Towards a Scalable Performance-Portable Software Infrastructure for the Gungho Dynamical Core

Rupert Ford
[Graham Riley, Stephen Pickles, David Ham]

Science & Technology Facilities Council

ECMWF Workshop 2012

NGWCP

- Next Generation Weather & Climate Prediction Programme
- http://www.nerc.ac.uk/research/programmes/ngwcp
- Met Office, NERC, STFC
  - Goal A : Resolution of small scale weather systems in the atmosphere and ocean
  - Goal B : Use of observations to initialise climate predictions

GungHo Project

- New Met Office Dynamical Core
- Timeframe : 2018-2020
- Why : scalability, re-write UM, weaknesses in New Dynamics
- Non latitude-longitude grid
- Investigate both implicit and explicit solvers
- Investigate advection schemes

GungHo Effort

- 5 FTE's from Met Office (Dynamics research and HPC optimisation)
- 5 FTE's from NERC (Bath, Exeter, Imperial, Leeds, Manchester, Reading)
- 2 FTE's from STFC

Why GungHo?

- Andrew Staniforth : “GungHo grids”
- Globally Uniform Next Generation Highly Optimised
- “Working together harmoniously”

工合

Project Structure

- 5 year project
  - Phase 1 : Feb 2011 - Jan 2013
  - Phase 2 : Feb 2013 - Jan 2016
- First Phase 5 themes …
  - Quasi-Uniform grids
  - Advection Schemes
  - Time Schemes
  - Test cases
  - Computational Science Aspects

Project Requirements

- NWP and Climate : 100m local, 10km global, to 150km climate
- Single dynamical core on a single grid (simple switches)
- Scalable code
- Conservation of tracers
- Comparable accuracy to current solution
- Regional modelling supported
- Dynamic adaptability not required (but ...)
- Whole atmosphere modelling : 600km height, 400km climate
- Reproducibility for different processor configurations not required

End of Phase 1 Goals

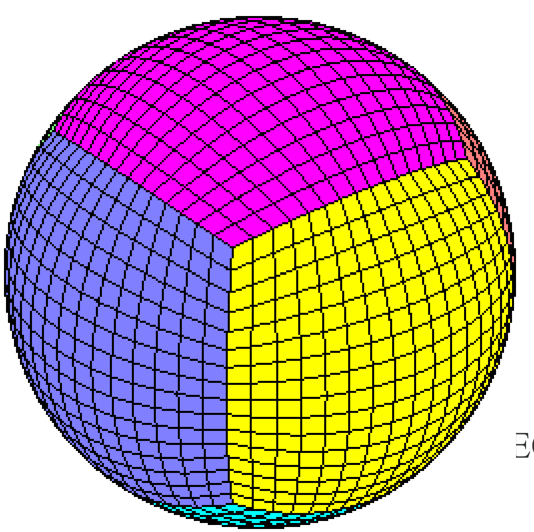
- Single model formulation chosen is prefererable
- Possibly keep more than one for full implementation if more than one option. Must be same in a switchable framework
- Perhaps different grids, or explicit vs. implicit but not different discretisations e.g. TriSK vs. fe-based

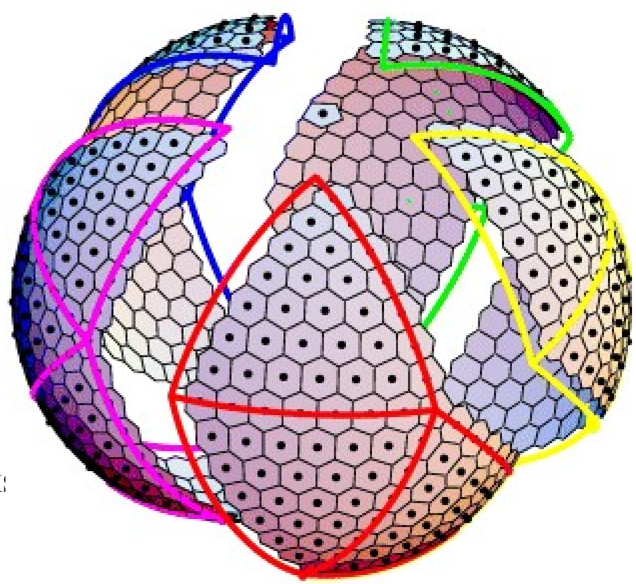
Infrastructure

- Data Structures
- Multiple Grids
- Existing tools
- Support for threading (cores, gpu's, hybrid)
- Futureproof for different discretisations?

Quasi-uniform Grids

- Cubed sphere, icosohedral-hexagonal, triangular, …
- http://kiwi.atmos.colostate.edu/BUGS/geodesic/text.html
- Regular Grid-specific data structures, or general irregular?
- MacDonald et al., A general method for modeling on irregular grids International Journal of High Performance Computing Applications November 2011 25: 392-403
- Regular in the vertical → aleviates cost of indirection





Same Code, Multiple Grids

- General irregular data structures
- Capture topology
- elements, nodes, edges, faces
- Support multiple grids via configuration
- Write code to support different grids – isolate as a “weights” issue when mapping from nodes to elements to edges
- Pre-compute weights (as fixed grid)
- John Thuburn prototype code

Existing tools

- Don't re-invent the wheel
- Partitioned grids and halo definitions
  - Metis, Scotch, ...
- Support for halos and repartitioning
  - Provision in ESMF and MCT for irregular grids
  - ESMF have plans to support determining halos for irregular grids
- Regridding
  - ESMF some support for regridding with irregular grids (triangles and quadrilaterals)
- ESMF
  - Logging
  - calendar and time support

ESMF halo support

distgrid = ESMF_DistGridCreate (arbSeqIndexList = elementIDs, rc=rc)

array = ESMF_ArrayCreate(distgrid, tempPtr, haloSeqIndexList=haloSeqIndexList, rc=rc)

call ESMF_ArrayHaloStore (array, routehandle = haloHandle, rc=rc)

call ESMF_ArrayHalo(array, routehandle= haloHandle, rc=rc)

- Support synchronous or asynchronous

Threading

- Layered approach
- Kernel code which knows nothing about threading (or distributed comms)
- Algorithm/control code which calls kernel functions in apropriate order
- Threading and Comms layer inbetween the two

Threading

- Layered approach
- Kernel code which knows nothing about threading (or distributed comms)
- Algorithm/control code which calls kernel functions in apropriate order
- Threading and Comms layer inbetween the two

Kernel Code : from faces to vertices

```
SUBROUTINE operR(f1,f2,igrid,nf,nv,nz)
DO if1 = 1, nface(igrid)
  ne1 = neoff(if1,igrid)
  ! Share out this face's contributions to its surrounding vertices
  DO ix1 = 1, ne1
    iv1 = voff(if1,ix1,igrid)
    DO k = 1, nz
      f2(nz,iv1) = f2(nz,iv1) + f1(nz,if1)*rcoeff(if1,ix1,igrid)
    ENDDO
  ENDDO
ENDDO
```

Algorithm/Control level

```
CALL HodgeI(f,temp1,igrid,nf)
CALL Ddual1(temp1,temp2,igrid,nf,ne)
CALL HodgeH(temp2,temp3,igrid,ne)
temp2 = temp3*nusq(:,igrid)
CALL Dprimal2(temp2,hf,igrid,ne,nf)
hf = hf - f
```

Threading and Comms Level

```
Subroutine operR()
  Call haloUpdateComplete(...)
  ! OpenMP and/or OpenAcc
  Do i=1,nThreads
    Call operR(....)
  End do
  Call haloUpdateStart(...)
End Subroutine operR
```

Support Different Discretisations?

- Support multiple cores
  - Re-use / future proof
  - Include other cores
- Radical approach (Imperial)
  - Topological mapping of variables is configurable
  - Kernel specifies data requirements in a fetch/execute model
  - Kernel specifies computation at a single grid point (could also do a column)
  - Generate threading/MPI etc code
- MPAS/WRF approach
  - Code specifies its data structures in a registry file
  - Registry file used to populate generic infrastructure with core specific data structures
- What execution model do scientists prefer?

Summary

- GungHo Phase1 nearly complete
- Support multiple grids and perhaps implicit and explicit timestepping switches
- ESMF under serious consideration
- Possible layered architecture for threading
- Support one discretisation but should infrastructure allow this to change?