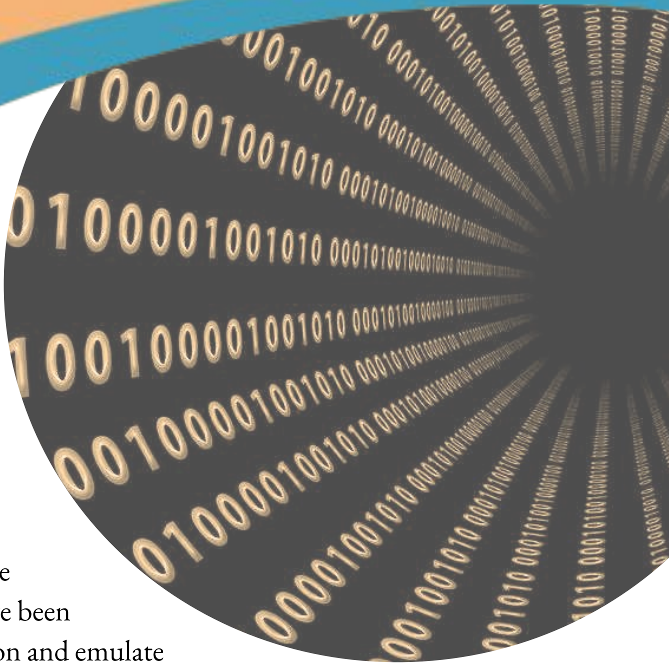
Sparse

Project Overview

The aim of PyData/Sparse is to create sparse containers that implement the ndarray interface. Traditionally in the PyData ecosystem, sparse arrays have been provided by the `scipy.sparse` submodule. All containers there depend on and emulate the `numpy.matrix` interface. This means that they are limited to two dimensions and also do not work well in places where `numpy.ndarray` would work.



PyData/Sparse is well on its way to replacing `scipy.sparse` as the de-facto sparse array implementation in the PyData ecosystem.

Roadmap Priorities

- More storage formats (the most important being CSF, a generalisation of CSR/CSC).
- Better performance/algorithms
- Covering more of the NumPy API
- SciPy Integration
- Dask integration for high scalability
- CuPy integration for GPU-acceleration
- Maintenance and General Improvements

QUANSIGHT

Sparse

More Storage Formats

In the sparse domain, you have to make a choice of format when representing your array in memory, and different formats have different trade-offs. For example:

- CSR/CSC are usually expected by external libraries, and have good space characteristics for most arrays
- DOK allows in-place modification and writes
- LIL has faster writes if written to in-order.
- BSR allows block-writes and reads

The most important formats are, of course, CSR and CSC, because they allow zero-copy interaction with a number of libraries including MKL, LAPACK and others. This will allow PyData/Sparse to quickly reach the functionality of `scipy.sparse`, accelerating the path to its replacement.

Better Performance/Algorithms

There are a few places in PyData/Sparse where algorithms are sub-optimal, sometimes due to reliance on NumPy which does not have these algorithms. PyData/Sparse intends to both improve the algorithms in NumPy, giving the broader community a chance to use them; as well as in PyData/Sparse, to reach optimal efficiency in the broadest use-cases.

Covering More of the NumPy API

Our eventual aim is to cover all areas of NumPy where algorithms exist that give sparse arrays an edge over dense arrays. Currently, PyData/Sparse supports reductions, element-wise functions, and other common functions such as stacking, concatenating and tensor products. Common uses of sparse arrays include linear algebra and graph theoretic subroutines, so those are the first areas that will be investigated.

SciPy Integration

PyData/Sparse aims to build containers and elementary operations on them, such as element-wise operations, reductions and so on. The plan is to modify the current graph theoretic subroutines in `scipy.sparse.csgraph` to support PyData/Sparse arrays. The same applies for linear algebra and `scipy.sparse.linalg`.

Dask Integration for High Scalability

Dask is a project that takes `ndarray` style containers and then allows them to scale across multiple cores or clusters. PyData/Sparse plans on tighter integration and cooperation with the Dask team to ensure the highest amount of Dask functionality works with sparse arrays.

CuPy integration for GPU-acceleration

CuPy is a project that implements a large portion of NumPy's ndarray interface on GPUs. PyData/Sparse plans to integrate with CuPy so that it is possible to accelerate sparse arrays on GPUs.

LEARN MORE
sales@quansight.com