Numerical Weather Prediction (NWP)

Theories

Numerical Weather Prediction

Skamarock 2012
MPAS-A seminal paper

Governing Partial Differential Equations,

- Fluid dynamics
- Thermodynamics
- Moisture, source & sink terms

=> next slides

$$\frac{\partial \mathbf{V}_H}{\partial t} = -\frac{\rho_d}{\rho_m}\left[\nabla_\zeta\left(\frac{p}{\zeta_z}\right) - \frac{\partial \mathbf{z}_H p}{\partial \zeta}\right] - \eta \mathbf{k} \times \mathbf{V}_H$$
$$- \boldsymbol{v}_H \nabla_\zeta \cdot \mathbf{V} - \frac{\partial \Omega \boldsymbol{v}_H}{\partial \zeta} - \rho_d \nabla_\zeta K$$
$$- eW \cos\alpha_r - \frac{\boldsymbol{v}_H W}{r_e} + \mathbf{F}_{\mathbf{V}_H}, \qquad (3)$$

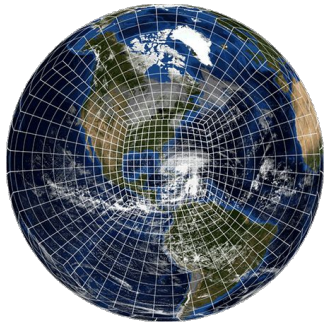
$$\frac{\partial W}{\partial t} = -\frac{\rho_d}{\rho_m}\left[\frac{\partial p}{\partial \zeta} + g\tilde{\rho}_m\right] - (\nabla \cdot \boldsymbol{v} W)_\zeta + \frac{uU + vV}{r_e}$$
$$+ e(U \cos\alpha_r - V \sin\alpha_r) + F_W, \qquad (4)$$

$$\frac{\partial \Theta_m}{\partial t} = -(\nabla \cdot \mathbf{V}\theta_m)_\zeta + F_{\Theta_m}, \qquad (5)$$

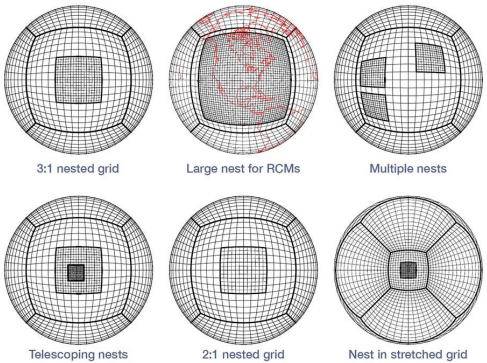
$$\frac{\partial \tilde{\rho}_d}{\partial t} = -(\nabla \cdot \mathbf{V})_\zeta, \quad \text{and} \qquad (6)$$

$$\frac{\partial Q_j}{\partial t} = -(\nabla \cdot \mathbf{V} q_j)_\zeta + F_{Q_j}. \qquad (7)$$

Many models and gridings for solving PDEs:
Just pick an example:



GFS-FV3 model
https://www.gfdl.noaa.gov/fv3/



Governing Equations

Rate of change of something	=	terms related to itself and some other things in neighbouring positions
Derivatives with respect to time	has a relation to	Spatial operators Divergence Gradient Curl

Systems of equations - variables are "coupled".

Dry atmosphere: V, W, theta, rho

- Horizontal wind
- Vertical wind
- Potential temperature
- Density

Wet atmosphere: add qv, qc, qi, qr, …

- Mixing ratio of moisture in different states

Relations:

Newton's classical mechanics:

- Conservation of momentum
- Conservation of mass (of dry air, moisture of different states)

Thermodynamics:

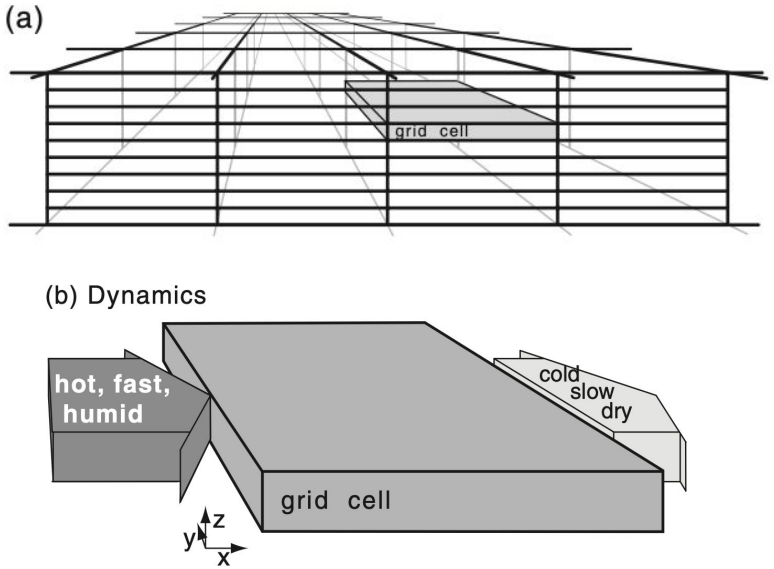
- Ideal gas law
- Conservation of energy (between heat and kinetic energy)

… and external forcing

Rectangular Grid - Dynamics

Figure 20.3
(a) The forecast **domain** *(the portion of atmosphere we wish to forecast) is split into discrete grid cells, such as the shaded one. The 3-D grid cells are relatively thin, with sizes on the order of 10s m in the vertical, and 10s km in the horizontal.*

(b) Enlargement of the shaded grid cell, illustrating one **dynamics** *process (advection in the x-direction). Namely, the resolved U wind is blowing in hot, fast, humid air from the upwind neighboring grid cell, and is blowing out colder, slower, drier air into the downwind neighboring cell. Simultaneously, advection could be occurring by the V and W components of wind (not shown). Not shown are other resolved forcings, such as Coriolis and pressure-gradient forces.*



Stull 2017 Practical Meteorology: An Algebra-based Survey of Atmospheric Science
https://www.eoas.ubc.ca/books/Practical_Meteorology/
Chapter 20 Numerical Weather Prediction (NWP)

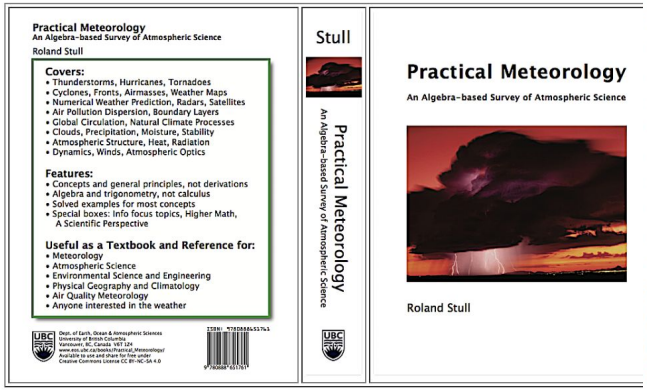
By the way, a great textbook

https://www.eoas.ubc.ca/books/Practical_Meteorology/

Practical Meteorology:

An Algebra-based Survey of Atmospheric Science

Available to use and share for free under a Creative Commons License.
(Click on chapter links below to get pdfs of individual chapters.
See bottom of this web page for license details.)



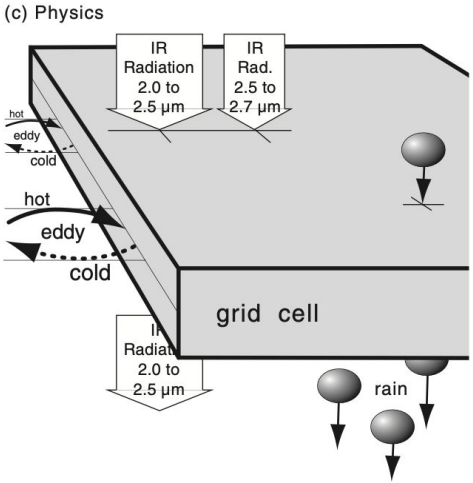
Cover and spine photo copyright © by Warren Faidley / weatherstock.com
Used with permission.

Stull, R., 2017: "Practical Meteorology: An Algebra-based Survey of Atmospheric Science" -version 1.02b. Univ. of British Columbia. 940 pages. isbn 978-0-88865-283-6 .

Acknowledgement - Credit: Roland Stull.

https://www.eoas.ubc.ca/books/Practical_Meteorology/
Last updated Nov 2018 by R. Stull

Physics Parameterization for unresolved physical processes



Source & Sink terms,
or called
Forcing terms,
in the
governing PDEs

Example:

Table 1: Physics Parameterization Schemes used in the experiments

Physics	Scheme
Convection	Scale-aware New Tiedtke (MPAS-A v6.x-openacc branch)
Microphysics	WSM6 (MPAS-A 7.0 = WRF 4.1)
Land surface	Noah (MPAS-A 7.0 = WRF 4.0.3)
Boundary layer	YSU (MPAS-A 7.0 = WRF 4.0.3)
Surface layer	Monin-Obukhov (MPAS-A 7.0 = WRF 4.0.3)
Radiation	RRTMG (MPAS-A 7.0 = WRF 3.8.1)
Cloud fraction	Xu and Randall (MPAS-A 7.0 = WRF 3.8.1)
Gravity wave drag	YSU (MPAS-A 7.0 = WRF 4.0.3)

(c) Further enlargement, illustrating **physics** *such as turbulence, radiation, and precipitation. Turbulence is causing a net heat flux into the left side of the grid cell in this example, even though the turbulence has no net wind (i.e., the wind-gust arrows moving air into the grid cell are balanced by gusts moving the same amount of air out of the grid cell). Two of the many radiation bands are shown, where infrared (IR) wavelengths in the 2.0 to 2.5 μm "window" band shine through the grid cell, while wavelengths in the 2.5 to 2.7 band are absorbed by water vapor and carbon dioxide (see the Satellites & Radar chapter), causing warming in the grid cell. Some liquid water is falling into the top of the grid cell from the cell above, but even more is falling out the bottom into the grid cell below, suggesting a removal of water and net latent heating due to condensation.*

We will revisit physics parameterization later.

Introducing the MPAS-A model

https://mpas-dev.github.io

MPAS
Model for Prediction Across Scales

MPAS Home

Overview
MPAS-Atmosphere
MPAS-Albany Land Ice
MPAS-Ocean
MPAS-Seaice
Data Assimilation
Publications
Presentations

Download
MPAS-Atmosphere download
MPAS-Albany Land Ice download
MPAS-Ocean download
MPAS-Seaice download

Resources
License Information
Wiki
Bug Tracker
Mailing Lists
MPAS Developers Guide
MPAS Mesh Specification Document

MPAS Atmosphere

Overview

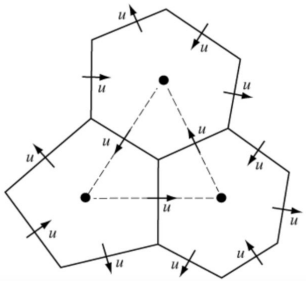
The atmospheric component of MPAS, as with all MPAS components, uses an unstructured centroidal Voronoi mesh (grid, or tessellation) and C-grid staggering of the state variables as the basis for the horizontal discretization in the fluid-flow solver. The unstructured variable resolution meshes can be generated having smoothly-varying mesh transitions (see the figure to the right); we believe that this capability will ameliorate many issues associated with the traditional mesh refinement strategy of one-way and two-way grid nesting where the transitions are abrupt. Using the flexibility of the MPAS meshes, we are working towards applications in high-resolution numerical weather prediction (NWP) and regional climate, in addition to global uniform-resolution NWP and climate applications.

The MPAS atmosphere consists of an atmospheric fluid-flow solver (the *dynamical core*) and a subset of the Advanced Research WRF (ARW) model atmospheric physics. Work is underway to port the MPAS atmospheric dynamical core to the Community Atmosphere Model (CAM) in the Community Earth Systems Model (CESM), which will provide coupling between MPAS Ocean and MPAS Atmosphere and coupling to the CAM physics and other components of the CESM system. Work is also progressing on porting the National Centers for Environmental Prediction (NCEP) Global Forecast System (GFS) atmospheric physics to MPAS.

A variable resolution MPAS Voronoi mesh

Dynamical Core

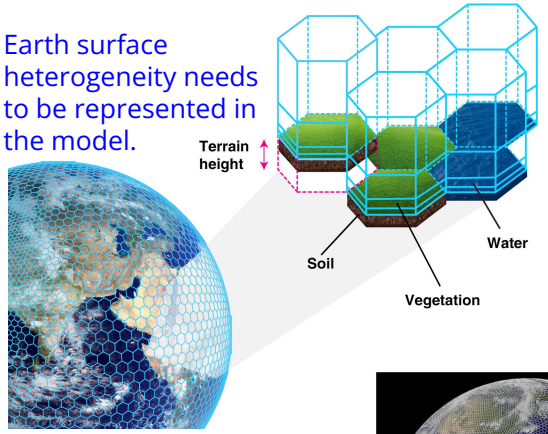
The MPAS atmospheric dynamical core solves the fully compressible nonhydrostatic equations of motion. The horizontal Voronoi mesh, depicted to the right, uses a C-grid staggering of the state variables; the horizontal velocity u is defined as the normal velocity on Voronoi cell faces while the other state variables are defined at the cell centers. The dual of the Voronoi mesh is the triangular mesh shown in dashed lines in the figure. The variable resolution meshes a predominantly comprised of hexagons, but pentagons and septagons are occasionally present. The primary advances associated with the C-grid-staggered Voronoi mesh can be found in Thuburn et al JCP (2009) and Ringler et al JCP (2010).



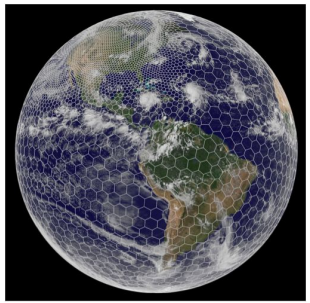
A description of the compressible nonhydrostatic atmospheric solver can be found in Skamarock et al MWR (2012). The fully compressible nonhydrostatic equations are cast in terms of a geometric-height vertical coordinate, and the solver makes use of a split-explicit time integration scheme that is described in Klemp et al MWR (2007). The time-integration scheme employs a 3rd-order Runge-Kutta method, and large time step, for

Vertical layers

Earth surface heterogeneity needs to be represented in the model.



Variable-resolution



NCAR MPAS-Atmosphere model
http://mpas-dev.github.io/

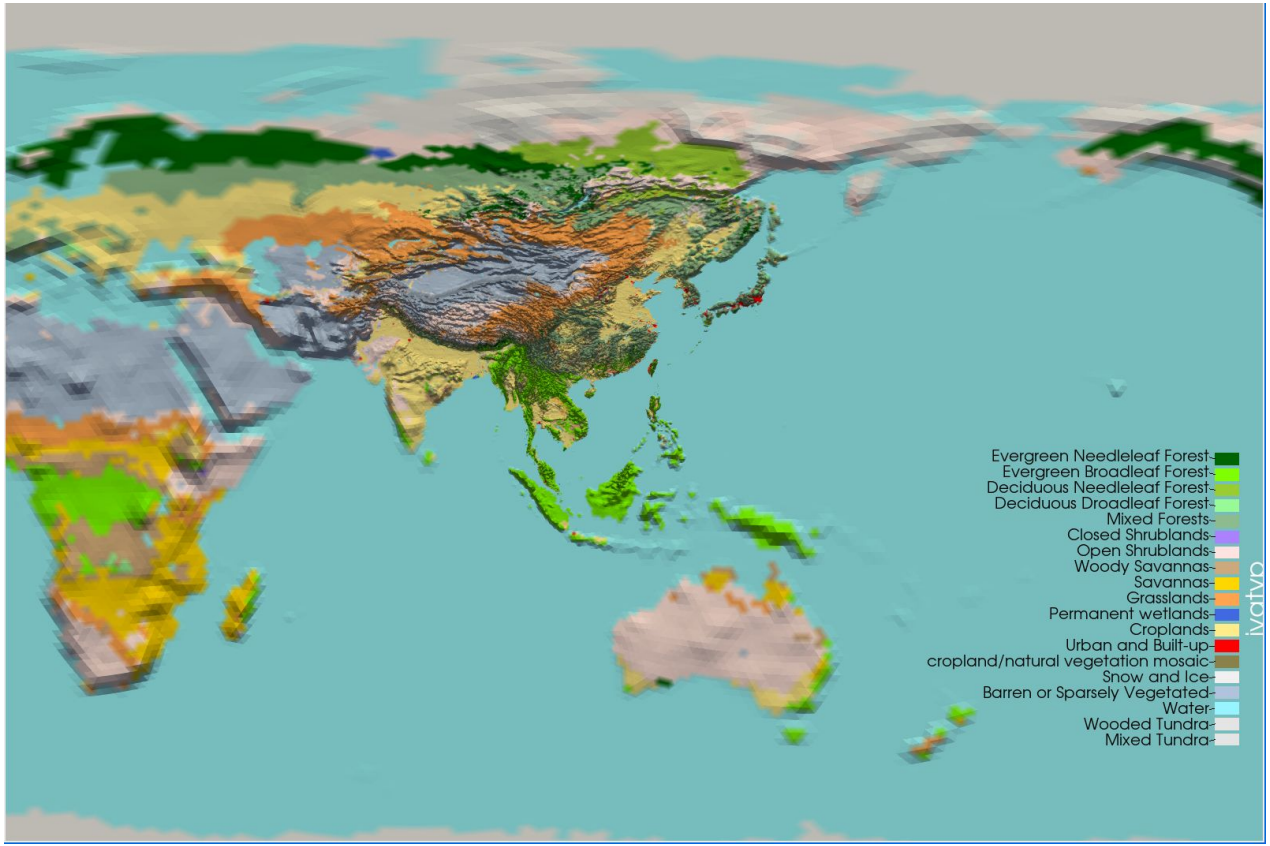
High resolution for cities / provinces

E.g. What fine geographical features needs to be represented for 5-day ahead prediction for Hong Kong?

- Terrain in south China?
- Land-use in south China?

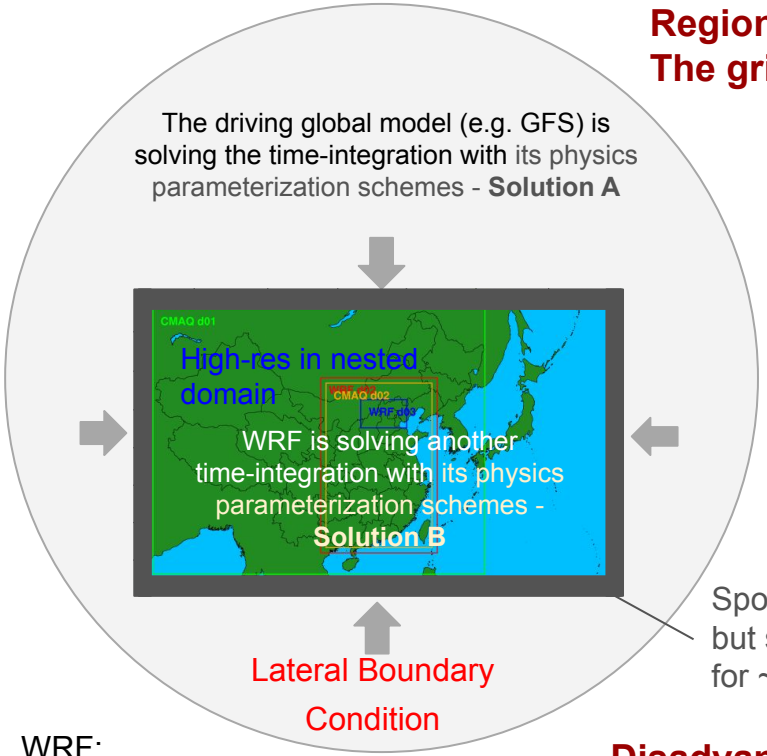
- Terrain of Indochinese Peninsula?
- Terrain of Taiwan / N. Philippines (Tropical cyclone passage)?

- Farther away less important?



Benefit of using variable-resolution unstructured grid for atmospheric simulation

Regional vs global model



Regional model:
The grid covers a region only

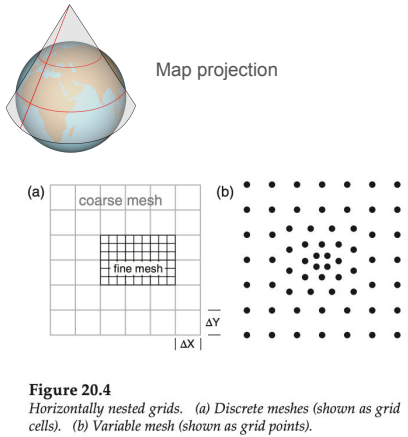


Figure 20.4
Horizontally nested grids. (a) Discrete meshes (shown as grid cells). (b) Variable mesh (shown as grid points).

Sponge layers help, but simulation valid for ~72 hours only

WRF:

- A popular regional model

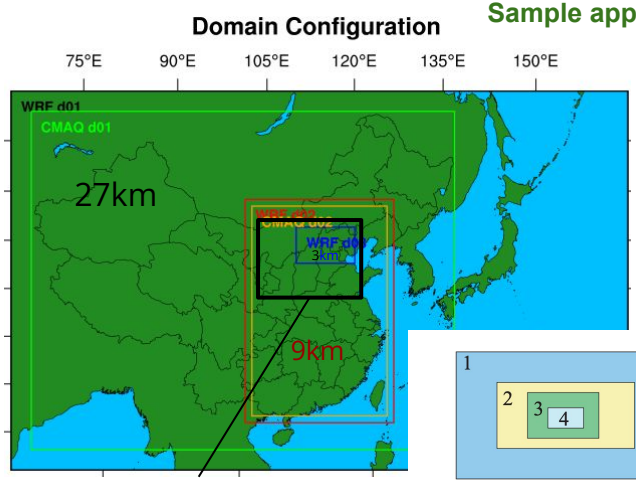
Disadvantage:
Limited forecast hours

MPAS-A:

- A single **global model** with refinement regions
- A set of **consistent**, **scale-aware (improving)** physics parameterization schemes.
- **Longer period with valid forecast**

Inflexibility of domain nesting

Sample application: NWP driving Air Quality model CMAQ



E.g. to resolve mountains (not aligned with N-S, W-E direction) to predict a better timing of cold front arrival

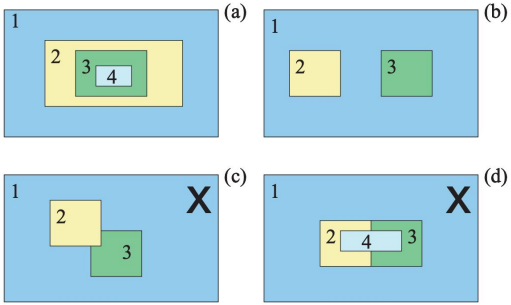


Figure 7.2: Various nest configurations for multiple grids. (a) Telescoping nests. (b) Nests at the same level with respect to a parent grid. (c) Overlapping grids: not allowed with feedback activated. (d) Inner-most grid has more than one parent grid: not allowed

Constrained refinement domain:
- rectangular,
- aligned with N-S, W-E direction

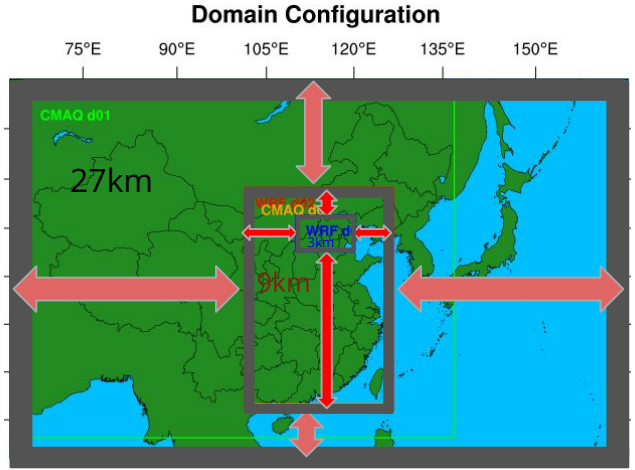
Costly for enlarged refinement.

WRF Nesting

Skamarock, W. C., Klemp, J. B., Dudhia, J., Gill, D. O., Liu, Z., Berner, J., … Huang, X. -yu. (2021). A Description of the Advanced Research WRF Model Version 4.3 (No. NCAR/TN-556+STR). doi:10.5065/1dfh-6p97

MPAS variable-resolution unstructured mesh -> later slide

Domain nesting vs resolution transition

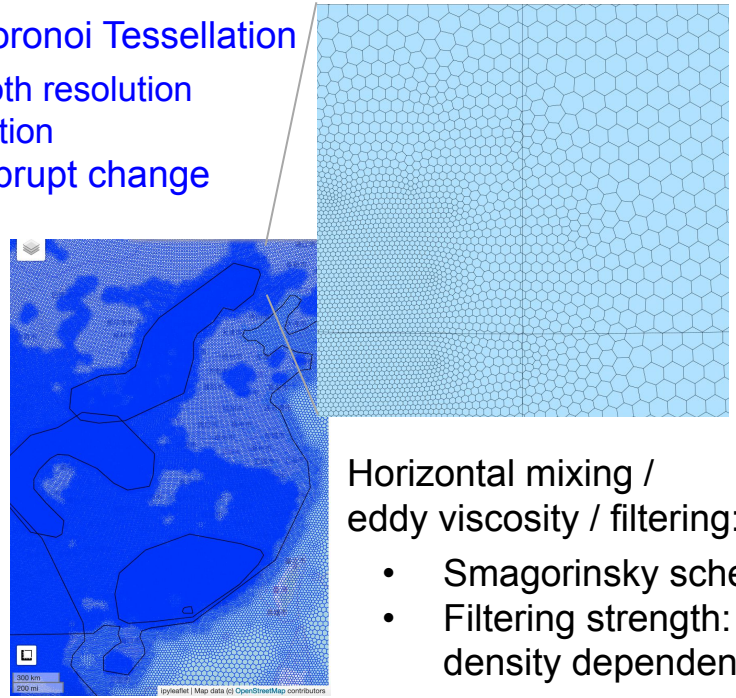


WRF nesting's Lateral Boundary

- Abrupt change of resolution
- Reflection of wave handed by sponge layers
- Best practice: far away from region of interest.

MPAS-A Voronoi Tessellation

- Smooth resolution transition
- No abrupt change



Horizontal mixing / eddy viscosity / filtering:

- Smagorinsky scheme
- Filtering strength: mesh density dependent

Sample application:
Nesting for densely populated area in China

Hello Finite Volume Method

My favorite Finite Difference Method?

- Unstructured grid
  - No more Cartesian (x,y,z)
  - How do I do finite difference?
- Coordinates for sphere
  - Zonal (W-E direction)
  - Meridional (S-N direction)

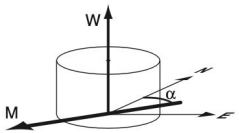


Figure 1.3
Notation used in cylindrical coordinates for velocity.

https://www2.mmm.ucar.edu/projects/mpas/tutorial/Boulder2019/slides/07.MPAS_solver.pdf
Slide 2

MPAS
Model for Prediction Across Scales

MPAS Nonhydrostatic Atmospheric Solver

Nonhydrostatic formulation

Equations

- Prognostic equations for coupled variables.
- Generalized height coordinate.
- Horizontally vector invariant eqn set.
- Continuity equation for dry air mass.
- Thermodynamic equation for coupled potential temperature.

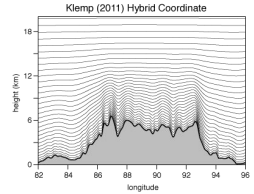
Time integration scheme

As in Advanced Research WRF - Split-explicit Runge-Kutta (3rd order), with additional splitting options.

Variables:
$(U,V,\Omega,\Theta,Q_j) = \tilde{\rho}_d \cdot (u,v,\dot{\eta},\theta,q_j)$

Vertical coordinate:
$z = \zeta + A(\zeta)\, h_s(x,y,\zeta)$

Prognostic equations:



$$\frac{\partial \mathbf{V}_H}{\partial t} = -\frac{\rho_d}{\rho_m}\left[\nabla_\zeta\left(\frac{p}{\zeta_z}\right) - \frac{\partial \mathbf{z}_H p}{\partial \zeta}\right] - \eta\,\mathbf{k}\times\mathbf{V}_H - \mathbf{v}_H\nabla_\zeta\cdot\mathbf{V} - \frac{\partial \Omega \mathbf{v}_H}{\partial \zeta} - \rho_d \nabla_\zeta K + \mathbf{F}_{V_H}$$

$$\frac{\partial W}{\partial t} = -\frac{\rho_d}{\rho_m}\left[\frac{\partial p}{\partial \zeta} + g\tilde{\rho}_m\right] - \left(\nabla\cdot\mathbf{v}\,W\right)_\zeta + F_W$$

$$\frac{\partial \Theta_m}{\partial t} = -\left(\nabla\cdot\mathbf{V}\,\theta_m\right)_\zeta + F_{\Theta_m}$$

$$\frac{\partial \rho_d}{\partial t} = -\left(\nabla\cdot\mathbf{V}\right)_\zeta$$

$$\frac{\partial Q_j}{\partial t} = -\left(\nabla\cdot\mathbf{V}\,q_j\right)_\zeta + F_{Q_j}$$

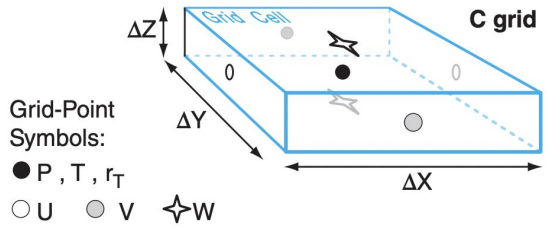
Diagnostics and definitions:

$$\theta_m = \theta[1+(R_v/R_d)q_v] \qquad p = p_0\left(\frac{R_d\zeta_z\Theta_m}{p_0}\right)^\gamma$$

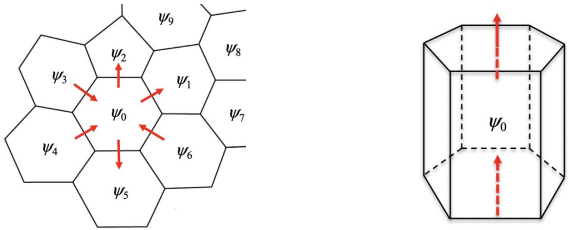
$$\frac{\rho_m}{\rho_d} = 1 + q_v + q_c + q_r + \ldots$$

Scalar at cell center, wind on cell face!

On rectangular grid, called "C-grid staggering"



On unstructured grid, this is natural!



MPAS
Model for Prediction Across Scales

MPAS Nonhydrostatic Atmospheric Solver

Prognostic equations:

$$\frac{\partial \mathbf{V}_H}{\partial t} = -\frac{\rho_d}{\rho_m}\left[\nabla_\zeta\left(\frac{p}{\zeta_z}\right) + \frac{\partial \mathbf{z}_H p}{\partial \zeta}\right] - \eta\,\mathbf{k}\times\mathbf{V}_H - \mathbf{v}_H\nabla_\zeta\cdot\mathbf{V} - \frac{\partial \Omega \mathbf{v}_H}{\partial \zeta} - \rho_d\nabla_\zeta K + \mathbf{F}_{V_H}$$

$$\frac{\partial W}{\partial t} = -\frac{\rho_d}{\rho_m}\left[\frac{\partial p}{\partial \zeta} + g\tilde{\rho}_m\right] - \left(\nabla\cdot\mathbf{v}W\right)_\zeta + F_W$$

$$\frac{\partial \Theta_m}{\partial t} = -\left(\nabla\cdot\mathbf{V}\,\theta_m\right)_\zeta + F_{\Theta_m}$$

$$\frac{\partial \rho_d}{\partial t} = -\left(\nabla\cdot\mathbf{V}\right)_\zeta$$

$$\frac{\partial Q_j}{\partial t} = -\left(\nabla\cdot\mathbf{V}\,q_j\right)_\zeta + F_{Q_j}$$

(1) Gradient operators
(2) Flux divergence operators
(3) Nonlinear Coriolis term

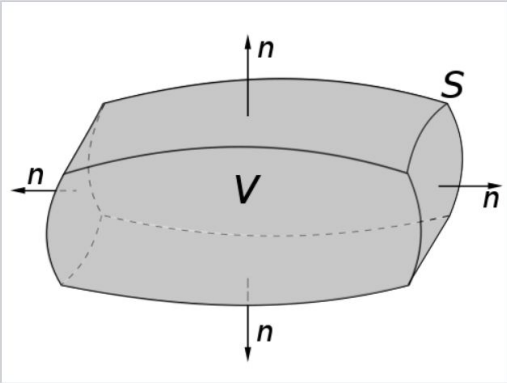
Divergence theorem

Mathematical statement [edit]

Suppose V is a subset of $\mathbb{R}^n$ (in the case of $n = 3$, V represents a volume in three-dimensional space) which is compact and has a piecewise smooth boundary S (also indicated with $\partial V = S$). If $\mathbf{F}$ is a continuously differentiable vector field defined on a neighborhood of V, then:[4][5]

$$\iiint_V (\nabla \cdot \mathbf{F})\, \mathrm{d}V = \oiint_S (\mathbf{F} \cdot \hat{\mathbf{n}})\, \mathrm{d}S.$$

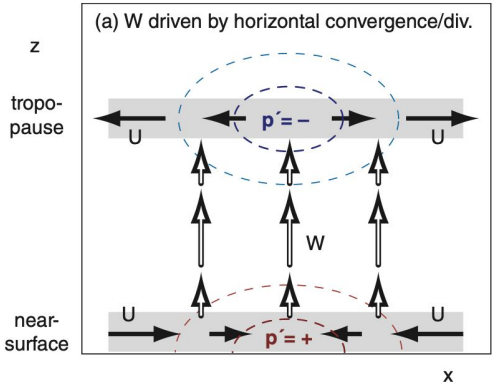
The left side is a volume integral over the volume V, the right side is the surface integral over the boundary of the volume V. The closed manifold ∂V is oriented by outward-pointing normals, and $\hat{\mathbf{n}}$ is the outward pointing unit normal at each point on the boundary ∂V. ($\mathrm{d}\mathbf{S}$ may be used as a shorthand for $\mathbf{n}\mathrm{d}S$.) In terms of the intuitive description above, the left-hand side of the equation represents the total of the sources in the volume V, and the right-hand side represents the total flow across the boundary S.



A region V bounded by the surface $S = \partial V$ with the surface normal n

https://en.wikipedia.org/wiki/Divergence_theorem

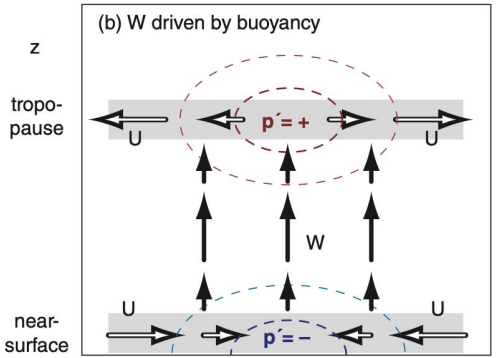
Mass continuity



11.4.1.1. Horizontal Convergence/Divergence

If external forcings cause air near the ground to converge horizontally, then air molecules accumulate. As density ρ increases according to eq. (10.60), the ideal gas law tells us that p' will also become positive (Fig. 11.16a). Non-zero p' implies that a pressure gradient exists, which will drive winds.

Positive p' does two things: it (1) decelerates the air that was converging horizontally, and (2) accelerates air vertically in the column. Thus, the pressure perturbation causes **mass continuity** (horizontal inflow near the ground balances vertical outflow).

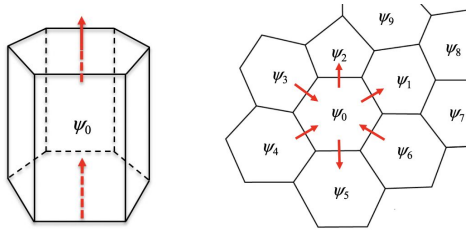


11.4.1.2. Buoyant Forcings

For a different scenario, suppose air in a column is positively buoyant, such as in a thunderstorm where water-vapor condensation releases lots of latent heat. This vertical buoyant force creates upward motion (i.e., warm air rises, as in Fig. 11.16b).

As air in the thunderstorm column moves away from the ground, it removes air molecules and lowers the density and the pressure; hence, p' is negative near the ground. This suction under the updraft causes air near the ground to horizontally converge, thereby conserving mass.

Figure 11.16



Stull 2017 Practical Meteorology
https://www.eoas.ubc.ca/books/Practical_Meteorology/
Chapter 11 General Circulation

For conservative physical quantities

Partial Differential Equations:

- Derivative w.r.t time ~ divergence term

Finite Volume Method:

- Flux: How much quantity goes out (and in) to the 3D grid cell
- => net flux
- => rate of increase of the quantity inside the 3D grid cell

https://www2.mmm.ucar.edu/projects/mpas/tutorial/Boulder2019/slides/07.MPAS_solver.pdf
Slide 18

MPAS
Model for Prediction Across Scales

Operators on the Voronoi Mesh

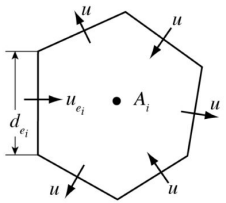
Flux divergence and transport

Transport equation, conservative form:
$$\frac{\partial(\rho\psi)}{\partial t} = -\nabla \cdot \mathbf{V}(\rho\psi)$$

Finite-Volume formulation, Integrate over cell:
$$\int_D \left[\frac{\partial}{\partial t}(\rho\psi) = -\nabla \cdot \mathbf{V}(\rho\psi)\right] dV$$

Apply divergence theorem:
$$\frac{\partial(\overline{\rho\psi})}{\partial t} = -\frac{1}{V}\int_\Sigma (\rho\psi)\,\mathbf{V}\cdot\mathbf{n}\; d\sigma$$

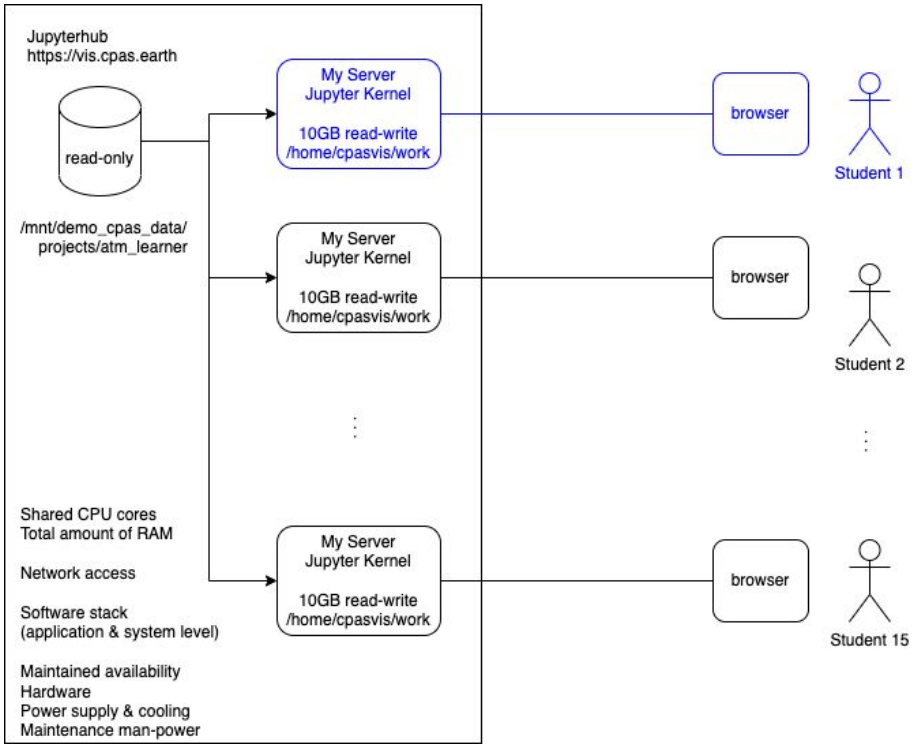
Discretize in time and space:
$$(\rho\psi)_i^{t+\Delta t} = (\rho\psi)_i^t - \Delta t \frac{1}{A_i}\sum_{n_{e_i}} d_{e_i}\overline{(\rho\mathbf{V}\cdot\mathbf{n}_{e_i})\psi}$$

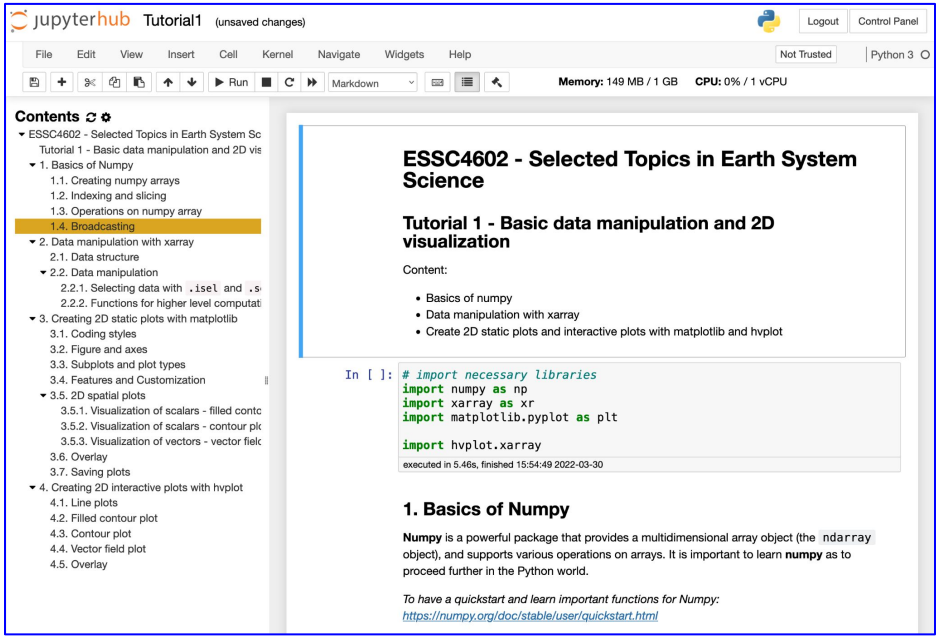


Velocity divergence operator is 2nd-order accurate for edge-centered velocities.

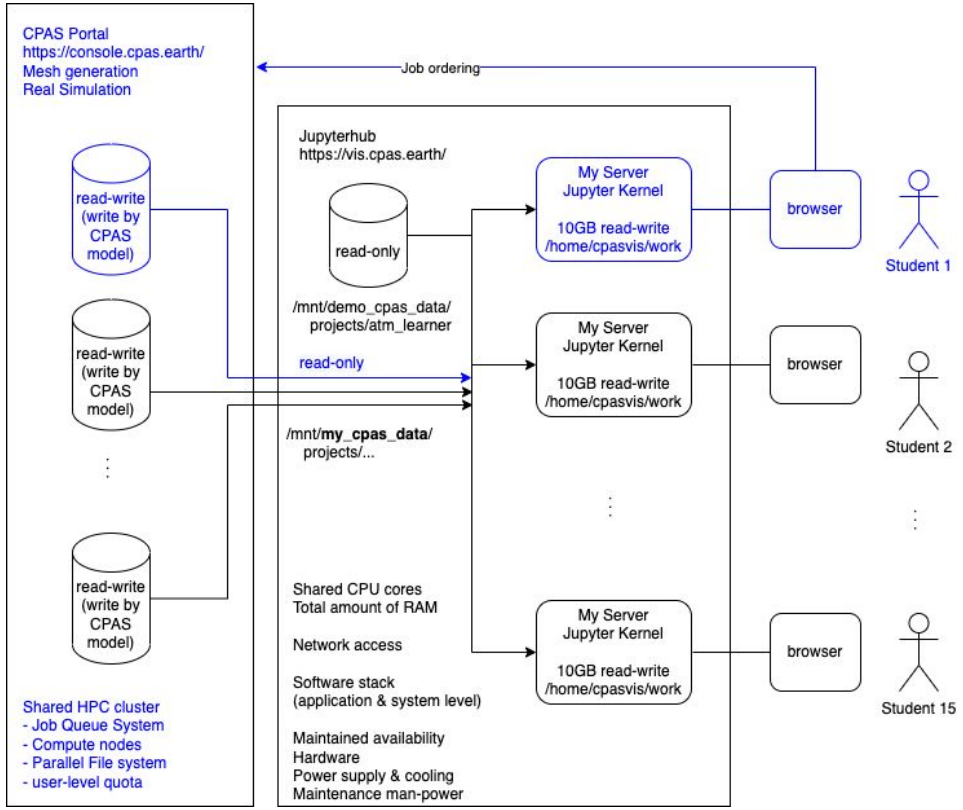
Online Programming Framework

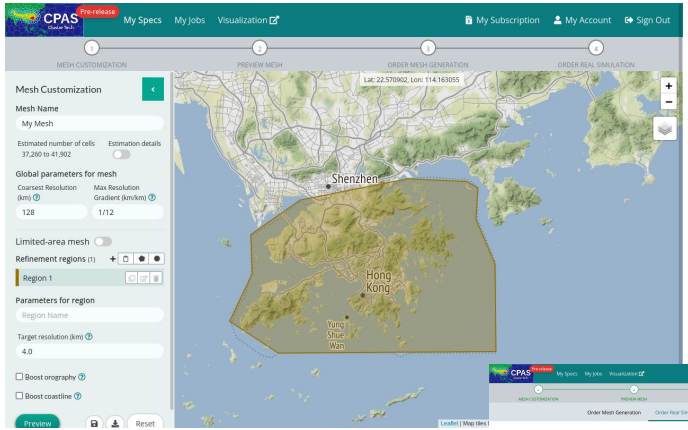
Your learning infrastructure - CPAS Visualization system





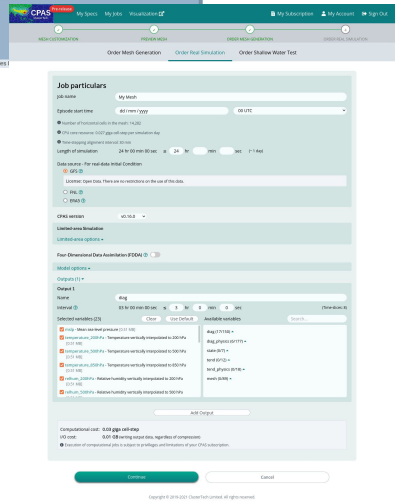
A fuller picture of CPAS





Mesh generation

Real Simulation



Plain Python?

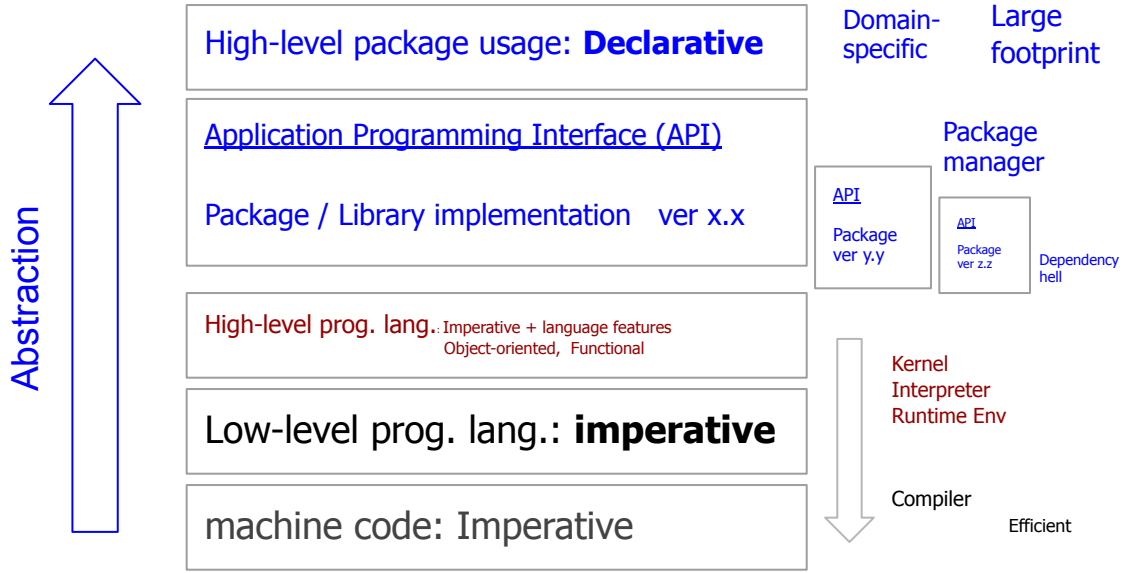
Imperative programming style

```
my_list = [1, 2, 3, 4, 5]

sum = 0
for x in my_list:
    sum += x
print(sum)
```

- Code *how* to achieve the goal
- Number of lines of code: high
- Tedious

Functional programming style

```
import functools

sum = functools.reduce(
    (lambda a, b: a+b), my_list)
print(sum)
```



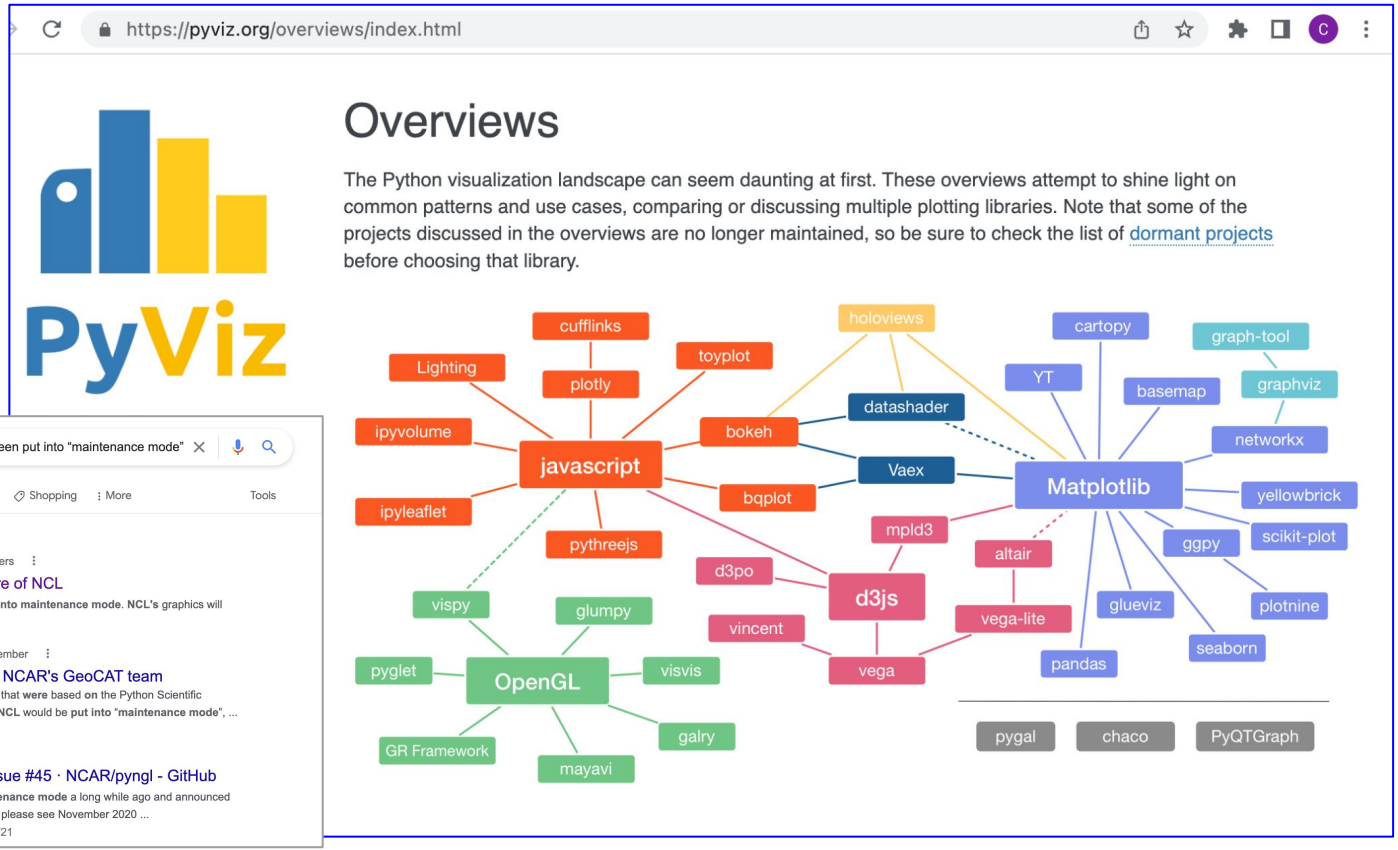
Pyviz - high level data visualization

This course:

Matplotlib
Cartopy
Holoviews
Bokeh
Plotly
...

Giving up

NCL
PyNGL



Code *what* you want

https://dev.to/ruizb/declarative-vs-imperative-4a7l

Log in | Create account

Making a chocolate cake

Let's take an example from the real world: we would like to make a chocolate cake. How would that look like with these 2 paradigms?

The imperative way

1. First, turn on the oven to preheat it at 180°C.
2. Next, add flour, sugar, cocoa powder, baking soda and salt to a large bowl, then stir the mixture with a paddle.
3. Then, add milk, vegetable oil, eggs and vanilla extract to the mixture, and mix together on medium speed until well combined.
4. Distribute the cake batter evenly in a large cake pan, then bake it for approx. 30 minutes.
5. Remove the pan from the oven with a pot holder, let it cool for 10 minutes.
6. Finally, remove the cake from the pan with the tapping method, and frost it evenly with chocolate frosting.

Benoit Ruiz

Follow

The declarative way

1. You have to preheat the oven to 180 °C.
2. You have to mix dry ingredients in a bowl.
3. Once dry ingredients are mixed, you have to add wet ingredients to the mixture, and mix together to form the cake batter.
4. Once the oven and batter are ready, you have to put the batter in a pan, then bake it for 30 minutes.
5. Once baked, you have to remove the pan from the oven and let it cool for 10 minutes.
6. Finally, you have to remove the cake from the pan, and frost it.
7. Ready? Go!

https://dev.to/ruizb/declarative-vs-imperative-4a7l

Labour replaced by software packages

declarative code

Chief chef

https://pngset.com/download-free-png-ubtqm

Typical code you'll write ...

1

```
# import necessary libraries
import numpy as np
import xarray as xr
import matplotlib.pyplot as plt

import hvplot.xarray
```

Memory: 143.6 MB / 1 GB CPU: 0% / 1 vCPU

CPU: 0% / 1 vCPU

Memory: 928.3 MB / 1 GB

⚠ Reduce memory usage by terminating other notebooks or restart kernel, or buy more resources.

Memory: 469.6 MB / 1 GB CPU: 118.1% / 1 vCPU

2

```
# Open .nc file with open_dataset as xarray.Dataset
ds = xr.open_dataset("data/ERA5_sample.nc")
ds  # show the data
```

executed in 157ms, finished 08:48:27 2022-04-13

xarray.Dataset

▸ Dimensions: (**longitude**: 1440, **latitude**: 721, **time**: 12)

▾ Coordinates:

longitude	(longitude)	float32	0.0 0.25 0.5 ... 359.2 359
latitude	(latitude)	float32	90.0 89.75 89.5 ... -89.7
time	(time)	datetime64[ns]	2021-01-01 ... 2021-12-

▾ Data variables:

u10	(time, latitude, longitude)	float32	...
v10	(time, latitude, longitude)	float32	...
t2m	(time, latitude, longitude)	float32	...
msl	(time, latitude, longitude)	float32	...

▾ Attributes:

Conventions :	CF-1.6
history :	2022-03-01 04:10:10 GMT by grib_to_netcdf-2.24.2: /opt/ecmwf/m

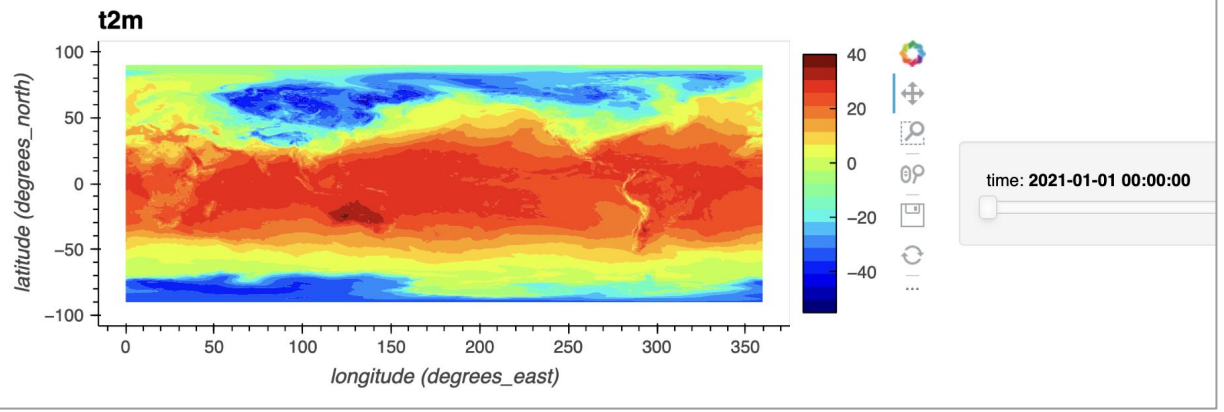
3

```
# reassign the variables from 'data' dataset
display(ds)

# Store the available data arrays from the data to the variables
# this time we retain all the dimension of the data arrays
t2m = ds.t2m-273.15  # convert the unit from K to deg C
msl = ds.msl/100    # convert the unit from Pa to hPa
u10 = ds.u10
v10 = ds.v10
```

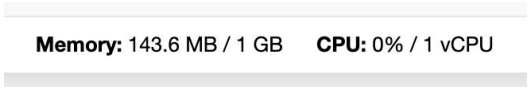
4

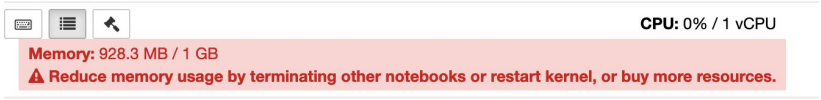
```
# Customize the contourf plot by adding different parameters
t2m.hvplot.contourf(x='longitude', y='latitude',
                    cmap='jet',
                    levels=20,  # set the number of contour levels
                    colorbar=True,
                    title="t2m",  # set the title of the plot
                    fontscale=1.2  # scale up the font size of all texts by 1.2
                    )
```

executed in 1.92s, finished 08:49:42 2022-04-13



Typical error you'll encounter …
… how students become reliable professionals





High-level package usage: **Declarative**

Application Programming Interface (API)

Package / Library implementation ver x.x

High-level prog. lang. Imperative + language features
Object-oriented, Functional

Domain-specific

Large footprint

Package manager

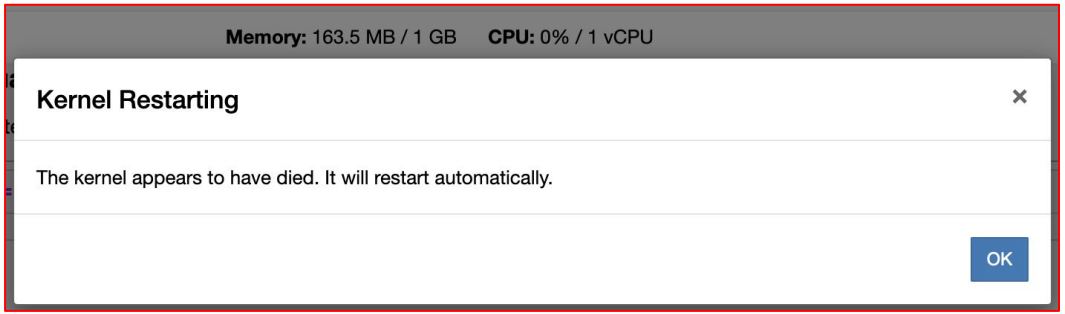
API

Package ver y.y

API

Package ver z.z

Dependency hell

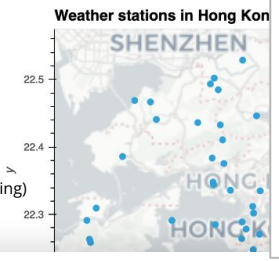


Expect to encounter error …
… Keep calm and read the error message … understand … decide …

In [34]:
```
# create an interactive map
geo_hk_station.hvplot(geo=True,tiles='CartoLight', # add map tiles
                      hover_cols='name_tc', # show the names of stations when hovering over the point
                      size=40,  # change the size of points
                      title="Weather stations in Hong Kong"
                     )
```

```
/opt/conda/lib/python3.8/site-packages/cartopy/crs.py:825: ShapelyDeprecationWarning: __len__ for multi-part geometries is deprecated and will be removed in Shapely 2.0. Check the length of the `geoms` pr
operty instead to get the  number of parts of a multi-part geometry.
  if len(multi_line_string) > 1:
/opt/conda/lib/python3.8/site-packages/cartopy/crs.py:877: ShapelyDeprecationWarning: Iteration over multi-part geometries is deprecated and will be removed in Shapely 2.0. Use the `geoms` property to acc
ess the constituent parts of a multi-part geometry.
  for line in multi_line_string:
/opt/conda/lib/python3.8/site-packages/cartopy/crs.py:944: ShapelyDeprecationWarning: __len__ for multi-part geometries is deprecated and will be removed in Shapely 2.0. Check the length of the `geoms` pr
operty instead to get the  number of parts of a multi-part geometry.
  if len(p_mline) > 0:
/opt/conda/lib/python3.8/site-packages/cartopy/crs.py:825: ShapelyDeprecationWarning: __len__ for multi-pa
operty instead to get the  number of parts of a multi-part geometry.
  if len(multi_line_string) > 1:
/opt/conda/lib/python3.8/site-packages/cartopy/crs.py:877: ShapelyDeprecationWarning: Iteration over multi
ess the constituent parts of a multi-part geometry.
  for line in multi_line_string:
/opt/conda/lib/python3.8/site-packages/cartopy/crs.py:944: ShapelyDeprecationWarning: __len__ for multi-pa
operty instead to get the  number of parts of a multi-part geometry.
  if len(p_mline) > 0:
/opt/conda/lib/python3.8/site-packages/cartopy/crs.py:825: ShapelyDeprecationWarning: __len__ for multi-pa
operty instead to get the  number of parts of a multi-part geometry.
  if len(multi_line_string) > 1:
/opt/conda/lib/python3.8/site-packages/cartopy/crs.py:877: ShapelyDeprecationWarning: Iteration over multi
ess the constituent parts of a multi-part geometry.
  for line in multi_line_string:
/opt/conda/lib/python3.8/site-packages/cartopy/crs.py:944: ShapelyDeprecationWarning: __len__ for multi-pa
operty instead to get the  number of parts of a multi-part geometry.
  if len(p_mline) > 0:
```

Out[34]:




```
from shapely.errors import ShapelyDeprecationWarning
import warnings
warnings.filterwarnings(action="ignore", category= ShapelyDeprecationWarning)
```

High-level package usage: **Declarative**

Domain-specific

Large footprint

Application Programming Interface (API)

Package / Library implementation ver x.x

Package manager

API

Package ver y.y

API

Package ver z.z

https://en.wikipedia.org/wiki/Dependency_hell

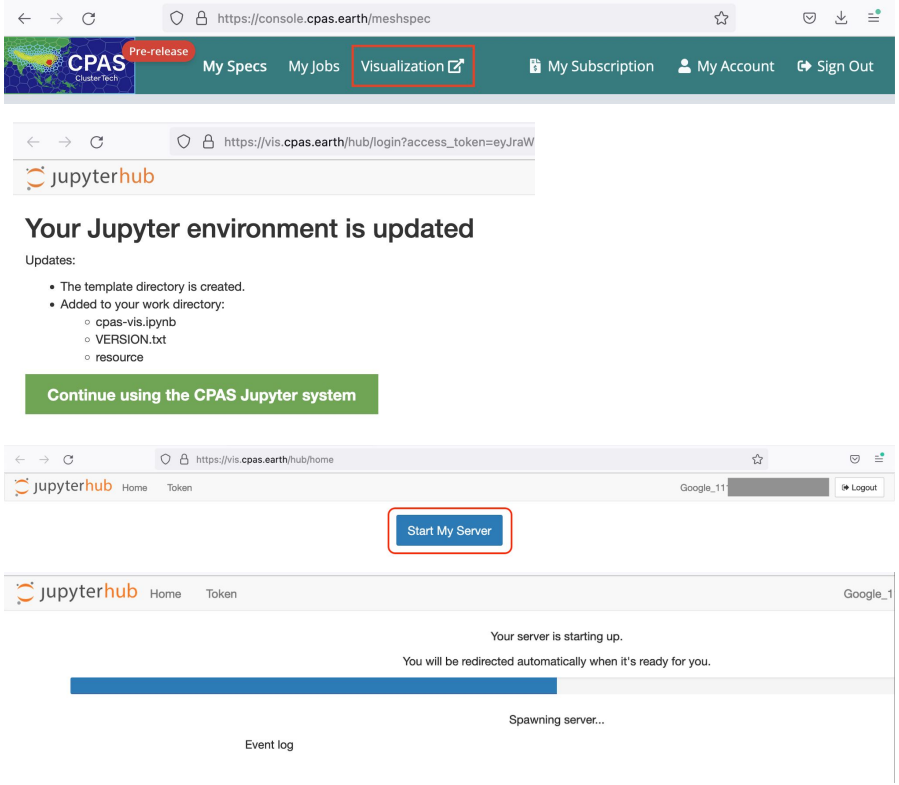
Dependency hell

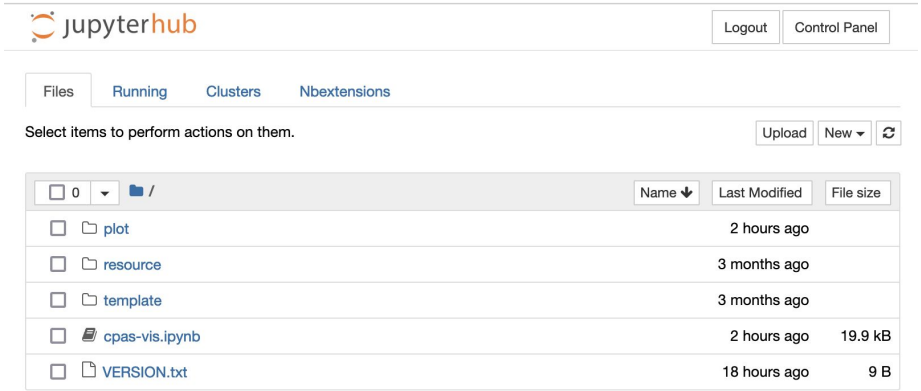
Lab 1
Hello My Jupyter Server

5 minutes

Let there be My Server

1. Click "Visualization"
2. (For new user / system update)
   See the Update message.
   Click "Continue using …"
3. Click "Start My Server".
4. See the Tree view of your files.

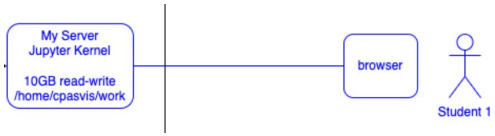




Hello Terminal

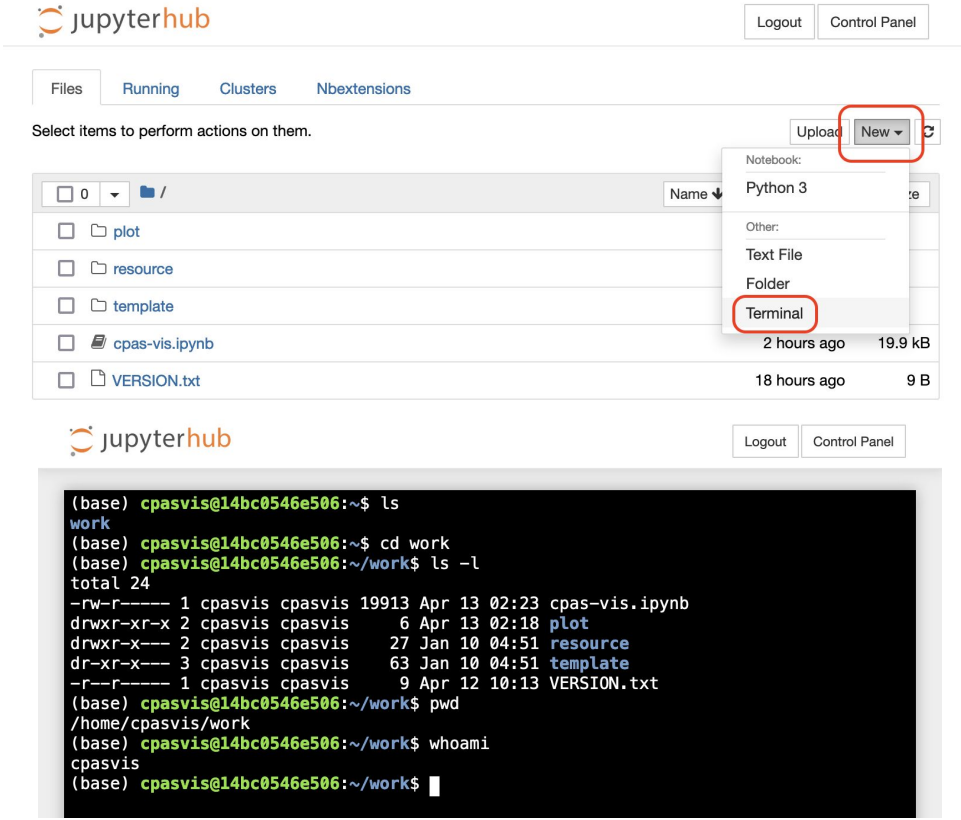
If the Terminal becomes not responsive, close the tab, start a new Terminal

1. Click "New", "Terminal"
2. See the **Linux** terminal.
3. Type Linux command "ls"
4. $ cd work
   $ ls -l
   $ pwd
   $ whoami



5. Note: Only the directory (and its sub-directory)
   /home/cpasvis/work
   is permanent storage (during subscription)

   You should not write files to other location.
   They may be lost (e.g. "Stop Server")



Soft-link data; copy your own .ipynb

Course materials are in

/mnt/demo_cpas_data/projects/atm_learner

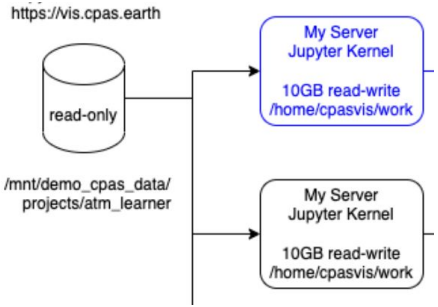
Use soft-link to memorize the long path

$ ln -s /mnt/demo_cpas_data/projects/atm_learner ./
$ ls -l atm_learner

Your storage space is precious;
Avoid making copies of common data;
but you need your own writable .ipynb

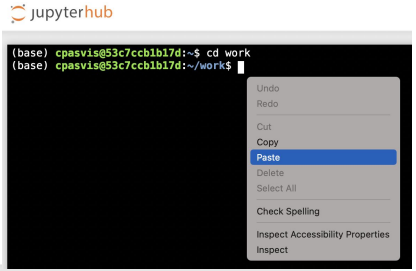
$ mkdir my_atm_learner
$ cd my[Tab autocompletion]_atm_learner/

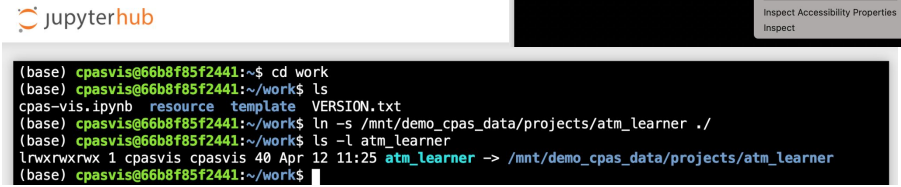
$ ln -s ../atm[Tab]/data_for_tutorials/ ./
$ cp -r ../atm[Tab]/<which lesson>/ ./
$ ls -l

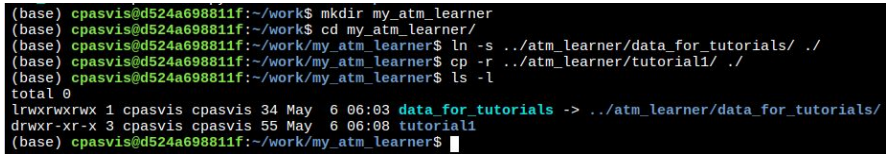


Reminder:

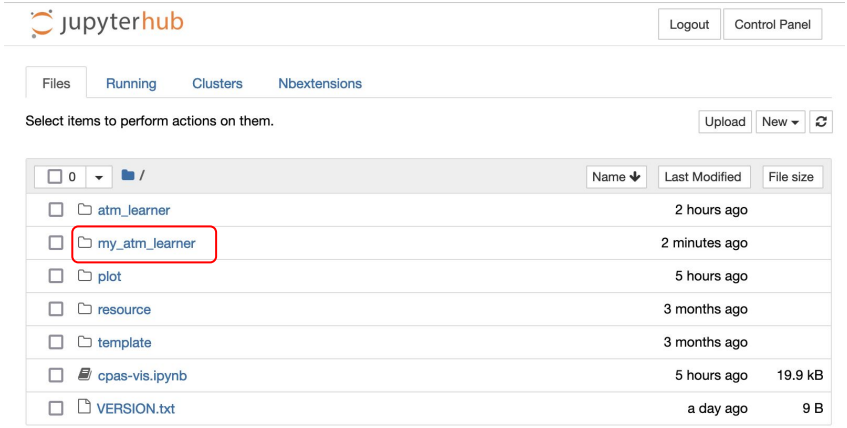
Send Slide URL in Zoom chat
Copy command from slide
Paste to the Terminal

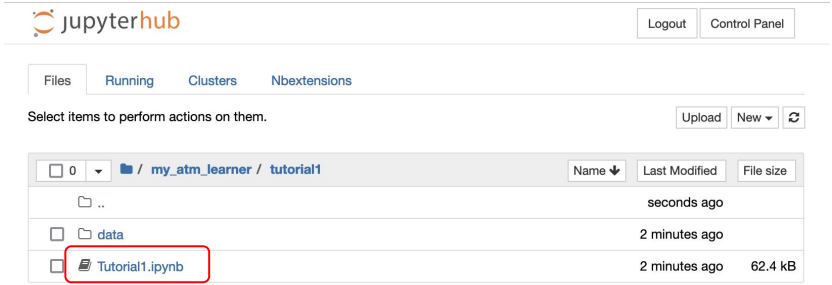


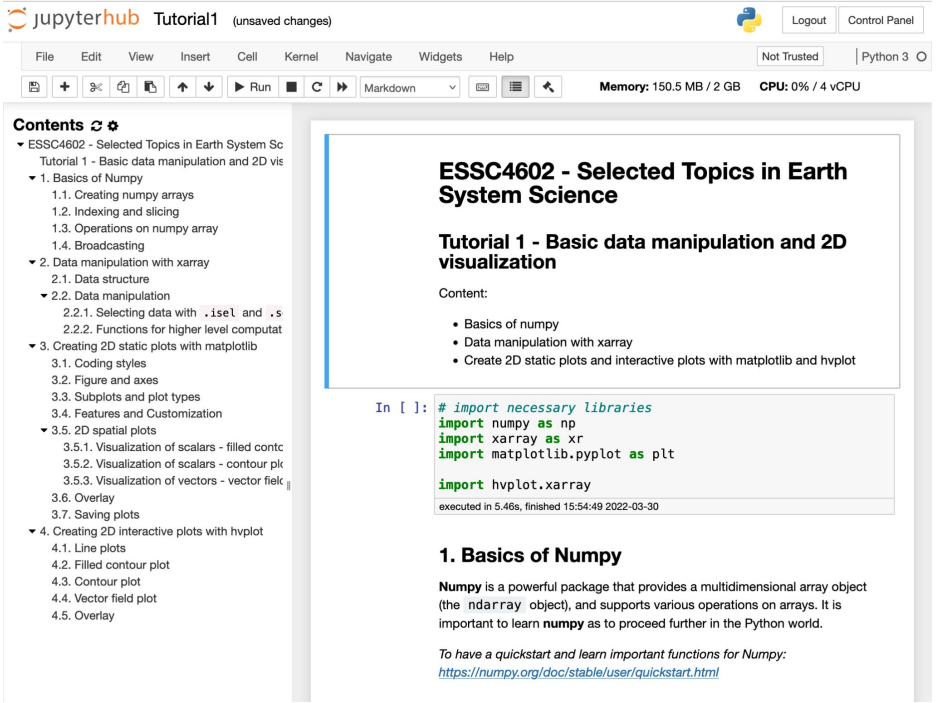




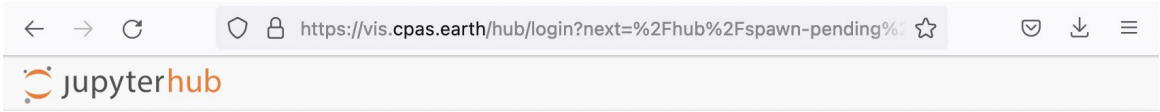
Hello Tutor's Notebook



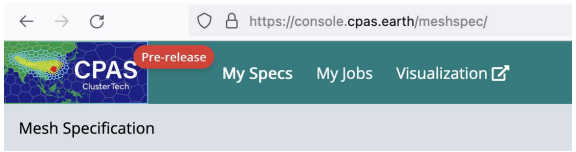




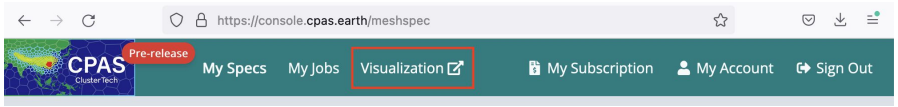
If idle too long / lost connection



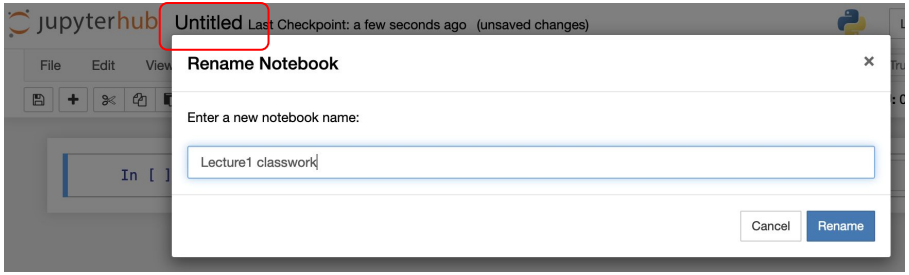
1 **Refresh** this tab:



2 Click "Visualization" link again.

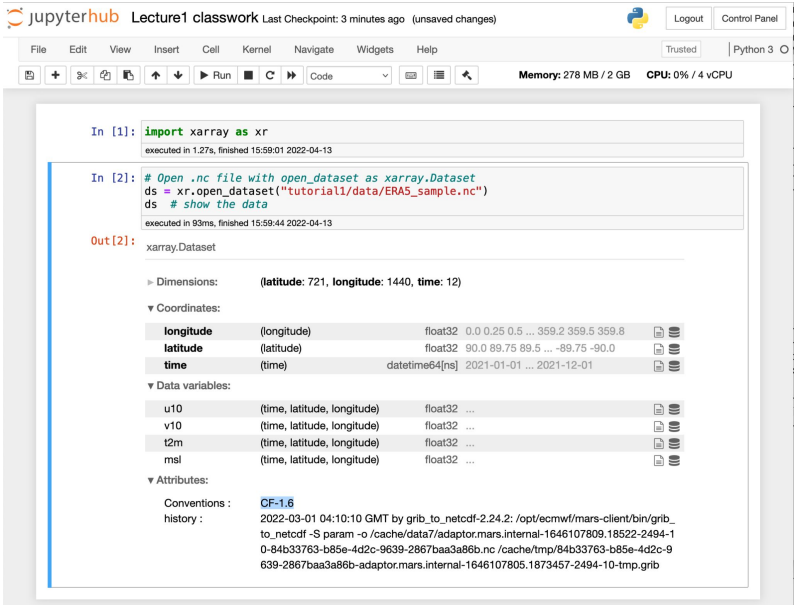


Create my own classwork Notebook



Task: Open a .nc data file

Hint: xarray provides open_dataset()



Lab 1
Hello My Jupyter Server

All students should be done - Voice out if you haven’t.

You cannot do later Labs if you have not done this step

netCDF file data structure

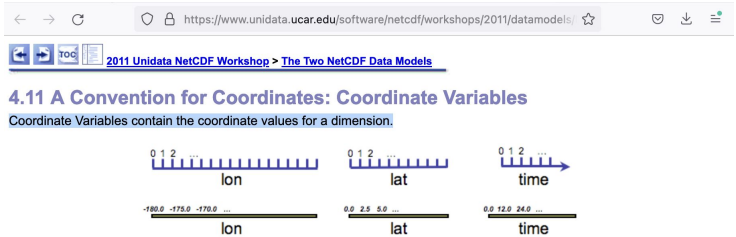
Data structure:

Data variables

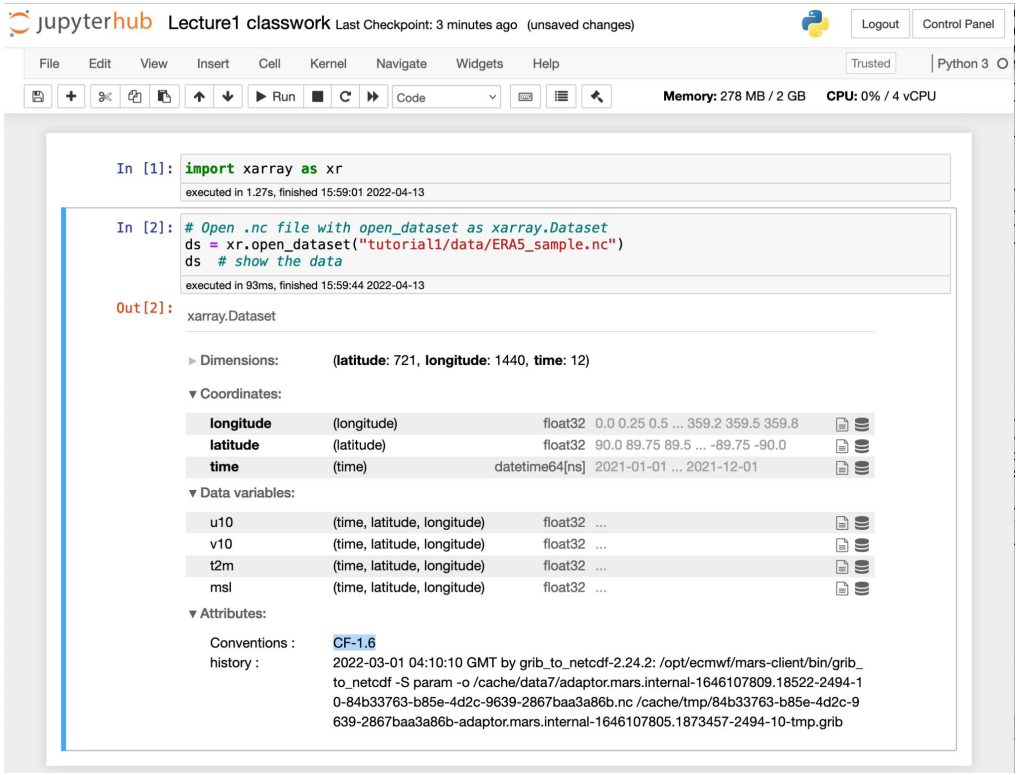
- n-D array with name and attributes

Dimensions

- Name of dimensions and length



https://www.unidata.ucar.edu/software/netcdf/workshops/2011/datamodels/NcCVars.html



Lab 2
What is “CF” convention?

5 minutes

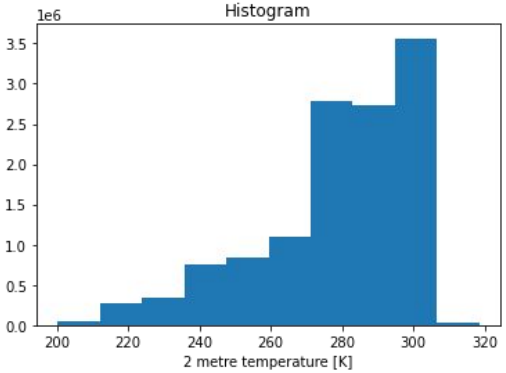
What you can do with a ds object?

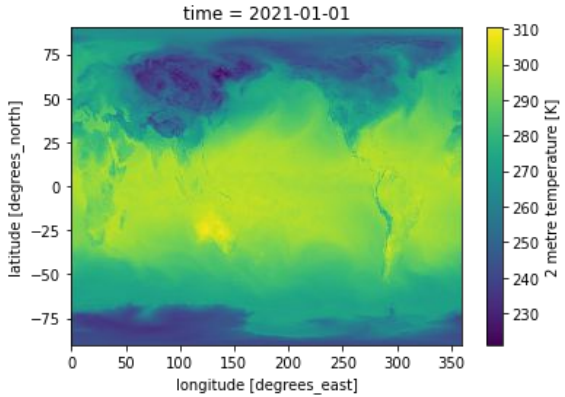
Pre-condition:

```
import xarray as xr
ds = xr.open_dataset("tutorial1/data/ERA5_sample.nc")
```

Explore the following:

1. ds.t2m.plot()
2. ds.t2m.isel(time=0).plot()





Advice:

Get used to typing code! => Proficiency

If you copy-and-paste, you don't know what you are coding.

How does the software know ??

I didn't put in code

"2 metre temperature [K]"

"latitude [degrees_north]"

"Longitude [degrees_east]"

Explore here:

```
In [2]: # Open .nc file with open_dataset as xarray.Dataset
        ds = xr.open_dataset("tutorial1/data/ERA5_sample.nc")
        ds  # show the data
```

executed in 78ms, finished 17:08:20 2022-04-13

Out[2]: xarray.Dataset

▸ Dimensions: (**latitude**: 721, **longitude**: 1440, **time**: 12)

▾ Coordinates:

longitude	(longitude)	float32	0.0 0.25 0.5 ... 359...
latitude	(latitude)	float32	90.0 89.75 89.5 ... ...
time	(time)	datetime64[ns]	2021-01-01 ... 202...

Show/Hide attributes

▾ Data variables:

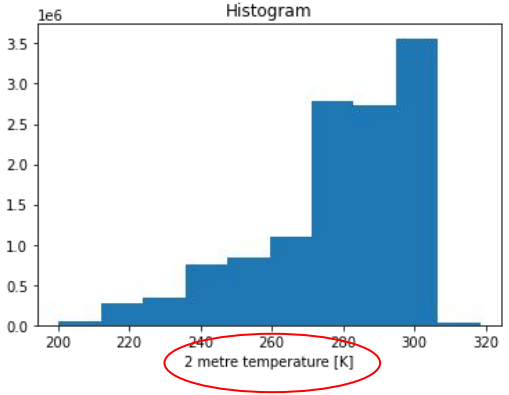
u10	(time, latitude, longitude)	float32	...
v10	(time, latitude, longitude)	float32	...
t2m	(time, latitude, longitude)	float32	...
msl	(time, latitude, longitude)	float32	...

▾ Attributes:

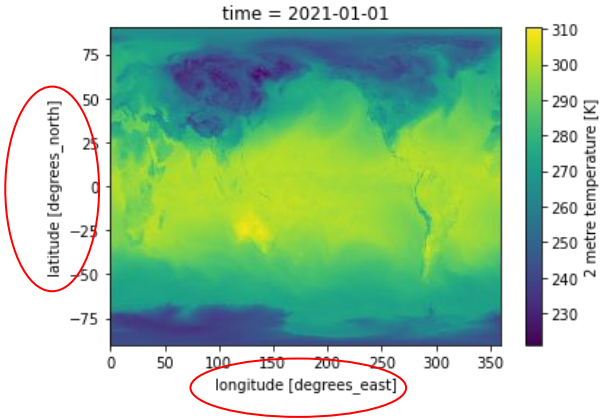
Conventions : CF-1.6

history : 2022-03-01 04:10:10 GMT by grib_to_netcdf-2.24.2: /opt/ecmwf/mars-client/bin/grib_to_netcdf -S param -o /cache/data7/adaptor.mars.internal-1646107809.18522-2494-10-84b33763-b85e-4d2c-9639-2867baa3a86b.nc /cache/tmp/84b33763-b85e-4d2c-9639-2867baa3a86b-adaptor.mars.internal-1646107805.1873457-2494-10-tmp.grib

1.ds.t2m.plot()



ds.t2m.plot()



Nice data producer

The data producer already put meta-data to netCDF attributes, following the CF convention.

```
# Open .nc file with open_dataset as xarray.Dataset
ds = xr.open_dataset("tutorial1/data/ERA5_sample.nc")
ds  # show the data
```

executed in 78ms, finished 17:08:20 2022-04-13

xarray.Dataset

▸ Dimensions: (**latitude**: 721, **longitude**: 1440, **time**: 12)

▾ Coordinates:

longitude	(longitude)	float32	0.0 0.25 0.5 ... 359...
units :	degrees_east		
long_name :	longitude		
latitude	(latitude)	float32	90.0 89.75 89.5 ... ...
units :	degrees_north		
long_name :	latitude		
time	(time)	datetime64[ns]	2021-01-01 ... 202...

▾ Data variables:

u10	(time, latitude, longitude)	float32	...
v10	(time, latitude, longitude)	float32	...
t2m	(time, latitude, longitude)	float32	...
msl	(time, latitude, longitude)	float32	...

▾ Attributes:

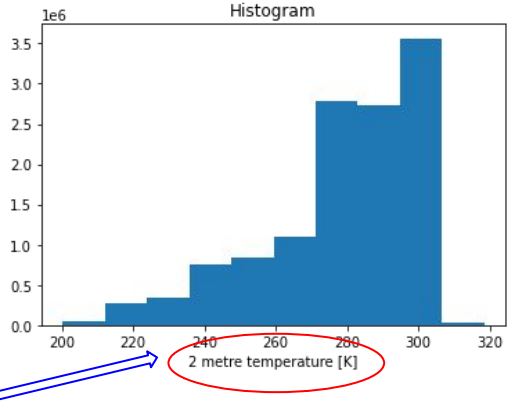
Conventions : CF-1.6

history : 2022-03-01 04:10:10 GMT by grib_to_netcdf-2.24.2: /opt/ecmwf/mars-client/bin/grib_to_netcdf -S param -o /cache/data7/adaptor.mars.internal-1646107809.18522-2494-10-84b33763-b85e-4d2c-9639-2867baa3a86b.nc /cache/tmp/84b33763-b85e-4d2c-9639-2867baa3a86b-adaptor.mars.internal-1646107805.1873457-2494-10-tmp.grib

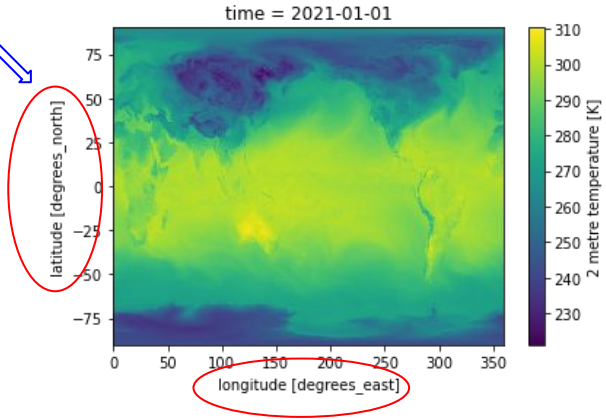
1.ds.t2m.plot()

Domain-specific:
The package knows CF

Labour replaced by software packages



ds.t2m.plot()



Meet hvplot(), and its (default) bokeh backend

Pre-condition:

```
import xarray as xr
ds = xr.open_dataset("tutorial1/data/ERA5_sample.nc")
```

Explore the following:

1. ds.t2m.plot()
2. ds.t2m.isel(time=0).plot()

3. import hvplot.xarray
   ds.t2m.isel(time=0).hvplot()

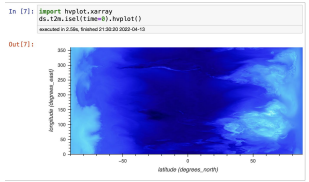
Reference:
https://hvplot.holoviz.org/user_guide/Geographic_Data.html

Question:

1) Is plot() or hvplot() smarter?
   Which one missed what desirable feature?

Hint: Use your curiosity and mouse to explore.

Problem: Longitude as y-axis
Latitude as x-axis



2) How to make hvplot() do exactly what I want?

Hint:
i) Give more specific message to hvplot() via input arguments.

ii) Try the arguments: x, y, geo

iii) Can't recognize the place in 2D horizontal plot? Try argument: coastline

Putting Coordinates to the correct axis

Another example put Latitude and Longitude to the correct axis

https://hvplot.holoviz.org/user_guide/Gridded_Data.html

How does it do that?

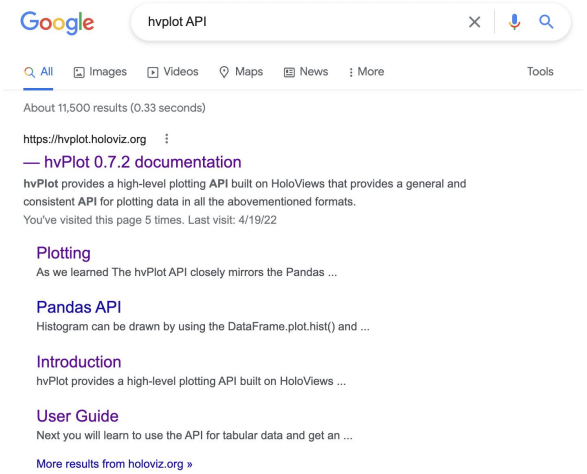
- Attributes 'axis' in Coordinates

▸ Dimensions: (**lat**: 25, **lon**: 53, **time**: 2920)

▾ Coordinates:

lat	(lat)	float32	75.0 72.5 70.0 ... 20.0 17.5 15.0	
standard_name :	latitude			
long_name :	Latitude			
units :	degrees_north			
axis :	Y			
lon	(lon)	float32	200.0 202.5 205.0 ... 327.5 330.0	
standard_name :	longitude			
long_name :	Longitude			
units :	degrees_east			
axis :	X			
time	(time)	datetime64[ns]	2013-01-01 ... 2014-12-31T18:00:00	

Any other way? Google "hvplot API"



https://hvplot.holoviz.org/user_guide/Plotting.html

<data obj>.hvplot(x="Latitude", y="Longitude")

Lab 2
What is “CF” convention?

Time’s up

One solution to get something plotted out by hvplot() in a way you may like:

```
import hvplot.xarray
ds.t2m.isel(time=5).hvplot(
    x='longitude', y='latitude',
    geo=True, coastline='110m')
```

Use of multiple CPU core / under RAM size limitation

xarray chunks and parallel processing

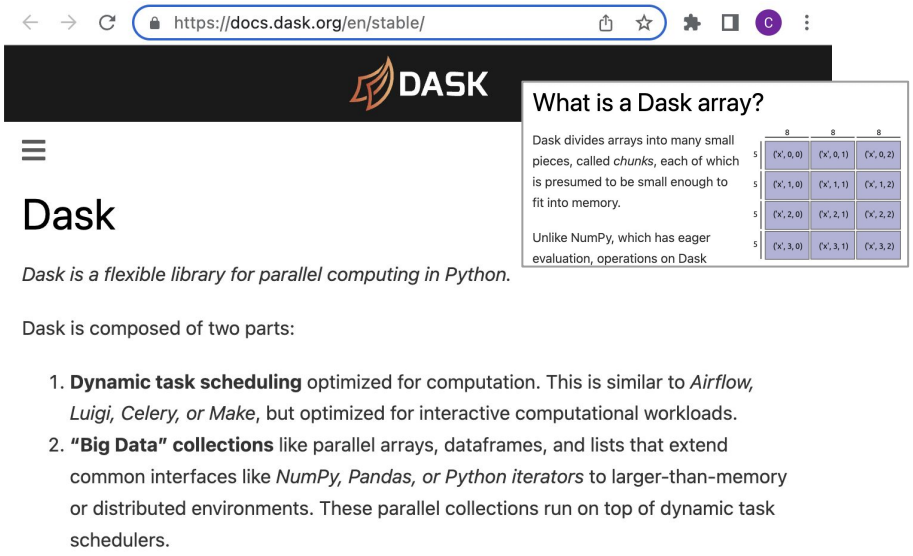
Xarray is integrated with Dask

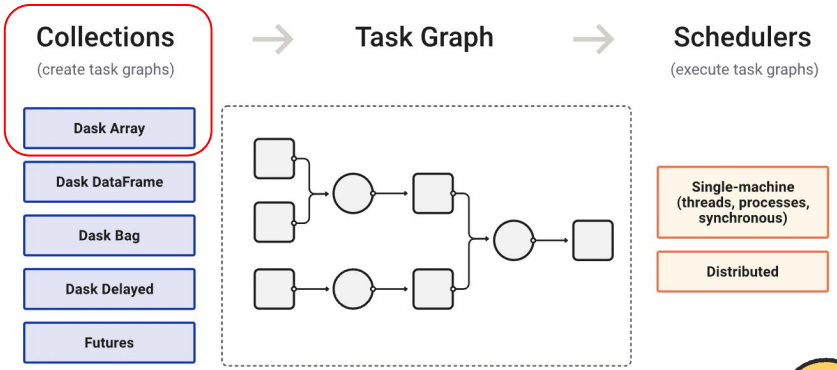
https://docs.xarray.dev/en/latest/user-guide/dask.html

(Dask documentation itself talks more about Pandas.)

Tell xarray to cut the n-D array data into chunks

- Add argument
  - `chunks={...}`
- to
  - `xr.open_dataset("/the/data.nc", ...)`





High level collections are used to generate task graphs which can be executed by schedulers on a single machine or a cluster.

xarray chunks and parallel processing

```
ds = xr.open_dataset("tutorial1/data/ERA5_sample.nc", chunks={})
ds  # show the data
```

executed in 132ms, finished 11:21:58 2022-04-20

xarray.Dataset

▸ Dimensions: (**latitude**: 721, **longitude**: 1440, **time**: 12)

Choose a dimension to decompose

▾ Coordinates:

longitude	(longitude)	float32	0.0 0.25 0.5 ... 359...
latitude	(latitude)	float32	90.0 89.75 89.5 ... ...
time	(time)	datetime64[ns]	2021-01-01 ... 202...

▾ Data variables:

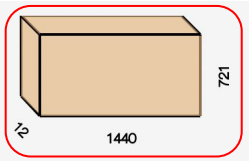
u10	(time, latitude, longitude)	float32	dask.array<chunks...
v10	(time, latitude, longitude)	float32	dask.array<chunks...
t2m	(time, latitude, longitude)	float32	dask.array<chunks...
msl	(time, latitude, longitude)	float32	dask.array<chunks...

```
ds.msl
```

executed in 42ms, finished 11:22:19 2022-04-20

xarray.DataArray 'msl' (**time**: 12, **latitude**: 721, **longitude**: 1440)

	Array	Chunk
Bytes	47.53 MiB	47.53 MiB
Shape	(12, 721, 1440)	(12, 721, 1440)
Count	2 Tasks	1 Chunks
Type	float32	numpy.ndarray



▾ Coordinates:

longitude	(longitude)	float32	0.0 0.25 0.5 ... 359.2 359.5 359.8
latitude	(latitude)	float32	90.0 89.75 89.5 ... -89.75 -90.0
time	(time)	datetime64[ns]	2021-01-01 ... 2021-12-01

```
ds = xr.open_dataset("tutorial1/data/ERA5_sample.nc", chunks={'longitude' : 100})
ds  # show the data
```

executed in 68ms, finished 11:27:13 2022-04-20

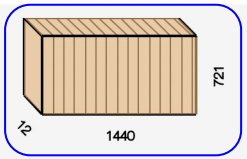
CPU: 0% / 4 vCPU

```
ds.msl
```

executed in 34ms, finished 11:29:09 2022-04-20

xarray.DataArray 'msl' (**time**: 12, **latitude**: 721, **longitude**: 1440)

	Array	Chunk
Bytes	47.53 MiB	3.30 MiB
Shape	(12, 721, 1440)	(12, 721, 100)
Count	16 Tasks	15 Chunks
Type	float32	numpy.ndarray



Lazy evaluation

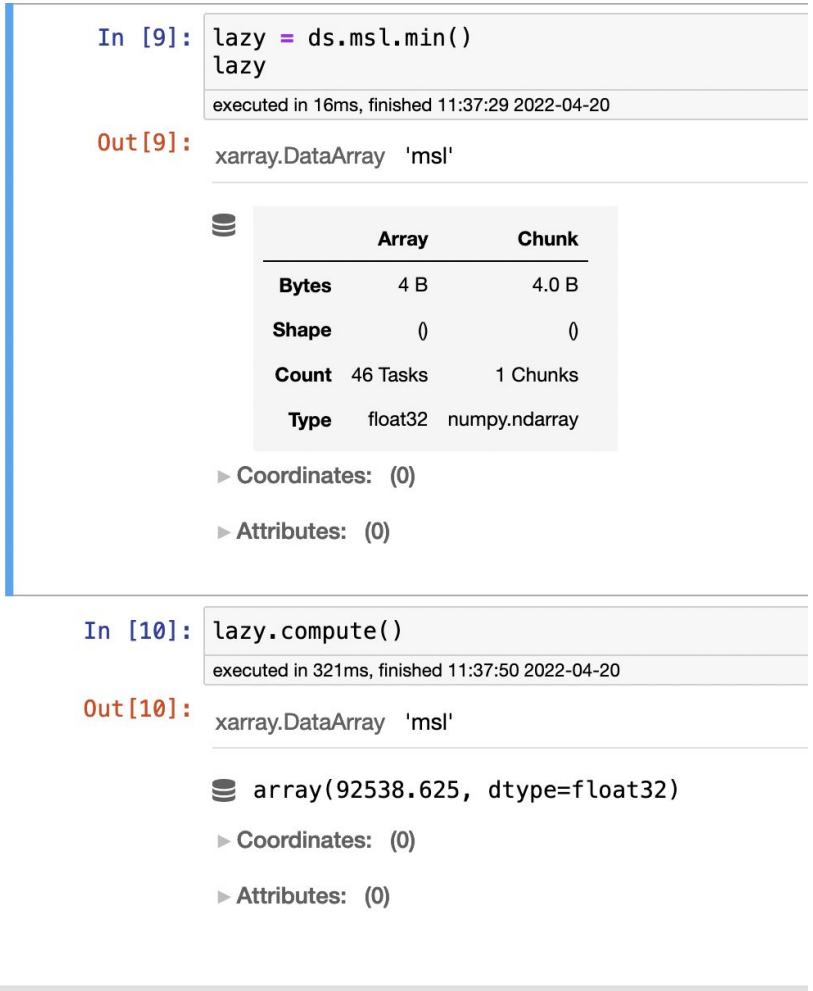
The object memorize what needs to be done.

=> task graph

Execute the computation only when triggered

```
.compute()
.load()
```

Triggers computation.



Lab 3
Make multi-cores busy

5 minutes

Try massive data... and monitor the system

- Open a terminal, run
  - (base) cpasvis@3cxxxxxxxxfb:~$ htop
- Separate the browser tab as a window
- View both Notebook and htop windows
  - side by side

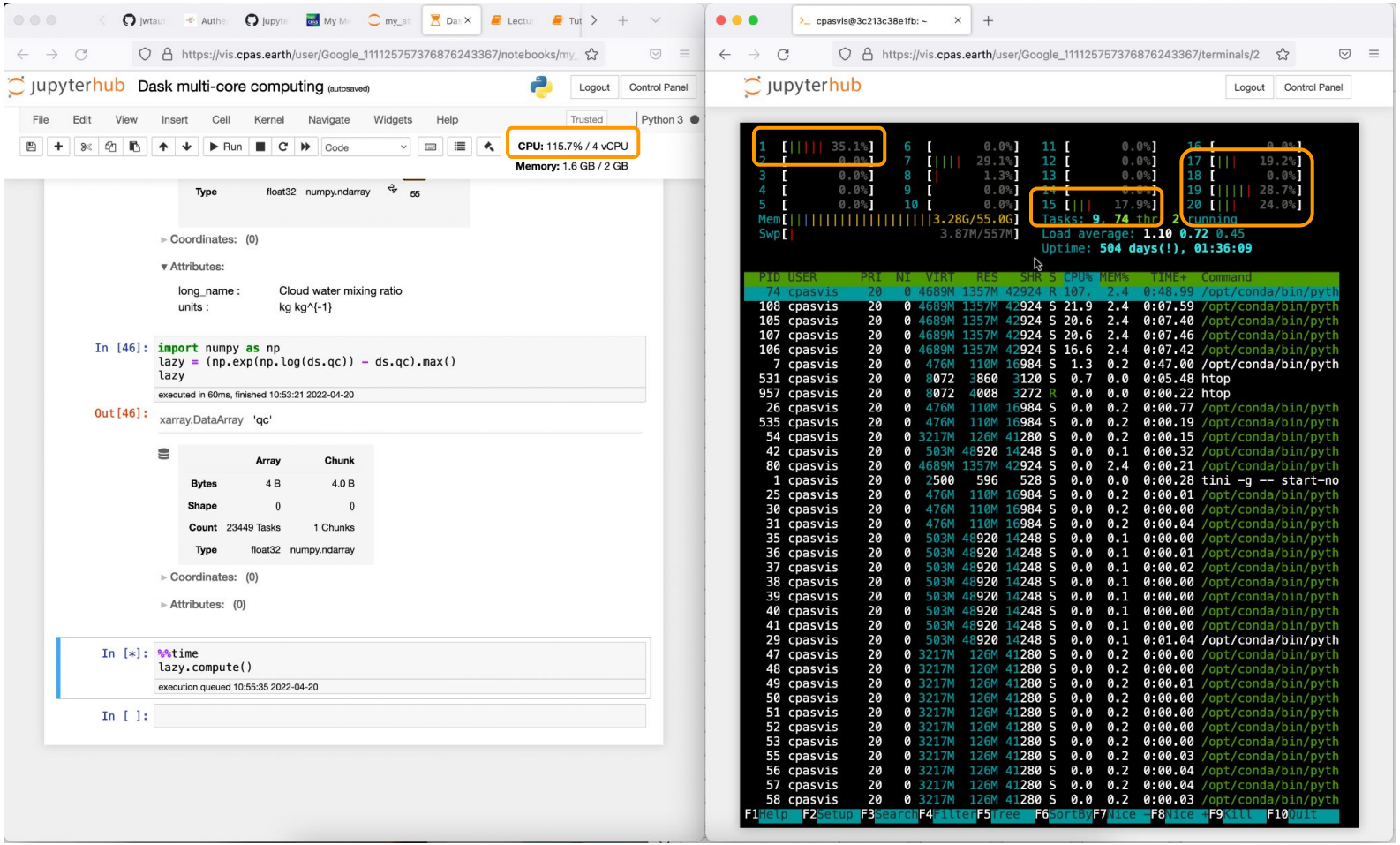
```
import xarray as xr

ds = xr.open_mfdataset(

"/home/cpasvis/work/atm_learner/demo_mangkhut/rsim/2018091300/atm/diag.2018-09-13_*.nc",

    concat_dim='Time', combine="nested",

    chunks={"nCells": 1000}

)
```

```
import numpy as np

lazy = (np.exp(np.log(ds.qc)) - ds.qc).max()

lazy
```

```
%%time

lazy.compute()
```

- Try also

```
xr.open_mfdataset(

"/multiple/files/something*.nc",

    parallel=True)
```


jupyterhub
Dask multi-core computing (autosaved)
Logout
Control Panel
File Edit View Insert Cell Kernel Navigate Widgets Help
Trusted
Python 3
Run
Code
CPU: 115.7% / 4 vCPU
Memory: 1.6 GB / 2 GB
Type float32 numpy.ndarray
55
Coordinates: (0)
Attributes:
long_name : Cloud water mixing ratio
units : kg kg^{-1}
In [46]:
import numpy as np
lazy = (np.exp(np.log(ds.qc)) - ds.qc).max()
lazy
executed in 60ms, finished 10:53:21 2022-04-20
Out[46]:
xarray.DataArray 'qc'
Array Chunk
Bytes 4 B 4.0 B
Shape () ()
Count 23449 Tasks 1 Chunks
Type float32 numpy.ndarray
Coordinates: (0)
Attributes: (0)
In [*]:
%%time
lazy.compute()
execution queued 10:55:35 2022-04-20
In [ ]:
cpasvis@3c213c38e1fb: ~
https://vis.cpas.earth/user/Google_111125757376876243367/terminals/2
jupyterhub
Logout
Control Panel
Tasks: 9, 74 thr; 2 running
Load average: 1.10 0.72 0.45
Uptime: 504 days(!), 01:36:09
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
74 cpasvis 20 0 4689M 1357M 42924 R 107. 2.4 0:48.99 /opt/conda/bin/pyth
108 cpasvis 20 0 4689M 1357M 42924 S 21.9 2.4 0:07.59 /opt/conda/bin/pyth
105 cpasvis 20 0 4689M 1357M 42924 S 20.6 2.4 0:07.40 /opt/conda/bin/pyth
107 cpasvis 20 0 4689M 1357M 42924 S 20.6 2.4 0:07.46 /opt/conda/bin/pyth
106 cpasvis 20 0 4689M 1357M 42924 S 16.6 2.4 0:07.42 /opt/conda/bin/pyth
7 cpasvis 20 0 476M 110M 16984 S 1.3 0.2 0:47.00 /opt/conda/bin/pyth
531 cpasvis 20 0 8072 3860 3120 S 0.7 0.0 0:05.48 htop
957 cpasvis 20 0 8072 4008 3272 R 0.0 0.0 0:00.22 htop
26 cpasvis 20 0 476M 110M 16984 S 0.0 0.2 0:00.77 /opt/conda/bin/pyth
535 cpasvis 20 0 476M 110M 16984 S 0.0 0.2 0:00.19 /opt/conda/bin/pyth
54 cpasvis 20 0 3217M 126M 41280 S 0.0 0.2 0:00.15 /opt/conda/bin/pyth
42 cpasvis 20 0 503M 48920 14248 S 0.0 0.1 0:00.32 /opt/conda/bin/pyth
80 cpasvis 20 0 4689M 1357M 42924 S 0.0 2.4 0:00.21 /opt/conda/bin/pyth
1 cpasvis 20 0 2500 596 528 S 0.0 0.0 0:00.28 tini -g -- start-no
25 cpasvis 20 0 476M 110M 16984 S 0.0 0.2 0:00.01 /opt/conda/bin/pyth
30 cpasvis 20 0 476M 110M 16984 S 0.0 0.2 0:00.00 /opt/conda/bin/pyth
31 cpasvis 20 0 476M 110M 16984 S 0.0 0.2 0:00.04 /opt/conda/bin/pyth
35 cpasvis 20 0 503M 48920 14248 S 0.0 0.1 0:00.00 /opt/conda/bin/pyth
36 cpasvis 20 0 503M 48920 14248 S 0.0 0.1 0:00.01 /opt/conda/bin/pyth
37 cpasvis 20 0 503M 48920 14248 S 0.0 0.1 0:00.02 /opt/conda/bin/pyth
38 cpasvis 20 0 503M 48920 14248 S 0.0 0.1 0:00.00 /opt/conda/bin/pyth
39 cpasvis 20 0 503M 48920 14248 S 0.0 0.1 0:00.00 /opt/conda/bin/pyth
40 cpasvis 20 0 503M 48920 14248 S 0.0 0.1 0:00.00 /opt/conda/bin/pyth
41 cpasvis 20 0 503M 48920 14248 S 0.0 0.1 0:00.00 /opt/conda/bin/pyth
29 cpasvis 20 0 503M 48920 14248 S 0.0 0.1 0:01.04 /opt/conda/bin/pyth
47 cpasvis 20 0 3217M 126M 41280 S 0.0 0.2 0:00.00 /opt/conda/bin/pyth
48 cpasvis 20 0 3217M 126M 41280 S 0.0 0.2 0:00.00 /opt/conda/bin/pyth
49 cpasvis 20 0 3217M 126M 41280 S 0.0 0.2 0:00.01 /opt/conda/bin/pyth
50 cpasvis 20 0 3217M 126M 41280 S 0.0 0.2 0:00.00 /opt/conda/bin/pyth
51 cpasvis 20 0 3217M 126M 41280 S 0.0 0.2 0:00.00 /opt/conda/bin/pyth
52 cpasvis 20 0 3217M 126M 41280 S 0.0 0.2 0:00.00 /opt/conda/bin/pyth
53 cpasvis 20 0 3217M 126M 41280 S 0.0 0.2 0:00.00 /opt/conda/bin/pyth
55 cpasvis 20 0 3217M 126M 41280 S 0.0 0.2 0:00.03 /opt/conda/bin/pyth
56 cpasvis 20 0 3217M 126M 41280 S 0.0 0.2 0:00.04 /opt/conda/bin/pyth
57 cpasvis 20 0 3217M 126M 41280 S 0.0 0.2 0:00.04 /opt/conda/bin/pyth
58 cpasvis 20 0 3217M 126M 41280 S 0.0 0.2 0:00.03 /opt/conda/bin/pyth
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice -F8Nice +F9Kill F10Quit

Execution time can be subtle ...

chunks = {}

Care about it only when the job is really slow.
- just let you know such support is right there.
- more other parallel computing skill can be explored ...
  if repeated execution time > your learning + development time,
  or you think it is interesting itself (hello computer scientist)

```
In [33]: %%time
         lazy.compute()
```

executed in 7.22s, finished 10:42:01 2022-04-20

```
CPU times: user 462 ms, sys: 334 ms, total: 796 ms
Wall time: 7.2 s
```

chunks={"nCells": 1000}

```
In [37]: %%time
         lazy.compute()
```

executed in 2.88s, finished 10:44:00 2022-04-20

```
CPU times: user 2.53 s, sys: 629 ms, total: 3.16 s
Wall time: 2.86 s
```

Lab 3
Make multi-cores busy

Time’s up

More about Dask and effective data processing

Out of syllabus, only for those interested in computer science:

https://docs.xarray.dev/en/latest/user-guide/dask.html#chunking-and-performance

https://docs.dask.org/en/stable/array.html

https://stephanhoyer.com/2015/06/11/xray-dask-out-of-core-labeled-arrays/
(An old article, but stated a case Dask can handle data with size bigger than RAM size and chunks)

https://examples.dask.org/delayed.html#Visualize-computation

https://www.youtube.com/watch?v=mDrjGxaXQT4
(highlight: 6:00 - 8:54, 11:00 - 12:20)