Sharing and analysing large datasets: a COSIMA case study

Andrew Kiss (Research School of Earth Sciences, ANU)

Credits: James Munroe, Aidan Heerdegen, Angus Gibson, Micael Oliviera, Navid Constantinou, Andy Hogg, Nic Hannah, Russ Fiedler and others in the COSIMA community

Australian National University

NATURAM PRIMUM COGNOSCERE RERUM

COSIMA

Consortium for Ocean and Sea Ice Modelling in Australia

Acknowledgment of Country

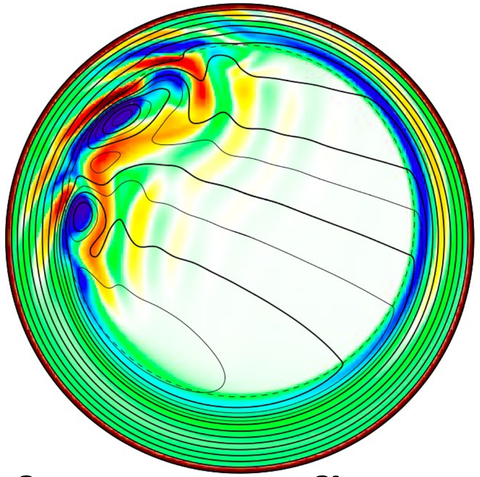
We acknowledge and celebrate the First Australians on whose traditional lands we meet, and pay our respect to the elders past and present.

At ANU we are surrounded by history 100 times older than colonisation. We are on the lands of the Ngunnawal people, which they have occupied for at least 20,000 years, deep in the last glacial period. There are rock art sites, ceremonial gathering places and an initiation site within a short walk of the ANU campus.

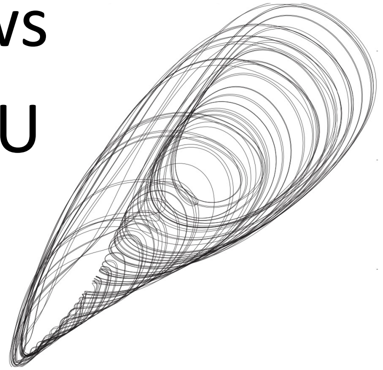
As a mark of respect, please make a well-informed choice in the referendum on recognition of First Nations people via a Voice to Parliament enshrined in the Constitution.

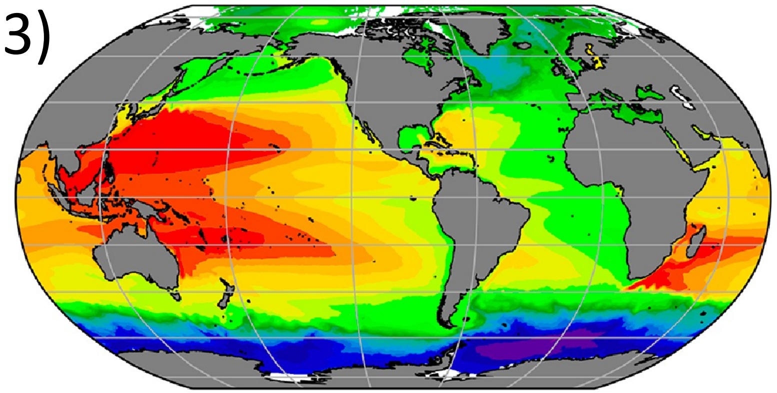
My Country, Ngunnawal Country by Adrian Brown

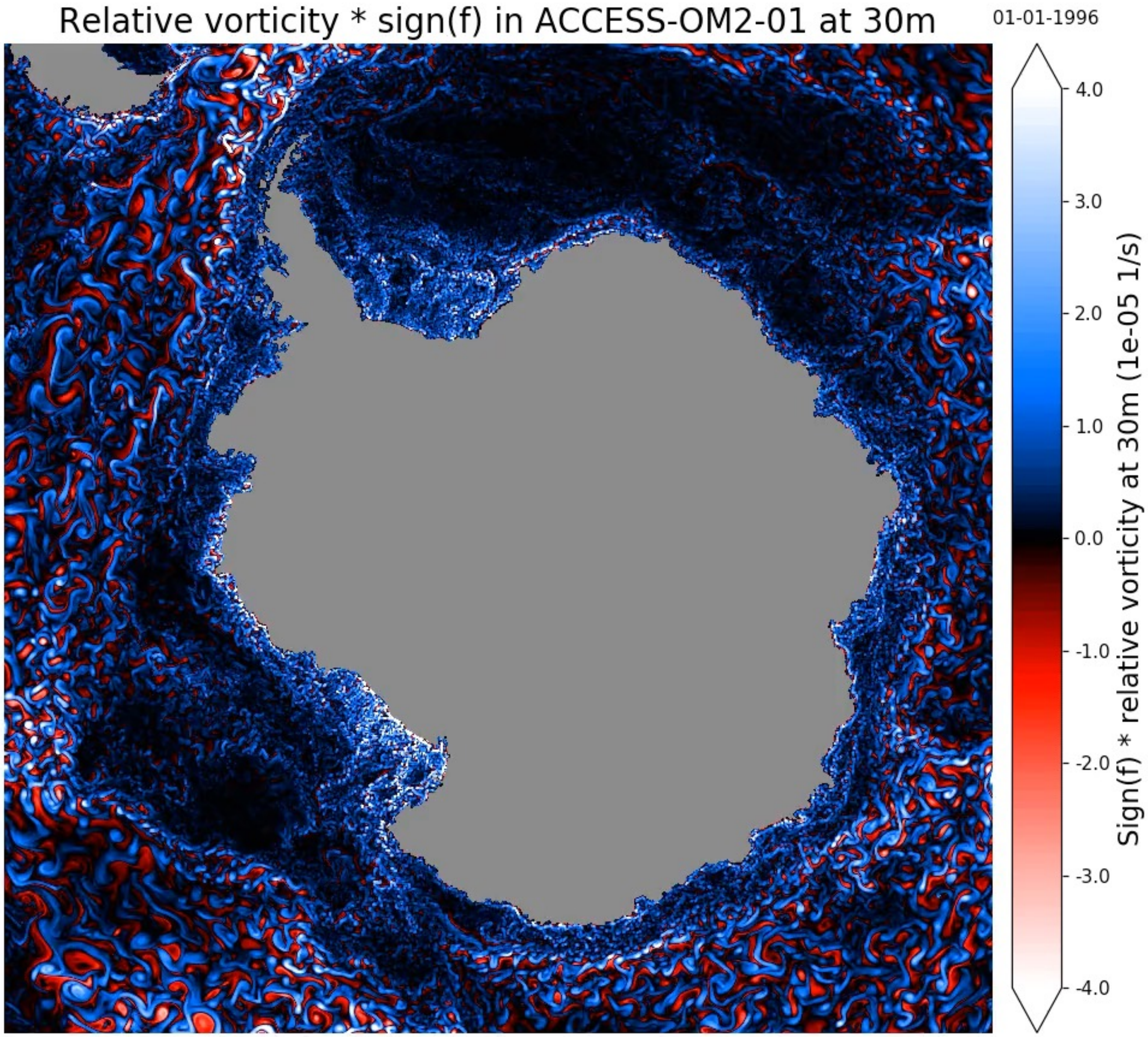
A little about me



- BSc (Hons Physics), ANU
- PhD in geophysical fluid dynamics, ANU: dynamics of rotating flows
- Postdoc in dynamical systems approaches to ocean instability, ANU
- Lecturer in Oceanography and Mathematics, UNSW Canberra
- Now: Fellow at Research School of Earth Sciences, ANU
  - In COSIMA team that developed ACCESS-OM2 global ocean and sea-ice model
  - Leading COSIMA and ACCESS-NRI team to develop Australia's next-generation global ocean and sea-ice model configurations (ACCESS-OM3)

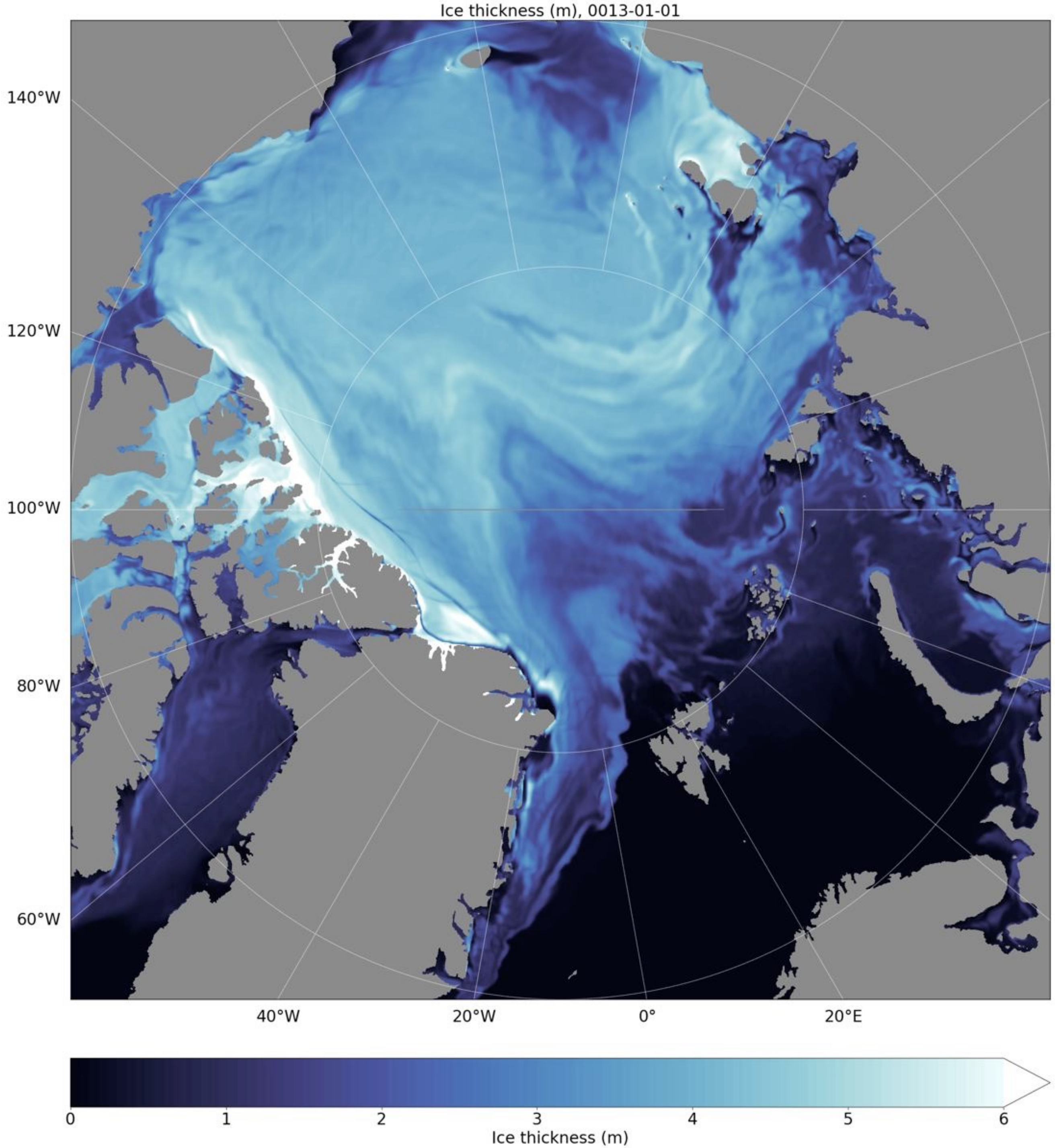






What's an ocean model?

- Specialized CFD codes for **3D density-stratified rotating flows** in an irregular, very thin spherical shell, driven by spatiotemporally variable surface stress and buoyancy flux
- Useful for understanding ocean circulation: dynamics and impact on climate change, ocean conveyor circulation, marine heatwaves, and as a component in climate models and ocean forecasting
- Represented by approximations to the **Navier-Stokes equations** and **advection-diffusion of heat and salt**, with **turbulence closures and parameterisations** for unresolved scales
- Coriolis and stratification dominate dynamics
- Similar problem to weather forecasting, but ocean is more expensive to simulate due to main eddy scales being >10x smaller – so need much higher resolution to represent them (e.g. for ocean forecasting), or use lower resolution and eddy parameterization (e.g. for climate models)
- Movie: vorticity at 30m in a global 0.1° model
    - Finite-volume model
    - Structured grid, 3600 x 2700 x 75 = 729M cells



What’s a sea ice model?

- Sea ice is frozen seawater, partially covering ocean, floes < 1m to many km
- Crucial component of Earth's climate system, and therefore climate models
- Major influence on ocean surface stress, buoyancy flux, reflectivity, gas exchange
- Granular medium, but represented in models as a bulk **viscoplastic fluid** with a distribution of thicknesses at each point
- Thermodynamic growth and decay, and mechanical ridging, rafting, advection
- Structured grid; ridging/rafting represented by mass transfer between discrete thickness categories in each cell
- Movie: sea ice thickness in ACCESS-OM2 global 0.1° model
  - Finite-volume model
  - Structured grid, 3600 x 2700 = 9.7M cells, each with 4 ice layers, 1 snow layer and 5 thickness categories

ACCESS-OM2 Codebase

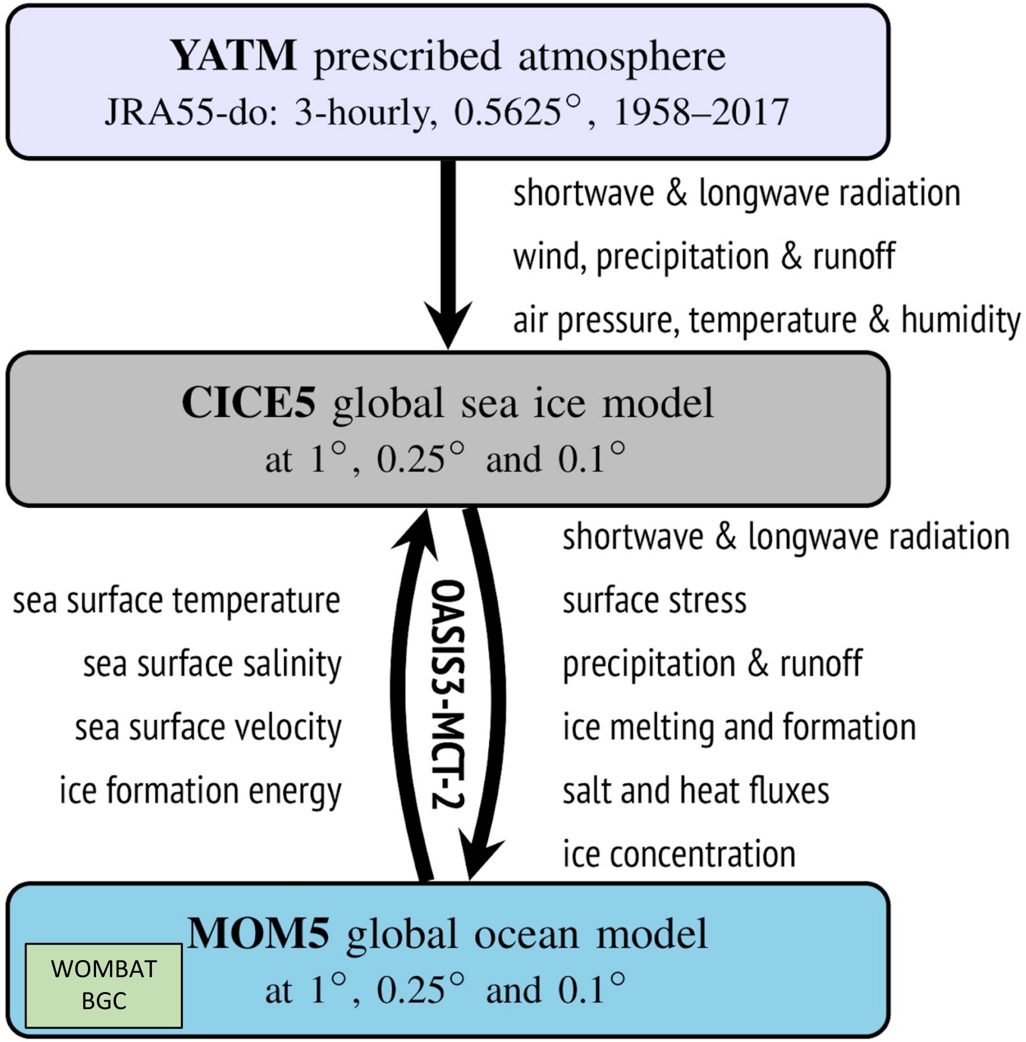
- Open source: https://github.com/COSIMA/access-om2
- Fortran, parallelised with MPI (horizontal tiling)
- Ocean model: MOM5 (GFDL, Princeton)
- Sea ice model: CICE5 (Los Alamos)
- Fields coupled between components in parallel every timestep using the OASIS3-MCT library
- Our analysis tools are also fully open-source (Python, xarray, dask, COSIMA Cookbook)

ACCESS-OM2 components

1 core

1682 cores
at 0.1°

10,461 cores
at 0.1°



Realistic prescribed surface forcing

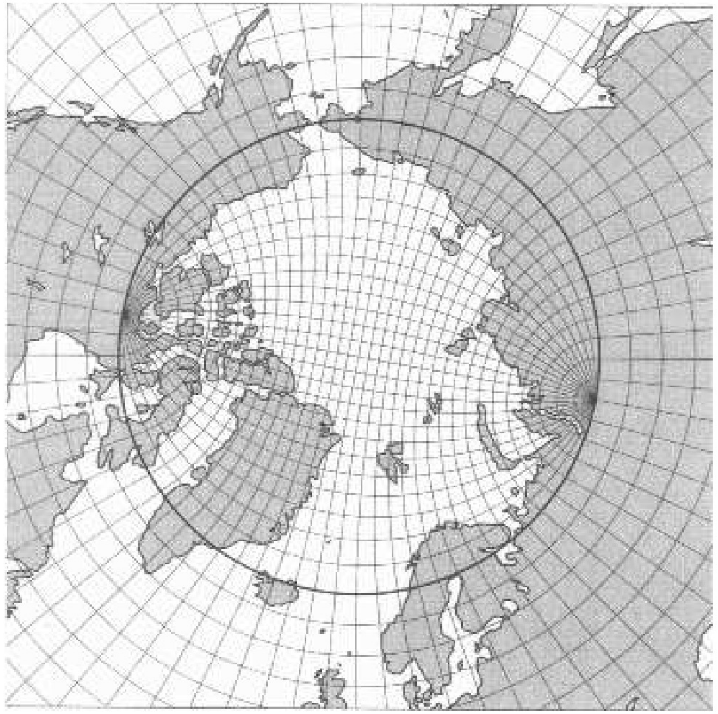
Global sea ice model: CICE5

Coupling between components every 3D timestep

Global ocean model: MOM6

The tyranny of the CFL stability condition

- Explicit timestepping, so timestep is limited by CFL: require **$\Delta t < \Delta x/c$ to avoid numerical instability**
- Doubling horizontal resolution costs 8x more
  - 4x as much data, and Δt half as long
  - 10x finer horizontal resolution costs 1000x more
- Fastest speed c is surface gravity waves (~200m/s)
  - At 0.1° this requires $\Delta t < 10$s, even though we only care about timescales of a day or more
  - Use **split timestepping**: solve 2D depth-integrated equations for surface waves with 80 sub-timesteps (Δt_{2D} ~ 8s) within each 3D timestep (Δt_{3D} ~600s)
- Use **tripolar mesh** to avoid very small Δx at north pole
  - Moves grid singularities onto land



Balaji & Liang 2007

Ocean models at different resolutions

ACCESS-OM2

- 1° horizontal grid
  360 × 300 cells, 24–111 km
- 50 z^* levels
  $\Delta z = 2.3$–220 m
- fast and cheap
  ~ 24min/yr on 252 PEs, dt=5400s
- not eddy-resolving

ACCESS-OM2-025

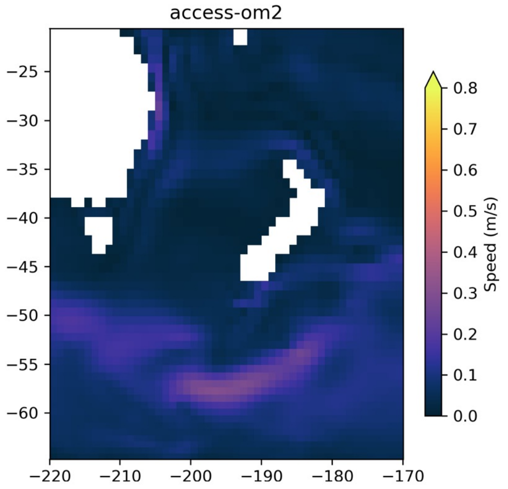
- 0.25° horizontal grid
  1440 × 1080 cells, 6.0–27.8 km
- 50 z^* levels
  $\Delta z = 2.3$–220 m
- fairly fast, less cheap
  105 min/yr on 1824 PEs, dt=1800s
- eddy "permitting"

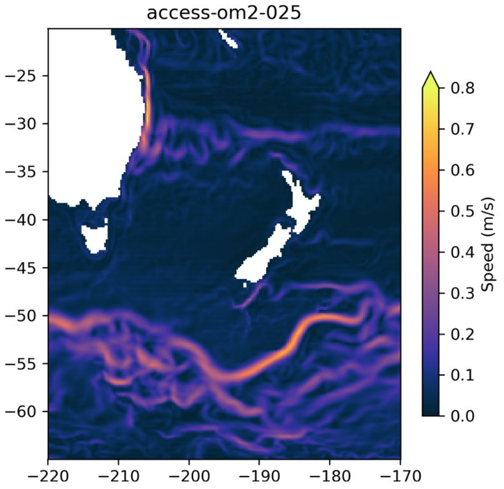
ACCESS-OM2-01

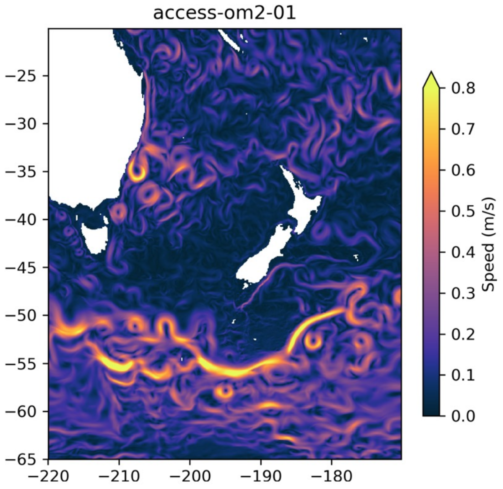
- 0.1° horizontal grid
  3600 × 2700 cells, 2.2–11.1 km
- 75 z^* levels
  $\Delta z = 1.1$–198 m
- slow, expensive
  10 hr/yr on 5744 PEs, dt=720s
- eddy-rich

Including biogeochemistry doubles this expense

Typical resolution in climate models: all turbulence is parameterised







Largest turbulent scales explicitly represented, but 500-1000x more expensive

COSIMA cosima.org.au

- Consortium for Ocean and Sea Ice Modelling in Australia
- Began in 2012; now >100 members including 2 FTE supported by two successive ARC Linkage grants since 2017
- ***Culture is key.*** *Shared models, data and tools build community.*

- **Developed ACCESS-OM2** (MOM5-CICE5) global ocean – sea ice model suite
    - Adopted in climate models (ACCESS-CM2, ACCESS-CM2-025) and Bureau of Meteorology ocean forecasts (Bluelink)
    - **550Tb of shared output data**, easily explored and analysed; >190 users
    - Underpinned [>60 papers since 2019] (>800 citations in total), >50 ongoing research projects

Cooperative modelling in COSIMA

- **Model runs are very expensive**, and output data is valuable for many research questions
  - One experiment (global 0.1° with biogeochemistry):
    - ~15Tb memory
    - >12,000 cores (>250 Cascade Lake nodes)
    - Up to ~12hr, ~350kSU per simulated year
    - Runs for months on Gadi to simulate 60 years
    - **Cost 10s of MSU**
    - **Generates ~40Tb of output**
- Few groups in Australia have resources to run these models, so in COSIMA we cooperate:
  - Piggyback: save outputs for multiple projects in each model run
  - Share all data freely on Gadi: **>550Tb of shared data so far**
- *How to make this useful to people?*

The COSIMA community

COSIMA supports a diverse set of ACCESS-OM2 users:

- Many just want to analyse existing output data, not run models
  - *so **we seek requests for diagnostics** to output from runs, and **make data discoverable and accessible** via the COSIMA Cookbook*
- Many also want to run standard executables with configuration changes
  - *so **we supply 12 standard configurations, with input files and precompiled executables***
- A few users want to make code changes
  - *so **a single git clone downloads all model source code**, with **1-line build**, and it's **easy to contribute source code improvements** via GitHub pull requests with automated tests*
- Core team of developers
  - *handle model development, maintenance, performance, porting, bug fixes, PR reviews, etc*

Data: discoverable, accessible, computable

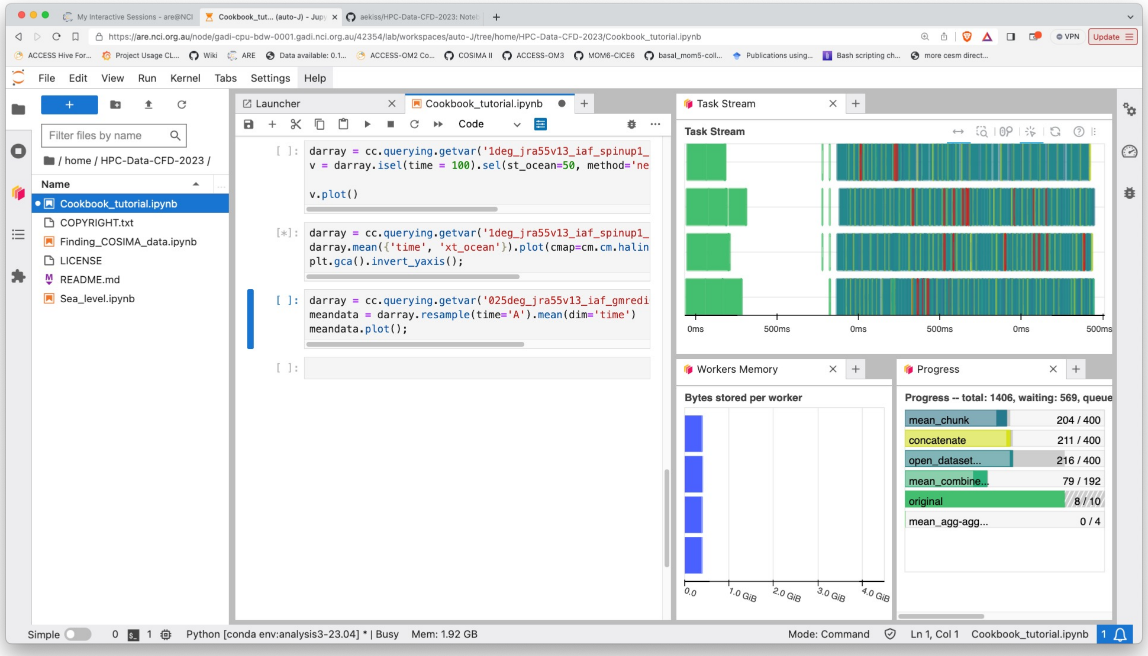
- **Discoverable**: allow non-modellers to find what they need without knowing all the technical detail
- **Accessible**: easy to obtain and understand data, with *provenance* connecting it to all details of model configuration that created it
- **Computable**: systems to enable rapid calculation on huge datasets

Analyse data in-place

- Don’t want to download 10s of Tb
- Couldn’t process it on a local machine anyway: can’t fit into memory
- So we **analyse data in-situ and in parallel** using:
  - **ARE**: NCI’s web-based graphical interface for computation on Gadi
  - **Xarray**: python package to use netCDF metadata to combine and subset individual files without loading data until it actually needed: lazy evaluation
  - **Dask**: python package to process large calculations in parallel without needing to fit it all in memory
  - **COSIMA Cookbook**: experiment database that knows where all the files live
  - **COSIMA Recipes**: community-contributed working cookbook examples

NCI's **ARE** (Australian Research Environment)

- are.nci.org.au
- web-based graphical interface for computational research
- access to NCI's Gadi supercomputer and data collections
- We use it to run **Python notebooks** in **JupyterLab** to interactively **analyse COSIMA data in-place and in parallel on Gadi**



NetCDF https://www.unidata.ucar.edu/software/netcdf/

- All COSIMA output data is in **NetCDF** files
- NetCDF is
  - A community standard, very widely used and supported in the geosciences
  - Designed for large, multidimensional numerical arrays
  - **Self-describing**: contains rich metadata
  - Compact: losslessly compressed binary
  - Portable and machine-independent
  - High performance: can be written and read in parallel
  - Supported by libraries/packages in many languages (Fortran, C, C++, Python, Julia, R, IDL, Matlab, …)
    - So we write NetCDF output from our Fortran models and open it in Python for analysis
  - Many tools to manipulate / view NetCDF (NCO, CDO, ncview, Paraview, …)
- Alternatives: Zarr, HDF, etc

Example NetCDF file, as represented by ncdump

Metadata header (small data)

- Dimension names and sizes
- Variable names, dimensions and attributes (description, units, etc.)

Data for each variable (huge)

```
netcdf oceanbgc-3d-phy-1-daily-mean-3-sigfig-5-daily-ymd_2018_12_27 {
dimensions:
	time = UNLIMITED ; // (1 currently)
	st_ocean = 75 ;
	yt_ocean = 2700 ;
	xt_ocean = 3600 ;
variables:
	float phy(time, st_ocean, yt_ocean, xt_ocean) ;
		phy:long_name = "phytoplankton" ;
		phy:units = "mmol/m^3" ;
		phy:cell_methods = "time: mean" ;
double st_ocean(st_ocean) ;
		st_ocean:long_name = "tcell zstar depth" ;
		st_ocean:units = "meters" ;
		st_ocean:cartesian_axis = "Z" ;
double time(time) ;
		time:long_name = "time" ;
		time:units = "days since 1900-01-01 00:00:00" ;
		time:cartesian_axis = "T" ;
double xt_ocean(xt_ocean) ;
		xt_ocean:long_name = "tcell longitude" ;
		xt_ocean:units = "degrees_E" ;
		xt_ocean:cartesian_axis = "X" ;
	double yt_ocean(yt_ocean) ;
		yt_ocean:long_name = "tcell latitude" ;
		yt_ocean:units = "degrees_N" ;
		yt_ocean:cartesian_axis = "Y" ;
// global attributes:
		:filename = "oceanbgc-3d-phy-1-daily-mean-5-daily-ymd_2018_12_27.nc";
data:
 phy = 0.5517578, 0.5673828, … 0.2738401 ;
 st_ocean = 0.541280746459961, 1.68073463439941, … 5709.44287109375 ;
 time = 43459.5 ;
 xt_ocean = -279.95, -279.85, … 79.9499999999949 ;
 yt_ocean = -81.1086316783548, -81.0663923239743, … 89.9788960982565 ;
}
```

Xarray

- Open-source Python package https://github.com/pydata/xarray
- Provides in-memory representation of NetCDF files
  - datasets of N-dimensional arrays with labelled dimensions
- Can create virtual datasets by concatenating and subsetting multiple NetCDF files into one object
- Importantly, this can be done **lazily**, just using metadata headers, until the last moment (e.g. saving or plotting result), when only the required subset of NetCDF data needs to be loaded

Xarray

- Rich API with extensive documentation https://docs.xarray.dev
- Apply operations over dimensions by name rather than index: `x.sum('time')`
- Select values by label instead of integer location:
  - `x.loc['2014-01-01']` or `x.sel(time='2014-01-01')`
- Mathematical operations (e.g., `x - y`) vectorize across multiple dimensions (array broadcasting) based on dimension names, not shape
- Interpolation, grouping, binning, etc.
- Flexible split-apply-combine operations: `x.groupby('time.dayofyear').mean()`
- Keep track of arbitrary metadata in the form of a Python dictionary: `x.attrs`

https://github.com/pydata/xarray

Xarray data analysis example

xarray

https://nbviewer.org/github/aekiss/HPC-Data-CFD-2023/blob/main/xarray_demo.ipynb

```
import xarray as xr
```

Lazy evaluation using NetCDF metadata only *(fast!)*

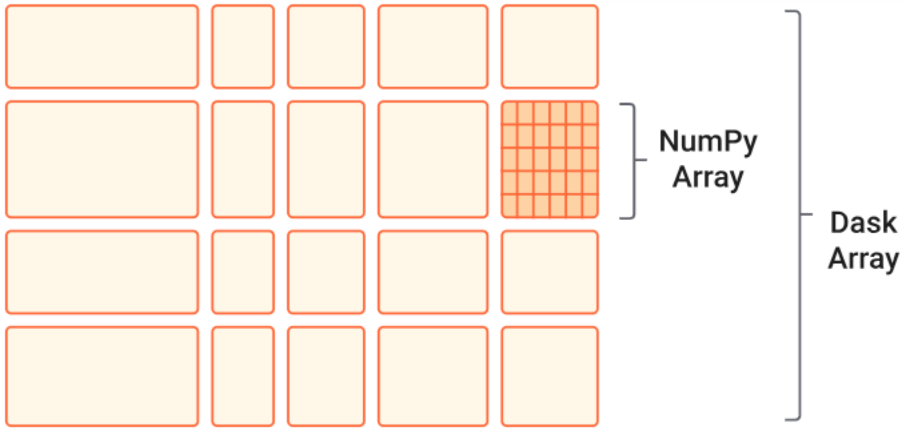
```
ds = xr.open_mfdataset('my/files/*.nc', parallel=True)   # concatenate many NetCDF files into one virtual dataset (a dict-like object containing multiple arrays)
t = ds.temp   # extract 4D temperature variable dataarray from dataset
t = t.sel(time=slice('2010-01-01', '2022-01-01'))   # subset on time axis
t = t.interp(st_ocean=100, method='linear')   # extract 2D interpolated data at 100m depth
t = t.groupby('time.month').mean()   # monthly averages
t = t.sel(month=1) - 273.15   # January mean, converted to °C
```

Actually read NetCDF data

t.plot() *# plot – access required* ***subset*** *of NetCDF data and do interpolation, averaging*

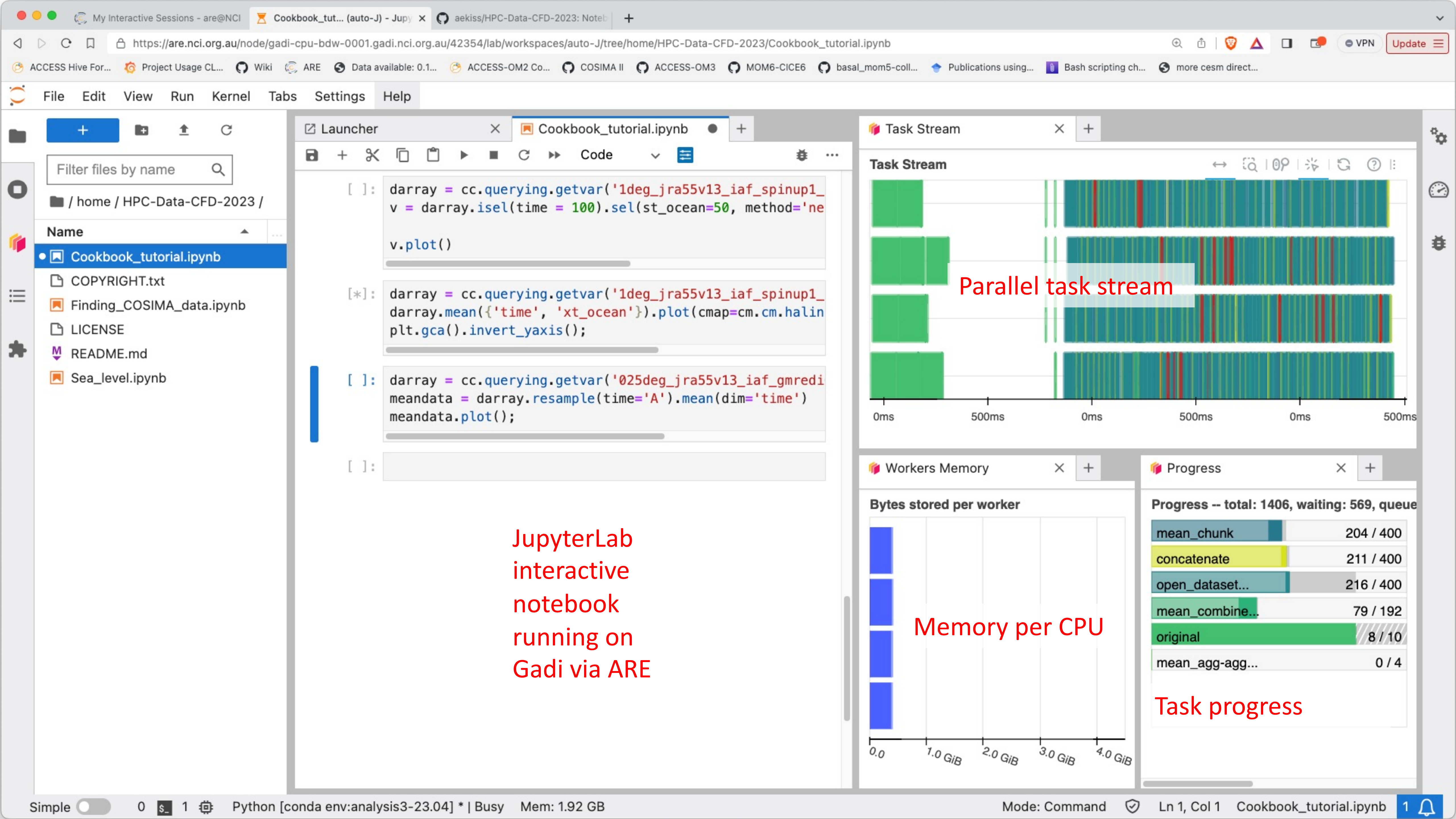
Dask: parallel calculation on data > memory

- Python package for distributed parallel tasks https://www.dask.org/
- **Dask arrays** are divided into **chunks small enough to fit into memory**



- Lazy evaluation
  - Operations queue up in series of tasks mapped over chunks
  - Computation takes place only when values are needed (e.g. to plot or output)
  - **Computation in parallel** over multiple processors or threads
- Xarray can use Dask arrays to support **parallel computations and streaming computation on datasets that don't fit into memory**

https://tutorial.dask.org/02_array.html



https://are.nci.org.au/node/gadi-cpu-bdw-0001.gadi.nci.org.au/42354/lab/workspaces/auto-J/tree/home/HPC-Data-CFD-2023/Cookbook_tutorial.ipynb
File
Edit
View
Run
Kernel
Tabs
Settings
Help
Launcher
Cookbook_tutorial.ipynb
/ home / HPC-Data-CFD-2023 /
Name
Cookbook_tutorial.ipynb
COPYRIGHT.txt
Finding_COSIMA_data.ipynb
LICENSE
README.md
Sea_level.ipynb
darray = cc.querying.getvar('1deg_jra55v13_iaf_spinup1_
v = darray.isel(time = 100).sel(st_ocean=50, method='ne
v.plot()
darray = cc.querying.getvar('1deg_jra55v13_iaf_spinup1_
darray.mean({'time', 'xt_ocean'}).plot(cmap=cm.cm.halin
plt.gca().invert_yaxis();
darray = cc.querying.getvar('025deg_jra55v13_iaf_gmredi
meandata = darray.resample(time='A').mean(dim='time')
meandata.plot();
JupyterLab interactive notebook running on Gadi via ARE
Task Stream
Parallel task stream
0ms
500ms
0ms
500ms
0ms
500ms
Workers Memory
Bytes stored per worker
Memory per CPU
0.0
1.0 GiB
2.0 GiB
3.0 GiB
4.0 GiB
Progress
Progress -- total: 1406, waiting: 569, queue
mean_chunk 204 / 400
concatenate 211 / 400
open_dataset... 216 / 400
mean_combine... 79 / 192
original 8 / 10
mean_agg-agg... 0 / 4
Task progress
Simple
Python [conda env:analysis3-23.04] * | Busy
Mem: 1.92 GB
Mode: Command
Ln 1, Col 1
Cookbook_tutorial.ipynb

What the COSIMA ocean data looks like

- **Hundreds of experiments**
- **Dozens to hundreds of output variables in each experiment**
  - u, v, w, temperature, salinity, sea surface height, 10 biogeochemical tracers, plus many online-calculated diagnostics
- Large experiments may contain **>100,000 files, tens of Tb**
  - Each experiment may have hundreds of run directories (due to queue limits)
  - Each run directory may contain hundreds of output NetCDF files

Nobody wants to deal with this complexity!

Want a simple answer to a simple request:

Give me the variable I want at the time resolution and date range I need

Don’t want to know that this data is actually spread over many files in an obscure directory tree…

COSIMA Cookbook and recipes

- Open-source Python package https://github.com/COSIMA/cosima-cookbook
- Framework for **indexing** and **querying** ocean-sea ice model output
- Uses SQLite **database of metadata from all ACCESS-OM2 experiments**, updated nightly
- *Eliminates need to know file names or directory structure*
- Ask for variable in an experiment, and it calls `xr.open_mfdataset` to return a Dask dataset concatenated from the required NetCDF files
- Also has a **GUI database explorer** tool for finding experiments and variables of interest
- **COSIMA recipes:** Shared demo notebooks and tutorials for common COSIMA Cookbook use cases – community can learn from and teach one another
  - https://github.com/COSIMA/cosima-recipes

- Alternative database system: Intake https://intake.readthedocs.io/

Credits: James Munroe, Angus Gibson, Aidan Heerdegen, Micael Oliveira and many more

Cookbook example:
loading 61 years of global daily sea surface temperature and salinity from hundreds of files in 4 lines

```
import cosima_cookbook as cc # load COSIMA cookbook package

session = cc.database.create_session() # connect to COSIMA cookbook database

SST = cc.querying.getvar('01deg_jra55v140_iaf', 'surface_pot_temp', session,
frequency='1 daily’) # load 1958-2018 daily SST

SSS = cc.querying.getvar('01deg_jra55v140_iaf', 'surface_salt', session,
frequency='1 daily’) # load 1958-2018 daily SSS
```

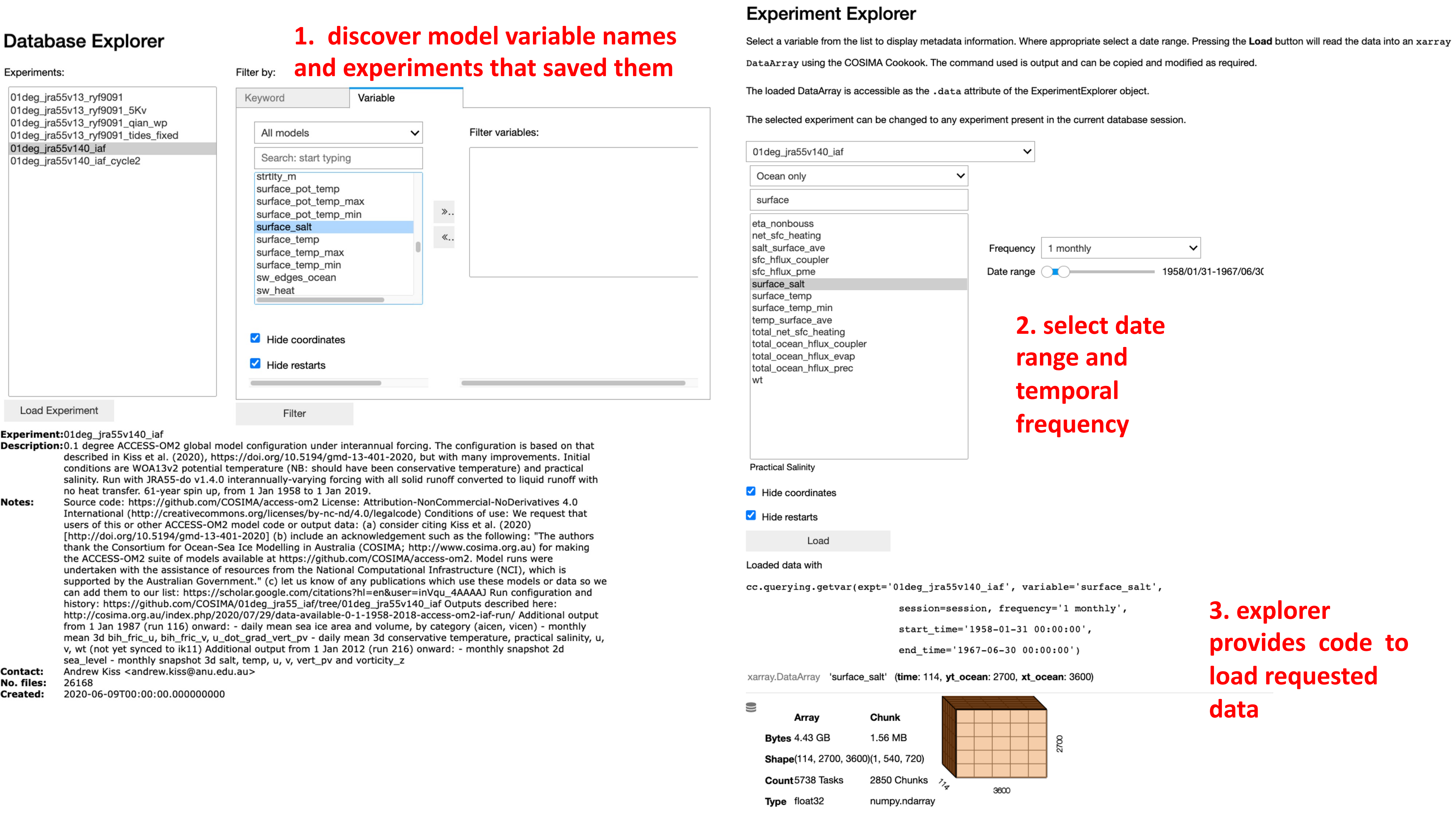
No file paths: just requires you to know the experiment and variable names

– *but how do you discover them?*

Easy COSIMA data discovery with Database Explorer

- **Database Explorer** makes it easy to discover
  - names of variables for the data you need
  - experiments that saved that variable
  - spatial & temporal resolution and time period of data
  - code needed to access the required data

Credit: Aidan Heerdegen


Database Explorer
1. discover model variable names and experiments that saved them
Experiments:
Filter by:
01deg_jra55v13_ryf9091
01deg_jra55v13_ryf9091_5Kv
01deg_jra55v13_ryf9091_qian_wp
01deg_jra55v13_ryf9091_tides_fixed
01deg_jra55v140_iaf
01deg_jra55v140_iaf_cycle2
Keyword
Variable
All models
Search: start typing
Filter variables:
strtlty_m
surface_pot_temp
surface_pot_temp_max
surface_pot_temp_min
surface_salt
surface_temp
surface_temp_max
surface_temp_min
sw_edges_ocean
sw_heat
Hide coordinates
Hide restarts
Load Experiment
Filter
Experiment:01deg_jra55v140_iaf
Description:0.1 degree ACCESS-OM2 global model configuration under interannual forcing. The configuration is based on that described in Kiss et al. (2020), https://doi.org/10.5194/gmd-13-401-2020, but with many improvements. Initial conditions are WOA13v2 potential temperature (NB: should have been conservative temperature) and practical salinity. Run with JRA55-do v1.4.0 interannually-varying forcing with all solid runoff converted to liquid runoff with no heat transfer. 61-year spin up, from 1 Jan 1958 to 1 Jan 2019.
Notes: Source code: https://github.com/COSIMA/access-om2 License: Attribution-NonCommercial-NoDerivatives 4.0 International (http://creativecommons.org/licenses/by-nc-nd/4.0/legalcode) Conditions of use: We request that users of this or other ACCESS-OM2 model code or output data: (a) consider citing Kiss et al. (2020) [http://doi.org/10.5194/gmd-13-401-2020] (b) include an acknowledgement such as the following: "The authors thank the Consortium for Ocean-Sea Ice Modelling in Australia (COSIMA; http://www.cosima.org.au) for making the ACCESS-OM2 suite of models available at https://github.com/COSIMA/access-om2. Model runs were undertaken with the assistance of resources from the National Computational Infrastructure (NCI), which is supported by the Australian Government." (c) let us know of any publications which use these models or data so we can add them to our list: https://scholar.google.com/citations?hl=en&user=inVqu_4AAAAJ Run configuration and history: https://github.com/COSIMA/01deg_jra55_iaf/tree/01deg_jra55v140_iaf Outputs described here: http://cosima.org.au/index.php/2020/07/29/data-available-0-1-1958-2018-access-om2-iaf-run/ Additional output from 1 Jan 1987 (run 116) onward: - daily mean sea ice area and volume, by category (aicen, vicen) - monthly mean 3d bih_fric_u, bih_fric_v, u_dot_grad_vert_pv - daily mean 3d conservative temperature, practical salinity, u, v, wt (not yet synced to ik11) Additional output from 1 Jan 2012 (run 216) onward: - monthly snapshot 2d sea_level - monthly snapshot 3d salt, temp, u, v, vert_pv and vorticity_z
Contact: Andrew Kiss <andrew.kiss@anu.edu.au>
No. files: 26168
Created: 2020-06-09T00:00:00.000000000
Experiment Explorer
Select a variable from the list to display metadata information. Where appropriate select a date range. Pressing the Load button will read the data into an xarray DataArray using the COSIMA Cookook. The command used is output and can be copied and modified as required.
The loaded DataArray is accessible as the .data attribute of the ExperimentExplorer object.
The selected experiment can be changed to any experiment present in the current database session.
01deg_jra55v140_iaf
Ocean only
surface
eta_nonbouss
net_sfc_heating
salt_surface_ave
sfc_hflux_coupler
sfc_hflux_pme
surface_salt
surface_temp
surface_temp_min
temp_surface_ave
total_net_sfc_heating
total_ocean_hflux_coupler
total_ocean_hflux_evap
total_ocean_hflux_prec
wt
Frequency
1 monthly
Date range
1958/01/31-1967/06/30
2. select date range and temporal frequency
Practical Salinity
Hide coordinates
Hide restarts
Load
Loaded data with
cc.querying.getvar(expt='01deg_jra55v140_iaf', variable='surface_salt',
session=session, frequency='1 monthly',
start_time='1958-01-31 00:00:00',
end_time='1967-06-30 00:00:00')
3. explorer provides code to load requested data
xarray.DataArray 'surface_salt' (time: 114, yt_ocean: 2700, xt_ocean: 3600)
Array
Chunk
Bytes 4.43 GB
1.56 MB
Shape(114, 2700, 3600)(1, 540, 720)
Count 5738 Tasks
2850 Chunks
Type float32
numpy.ndarray
2700
3600
114

Thursday's tutorial

- Material available at https://github.com/aekiss/HPC-Data-CFD-2023 (will be updated between now and Thursday)
- **BEFORE attending** you need
  - An NCI account
  - Membership of a project with a compute allocation
  - Request membership of hh5, ik11 and cj50 for the tutorial

Take-away messages

You can analyse huge datasets easily if you use the right tools

- **Move the computation to the data** (e.g. use ARE)
- Use a **self-describing file format** (e.g. NetCDF, Zarr, HDF)
- **Use file metadata** to select and process data subsets (e.g. xarray)
- Use **parallel file access and processing** (e.g. dask)
- Access your data via a **database** (e.g. COSIMA Cookbook, Intake)

Questions?