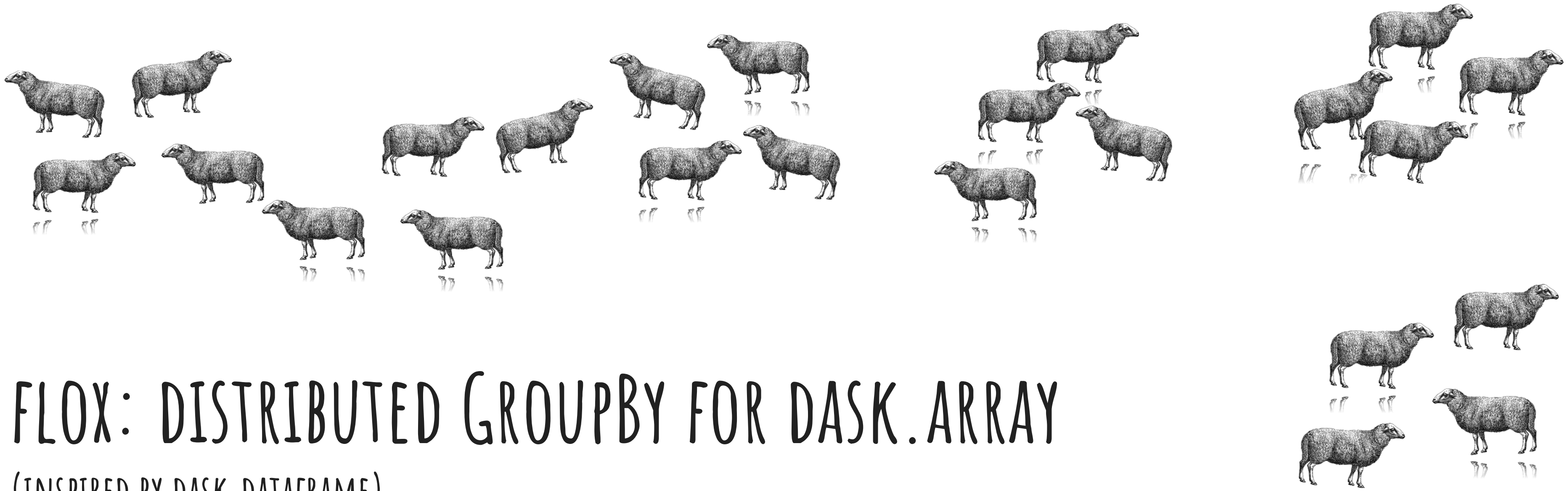


FLOX: DISTRIBUTED GROUPBY FOR DASK.ARRAY

(INSPIRED BY DASK.DATAFRAME)

Funding:
NASA-ACCESS 80NSSC18M0156 (PI J. Hamman)
NSF / NCAR Earth System Data Science Initiative

PANGEO

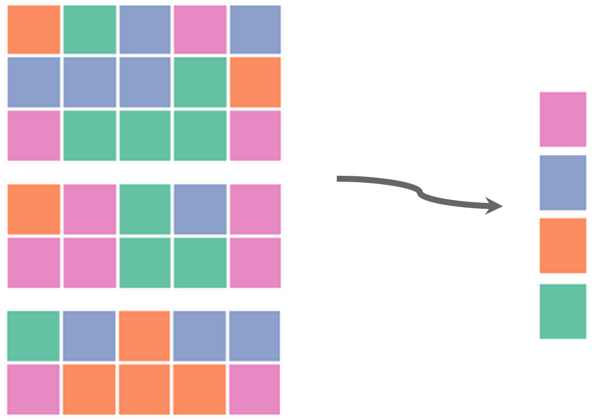
dcherian/flox

Deepak Cherian
National Center for Atmospheric Research
@dcherian | cherian.net

"GroupBy" or split-apply-combine: a very common pattern

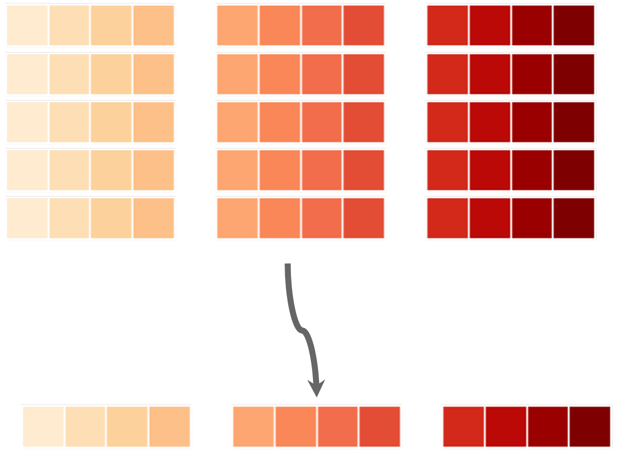
"Binning" or "histogramming"
Or "compositing"

```
groupby(crop_type).sum(),
groupby_bins(temperature, bins).mean()
groupby(enso_phase).mean()
```



"Climatology" or monthly means

```
groupby("time.month")
```



Resampling: daily to monthly

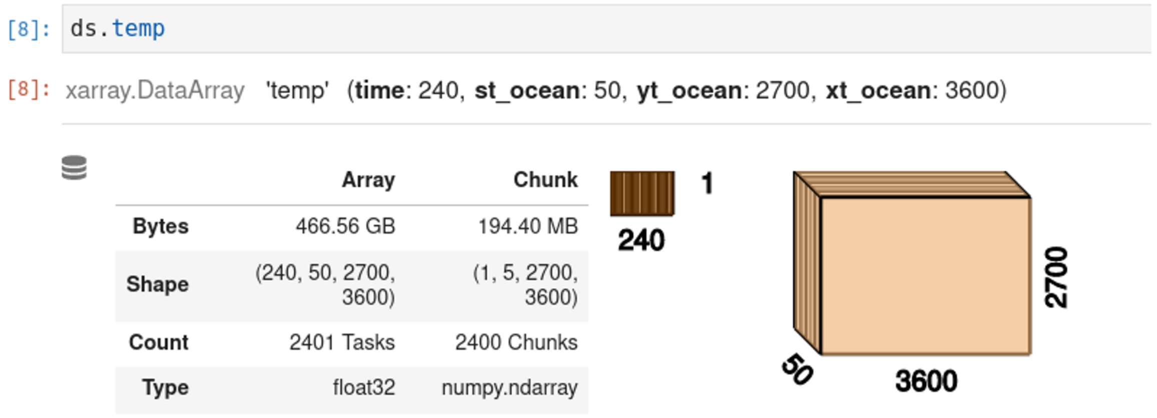
```
resample(time="M").mean()
```

dcherian/flox

One Dataset, Two GroupBys

GFDL CM2.6 OCEAN
Monthly means

```
chunks =
  {"time": 1,
   "Z": 5,
   "Y": 2700,
   "X": 3600}
```



```
[8]: ds.temp
```

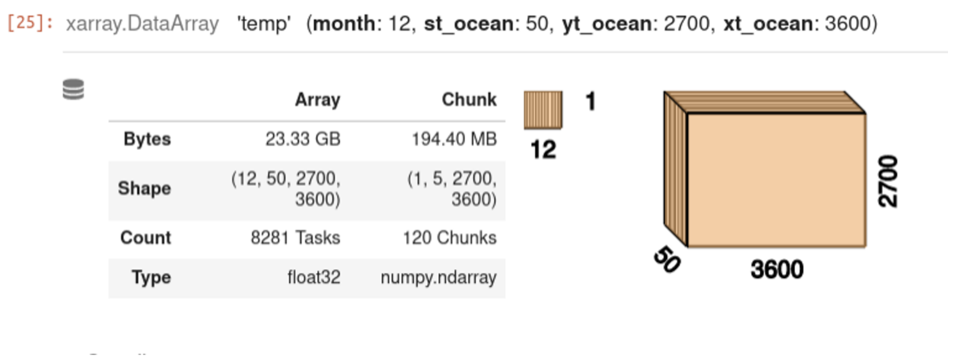
```
[8]: xarray.DataArray 'temp' (time: 240, st_ocean: 50, yt_ocean: 2700, xt_ocean: 3600)
```

	Array	Chunk
Bytes	466.56 GB	194.40 MB
Shape	(240, 50, 2700, 3600)	(1, 5, 2700, 3600)
Count	2401 Tasks	2400 Chunks
Type	float32	numpy.ndarray

PANGEO

One Dataset, Two GroupBys: Standard Xarray

‘Monthly climatology’
ds.temp.groupby(“time.month”).mean()
8281 tasks

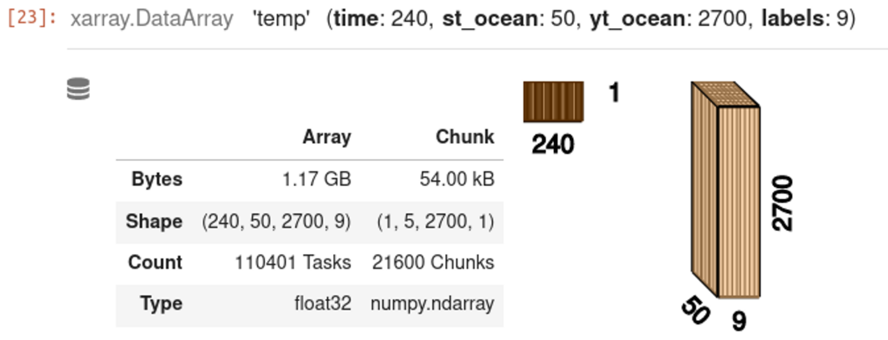


[25]: xarray.DataArray 'temp' (**month**: 12, **st_ocean**: 50, **yt_ocean**: 2700, **xt_ocean**: 3600)

	Array	Chunk
Bytes	23.33 GB	194.40 MB
Shape	(12, 50, 2700, 3600)	(1, 5, 2700, 3600)
Count	8281 Tasks	120 Chunks
Type	float32	numpy.ndarray

ONE GROUP PER BLOCK

‘Regional mean’: longitudinal bins
ds.temp.groupby(regions).mean()
110401 tasks!



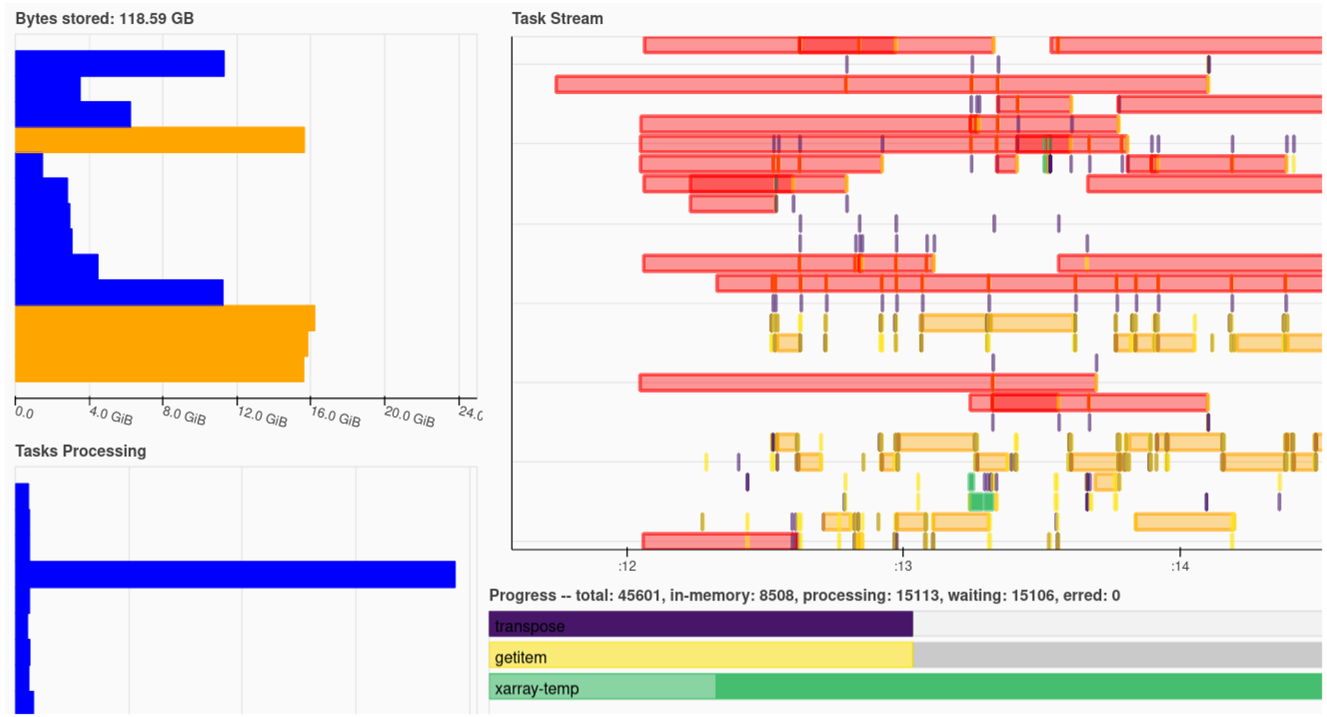
[23]: xarray.DataArray 'temp' (**time**: 240, **st_ocean**: 50, **yt_ocean**: 2700, **labels**: 9)

	Array	Chunk
Bytes	1.17 GB	54.00 kB
Shape	(240, 50, 2700, 9)	(1, 5, 2700, 1)
Count	110401 Tasks	21600 Chunks
Type	float32	numpy.ndarray

MANY GROUPS PER BLOCK

Let's Try It... maybe... not really...

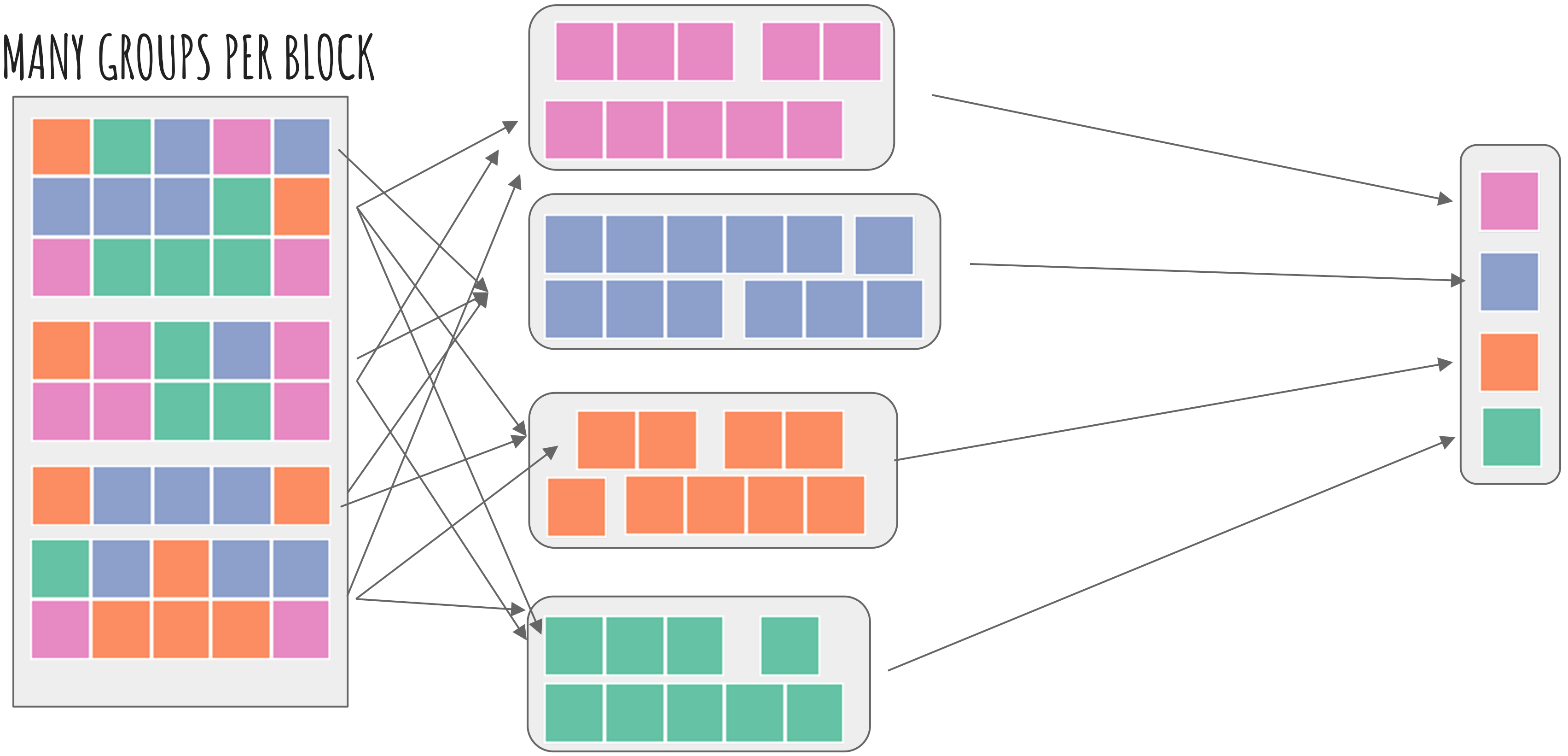
"Regional mean": `ds.temp.groupby(regions).mean()`



Progress -- total: 45601, in-memory: 8508, processing: 15113, waiting: 15106, erred: 0

Xarray's GroupBy: communicate + reduce

Many groups per block



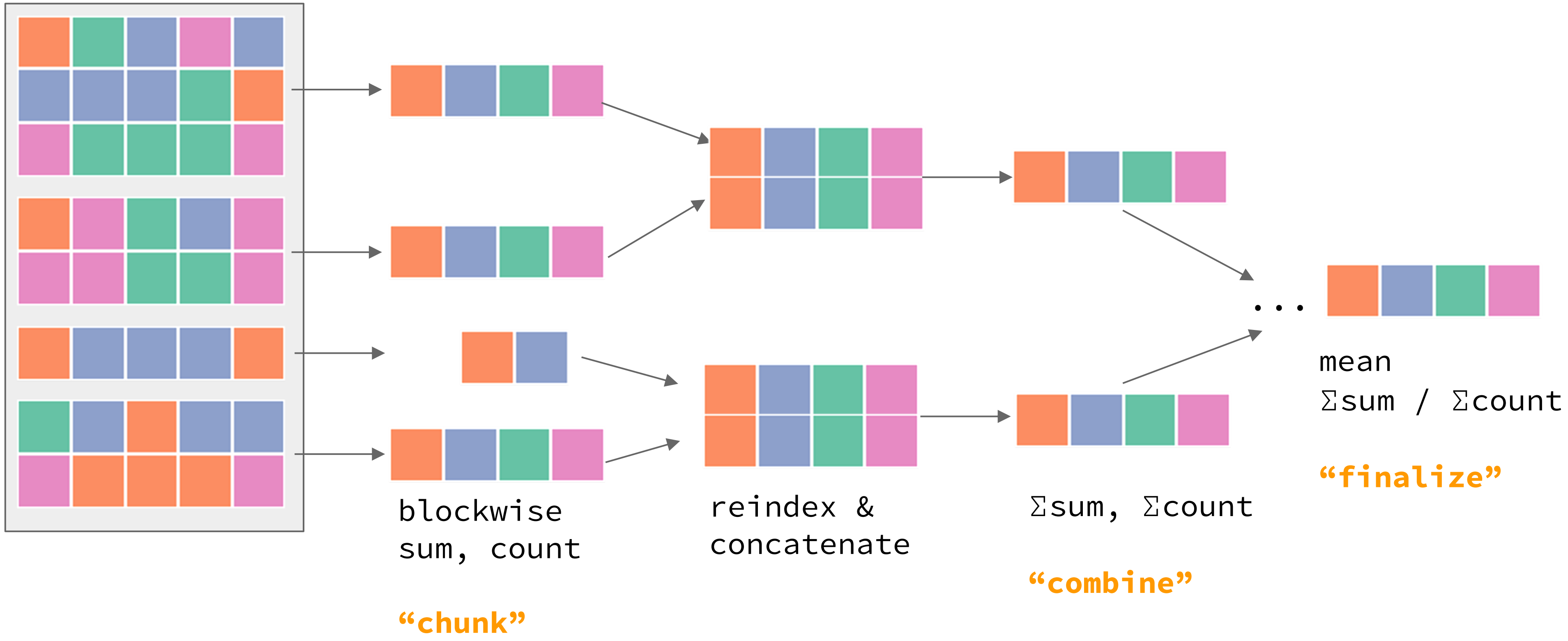
“My dask-workers could handle ten times the chunks if they weren’t busy apologizing for your codebase”

≈ Gilfoyle, Silicon Valley

dcherian/flox

Let's try map-reduce

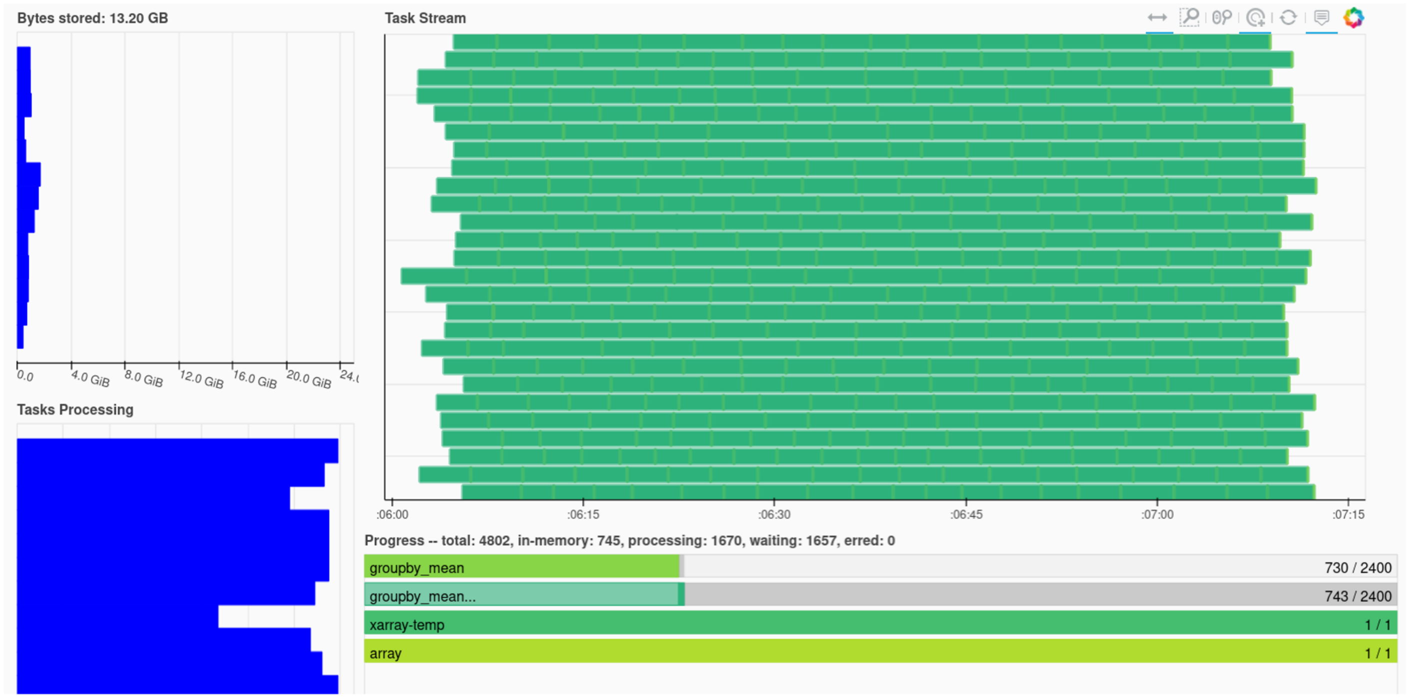
Many groups per block



dcherian/flox

Does it work? 17Gb memory… 11 mins with 12 workers

```
>> flox.xarray.xarray_reduce(..., func="mean")
```



Progress -- total: 4802, in-memory: 745, processing: 1670, waiting: 1657, erred: 0

```
groupby_mean        730 / 2400
groupby_mean...     743 / 2400
xarray-temp         1 / 1
array               1 / 1
```

dcherian/flox

Works when a blockwise reduction is Effective

'Mo

[52

ONE GROUP PER BLOCK

'Regional mean'

[28]:

	Array	Chunk
Bytes	1.17 GB	486.00 kB
Shape	(240, 50, 2700, 9)	(1, 5, 2700, 9)
Count	9602 Tasks	2400 Chunks
Type	float32	numpy.ndarray

MANY GROUPS PER BLOCK

Big Problem: Not Great for TIME Grouping

(1) We know the time vector, so we know where the groups are
(2) The groups have patterns

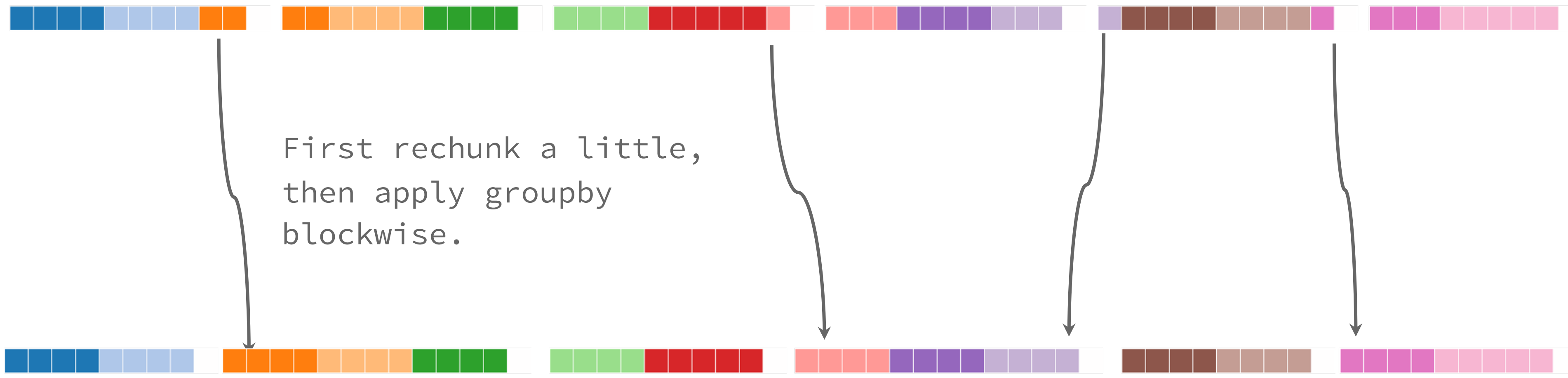
Example 1: resampling from daily to monthly

Example 2: “monthly climatology”: groupby(“time.month”)

"Blockwise" Resampling (copied from dask.dataframe)

Groups are sequential, *approximately* equal length

E.g. resampling from daily to monthly frequency

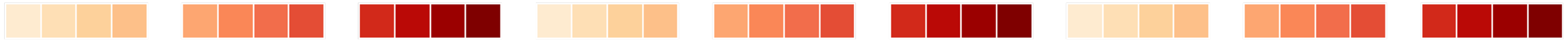


flox.xarray.xarray_reduce(..., **method="blockwise"**)
flox.xarray.rechunk_for_blockwise(...)

Climatologies

e.g. groupby("time.month")

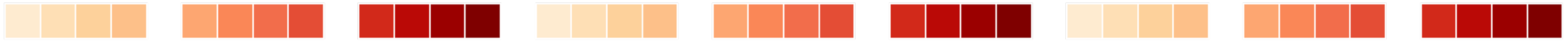
- Groups are sequential
    - Jan is always before Feb



- And periodic!
    - So we can't use the resampling strategy

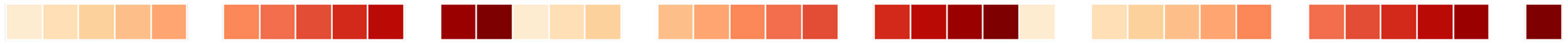
The "cohorts" Idea: groups of groups

Idea: Let's extract groups that tend to occur together: "cohorts"



```
>>> flox.core.find_group_cohorts(labels, array.chunks[-1]))
[[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]  # 3 cohorts
```

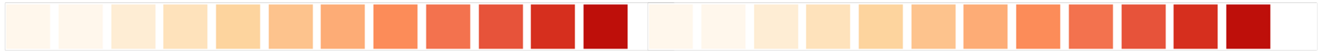
```
for cohort in cohorts:
    # subset array to pick only groups in cohort
    # map-reduce
```



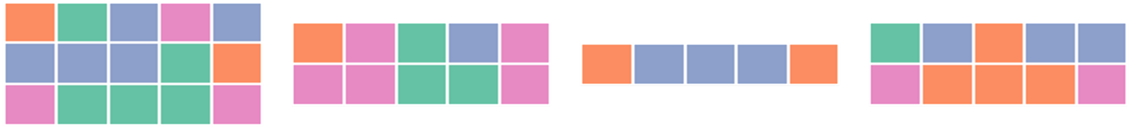
```
>>> flox.core.find_group_cohorts(labels, array.chunks[-1]))
[[1], [2, 3], [4, 5], [6], [7, 8], [9, 10], [11], [12]]  # 8 cohorts
```

The Cohorts strategy generalizes nicely

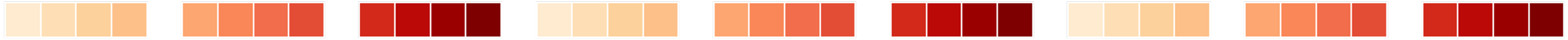
Recreates Xarray's current strategy when that is optimal.
E.g. one month per block



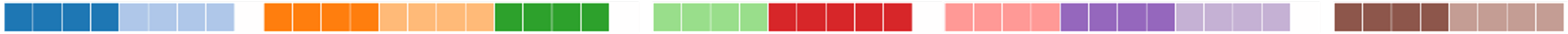
Could "map-reduce" this distribution of groups, if generalized to nD



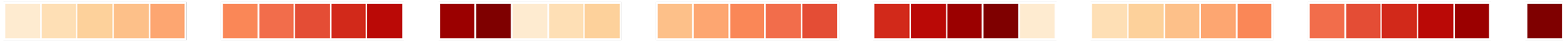
Can avoid unnecessary communication with "map-reduce"



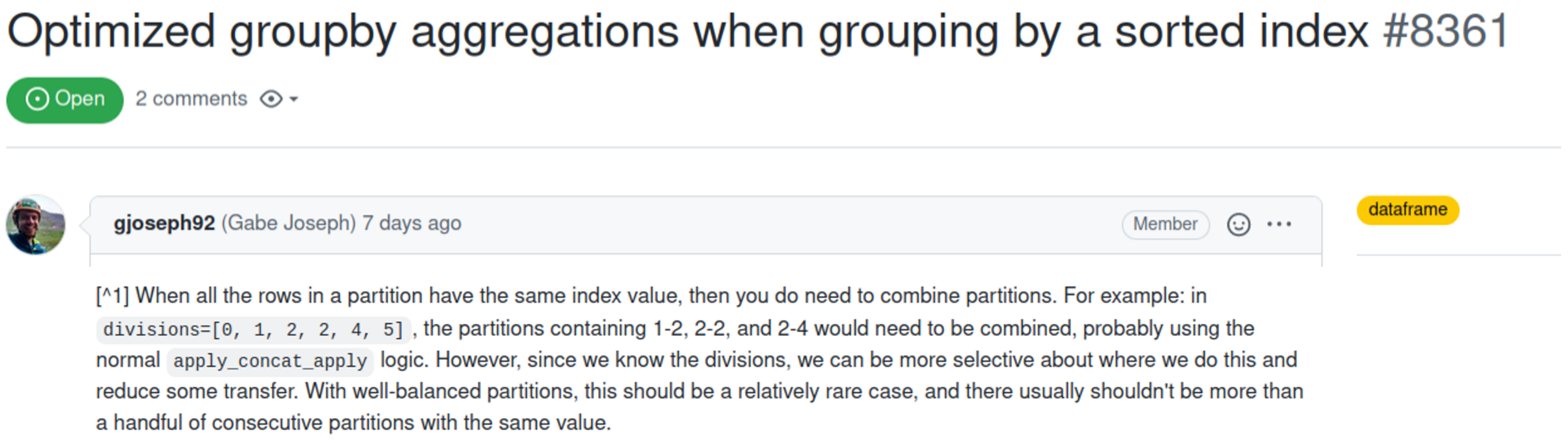
Works "blockwise" for resampling **after** rechunking



"Out of phase group pattern"???? map-reduce ????

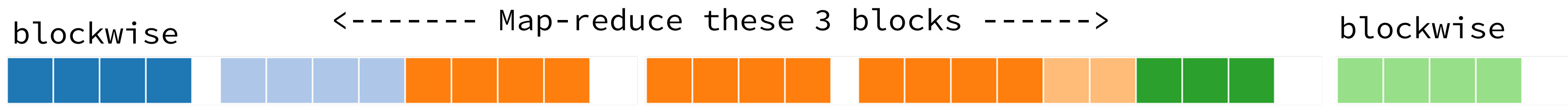


This is the same idea! Minimize communication



Optimized groupby aggregations when grouping by a sorted index #8361

[^1] When all the rows in a partition have the same index value, then you do need to combine partitions. For example: in `divisions=[0, 1, 2, 2, 4, 5]`, the partitions containing 1-2, 2-2, and 2-4 would need to be combined, probably using the normal `apply_concat_apply` logic. However, since we know the divisions, we can be more selective about where we do this and reduce some transfer. With well-balanced partitions, this should be a relatively rare case, and there usually shouldn't be more than a handful of consecutive partitions with the same value.



Current Status: Try it out!

- https://github.com/dcherian/flox
  - Pip / conda-forge
  - Beta quality

- Integration into xarray
  - https://github.com/pydata/xarray/pull/5734 Tests pass!
  - ds.groupby(“time.month”).mean(method=”cohorts”, engine=”numba”)

- There doesn’t seem to be one optimal strategy.
  - Depends on how groups are distributed across blocks
  - Needs testing / benchmarking
  - Document “lessons learned” discourse.pangeo.io

- Are there other common group patterns that we could optimize for?