PERFORMANCE COMPARISON FOR NEUROSCIENCE APPLICATION BENCHMARKS

IWOPH19 PAPER PRESENTATION

20 June 2019 | **Andreas Herten**, Thorsten Hater, Wouter Klijn, Dirk Pleiter | Forschungszentrum Jülich

Outline

ICEI

ICEI

IPA: ˈaɪsi

- Human Brain Project (HBP): European endeavor to advance understanding of human brain
  - Funded by European Commission; *Flagship* project
  - Multi-faceted: neuroscience, computing, brain-related medicine, …

- Interactive Computing E-Infrastructure (ICEI): Infrastructure for HBP+
  - Services located at large EU supercomputing centers (BSC, JSC, CEA, CINECA, CSCS)
  - Federated to form **Fenix Infrastructure**
  - Computing services (interactive, scalable, VMs)
  - Data services (active, archive, data movers)
  - Other services (authentication, monitoring, …)
  - Customer: HBP, and others through PRACE

FENIX

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

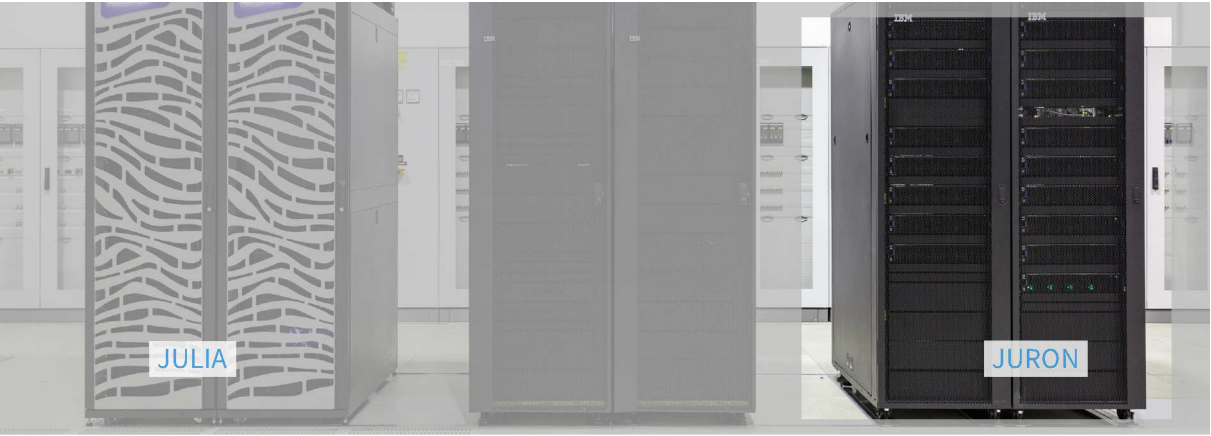
ICEI Benchmark Suite

- Collection of real-world applications from HBP
- Study and compare performance of applications

Characterization	▪ General running, scaling behaviour ▪ Impact of software dependencies ▪ Behaviour on different computer architectures
As Metric	▪ Use applications as metrics for procurements

- First gathered (and used) by JSC,
  by now used by further centers (with individual adaptions and focus)

Infrastructure



JURON – A Human Brain Project *Pilot System*

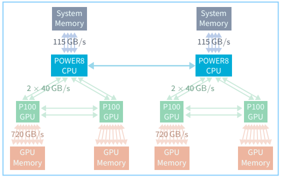
- 18 nodes with IBM POWER8*NVL* CPUs (2 × 10 cores) (*Minsky*)
- Per Node: 4 NVIDIA Tesla P100 cards (16 GB HBM2 memory), connected via NVLink
- GPU: 0.38 PFLOP/s peak performance

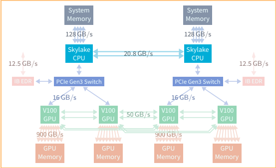
JUWELS – Jülich's New Scalable System

- 2500 nodes with Intel Xeon CPUs (2 $\times$ 24 cores) (*Skylake*)
- 46 + 10 nodes with 4 NVIDIA Tesla V100 cards
- 10.4 (CPU) + 1.6 (GPU) PFLOP/s peak performance (Top500: #30)

System Comparison in Numbers

	JURON	JUWELS
Type of CPU	POWER8	Xeon Platinum 8168 / Xeon Gold 6148 (GPU-acc.)
Number of CPUs	2	2
Number of cores	20	48 / 40
Number of hardware threads	160	96 / 80
SIMD width / bit	128	512
Throughput / Flop/Cycle	160	1536 / 1280
Memory capacity / GiB	256	$\leq$96
Memory bandwidth / GB/s	230	255
LLC capacity / MiB	160	66 / 27.5
Type of GPU	P100 SXM2	V100 SXM2
Number of GPUs	4	– / 4
Throughput / Flop/Cycle	14 336	20 480
Memory capacity / GiB	64	64
Memory bandwidth / GB/s	2880	3600





JÜLICH Forschungszentrum
JÜLICH SUPERCOMPUTING CENTRE

Software Infrastructure

- Job scheduling

  JURON LSF (*IBM Platform LSF*)

  JUWELS Slurm (*ParaStation Slurm*)

- Module system to encapsulate modules (*LMod*)

  Modules chosen from scientist requirements

- Benchmarks include verification tests
- Benchmarking workflow in **JUBE** for repeatability/reproducibility
  - Configuration, option population, compilation
  - Parameter scan; job submission
  - Result collection (incl. verification), display
  - Lightweight sand-boxing; archive of parameters and data

  → `http://www.fz-juelich.de/jsc/jube`

JUBE BENCHMARKING ENVIRONMENT

Benchmarks

Benchmarks Overview

- 5 neuroscientific applications investigated for ICEI
  - NEST
  - Arbor
  - TVB-HPC
  - Elephant ASSET
  - Neuroimaging Deep-Learning
- Also available for other ICEI partners: Neuron, CoreNeuron
- For procurements: augmented by synthetic benchmarks (IOR, ...)
- Applications use different *technologies*; see key on right

C/C++
Python

MPI
OpenMP

CUDA
Numba
TF

Benchmarks

NEST

NEST Description

- Simulator for spiking neural network models
- Focus: Dynamics, size, structure of neural systems (not exact morphology)
- Provides large number of (published) neuron, synapse models
- Large, active user base
- Key design goal: Extreme (weak) scalability; many systems

→ `https://www.nest-simulator.org/`

NEST Benchmark Description

C/C++

MPI

OpenMP

- NEST version 2.14.0 [4]
- Dependencies

  JURON GCC 5.4.0, OpenMPI 3.1.3, GSL 2.4

  JUWELS Intel 2018.2.199/**GCC 5.5.0**, ParaStationMPI 5.2.1-1, GSL 2.4
- Based on benchmark for DEEP-EST
  - Fixed problem size; MPI tasks and OpenMP threads varied
  - NEST internal working unit: Virtual Processes (VP) ($= N_{\text{Nodes}} \times N_{\text{Tasks/Node}} \times N_{\text{Threads/Task}}$)
  - Algorithm steps: 1 Create neurons, connections 2 Simulate spikes
- Parameters
  - Network: 112 500 randomly connected neurons
  - Connections: each neuron connected to $\approx$ 10 % of other neurons
  - Simulate: 1000 ms

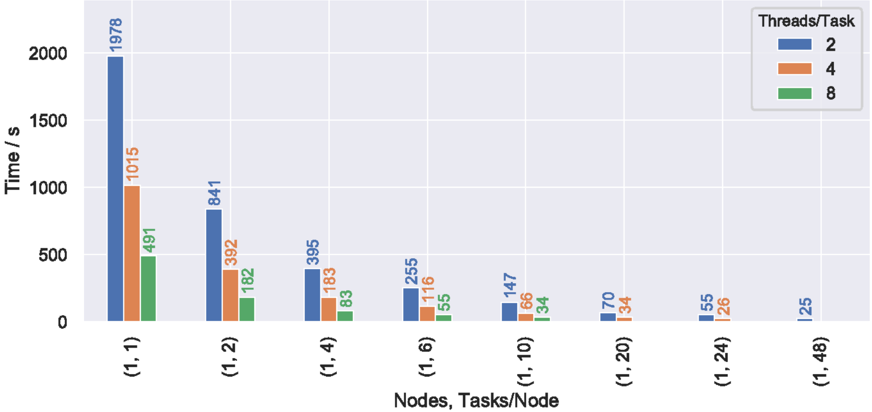
Performance Scaling Analysis

JUWELS

- Metric: Simulation Time
- Different tasks per node
- Different threads per task

Performance Scaling Analysis

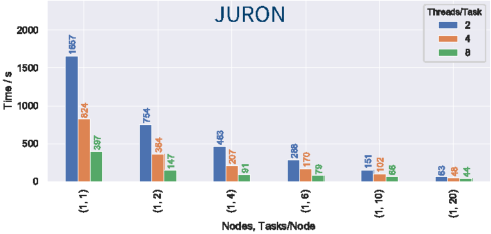
JUWELS

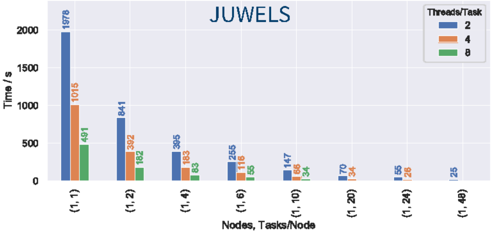


- Metric: Simulation Time
- Different tasks per node
- Different threads per task

Performance Scaling Analysis

JURON/JUWELS Comparison





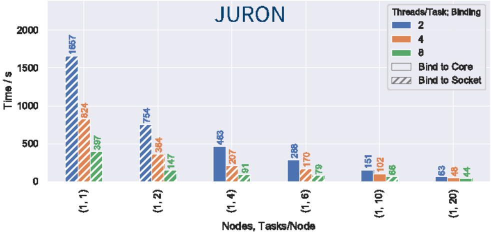
General NEST leverages task- and thread-level parallelism (*SIMD?*)

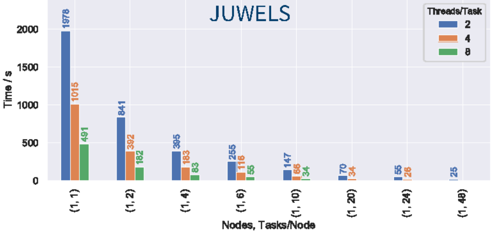
JURON Profit from large number of threads (SMT-8!); high clock rate

JUWELS Utilize higher core count well for larger VP

Performance Scaling Analysis

JURON/JUWELS Comparison





General NEST leverages task- and thread-level parallelism (*SIMD?*)

JURON Profit from large number of threads (SMT-8!); high clock rate

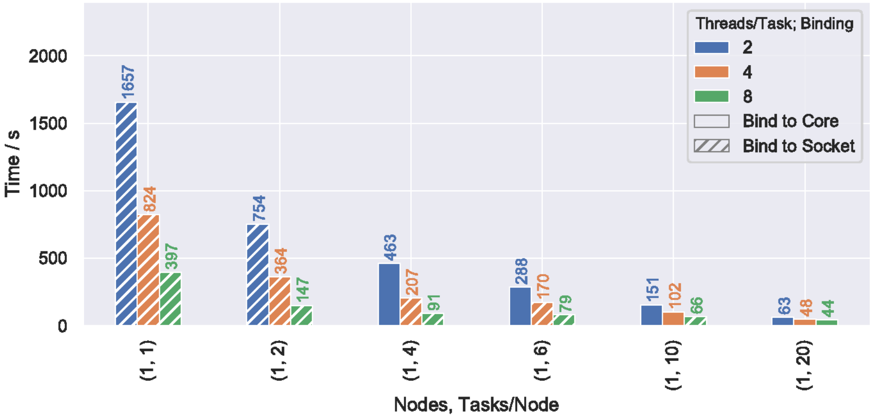
JUWELS Utilize higher core count well for larger VP

⇒ Pinning investigation on JURON

JUWELS: `hwloc` masks for binding

JURON: Task and Thread Distribution

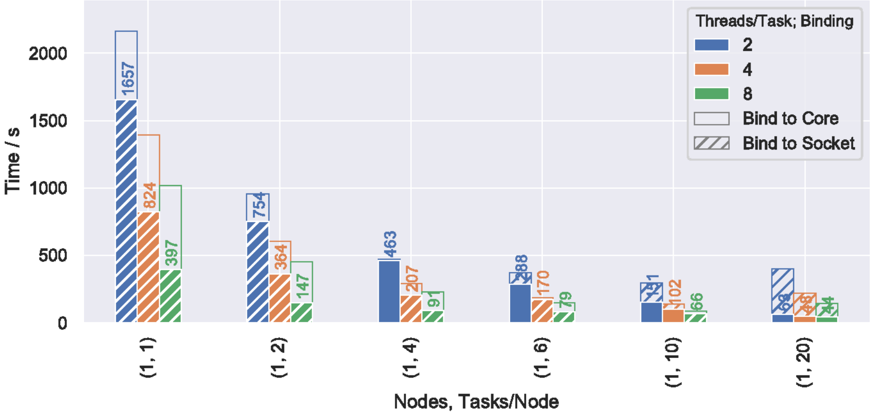
Mapping, Binding, Pinning



- Different tasks, threads → different optimal pinning
- OpenMP: `OMP_PLACES=cores`, `OMP_PROC_BIND=true`
- MPI (OpenMPI):
  - `--map-by socket`
  - `--bind-to core`, `--bind-to socket`

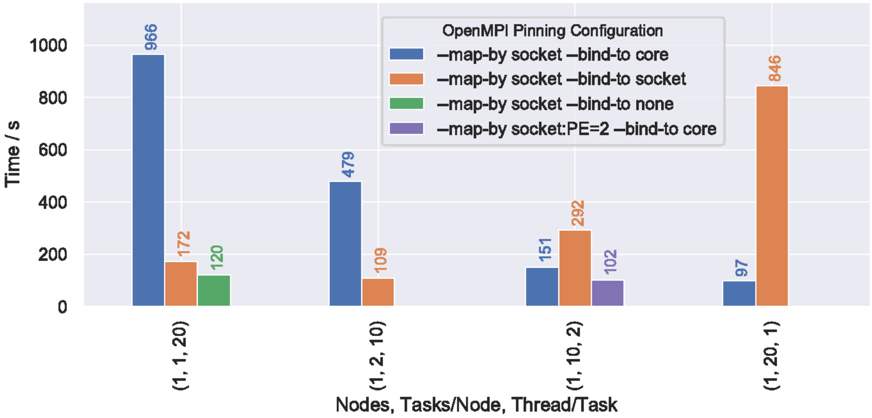
JURON: Task and Thread Distribution

Mapping, Binding, Pinning



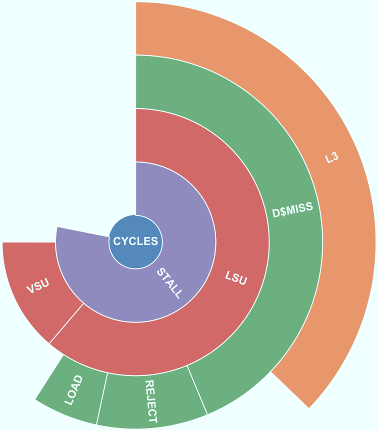
- Different tasks, threads → different optimal pinning
- OpenMP: `OMP_PLACES=cores`, `OMP_PROC_BIND=true`
- MPI (OpenMPI):
  - `--map-by socket`
  - `--bind-to core`,
  - `--bind-to socket`
- MPI difference: 30 % to 600 %

JURON: OpenMPI Pinning Evaluation



- Distribute 20 VPs to various tasks/threads configurations
- Few tasks (*left*): Bind to socket → Task has full socket to place processes
- Few threads (*right*): Bind to core → Tasks align to physical cores
- (1, 1, 20): *None* – **2** sockets are allowed
- (1, 10, 2): *PE=2, core* – bind exactly 2 cores to each task; best fit

JURON: Performance Counters



- First glimpse at analysis of performance counters
- Many cycles stalled
- … due to Load/Store Unit
- … due to misses in Data Cache
- … mainly, missing the L3 cache

Scaling Beyond One Node

Choosing each best run configuration (Tasks, Threads)

	JURON		JUWELS	
Nodes	Network Build / s	Simulation / s	Network Build / s	Simulation / s
1	2.58	44.50	1.25	25.15
2	2.34	32.39	0.81	15.15
4	1.39	18.80	0.63	5.88

Benchmarks

Arbor

Arbor Description

- Simulator for neural network models
- Focus: Morphologically-detailed neurons (*with inner structure*)
- Key feature: Half-time steps (*overlapped*)
  1. Update state of cell (compute-intensive; SIMD, GPUs, …); cell group sizes suited for architectures
  2. Exchange spikes with network (bandwidth-intensive; memory/interconnect)
- Software development targets accelerators, in portable manner

→ `https://github.com/arbor-sim/arbor`

Arbor Benchmark Description

C/C++

MPI

OpenMP

CUDA

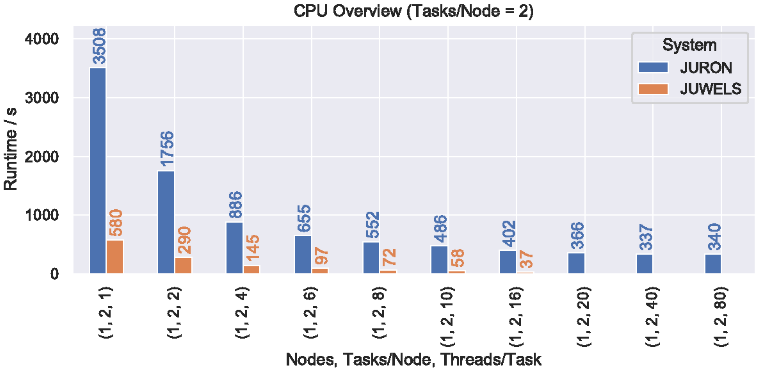
- Arbor version 0.1 [6]
- Dependencies

JURON		GCC 6.3.0, OpenMPI 2.1.2, CUDA 9.2.148
JUWELS	CPU	GCC 8.2.0, ParaStationMPI 5.2.1-1
	GPU	GCC 7.3.0, MVAPICH 2.3-GDR, CUDA 9.2.148

- Vectorization only on x86, not on ppc64le
- Based on Arbor NSuite benchmarks
- Parameters
  - Network: 10 000 cells
  - Simulate: CPU – 100 ms; GPU – 1000 ms

Performance Scaling Analysis

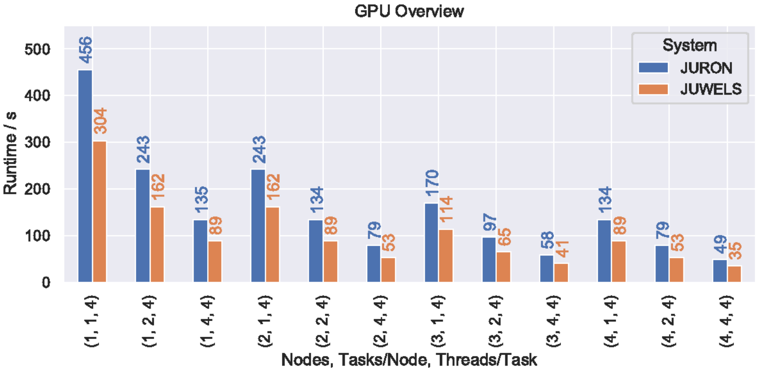
CPU



- Metric: Simulation time
- Biological time: 100 ms
- Different threads per task
- JUWELS: Benefit from large Single Instruction, Multiple Data (SIMD) units
- JURON: Little benefit from larger number of threads, higher clock
- Performance ratio roughly matched floating-point throughput of systems

Performance Scaling Analysis

GPU

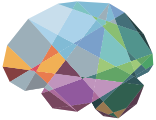


- Biological time: 1000 ms
- Different tasks per node; 1 GPU per task
- System measurement differences match GPU architecture differences: V100 has ≈ 40 % higher floating-point throughput as P100

Benchmarks

TVB-HPC

TVB-HPC Description



- The Virtual Brain (TVB): Full brain network simulation, dynamics, large-scale
  - Mesoscopic models of neural dynamics; whole brain regions
  - Structural connectivity data sets for connecting regions (*connectome*)
  - Output in different experimental forms (EEG, fMRI, …)
    $\rightarrow$ compare simulation to experimental data
- TVB-HPC: HPC-targeted implementation
  - Parallel in parameters
  - Leverage modern software infrastructure:
    - Use Numba for Just-in-Time Compilation (JIT) of Python code
    - Also: Use Numba for GPU offloading (via LLVM)
    - Optional: Use Loopy for code-generation

$\rightarrow$ `https://github.com/the-virtual-brain/tvb-hpc`

TVB-HPC Benchmark Description

Python

MPI

- Dependencies

  JURON GCC 5.4.0, OpenMPI 2.1.2 & mpi4py 3.0.0,
  Python 3.6.1 (Numpy 1.14.2), Numba 0.39.0

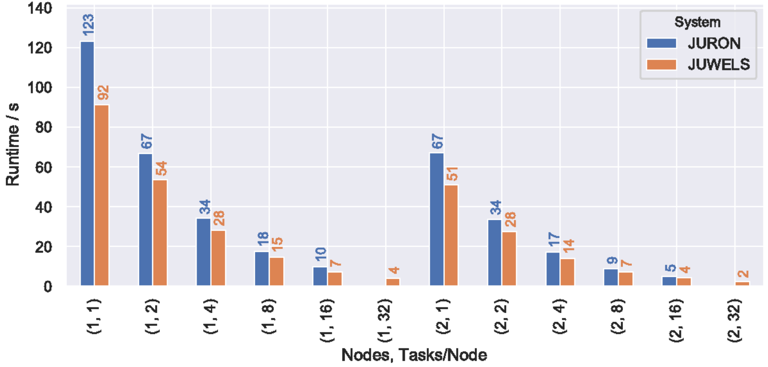
  JUWELS Intel 2019.0.117/GCC 7.3.0, Intel MPI 2019.0.117 & mpi4py 3.0.0,
  Python 3.6.6 (Numpy 1.15.2), Numba 0.40.1

Numba

- Based on custom, internal benchmark
- Parameters
  - Model based on Kuramoto model
  - Simulated time steps: 1600

Performance Scaling Analysis

CPU-Numba



- Metric: Simulation time
- Different nodes
- Different tasks per node
- Parallel efficiency: $\approx$ 80 % (16 MPI ranks)
- JURON always 35 % (1 node) / 13 % (2 nodes) slower than JUWELS
- TVB-HPC / Numba does not exploit SIMD

Benchmarks

Elephant ASSET

Elephant ASSET Description

- Elephant: Analysis of spike train data, time-series data from experiment/simulation
- Elephant ASSET: Automate processing of spike data for sequences of synchronous events
  - Python (distributed with MPI)
  - Some parts of ASSET compiled to C via Cython
  - Most of time: Calculation of survival functions (statistical distributions based on input)

→ `http://www.python-elephant.org`

Elephant ASSET Description

C/C++

Python

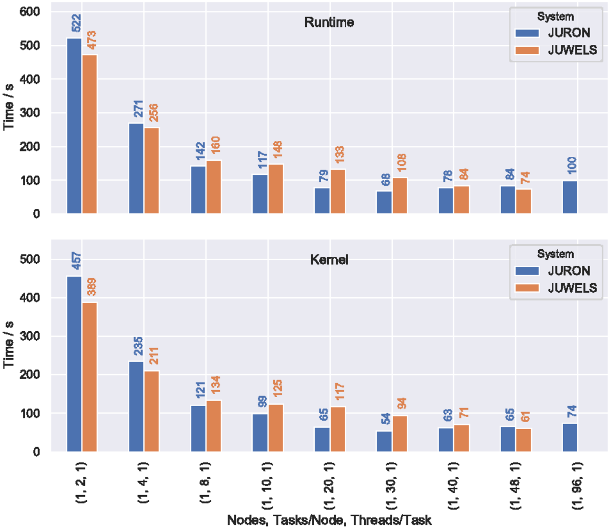
MPI

- Elephant version 0.5.0
- Dependencies

Both	Python libraries: Neo 0.6.1, scikit-learn 0.19.1
JURON	GCC 5.4.0, OpenMPI 2.1.2 & mpi4py 3.0.0, Python 3.6.1 (Numpy 1.14.2, SciPy 1.0.1)
JUWELS	GCC 8.2.0, ParaStationMPI 5.2.1-1 & mpi4py 3.0.0, Python 3.6.6 (Numpy 1.15.2, SciPy 1.1.0)

- Based on internal benchmark
- Parameters
  - Number of spike trains: 100
  - Number of surrogates: 10 000

Performance Scaling Analysis



- Metric: Full benchmark time; main kernel time (survival functions)
- Different tasks per node (MPI)
- Very limited exploitation of available computing resources
- Relatively poor scaling on JUWELS; JURON as fast as JUWELS for sufficient number of MPI tasks

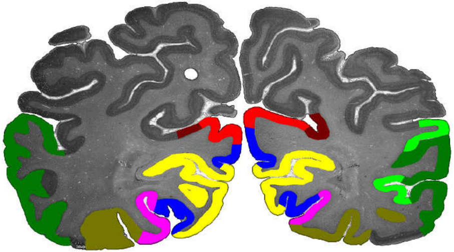
Benchmarks

Neuroimaging Deep Learning

Neuroimaging Deep-Learning Description

- *Neuroimaging Deep-Learning*: Analysis of high-resolution images of histological brain sections
  - Digitally segment into areas
  - Up to now: Manual, expert-driven process
  - This benchmark: Automate with deep learning
- Massive data challenge
  - Brain slice: 125 000 px × 90 000 px (≈ 10 GB per image)
  - Train on 100 slices

→ `http://www.jubrain.fz-juelich.de`



Neuroimaging Deep-Learning Benchmark Description

C/C++

Python

MPI

CUDA

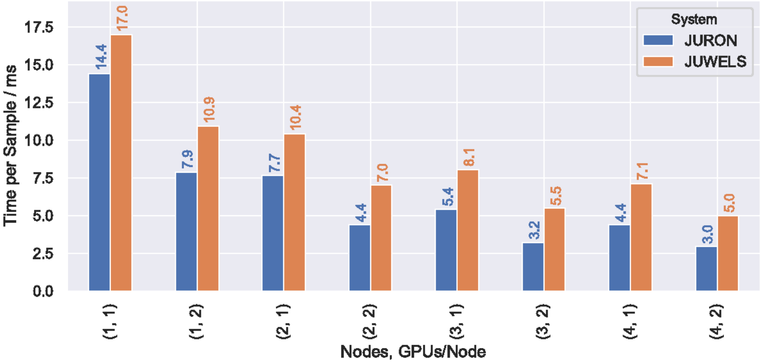
TF

- Benchmark: Mini application version of real code
  - Input data loaded into main memory before benchmark (independence of file storage system (*GPFS*); see [13])
  - First iteration excluded from benchmark (varying build-up times)
- Dependencies

Both	mpi4py 3.0.0, Horovod 0.14.1
JURON	GCC 5.4.0, OpenMPI 2.1.2, Python 3.6.1 (Numpy 1.13.1, SciPy 0.19.1), Tensorflow 1.4.1 & Keras 2.1.3, HDF5 1.8.18 & h5py 2.7.1
JUWELS	GCC 5.5.0, MVAPICH 2.3a-GDR, Python 3.6.5 (Numpy 1.15.4, SciPy 1.0.1) Tensorflow 1.8.0 & Keras 2.1.6, HDF5 1.8.20 & h5py 2.7.1

- Parameters
  - Total batch size: 30 (divided up to GPUs; Horovod)
  - Number of iterations: 40

Performance Scaling Analysis



- Metric: Time per sample (in ms)
- Different GPUs per node
- Scaling quite good, both intra- and inter-node. Losses:

  JURON 10 %, 6 %
  JUWELS 30 %, 23 %
- JURON: Better performance – due to better NVLink CPU-GPU bandwidth?

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

Conclusions

Conclusions

- ICEI Application Benchmark Suite: Domain applications for system research and procurement
- Standardized suite for whole ICEI collaboration (many machines to be studied!)
- Studied systems: JURON (POWER8*NVL*) and JUWELS (Skylake)
- Applications target different software and hardware features

  Arbor Can exploit wide SIMD units very well ⇒ performs poorly on JURON (wish: better floating-point throughput)

  NEST, TVB-HPC, Elephant ASSET Similar performance characteristics across syst

  NEST Pinning very important on POWER8 and with Open

  Neuroimaging Deep-Learning JURON better than JUWELS; NVLi

- Thanks to all scientists providing benchmarks for suite!

Thank you for your attention!

a.herten@fz-juelich.de

APPENDIX

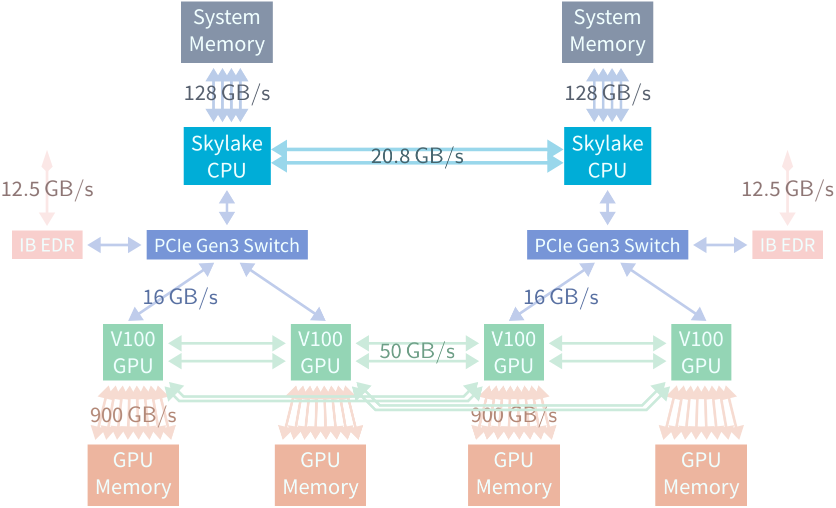
Appendix

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

Appendix

JUWELS System Numbers

System: JUWELS GPU



JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

System: JUWELS GPU

Intel **Skylake CPU**

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

0.3 TFLOP/s

NVLink (60 GB/s per dir.)

NVIDIA **V100 GPU**

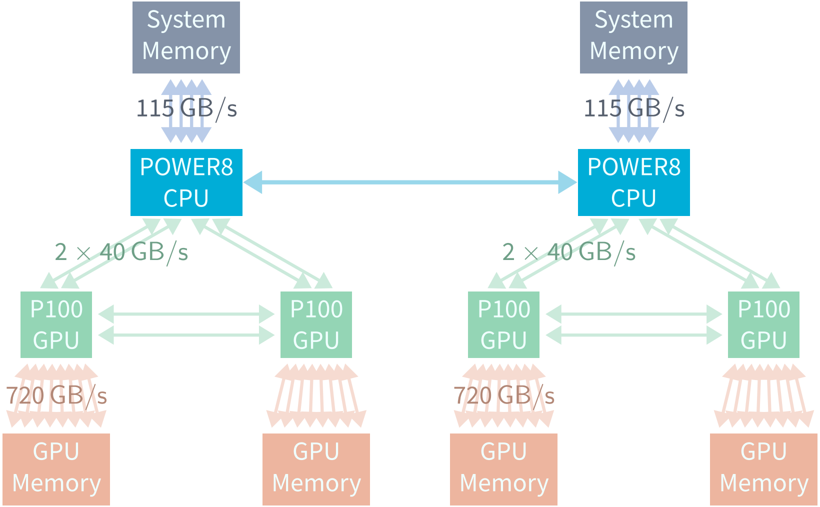
- 80 SMs
- 64 FLOP/Cycle/SM
- 16 GB memory (900 GB/s)
- L2 $: 6 MB
- Shared Memory: ≤ 96 kB
- Tensor Cores: 8 / SM (⇒125 FLOP/s)

31.2 TFLOP/s

Appendix

JURON System Numbers

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**

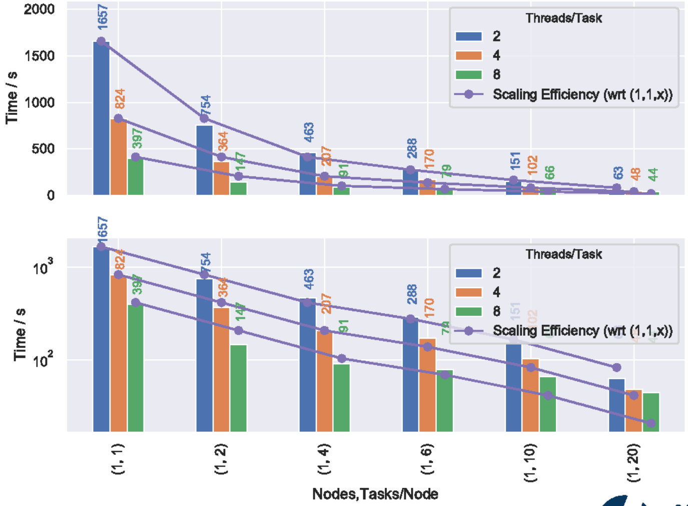
- 56 SMs
- 64 FLOP/Cycle/SM
- 16 GB memory (720 GB/s)
- L2 $: 4 MB
- Shared Memory: 64 kB

21.2 TFLOP/s

Appendix

NEST Supplemental

NEST Scaling Efficiency on JURON



Appendix

References & Glossary

References I

[2] Charl Linssen et al. *NEST 2.16.0*. Aug. 2018. DOI: 10.5281/ZENODO.1400175. URL: https://zenodo.org/record/1400175.

[3] Jakob Jordan et al. "Extremely Scalable Spiking Neuronal Network Simulation Code: From Laptops to Exascale Computers". In: *Frontiers in Neuroinformatics* 12 (2018), p. 2. ISSN: 1662-5196. DOI: 10.3389/fninf.2018.00002.

[4] Alexander Peyser et al. *NEST 2.14.0*. Oct. 2017. DOI: 10.5281/zenodo.882971. URL: https://doi.org/10.5281/zenodo.882971 (page 15).

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

References II

[5] Nora Abi Akar et al. "Arbor - A Morphologically-Detailed Neural Network Simulation Library for Contemporary High-Performance Computing Architectures". In: *27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, PDP 2019, Pavia, Italy, February 13-15, 2019*. IEEE, 2019, pp. 274–282. DOI: 10.1109/EMPDP.2019.8671560. URL: https://doi.org/10.1109/EMPDP.2019.8671560.

[6] Nora Abi Akar et al. *arbor-sim/arbor: Version 0.1: First release*. Oct. 2018. DOI: 10.5281/zenodo.1459679. URL: https://doi.org/10.5281/zenodo.1459679 (page 27).

[7] Viktor K. Jirsa et al. "Towards The Virtual Brain: network modeling of the intact and the damaged brain.". In: *Archives italiennes de biologie* 148(3) (2010), pp. 189–205.

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

References III

[8] Petra Ritter et al. "The Virtual Brain Integrates Computational Modeling and Multimodal Neuroimaging". In: *Brain Connectivity* 3.2 (2013), pp. 121–145. DOI: 10.1089/brain.2012.0120.

[9] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. "Numba: A LLVM-based Python JIT Compiler". In: *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*. LLVM '15. Austin, Texas: ACM, 2015, 7:1–7:6. ISBN: 978-1-4503-4005-2. DOI: 10.1145/2833157.2833162. URL: http://doi.acm.org/10.1145/2833157.2833162.

[10] Yoshiki Kuramoto. "Self-entrainment of a population of coupled non-linear oscillators". In: *International Symposium on Mathematical Problems in Theoretical Physics*. Ed. by Huzihiro Araki. Berlin, Heidelberg: Springer Berlin Heidelberg, 1975, pp. 420–422. ISBN: 978-3-540-37509-8.

References IV

[11] Emiliano Torre et al. “ASSET: Analysis of Sequences of Synchronous Events in Massively Parallel Spike Trains”. In: *PLOS Computational Biology* 12.7 (July 2016), pp. 1–34. DOI: 10.1371/journal.pcbi.1004939.

[12] H. Spitzer et al. “Parcellation of visual cortex on high-resolution histological brain sections using convolutional neural networks”. In: *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*. Apr. 2017, pp. 920–923. DOI: 10.1109/ISBI.2017.7950666.

[13] Lena Oden et al. “IO Challenges for Human Brain Atlasing Using Deep Learning Methods - An In-Depth Analysis”. In: *27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, PDP 2019, Pavia, Italy, February 13-15, 2019*. IEEE, 2019, pp. 291–298. DOI: 10.1109/EMPDP.2019.8671630. URL: https://doi.org/10.1109/EMPDP.2019.8671630 (page 40).

References: Images, Graphics I

[1] Alto Crew. *Iceberg Near Body of Water*. Freely available at Unsplash. URL: `https://unsplash.com/photos/Rv3ecImL4ak`.

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

Glossary I

Arbor Multi-compartment simulation of neural networks. 12, 26, 27, 43

Elephant ASSET Analysis of synchronous events in neural spike trains. 12, 35, 36, 43

Fenix Infrastructure Fenix Resarch Infrastructure to support the Human Brain Project. 4

JSC Jülich Supercomputing Centre, the supercomputing institute of Forschungszentrum Jülich, Germany. 5

JURON One of the two HBP pilot system in Jülich; name derived from Juelich and Neuron. 7, 9, 10, 15, 18, 19, 27, 28, 32, 33, 36, 37, 40, 41, 43

JUWELS Jülich's new supercomputer, the successor of JUQUEEN. 8, 9, 10, 15, 16, 17, 18, 19, 27, 28, 32, 33, 36, 37, 40, 41, 43

Glossary II

LLVM An open Source compiler infrastructure, providing, among others, Clang for C. 31

MPI The Message Passing Interface, a API definition for multi-node computing. 15, 20, 21, 33, 35, 37

NEST Simulator for spiking neural network models with a focus on dynamics, size and structure of neural systems. 2, 12, 13, 14, 15, 18, 19, 43, 53

Neuroimaging Deep-Learning Analysis of high-resolution images of histological brain sections using Deep Learning techniques. 12, 39, 40, 43

NVIDIA US technology company creating GPUs. 7, 8, 48, 61, 62, 63

NVLink NVIDIA's communication protocol connecting CPU $\leftrightarrow$ GPU and GPU $\leftrightarrow$ GPU with high bandwidth. 7, 41, 43, 48, 51, 62, 63

Glossary III

OpenMP Directive-based programming, primarily for multi-threaded machines. 15, 20, 21

P100 A large GPU with the Pascal architecture from NVIDIA. It employs NVLink as its interconnect and has fast *HBM2* memory. 7, 29, 51

Pascal GPU architecture from NVIDIA (announced 2016). 62

POWER CPU architecture from IBM, earlier: PowerPC. See also POWER8. 51, 62

POWER8 Version 8 of IBM's POWERprocessor, available also under the OpenPOWER Foundation. 9, 43, 62

POWER8*NVL* POWER8 processor generation with NVLink connection between Graphics Processing Unit (GPU) and Central Processing Unit (CPU). 7, 43

Tesla The GPU product line for general purpose computing computing of NVIDIA. 7, 8

Glossary IV

TVB-HPC High-Performance Computing sub-project of The Virtual Brain. 12, 31, 32, 33, 43

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. 29, 48

Volta GPU architecture from NVIDIA (announced 2017). 63