Ensuring the continued growth of Pandas

Abstract

...

0.1 3a. Progress Report (250 words)

Maintenance

Our maintenance work focused on

1. Organizing pandas' community of volunteers
2. Ensuring software quality by quickly responding to high-priority issues (e.g. https://github.com/pandas-dev/pandas/issues/35517#issuecomment-667668033)

Our maintainers have organized a large pool of eager volunteers to work on small pieces of large projects. Our last two releases had a total of 676 contributors, 457 of which were first-time contributors.

We've used 77% of our hours allocated to maintenance. We will have no trouble spending the remaining hours by the end of the grant.

Extension Types

We've continued to expand the extension array interface to work better with the rest of pandas. With funding from CZI's EOSS, we've added

- An interface for concatenating Extension types: https://github.com/pandas-dev/pandas/pull/33607
- Support for masked arrays in pandas' algorithms (https://github.com/pandas-dev/pandas/pull/33064, https://github.com/pandas-dev/pandas/pull/30982)
- Support for extension arrays in the index (in progress: https://github.com/pandas-dev/pandas/issues/22861)

We've spent 93% of our hours allocated to this task.

Native String Dtype

We worked with Maarten Breddels to implement a framework for string kernels in Apache Arrow. Several kernels (e.g. `str.lower`) have been implemented.

With the release of Arrow 1.0, we'll use it in pandas. That should finish by the end of September.

- https://issues.apache.org/jira/browse/ARROW-555
- https://issues.apache.org/jira/browse/ARROW-9133
- https://github.com/pandas-dev/pandas/issues/35169

We've spent about 30% of our hours allocated to this task. This number is low because Maarten did not start on until a month ago, and the work in pandas was blocked by the release of Arrow 1.0. We'll quickly exhaust the remaining hours now that Maarten is free and Arrow 1.0 is out.

0.2 4. Proposal Purpose: (255 characters)

We aim to ensure the continued health of the pandas library by dedicating resources to specifically to maintenance and implementing consistent missing value handling.

0.3 6. Abstract (250 words)

This proposal seeks funding to help with the maintenance of pandas, a Python library providing high-performance, easy-to-use data structures. pandas is a foundational library in the Scientific Python Ecosystem, providing the most widely-used data structures for heterogenous, tabular data, and is the most-used Python tag on StackOverflow. Pandas documentation averages over 1,000,000 unique visitors per month. Specifically within the biomedical and life sciences, Pandas is used directly by researchers and by libraries. We plan to fund the continued health of the project through two main avenues:

First, we'd fund pandas maintainers to do the day-to-day work of maintaining a large project like pandas. This would include triaging issues, reviewing pull requests, maintaining release infrastructure, community building, and periodically cleaning up the issue tracker.

Second, we'd work to improve missing data handling within pandas. pandas' current handling of missing data is inconsistent across data types, resulting in confusing behavior for users and additional maintenance burden to handle the various cases. By implementing consisting missing data handling for all data types we hope to reduce the number of special cases within pandas, giving a simpler and easier to use library.

0.4 7. Work Plan (750 words)

Our work plan is broken into two main sections: maintenance and work on nullable data types.

Maintenance

Maintainers funded by this grant would be expected to

1. Promptly triage newly opened issues.
2. Promptly review new pull requests.
3. Ensure conversations on open issues are progressing and not waiting on maintainer feedback.
4. Ensure that Pull Requests are not going stale waiting for maintainer feedback or contributor responses to feedback.
5. Periodically review the issue backlog to find and close issues that are duplicates or no longer relevant. We should ensure that open issues are clearly described and scoped.
6. Review and address performance regressions.
7. Engage in discussions on the Pandas mailing list.
8. Mentor contributors, especially from those from underrepresented groups, who would like to become maintainers.

Over the past year, we've had success with maintainers coordinating pandas' large pool of willing volunteers to chip away at large projects. For example, we coordinated dozens of contributors to update pandas' codebase to use f-strings in https://github.com/pandas-dev/pandas/issues/29547.

Maintainers have also spent time going through pandas' issue backlog to find issues that are fixed but not yet closed. We then comment that this could use a test, and a community member will often come along with a Pull Request adding a test to ensure that the behavior is tested and close the issue (https://github.com/pandas-dev/pandas/issues/28397#issuecomment-623863221). This is a great way for new contributors to join the project, and increases the quality of pandas' issue backlog by closing out fixed items.

This item requires some familiarity with Pandas' codebase, community, and workflow. We hope to draw from Pandas' current pool of maintainersR1 and triagersR2 to find people with the necessary skills and experience. We expect this to take about 1 FTE over the course of the grant.

Nullable Data Types

In pandas 1.0, experimental "nullable" data types were introduced: the nullable integer, string and boolean data types use the new `pandas.NA` value to represent scalar missing values. These nullable data types provide consistent missing value handling accross the data types (with a behaviour that deviates from `np.nan` which historically was used as missing value indicator, but is limited to float and object data types).

We'd like to expand the nullable data type paradigm to more data types (e.g. float, categorical, and possibly date-like data types). We think that the nullable data types are more natural, powerful, and easier to use than pandas' current handling of missing values. Additionally, in a world where nullable types are consistently used for all data types, pandas will be able to dramatically simplify

its internal dtype casting logic that is currently necessary. This will help make pandas a more maintainable project going forward.

The recently added nullable data types are currently experimental and opt-in (the user must explicitly request the nullable versions if they wish to use them). While we'd like the nullable versions to be stable and the default, we must first update many of pandas' internal methods to understand these nullable types. We've made some progress on this with our current EOSS funding, and would like to continue that work.

Once the nullable types are working well with the rest of pandas, we'll add an option for users to globally opt-in to getting nullable types by default (e.g. from `read_csv`, when constructing a `DataFrame` from Python lists, etc.).

When it's time, we can make using nullable types the default. We'll provide users with a path to migrate to the new behavior with an option to opt-out, so that users will get them automatically.

0.5 8. Milestones and Deliverables (500 words)

Maintenance

Maintenance is a never-ending, hard-to-quantify task. That said, we can attempt to quantify it in a few ways.

- Fewer open issues in the backlog (3,500 at the time of writing) / more issues closed (160 over the past month)
- Fewer open pull requests (164 at the time of writing) / more merged pull requests (172 over the past month)
- Fewer open regressions (61 at the time of writing, soon after major release)

This component does not have any specific milestones; the work is ongoing. The timeline spans span the duration of the grant.

Nullable Data Types

Ensuring the nullable data types are a full replacement for the current data types will require both improvements to the general ExtensionArray mechanism as specific enhancements to the nullable data types. The following specific work items have been identified (with the expected completion time from the start of the funding period):

- Provide a mechanism to easily opt-in to use the nullable data types across the [Month 3].
- General improvements to the Extension Array interface:
  - Enable using extension arrays for the Index [Month 3]
  - Better support and customization of construction and casting operations (`astype()`) [Month 6]

  - Remaining numerical operations (e.g. `round`, `count`, cumulative methods, numpy protocols, ...) [Month 12]
- Specific improvements to the masked array implementation (these nullable arrays are composed of an array of values and a second `mask` array indicating whether each value is valid or `NA`) and algorithm support of the nullable data types:
  - Investigate ways to optimize the storage (optional masks, bitmasks) [Month 12]
  - Support masked arrays directly in more algorithms [Month 6]
- Expand nullable support to new data types (pending discussions on which data types to support) [Month 12].

The general improvements to the ExtensionArray interface will also benefit other projects making use of this interface (e.g. pint-pandas, awkward-array, GeoPandas, etc).

As these nullable types are experimental and opt-in, each pandas release can incrementally include those new features. When the project is ready for pandas 2.0, we have choice to make them the default for all users.

0.6 Landscape Analysis

Several other Python libraries provide a dataframe implementation that overlaps with what pandas provides. Of these, pandas is one of the oldest and most popular.

- cudf: Like pandas, but on the GPU
- dask.dataframe: Parallel / distributed dataframes. Uses pandas internally.
- vaex: scalable / parallel / larger-than-memory dataframe
- modin: Like pandas, uses Ray for parallelism
- koalas / pyspark: dataframes, executes on the JVM using Spark

Outside of Python, most languages used for scientific computing will have one or more dataframe implementations.

Pandas builds on top of other libraries like NumPy, pytz, and python-dateutil. We have optional features that build on libraries like Apache Arrow, PyTables, xarray, SQLAlchemy, fsspec, and others.

Other libraries build on top of pandas, for example statsmodels, featuretools, altair, seaborn, xarray, geopandas, Dask, and others. See https://pandas.pydata.org/docs/ecosystem.html for more.

0.7 11. Diversity, Equity, and Inclusion Statement

The pandas project and community recognizes that a diverse and inclusive community is essential to the success of the project.

We expect that all members of the communities, including veteran maintainers, adhere to and demonstrate the values laid out in our Code of Conduct.

We recognize that the limited diversity in the current maintainers does not reflect the diversity of pandas' users, much less the general population. Simply saying that "all contributions are welcome" is not enough to overcome the entrenched biases in open source that put hurdles in the way of contributing. So while contributions *are* always welcome, we place particular import on encouraging contributions from individuals from underrepresented communities.

We applaud and love working with groups like pandanistas We thank them not just for their contributions, but for their mentorship and for publicizing pandas as a great project for women and other under-represented groups in technology to contribute to.