# *Project ‘Carme’*

## *An Open Source Framework for Multi-User, Interactive Machine Learning and Data Analytics on Distributed (GPU) Systems*

***www.open-carme.org***

Competence Center High Performance Computing Fraunhofer ITWM, Kaiserslautern

Fraunhofer

# Motivation

## Data Analytics and Machine Learning

- Large investments in (multi)-GPU hardware for data analytics and machine (deep) learning

- Main problems AFTER buying the hardware:

  - How to manage the resources?
  - How to scale applications to more than one GPU?
  - How to manage data I/O and storage?

# Motivation

## Data Analytics and Machine Learning

- Large investments in (multi)-GPU hardware for data analytics and machine (deep) learning

- Main problems AFTER buying the hardware:
  - How to manage the resources?
  - How to scale applications to more than one GPU?
  - How to manage data I/O and storage?

**NOT a solution*:**

+

* we are seeing this being done

## Part I: Low Cost Hardware Setup

ITWM’s high performance multi-GPU cluster build on gaming hardware

## Part II: ‘*Carme*’

ITWM’s open source software stack for multi-user GPU clusters

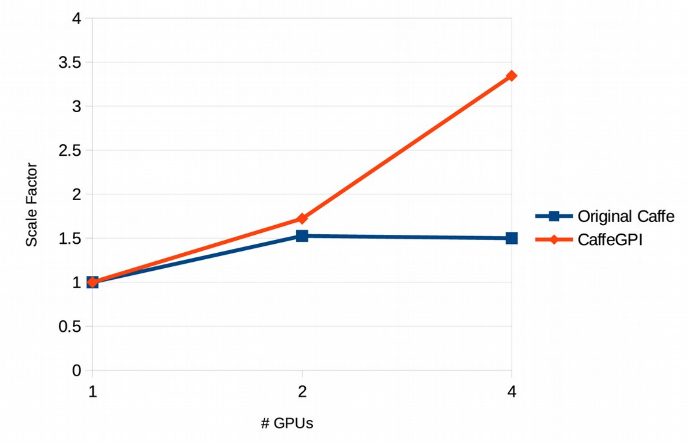
# Deep Learning Setup

## Multi-GPU Hardware Overview

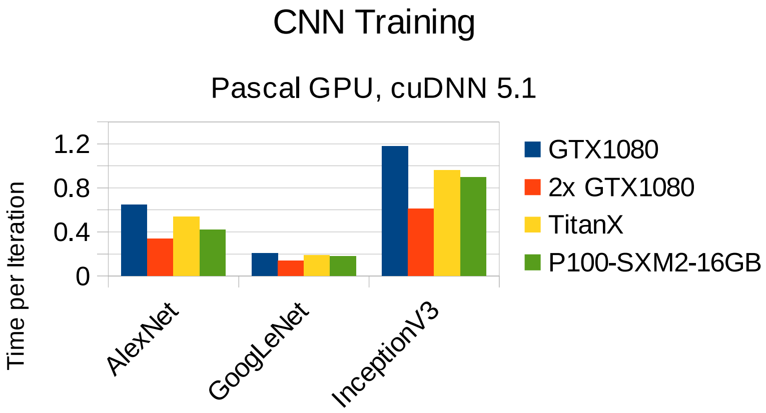
- **High Performance (NVLINK) Multi-GPU system are expensive!**
  - NVIDIA DGX-1 (8x P100/V100) ~ 130k EUR*
  - IBM Witherspoon (6x V100) ~ 60k EUR*
- **PCIe hosting of GPUs is to slow (!)**

  → **no multi-GPU training possible**
- **Gaming GPUs like GTX1080ti are a cheap alternative**
  - e.g. price GTX1080ti (11 GB) ~600 EUR*,

    P100 (16 GB)~ 6000 EUR*
  - Some Benchmarks:



Scaling performance of a Multi-GPU training on a PCIe System (blue line)



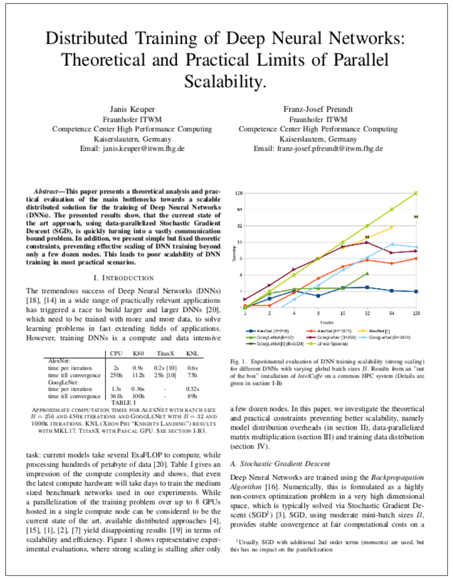
* estimated retail prices

# Low Cost Deep Learning Setup

## ITWM Approach

- **Research results: GPU performance is hardly ever the bottleneck**
  - I/O is the dominating factor
    - GPU-2-GPU and GPU-2-CPU communication
    - Train data I/O

- **Main Ideas:**
  - Combine cheap gaming hardware (GPUs, CPU and main boards) with sophisticated HPC components (Network)
  - Use existing ITWM HPC Software-Stack
    - BeeGFS distributed File System
    - GPI Communication Model
    - CaffeGPI and HP-DLF Deep Learning Frameworks



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 (moments) are used, but this has no impact on the parallelization.

Based on our paper from
ACM SuperComputing 16

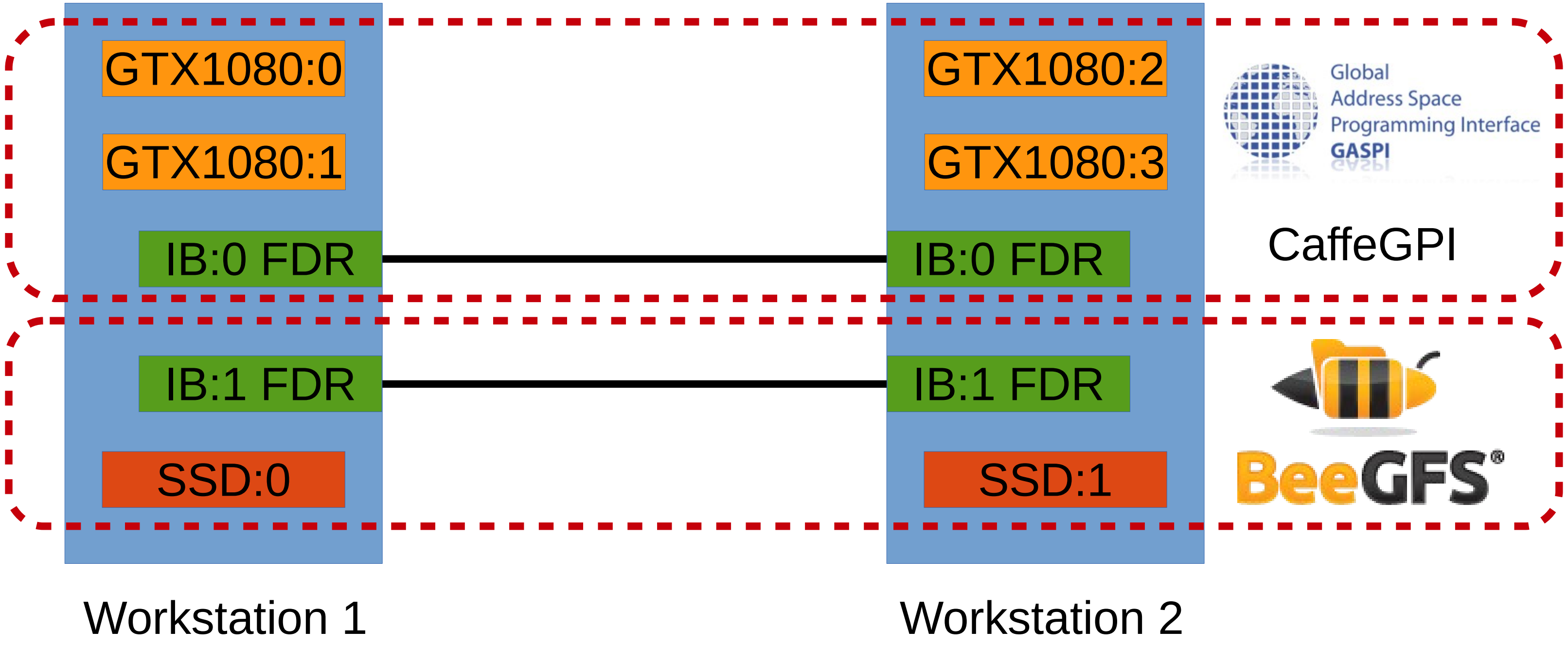
BeeGFS®

Global
Address Space
Programming Interface
GASPI

# Low Cost Deep Learning Setup

## Prototype 2-Node System

**Price: <8000 EUR, standard components**

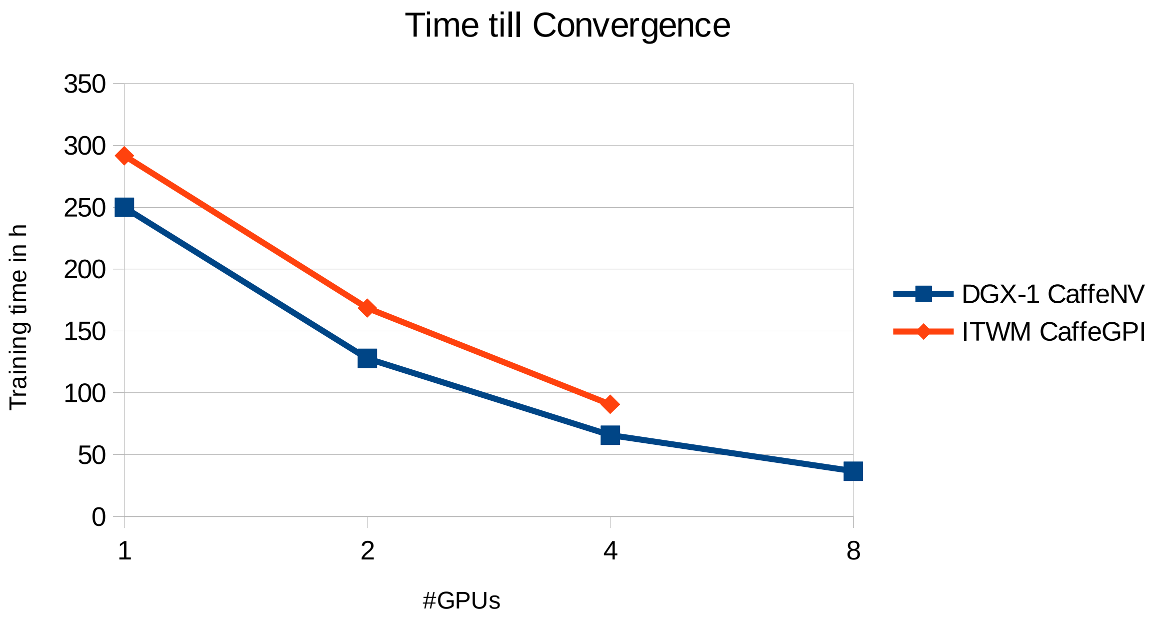


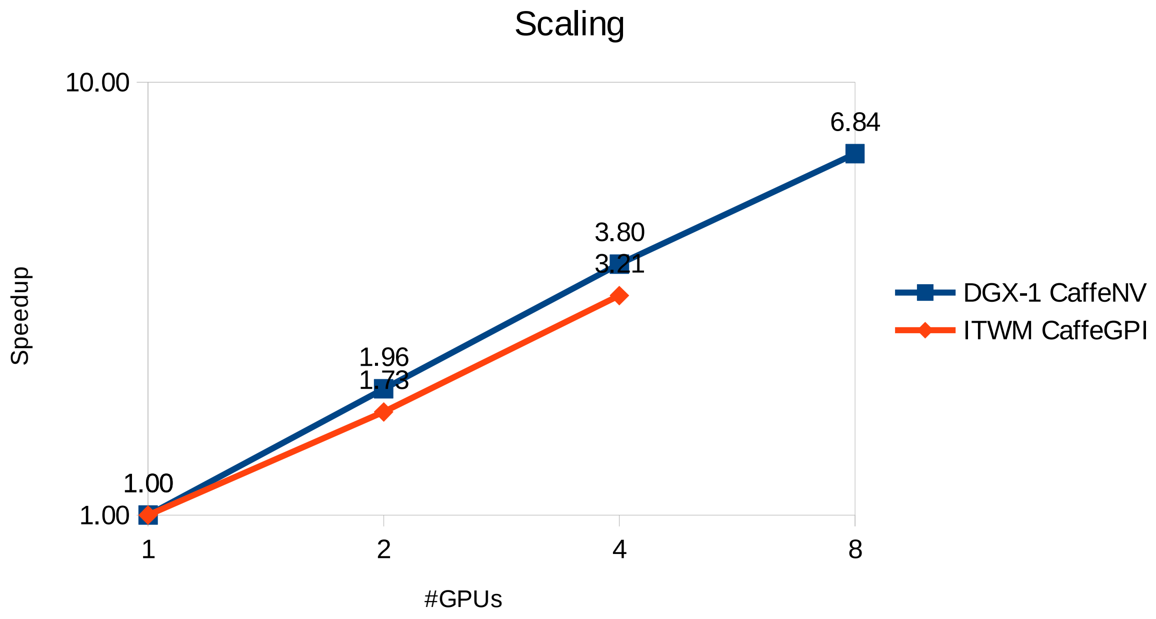
**Specs**:

- 32 GB GPU Mem
- 64 GB PGAS Mem
- 2TB BeeGFS for Train Data
- GPU interconnect: PCIe

# Low Cost Deep Learning Setup

## Prototype 2-Node System Benchmarks





Specs: Topology: GoogLeNet, Cuda 8, cuDNN 5.1, CaffeNV 16.4, Batch Size/Node: 64

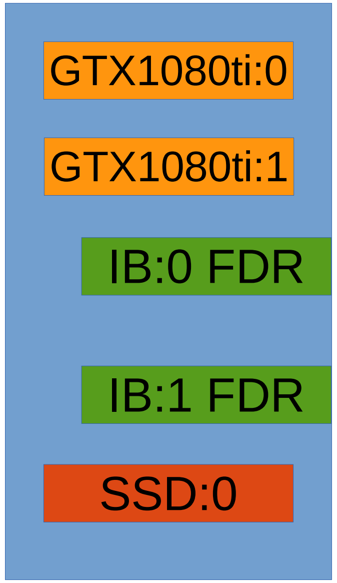
# Low Cost Deep Learning Setup

## Currently building: Prototype 16-Node System

Bundesministerium
für Bildung
und Forschung

**for BMBF founded Fraunhofer Consortium @ITWM**

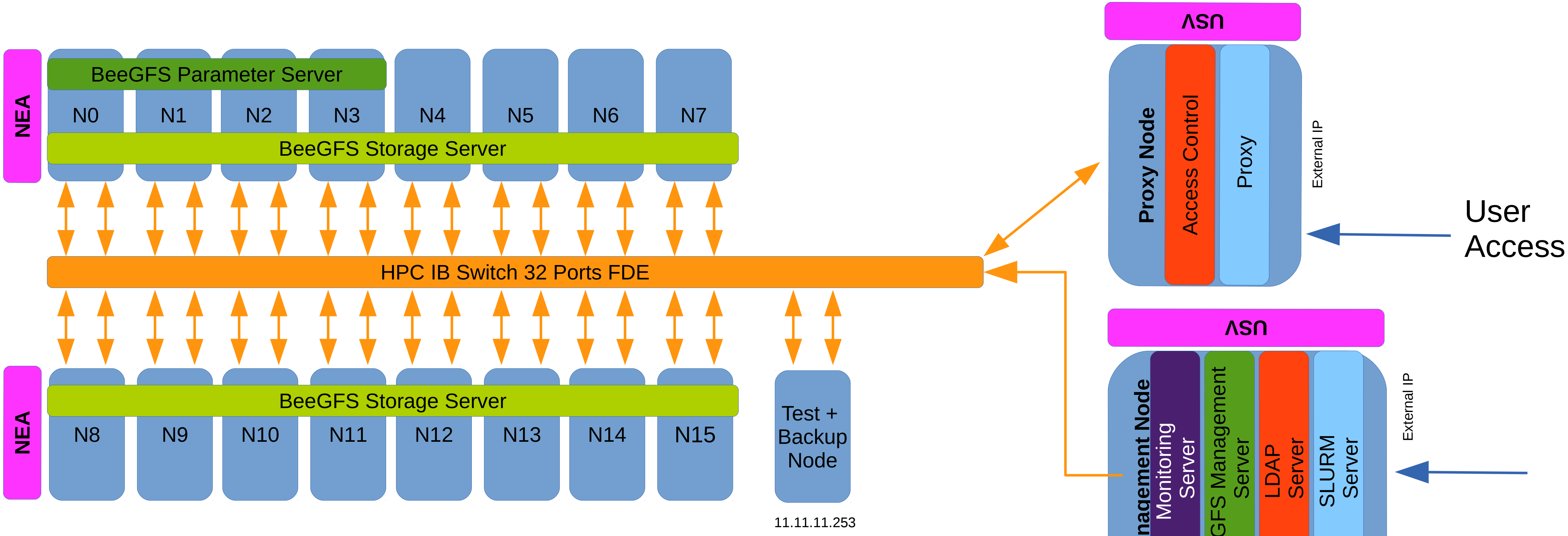
- **Low cost Hardware**
  - Consumer GPUs (GTX1080ti)
  - Novel AMD architecture provides enough PCIe-lanes
  - Hosting Cost per GPU ~ 1.25 k EUR. Compared to DGX-1 ~ 10k
- **Fast HPC Interconnect and Data I/O**
  - Infiniband Network
  - Parallel HPC file system with local SSD
- **Multi-GPU performance**
  - Scalable multi-GPU training



Node configuration

# Low Cost Deep Learning Setup

## Currently building: Prototype 16-Node System



# Project Carme

## An open source software stack for multi-user GPU clusters

***Carme*** (/ˈkɑːrmiː/ KAR-mee; Greek: Κάρμη) is a **Jupiter** moon, also giving the name for a **Cluster** of Jupiter moons (the carme group).

Or in our case:

an open source frame work to mange resources for  multiple users running **Jupyter** notebooks on a **Cluster** of compute nodes.

# Project Carme

## An open source software stack for multi-user GPU clusters

### Common problems in GPU-Cluster operation:

- **Interactive, secure multi user environment**
  - ML and Data Science users want interactive GUI access to compute resources

- **Resource Management**
  - How to assign (GPU) resources to competing users?
    - User management
    - Accounting
    - Job scheduling
    - Resource reservation

- **Data I/O**
  - Get user data to compute nodes (I/O Bottleneck)

- **Maintenance**
  - Meet (fast changing and diverse) software demands of users

# Project Carme

## An open source software stack for multi-user GPU clusters

### *Carme* core idea:

- **Combine established open source ML and DS tools with HPC back-ends**
  - Use containers
    - (for now) Docker
  - Use Jupyter Notebooks as main web based GUI-Frontend
    - All web front-end (OS independent, no installation on user side needed)
  - Use HPC job management and scheduler
    - SLURM
  - Use HPC data I/O technology
    - ITWM’s BeeGFS
  - Use HPC maintenance and monitoring tools

# Project Carme

## An open source software stack for multi-user GPU clusters

***Carme* features:**

- **Open source**
  - Carme uses only opensource components that allow commercial usage
  - Carme is open source, allowing commercial usage
- **User Management**
  - User quotas (GPU time, priority, GPUs per job, jobs per time, Disk quota)
  - Different User Roles (Quotas, right to add containers)
- **Container Management**
  - Container store (user selects from predefined containers)
  - Adding of user defined containers

# Project Carme

## An open source software stack for multi-user GPU clusters

***Carme*** **features:**

- **Scheduler**
  - Resource reservation (calender)
  - Job queues for large jobs and instant interactive access for small jobs
- **Data Management and I/O**
  - Redundant, global file system (BeeGFS), mounts into container
  - Temporary job FS on local SSDs for max performance (BeeOND)
- **Web-Interface**
  - HTTPS and SSH (if allowed) access via proxy
  - Web front-end (management and IDE)

# Project Carme

## An open source software stack for multi-user GPU clusters

***Carme* features:**

- **Scalable Framework**
  - Single GPU to distributed multi-GPU scaling
  - Add GPUs to running job
  - Strong and weak scaling of DL training
  - Works alongside existing *Slurm* systems on existing (HPC) clusters
- **Cluster Maintenance + Monitoring**
  - auto Worker updates
  - Easy hardware scale up (adding more compute hardware later)

# Project Carme

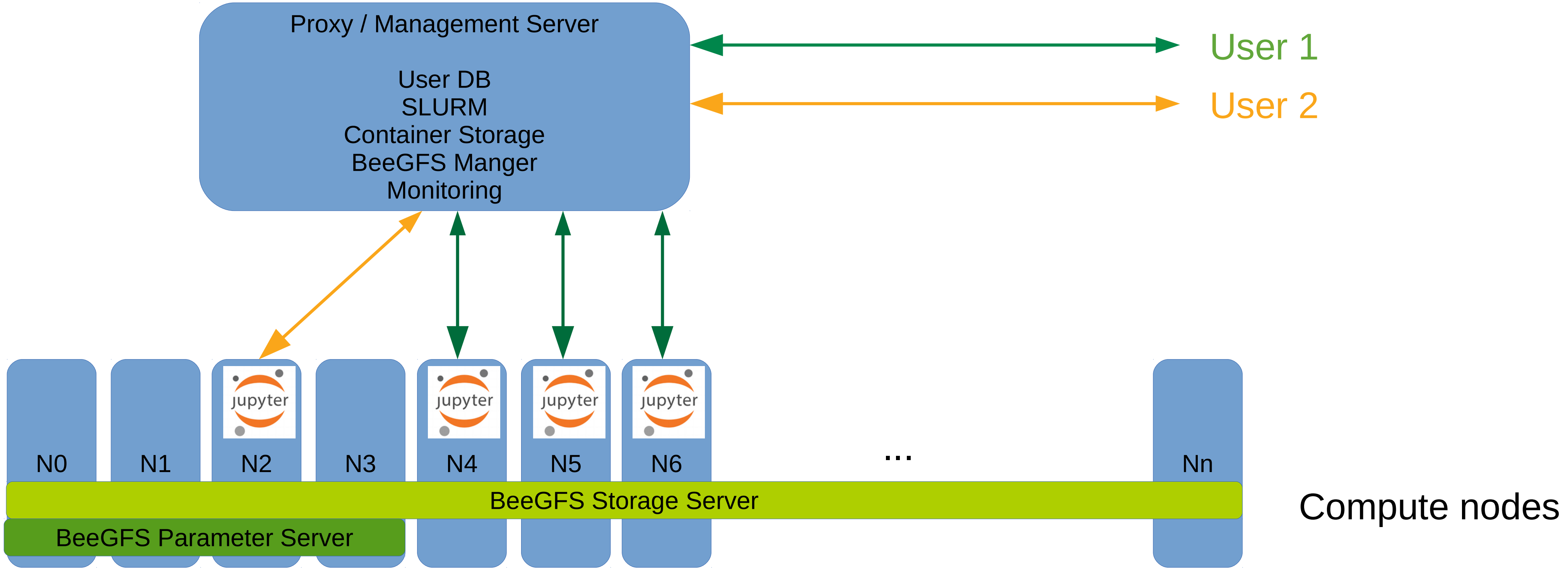
## An open source software stack for multi-user GPU clusters

***Carme*** **features:**

- **Web-Interface**
  - HTTPS and SSH (if allowed) access via proxy
  - Web front-end (management and IDE)
  - Jupyter Notebooks + plugins
  - Tensorboard Server  (other web application possible)

# Project Carme

## An open source software stack for multi-user GPU clusters



# Project Carme

## An open source software stack for multi-user GPU clusters

**Carme road map:**

Running beta Version by 04/18

Public 0.9 beta release by 06/18 (ISC High Performance Conference)

→ AWS live demo

Version 1.0 with deployment tools by 12/18 (NIPS)

Contact: janis.keuper@itwm.fhg.de

Fraunhofer