



Awkward Arrays in Dask

Doug Davis

March 28, 2022



Outline

- 1 | Dask Overview
- 2 | Dask-ifying Awkward
- 3 | Notebook Demo

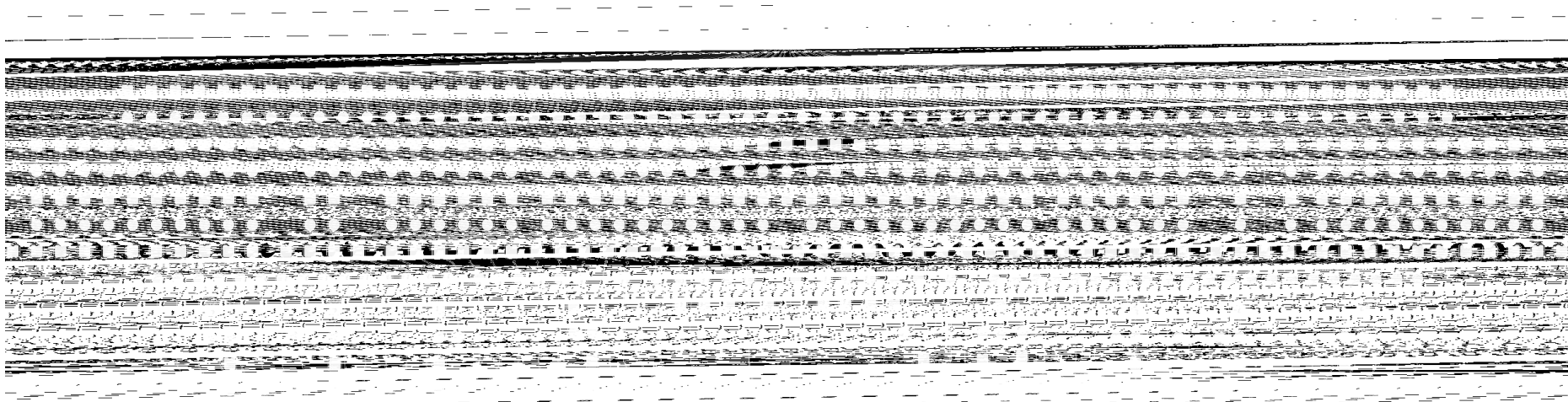


Dask Overview



Dask

creates and executes task graphs.



Dask

creates and executes task graphs.

Collections

(create task graphs)



Dask Array



Dask DataFrame



Dask Bag



Dask Delayed



Futures



Dask Awkward

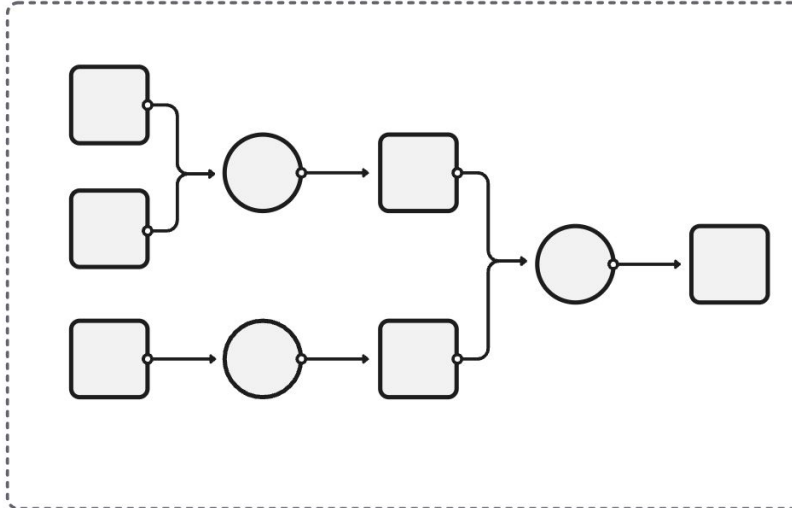


Task Graph



Schedulers

(execute task graphs)



Single-machine
(threads, processes,
synchronous)

Distributed



Dask Collections

provide a high level API for creating tasks.

```
da.ones(300, chunks=100)
```



Dask Collections


provide a high level API for creating tasks.

```
x = da.ones(300, chunks=100)
y = x + x
```

```
x ~ {
  ("ones-abc123", 0): (np.ones, 100),
  ("ones-abc123", 1): (np.ones, 100),
  ("ones-abc123", 2): (np.ones, 100),
}
```

*keys become dependencies
when building graphs*

```
y ~ {
  ("add-def456", 0): (operator.add, ("ones-abc123", 0), ("ones-abc123", 0)),
  ("add-def456", 1): (operator.add, ("ones-abc123", 1), ("ones-abc123", 1)),
  ("add-def456", 2): (operator.add, ("ones-abc123", 2), ("ones-abc123", 2)),
}
```



Daskifying Awkward



Partitioning an Awkward Array

```
ak.Array([\n    [{"x": 1.1, "y": [1]}, {"x": 2.2, "y": [1, 2]}, {"x": 3.3, "y": [1, 2, 3]}],\n    [],\n    [{"x": 2.2, "y": [1, 2]}, {"x": 8.5, "y": []}],\n    [],\n    [{"x": 3.3, "y": [1, 2, 3]}, {"x": 2.2, "y": []}, {"x": 3.3, "y": [3, 2, 1]}],\n    [{"x": 4.4, "y": [1, 2, 3, 4]}, {"x": 5.5, "y": [1, 2, 3, 4, 5]}],\n    [{"x": 5.5, "y": [1, 2, 3]}, {"x": 2.2, "y": [1, 2]}, {"x": 3.3, "y": [1, 2]}],\n    [{"x": 6.6, "y": [1, 2, 0, 0]}, {"x": 8.5, "y": []}],\n    []\n])
```

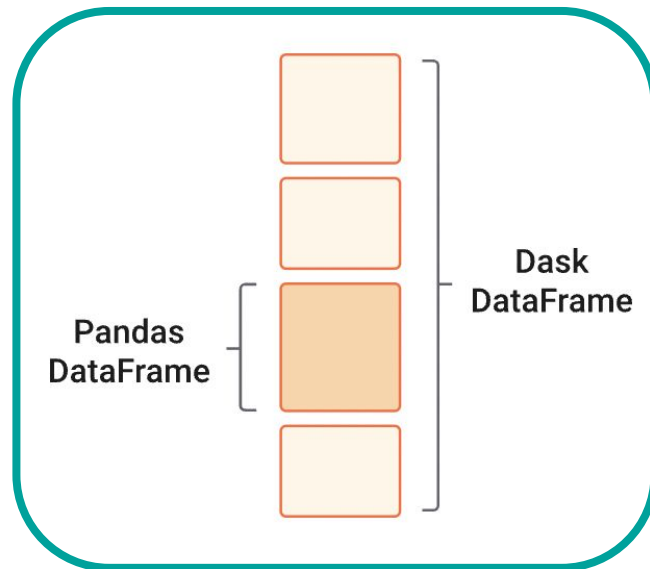
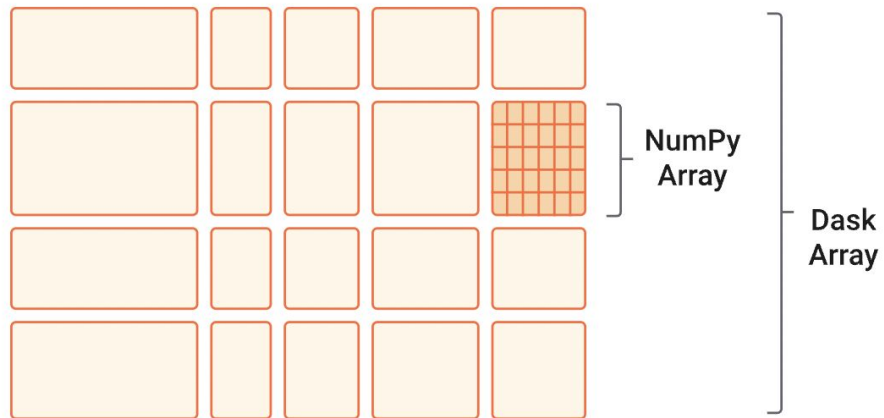


Partitioning an Awkward Array

```
[
  [{"x": 1.1, "y": [1]}, {"x": 2.2, "y": [1, 2]}, {"x": 3.3, "y": [1, 2, 3]}],
  [],
  [{"x": 2.2, "y": [1, 2]}, {"x": 8.5, "y": []}],
  # -----
  [],
  [{"x": 3.3, "y": [1, 2, 3]}, {"x": 2.2, "y": []}, {"x": 3.3, "y": [3, 2, 1]}],
  [{"x": 4.4, "y": [1, 2, 3, 4]}, {"x": 5.5, "y": [1, 2, 3, 4, 5]}],
  # -----
  [{"x": 5.5, "y": [1, 2, 3]}, {"x": 2.2, "y": [1, 2]}, {"x": 3.3, "y": [1, 2]}],
  [{"x": 6.6, "y": [1, 2, 0]}, {"x": 8.5, "y": []}],
  [],
]
```



is similar to a DataFrame



Scope highlights

- *dask-awkward is in development but welcomes curious users.*
- Some things currently supported:
 - Element-wise operations (e.g. NumPy style ufuncs)
 - Many getitem/slicing operations
 - `x.foo`, `x["foo"]`, `x[["foo", "bar"]]`, `x[:, 0]`
 - Row-wise operations (e.g. boolean selection, (some) inner-axis operations)
 - to/from `dask.array` & `dask.delayed`
- Some things partially supported (in progress) or planned:
 - Reductions and structure functions
 - I/O: (relatively) simple JSON input supported; parquet in progress; ROOT experimentation.
 - to/from `dask.dataframe`
 - Numba has been experimented with
 - Improved Jupyter support (notebook representation of the collection).



Live Demo

<https://github.com/douglasdavis/dask-awkward-demo>

