HELMHOLTZ

RESEARCH FOR
GRAND CHALLENGES

HEAT

Helmholtz Analytics Toolkit

High-performance computing for the SciPy/Pytorch ecosystem

Dr. Claudia Comito, Jülich Supercomputing Centre

Team Heat

(Helmholtz+GSoC)

- Michael Tarnawa
- Björn Hagemeier
- Kaj Krajsek
- Claudia Comito
- (Ben Bourgart)
- Markus Götz
- Daniel Coquelin
- (Charlie Debus)
- Juan Pedro Gutiérrez
- Philipp Knechtges
- Alex Rüttgers
- Fabian Hoppe
- Pratham Shah
- Sai Suraj
- Neo Sun Han
- Tewodros Mesfin
- Ashwath V A

JÜLICH Forschungszentrum

KIT Karlsruher Institut für Technologie

DLR Deutsches Zentrum für Luft- und Raumfahrt

Framing the problem

Processing/analysing massive datasets

- SciPy: foundation of data processing/analysis pipelines across all of science
- Single-node RAM limitation, no GPU
- For massive datasets:
  - Distribute **tasks** via Dask/Ray
  - Data broken up in self-contained chunks
  - GPU mostly CUDA only

- Inefficiencies:
  1. Porting/optimizing existing code, chunks size
  2. Tasks still single-node RAM limited
  3. Task managers = significant runtime overhead!
  4. (Dask) CUDA ecosystem only
  5. (Ray) few processing ops

Massive data processing with Heat

Helmholtz Analytics Toolkit (FZJ, DLR, KIT)

- Inefficiencies:
  1. Porting/optimizing existing code, chunks size
  2. Tasks still single-node RAM limited
  3. Task managers = significant runtime overhead!
  4. (Dask) CUDA ecosystem only
  5. (Ray) few processing ops

- Heat:
  1. Strictly NumPy / scikit-learn API
  2. Multi-node memory-distributed array operations, total RAM available
  3. Async MPI under the hood
  4. PyTorch-based: CUDA, upcoming ROCm, Apple MPS support
  5. Broad set of functionalities incl. linear algebra, array manipulation, statistics, ML algos

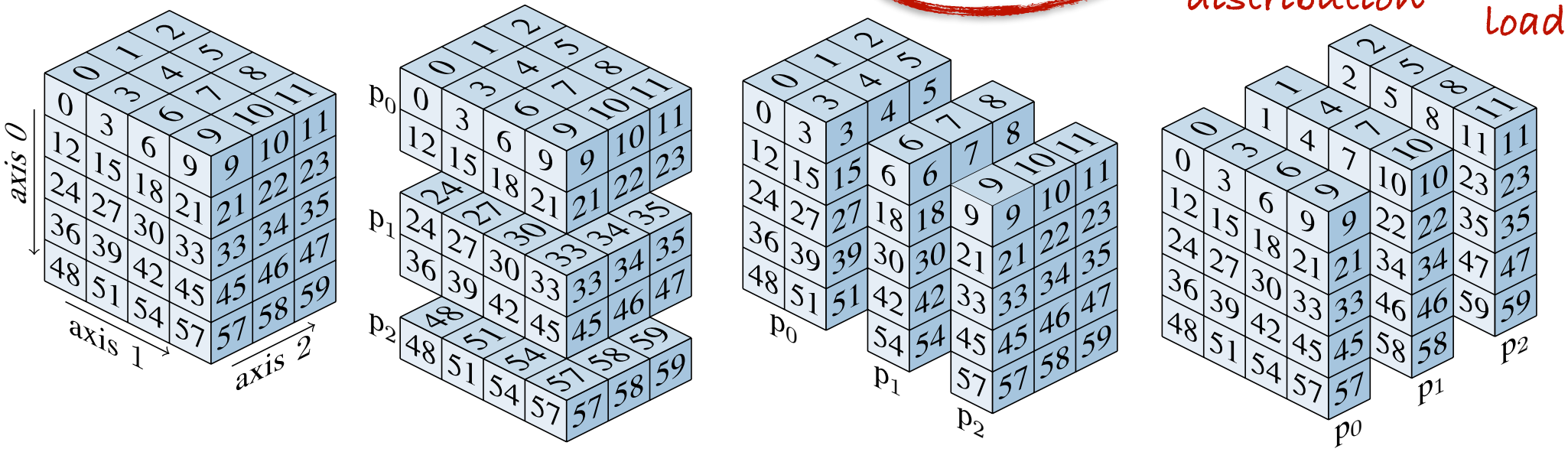
Heat usage as “NumPy drop-in replacement”

NumPy, scikit-learn API plus data distribution

```
import heat as ht
data = ht.arange(60).reshape(5,4,3)
data = ht.array(data, split=0)
```

Enable data distribution

Built-in load balance



(a) `split=None` (b) `split=0` (c) `split=1` (d) `split=2`

[Götz et al., IEEE BigData 2020]

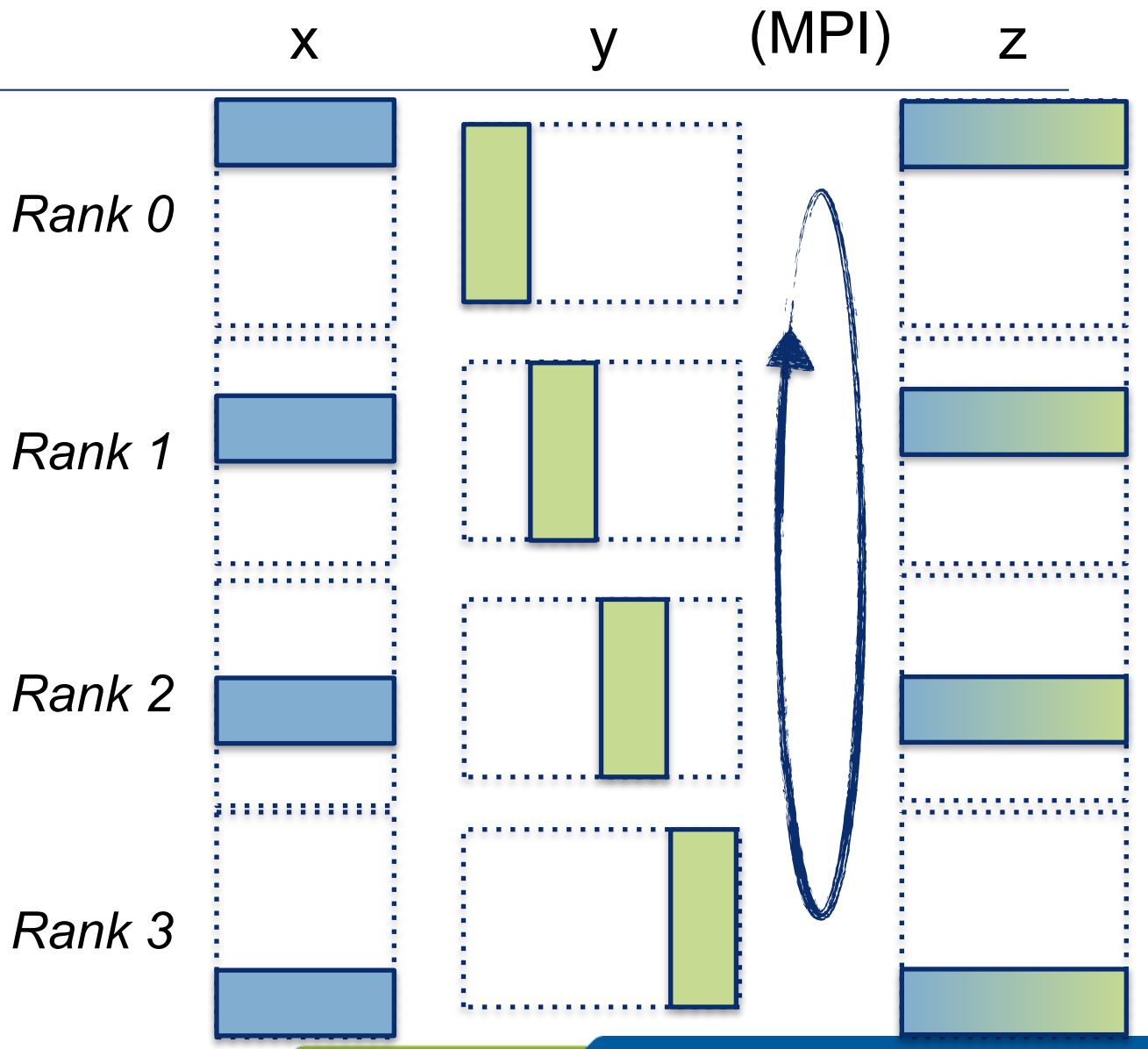
Example: memory-distributed matrix multiplication

```
import heat as ht

# x from file/array/tensor
x = ht.array(x, split=0)

y = ht.array(y, split=1)

z = x @ y
```



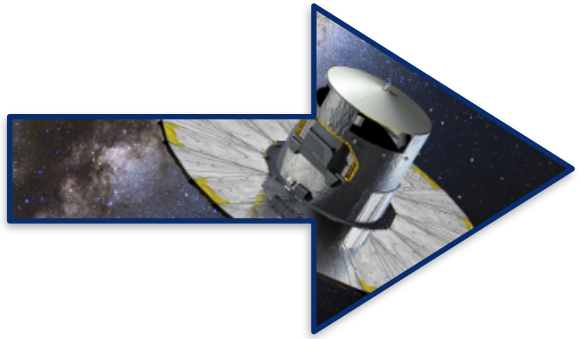
On your laptop:
> *mpirun -n 4 python my_code.py*
On your cluster:
> *srun —ntasks-per-node […] python my_code.py*

Heat Machine Learning

Example: memory-distributed KMeans

```
# scikit-learn API
data = ht.load(my_datafile, split=0)
kmeans = ht.cluster.KMeans(n_clusters, max_iter)
kmeans.fit(data)
```

- Clustering/classification: K-means, k-nn, spectral clustering, LASSO regression…



Ongoing: S. Pfalzner, PhD thesis M. Sowinski, large-scale study of star clusters (GAIA/PUNCH)

Under the hood: PyTorch + mpi4py

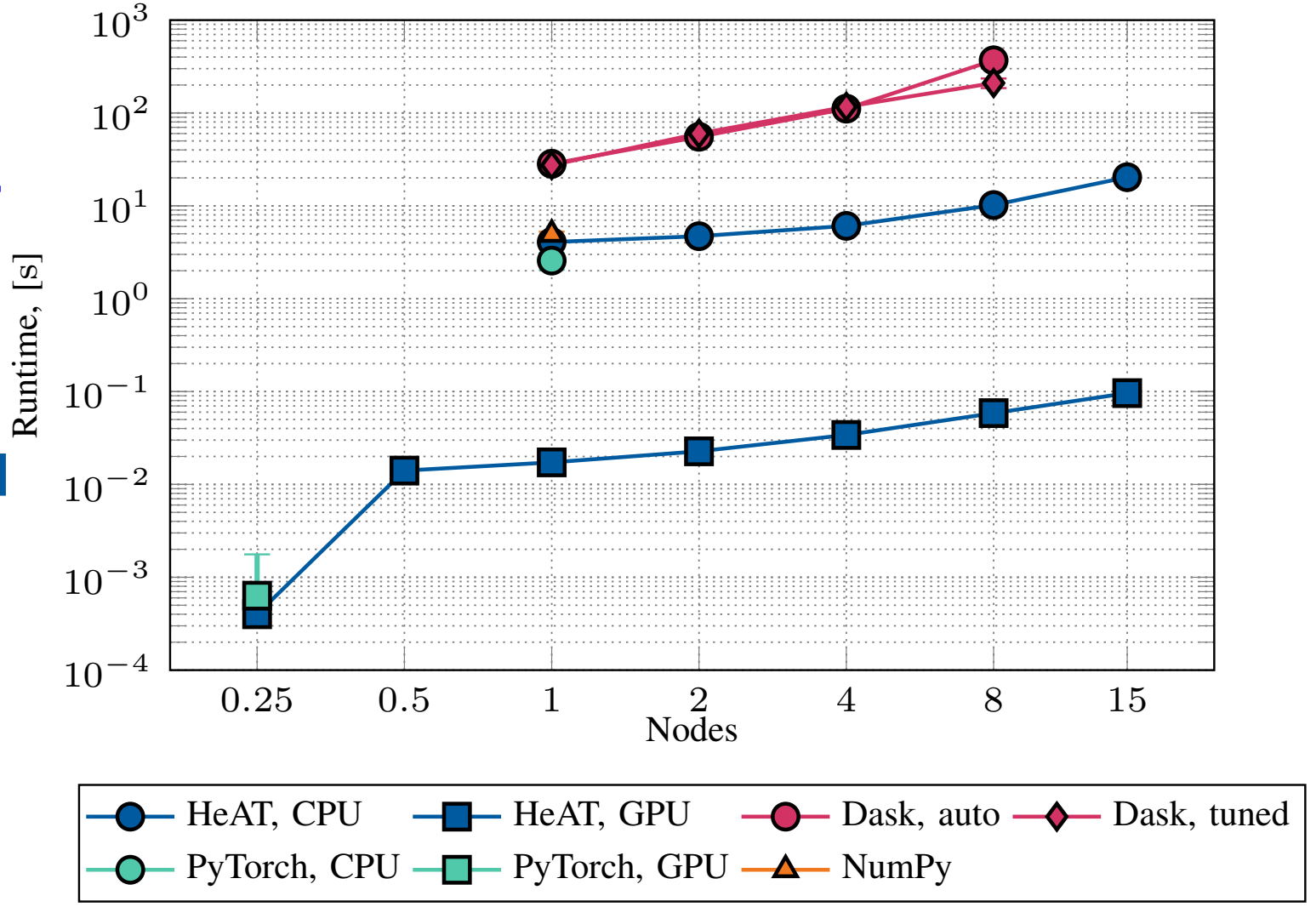
- DNDarray: Memory-distributed PyTorch tensor + MPI_WORLD awareness
- Process-local ops = PyTorch ops
- (Multi-node) communication: MPI via mpi4py
- PyTorch updates eventually incorporated
- Acceleration: CUDA, ROCm, [Apple MPS dev branch]

```
data = ht.array(data, device="gpu")
```

Benchmarks vs. Dask/CuPy: weak scaling

i.e. constant workload on each MPI process

- Pairwise Euclidean distances on SUSY dataset
- 5,000,000 entries, 18 features
- On each node:
  - 2x12-core Intel Xeon Gold 6126
  - 4xTesla V100 GPUs

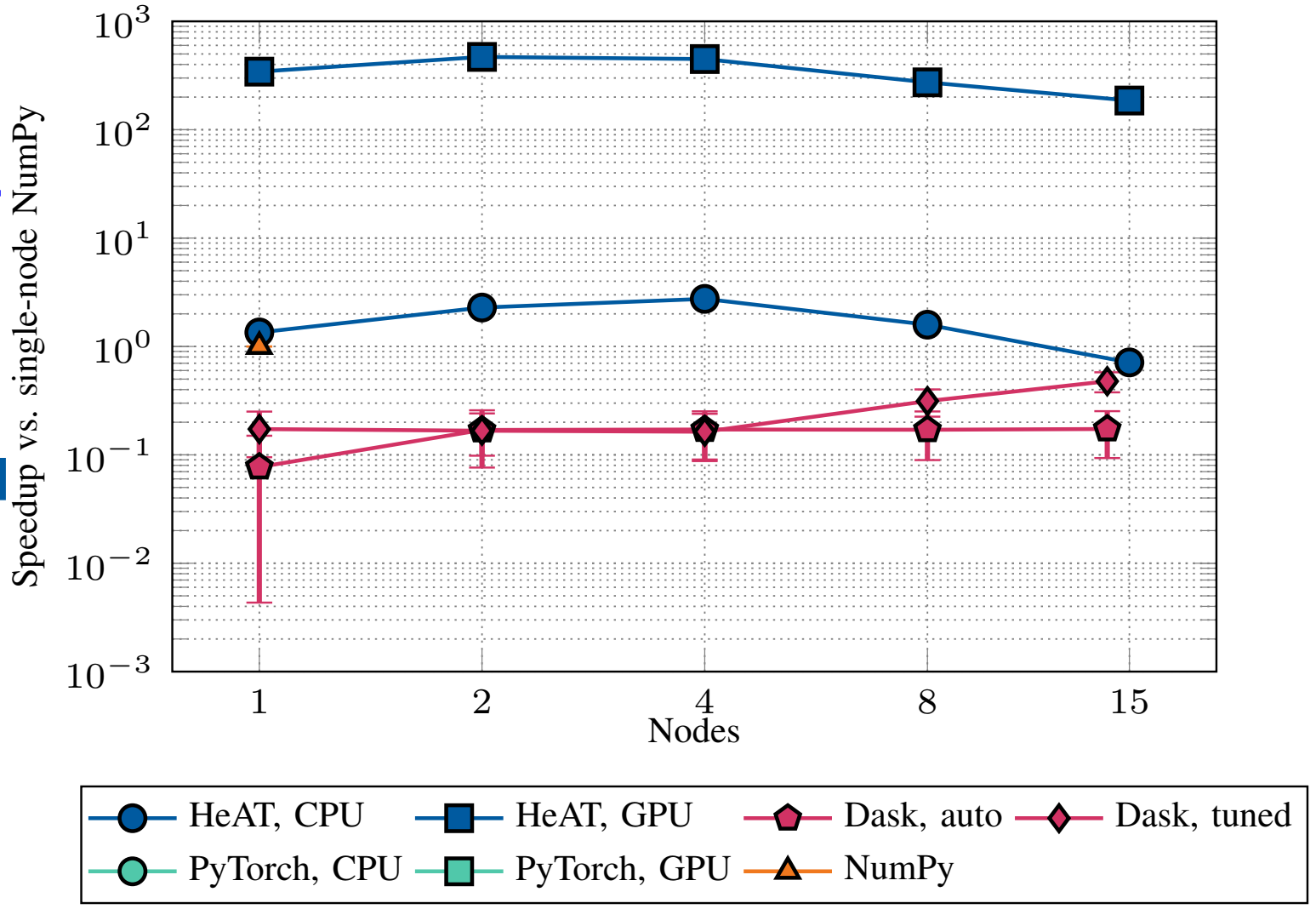


Götz et al., IEEE BigData 2020

Benchmarks vs. Dask/CuPy: strong scaling

i.e. constant workload on system

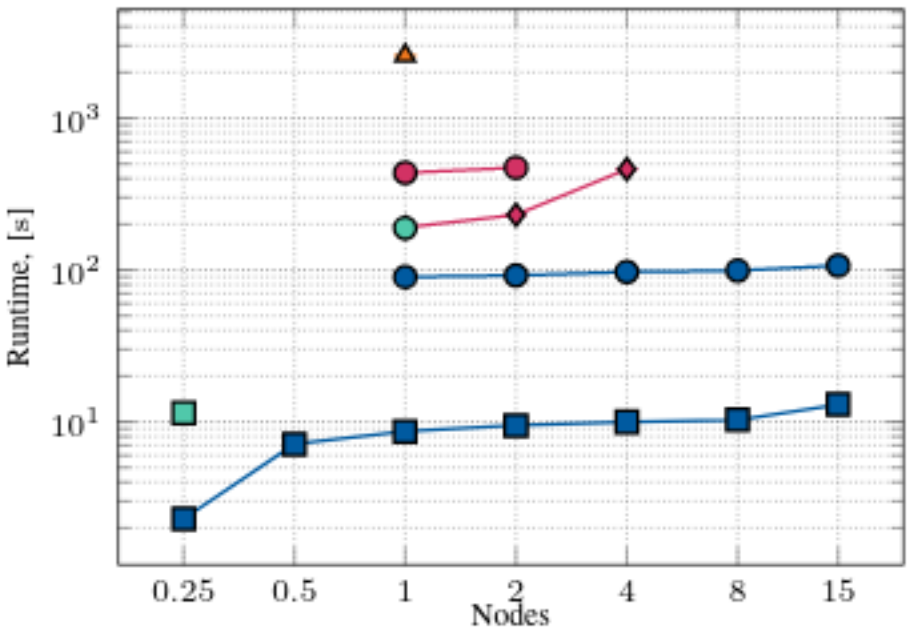
- Pairwise Euclidean distances on SUSY dataset
- 5,000,000 entries, 18 features
- On each node:
  - 2x12-core Intel Xeon Gold 6126
  - 4xTesla V100 GPUs



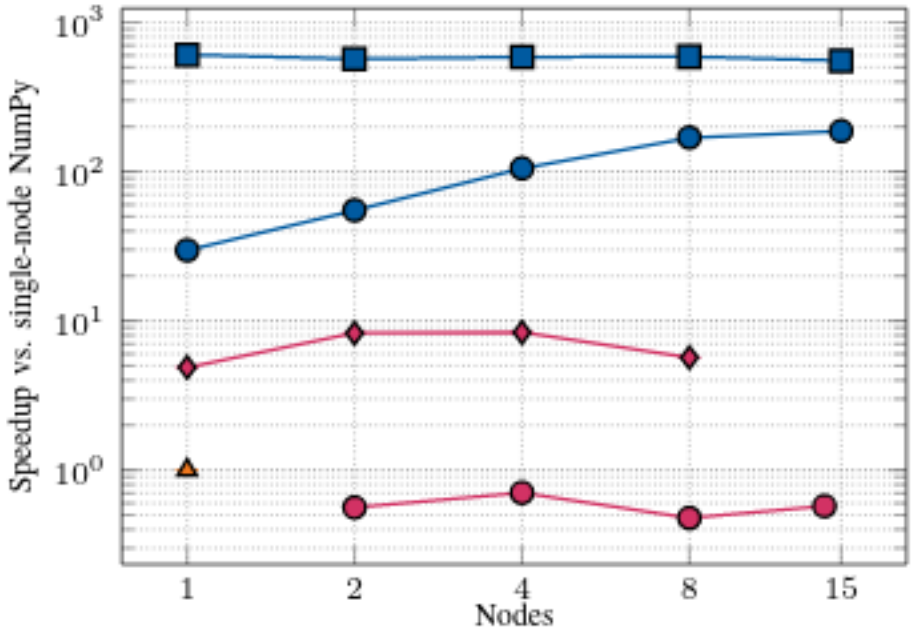
Götz et al., IEEE BigData 2020

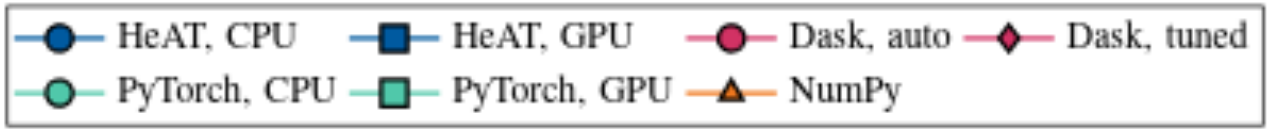
K-means clustering (Götz et al. 2020)

Weak Scaling



Strong Scaling





Ongoing work

- **Heat v1.3 target June 2023**
  - Distributed truncated SVD (engineering use cases ESA/DLR)
  - Signal processing, fully distributed convolutions
  - Improvements in performance & memory footprint
  - AMD GPU support
  - [hacky/tentative] Apple MPS support
  - Continuous benchmarking
- **Target v1.4 EOY2023**
  - Distributed FFT/iFFT
  - Xarray support
  - …

Interoperability & user support

Heat as infrastructure for data-intensive research projects

- Interoperability:
  - Parallel I/O (HDF5, netcdf4, csv) from shared memory
  - Mix&match array-API libraries as needed (incl. data-parallel NN training)
  - Workflows?? Containerised release in the works (docker/ singularity)

- User support:
  - 1-on-1 onboarding, code optimisation if desired
  - Compute resources for testing and prototyping
  - [tentative] research paper writing support

Community development

Co-design = please talk to us!

- Helmholtz long-term commitment
- Software development best practices
- Co-design with use cases
- Open-source developers (Google Summer of Code)

- USER INTERACTION
- User contributions (mutual need!)

Get it touch! *c.comito AT fz-juelich.de*

Distributed array computing in Python