DistArray: Distributed NumPy

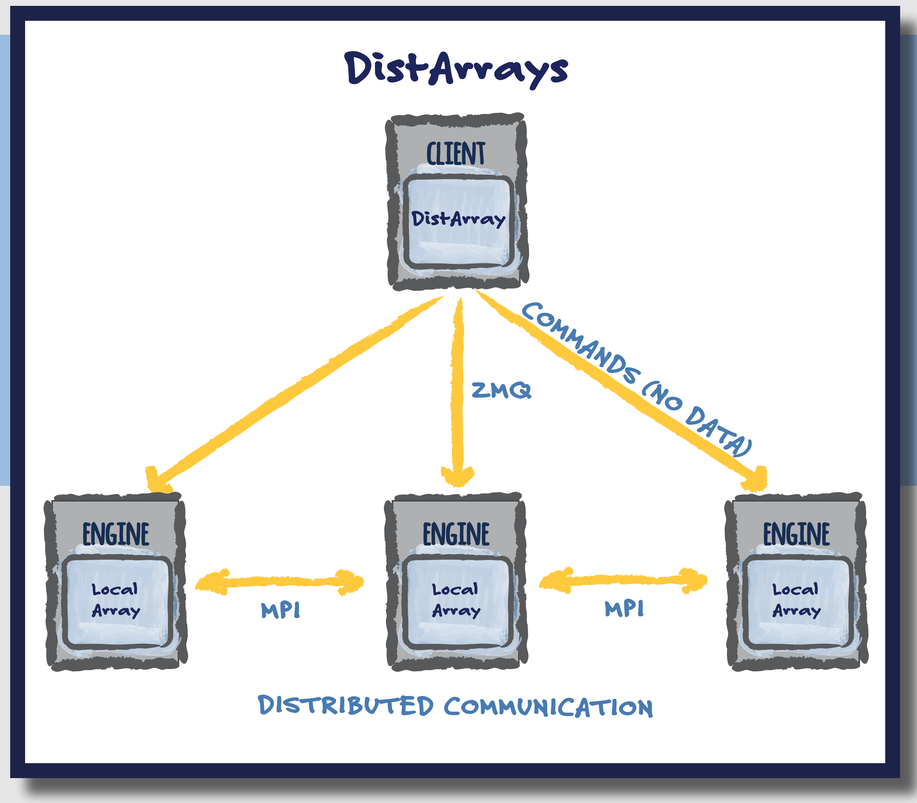
KW Smith, RD Grant, WF Spotz

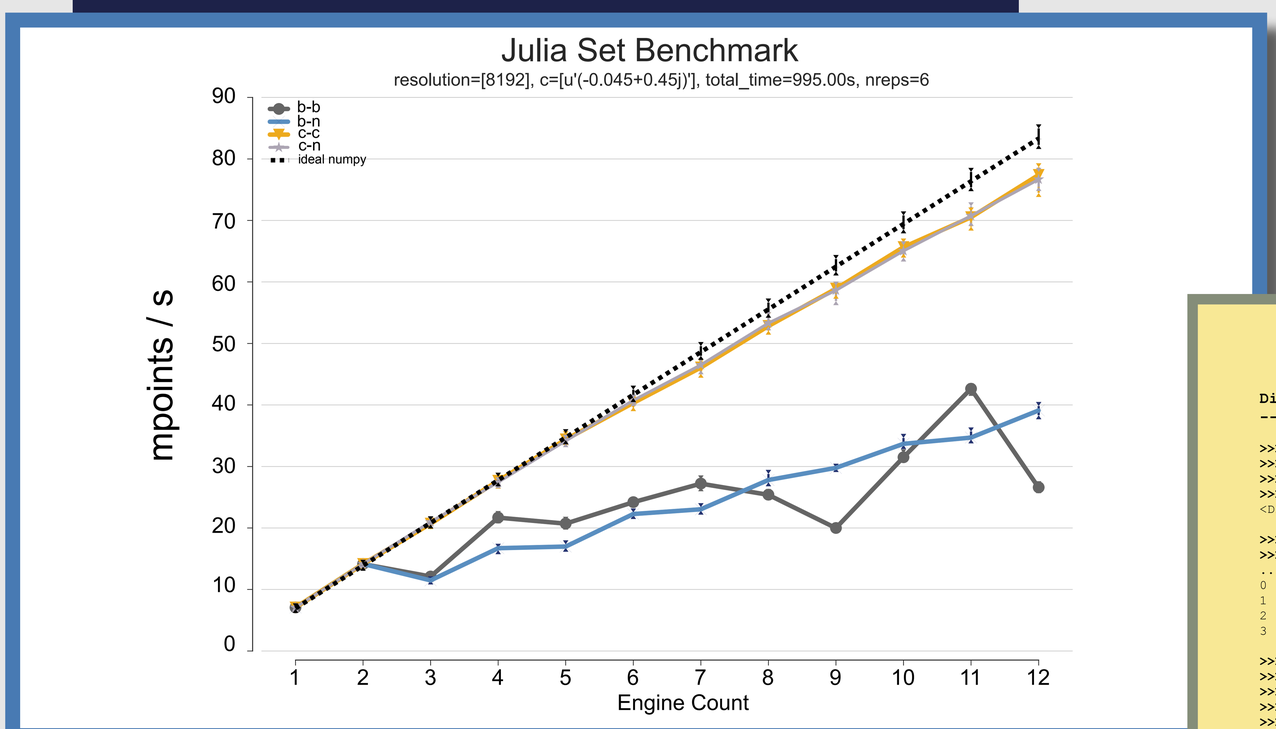
ENTHOUGHT

Overview

DistArray is an up-and-coming Python package providing ***distributed*** NumPy-like multidimensional arrays, ufuncs, and IO to bring the strengths of NumPy to data-parallel high-performance computing (HPC).

We build on widely-used Python HPC libraries and have introduced the Distributed Array Protocol to exchange arrays without copying with external distributed libraries like Trilinos.

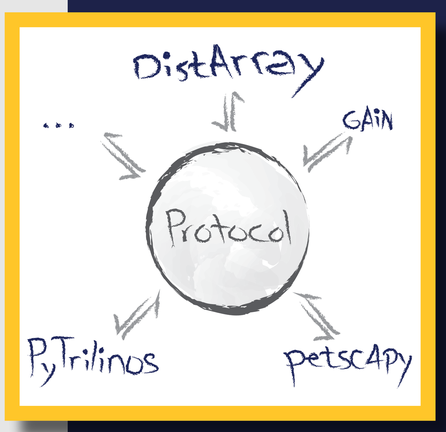




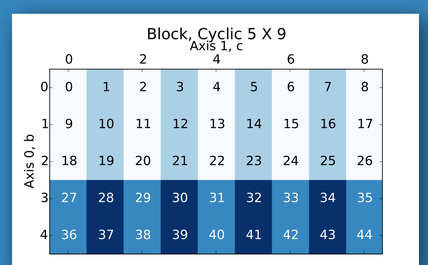

```
Distributed Slicing
-------------------

>>> darr.localshapes()
[(1, 5), (1, 5), (1, 5), (1, 5)]

>>> print(darr[:, 2::2])
>>> print(darr[:-1, 2::2].toarray())
<DistArray(shape=(4, 2), targets=[0, 1, 2, 3])>
[[ 0.    0.06]
 [ 0.79  0.54]
 [ 0.71  0.98]]

Reductions
----------

>>> print("sum:", darr.sum(), darr.sum().toarray())
>>> print("sum over an axis:", darr.sum(axis=1),
...       darr.sum (axis=1).toarray())
sum: <DistArray(shape=(), targets=[0])> 9.67697548135
sum over an axis: <DistArray(shape=(4,), targets=[0, 1, 2, 3])>
[ 2.01 2.51  3.48  1.67]

IO Support
----------

>>> # save DistArrays to .hdf5 files in parallel
>>> context.save_hdf5("/tmp/outfile.hdf5", darr, mode='w')

>>> # load DistArrays from .hdf5 files in parallel (using h5py)
>>> context.load_hdf5("/tmp/outfile.hdf5", distribution)
<DistArray(shape=(4, 5), targets=[0, 1, 2, 3])>

Context.apply
-------------

>>> def get_local_random():
...     import numpy
...     return numpy.random.randint(10)
>>> context.apply(get_local_random)
[9, 7, 1, 8]

>>> def get_local_var(darr):
...     return darr.ndarray.var()
>>> context.apply(get_local_var, args=(darr.key,))
[0.10035742766264857,
 0.059686240460491848,
 0.048523006620087086,
 0.036220776795190651]

Registering Functions in a Context
----------------------------------

>>> def local_demean(la):
...     """Return the local array with the mean removed."""
...     return la.ndarray - la.ndarray.mean()
>>> context.register(local_demean)

>>> context.local_demean(darr)
[array([[ 0.08,  0.35, -0.4 ,  0.31, -0.34]]),
 array([[-0.29,  0.24,  0.29, -0.28,  0.04]]),
 array([[ 0.15, -0.37,  0.01, -0.07,  0.28]]),
 array([[ 0.44,  0.11,  0.11, -1.  ,  0.34]])]
```

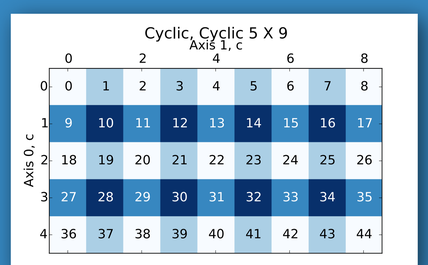
```
DistArrays
----------

>>> from distarray.globalapi import Context
>>> context = Context()
>>> darr = context.fromarray(nparr)
>>> darr
<DistArray(shape=(4, 5), targets=[0, 1, 2, 3])>

>>> # parts of the array are stored on each engine
>>> for i, a in enumerate(darr.get_localarrays()):
...     print(i, a)
0 [[ 0.48  0.75  0.    0.71  0.06]]
1 [[ 0.21  0.74  0.79  0.23  0.54]]
2 [[ 0.84  0.33  0.71  0.62  0.98]]
3 [[ 0.46  0.13  0.12  0.61  0.35]]

>>> # DistArray attributes
>>> print("type:", type(darr))
>>> print("dtype:", darr.dtype)
>>> print("ndim:", darr.ndim)
>>> print("shape:", darr.shape)
>>> print("itemsize:", darr.itemsize)
>>> print("nbytes:", darr.nbytes)
type: <class 'distarray.globalapi.distarray.DistArray'>
dtype: float64
ndim: 2
shape: (4, 5)
itemsize: 8
nbytes: 160

>>> # with some extra...
>>> print("targets:", darr.targets)
>>> print("context:", darr.context)
>>> print("distribution:", darr.distribution)
targets: [0, 1, 2, 3]
context: <[...]IPythonContext object at 0x10809f850>
distribution: <[...]Distribution object at 0x108763650>

Universal Functions (ufuncs)
----------------------------

>>> # DistArray
>>> import distarray.globalapi as da
>>> da.sin(darr)
<DistArray(shape=(4, 5), targets=[0, 1, 2, 3])>

>>> da.sin(darr).toarray()
array([[ 0.46,  0.69,  0.  ,  0.65,  0.06],
       [ 0.21,  0.67,  0.71,  0.22,  0.51],
       [ 0.75,  0.32,  0.65,  0.58,  0.83],
       [ 0.44,  0.13,  0.12,  0.57,  0.35]])
```

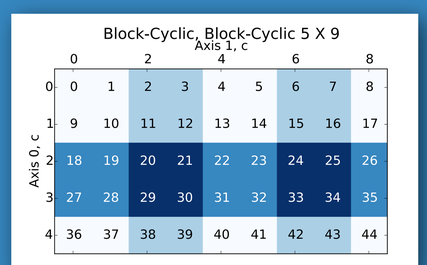
DISTRIBUTED ARRAY PROTOCOL

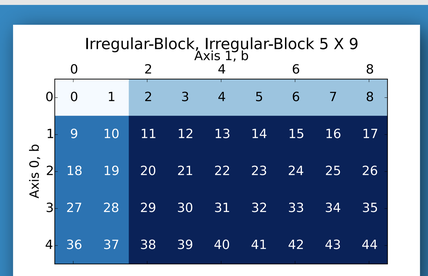


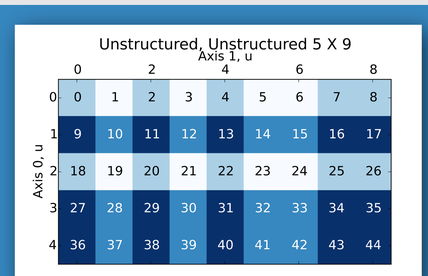
* Python-level protocol to share distributed arrays without copying.
* Process-local description of array's distribution.
* Supported in DistArray and PyTrilinos projects.
* Compatible with PETSc4Py, GAiN, and other projects.
* Per-dimension description. Supports block, cyclic, block-cyclic, unstructured.

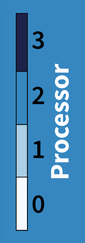












ksmith@enthought.com distarray@googlegroups.com

This material is based upon work supported by the Department of Energy under Award Number DE-SC0007699.