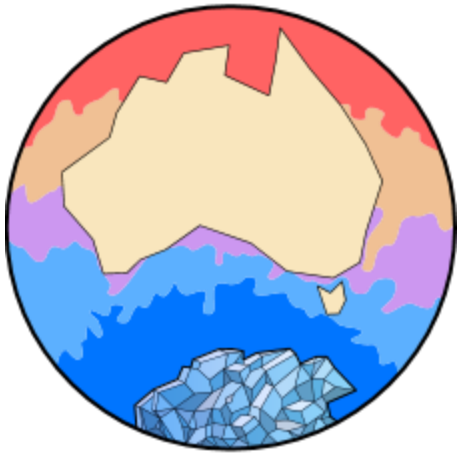

Making it go

Optimising dataset access through the intake-catalogue and Cosima Recipes

Andy Hogg, **Charles Turner**, Dougie Squire, Marc White, Romain Beucher, Anton Steketee

ACCESS-NRI Intake Catalog: Making it go

- Transitioning from cosima-cookbook to ACCESS-NRI Intake Catalog, opening some datasets has gotten much slower.
- `datastore.to_dask()` and `cc.querying.getvar()` pass different default arguments to `xr.open_mfdataset()`.
- These performance issues can (usually) be fixed with a few keyword arguments.

Today we will focus on `datastore.to_dask()`.

The exact same arguments can be used for `datastore.to_dataset_dict()`.

`xarray_open_kwargs`

Key arguments:

- `chunks`: Using good chunk sizes can (marginally) speed up reads, will often substantially speed up operations on open datasets.
- `use_cftime, decode_times, decode_timedelta`: Fiddling with these can massively reduce the number of warnings.

Any argument can be passed to `xarray.open_dataset()`.

https://docs.xarray.dev/en/stable/generated/xarray.open_dataset.html

xarray_combine_by_coords_kwargs

Key arguments:

- `compat`: Setting `compat='override'` can massively speed up loading datasets in some instances - ~5min => ~10 seconds. Most effective with multidimensional coordinate arrays.
- `data_vars, coord_vars` : Using `compat='override'` can sometimes cause errors unless these are set to `` `minimal` ``
- `decode_times, decode_timedelta:` Fiddling with these can massively reduce the number of warnings.

https://docs.xarray.dev/en/stable/generated/xarray.combine_by_coords.html

Rules of thumb

- Provided you know your model grid does not change, `compat='override'` is likely to provide the greatest performance increase.
- Chunking may improve performance, particularly when operating on datasets. However, choosing poor chunks will likely slow things down.
- Aim for 300MiB chunks when loading, as a general rule, and try to load files as a single chunk if possible.
- Selecting the minimal dataset before loading with `` `.to_dask()` `` will often yield greater performance gains

Bonus: Selecting date ranges from a datastore

- intake-esm does not provide `start_time`, `end_time` arguments like `cc.querying.getvar()` does.
- Instead, we need to search for `start_date`, or `end_date` to limit our query.
- This will likely speed up loads **substantially.**
- We can use regex (regular expressions) to do this, eg:

```
[15]: datastore = catalog['01deg_jra55v13_ryf9091']
      min(datastore.unique().start_date), max(datastore.unique().end_date)

[15]: ('1900-01-01, 00:00:00', '2180-01-01, 00:00:00')

[16]: ds_1970s = datastore.search(start_date='197\d{1}.*')
      # or ds_1970s = datastore.search(start_date='197.*') - if you don't have years including 197 or 19700, etc.
      min(ds_1970s.unique().start_date), max(ds_1970s.unique().end_date)

[16]: ('1970-01-01, 00:00:00', '1980-01-01, 00:00:00')

[17]: ds_2004 = datastore.search(start_date='2004.*')
      min(ds_2004.unique().start_date), max(ds_2004.unique().end_date)

[17]: ('2004-01-01, 00:00:00', '2005-01-01, 00:00:00')
```