< (/nDarrays/07-groupby-processing/)

out-of-core computation

> (/nDarrays/09-masking/)

Overview

Teaching: 5 min
Exercises: 5 min
Questions

- How can we do computations on array datasets that are too large to fit into memory on a local machine?

Objectives

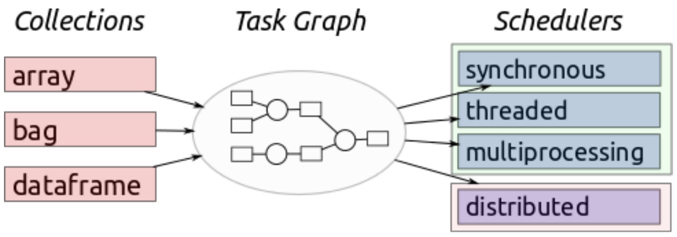
- understand best practices for reading and storing large gridded datasets
- using multi-threading libraries to facilitate manipulation of larger-than-memory grids

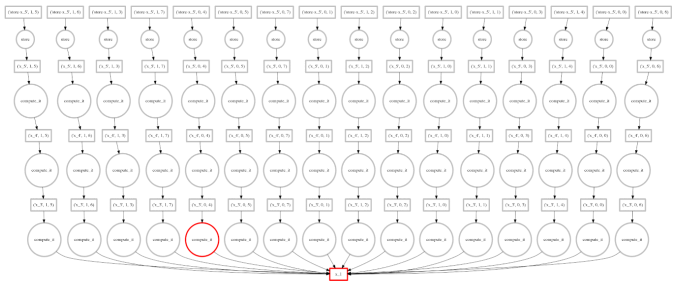
Handling large grids

When xarray carries out processing on an array it must load it into memory. Many datasets are becoming too large for this to be carried out on a typical laptop. For this reason, xarray integrates with a parallel computing library called Dask (http://xray.readthedocs.org/en/stable/dask.html). Dask uses task scheduling and blocked algorithms to enable processing of datasets that "fit on disk" even if they do not "fit in memory".

Dask:

- dask.array = numpy + threading
- dask.bag = map, filter, toolz + multiprocessing
- dask.dataframe = pandas + threading





Opening multiple netCDF files, and using Dask

We will use the mfdataset (http://xarray.pydata.org/en/stable/generated/xarray.open_mfdataset.html#xarray.open_mfdataset) option that opens multiple files as a single xarray dataset. This automatically invokes the dask functionality:

```
ds = xr.open_mfdataset('<root_dir>*_AK.nc', chunks = {'time':10})
```

Chunk sizes:

Without specifying chunk size, open_mfdataset chunks along existing dimensions. Getting the chunk size right is the crucial step to optimize working with xarray/dask. We recommend following this advice (http://xarray.pydata.org/en/stable/dask.html?highlight=rechunk#chunking-and-performance). You should use chunk sizes of about 1 million elements. In our case: 480* 241 = 115680, so make the time chunk 10 to get around 1 million. Note that we are only chunking the time dimension. Choice depends on how you will be working with the data.

Now when can carry out any processes on the `Dataset`, `dask` will be invoked. It is wise to include the `ProgressBar` tool from `dask.diagnostics` to track the processing:

```
from dask.diagnostics import ProgressBar
with ProgressBar():
  ds.sst.groupby('time.year').mean().plot()
```

Key Points

- dask integration with xarray allows you to work with large datasets that “fit on disk” rather than having to “fit in memory”.
- It is important to chunk the data correctly for this to work.