Biggus Documentation

Release 0.15.0

Richard Hattersley

Jun 20, 2017

Contents

Virtual arrays of arbitrary size, with arithmetic and statistical operations, and conversion to NumPy ndarrays.

Virtual arrays can be stacked to increase their dimensionality, or tiled to increase their extent.

Includes support for easily wrapping data sources which produce NumPy ndarray objects via slicing. For example: netcdf4python Variable instances, and NumPy ndarray instances.

All operations are performed in a lazy fashion to avoid overloading system resources. Conversion to a concrete NumPy ndarray requires an explicit method call.

For example:

```
# Wrap two large data sources (e.g. 52000 x 800 x 600).
measured = OrthoArrayAdapter(netcdf_var_a)
predicted = OrthoArrayAdapter(netcdf_var_b)

# No actual calculations are performed here.
error = predicted - measured

# *Appear* to calculate the mean over the first dimension, and
# return a new biggus Array with the correct shape, etc.
# NB. No data are read and no calculations are performed.
mean_error = biggus.mean(error, axis=0)

# *Actually* calculate the mean, and return a NumPy ndarray.
# This is when the data are read, subtracted, and the mean derived,
# but all in a chunk-by-chunk fashion which avoids using much
# memory.
mean_error = mean_error.ndarray()
```

CHAPTER 1

Creating an Array

To make it easy to create an Array from existing data sources, biggus provides two adapter classes:

class biggus.**NumpyArrayAdapter**(*concrete*, *keys=()*)
Exposes a "concrete" data source which supports NumPy "fancy indexing" as a biggus.Array.

A NumPy ndarray instance is an example suitable data source.

NB. NumPy "fancy indexing" contrasts with orthogonal indexing which treats multiple iterable index keys as independent.

class biggus.**OrthoArrayAdapter**(*concrete*, *keys=()*)
Exposes a "concrete" data source which supports orthogonal indexing as a biggus.Array.

Orthogonal indexing treats multiple iterable index keys as independent (which is also the behaviour of a biggus.Array).

For example:

```
>>> ortho = OrthoArrayAdapter(ConstantArray(shape=[100, 200, 300]))
>>> ortho.shape
(100, 200, 300)
>>> ortho[(0, 3, 4), :, (1, 9)].shape
(3, 200, 2)
```

A netCDF4.Variable instance is an example orthogonal concrete array.

NB. Orthogonal indexing contrasts with NumPy "fancy indexing" where multiple iterable index keys are zipped together to allow the selection of sparse locations.

Constants

For creating an Array with the same value in every element, biggus provides a simple class and two convenience functions which mimic NumPy.

class biggus.**ConstantArray**(*shape*, *value=0.0*, *dtype=None*)
An Array which is completely filled with a single value.

Parameters

- **shape** (*int or sequence of ints*) – The shape for the new Array.
- **value** (*obj, optional*) – The value to fill the Array with. Defaults to 0.0.
- **dtype** (*obj, optional*) – Object to be converted to data type. Default is None which instructs the data type to be determined as the minimum required to hold the given value.

Returns An Array entirely filled with 'value'.

Return type Array

biggus.**zeros**(*shape*, *dtype=<type 'float'>*)
Return an Array which is completely filled with zeros.

Parameters

- **shape** (*int or sequence of ints*) – The shape for the new Array.
- **dtype** (*obj, optional*) – Object to be converted to data type. Default is *float*.

Returns An Array entirely filled with zeros.

Return type Array

biggus.**ones**(*shape*, *dtype=<type 'float'>*)
Return an Array which is completely filled with ones.

Parameters

- **shape** (*int or sequence of ints*) – The shape for the new Array.
- **dtype** (*obj, optional*) – Object to be converted to data type. Default is *float*.

Returns An Array entirely filled with ones.

Return type Array

Combining arrays

Multiple arrays can be combined by stacking them along a new dimension, or concatenating along an existing dimension:

class biggus.**ArrayStack**(*stack*)
An Array made from a homogeneous array of other Arrays.

Parameters **stack** (*array-like*) – The array of Arrays to be stacked, where each Array must be of the same shape.

class biggus.**LinearMosaic**(*tiles*, *axis*)

CHAPTER 2

Computations with Arrays

NB. When using biggus, all computations are deferred until the results are explicitly requested.

Statistical operations

biggus.**mean**(*a*, *axis=None*, *mdtol=1*)

Request the mean of an Array over any number of axes.

Note: Currently limited to operating on a single axis.

Parameters

- **axis** (*None, or int, or iterable of ints.*) – Axis or axes along which the operation is performed. The default (axis=None) is to perform the operation over all the dimensions of the input array. The axis may be negative, in which case it counts from the last to the first axis. If axis is a tuple of ints, the operation is performed over multiple axes.
- **mdtol** (*float*) – Tolerance of missing data. The value in each element of the resulting array will be masked if the fraction of masked data contributing to that element exceeds mdtol. mdtol=0 means no missing data is tolerated while mdtol=1 will mean the resulting element will be masked if and only if all the contributing elements of the source array are masked. Defaults to 1.

Returns The Array representing the requested mean.

Return type Array

biggus.**std**(*a*, *axis=None*, *ddof=0*)

Request the standard deviation of an Array over any number of axes.

Note: Currently limited to operating on a single axis.

Parameters

- **axis** (*None, or int, or iterable of ints.*) – Axis or axes along which the operation is performed. The default (axis=None) is to perform the operation over all the dimensions of the input array. The axis may be negative, in which case it counts from the last to the first axis. If axis is a tuple of ints, the operation is performed over multiple axes.
- **ddof** (*int*) – Delta Degrees of Freedom. The divisor used in calculations is N - ddof, where N represents the number of elements. By default ddof is zero.

Returns The Array representing the requested standard deviation.

Return type Array

biggus.**var**(*a*, *axis=None*, *ddof=0*)

Request the variance of an Array over any number of axes.

Note: Currently limited to operating on a single axis.

Parameters

- **axis** (*None, or int, or iterable of ints.*) – Axis or axes along which the operation is performed. The default (axis=None) is to perform the operation over all the dimensions of the input array. The axis may be negative, in which case it counts from the last to the first axis. If axis is a tuple of ints, the operation is performed over multiple axes.
- **ddof** (*int*) – Delta Degrees of Freedom. The divisor used in calculations is N - ddof, where N represents the number of elements. By default ddof is zero.

Returns The Array representing the requested variance.

Return type Array

Elementwise operations

biggus.**add**(*a*, *b*)

Return the elementwise evaluation of np.add(a, b) as another Array.

biggus.**sub**(*a*, *b*)

Return the elementwise evaluation of np.subtract(a, b) as another Array.

Evaluation

It is always possible to use the *masked_array()* and/or *ndarray()* methods which are present on every biggus Array.

Array.**masked_array**()

Returns the NumPy MaskedArray instance that corresponds to this virtual array.

Array.**ndarray**()

Returns the NumPy ndarray instance that corresponds to this virtual array.

But for multiple expressions with shared sub-expressions it is more efficient to request the evaluations in a single call.

biggus.**ndarrays**(*arrays*)
Return a list of NumPy ndarray objects corresponding to the given biggus Array objects.

This can be more efficient (and hence faster) than converting the individual arrays one by one.

For expressions whose results are too large to return as a numpy ndarray it is possible to request they be sent piece-by-piece to an alternative output, e.g. an HDF variable.

biggus.**save**(*sources*, *targets*, *masked=False*)
Save the numeric results of each source into its corresponding target.

Parameters

- **sources** (*list*) – The list of source arrays for saving from; limited to length 1.
- **targets** (*list*) – The list of target arrays for saving to; limited to length 1.
- **masked** (*boolean*) – Uses a masked array from sources if True.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

b

Index