The **Helmholtz Analytics Toolkit**

or:

D. Coquelin, C. Comito, M. Götz, B. Hagemeier, P. Knechtges, K. Krajsek, M. Siggel
What is HeAT?
The Helmholtz Analytics Toolkit

- Data analytics framework for transparent distributed computation
- Build on top of PyTorch - written in Python
- Part of the Helmholtz Analytics Framework (HAF)
- Project start: May 2018
- Developed in the open:
  - https://github.com/helmholtz-analytics
  - https://pypi.org/project/heat
- Liberally licensed: MIT
- Designed for extreme data scales
HAF

Who is involved?

April 15, 2019   Slide 3
Where Do I Fit In?

Earth Systems Modeling:
- Terrestrial Systems Modeling Platform (TerrSysMP)
- Simulation Laboratories
  - SLTS / CSLTS
# HAF

## What are the goals?

<table>
<thead>
<tr>
<th>Scientific Big Data Analytics:</th>
<th>● Develop and expand on Methodologies and tools for problems of the highest data and computational complexity</th>
</tr>
</thead>
</table>
| HeAT:                         | ● Foster data science in HAF  
                                ● Develop and exploit the Helmholtz Data Federation (HDF) |
| Use case driven co-design between - | ● Domain scientists  
                                ● Data experts  
                                ● Infrastructure professionals |
| Create Data analysis Techniques - | ● In a systematic manner  
                                ● Domain-specific as well as generalized and standardized |
## HAF Use Case Methodologies

<table>
<thead>
<tr>
<th>Methodology</th>
<th>Technique/Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clustering</td>
<td>K-means, mean shift clustering</td>
</tr>
<tr>
<td>Uncertainty Quantification</td>
<td>Ensemble methods</td>
</tr>
<tr>
<td>Dimension Reduction</td>
<td>Autoencoder, reduced order models</td>
</tr>
<tr>
<td>Feature Learning</td>
<td>Image descriptors, autoencoder</td>
</tr>
<tr>
<td>Data assimilation</td>
<td>Kalman filter, 4Dvar, particle filter/smoother</td>
</tr>
<tr>
<td>Classification/Regression</td>
<td>Random forest, CNN, SVM</td>
</tr>
<tr>
<td>Modelling</td>
<td>Fiber tractography, point processes</td>
</tr>
<tr>
<td>Optimization techniques</td>
<td>L-BFGS, simulated annealing</td>
</tr>
<tr>
<td>Hyper-parameter Optimization</td>
<td>Evidence framework, grid search</td>
</tr>
<tr>
<td>Interpolation</td>
<td>Radial basis function, kriging</td>
</tr>
<tr>
<td>Data mining</td>
<td>Frequent itemset mining</td>
</tr>
</tbody>
</table>
Why is HeAT needed?

Data scale + TSMP

- Extreme data scales in modern sciences
  - 50 GB of data generated daily by simulations in TSMP

- Extreme-scale data mandates data distribution across computing nodes
  - Pro: more computing power
  - Con: communication overhead
Why is HeAT Needed?

Why not use an existing data analysis framework?

<table>
<thead>
<tr>
<th>Framework</th>
<th>Spark</th>
<th>BigDL</th>
<th>PyTorch</th>
<th>MXNet</th>
<th>TensorFlow</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI</td>
<td>✗</td>
<td>✗</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>GPU</td>
<td>✗</td>
<td>✗</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>ML</td>
<td>✗</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>ND-Tensors</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Transparent</td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Distributed</td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Linear Algebra</td>
<td>✗</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>
Why HeAT?

How does HeAT solve the problem?

- Split data into multiple PyTorch tensors
- Aim to keep the Numpy API
- Run on both CPU and GPU
- Designed for a distributed data environment
PyTorch

Runs on:

- CPU
- GPU

Data structure
- ND-Tensor

Operations
- Elementwise operations
- Slicing
- Matrix operations
- Reduction
- Automatic differentiation
How does HeAT work?

Runs on:
- Multi-core CPUs
- Multi-core GPUs
- MPI

Data structure: ND-Tensor
- shape: (4, 3, 2)

Operations:
- Elementwise operations
- Slicing
- Matrix operations
- Reduction
- Automatic differentiation
HeAT Basics

Example:

```python
import heat as ht

# construct a range tensor
>>> range_data = ht.arange(6, split=0)

>>> range_data.mean()
2.5

>>> range_data.argmax()
5
```
HeAT

Asset Support Phase

Available features

- Global and local setters and getters
- CPU mean and std
- Transpose
- Parallel IO
- Many elementwise functions
- Max, Argmax, Min, Argmin

Upcoming features

- GPU mean and std
- where
- balance
- convolve

Distant Features:

- PCA via SVD
- Non-negative Matrix Decomposition
- + other data science methods!
HeAT

Challenges

- Random number generator
  - PyTorch randn dependent on tensor size
- Matrix multiplication
- Distributed Eigenvalue solver
Summary

- HeAT developed along scientific use cases
- HeAT extends PyTorch for transparent distributed computation
- Pre-alpha phase, basic operations implemented
- Light weight data analysis methods will be build based on HeAT

https://github.com/helmholtz-analytics/heat/
Thank You