A new partitioning approach for ECMWF's Integrated Forecasting System (IFS)

George.Mozdzynski@ecmwf.int

Outline

- **IFS**
  - highly optimized (MPI & OpenMP)
  - depressingly flat profile
- **Partitioning**
  - Grid point space
    - 2D (NPRGPNS x NPRGPEW = NPROC)
    - eq_regions (NPROC)
  - Fourier space, 2D (lats, levels)
  - Spectral space, 2D (waves, levels)
- **Performance (2D v. eq_regions)**
  - Forecast model
  - 4D-Var

Integrated Forecasting System (IFS)

- IFS 1992 - today
  - Collaboration between Meteo France and ECMWF
  - Source ~ 1.8 million lines
  - Fortran 95, some C
  - Good performance on scalar and vector systems
- IFS model characteristics:
  - Spectral
  - Semi-implicit
  - Semi-Lagrangian

IFS - Parallelised using 'mixed' MPI and OpenMP

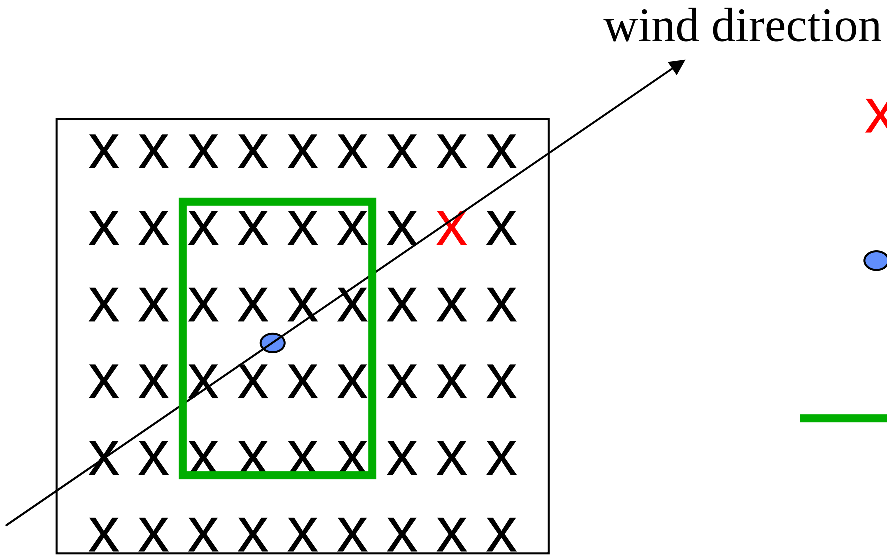
MPI communications

- Transpositions
  - Between Grid point, Fourier and Spectral spaces
- Wide halo exchange
  - Semi Lagrangian method
  - Radiation grid interpolation
- Long messages
- Typically MPI_ISEND/RECV/WAITALL or collective

OpenMP

- Shared memory nodes
- Memory efficient
- Use 4/8 threads

IFS – Semi-Lagrangian Advection



x arrival point at t(n)

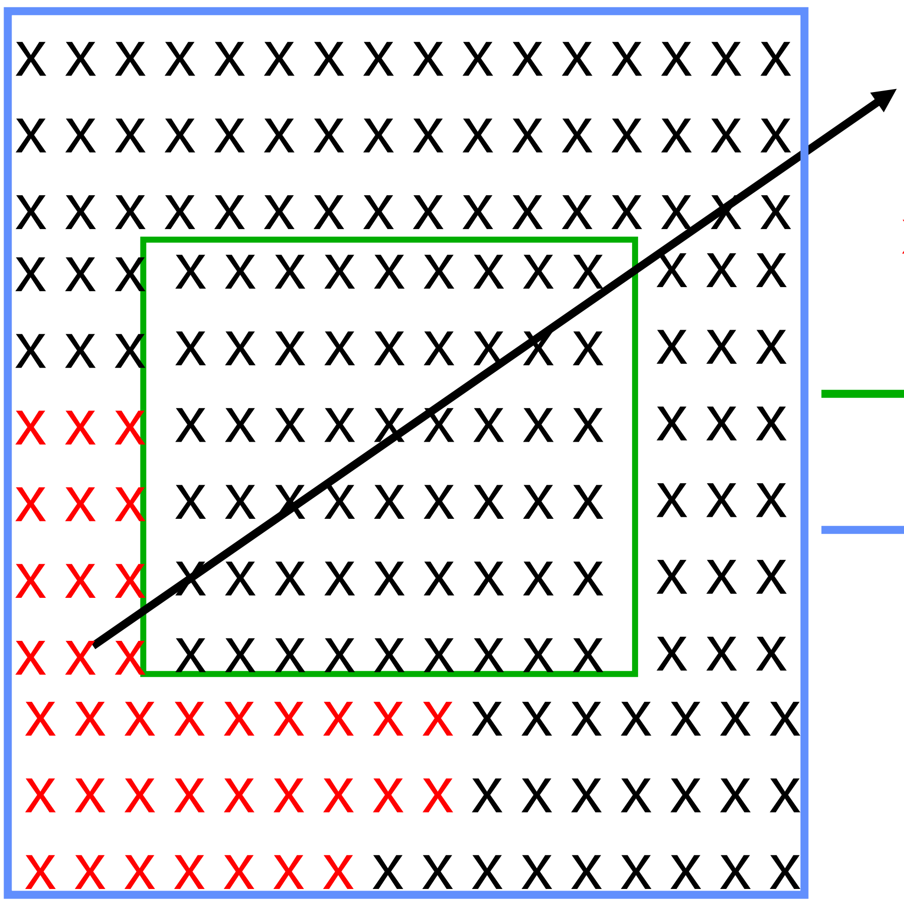
departure point at t(n-1)

interpolation stencil

Horizontal section

Full interpolation in 3-D is 32 point

IFS – Semi-Lagrangian 'On Demand'

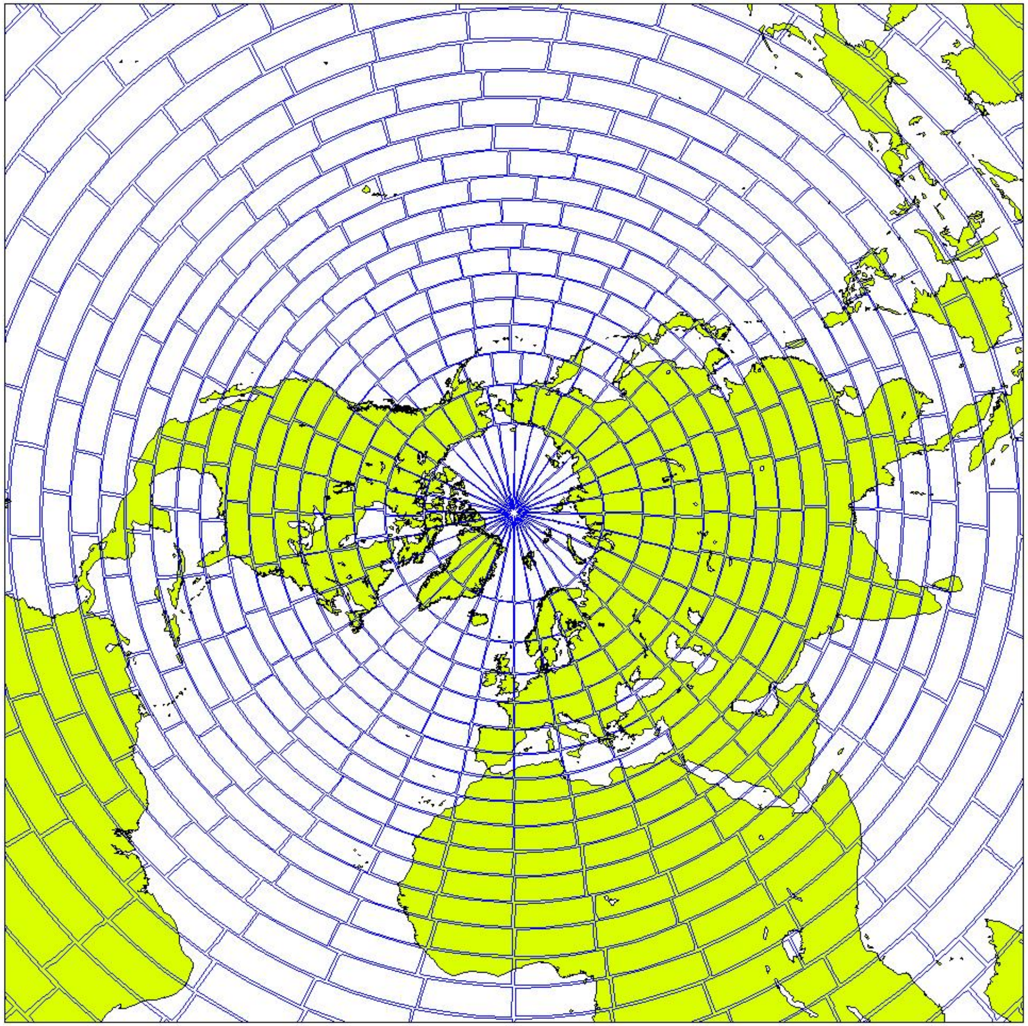


x points needed in halo

points located on task n

halo

T_L799 1024 tasks 2D partitioning



2D partitioning results in non-optimal Semi-Lagrangian comms requirement at poles and equator!

Square shaped partitions are better than rectangular shaped partitions.

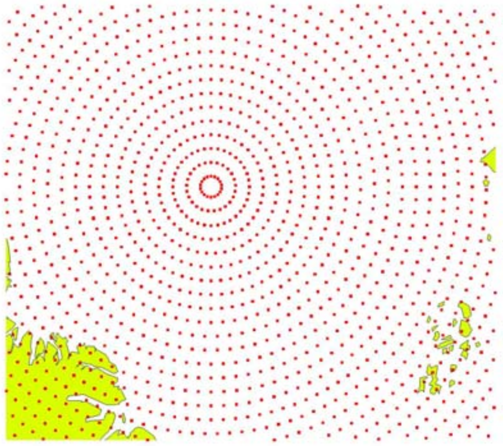
Model / Radiation Grids

- **Radiation computations are expensive**
- **To reduce this cost we,**
  - Run radiation computations every hour
    - every 5th timestep for T_L799 model
  - Run radiation computations on a courser grid T_L399
    - requires interpolation
- **Two interpolation possibilities**
  - Gather global fields to different tasks (non-scalable)
    - global comms is bad; # fields can be less then # tasks
  - Perform interpolation with only local comms for halo (scalable)
    - implemented in IFS this way

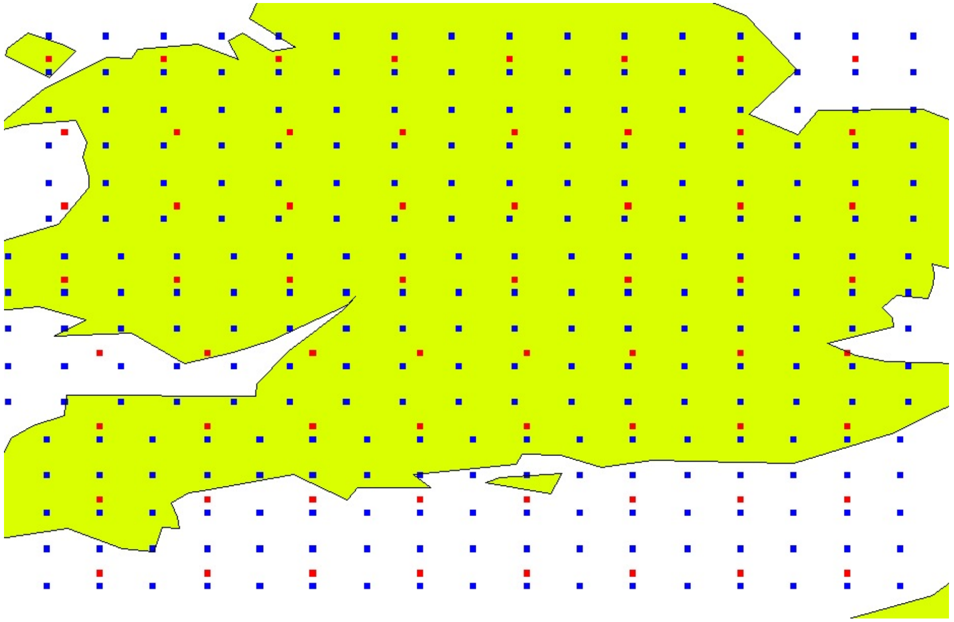
Reduced grids (linear)

```
&NAMRGRI
 NRGRI(1)=   18,
 NRGRI(2)=   25,
 NRGRI(3)=   36,
 NRGRI(4)=   40,
 NRGRI(5)=   45,
 NRGRI(6)=   50,
 NRGRI(7)=   60,
 NRGRI(8)=   64,
 NRGRI(9)=   72,
 NRGRI(10)=   72,
 NRGRI(11)=   75,
 NRGRI(12)=   81,
 NRGRI(13)=   90,
 NRGRI(14)=   96,
  ...

 NRGRI(200)=  800,
  ...

 NRGRI(398)=   36,
 NRGRI(399)=   25,
 NRGRI(400)=   18,
/
```

T_L399



note only factors 2, 3, and 5
for fourier transforms

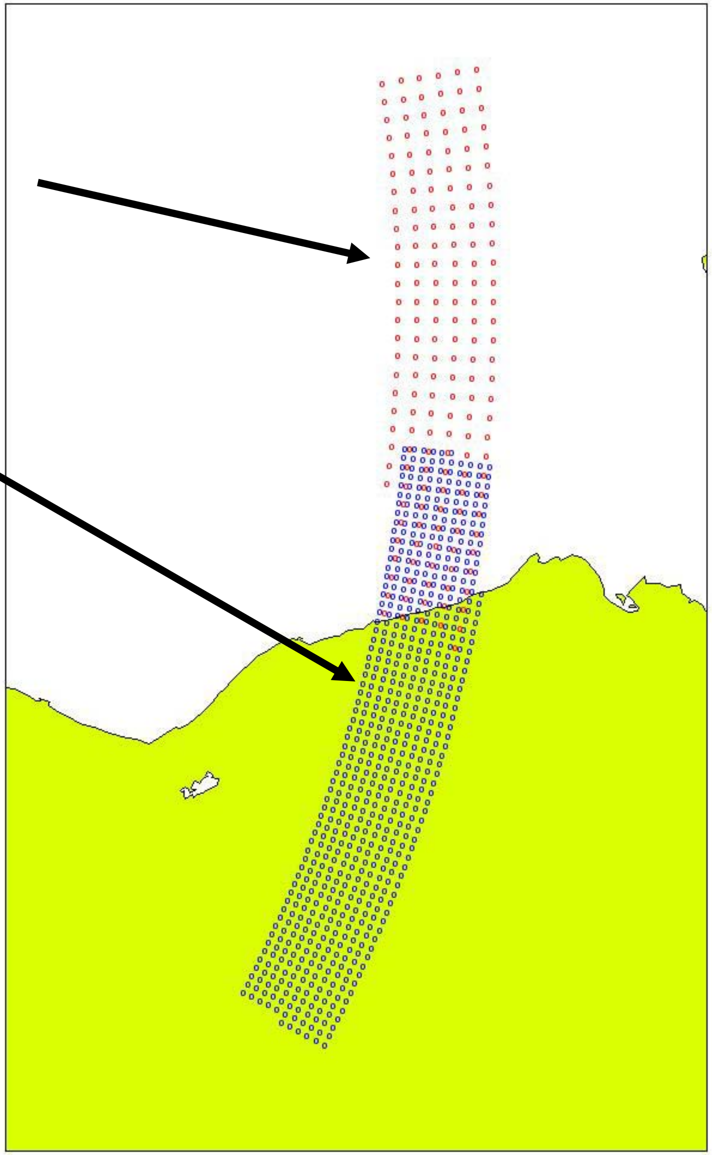


T799 model grid (blue)

T399 radiation grid (red)

PE=293, Radiation Grid T_L255

PE=293, Model Grid T_L511



Model and Radiation grids for same partition are offset geographically, because

- Use of reduced grid (linear)
- T_L255 is not a projection of T_L511
- Long thin partitions make matters worse

eq_regions algorithm

A PARTITION OF THE UNIT SPHERE INTO REGIONS OF EQUAL AREA AND SMALL DIAMETER

PAUL LEOPARDI *

Abstract. The recursive zonal equal area (EQ) sphere partitioning algorithm is a practical algorithm for partitioning higher dimensional spheres into regions of equal area and small diameter. This paper describes the partition algorithm and its implementation in Matlab, provides numerical results and gives a sketch of the proof of the bounds on the diameter of regions. A companion paper [13] gives details of the proof.
Keywords: Sphere, partition, area, diameter, zone.

1. Introduction. For dimension d, the unit sphere $\mathbb{S}^d$ embedded in $\mathbb{R}^{d+1}$ is

$$\mathbb{S}^d := \left\{ x \in \mathbb{R}^{d+1} \,\middle|\, \sum_{k=1}^{d+1} x_k^2 = 1 \right\}. \qquad (1.1)$$

This paper describes a partition of the unit sphere $\mathbb{S}^d \subset \mathbb{R}^{d+1}$ which is here called the recursive zonal equal area (EQ) partition. The partition $\mathrm{EQ}(d, N)$ is a partition of the unit sphere $\mathbb{S}^d$ into N regions of equal area and small diameter. It is defined via the algorithm given in Section 3.

Figure 1.1 shows an example of the partition $\mathrm{EQ}(2, 33)$, the recursive zonal equal area partition of $\mathbb{S}^2$ into 33 regions. A movie showing the build-up of an example of the partition $\mathrm{EQ}(3, 99)$ is available at http://web.maths.unsw.edu.au/~leopardi/.

For the purposes of this paper, we define an equal area partiton of $\mathbb{S}^d$ in the following way.

DEFINITION 1.1. *An* equal area partition *of $\mathbb{S}^d$ is a nonempty finite set P of* regions, *which are closed Lebesgue measurable subsets of $\mathbb{S}^d$ such that*

1. *the regions cover $\mathbb{S}^d$, that is*

$$\bigcup_{R \in P} R = \mathbb{S}^d;$$

2. *the regions have equal area, with the Lebesgue area measure σ of each $R \in P$ being*

$$\sigma(R) = \frac{\sigma(\mathbb{S}^d)}{|P|},$$

where $|P|$ denotes the cardinality of P; and

3. *the boundary of each region has area measure zero, that is, for each $R \in P$, $\sigma(\partial R) = 0$.*

Note that conditions 1 and 2 above imply that the intersection of any two regions of P has measure zero. This in turn implies that any two regions of P are either disjoint or only have boundary points in common. Condition 3 excludes pathological cases which are not of interest in this paper.

This paper considers the Euclidean diameter of each region, defined as follows.

DEFINITION 1.2. *The* diameter *of a region $R \in \mathbb{S}^d \subset \mathbb{R}^{d+1}$ is*

$$\operatorname{diam} R := \sup\{e(x, y) \mid x, y \in R\},$$

*paul.leopardi@unsw.edu.au, School of Mathematics, University of New South Wales.

1

Developed by Paul Leopardi et al. , School of Mathematics, Univ. of New South Wales, Australia.

Paper:

http://www.maths.unsw.edu.au/applied/files/2005/amr05_18.pdf

eq_regions algorithm

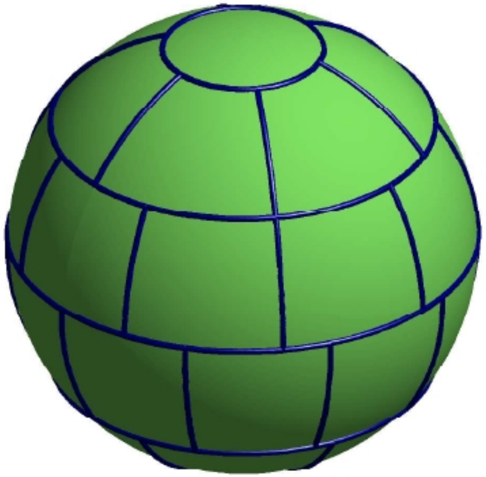


FIG. 1.1. *Partition* EQ(2, 33)

where $e(x, y)$ *is the* $\mathbb{R}^{d+1}$ *Euclidean distance* $\|x - y\|$.

The following definitions are specific to the main theorems stated in this paper.

DEFINITION 1.3. *A set* Z *of partitions of* $\mathbb{S}^d$ *is said to be* diameter-bounded *with* diameter bound $K \in \mathbb{R}_+$ *if for all* $P \in Z$, *for each* $R \in P$,

$$\operatorname{diam} R \leqslant K\,|P|^{-1/d}.$$

DEFINITION 1.4. *The set of recursive zonal equal area partitions of* $\mathbb{S}^d$ *is defined as*

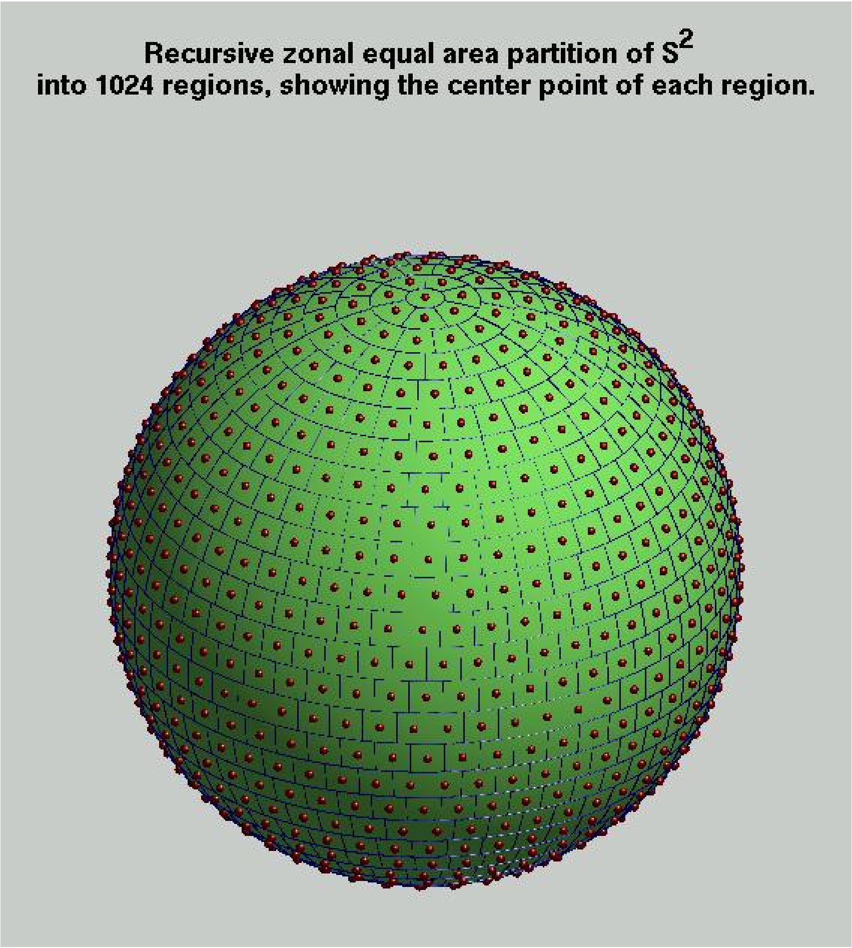
$$\mathrm{EQ}(d) := \{\mathrm{EQ}(d, N) \mid N \in \mathbb{N}_+\}. \qquad (1.2)$$

where $\mathrm{EQ}(d, N)$ *denotes the recursive zonal equal area partition of the unit sphere* $\mathbb{S}^d$ *into* N *regions, which is defined via the algorithm given in Section 3.*

This paper claims that the partition defined via the algorithm given in Section 3 is an equal area partition which is diameter bounded. This is formally stated in the following theorems.

THEOREM 1.5. *For* $d \geqslant 1$ *and* $N \geqslant 1$, *the partition* $\mathrm{EQ}(d, N)$ *is an equal area partition of* $\mathbb{S}^d$.

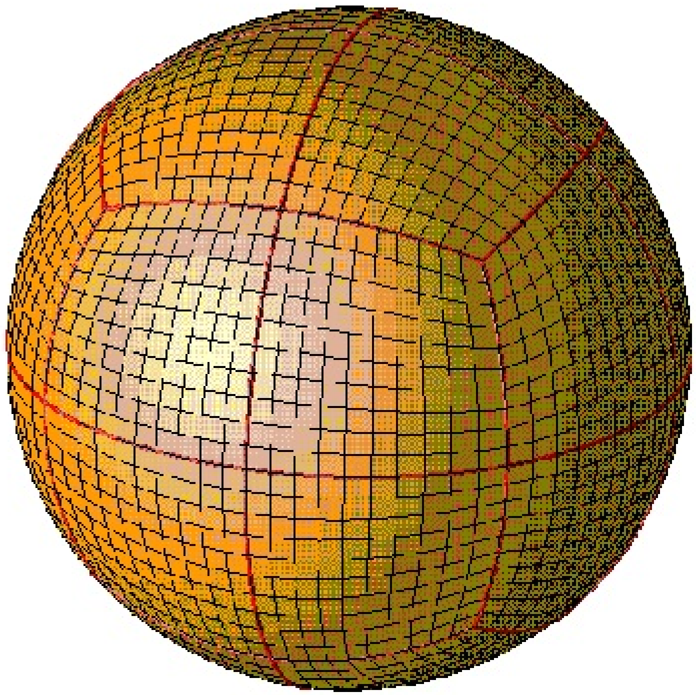
THEOREM 1.6. *For* $d \geqslant 1$, $\mathrm{EQ}(d)$ *is diameter-bounded in the sense of Definition 1.3.*



Why eq_regions?

- **eq_regions partitioning is 'broadly' similar to existing IFS 2D partitioning**
  - 2D 'A-Sets' similar to eq_regions 'bands'
  - 2D partitioning good for a regular lat - lon grid
  - eq_regions partitioning more suited to a reduced grid
- **Only one new data structure required**
  - N_REGIONS
- **Code changes straightforward (example follows)**
- **eq_regions partitioning works for any number of tasks and not just task numbers that have 'nice' factors**

Other partitioning approaches: e.g. quadrangles



Difficult to implement in IFS (but not impossible).

Nothing in common with 2D partitioning approach.

C. Lemaire/J.C. Weill, March 23 2000, Partitioning the sphere with constant area quadrangles, 12th Canadian Conference on Computational Geometry

Example - gather/scatter loops

ORIGINAL (2D)

```
DO JB=1,NPRGPEW
  DO JA=1,NPRGPNS

    ...

  ENDDO
ENDDO
```

NEW (eq_regions and 2D)

```
DO JA=1,N_REGIONS_NS
  DO JB=1,N_REGIONS(JA)

    ...

  ENDDO
ENDDO
```

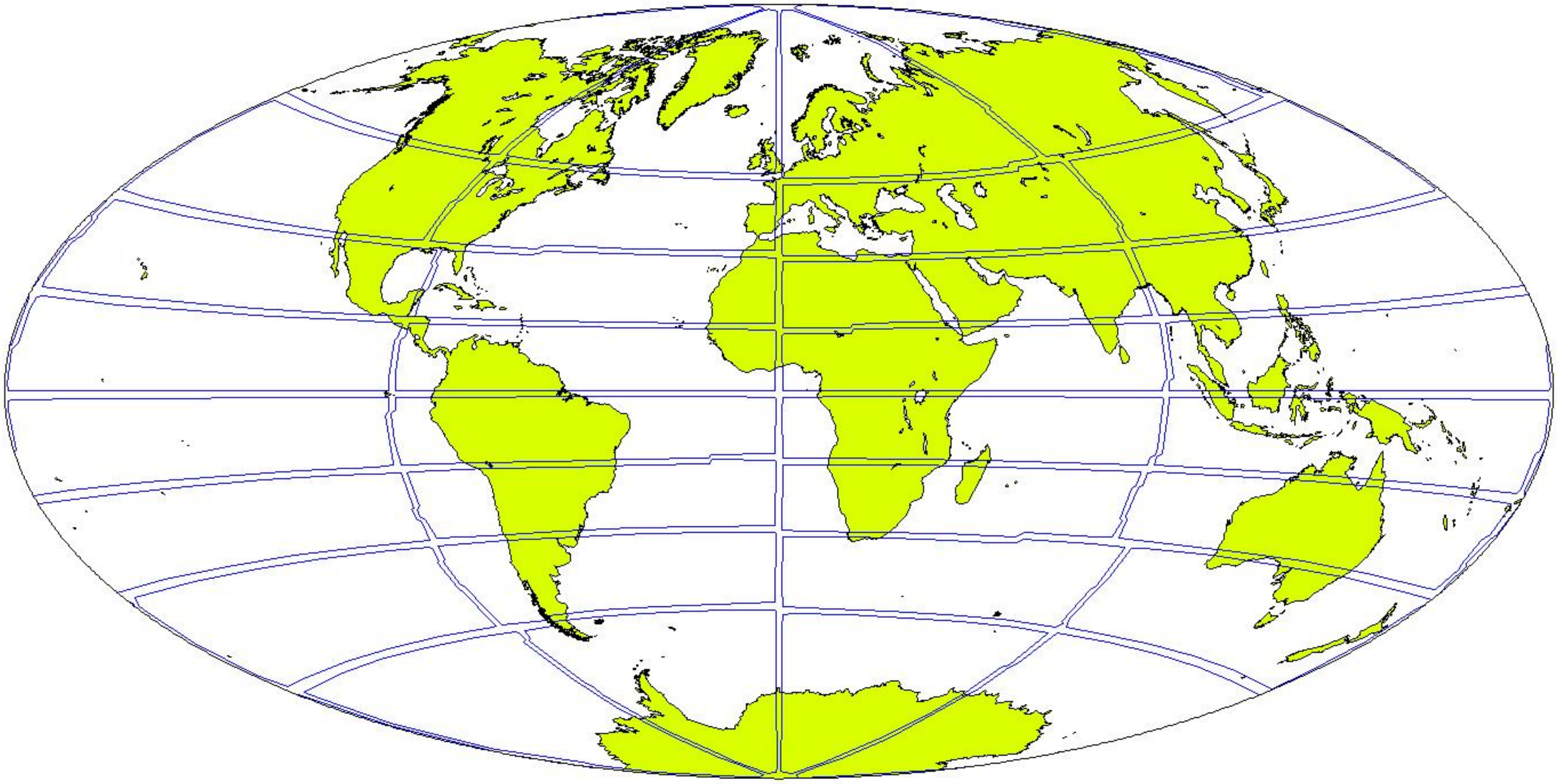
In the future we expect to collapse the 'New' loop structure into a single loop to improve OpenMP performance (where applicable).

eq_regions IFS implementation

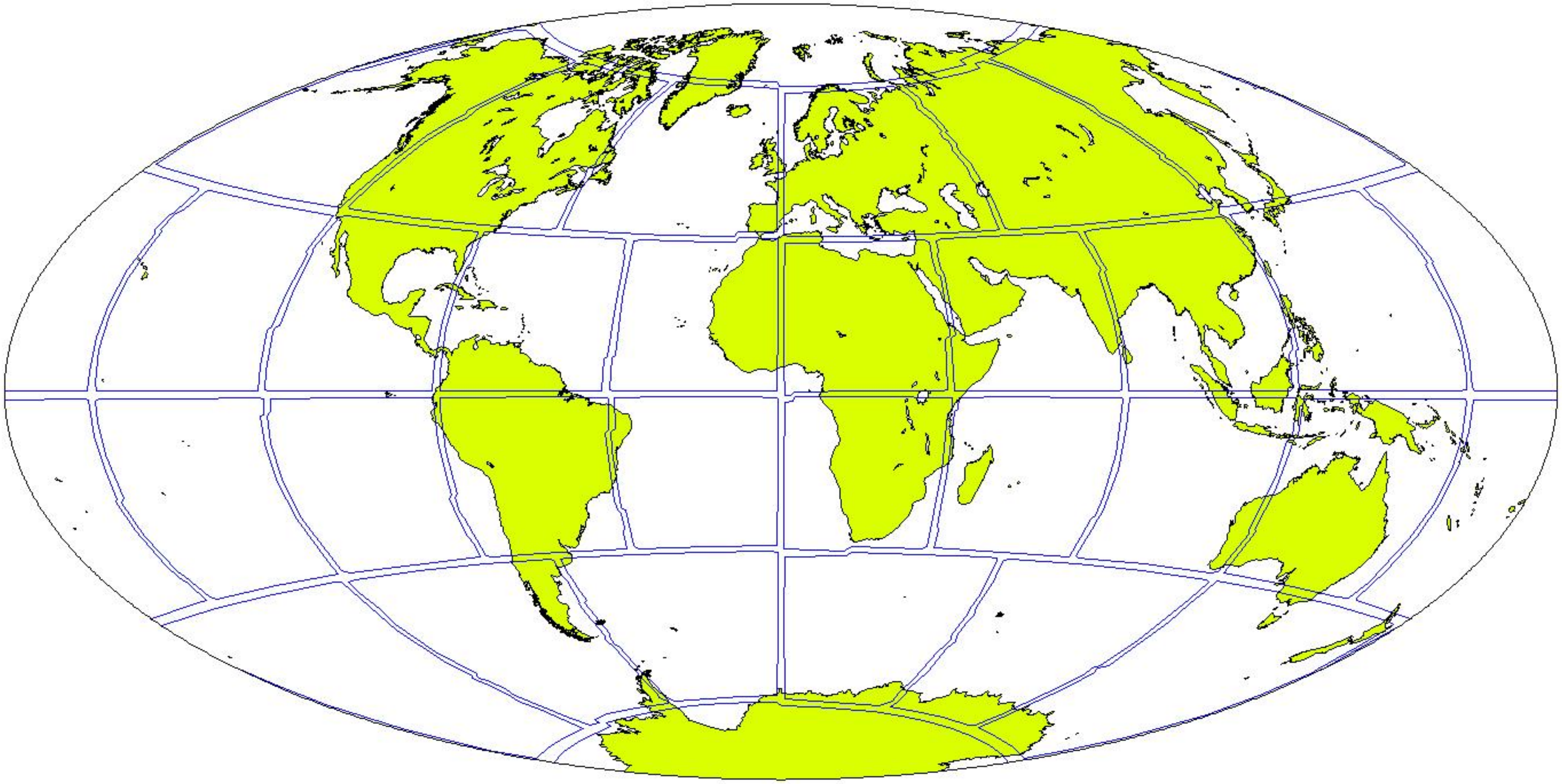
- eq_regions algorithm adapted for IFS reduced grids to give ideal load balance of grid points per partition
- IFS reduced grids have approx 20% more grid points at polar latitudes than equatorial latitudes
- Adaptation is to discard angular partitioning information from original eq_regions algorithm such as lat(beg,end), lon(beg,end)
- We simply use the high level partitioning information,
  - # of bands (called collars in eq_regions speak) and
  - # of partitions per band
- The 'nitty gritty' detailed partitioning is then done in the IFS transform (trans) library

2D partitioning T159 32 tasks (NS=8 x EW=4)

Aitoff projection

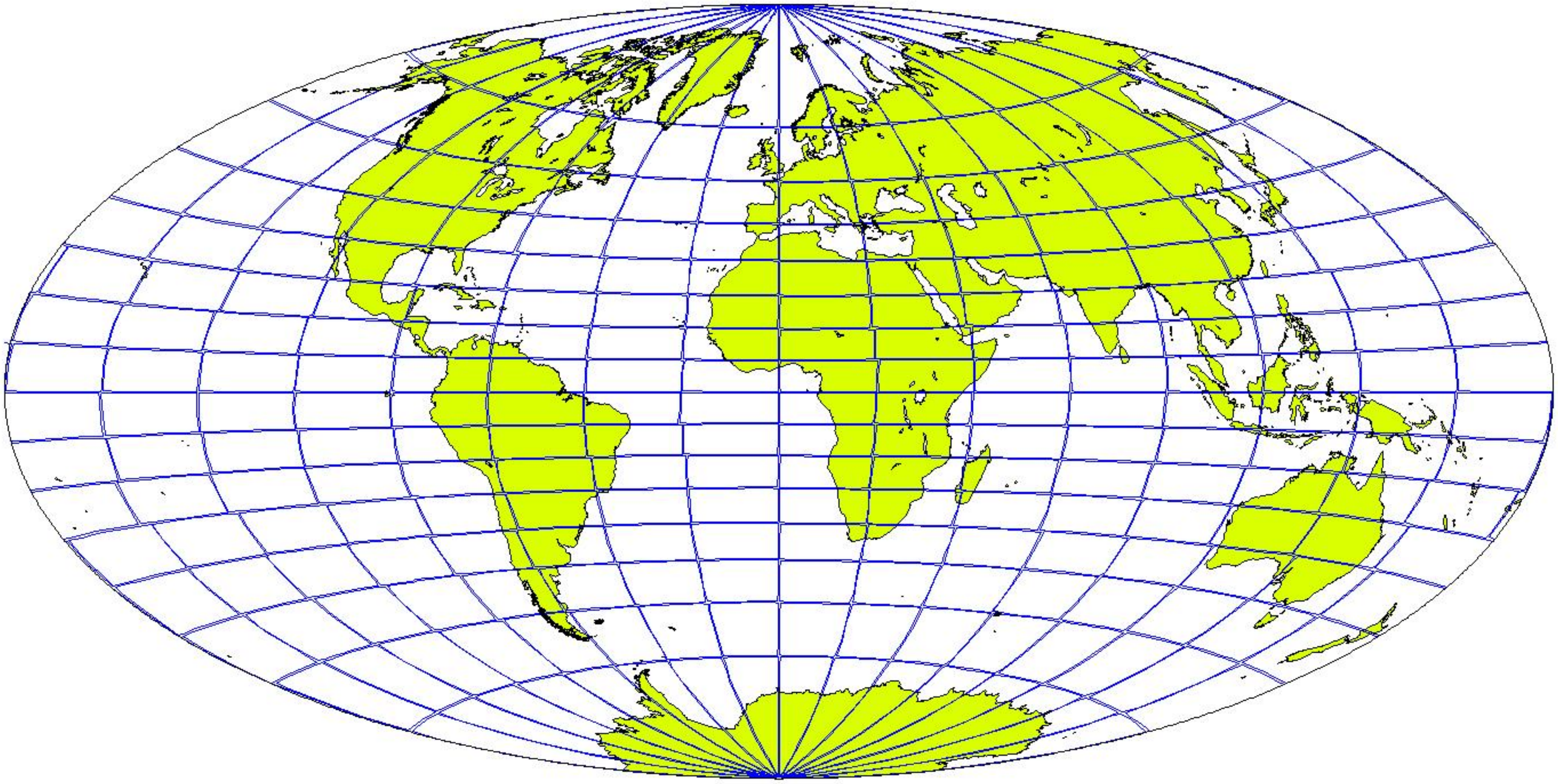


eq_regions partitioning T159 32 tasks

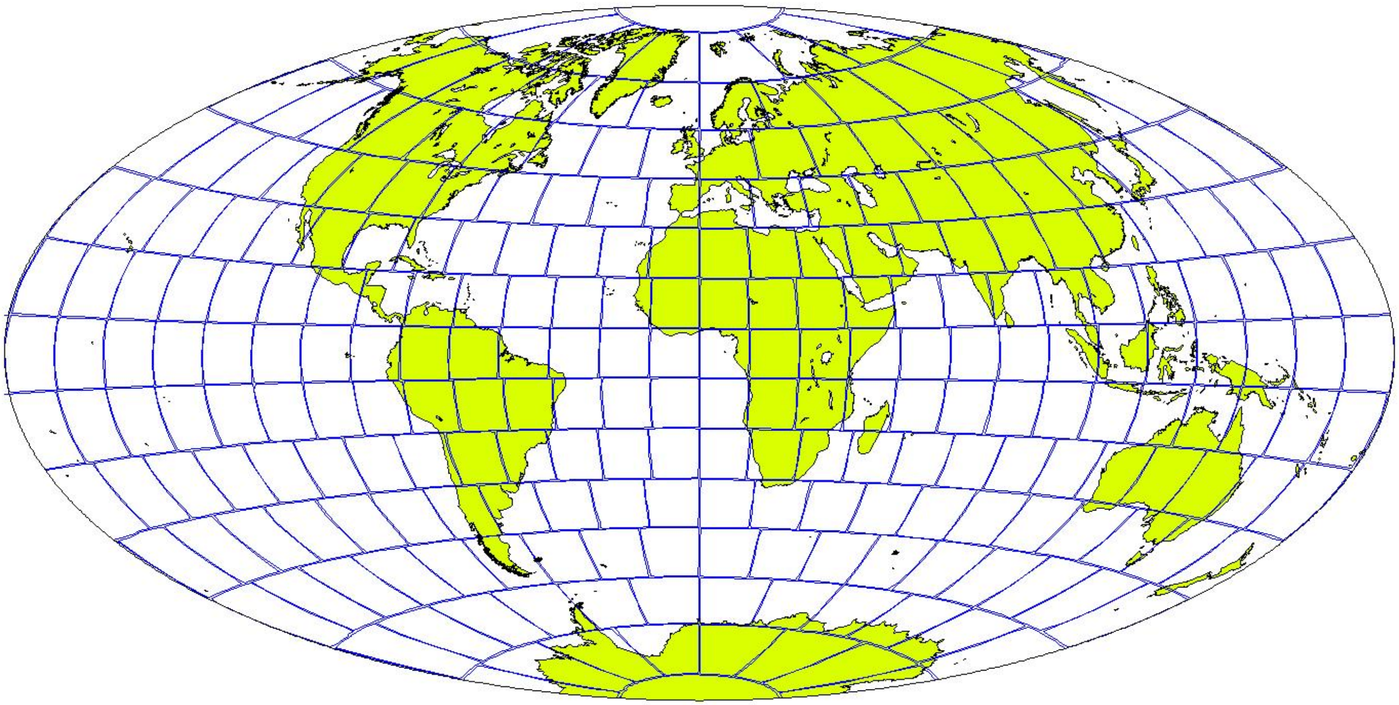


ECMWF

2D partitioning T799 256 tasks (NS=16 x EW=16)

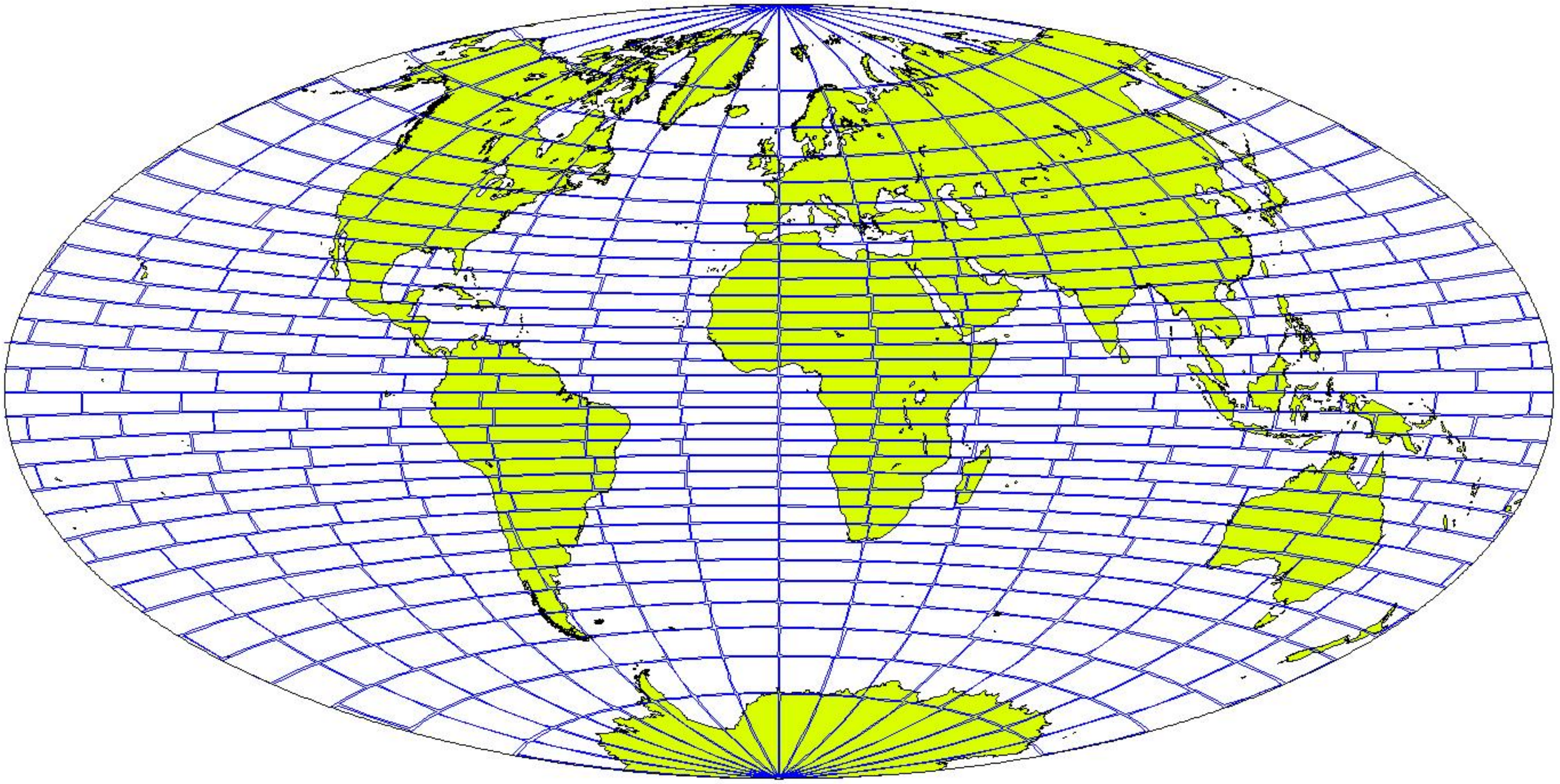


eq_regions partitioning T799 256 tasks

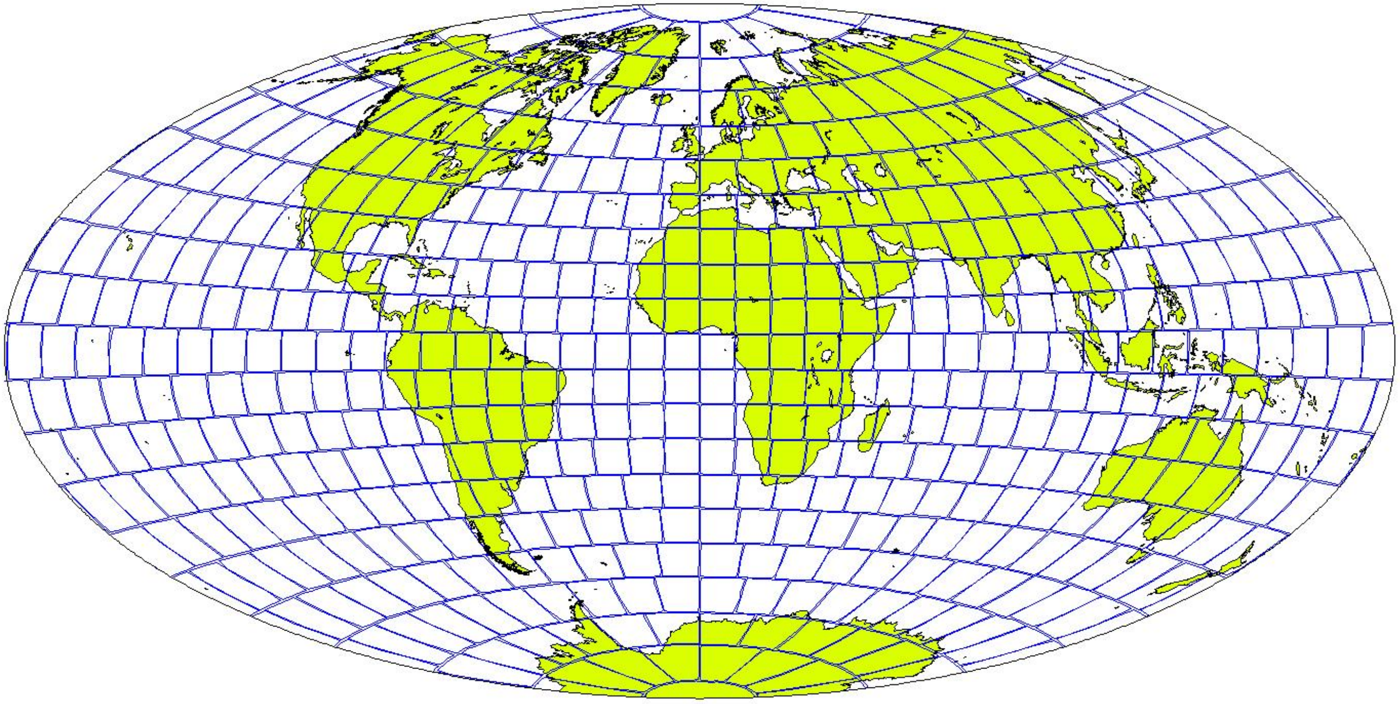


```
N_REGIONS( 1)=   1
N_REGIONS( 2)=   7
N_REGIONS( 3)=  12
N_REGIONS( 4)=  18
N_REGIONS( 5)=  22
N_REGIONS( 6)=  26
N_REGIONS( 7)=  28
N_REGIONS( 8)=  28
N_REGIONS( 9)=  28
N_REGIONS(10)=  26
N_REGIONS(11)=  22
N_REGIONS(12)=  18
N_REGIONS(13)=  12
N_REGIONS(14)=   7
N_REGIONS(15)=   1
```

2D partitioning T799 512 tasks (NS=32 x EW=16)

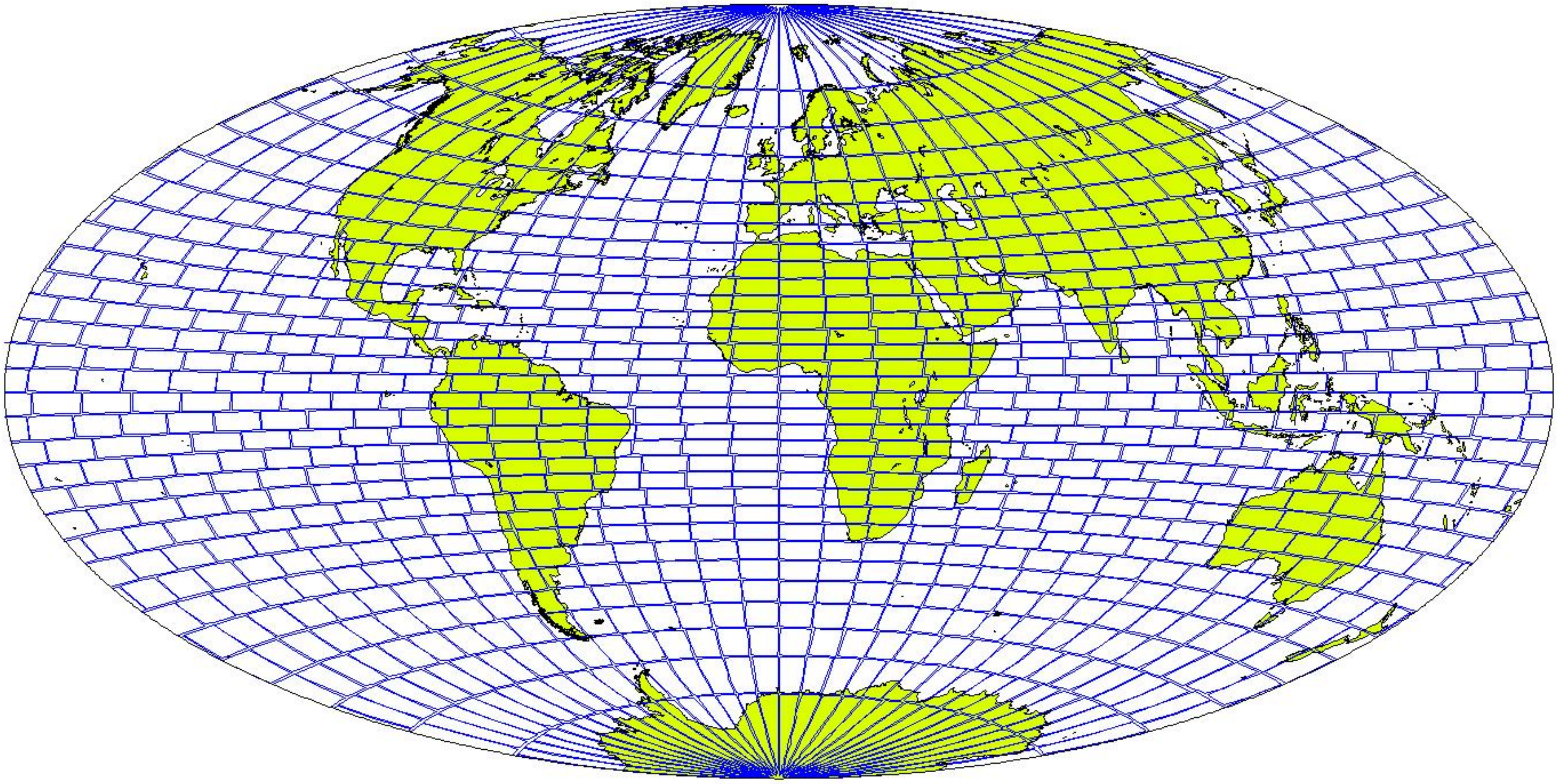


eq_regions partitioning T799 512 tasks

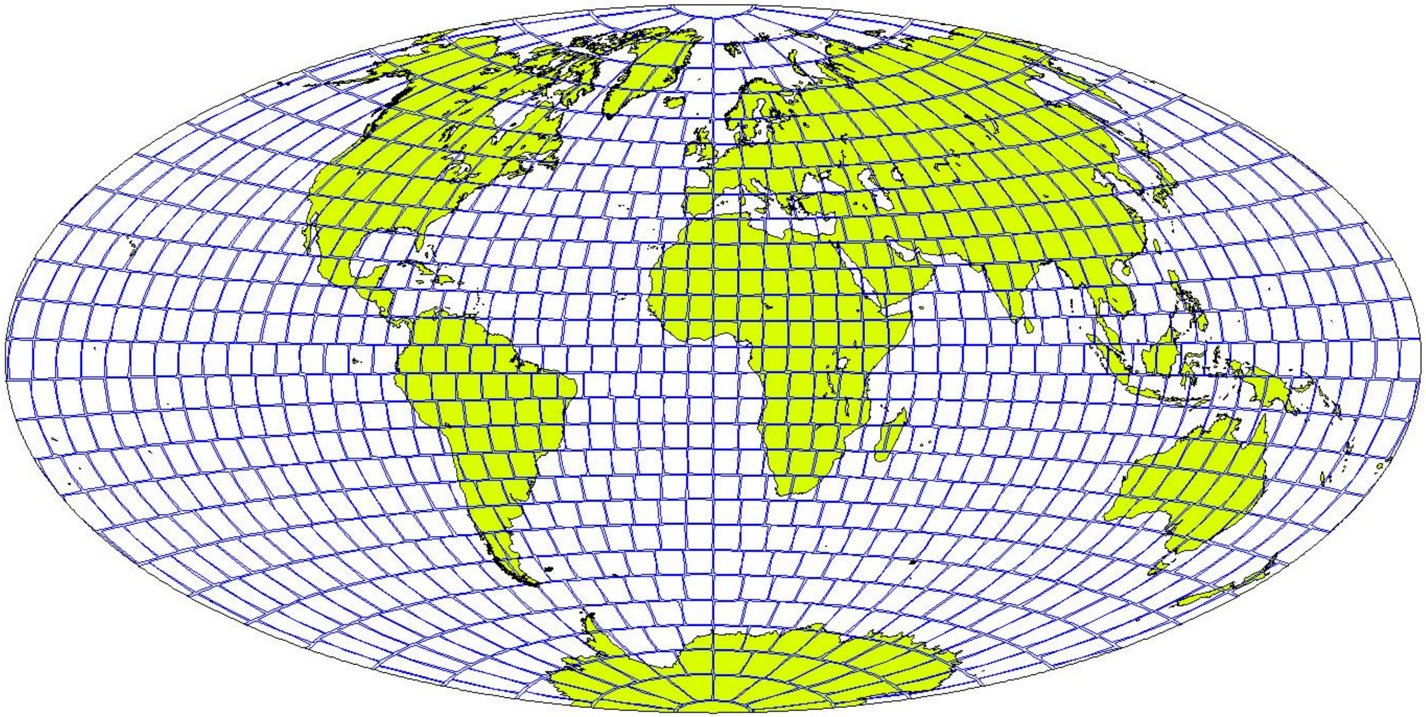


```
N_REGIONS( 1)=  1
N_REGIONS( 2)=  7
N_REGIONS( 3)= 12
N_REGIONS( 4)= 19
N_REGIONS( 5)= 23
N_REGIONS( 6)= 29
N_REGIONS( 7)= 32
N_REGIONS( 8)= 36
N_REGIONS( 9)= 38
N_REGIONS(10)= 39
N_REGIONS(11)= 40
N_REGIONS(12)= 39
N_REGIONS(13)= 38
N_REGIONS(14)= 36
N_REGIONS(15)= 32
N_REGIONS(16)= 29
N_REGIONS(17)= 23
N_REGIONS(18)= 19
N_REGIONS(19)= 12
N_REGIONS(20)=  7
N_REGIONS(21)=  1
```

2D partitioning T799 1024 tasks (NS=32 x EW=32)



eq_regions partitioning T799 1024 tasks

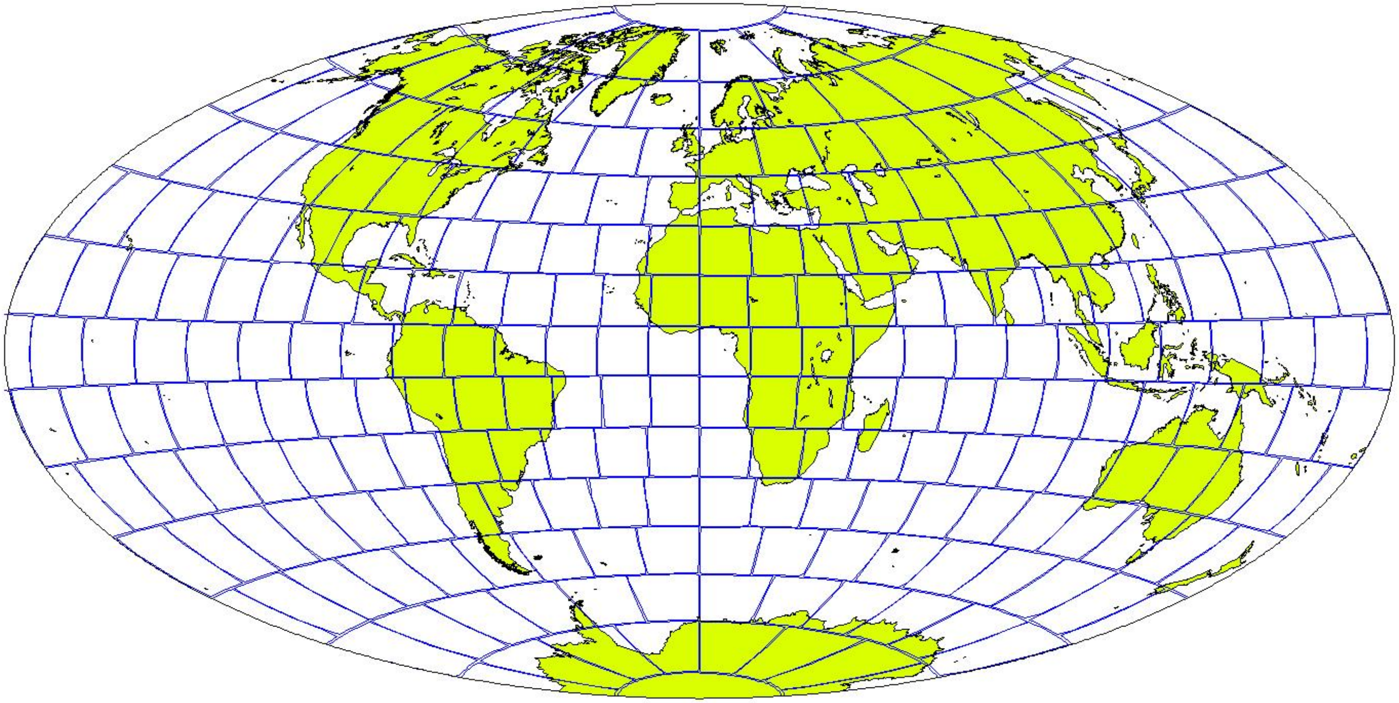


```
N_REGIONS( 1)=  1
N_REGIONS( 2)=  7
N_REGIONS( 3)= 13
N_REGIONS( 4)= 19
N_REGIONS( 5)= 25
N_REGIONS( 6)= 31
N_REGIONS( 7)= 35
N_REGIONS( 8)= 41
N_REGIONS( 9)= 45
N_REGIONS(10)= 48
N_REGIONS(11)= 52
N_REGIONS(12)= 54
N_REGIONS(13)= 56
N_REGIONS(14)= 56
N_REGIONS(15)= 58
N_REGIONS(16)= 56
N_REGIONS(17)= 56
N_REGIONS(18)= 54
N_REGIONS(19)= 52
N_REGIONS(20)= 48
N_REGIONS(21)= 45
N_REGIONS(22)= 41
N_REGIONS(23)= 35
N_REGIONS(24)= 31
N_REGIONS(25)= 25
N_REGIONS(26)= 19
N_REGIONS(27)= 13
N_REGIONS(28)=  7
N_REGIONS(29)=  1
```

2D partitioning T799 251 tasks (NS=251 x EW=1, 251 is a prime ☺)

ECMWF

eq_regions partitioning T799 251 tasks

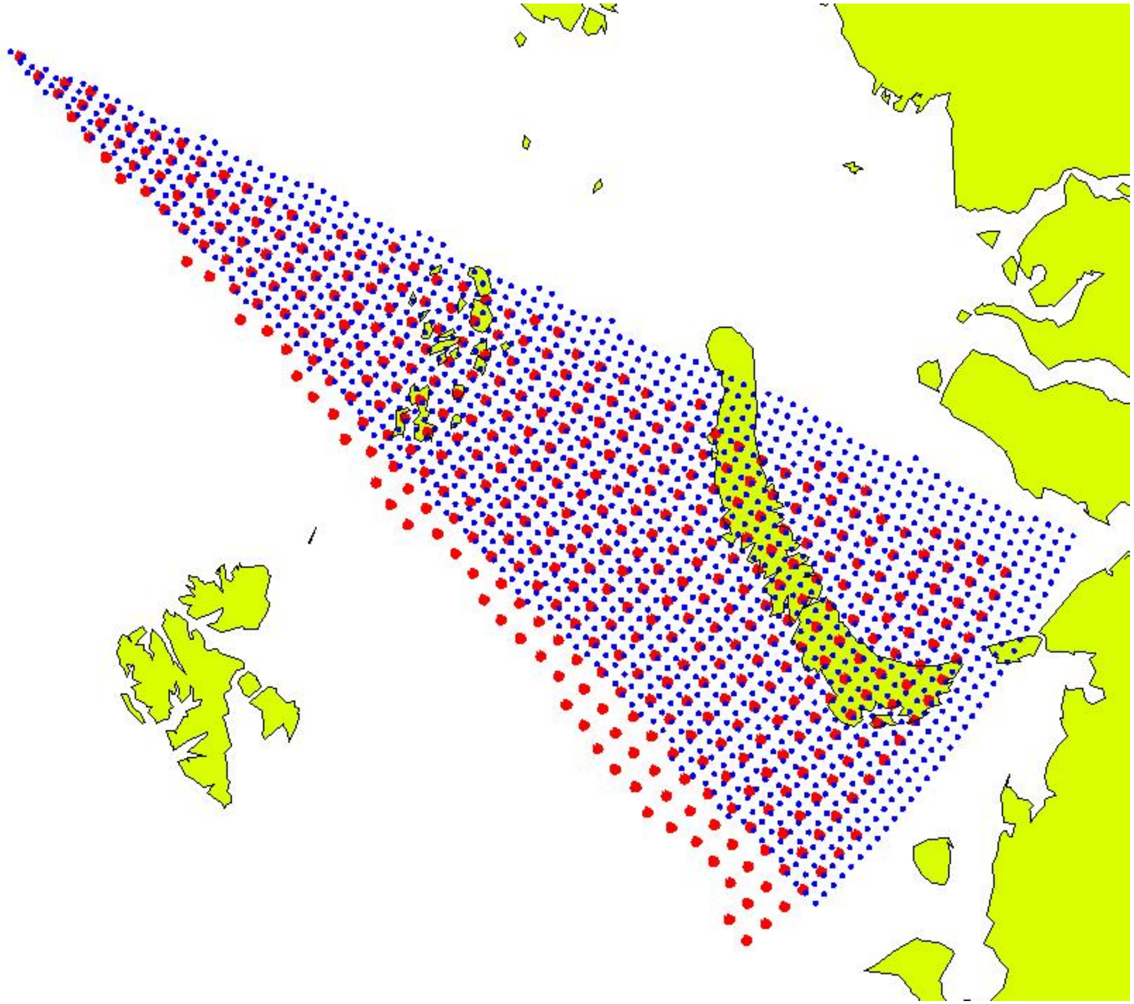


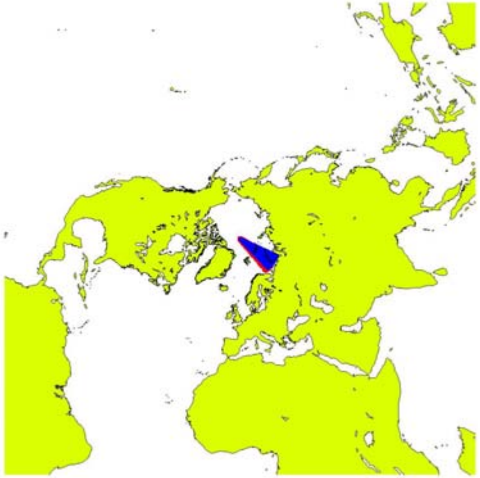
```
N_REGIONS( 1)=  1
N_REGIONS( 2)=  7
N_REGIONS( 3)= 12
N_REGIONS( 4)= 17
N_REGIONS( 5)= 22
N_REGIONS( 6)= 25
N_REGIONS( 7)= 28
N_REGIONS( 8)= 27
N_REGIONS( 9)= 28
N_REGIONS(10)= 25
N_REGIONS(11)= 22
N_REGIONS(12)= 17
N_REGIONS(13)= 12
N_REGIONS(14)=  7
N_REGIONS(15)=  1
```

T799 512 tasks, 2D, task11

T799 model grid

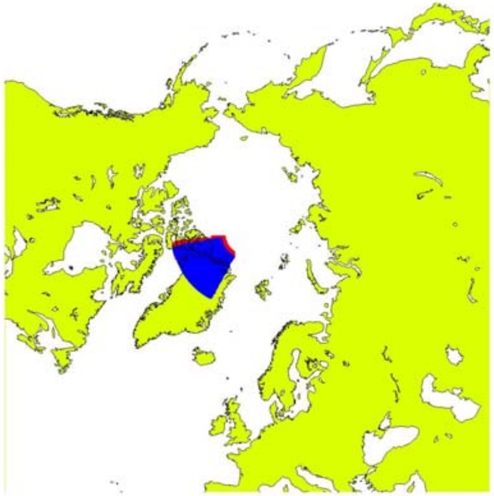
T399 radiation grid

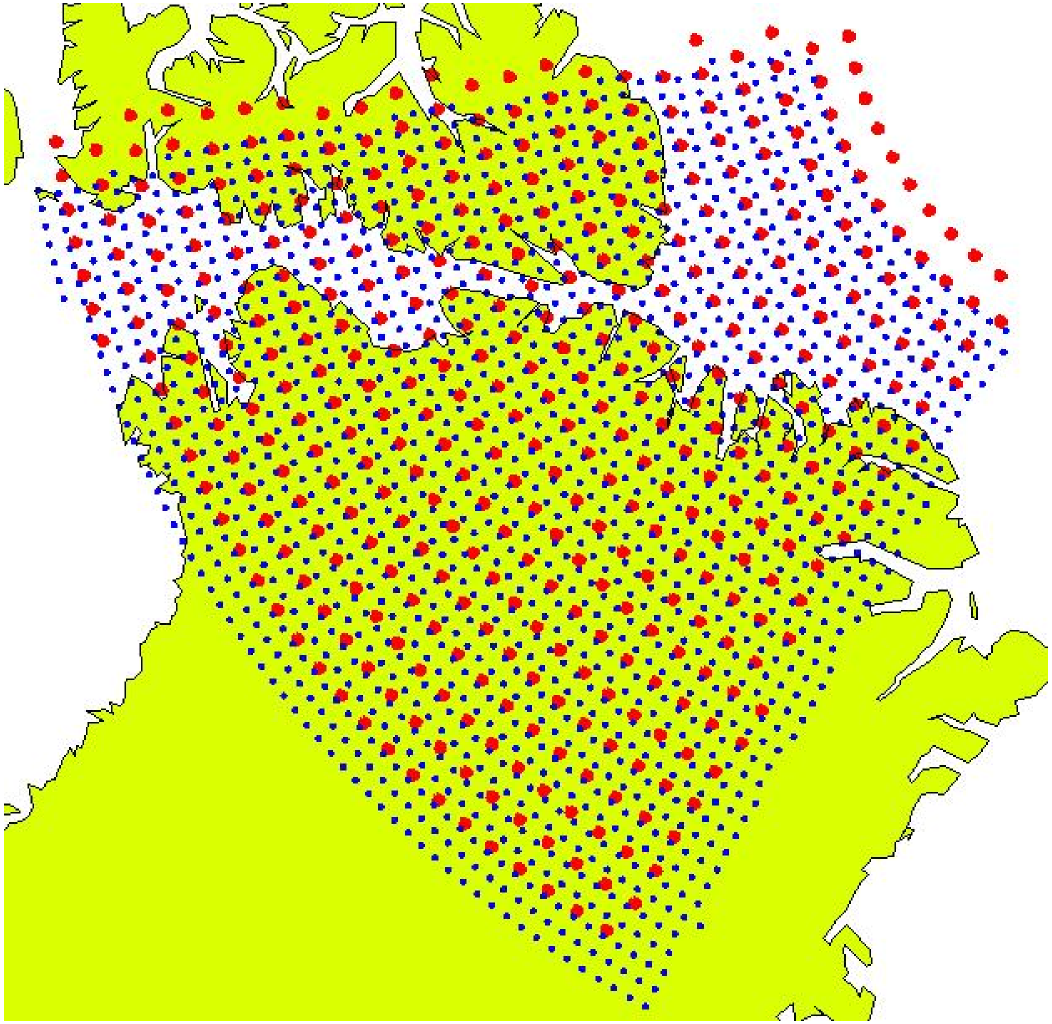




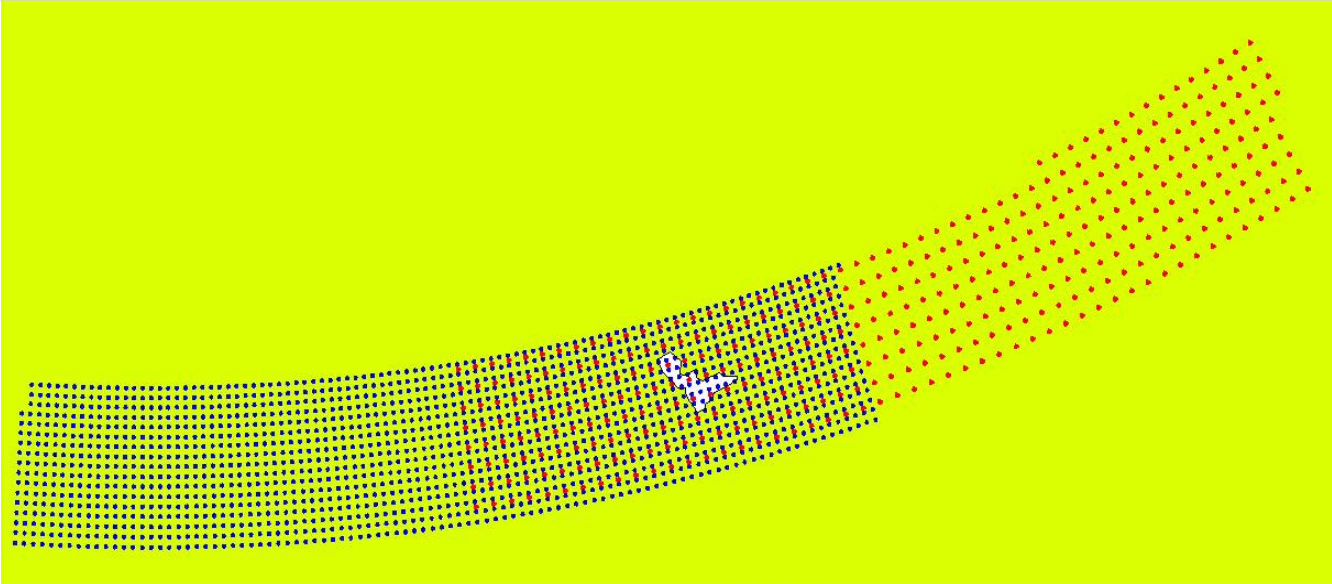
T799 512 tasks, eq_regions, task 4

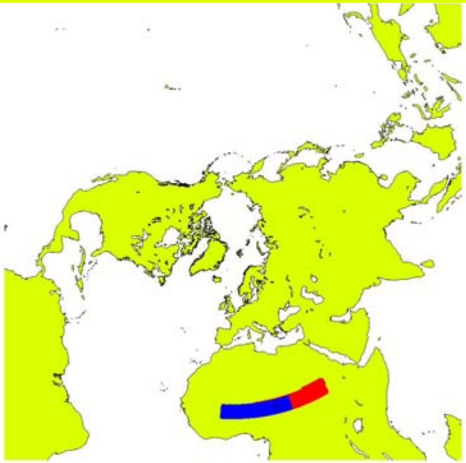
T799 model grid
T399 radiation grid





T799 512 tasks, 2D, task 201

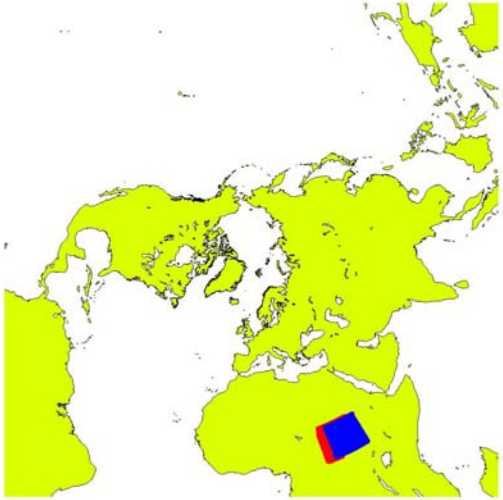


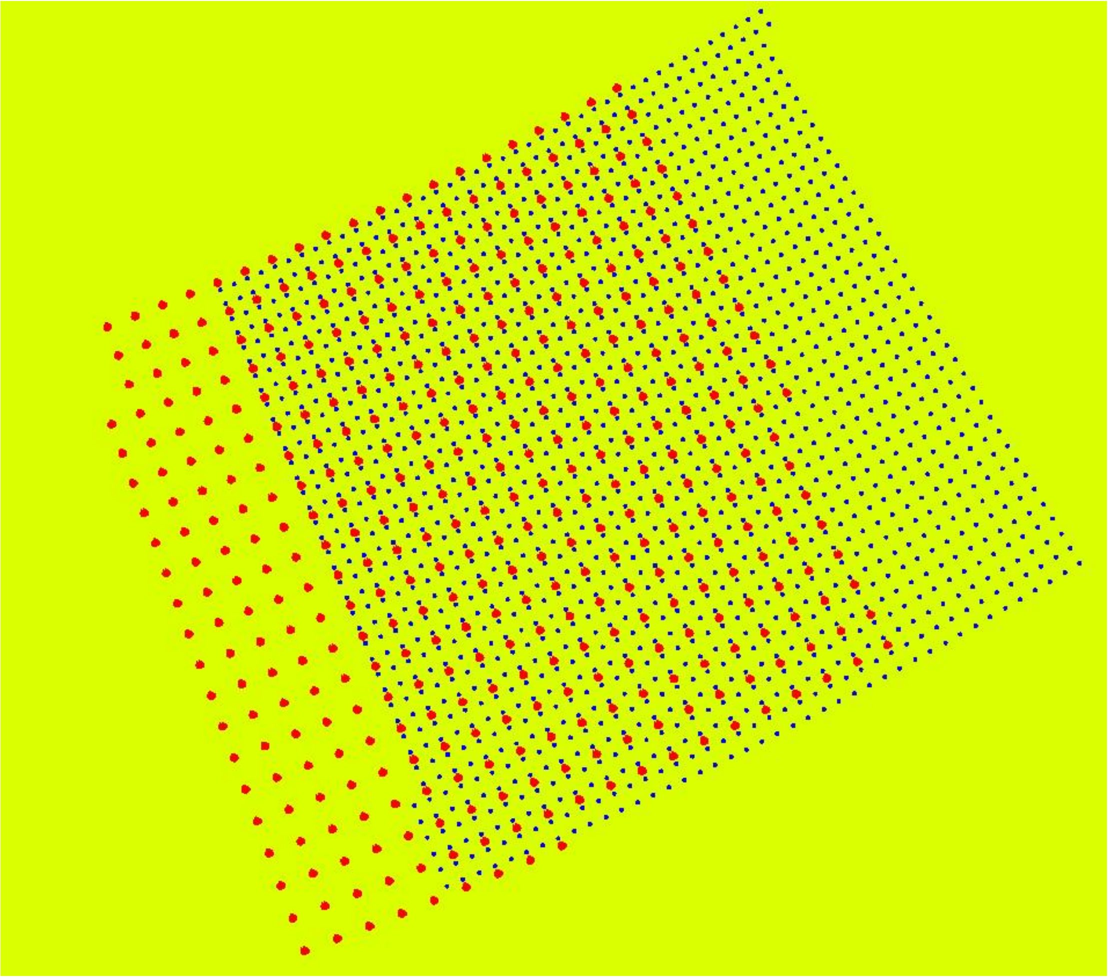


T799 model grid
T399 radiation grid

T799 512 tasks, eq_regions, task 220

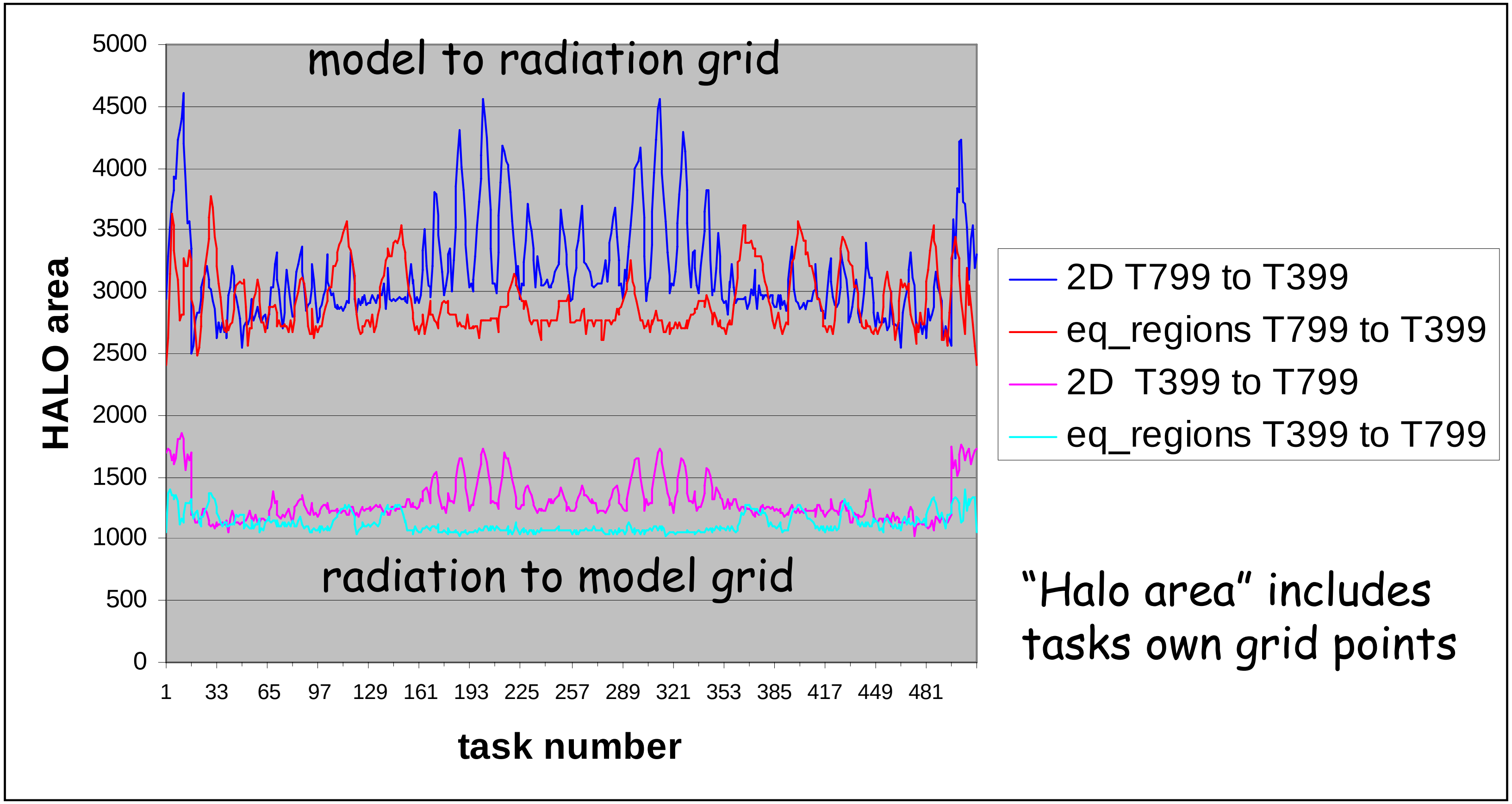
T799 model grid
T399 radiation grid





ECMWF

Grid interpolation HALO area
(512 tasks, T799=model grid, T399=radiation grid)



eq_regions in 4D-Var

- **JB wavelet code in 4D-Var minimisation steps**
- **Used full grids (lat x lon) for the wavelet scales**
  - E.g. min1 scales are T255 T213 T159 T127 T95 T63 T42 T30 T21 T15
- **The problem**
  - T15 has 16 lat x 32 lon points = 512 grid points
  - IFS partitioning restriction, max one split per latitude
  - Full grids not compatible with eq_regions partitioning for smallest scales on 100's of tasks

eq_regions in 4D-Var

- **Solution was to use reduced grids instead of full grids for wavelet scales**
  - T799 4D-Var 192 tasks x 4 threads
  - 1.8% performance improvement overall
  - 7% reduction in memory for min1
  - In the future all wavelet scales will be reset to be linear grids to give a further small improvement in performance
- **Above performance improvement not included when comparing 2D and eq_regions partitioning (next slide)**
- **Thanks to**
  - Mike Fisher (JB wavelet)
  - Mariano Hortal (linear grids)

T799 performance (comparing 2D & eq_regions)

Application	tasks x threads	2D partitioning secs	eq_regions partitioning secs	2D / eq_regions
model	512 x 2	3648	3512	1.039
4D-Var	96 x 8	3563	3468	1.027

Good: Reduced semi-lagrangian comms

Reduced memory requirements

Bad: Increased TRGTOL/TRLTOG comms (grid to fourier space)

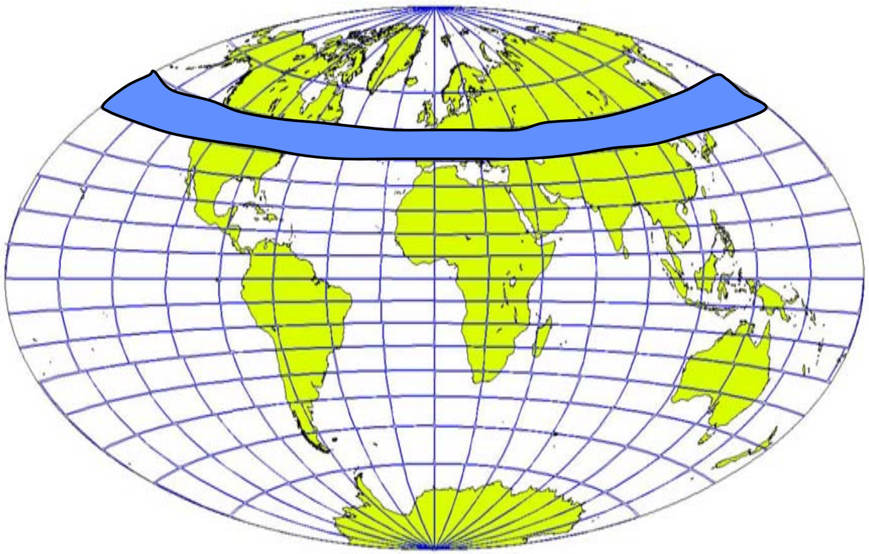
Less of an issue for 'thin' nodes as relatively more comms is 'on switch'

Grid to/from Fourier space transposition (full lats, some levels) 256 tasks x 2 threads, node 3 marked,16 CPU nodes

2D partitioning

Grid/Fourier transposition is mostly performed within each node.

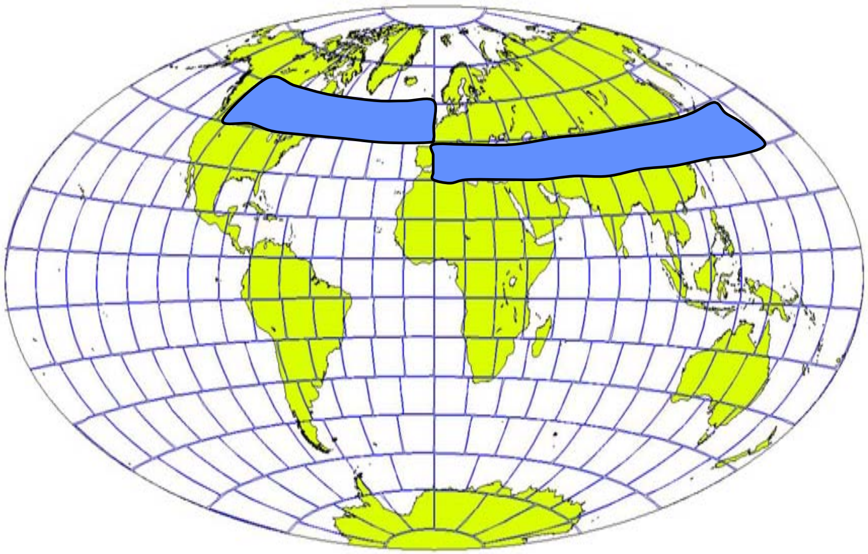
Very little comms required.



eq_regions partitioning

grid / fourier transposition is performed within each node.

Approx half data is off-node.



ECMWF

summary

- **eq_regions partitioning implemented in IFS**
  - Both 2D and eq_regions partitioning are supported
  - eq_regions is the default partitioning
  - Available in IFS cycle CY31R2
- **eq_regions reduces semi-lagrangian communication cost**
  - Also for model / radiation grid interpolation
- **eq_regions has small performance advantage over 2D partitioning**


QUESTIONS?