Contribution ID: **72** Type: **Tutorial or Skill-Up**

Accelerating massive data processing in Python with Heat

Wednesday, March 6, 2024 1:30 PM (1h 30m)

Manipulating and processing massive data sets is challenging. For the vast majority of research communities, the standard approach involves setting up Python pipelines to break up and analyze data in smaller chunks, an inefficient and prone-to-errors process. The problem is exacerbated on GPUs, because of the smaller available memory.

Popular solutions to distribute NumPy/SciPy computations are based on task parallelism, introducing significant runtime overhead, complicating implementation, and often limiting GPU support to specific vendors.

In this tutorial, we will show you an alternative based on data parallelism. The open-source library Heat [1] builds on PyTorch and mpi4py to simplify porting of NumPy/SciPy-based code to GPU (CUDA, ROCm, including multi-GPU, multi-node clusters). Under the hood, Heat distributes massive memory-intensive operations and algorithms via MPI communication, achieving significant speed-ups compared to task-distributed frameworks. On the surface however, Heat implements a NumPy-like API, is largely interoperable with the Python array ecosystem, and can be employed seamlessly as a backend to accelerate existing single-CPU pipelines, as well as develop new HPC applications from scratch.

You will get an overview of:
- Heat's basics: getting started with distributed I/O, data decomposition scheme, array operations
- Existing functionalities: multi-node linear algebra, statistics, signal processing, machine learning…
- DIY how-to: using existing Heat infrastructure to build your own multi-node, multi-GPU research software.

We'll also touch upon Heat's implementation roadmap, and possible paths to collaboration.

[1] M. Götz et al., "HeAT –a Distributed and GPU-accelerated Tensor Framework for Data Analytics," 2020 IEEE International Conference on Big Data (Big Data), Atlanta, GA, USA, 2020, pp. 276-287, doi: 10.1109/BigData50022.2020.9378050.

Slot length

other(help with comment)

Primary author: COMITO, Claudia (Forschungszentrum Jülich, Jülich Supercomputing Centre)

Co-authors: HOPPE, Fabian (DLR - Institut für Softwaretechnologie - HPC); GUTIRREZ HERMOSILLO MURIEDAS, Juan Pedro (SCC); KRAJSEK, Kai (Forschungszentrum Jülich GmbH); TARNAWA, Michael (Forschungszentrum Jülich)

Presenters: COMITO, Claudia (Forschungszentrum Jülich, Jülich Supercomputing Centre); HOPPE, Fabian (DLR - Institut für Softwaretechnologie - HPC)