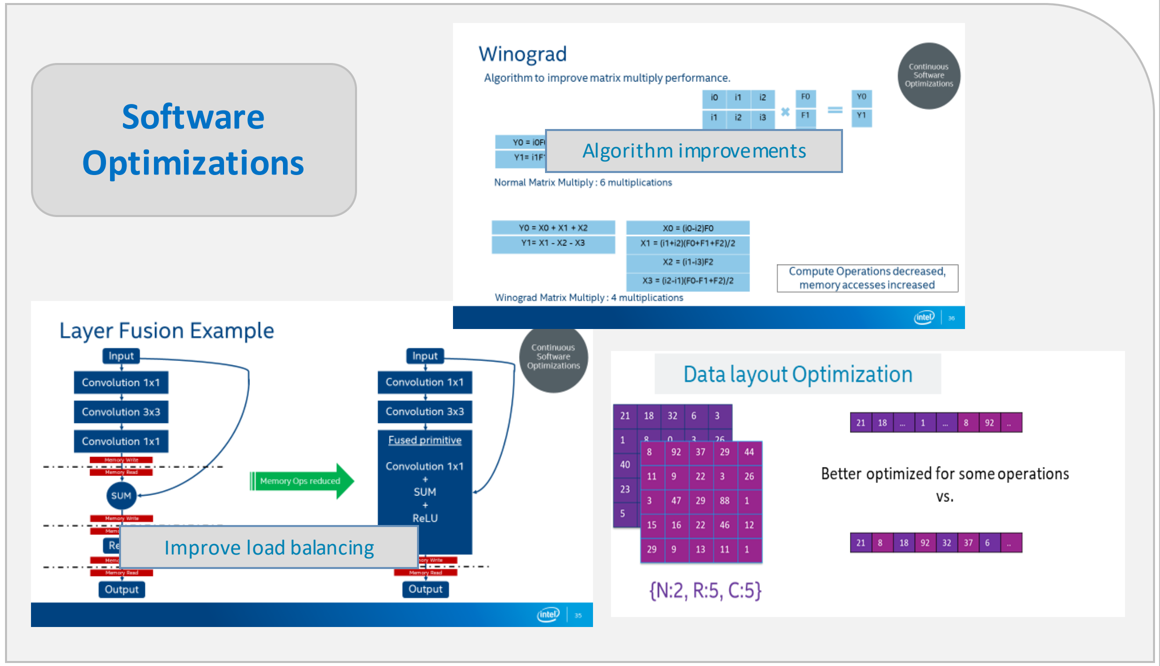
Intel AI Software Stack

OpenVINO™ TOOLKIT · Microsoft CNTK · torch · Chainer · Caffe · PaddlePaddle · BigDL · theano · PYTORCH · dmlc · Caffe2 · mxnet · TensorFlow

Intel MKL

Intel MKL-DNN

Intel Processors

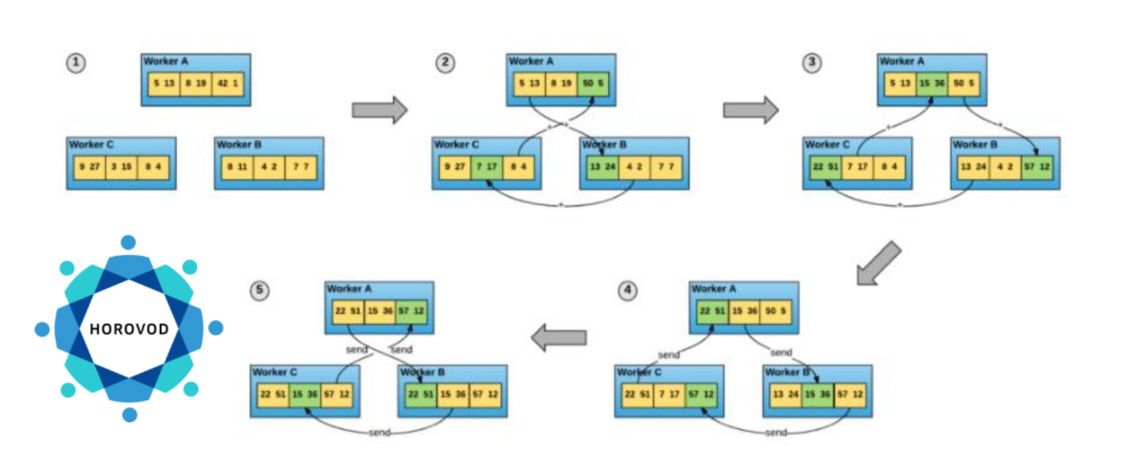


Optimized vs Out of Box Framework Performnce

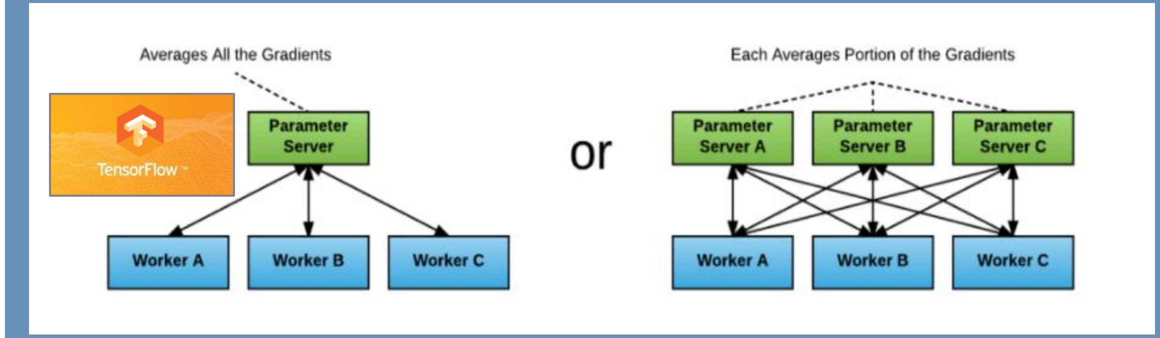
TensorFlow

14x ResNet-50	3.2x Inception-V3
TRAINING THOUGPUT IMPROVEMENT	INFERENCE THOUGPUT IMPROVEMENT

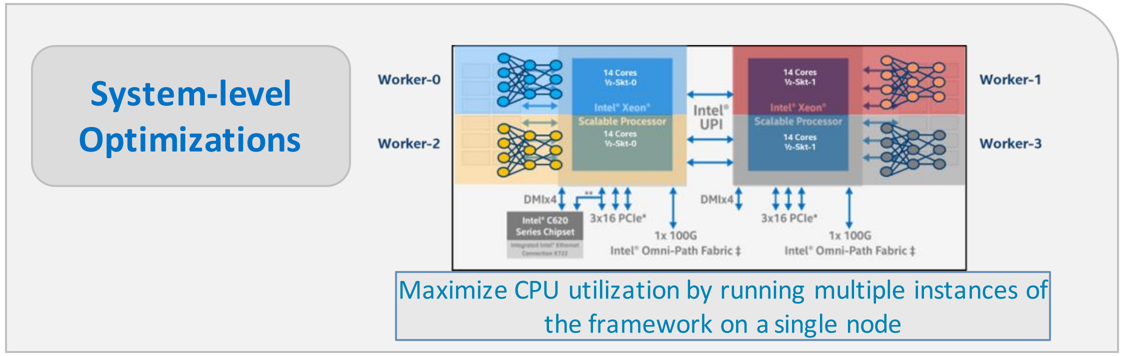
Distributed Mechanisms

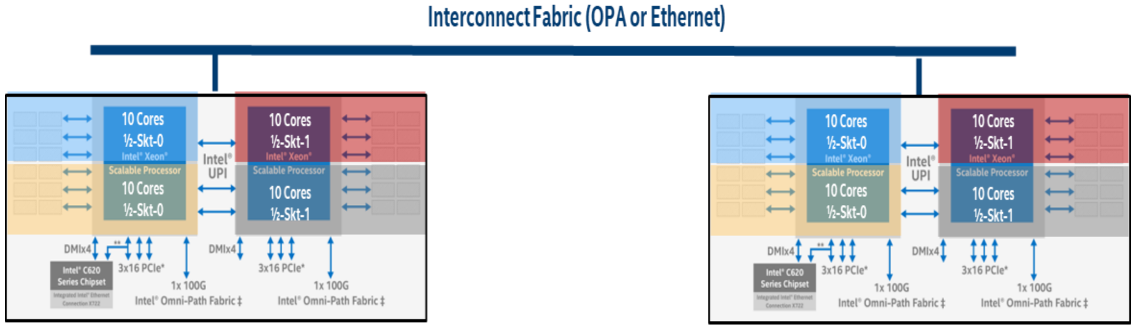


VS



System-level Optimizations

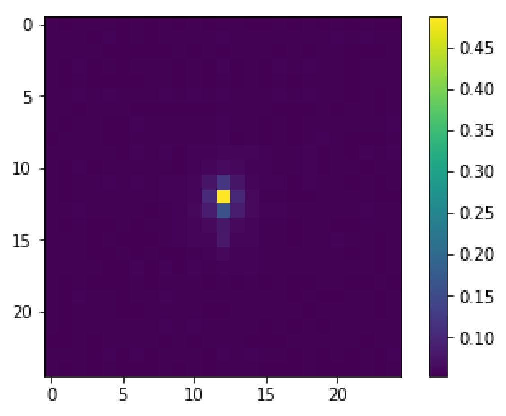


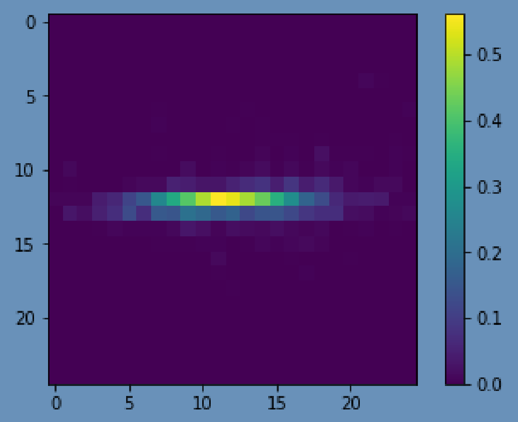


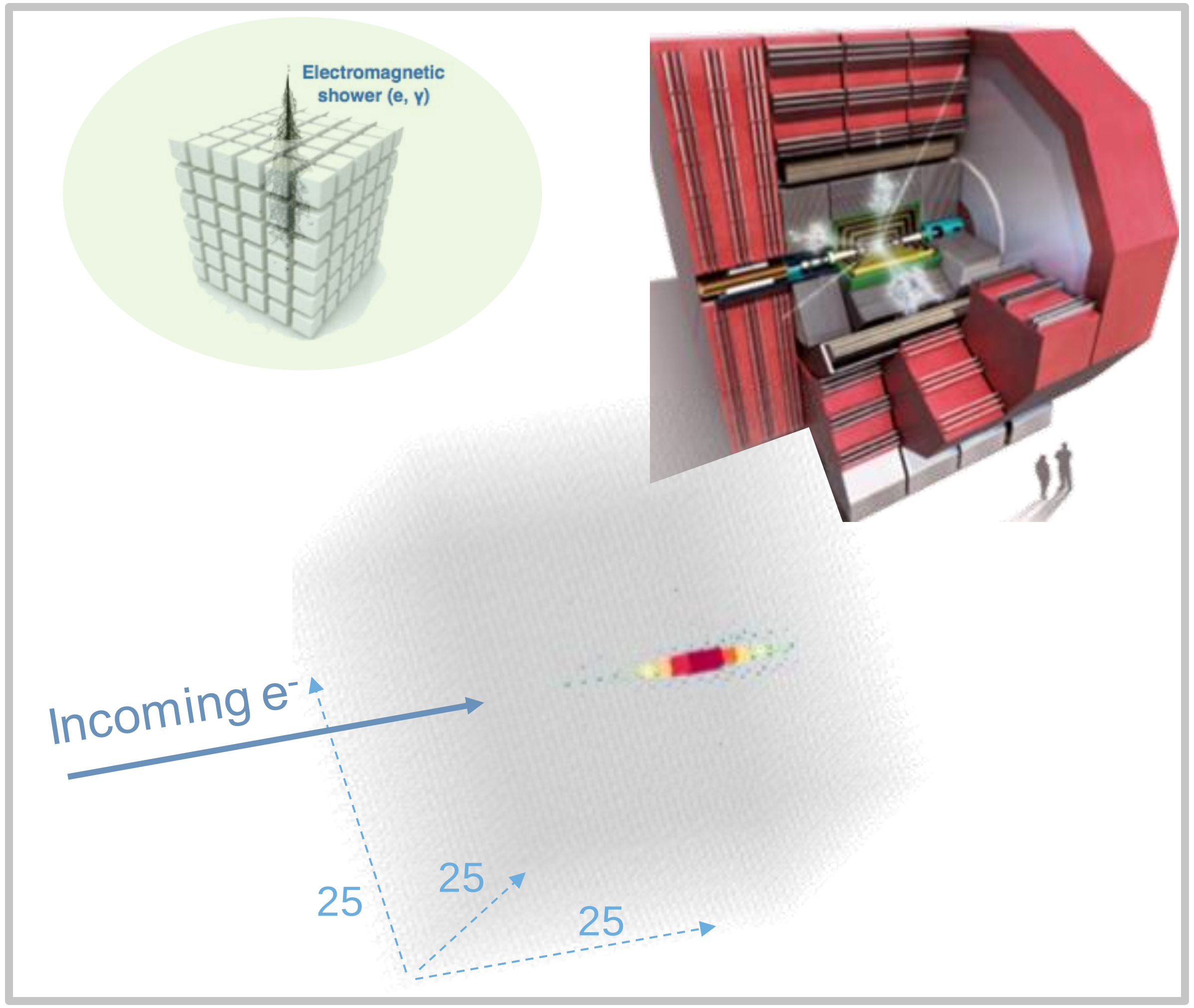
CERN Compact Linear Collider (CLIC)

Detecting and Identifying High Energy Physic Particles

- CLIC Electromagnetic calorimeter
  - Sparse images
  - Highly segmented (pixelized)
  - Large dynamic range
- Segmentation is critical for particle identification and energy determination



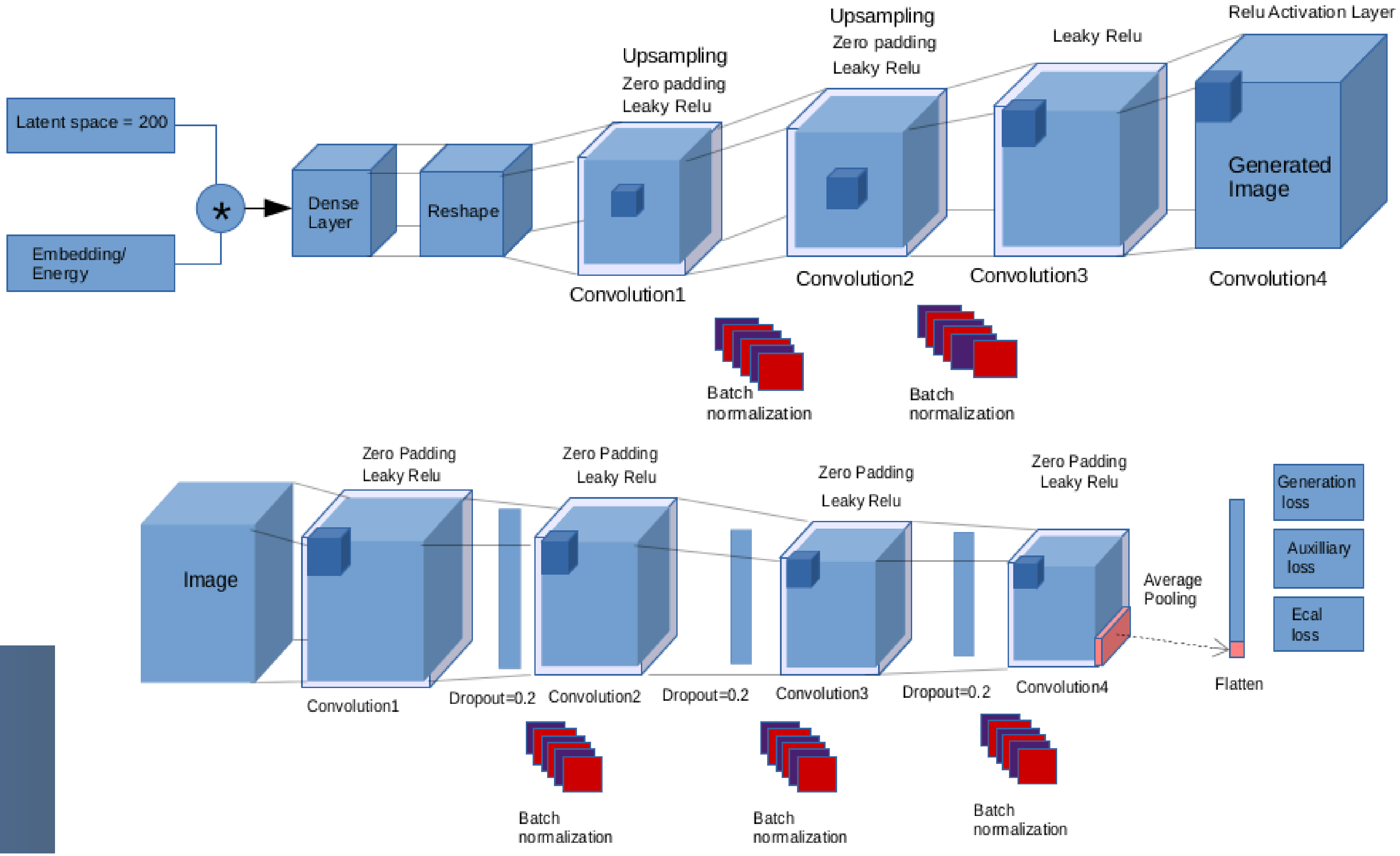




Future

3D Convolutional GAN

Generator



Discriminator

~1M parameters

Total model Size: 3.8MB

Charliecloud Containers in HPC

- Easy to install
- Charliecloud was developed to be run on highly secure HPC systems at US government labs
- Charliecloud runs entirely under the User ID
- Ability to run legacy design flows in containers
- Low overhead and ~ 800 lines of code
- LRZ deploys Charliecloud via Spack
- Charliecloud is available in the module system at LRZ

Deployment@LRZ

Achieving High Performance AI on Secure HPC Systems

Mechanism for deploying AI at LRZ

- Download the Intel optimized TensorFlow Docker Image (intelaipg Dockerhub)
- Modify the Linux Docker image for HPC
- Modify Python to enable distributed TensorFlow execution
- Copy the training data and execution scripts to the modified Docker image
- Convert to a Charliecloud UDSS and copy the file to the HPC system
- Load the Charlicloud module
- Execute on SuperMUC-NG via Slurm

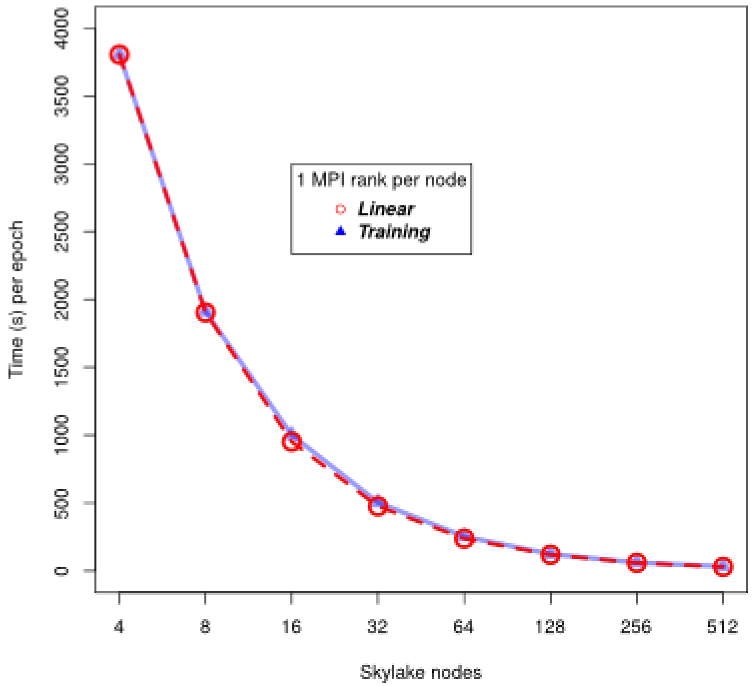
Deployment @ LRZ

Distributed TensorFlow Results LRZ SNG 1 MPI Rank per Node

1 MPI rank & 48 OpenMP threads per node

Intel Skylake Platinum Xeon 8174

Nodes	Training Time(S) per Epoch	Linear Time(S) per Epoch	Scaling Efficiency
4	3806	3806	-
8	1910	1903	99.6%
16	1001	951.5	95.1%
32	504	475.75	94.4%
64	253	237.87	94%
128	124	118.93	95.9%
256	61	59.46	97.5%
512	33	29.73	90.1%



Throughput Overheads

Benchmark	Free System Memory with Charliecloud (GB)	Free System Memory without Charliecloud (GB)
AlexNet with cifar	331.29	331.33
ResNet50 with imagenet	324.47	324.89

Memory Overheads

Benchmark	Free System Memory with Charliecloud (GB)	Free System Memory without Charliecloud (GB)
AlexNet with cifar	331.29	331.33
ResNet50 with imagenet	324.47	324.89

3DGAN Execution on SNG with >= 2 MPI Ranks per Node

Hyperthreading, 48 OpenMP threads per MPI task & 2 MPI ranks per node Intel Skylake Platinum Xeon 8174, Standard horovod + MPI

Nodes	Training Time(S) per Epoch	Linear Time(S) per Epoch	Scaling Efficiency
4	2302	2302	-
8	1238	1151	93%
16	638	575.5	90.2%
32	323	287.75	89.1%
64	164	143.87	87.7%
128	88	79.93	81.8%
256	47	35.96	76.6%
512	25	17.98	71.9%

12 OpenMP threads per MPI task & 4 MPI ranks per node Intel Skylake Platinum Xeon 8174, Standard horovod + MPI

Nodes	Training Time(S) per Epoch	Linear Time(S) per Epoch	Scaling Efficiency
4	959	959	-
8	507	479.5	94.6%
16	264	239.75	90.8%
32	137	119.87	87.5%
64	72	59.93	83.3%
128	39	29.96	76.8%
256	21	14.98	71.4%
512	12	7.49	62.5%

3DGAN Execution on SNG using Intel MPI from the System

Mounted the LRZ file system into the container and used the system version of Intel MPI.

ch-run -b /lrz/sys/.:/lrz/sys/ -w container_name – python /location/in/container/training_script.py

Nodes	Training Time(S) per Epoch	Linear Time(S) per Epoch	Scaling Efficiency
4	907.26	907.26	-
8	479.52	453.63	94.6%
16	244.42	226.82	92.8%
32	124.22	113.41	91.3%
64	62.24	56.70	91.1%
128	31.22	28.35	90..8%
256	15.63	14.18	90.7%
512	7.84	7.09	90.4%
768	3.94	3.54	89.9%

Nodes	Measured Performance petaflops	Percentage of Theoretical Peak
4	0.01099	66.17%
8	0.02199	66.21%
16	0.04450	67.01%
32	0.08386	63.14%
64	0.17313	65.17%
128	0.31878	67.60%
256	0.70547	66.39%
512	1.39412	65.60%
768	2.08143	65.29%

Beyond 768 nodes the constant set up costs become the dominant factor.

First Quarter 2020

2020

General HPC Docker image
Verified recipes to enable the deployment of AI on HPC systems using secure containers

Current Users

DLR German Aerospace Center, PyTorch, inferencing of high resolution satellite images on SuperMUC-NG

New Users & Infrastructure

More users; cloud providers; additional ML, AI & data analytics software; different operational modes.

Documentation & Contacts

- **High Performance AI (HPAI)**
- Github repository https://github.com/DavidBrayford/HPAI

- **Online Documentation**
- https://docs.docker.com/
- https://hpc.github.io/charliecloud/tutorial.html

- **Contacts**
- brayford@lrz.de (via LinkedIn)

Demonstration:

Using Charliecloud to Deploy a Containerized TensorFlow Workflow on a HPC System in the Cloud with the OpenHPC Software Stack.

5.Feb.2019| Atanas Atanasov, David Brayford

Acknowledgements

- **Acknowledgements**
- Adrian Reber (RedHat)
- Chris Downing (Amazon)
- Sarosh Quraishi (Intel)
- OpenHPC (https://openhpc.community/)

Docker Commands

- Install Docker on your local system.
- Download Docker images from DockerHub:
  - sudo docker pull image
- Build your own Docker image:
  - sudo docker build –t my_image ./Dockerfile
- View images:
  - sudo docker images
- Run a docker image:
  - sudo docker run –itd my_image

Docker Commands

- List all active Docker images:
  - sudo docker ps –a
- Start a bash shell in the Docker container:
  - sudo docker exec –it <container_ID> /bin/bash
- Install software in the container:
  - apt-get install
  - pip install
  - make
- Exit out of the container:
  - exit

Docker Commands

- List all active Docker images:
  - sudo docker ps –a
- Save the modified images:
  - sudo docker commit <CONTAINER_ID> new_container_name

Charliecloud Commands

- Build Docker image using Charliecloud:
  - ch-build -t hello .
- Build the Charliecloud compressed flat file:
  - sudo ch-builder2tar <Docker_file_name> /dir/to/store/
- Copy the tar.gz file to the HPC system:
- Unpack the Charliecloud tar.gz image:
  - ch-tar2dir Docker_file_name.tar.gz /foo/bar/

Charliecloud Commands

- Load the Charliecloud module:
- Execute the Charliecloud containerized command:
  - ch-run –w <container_name> -- bash
  - ch-run -w <container_name> -- python /model/train.py
  - ch-run -b /lrz/sys/.:/lrz/sys/ -w <container_name> -- bash
- Distributed execution line in a Slurm script:
  - mpiexec -n $SLURM_NTASKS -ppn $SLURM_NTASKS_PER_NODE ch-run -b /lrz/sys/.:/lrz/sys/ -w ./container_name -- python /model/train.py