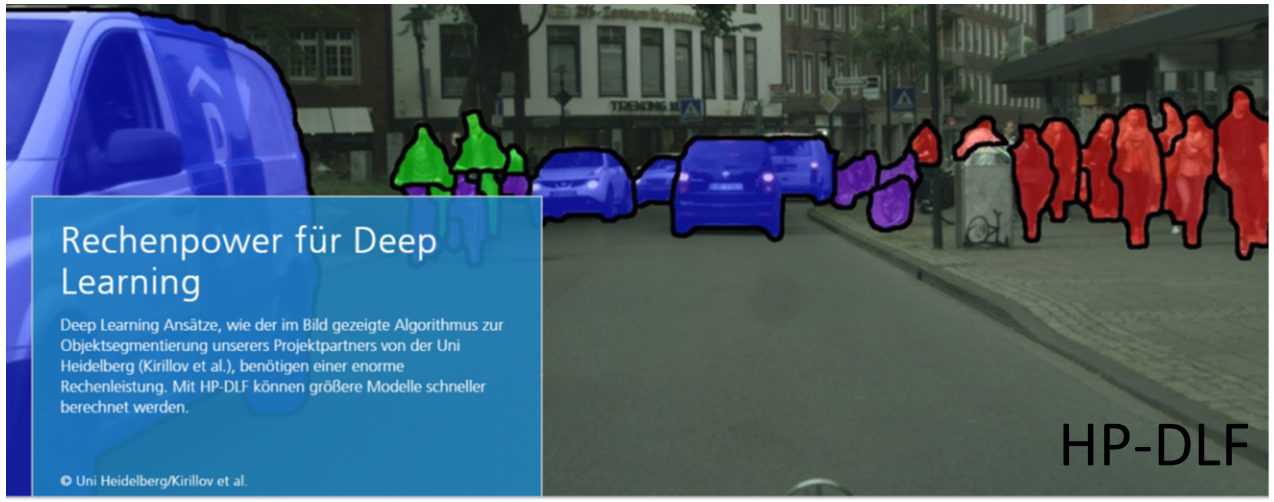
HPC and Deep Learning



Franz-Josef Pfreundt, Janis Keuper, Peter Labus

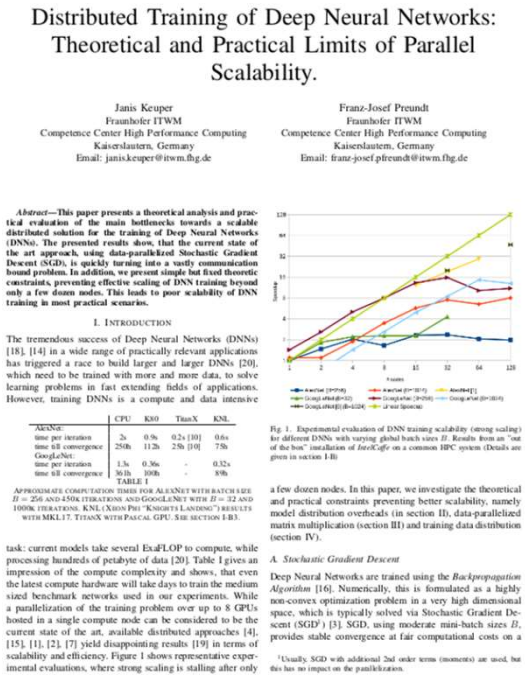
Competence Center for HPC , Fraunhofer ITWM

Distributed Training of DNN´s

Limitations of Scalability

1. **Communication bounded**
2. **Skinny Matrix Multiplication**
3. **I/O Bottleneck**

High Performance – Deep Learning Framework HP-DLF



Distributed Training of Deep Neural Networks: Theoretical and Practical Limits of Parallel Scalability.

Janis Keuper
Fraunhofer ITWM
Competence Center High Performance Computing
Kaiserslautern, Germany
Email: janis.keuper@itwm.fhg.de

Franz-Josef Preundt
Fraunhofer ITWM
Competence Center High Performance Computing
Kaiserslautern, Germany
Email: franz-josef.pfreundt@itwm.fhg.de

Abstract—**This paper presents a theoretical analysis and practical evaluation of the main bottlenecks towards a scalable distributed solution for the training of Deep Neural Networks (DNNs). The presented results show, that the current state of the art approach, using data-parallelized Stochastic Gradient Descent (SGD), is quickly turning into a vastly communication bound problem. In addition, we present simple but fixed theoretic constraints, preventing effective scaling of DNN training beyond only a few dozen nodes. This leads to poor scalability of DNN training in most practical scenarios.**

I. Introduction

The tremendous success of Deep Neural Networks (DNNs) [18], [14] in a wide range of practically relevant applications has triggered a race to build larger and larger DNNs [20], which need to be trained with more and more data, to solve learning problems in fast extending fields of applications. However, training DNNs is a compute and data intensive task: current models take several ExaFLOP to compute, while processing hundreds of petabyte of data [20]. Table I gives an impression of the compute complexity and shows, that even the latest compute hardware will take days to train the medium sized benchmark networks used in our experiments. While a parallelization of the training problem over up to 8 GPUs hosted in a single compute node can be considered to be the current state of the art, available distributed approaches [4], [15], [1], [2], [7] yield disappointing results [19] in terms of scalability and efficiency. Figure 1 shows representative experimental evaluations, where strong scaling is stalling after only a few dozen nodes. In this paper, we investigate the theoretical and practical constraints preventing better scalability, namely model distribution overheads (in section II), data-parallelized matrix multiplication (section III) and training data distribution (section IV).

	CPU	K80	TitanX	KNL
AlexNet:				
time per iteration	2s	0.9s	0.2s [10]	0.6s
time till convergence	250h	112h	25h [10]	75h
GoogLeNet:				
time per iteration	1.3s	0.36s	-	0.32s
time till convergence	361h	100h	-	89h

TABLE I
Approximate computation times for AlexNet with batch size $B = 256$ and 450k iterations and GoogLeNet with $B = 32$ and 1000k iterations. KNL (Xeon Phi "Knights Landing") results with MKL17, TitanX with Pascal GPU. See section I-B3.

Fig. 1. Experimental evaluation of DNN training scalability (strong scaling) for different DNNs with varying global batch sizes B. Results from an "out of the box" installation of *IntelCaffe* on a common HPC system (Details are given in section I-B)

A. Stochastic Gradient Descent

Deep Neural Networks are trained using the *Backpropagation Algorithm* [16]. Numerically, this is formulated as a highly non-convex optimization problem in a very high dimensional space, which is typically solved via Stochastic Gradient Descent (SGD[1]) [3]. SGD, using moderate mini-batch sizes B, provides stable convergence at fair computational costs on a

[1]Usually, SGD with additional 2nd order terms (momenta) are used, but this has no impact on the parallelization.

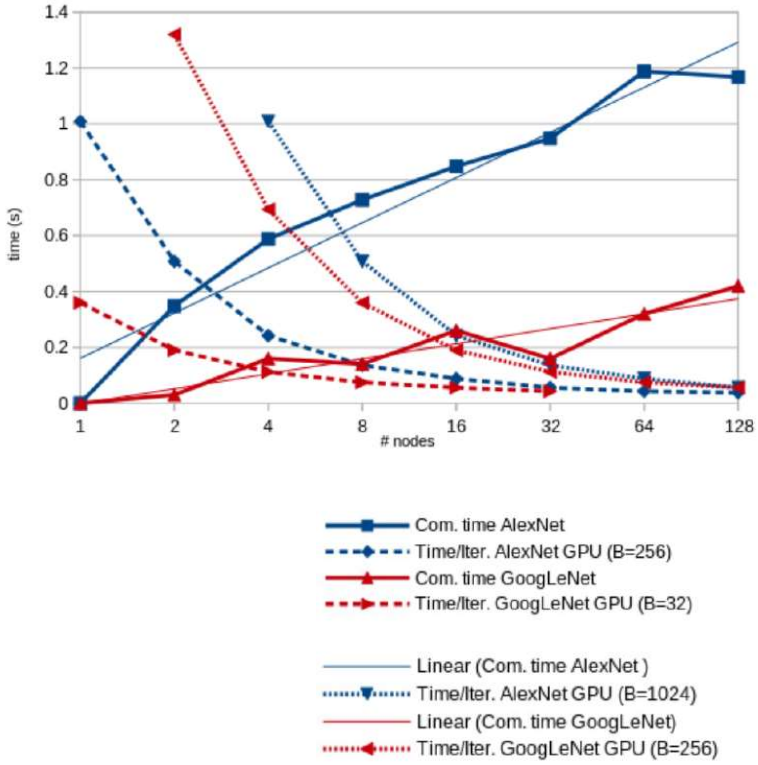
Paper SC 2016

Parallelism in DNN Training

A) Internal Parallelization: on layer level
- Mostly **dense matrix multiplication**,
- Task parallelization for special Layers (GPU)

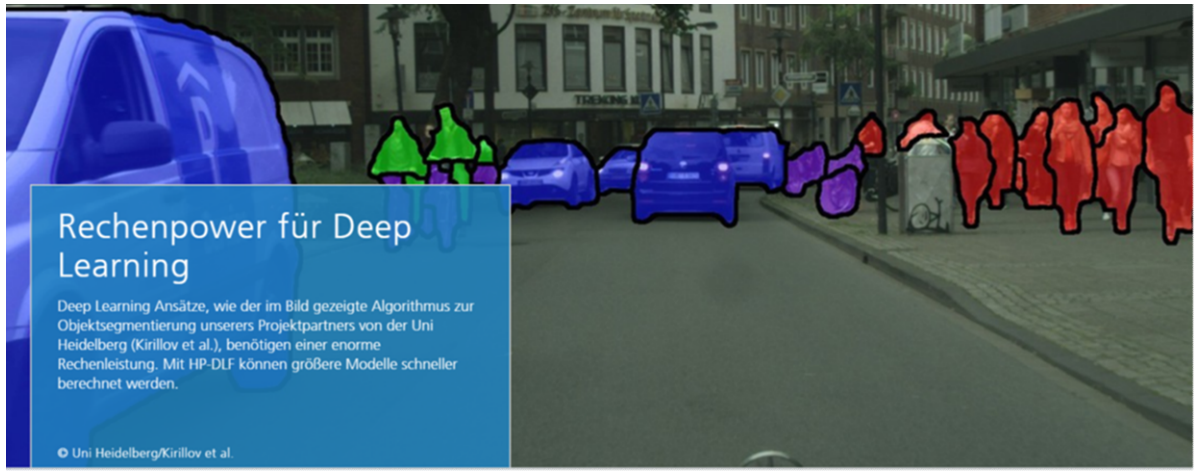
B) Data Parallelization over the Batch Data
- Limited Batch size → limited parallelism
- Every GPU (worker) runs only on a little part of the test data
- Full network & data have to be in memory

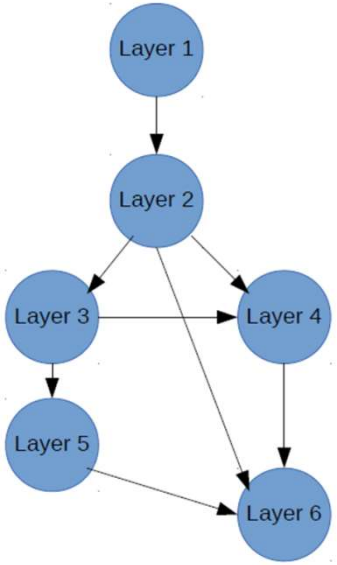
The standard case in a SGD parallelization



Model Parallelism

DNN models get large - Model and data do not fit in memory any more





→ Distribute the model over several workers : Can be done with Tensoflow but is very laborious
→ There is no other training framework that adresses this problem

HP-DLF wants to adress this problem in a very user friendly way

HP-DLF Project Partners

	Partner
Fraunhofer ITWM	Fraunhofer Gesellschaft e.V. Fraunhofer ITWM CC-HPC Dr. Franz-Josef Pfreundt
TECHNISCHE UNIVERSITÄT DRESDEN / ZIH Zentrum für Informationsdienste und Hochleistungsrechnen	Technische Universität Dresden Zentrum für Informationsdienste und Hochleistungsrechnen Prof. Dr. Wolfgang Nagel
UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386	Universität Heidelberg Visual Learning Lab Prof. Dr. Carsten Rother
DFKI	Deutsches Forschungszentrum für Künstliche Intelligenz Prof. Philipp Slusallek

Project start : 1.11.2017

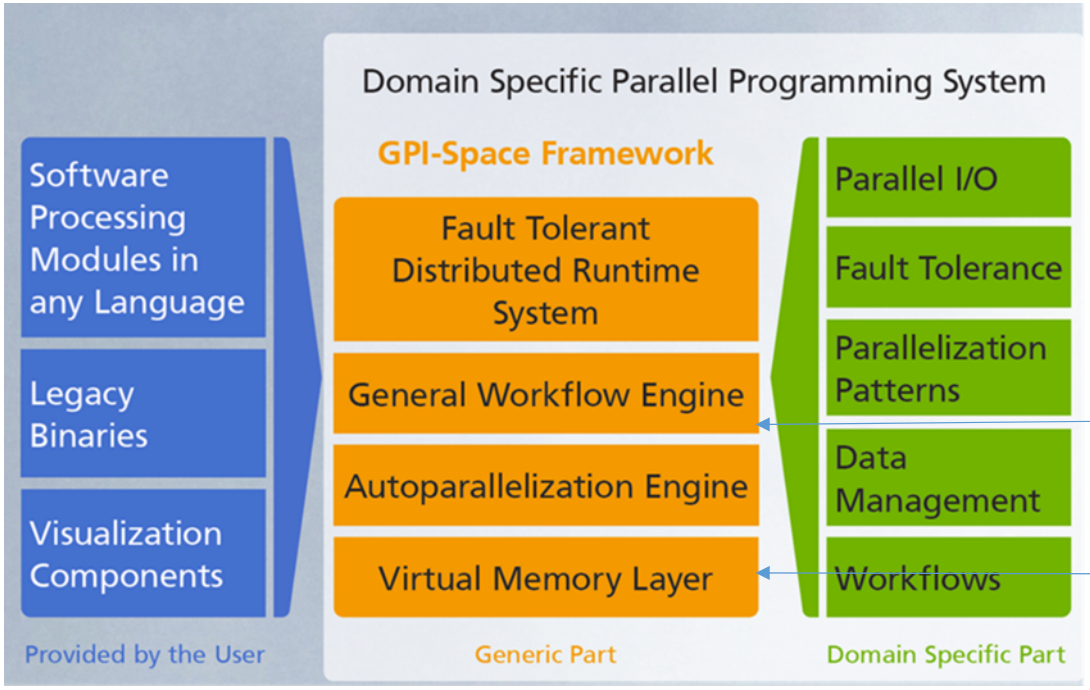
As part of the ML calls in 2017

HP-DLF goals:

- DNN training and inference on large, heterogeneous HPC systems
- Hide hardware complexity from user
- Scalable and energy efficient
- High portability
- Allow training of very large models without HPC knowledge

GPI-SPACE

Programming Enviroment for Memory Driven Computing



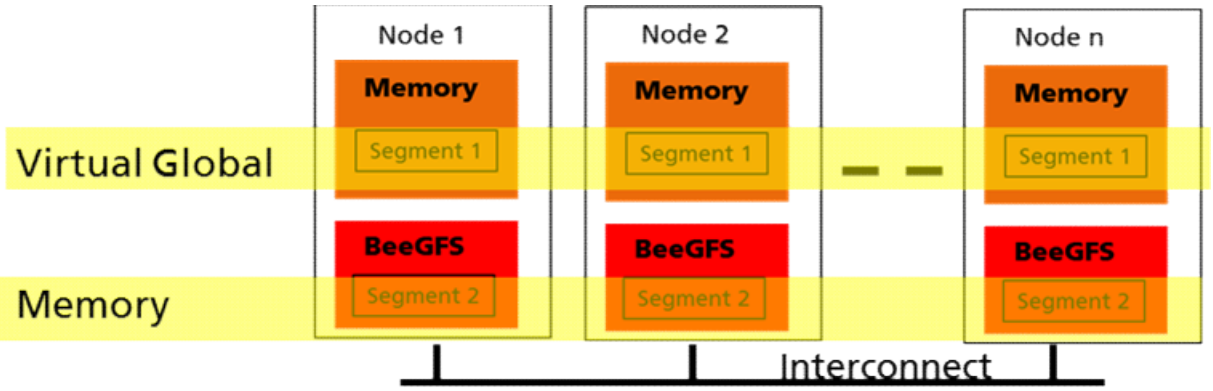
Currently used commercially in Oil& Gas Appllications

Big Data Appllications

Petri -Nets

Application Independeent Global memory

Application Independent Global Memory : VGM



Memory segments (GPI)

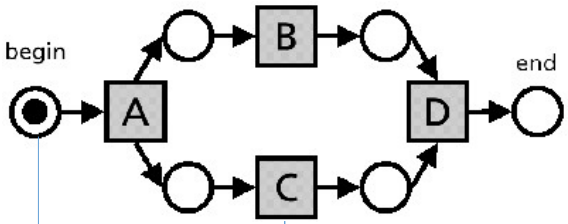
in HBM, DRAM, on accelerators

Memory Segments in
BeeGFS/BeeOND
(non volatile part)

API : allocate/delete memory , transfer cost, locality information , ...

- Can keep data without an application running
- All data data transfer is managed though the VGM
- Allows to couple tasks written in different languages
- Individual task can crash without effecting data

Coordination Language : Modified Petri Nets



Petri Nets

Carl Adam Petri 1962: Description language for asynchronous and concurrent systems in order to add resources to running jobs

Separates activation from execution

Task :

- Petri Net
- Expression
- Single threaded C++ Code
- .....
- Matlab module
- MPI / GPI Programm

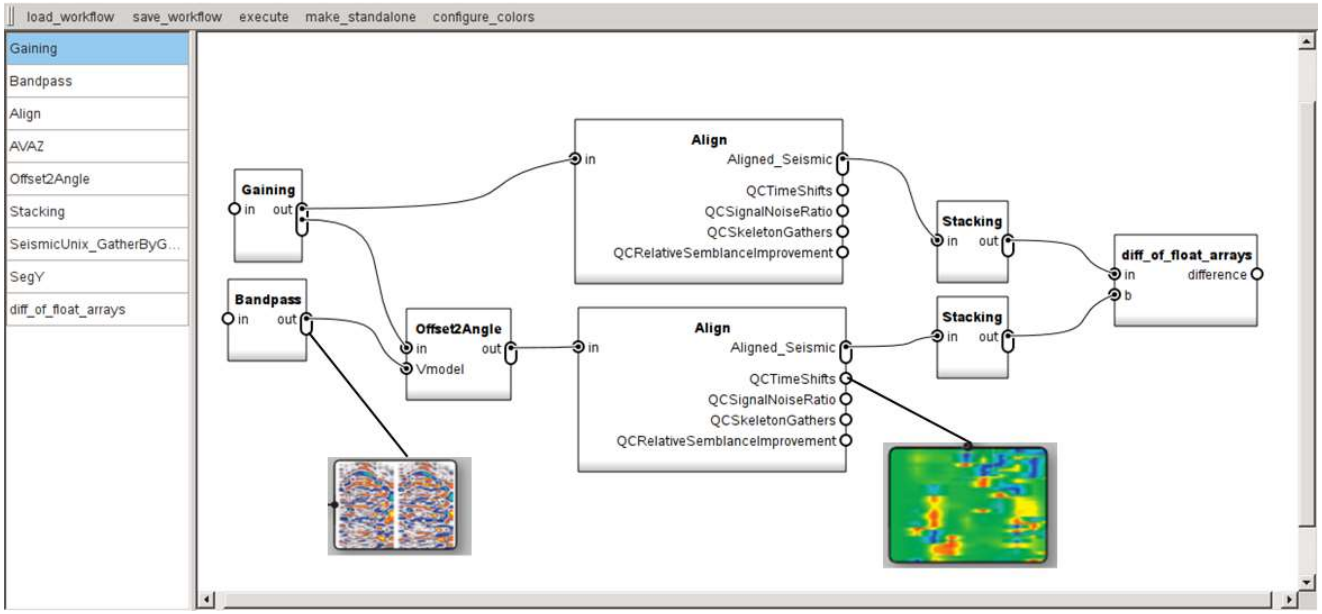
Data Parallelism:

Place with a token :

Tokens are consumed by the tasks and generate tokens on the output places
Tokens represent the data, multiple tokens (data parallelism)

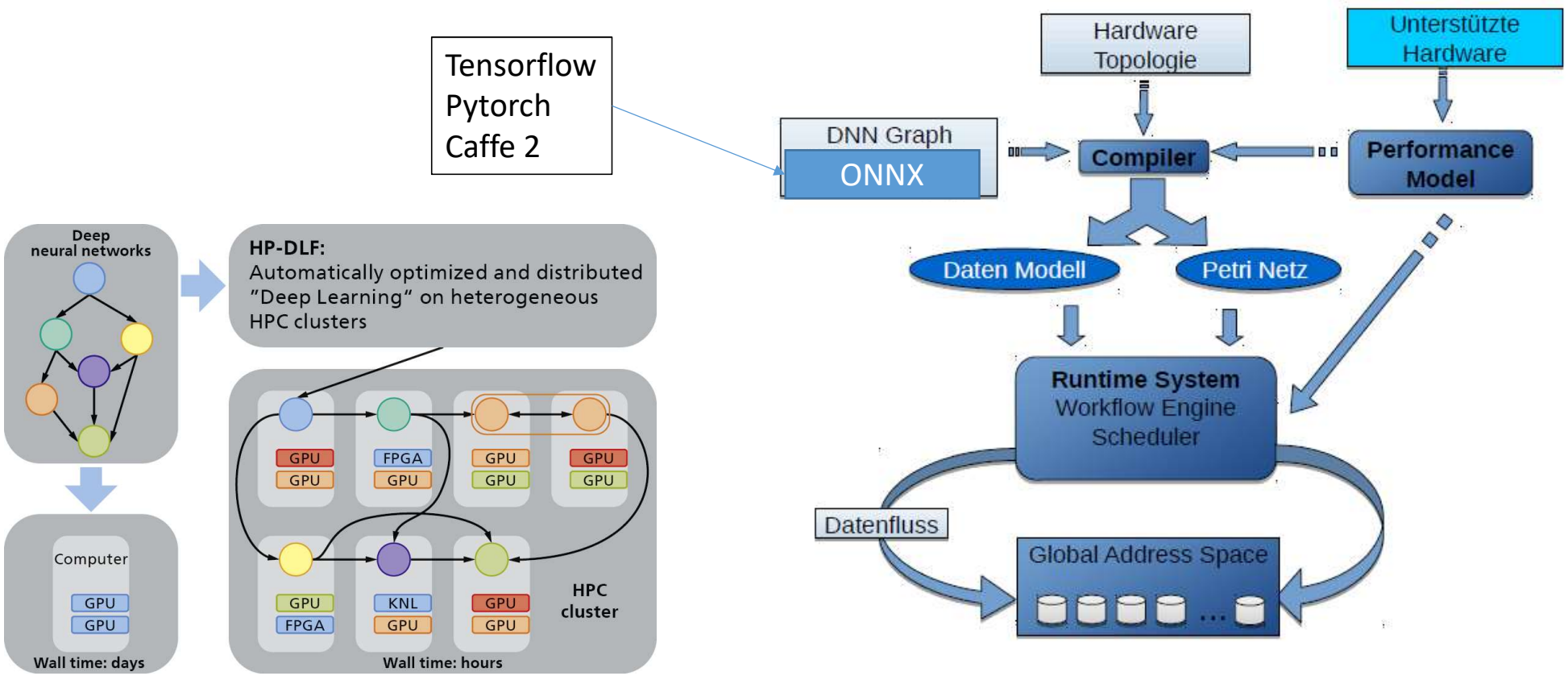
O&G Industry : Domain specific programming system

Graphical Programming & Plug in Architecture



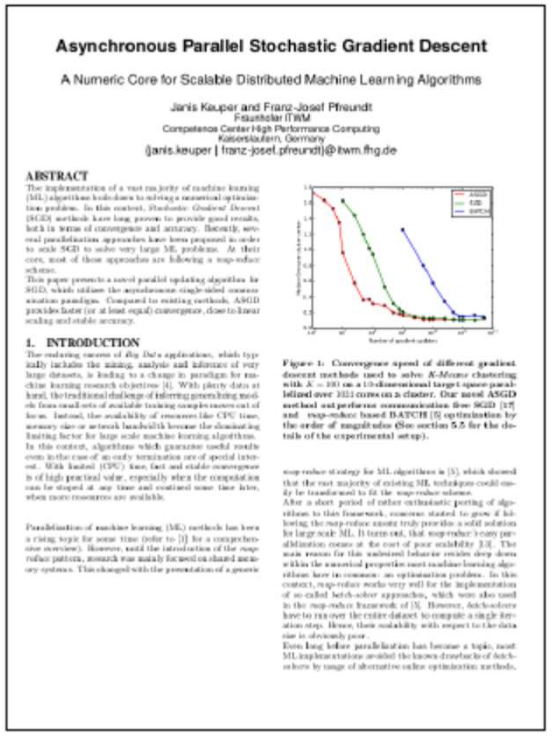
- Compiler generated Petri –Nets
- Parameter description language → Automatic GUI generation

HP-DLF Workplan



Ergebnisse bisher

- Teil des ONNX Standards ist implementiert
- Compiler als Test Prototype vorhanden
- Vampir Integration von GPI-Space
- Gesamt Prototyp vorhanden
- Benchmarks können durchgeführt werden
  - Inferenz für einfache CNNs geht (LeNet, AlexNet, VGG)
  - Training mit dem Asynchronen SGD fast fertig
    (Reduktion Kommunikation)
  - Aktuell RESnet und U-net in Arbeit
    (ONNX Interface Impelmentierung)



Asynchronous Parallel Stochastic Gradient Descent

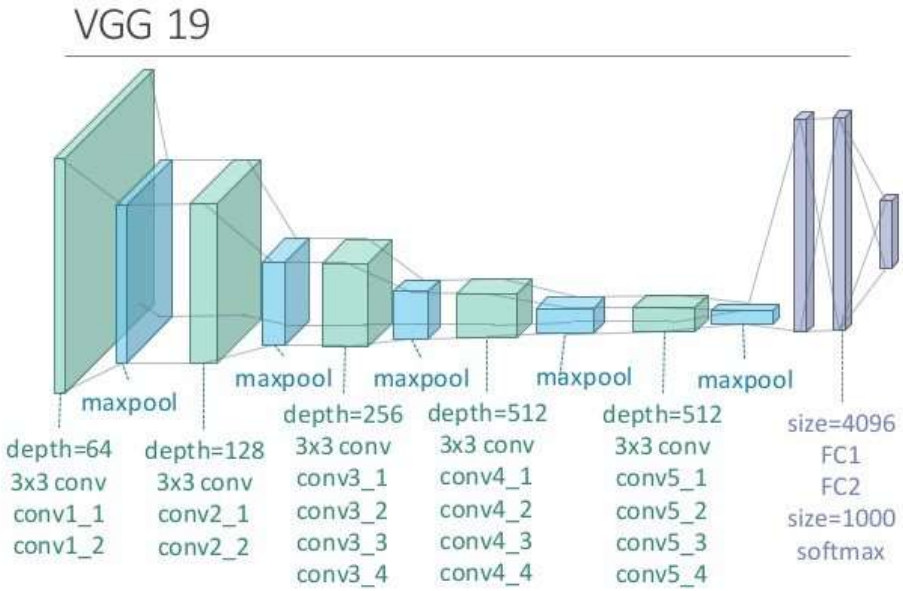
A Numeric Core for Scalable Distributed Machine Learning Algorithms

Janis Keuper and Franz-Josef Pfreundt
Fraunhofer ITWM
Competence Center High Performance Computing
Kaiserslautern, Germany
(janis.keuper | franz-josef.pfreundt)@itwm.fhg.de

ABSTRACT

1. INTRODUCTION

Beispiel VGG19

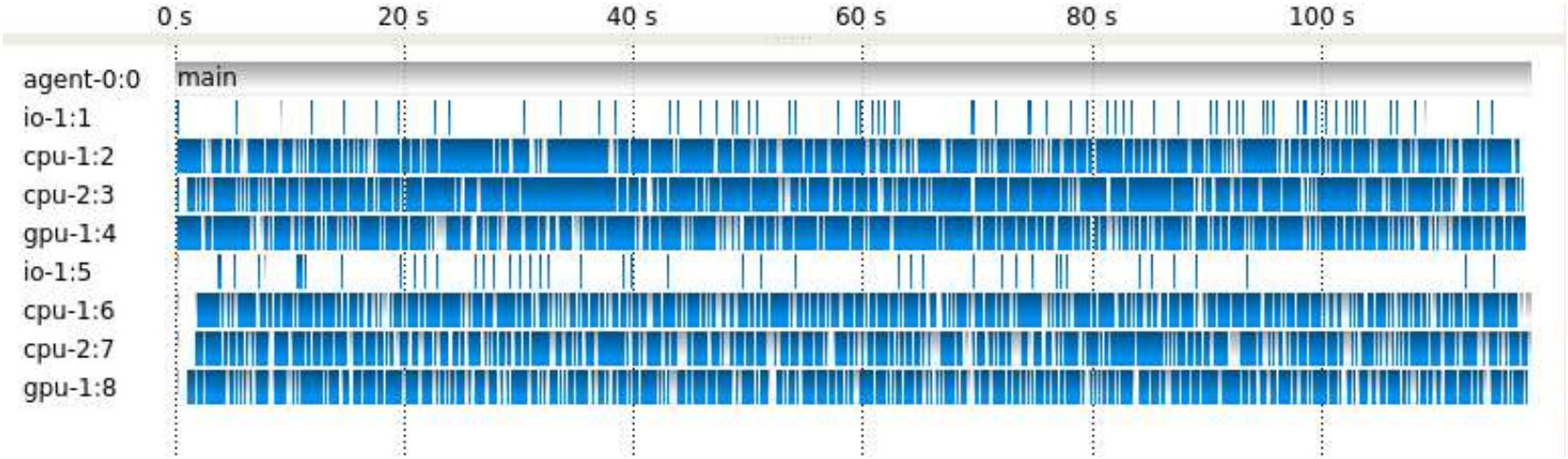


Kleines Netz: Nur 144 Mio Paramater

Compiler Challenge:

Automatisches Zusammenführen verschiedener Layer um Daten Transfers zu sparen

Vampir Diagramm auf 2 Knoten mit je 4 Workern



Nur Inference (kein Overlap von Model & Daten Parallelität)

@SC2018 in Dallas : Größere Probleme & Anzahl Knoten

ML users on HPC Systems

Jeder Nutzer hat seine bevorzugte

typischerweise verschiedene

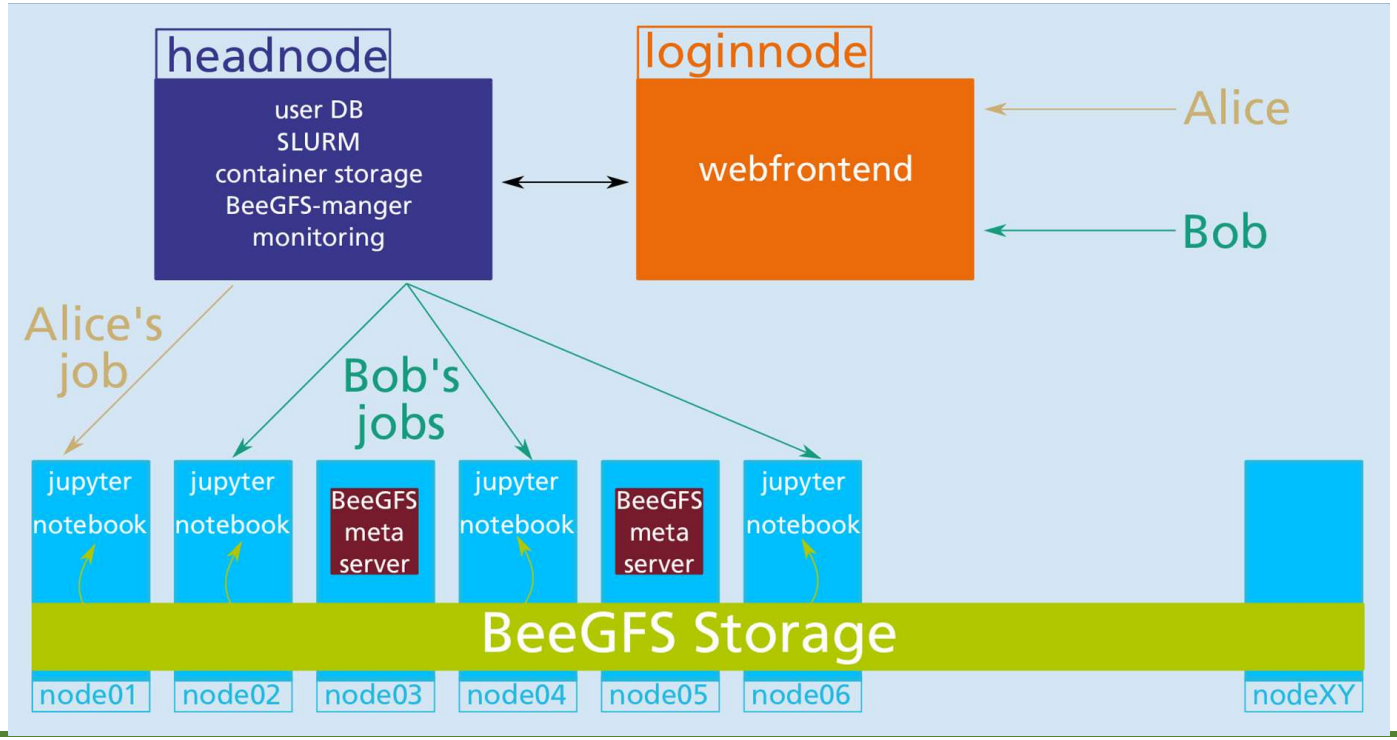
Arbeitsumgebung

Typisch Interaktive Nutzung

Batchsysteme Unbekannt

Carme

– An Open Source Framework for Multi-User, Interactive Machine Learning on Distributed GPU-Systems –



www.open-carme.org

Zugang zum System via Web Portal

Derzeit genutzt für DL Kurse

Applied ML Project

combine established open source ML tools with HPC back-ends

- ***Jupyter Notebooks*** as main web based GUI front-end
  - *web front-end*
- job management and scheduler
  - ***SLURM***
- data I/O technology
  - ***BeeGFS***
- use containers
  - ***Singularity***
- monitoring tools
  - ***Zabbix***