The Journal of Open Source Software

scida: scalable analysis for scientific big data

Chris Byrohl [1] **and Dylan Nelson** [1]

1 Heidelberg University, Institute for Theoretical Astronomy, Albert-Ueberle-Str. 2, 69120 Heideberg, Germany

DOI: 10.21105/joss.06064

Software

- Review
- Repository
- Archive

Editor: Monica Bobra

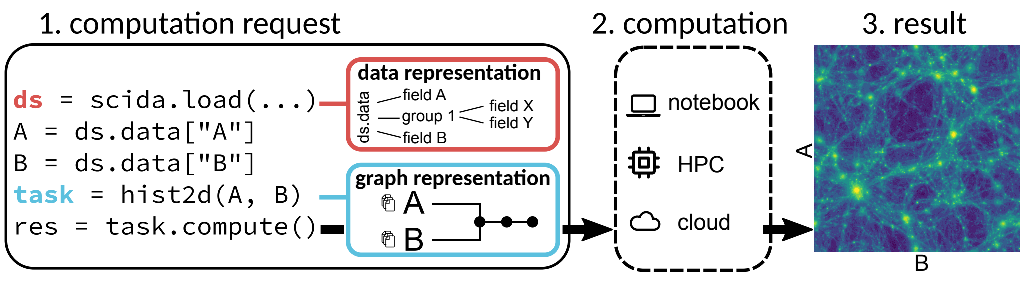
Reviewers:

- @egaraldi
- @kyleaoman

Submitted: 09 October 2023
Published: 23 February 2024

Summary

scida is a Python package for reading and analyzing large scientific data sets. Data access is provided through a hierarchical dictionary-like data structure after a simple load() function. Using the dask library for scalable, parallel and out-of-core computation (Dask Development Team, 2016), all computation requests from a user session are first collected in a task graph. Arbitrary custom analysis, as well as all available dask (array) operations, can be performed. The subsequent computation is executed only upon request, on a target resource (e.g. a HPC cluster, see Figure 1).



```
ds = scida.load(...)
A = ds.data["A"]
B = ds.data["B"]
task = hist2d(A, B)
res = task.compute()
```

Figure 1: Schematic of the workflow. In a user session, a recipe (i.e. sequence of analysis operations) for desired data product can be built by consecutive chaining of operations, which are internally represented by dask task graphs. Calculation is triggered by the compute() command, evaluating the graph on a target resource. The result, much smaller than the original data, is sent back to the user session for further analysis/plotting.

Features

Scida begins by providing a clean, abstract, dictionary-like interface to the underlying data, regardless of its file format or structure on disk. Physical units are automatically inferred, when possible, and attached to the dataset. Symbolic and automatic unit conversion is provided by the pint package (Grecco & others, 2012). This metadata can also be specified by the user via customizable configuration files.

Scida attempts to automatically determine the type of dataset loaded. At the file level, it currently supports zarr, single/multi-file HDF5 and FITS files. The analysis functionality then available depends on the dataset type. For example:

- Datasets which pair scattered pointsets with cataloged results of clustering algorithms have a natural association between such groups and their member points. Cosmological simulations in astrophysics provide one such motivating example, where halos and galaxies are identified in particle data. Scida then supports broadcasting of analysis and reduction operations over all groups for the associated particles.

- Datasets containing spatial coordinates in one, two, or three dimensions can be queried for spatial subsets.
- Datasets which are comprised of series, e.g. snapshots at different times, or parameter variation suites, are automatically inferred and supported.

The dataset-dependent functionality of scida is handled with a flexible and extensible mixin architecture.

Statement of need

Today, scientific "big data" is petabyte-scale. This makes traditional analysis by a researcher on their personal computer prohibitive. Writing analysis code which is distributed or out-of-core is complex, and is not the main focus of scientists. The need for significant data management creates (i) a barrier for new researchers, (ii) a substantial time commitment apart from the science itself, and (iii) an increased risk of errors and difficulties in reproducibility due to higher code complexity, while (iv) workflows are often not easily transferable to other datasets, nor (v) scalable and transferable to changing computing resources.

scida solves these problems by providing a simple interface to large datasets, hiding the complexity of the underlying data format and file structures, and transparently handling the parallelization of analysis computations. This is facilitated by the dask library, which naturally separates the definition of a given computation from its execution.

Initial support in scida is focused on astrophysical simulations and observations, but the package is designed to be easily extensible to other scientific domains. Existing analysis frameworks for astrophysical simulations include python packages such as yt (Turk et al., 2011), pynbody (Pontzen et al., 2013), pygad (Röttgers, 2018), nbodykit (Hand et al., 2018) and swiftsimio (Borrow & Kelly, 2021). None utilize the graph-based distributed analysis framework of dask. Often, existing analysis packages rely on the explicit loading of entire datasets into main memory. However, this approach is not transparently scalable to large data sets, and requires the user to explicitly manage the data chunks in custom analysis routines.

By providing a flexible interface to the dask library to handle large data sets in a scalable fashion, users can also leverage dask functionality and dask-based libraries such as dask-image (Kirkham & others, 2018) and datashader (Bednar et al., 2022).

scida was first utilized in Byrohl & Nelson (2023) for the analysis of cosmological and radiative transfer simulations, particularly the reduced chi-squared analysis exploring thousands of terabyte-sized models.

Target Audience

scida aims to simplify access to large scientific data sets. It lowers the barrier of entry for researchers to ask complex questions of big data. As such, the scida package is targeted at researchers with large data analysis problems. Its clean interface is appropriate for users with no prior experience with big data or distributed data analysis, as well as those who specifically want to leverage dask to make their workflows easier to read and scalable.

Domain specific analysis routines can be implemented on top of scida. Initial "out of the box" data support is currently focused on astrophysical data sets, but scida aims to support other scientific domains as well, where similar solutions will be beneficial.

Acknowledgements

CB and DN acknowledge funding from the Deutsche Forschungsgemeinschaft (DFG) through an Emmy Noether Research Group (grant number NE 2441/1-1).

References

Bednar, J. A., Crail, J., Crist-Harif, J., Rudiger, P., Brener, G., B, C., Thomas, I., Mease, J., Signell, J., Liquet, M., Stevens, J.-L., Collins, B., Thorve, A., Bird, S., thuydotm, esc, kbowen, Abdennur, N., Smirnov, O., … Bourbeau, J. (2022). *Holoviz/datashader: Version 0.14.3* (Version v0.14.3). Zenodo. https://doi.org/10.5281/zenodo.7331952

Borrow, J., & Kelly, A. J. (2021). *Projecting SPH particles in adaptive environments*. https://arxiv.org/abs/2106.05281

Byrohl, C., & Nelson, D. (2023). The cosmic web in Lyman-alpha emission. *Monthly Notices of the Royal Astronomical Society*, *523*, 5248–5273. https://doi.org/10.1093/mnras/stad1779

Dask Development Team. (2016). *Dask: Library for dynamic task scheduling*.

Grecco, H., & others. (2012). Pint: Makes units easy. In *GitHub repository*. GitHub. https://github.com/hgrecco/pint

Hand, N., Feng, Y., Beutler, F., Li, Y., Modi, C., Seljak, U., & Slepian, Z. (2018). nbodykit: An Open-source, Massively Parallel Toolkit for Large-scale Structure. *156*(4), 160. https://doi.org/10.3847/1538-3881/aadae0

Kirkham, J., & others. (2018). Dask-image: Distributed image processing. *GitHub Repository*. https://github.com/dask/dask-image

Pontzen, A., Roškar, R., Stinson, G. S., Woods, R., Reed, D. M., Coles, J., & Quinn, T. R. (2013). *pynbody: Astrophysics Simulation Analysis for Python*.

Röttgers, B. (2018). *pygad: Analyzing Gadget Simulations with Python* (p. ascl:1811.014). Astrophysics Source Code Library, record ascl:1811.014.

Turk, M. J., Smith, B. D., Oishi, J. S., Skory, S., Skillman, S. W., Abel, T., & Norman, M. L. (2011). yt: A Multi-code Analysis Toolkit for Astrophysical Simulation Data. *The Astrophysical Journal Supplement Series*, *192*, 9. https://doi.org/10.1088/0067-0049/192/1/9