An HPC Python library for Scientific Big Data Analytics

C. Comito, D. Coquelin, B. Hagemeier, K. Krajsek, JSC, FZ Jülich
M. Götz, S. Hanselmann, Karlsruhe Institute for Technology
C. Debus, P. Knechtges, S. Schmitz, M. Siggel, German Aerospace Center

Helmholtz Analytics Framework: motivation

- Data- and computation-intensive research is the new normal
- Scientific Big Data Analytics: different domains, similar techniques (data assimilation, machine learning, deep learning etc.)
- HAF: domain and infrastructure scientists co-design a generalised, standardised data analytics tool for HPC
- 8 use cases, 5 research fields… SO FAR: Earth system modeling, structural biology, aeronautics and aerospace research, medical imaging, and neuroscience

Helmholtz Analytics Toolkit: HeAT

- Started May 2018, early alpha-phase
- Open Source: https://github.com/helmholtz-analytics/heat
- NumPy-like API
- Builds on PyTorch as single-node backend
- Highly optimised algorithms and data structures for HPC via MPI
  - Matrix multiplication, Linear Algebra
- Transparent operations on natively distributed N-dimensional tensors
- CPUs/GPUs
- Coming up: Automatic Differentiation

Machine Learning and Deep Learning

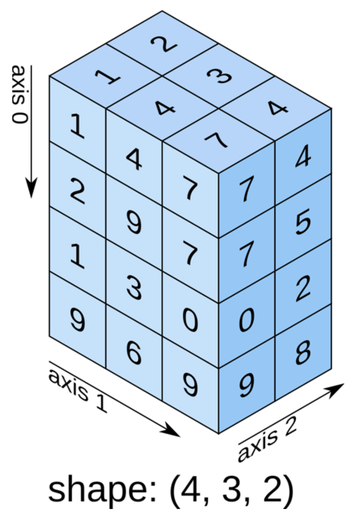
- Currently implemented: K-Means (clustering), Lasso regression
- Implementation priority driven by use cases
- Currently being developed: DBSCAN, Spectral Clustering, PCA, (Gaussian) Naïve Bayes, Neural Networks…
- sklearn-like API

```
# example: apply K-Means to a subset of the
# Open Exoplanets Catalogue
# http://www.openexoplanetcatalogue.com/
import heat as ht

# X is the data, a previously defined NumPy array
.
# Instantiate a distributed heat tensor:
X_heat = ht.array(X, split=0)

# now run K-Means as you would with scikit-learn
# operations are distributed courtesy of HeAT
k = 3
kmeans = ht.ml.cluster.KMeans(n_clusters=k)
model = kmeans.fit(X_heat)
centroids = model.cluster_centers_
result = kmeans.predict(X_heat)
```

Natively distributed ND-tensors

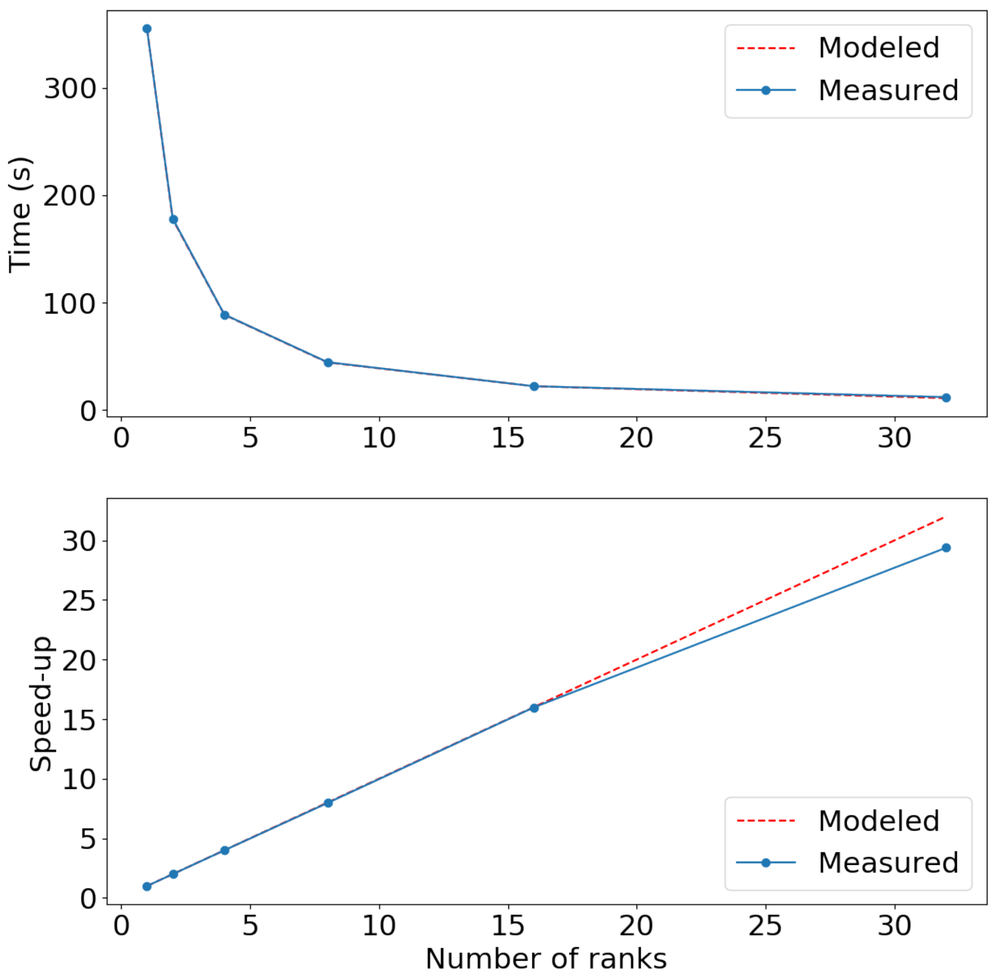
```
> import heat as ht
> a = ht.ones((3, 5), split=0)
> a.shape
(3, 5)
> a.sum(axis=0).shape
(5,)
#alternative on GPU
> a = ht.ones((3,5), split=0, device="gpu")
```

HPC with minimal code adaptation, no worries about size of chunks!



Check us out on GitHub!

HeAT Performance



K-Means clustering, dataset: 3D Point cloud (1.3 GB), one rank per node, Intel Xeon Gold 6148

Clustering analysis with HeAT. 1201 data points, k=3 centroids. Upper panel: data (Open Exoplanet Catalogue); center and lower panel: K-Means results with HeAT and sklearn respectively. Cluster centroids are shown as grey circles.