Too Big to Fail: Massive Scale Linear Algebra with Chapel and Arkouda

Christopher Hollis
U.S. Department of Defense

ABSTRACT

This presentation details the development of a linear algebra extension for Arkouda[1] (a NumPy-like Python application that utilizes Chapel for a backend server). This interface, dubbed AkSparse, allows for the creation and manipulation of sparse matrices at large scale with features designed to be familiar to users of SciPy's existing sparse array package[2]. This includes a sparse general matrix-matrix multiplication (SpGEMM) implemented with a novel algorithm leveraging the strengths of both Arkouda and Chapel. AkSparse allows users to integrate linear algebraic techniques into existing exploratory data analysis (EDA) workflows on datasets at a scale not previously possible.

1 INTRODUCTION

Effective EDA requires open-ended and frictionless interaction with data, but as datasets grow larger the performance of the necessary HPC tools begins to hinder the user's ability to easily investigate. Arkouda attempts to remedy this by providing a NumPy-like Python interface that prioritizes compatibility with existing data science workflows, while obscuring the underlying computations as they are handled by a Chapel server.

Linear algebra involving large-scale datasets has been an area of significant interest in the Arkouda community, as the application is particularly suited for interaction with data so large that it requires tools and techniques capable of running on multiple compute locales. The algorithms utilized by AkSparse takes advantage of Chapel's built-in support for multi-locale and multithreaded execution. AkSparse minimizes the communication costs inherent with the distributed-scale computation. The tool's performance on large-scale datasets will be included, highlighting the size of matrices that are now possible for use in EDA.

2 ALGORITHM

Large scale distributed SpGEMM faces many challenges, primarily involving the amount of memory needed to store the resulting matrix and the communication cost required to perform the multiplication. This becomes even more of an issue when the input matrices are highly unstructured, such as the graph Laplacian generated from a large network graph. AkSparse addresses the communication cost by leveraging the highly optimized sorting and groupby functionality provided by Arkouda. The issue of scale and available memory are handled by partitioning the input matrices, allowing for the SpGEMM computation to be broken into batches. To make this possible, AkSparse uses an "outer product" formulation of SpGEMM whose details will be discussed during the presentation.

3 CONCLUSION

This talk serves as an introduction to AkSparse and the problems it was designed to solve. The tool remains a work-in-progress with a number of plans for further optimization and support for additional linear algebra functionality.

REFERENCES

[1] GitHub - Bears-R-Us/arkouda: Arkouda (αρκούδα): Interactive Data Analytics at Supercomputing Scale -- a Python API powered by Chapel (https://github.com/Bears-R-Us/arkouda)
[2] Sparse matrices (scipy.sparse) – SciPy v1.10.1 Manual (https://docs.scipy.org/doc/scipy/reference/sparse.html)