Analysis of FastEddy Model Data on GPUs

Shay Liu
Dept. of Earth and Atmospheric Sciences
Indiana University
Bloomington, Indiana 47408
xuecliu@iu.edu

Supreeth Suresh,
Cena Miller,
and Jeremy Sauer
National Center for Atmospheric Research
Boulder, Colorado 80305

***Abstract*—Data analysis of atmospheric model outputs is often embarrassingly parallel and compute intensive, and is traditionally done on Central Processing Units (CPU). FastEddy is a General Purpose Graphical Processing Units (GPGPU) -based Large Eddy Simulation (LES) atmospheric model developed at NCAR-Research Application Laboratory that solves fundamental dynamical equations and computes turbulence at a high resolution, producing large datasets that reside on GPUs. To reduce the amount of data movement from GPUs to CPUs for analysis, and to reduce the need for end-users to learn complex GPU programming using CUDA, this work explores performing the analysis on GPUs using only Python libraries. The analysis of a single data file performed on one GPU is implemented using the Cupy library, and then is parallelized onto multi-GPU nodes using Dask. Using GPU acceleration for the data analysis provided speed up in both cases. Acceptable differences between CPU- and GPU- computed variables are observed. The difference is well-constrained for simple perturbation calculations and increased for more complicated flux calculations.**

1. Introduction

Traditionally, atmospheric models' outputs are analyzed on CPUs. These analyses are often embarrassingly parallel and compute intensive. These types of tasks, however, are well-suited for GPU acceleration. The FastEddy model developed at National Center for Atmospheric Research (NCAR) is a GPU-based large eddy simulation (LES) model that solves for cloud and perturbation properties. Despite the fast data production on GPUs, the analysis of FastEddy data still takes place on CPUs. In order to match the high-speed data production of FastEddy, a method that analyzes the data in-situ on GPUs is explored. Additionally, to avoid the need for atmospheric data scientists to learn CUDA languages, the approach we take is purely Python-based.

1.1. Cupy

Python-based data analysis on CPU involves intensive use of Numpy, a library that provides element-wise array operations. Cupy is a Python library that has a high compatibility with Numpy, and works on GPUs [1]. Using the Cupy library for GPU acceleration of data analysis provides many advantages, including drop-in replacement for a large portion of Numpy code [1], and the preservation of popular data structures (e.g., xarray dataset, Pandas dataframe).

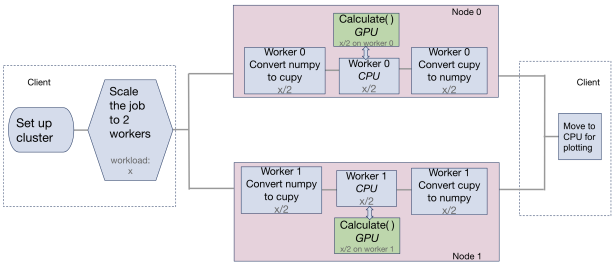

Figure 1. Dask workflow for 2-node GPU acceleration on two Dask workers.

1.2. Dask and Jupyterlab

Dask is a workflow scheduler and workload distributor. Incorporating Dask into the workflow simplifies the parallelization process by scaling up the resources per the user's request [2]. Dask also works with data structures found in the Pandas and xarray libraries.

JupyterLab is a web-based interactive development environment that supports Dask lab-extensions, which are used for monitoring the work process and data flow on individual Dask workers.

2. Dask workflow

While performing data analysis on a single GPU is straightforward, scaling to multi-node single-GPU is more complicated and involves the use of Dask. A workflow using Dask to schedule and scale the analysis of model output on GPUs is shown in Figure. 1. To initialize a Dask client, a cluster is first set up by passing job specifications. In our case of analyzing two output files at a time, the job is then scaled to two workers, and each worker takes half of the

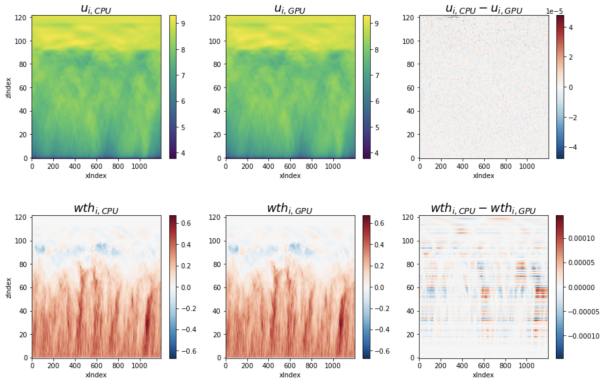

Figure 2. Validation of GPU-computed with CPU-computed FastEddy variables: horizontal perturbation wind speed (u_i, top row), and turbulent heat flux (wth_i, bottom row).

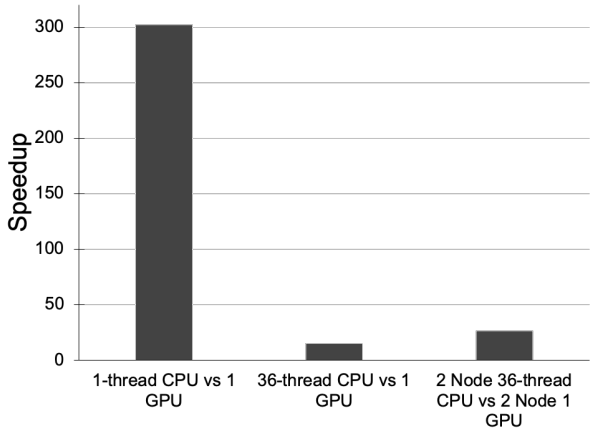

Figure 3. The speedup of single- and 2-node GPU acceleration with respect to single- and 2-node CPU analyses

workload. On each worker, the Numpy data are converted to Cupy data, and moved onto the GPU. The *Calculate()* function takes place on GPU, and completes all the calculations for the data analysis. On each worker, the computed data are then converted back to Numpy data and brought back to the CPU for any additional post-processing, such as plotting. This workflow is anticipated to be incorporated into the post-production of the FastEddy model, to evaluate processing the data in-situ on GPUs.

3. Results

To validate the analysis on GPUs, the GPU-calculated variables are compared against CPU-calculated variables. Timing information is gathered for the execution of the data analysis on both CPU and GPU.

3.1. Validation

Figure. 2 shows the plots of two FastEddy variables calculated on CPUs (left column) and on GPUs (middle column), and their differences (right column). The first variable we compare is the horizontal perturbation wind speed (u_i), shown in the upper panel. It is a relatively simple property calculated by taking the square root of the sum of the square of horizontal and vertical wind speed along the y dimension.

The difference residual is approximately white noise and on the magnitude of 10^{-5}. On the bottom panel is the turbulent heat flux (wth_i). This is a much more complicated variable that involves more layers of calculation. We can see the difference residual shows a larger magnitude of 10^{-4}, and a striped pattern that is likely developed from the accumulation of calculations. To use these types of GPU-computed variables may require investigation on their precision and uncertainties.

3.2. Speedup

Figure. 3 shows the speedup comparisons. The original data analysis was performed as a single-threaded CPU application. Using a single GPU, a 300x speedup is achieved versus single-thread CPU. A 15x speedup is achieved versus a 36-thread single-node CPU analysis. Using two GPUs separated on two nodes, a 26x speedup versus 36-thread two-node CPU analysis is observed.

4. Conclusion

Upon accomplishing the project, we find that using Dask and Cupy simplifies the process of GPU acceleration for data analysis. In the future, we would look to apply this technique to incorporate in-situ analysis of FastEddy data on the GPUs, to potentially increase the speed of the data science phase of FastEddy model, and reduce unnecessary data movement to the CPUs.

Acknowledgments

This research poster develops from a NCAR CISL SIParCS project. The authors would like to thank NCAR CISL for technical support and computing resources, as well as the SIParCS program and CODE team for funding and administrative support.

References

[1] R. Nishino and S. H. C. Loomis, "Cupy: A numpy-compatible library for nvidia gpu calculations," *31st confernce on neural information processing systems*, p. 151, 2017.

[2] M. Rocklin, "Dask: Parallel computation with blocked algorithms and task scheduling," in *Proceedings of the 14th python in science conference*, no. 130-136. Citeseer, 2015.