



JSC SUPERCOMPUTING SYSTEMS

JULAIN LAB VISIT JSC SIMLAB NEUROSCIENCE

9 May 2019 | Andreas Herten | Forschungszentrum Jülich

Overview

JSC

Systems

Running on Supercomputers

System Concepts

Compute Time

DL on Supercomputers

Further Resources

Jülich Supercomputing Centre

- Part of Institute for Advanced Simulation (IAS)
- Research next-gen supercomputers
- Support applications leveraging machines (*SimLabs*)
- Operate supercomputers and connected infrastructure
 - JURECA
 - JUWELS
 - DEEP, DEEP-ER, DEEP-EST
 - JURON
 - Former: JUQUEEN, JUROPA, JUGENE, JUDGE

Overview

JSC

Systems

Running on Supercomputers

System Concepts

Compute Time

DL on Supercomputers

Further Resources

Jülich Supercomputing Centre

- Part of Institute for Advanced Simulation (IAS)
- Research next-gen supercomputers
- Support applications leveraging machines (*SimLabs*)
- Operate supercomputers and connected infrastructure
 - **JURECA**
 - **JUWELS**
 - DEEP, DEEP-ER, DEEP-EST
 - **JURON**
 - Former: JUQUEEN, JUROPA, JUGENE, JUDGE



JURECA – Jülich's Multi-Purpose Supercomputer

- 1872 nodes with Intel Xeon E5 CPUs (2×12 cores)
- 75 nodes with 2 NVIDIA Tesla K80 cards (look like 4 GPUs)
- **JURECA Booster:** 1640 nodes with Intel Xeon Phi *Knights Landing*
- 1.8 (CPU) + 0.44 (GPU) + 5 (KNL) PFLOP/s peak performance (Top500: #44)
- Mellanox EDR InfiniBand



JUWELS – Jülich's New Scalable System

- 2500 nodes with Intel Xeon CPUs (2×24 cores)
- 46 + 10 nodes with 4 NVIDIA Tesla V100 cards
- 10.4 (CPU) + 1.6 (GPU) PFLOP/s peak performance (Top500: #26)

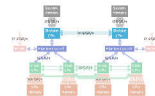


→ *Appendix*

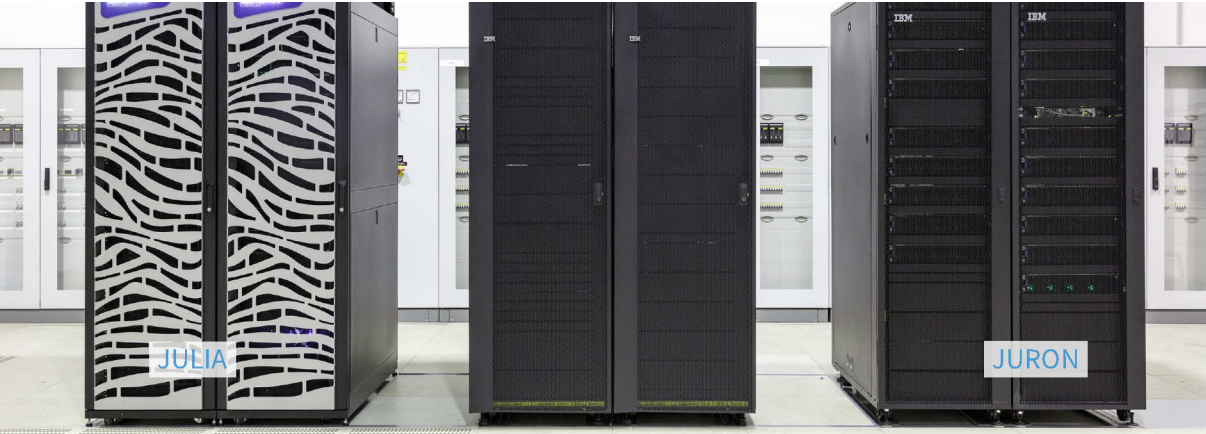


JUWELS – Jülich's New Scalable System

- 2500 nodes with Intel Xeon CPUs (2×24 cores)
- 46 + 10 nodes with 4 NVIDIA Tesla V100 cards
- 10.4 (CPU) + 1.6 (GPU) PFLOP/s peak performance (Top500: #26)



→ *Appendix*

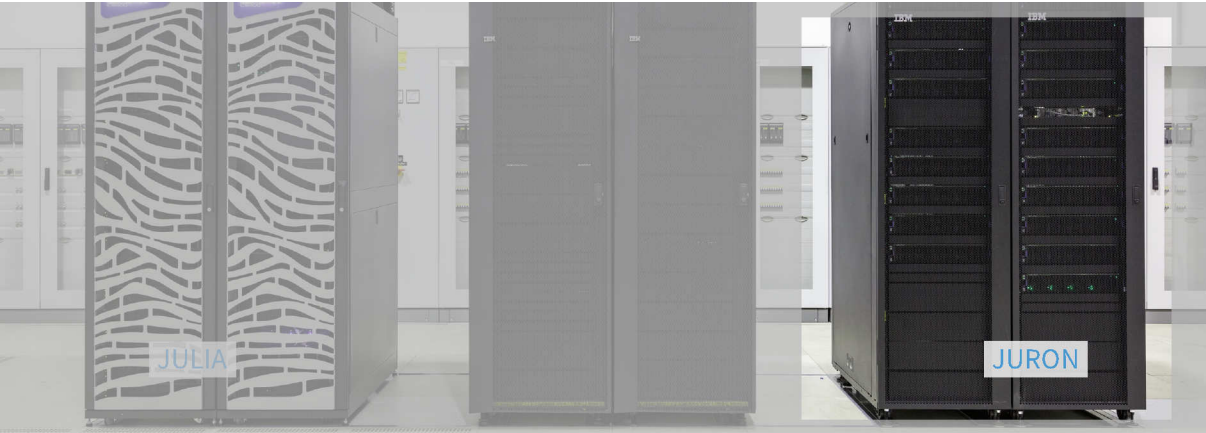


JURON – A Human Brain Project *Pilot System*

- 18 nodes with IBM POWER8NVL CPUs (2×10 cores)
- Per Node: 4 NVIDIA Tesla P100 cards, connected via NVLink
- GPU: 0.38 PFLOP/s peak performance



→ *Appendix*



JURON – A Human Brain Project *Pilot System*

- 18 nodes with IBM POWER8NVL CPUs (2×10 cores)
- Per Node: 4 NVIDIA Tesla P100 cards, connected via NVLink
- GPU: 0.38 PFLOP/s peak performance



→ *Appendix*

Running on Supercomputers

Accounts, Login, Projects

- Account and project management via JuDoor

<https://dsperv.zam.kfa-juelich.de/judoor/> 

The screenshot shows a web browser window with the address bar displaying 'dpserv.zam.kfa-juelich.de/judoor/account/a/JSC_LDAP/herten2/'. The page title is 'JuDoor Your account'. The user is logged in as 'herten2' and can click 'Logout'. The main heading is 'Andreas Herten'. Below this, account details are listed: 'Account' (herten2), 'E-mail address' (a.herten@gmail.com), 'Telephone' (-1825), and 'Address' (JSC, Wilhelm-Johnen-Strasse, 52425 Jülich, Germany). The 'Systems' section shows 'judac' with a link to 'Manage SSH-keys', a note about the usage agreement confirmed on 21.03.2019, and a 'TRAINING1909' tag. The 'Projects' section shows 'GPU Programming with CUDA Trainingscourse' with a 'TRAINING1909' tag and a 'Join a project' button.

Andreas Herten

Account herten2

E-mail address a.herten@gmail.com

Telephone -1825

Address JSC
Wilhelm-Johnen-Strasse
52425 Jülich
Germany

Systems

judac [Manage SSH-keys](#)
Usage agreement confirmed on 21.03.2019
TRAINING1909

Projects

GPU Programming with CUDA Trainingscourse TRAINING1909

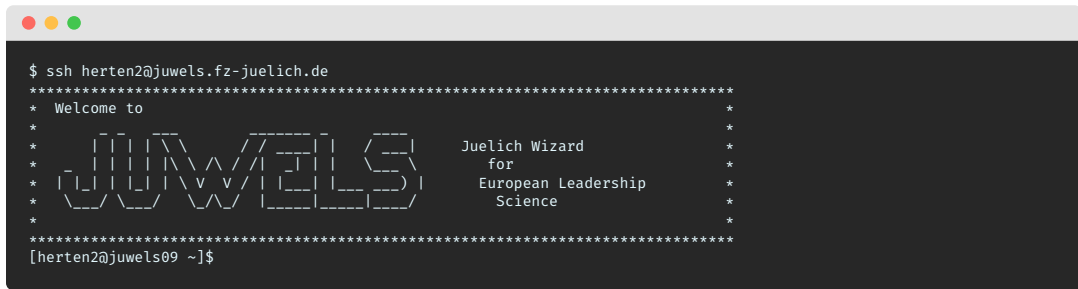
[Join a project](#)

Accounts, Login, Projects

- Account and project management via JuDoor
<https://dsperv.zam.kfa-juelich.de/judoor/> 
- Login via SSH: `ssh herten2@juwels.fz-juelich.de`  (or *Jupyter**)

Accounts, Login, Projects

- Account and project management via JuDoor
<https://dsperv.zam.kfa-juelich.de/judoor/> 
- Login via SSH: `ssh herten2@juwels.fz-juelich.de`  (or *Jupyter**)



```
$ ssh herten2@juwels.fz-juelich.de
*****
* Welcome to *
* *
* Juelich Wizard *
* for *
* European Leadership *
* Science *
* *
*****
[herten2@juwels09 ~]$
```

Accounts, Login, Projects

- Account and project management via JuDoor
<https://dsperv.zam.kfa-juelich.de/judoor/> 
- Login via SSH: `ssh herten2@juwels.fz-juelich.de`  (or *Jupyter**)
- After login: Activate project with `jutil` 

Accounts, Login, Projects

- Account and project management via JuDoor
<https://dsperv.zam.kfa-juelich.de/judoor/> 
- Login via SSH: `ssh herten2@juwels.fz-juelich.de`  (or Jupyter*)
- After login: Activate project with `jutil` 



```
$ jutil env activate -p myproj -A myproj
```

`jutil` options:

- p Set general environment variables (`$PROJECT`, ...)
- A Book subsequent computations to correct account

Login with Jupyter

- Run Jupyter Notebooks on JSC supercomputers from browser 
- JSC Jupyter Portal: jupyter-jsc.fz-juelich.de
- Login with JuDoor credentials (*JSC Web Account*)

- Run
- JSC
- Logi

Welcome to Jupyter@JSC

[Sign in with JSC Authentication Service](#)

Requirements	
JSC Webservice account	

Maintenance	
Jupyter@JSC	None
JUWELS	None Jupyter@JSC related
JURECA	None Jupyter@JSC related
JURON	None Jupyter@JSC related

Useful Links

Jupyter		
Home	Newsletter	Introduction Video
Blog	Documentation	O'Reilly on Jupyter
Twitter	Jupyter-Notebooks	new Install your own kernel

Juelich Supercomputing Centre

Login

- Run
- JSC
- Logi

The screenshot shows a JupyterLab web interface in a browser. The address bar shows the URL: `jupyter-jsc.fz-juelich.de/user/uid%3Da.herten@fz-juelich.de/lab?redirects=1`. The interface has a menu bar (File, Edit, View, Run, Kernel, Hub, Tabs, Settings, Help) and a left sidebar with a file explorer. The file explorer shows a list of files and folders: `Projects` (a day ago), `shared` (5 months ago), `Untitled.i...` (seconds ago), `testjob.sh` (2 months ago), and `x11-forwa...` (a month ago). The main area displays a code editor for a file named `Untitled.ipynb` using the `Python 3` kernel. The code cell contains the command `!lscpu`, and the output shows detailed system information:

```
[2]: !lscpu

Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:             Little Endian
CPU(s):                 80
On-line CPU(s) list:   0-79
Thread(s) per core:    2
Core(s) per socket:    20
Socket(s):              2
NUMA node(s):          2
Vendor ID:              GenuineIntel
CPU family:             6
Model:                 85
Model name:             Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz
Stepping:               4
CPU MHz:               1000.000
CPU max MHz:           2401.000
CPU min MHz:           1000.000
BogoMIPS:              4800.00
Virtualization:         VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              1024K
L3 cache:              28160K
NUMA node0 CPU(s):     0-19,40-59
NUMA node1 CPU(s):     20-39,60-79
Flags:                  fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36
```

System Environment with Modules

About Lmod

- Software, compilers available in **modules** 
- Encapsulation of different environments to limit interference

System Environment with Modules

About Lmod

- Software, compilers available in **modules** 
- Encapsulation of different environments to limit interference
- Modules setup hierarchically: Compiler $\Rightarrow$ MPI $\Rightarrow$ Other
(also: *Core* modules on Compiler-level)
- Every $1/2$ year: New set of modules in new **stage** (example: Stages/2019a)

System Environment with Modules

About Lmod

- Software, compilers available in **modules** 
- Encapsulation of different environments to limit interference
- Modules setup hierarchically: Compiler $\Rightarrow$ MPI $\Rightarrow$ Other
(also: *Core* modules on Compiler-level)
- Every 1/2 year: New set of modules in new **stage** (example: Stages/2019a)
- Commands

`module avail [NAME]` List available modules

`module spider NAME` Search for modules

`module load NAME` Load module by name

`module list` List loaded modules

`module purge` Remove all loaded modules



System Environment with Modules

Workflow Example

```
$ module avail
```

```
----- Core packages -----  
ARMForge/18.2.3          Rust/1.30.1  
ARMPerformanceReports/18.2.3  SciPy-Stack/2018b-Python-2.7.15  
Advisor/2018_update4  
Python/3.6.6              (D)  
  
----- Compilers -----  
GCC/7.3.0    GCC/8.2.0 (D)    Intel/2019.0.117-GCC-7.3.0    PGI/18.7-GCC-7.3.0
```

Where:

g: built for GPU
D: Default Module

System Environment with Modules

Workflow Example

```
$ module load GCC/7.3.0
```

```
$ module avail
```

```
----- MPI runtimes available for GNU compilers -----  
MVAPICH2/2.3-GDR (g)
```

```
----- Packages compiled with GNU compilers -----  
GSL/2.5
```

System Environment with Modules

Workflow Example

```
$ module load MVAPICH2/2.3-GDR
```

```
$ module avail
```

```
----- Packages compiled with MVAPICH2-GDR and GCC compilers -----  
FFTW/3.3.8          Horovod/0.15.2-GPU-Python-2.7.15 (g)    netCDF-C++4/4.3.0    netCDF/4.6.1  
HDF5/1.8.20 (D)     Horovod/0.15.2-GPU-Python-3.6.6 (g,D)    netCDF-Fortran/4.4.4  
  
----- MPI runtimes available for GNU compilers -----  
MVAPICH2/2.3-GDR (g,L)
```


System Environment with Modules

Workflow Example

```
$ module list
```

```
Currently Loaded Modules:
```

```
1) GCCcore/.7.3.0 (H) 4) jscslurm/.17.11.12 (H,S) 7) GCC/7.3.0 10) MVAPICH2/2.3-GDR (g)
2) binutils/.2.31.1 (H) 5) jsctools/.0.1 (H,S) 8) nvidia/.driver (H)
3) StdEnv (H) 6) .juwels-env (H) 9) CUDA/9.2.88 (g)
```

```
Where:
```

```
S: Module is Sticky, requires --force to unload or purge
g: built for GPU
H: Hidden Module
```

System Environment with Modules

Workflow Example

```
$ module purge
```

```
The following modules were not unloaded:
```

```
(Use "module --force purge" to unload all):
```

```
1) jscslurm/.17.11.12  2) jsctools/.0.1
```

```
$ module list
```

```
Currently Loaded Modules:
```

```
1) jscslurm/.17.11.12 (H,S)  2) jsctools/.0.1 (H,S)
```

```
Where:
```

```
S: Module is Sticky, requires --force to unload or purge
```

```
H: Hidden Module
```

Resource Management

About batch systems

- All supercomputers:
Few **login** (/gateway/front-end) nodes, many **compute** (back-end) nodes
- Rule of thumb:
 - Login nodes for compilation and setup
 - Compute nodes for running computations

*: Special flavor of Slurm, PSSHlurm.

Resource Management

About batch systems

- All supercomputers:
Few **login** (/gateway/front-end) nodes, many **compute** (back-end) nodes
- Rule of thumb:
 - Login nodes for compilation and setup
 - Compute nodes for running computations
- Computations run in **jobs** via resource manager
- Jülich: Slurm* (JURON: LSF) 

*: Special flavor of Slurm, *PSSlurm*.

Resource Management

About batch systems

- All supercomputers:
Few **login** (/gateway/front-end) nodes, many **compute** (back-end) nodes
- Rule of thumb:
 - Login nodes for compilation and setup
 - Compute nodes for running computations
- Computations run in **jobs** via resource manager
- Jülich: Slurm* (JURON: LSF) 
- Two workflow possibilities for job submission
 - Batch** All configuration enclosed in one file (for *production* runs)
 - Interactive** First allocate resources, then quickly run multiple individual jobs (for *development*)

*: Special flavor of Slurm, PSSlurm.

Resource Management

About batch systems

- All supercomputers:
Few **login** (/gateway/front-end) nodes, many **compute** (back-end) nodes
- Rule of thumb:
 - Login nodes for compilation and setup
 - Compute nodes for running computations
- Computations run in **jobs** via resource manager
- Jülich: Slurm* (JURON: LSF) 
- Two workflow possibilities for job submission
 - Batch** All configuration enclosed in one file (for *production runs*)
 - Interactive** First allocate resources, then quickly run multiple individual jobs (for *development*)

*: Special flavor of Slurm, *PSSlurm*.

Resource Management

Batch Example

```
#!/bin/bash
# file: job.sh
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=24
#SBATCH --partition=batch # or: devel, gpus, develgpus, ...
srun hostname
```

Resource Management

Batch Example

```
$ sbatch job.sh
Submitted batch job 1128473

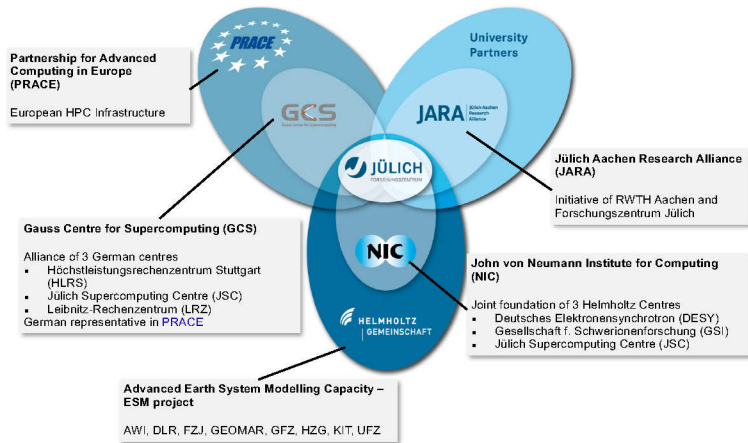
$ squeue -u $USER
          JOBID PARTITION    NAME    USER  ST       TIME  NODES NODELIST(REASON)
          1128473     devel  job.sh  herten1 CF        0:06      2 jwc00n[001-002]

$ head -2 slurm-1128473.out
jwc00n001.adm00.juwels.fzj.de
jwc00n002.adm00.juwels.fzj.de

$ cat slurm-1128473.out | sort | uniq -c
    24 jwc00n001.adm00.juwels.fzj.de
    24 jwc00n002.adm00.juwels.fzj.de
```


Compute Time Proposal

- Access to machine granted within compute time projects (JUWELS, JURECA)
- Apply for compute time through different channels, see [online](#) 
- JURON: No dedicated application needed



DL on Supercomputers

Getting Started with Deep Learning

Tutorial

- Tutorial by Fahad Khalid (and others)
- Showcase well-known Deep-Learning-frameworks on JURECA, JURON (JUWELS)
 - Tensorflow
 - Keras
 - PyTorch
 - Horovod
 - Caffe (outdated)
- Description, environment setup, example (+data)

→ gitlab.version.fz-juelich.de/khalid1/ml_dl_on_supercomputers 

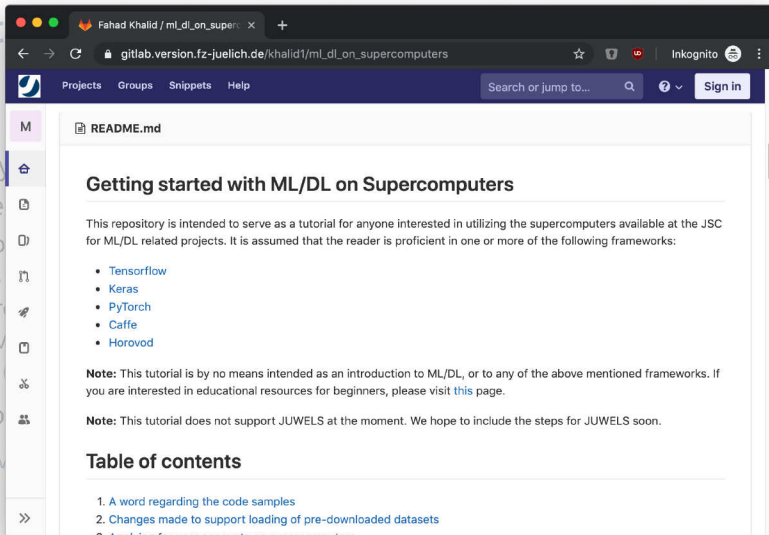
Getting Started Tutorial

- Tutorial by
- Showcase

- TensorFlow
- Keras
- PyTorch
- Horovod
- Caffe

- Description

→ gitlab.v



The screenshot shows a web browser window with the address bar displaying 'gitlab.version.fz-juelich.de/khalid1/ml_dl_on_supercomputers'. The page title is 'Getting started with ML/DL on Supercomputers'. The content includes a paragraph about the repository's purpose, a list of supported frameworks (Tensorflow, Keras, PyTorch, Caffe, Horovod), and two notes regarding the tutorial's scope and support for JUWELS. A 'Table of contents' section is also visible at the bottom.

Getting started with ML/DL on Supercomputers

This repository is intended to serve as a tutorial for anyone interested in utilizing the supercomputers available at the JSC for ML/DL related projects. It is assumed that the reader is proficient in one or more of the following frameworks:

- [Tensorflow](#)
- [Keras](#)
- [PyTorch](#)
- [Caffe](#)
- [Horovod](#)

Note: This tutorial is by no means intended as an introduction to ML/DL, or to any of the above mentioned frameworks. If you are interested in educational resources for beginners, please visit [this](#) page.

Note: This tutorial does not support JUWELS at the moment. We hope to include the steps for JUWELS soon.

Table of contents

1. [A word regarding the code samples](#)
2. [Changes made to support loading of pre-downloaded datasets](#)
3. [Looking for user accounts on supercomputers](#)

Getting Started with Deep Learning on supercomputers

Tutorial

- Tutorial by Fahad Khalid (and others)
- Showcase well-known Deep-Learning frameworks on JURECA, JURON (JUWELS)

- Tensorflow

- Keras

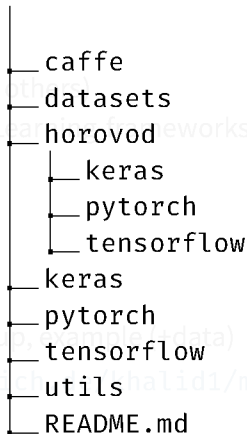
- PyTorch

- Horovod

- Caffe (outdated)

- Description, environment setup, example (data)

→ gitlab.version.fz-juelich.de/fkhalid1/ml_dl_on_supercomputers 



DL Tutorial Walkthrough

DL Tutorial Walkthrough

```
$ ssh herten1@jureca.fz-juelich.de
Last login: Mon May  6 12:57:58 2019 from zam060.zam.kfa-juelich.de
*****
*                               Welcome to JURECA                               *
*                                                                           *
* Information about the system, latest changes, user documentation and FAQs: *
*                               http://www.fz-juelich.de/ias/jsc/jureca         *
*****

# herten1 @ JURECA // jrl07 in ~ [12:58:17]
$

# herten1 @ JURECA // jrl07 in ~ [12:58:17]
$ jutil env activate -p cjsc -A zam
```

DL Tutorial Walkthrough

```
# herten1 @ JURECA // jrl07 in ~/Projects [12:59:20]
$ module load git-lfs/2.6.1

# herten1 @ JURECA // jrl07 in ~/Projects [12:59:28]
$ git lfs install
Git LFS initialized.

# herten1 @ JURECA // jrl07 in ~/Projects [12:59:47]
$ git lfs clone https://gitlab.version.fz-juelich.de/khalid1/ml_dl_on_supercomputers.git
Cloning into 'ml_dl_on_supercomputers'...
remote: Enumerating objects: 169, done.
remote: Counting objects: 100% (169/169), done.
remote: Compressing objects: 100% (90/90), done.
remote: Total 169 (delta 83), reused 146 (delta 71)
Receiving objects: 100% (169/169), 46.41 KiB | 0 bytes/s, done.
Resolving deltas: 100% (83/83), done.
```


DL Tutorial Walkthrough

```
# herten1 @ JURECA // jrl07 in ~/Projects [12:59:51]
$ cd ml_dl_on_supercomputers/keras

# herten1 @ JURECA // jrl07 in ~/Projects/ml_dl_on_supercomputers/keras on git:master o
$ sbatch submit_job_jureca_python3.sh
Submitted batch job 6999791

$ queue -u $USER
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	ODELIST(REASON)
6999791	develgpu	KERAS_MN	herten1	R	0:11	1	jrc0003

DL Tutorial Walkthrough

```
$ tail output_6999791.out
56448/60000 [=====>..] - ETA: 0s - loss: 0.0262 - acc: 0.9919
56960/60000 [=====>..] - ETA: 0s - loss: 0.0262 - acc: 0.9919
57472/60000 [=====>..] - ETA: 0s - loss: 0.0261 - acc: 0.9919
57984/60000 [=====>..] - ETA: 0s - loss: 0.0260 - acc: 0.9919
58496/60000 [=====>..] - ETA: 0s - loss: 0.0259 - acc: 0.9919
59008/60000 [=====>..] - ETA: 0s - loss: 0.0259 - acc: 0.9919
59520/60000 [=====>..] - ETA: 0s - loss: 0.0259 - acc: 0.9919
60000/60000 [=====] - 7s 121us/step - loss: 0.0258 - acc: 0.9919 -
  val_loss: 0.0297 - val_acc: 0.9909
Test loss: 0.02966399986098204
Test accuracy: 0.9909
```

DL Tutorial Walkthrough

```
$ tail output_6999791.out
56448/60000 [=====>..] - ETA: 0s - loss: 0.0262 - acc: 0.9919
56960/60000 [=====>..] - ETA: 0s - loss: 0.0262 - acc: 0.9919
57472/60000 [=====>..] - ETA: 0s - loss: 0.0261 - acc: 0.9919
57984/60000 [=====>..] - ETA: 0s - loss: 0.0260 - acc: 0.9919
58496/60000 [=====>..] - ETA: 0s - loss: 0.0259 - acc: 0.9919
59008/60000 [=====>..] - ETA: 0s - loss: 0.0259 - acc: 0.9919
59520/60000 [=====>..] - ETA: 0s - loss: 0.0259 - acc: 0.9919
60000/60000 [=====] - 7s 121us/step - loss: 0.0258 - acc: 0.9919 -
  val_loss: 0.0297 - val_acc: 0.9909
Test loss: 0.02966399986098204
Test accuracy: 0.9909
```

→ gitlab.version.fz-juelich.de/khalid1/ml_dl_on_supercomputers

Further Resources

Documentation [JUWELS Website](#)

[JURECA Website](#)

[JURON](#) On system: `man juron`, `man juron-lsf`

- Courses
- Introduction to Usage of Supercomputers at Jülich: [20 May](#), [28 Nov](#)
 - Introduction to Deep Learning Models: [21 May](#)
 - Introduction to GPU programming using OpenACC: [28 Oct](#)
 - [Overview](#)

Communication [Issue tracker](#) sc@fz-juelich.de

[High Messages](#) dispatch.fz-juelich.de:8812/HIGHMESSAGES

[Twitter](#) [@fzj_jsc](#)

Further Resources

Documentation JUWELS Website

JURECA Website

JURON On system: `man juron`, `man juron-lsf`

- Courses
- Introduction to Usage of Supercomputers at Jülich: 20 May, 28 Nov
 - Introduction to Deep Learning Models: 21 May
 - Introduction to GPU programming using OpenACC: 28 Oct
 - Overview

Communication Issue tracker `sc@fz-juelich.de`

High Messages `dispatch.fz-juelich.de:8812`

Twitter `@fzj_jsc`

Thank you
for your attention!
`a.herten@fz-juelich.de`

APPENDIX

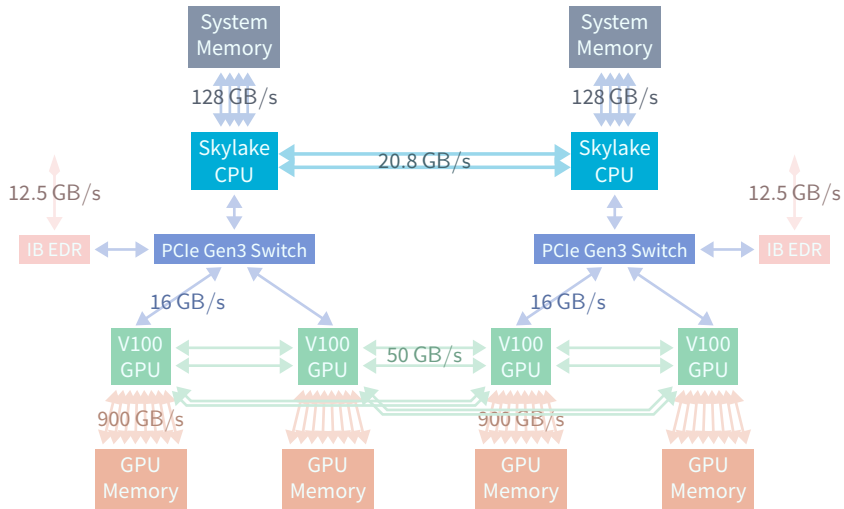
Appendix

JUWELS System Numbers

JURON System Numbers

Glossary

System: JUWELS GPU



System: JUWELS GPU

Intel **Skylake CPU**

- 2 sockets, each 20 cores, each $2 \times$ SMT
- 2.4 GHz to 3.7 GHz;
32 FLOP_{FMA}/Cycle_{2.2 GHz}/Core
- ≈ 175 GB memory (128 GB/s)
- L3, L2, L1 per core: 1.38 MB, 1 MB, 64 kB

0.3 TFLOP/s

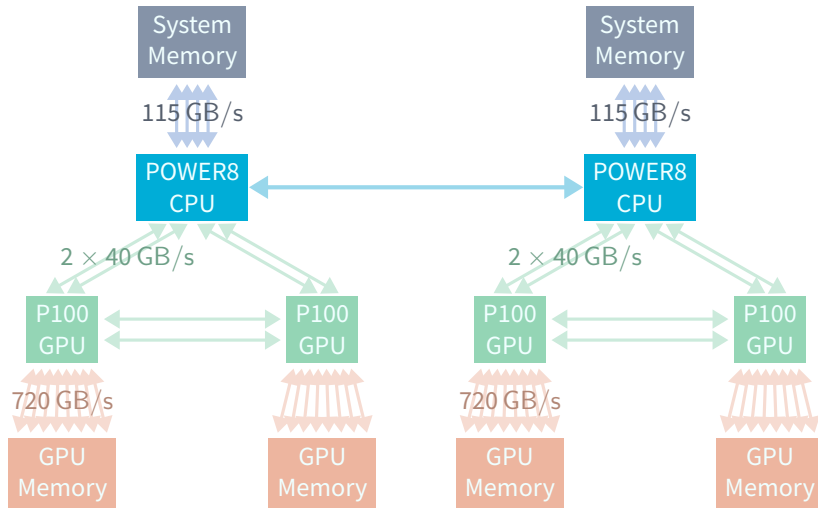
NVIDIA V100 GPU

- 80 SMs
- 64 FLOP/Cycle/SM
- 16 GB memory (900 GB/s)
- L2 \$: 6 MB
- Shared Memory: ≤ 96 kB
- Tensor Cores: 8 / SM ($\Rightarrow 125$ FLOP/s)

31.2 TFLOP/s

NVLink (60 GB/s per dir.)

System: JURON



System: JURON

IBM POWER8 CPU

- 2 sockets, each 10 cores, each 8× SMT
- 2.5 GHz to 5 GHz; 8 FLOP/Cycle/Core
- 256 GB memory (115 GB/s)
- L4 \$ per socket: 4 × 16 MB (Buffer Chip)
- L3, L2, L1 \$ per core: 8 MB, 512 kB, 64 kB

0.5 TFLOP/s

NVLink (2 × 40 GB/s per dir.)

NVIDIA P100 GPU

- 56 SMs
- 64 FLOP/Cycle/SM
- 16 GB memory (720 GB/s)
- L2 \$: 4 MB
- Shared Memory: 64 kB

21.2 TFLOP/s

Glossary I

JSC Jülich Supercomputing Centre, the supercomputing institute of Forschungszentrum Jülich, Germany. [52](#)

JURECA A multi-purpose supercomputer with 1800 nodes at JSC. [2](#), [3](#), [4](#)

JURON One of the two HBP pilot system in Jülich; name derived from Juelich and Neuron. [2](#), [3](#), [7](#), [8](#), [27](#), [28](#), [29](#), [30](#)

JUWELS Jülich's new supercomputer, the successor of JUQUEEN. [2](#), [3](#), [5](#), [6](#)

NVIDIA US technology company creating [GPUs](#). [4](#), [5](#), [6](#), [7](#), [8](#), [49](#), [52](#), [53](#)

NVLink [NVIDIA](#)'s communication protocol connecting [CPU](#) ↔ [GPU](#) and [GPU](#) ↔ [GPU](#) with high bandwidth. [7](#), [8](#), [49](#), [51](#), [52](#), [53](#)

P100 A large [GPU](#) with the [Pascal](#) architecture from [NVIDIA](#). It employs [NVLink](#) as its interconnect and has fast *HBM2* memory. [7](#), [8](#), [51](#)

Glossary II

Pascal GPU architecture from **NVIDIA** (announced 2016). 52

POWER CPU architecture from IBM, earlier: PowerPC. See also POWER8. 51, 53

POWER8 Version 8 of IBM's **POWER** processor, available also under the OpenPOWER Foundation. 7, 8, 53

Tesla The GPU product line for general purpose computing computing of **NVIDIA**. 4, 5, 6, 7, 8

V100 A large GPU with the **Volta** architecture from **NVIDIA**. It employs **NVLink** 2 as its interconnect and has fast *HBM2* memory. Additionally, it features *Tensorcores* for Deep Learning and Independent Thread Scheduling. 49

Volta GPU architecture from **NVIDIA** (announced 2017). 53

Glossary III

CPU Central Processing Unit. 4, 5, 6, 7, 8, 49, 51, 52, 53

GPU Graphics Processing Unit. 4, 5, 6, 7, 8, 49, 51, 52, 53

HBP Human Brain Project. 52

SM Streaming Multiprocessor. 49, 51

SMT Simultaneous Multithreading. 49, 51