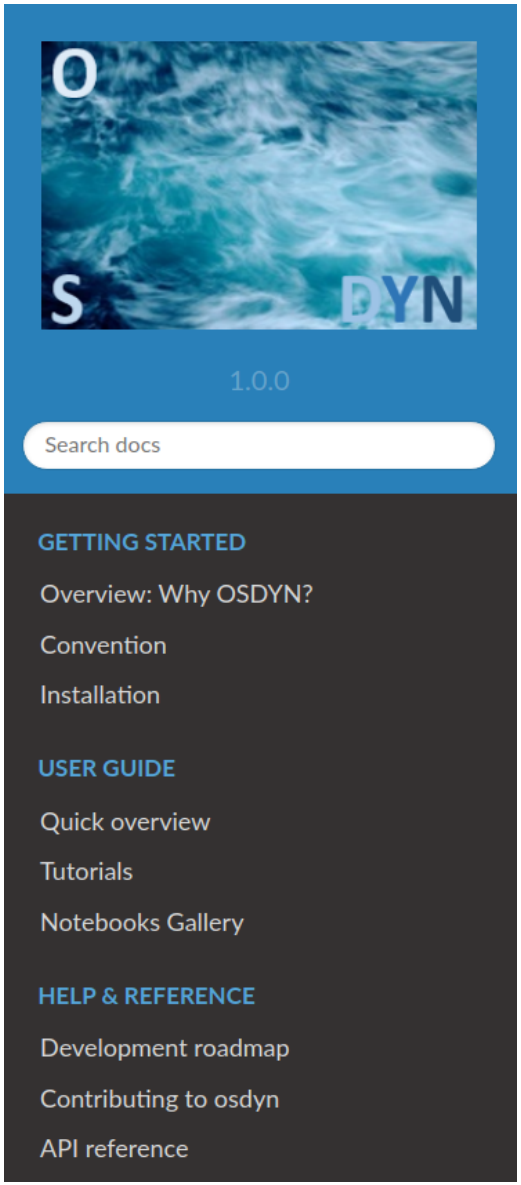
OSDYN: a new python tool for the analysis of high-volume ocean outputs.

V. Garnier, J.-F. Le Roux, J. Magin, T. Odaka (Ifremer, LOPS), S. Raynaud (SHOM),

C. Estournel (LEGOS), J. Beuvier (Mercator Ocean International, Toulouse),
P. Garreau (Ifremer, LOPS), M. Boutet (If Caen)

June 9, 2022

Observations and Simulations of the DYNamics



Scientific analyses of data from atmospheric, oceanic, wave models and remote or in-situ observations.

- large datasets (hourly, years, 100 meters)
- easily
- quickly

- (auto-)kerchunk package
- Osdyn readers
- Pangeo ecosystem (xarray, dask, xgcm, xesmf...)

https://osdyn.ifremer.fr/osdyn

auto-kerchunk package: pre-reading

This step is done once for good for each dataset.

- metadata of netCDF files gathered (.json)
- description file (.yaml) called by intake to get the dataset
- requires netcdf4 (hdf5) + chunks

```yaml
metadata:
    model:  mars
    positive:  up
name: marc_f2_1200_sn_T1Z60Y462X1100
sources:
  marc_f2_1200_sn_T1Z60Y462X1100:
    args:
      consolidated: false
      storage_options:
        fo: file:///home/datawork-lops-siam-moawi/PROJECTS/OSDYN/DATA/CATALOG/KERCHUNK/marc_f2_1200_sn_T1Z60Y462X1100.json.zst
        target_options:
          compression: zstd
        target_protocol: file
      urlpath: reference://
    description: description
    driver: zarr
    name: marc_f2_1200_sn_T1Z60Y462X1100
    parameters: {}

  marc_f2_1200_sn_T1Z60Y462X1100_grid:
    args:
      urlpath: /home/datawork-lops-siam-moawi/PROJECTS/COTACOT/DATA/MODELS/MARS/F2_1200_SN/SUBSET/grid_SUB_MARC_F2-MARS3D-MENOR1200
    description: description
    driver: netcdf
    name: marc_f2_1200_sn_T1Z60Y462X1100_grid
```

auto-kerchunk package: pre-reading

MEDRYS, a Mediterranean Sea reanalysis over the period 1992–2013 (Hamon et al., 2016)

```yaml
metadata:
    model:  nemo
    parameters:
        cgrid:
            description: data stored according to their location on the Arakawa c-grid
            type: str
            default: "gridT"
            allowed: ["grid2D", "gridT", "gridS", "gridU", "gridV"]
    positive:  down
name: medrys_T1Z75Y264X567
sources:
  medrys_T1Z75Y264X567:
    args:
      consolidated: false
      storage_options:
        fo: file:///home/datawork-lops-siam-moawi/PROJECTS/OSDYN/DATA/CATALOG/KERCHUNK/medrys_T1Z75Y264X567.json.zst/{{ cgrid }}.json.zst
        target_options:
          compression: zstd
        target_protocol: file
      urlpath: reference://
    description: description
    driver: zarr
    name: medrys_T1Z75Y264X567
    parameters: {}

  medrys_T1Z75Y264X567_grid:
    args:
      urlpath: /home/datawork-lops-siam-moawi/PROJECTS/COTACOT/DATA/MODELS/NEMO/MEDRYS1V2/medrys_grid_osdyn.nc
    description: description
    driver: netcdf
    name: medrys_T1Z75Y264X567_grid
~
```

read yaml from intake

Mars f2_sn

```
[4]: from intake import open_catalog
     yaml = "/home/datawork-lops-siam-moawi/PROJECTS/OSDYN/DATA/CATALOG/INTAKE/marc_f2_1200_sn_T
     cat = open_catalog(yaml)
     cat
```

```
marc_f2_1200_sn_T1Z60Y462X1100:
  args:
    path: /home/datawork-lops-siam-moawi/PROJECTS/OSDYN/DATA/CATALOG/INTAKE/marc_f2_1200_sn
  description: ''
  driver: intake.catalog.local.YAMLFileCatalog
  metadata:
    model: mars
    positive: up
```

Nemo Medrys

```
[8]: from intake import open_catalog
     yaml = "/home/datawork-lops-siam-moawi/PROJECTS/OSDYN/DATA/CATALOG/INTAKE/medrys_T1Z75Y264X
     cat = open_catalog(yaml)
     cat
```

```
medrys_T1Z75Y264X567:
  args:
    path: /home/datawork-lops-siam-moawi/PROJECTS/OSDYN/DATA/CATALOG/INTAKE/medrys_T1Z75Y26
  description: ''
  driver: intake.catalog.local.YAMLFileCatalog
  metadata:
    model: nemo
    parameters:
      cgrid:
        allowed:
        - grid2D
        - gridT
        - gridS
        - gridU
        - gridV
        default: gridT
        description: data stored according to their location on the Arakawa c-grid
        type: str
    positive: down
```

read yaml from intake

Mars f2_sn

```
%%time
ds = getattr(cat, cat.name).to_dask()
ds
```

```
CPU times: user 308 ms, sys: 12 ms, total: 320 ms
Wall time: 381 ms
```

▸ Dimensions: (**level**: 60, **nj**: 462, **ni**: 1100, **nj_u**: 462, **ni_u**: 1100, **nj_v**: 462, **ni_v**: 1100, **time**: 20407, **ni_f**: 1100, **nj_f**: 462)

nj	(nj)	float32	1.0 2.0 3.0 ... 460.0 46...
nj_f	(nj_f)	float32	1.5 2.5 3.5 ... 460.5 46...
nj_u	(nj_u)	float32	1.0 2.0 3.0 ... 460.0 46...
nj_v	(nj_v)	float32	1.5 2.5 3.5 ... 460.5 46...
time	(time)	datetime64[us]	2015-01-01 ... 2021-1...

▾ Data variables:

Csu_sig	(level)	float32	dask.array<chunksize=...
H0	(nj, ni)	float32	dask.array<chunksize=...
HX	(nj_u, ni_u)	float32	dask.array<chunksize=...
HY	(nj_v, ni_v)	float32	dask.array<chunksize=...
SAL	(time, level, nj, ni)	float32	dask.array<chunksize=...
SIG	(level)	float32	dask.array<chunksize=...
TAUX	(time, nj_u, ni_u)	float32	dask.array<chunksize=...
TAUY	(time, nj_v, ni_v)	float32	dask.array<chunksize=...
TEMP	(time, level, nj, ni)	float32	dask.array<chunksize=...

Nemo Medrys

▸ Dimensions: (**deptht**: 75, **y**: 264, **x**: 567, **time_counter**: 6752)

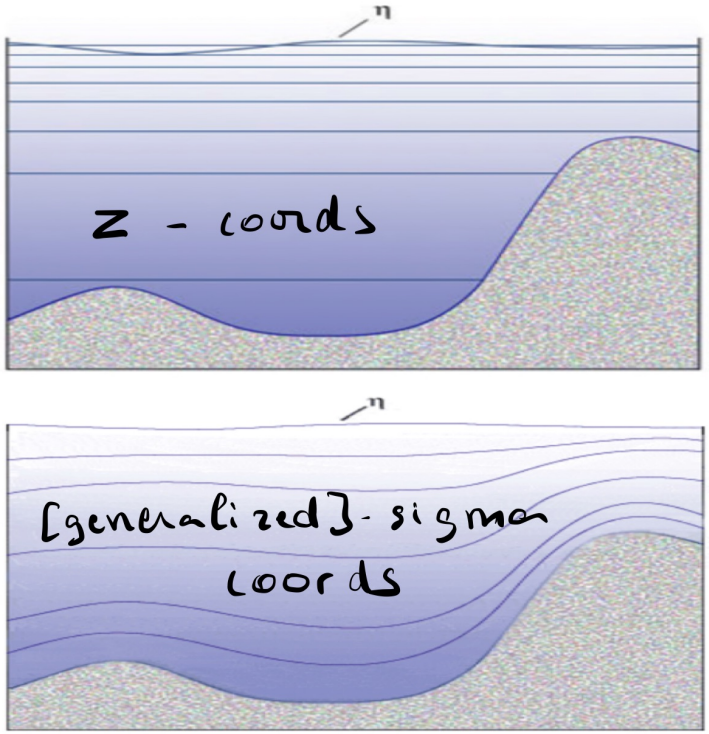
▾ Coordinates:

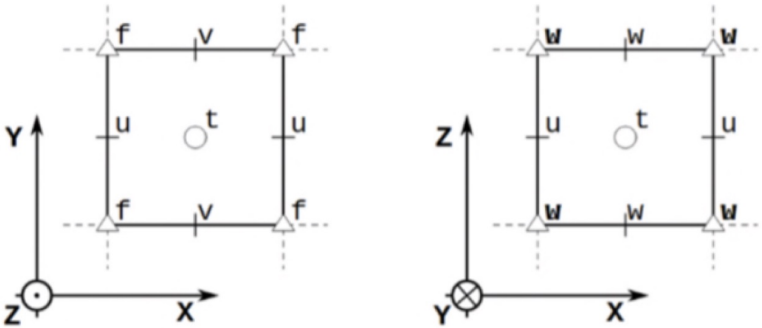
deptht	(deptht)	float32	0.5165 1.622 ... 4.
nav_lat	(y, x)	float32	dask.array<chunk..
nav_lon	(y, x)	float32	dask.array<chunk..
time_counter	(time_counter)	datetime64[us]	1995-01-01T12:...
x	(x)	float64	1.0 2.0 3.0 ... 565..
y	(y)	float64	1.0 2.0 3.0 ... 262..

▾ Data variables:

vodensity	(time_counter, deptht, y, x)	float32	dask.array<chunk..
votemper	(time_counter, deptht, y, x)	float32	dask.array<chunk..

Osdyn readers: example for model outputs



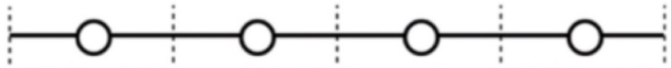


C-grid — horizontal view

C-grid — vertical view

Image from the pycomodo project

XGCM grid convention



center		f[0]	f[1]	…	f[n-1]
left	f[0]	f[1]	…	f[n-1]	
right		f[0]	f[1]	…	f[n-1]
inner		f[0]	…	f[n-2]	
outer	f[0]	f[1]	…	f[n-1]	f[n]

The different possible positions of a variable `f` along an axis.

OSDYN name convention

[medium_] [axe] **name**_adjectif [_niveau] [_direction] [_location]

oce	x	**temp**_insitu	_sfc	_down	_t _u _v _f, _w [_wu...]
atm	y	**cur**_btrope	_bot	_from	
wav	z	**wnd**_bcline	_10m		
obs		**cur**_geos	_top		
ice		**h**_smooth	...		
lake		X_wet			
town		X_tcrit002			
		X_dcrit003			
		X_net			
		X_flx			
		X_along (parallèle à trace)			
		X_cross (perpendiculaire à trace)			
		...			

auto-kerchunk + osdyn reader

Read yaml from intake

```
yaml = os.path.join(path,"marc_f2_1200_sn_T1Z60Y463X1101.yaml")
```

and get the dataset

```
%%time
mc = moawi(yaml)
```

```
Wall time: 3.45 s
```

in

```
%%time
mc.ds
```

Xarray dataset

```
CPU times: user 0 ns, sys: 0 ns, total: 0 ns
Wall time: 5.48 µs
```

xarray.Dataset

▸ Dimensions: (nz_g: 61, nz_c: 60, ny_g: 462, ny_c: 462, nx_g: 1100, nx_c: 1100, nt: 20407)

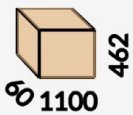
▾ Coordinates:

nz_g	(nz_g)	float64	0.5 1.5 2.5 3.5 ... 58.5 59.5 60.5
lat_t	(ny_c, nx_c)	float64	dask.array<chunksize=(462, 110...
lon_v	(ny_g, nx_c)	float64	dask.array<chunksize=(...
time	(nt)	datetime64[ns]	2015-01-01 ... 2021-12...

▾ Data variables:

salt	(nt, nz_c, ny_c, nx_c)	float32	dask.array<chunksize=(...
temp	(nt, nz_c, ny_c, nx_c)	float32	dask.array<chunksize=(...
xcur	(nt, nz_c, ny_c, nx_g)	float32	dask.array<chunksize=(...
ycur	(nt, nz_c, ny_g, nx_c)	float32	dask.array<chunksize=(...
z_sfc	(nt, ny_c, nx_c)	float32	dask.array<chunksize=(...
sc_cs_t			

Array	Chunk
2.26 TiB	116.32 MiB
(20407, 60, 462, 1100)	(1, 60, 462, 1100)
20408 Tasks	20407 Chunks



Application: 7-year average of sea surface temperature

```
mc.ds["temp"]
```

xarray.DataArray 'temp' (nt: 20407, nz_c: 60, ny_c: 462, nx_c: 1100)

	Array	Chunk
Bytes	2.26 TiB	116.32 MiB
Shape	(20407, 60, 462, 1100)	(1, 60, 462, 1100)
Count	20408 Tasks	20407 Chunks
Type	float32	numpy.ndarray

SST

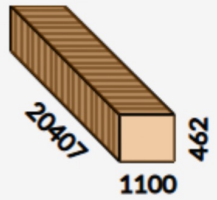
```
# index of the sea surface
ksurf = mc.get_surface_index()
ksurf
```

```
59
```

```
sst = mc.ds["temp"].isel(nz_c=ksurf)
sst
```

xarray.DataArray 'temp' (nt: 20407, ny_c: 462, nx_c: 1100)

	Array	Chunk
Bytes	38.63 GiB	1.94 MiB
Shape	(20407, 462, 1100)	(1, 462, 1100)
Count	40815 Tasks	20407 Chunks
Type	float32	numpy.ndarray



LocalCluster

dask-worker-local

Dashboard: http://127.0.0.1:8787/status	**Workers:** 8
Total threads: 56	**Total memory:** 115.00 GiB
Status: running	**Using processes:** True

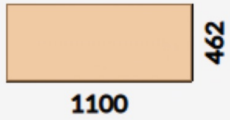
Application: 7-year average of sea surface temperature

```
%%time
surf_tmean = mc.ds["temp"].isel(nz_c=ksurf).mean(dim="nt")
surf_tmean
```

```
CPU times: user 108 ms, sys: 8 ms, total: 116 ms
Wall time: 112 ms
```

xarray.DataArray 'temp' (ny_c: 462, nx_c: 1100)

	Array	Chunk
Bytes	1.94 MiB	1.94 MiB
Shape	(462, 1100)	(462, 1100)
Count	68027 Tasks	1 Chunks
Type	float32	numpy.ndarray




```
%%time
mc.ds["temp"].isel(nz_c=ksurf).mean(dim=["ny_c", "nx_c"])
```

```
CPU times: user 148 ms, sys: 4 ms, total: 152 ms
Wall time: 150 ms
```

xarray.DataArray 'temp' (nt: 20407)

	Array	Chunk
Bytes	79.71 kiB	4 B
Shape	(20407,)	(1,)
Count	81629 Tasks	20407 Chunks
Type	float32	numpy.ndarray


```
%%time
sst_tmean = surf_tmean.compute()
```

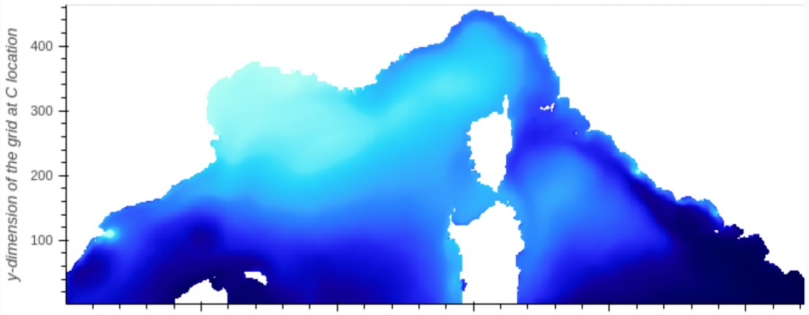
```
CPU times: user 1min 45s, sys: 6.62 s, total: 1min 51s
Wall time: 5min 40s
```

```
%%time
sst_xymean = mc.ds["temp"].isel(nz_c=ksurf).mean(dim=["ny_c", "nx_c"]).compute()
```

```
CPU times: user 1min 51s, sys: 5.87 s, total: 1min 56s
Wall time: 5min 43s
```

```
%%time
sst_tmean.hvplot(datashade=True)
```

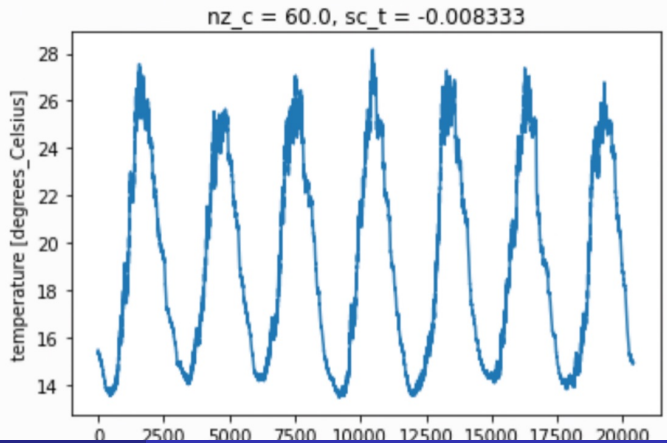
```
CPU times: user 3.04 s, sys: 116 ms, total: 3.15 s
Wall time: 10.7 s
```




```
sst_xymean.plot()
```

```
CPU times: user 28 ms, sys: 8 ms, total: 36 ms
Wall time: 189 ms
```

```
[<matplotlib.lines.Line2D at 0x2aac4b656850>]
```



Application: 7-year average of kinetic energy at 200m

```
yaml = os.path.join(path,"marc_f2_1200_sn_noon_T1Z60Y463X1101.yaml")
```

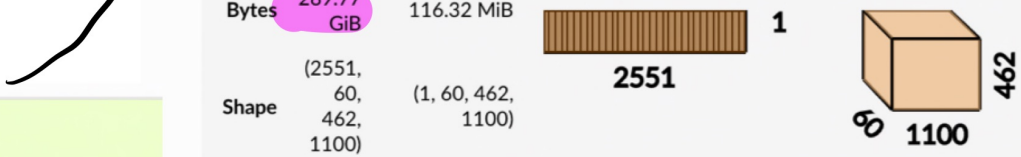
```
%%time
mc = moawi(yaml)
```

```
[WARNING]:  cf.py in name2key - line: 1289
 No osdyn key for this variable : sc_t
[WARNING]:  cf.py in name2key - line: 1289
 No osdyn key for this variable : sc_w
CPU times: user 716 ms, sys: 44 ms, total: 760 ms
Wall time: 1.35 s
```

```
mc.ds.z_t
```

xarray.DataArray 'z_t' (nt: 2551, nz_c: 60, ny_c: 462, nx_c: 1

	Array	Chunk
Bytes	289.77 GiB	116.32 MiB
Shape	(2551, 60, 462, 1100)	(1, 60, 462, 1100)
Count	7670 Tasks	2551 Chunks
Type	float32	numpy.ndarray

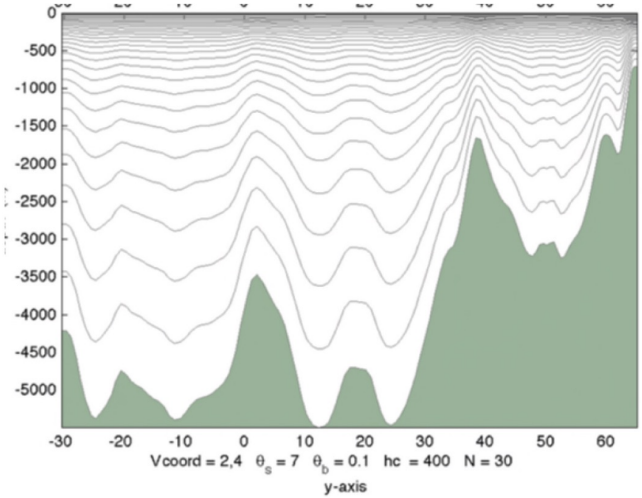


4D variables

```
%%time
mc.set_z(at="t")
# does nothing if depths are already in the dataset
```

for genericity

```
CPU times: user 376 ms, sys: 4 ms, total: 380 ms
Wall time: 374 ms
```



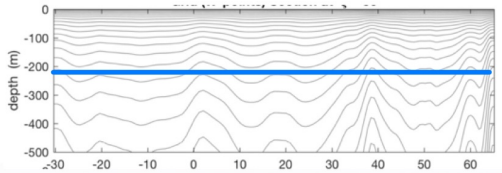

```
mc.ds.xcur
```

xarray.DataArray 'xcur' (nt: 2551, nz_c: 60, ny_c: 462, nx_g:

	Array	Chunk
Bytes	289.77 GiB	116.32 MiB
Shape	(2551, 60, 462, 1100)	(1, 60, 462, 1100)
Count	2552 Tasks	2551 Chunks
Type	float32	numpy.ndarray

Application: 7-year average of kinetic energy at 200m

Projection to -200 m

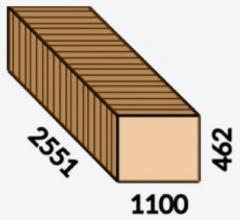



```
%%time
mc.set_var_at_z("xcur", [-200])
mc.ds.xcur_200m
```

```
CPU times: user 2.48 s, sys: 32 ms, total: 2.52 s
Wall time: 2.5 s
```

xarray.DataArray 'xcur_200m' (nt: 2551, ny_c: 462,

	Array	Chunk
Bytes	4.83 GiB	1.94 MiB
Shape	(2551, 462, 1100)	(1, 462, 1099)
Count	58692 Tasks	5102 Chunks
Type	float32	numpy.ndarray




```
mc.ds.z_t
```

xarray.DataArray 'z_t' (nt: 2551, nz_c: 60, ny_c: 462, nx_c: 1

	Array	Chunk
Bytes	289.77 GiB	116.32 MiB
Shape	(2551, 60, 462, 1100)	(1, 60, 462, 1100)
Count	7670 Tasks	2551 Chunks
Type	float32	numpy.ndarray

4D variables

6 nodes 115 GiB

2 workers / nodes

PBSCluster

dask-worker-datarmor

Dashboard: /user/jfleroux/proxy/8787/status	**Workers:** 12
Total threads: 12	**Total memory:** 670.56 GiB

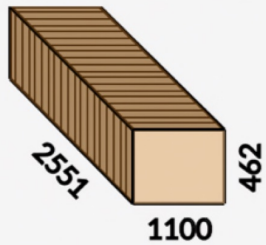
Application: 7-year average of kinetic energy at 200m

```
%%time
ke = kinetic_energy(mc.ds, mc.xgcmgrid, u="xcur_200m", v="ycur_200m")
ke
```

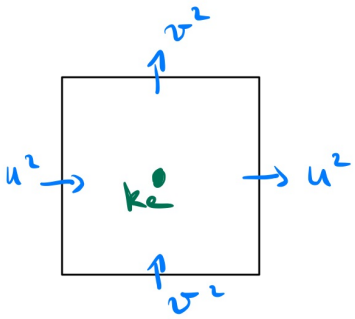
```
CPU times: user 12.4 s, sys: 284 ms, total: 12.7 s
Wall time: 12.6 s
```

xarray.DataArray 'ke_200m' (nt: 2551, ny_c: 462, nx_c: 1100)

	Array	Chunk
Bytes	4.83 GiB	1.93 MiB
Shape	(2551, 462, 1100)	(1, 460, 1098)
Count	316344 Tasks	22959 Chunks
Type	float32	numpy.ndarray



$$ke = \frac{\overline{u^2}^x + \overline{v^2}^y}{2}$$

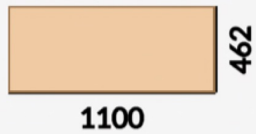



```
%%time
ke_tmean = ke.mean(dim="nt")
ke_tmean
```

```
CPU times: user 88 ms, sys: 0 ns, total: 88 ms
Wall time: 88.9 ms
```

xarray.DataArray 'ke_200m' (ny_c: 462, nx_c: 1100)

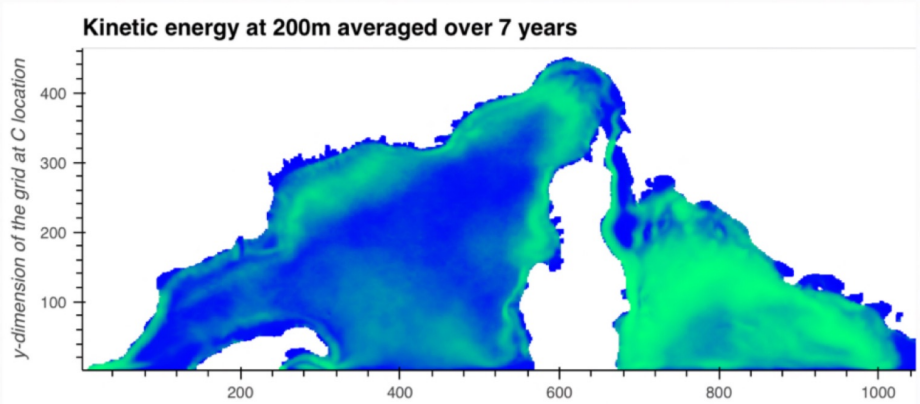
	Array	Chunk
Bytes	1.94 MiB	1.93 MiB
Shape	(462, 1100)	(460, 1098)
Count	346971 Tasks	9 Chunks
Type	float32	numpy.ndarray




```
%%time
ke_tmean_np = ke_tmean.compute()
```

```
CPU times: user 7min 52s, sys: 14.8 s, total: 8min 6s
Wall time: 19min 40s
```

```
ke_tmean_np.hvplot(datashade=True, title="Kinetic energy at 200m averaged over 7 years", cr
```



Summary

auto-kerchunk + intake

Fast access to huge dataset of netcdf files.

osdyn readers

- get rid of specificities in model outputs
- available for NEMO, MARS, Symphonie, CROCO (ocean), MesoNH (atmosphere), WW3 (wave) and any file gridded over a simple grid.

osdyn convention

- genericity of the diagnostics
- bank of diagnostics

osdyn

Takes benefit of pangeo ecosystem (xarray, xgcm, xesmf... and dask parallelization).

A dream ?

[auto]-kerchunk / intake

- Change the chunks when creating the dataset?

dask

- Cluster according to the memory.
- Balance between the numbers of tasks and the size of the chunks.
- Keep track of the fields of the same record?

MERCI A VOUS TOUS POUR VOTRE ATTENTION