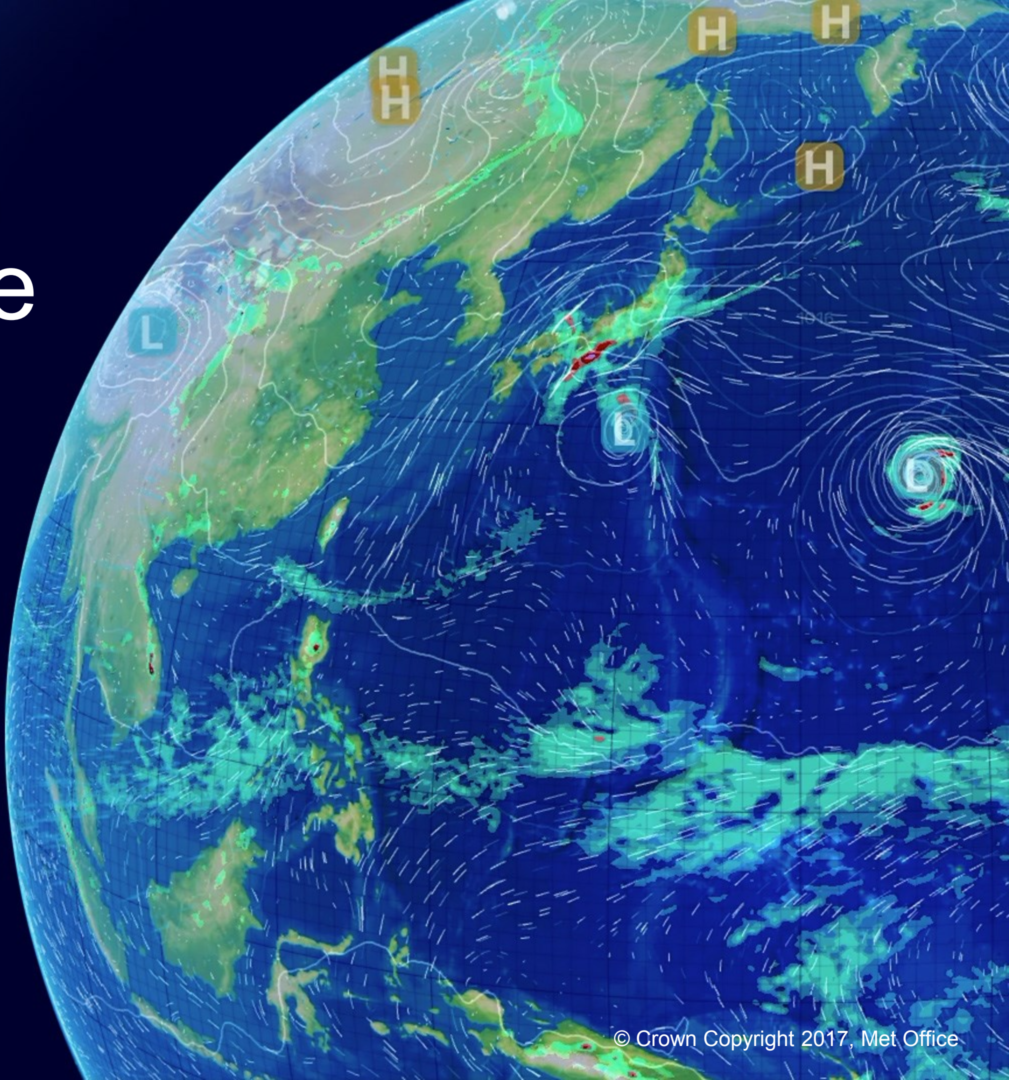
I/O challenges for the LFRic Project

Samantha V. Adams

SIG-IO-UK

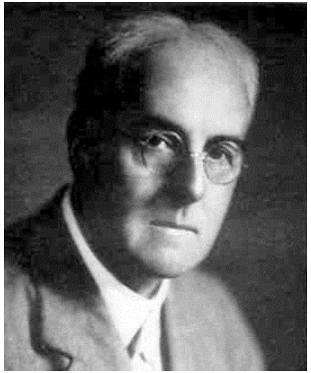
4th June 2018, Reading University, UK.



Talk Overview

- Background and Motivation for the LFRic project
- Why parallel I/O is important & challenges
- Progress and current capability
- Next steps

Met Office



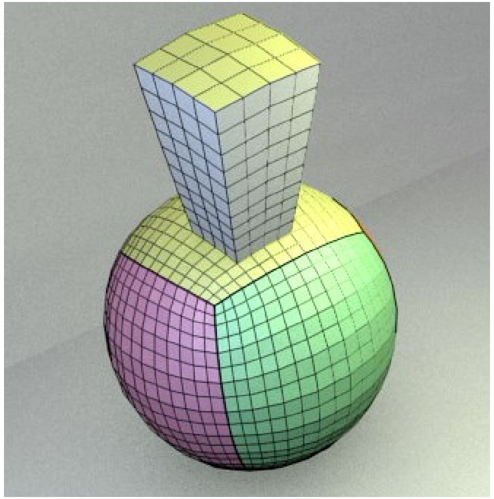
What is LFRic?

- **L**ewis **F**ry **Ric**hardson
- A project to rewrite Met Office modelling infrastructure
- GungHo Project Recommendations (*Met Office, NERC, STFC*)
- Develop science for a new dynamical core
  - Keep the best of current MO dynamical core (EndGame)
  - Improve where possible (e.g. Conservation)
- LFRic infrastructure will address two main (computational science) issues:
  - Scalability looking forward to Exascale
  - Flexible deployment for future HPC architectures

Why is Parallel I/O important now?

- Not in remit of GungHo project or early stages of LFRic
- LFRic science and infrastructure still in development, but we need to:
  - Assess likely impact of I/O on performance
  - Facilitate Science assessment on larger jobs
  - Provide information to other Met Office teams and UM partners about our future output formats
- Prime requirement is to handle parallel read/write efficiently and be scalable (i.e. not destroy the compute performance we are working hard to achieve!)

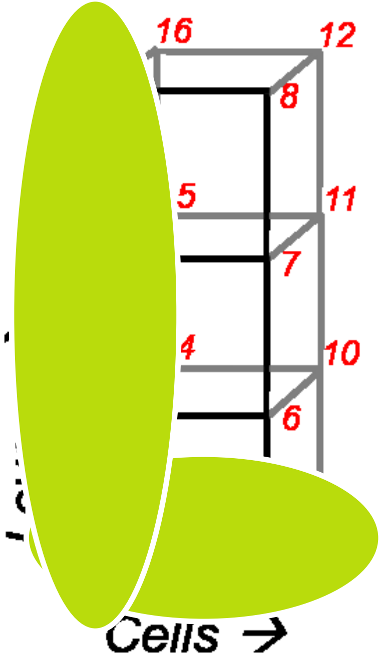
Challenges



- LFRic data layout and file format
  - Semi-structured mesh (cubed sphere)
  - Mixed finite element formulation for the dynamical core
  - UGRID file format
  - These choices complicate things (more later)
- Decided not to write our own parallel I/O framework but instead rely on community solutions and expertise (this is a benefit but also a challenge!)
- Integrate a parallel I/O system with LFRic using a minimal interface (avoid dependencies on specific solutions)

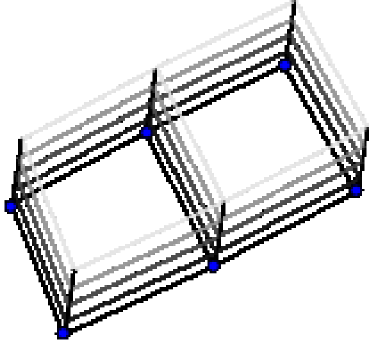
Data Layout

- LFRic infrastructure supports fully unstructured mesh in horizontal and structured in vertical
- Consequently, we have indirect addressing in the horizontal
- To maintain reasonable cache use and sensible vector lengths, the dynamical core uses a ‘k-contiguous’ data layout - Data points (‘dofs’) are column ordered
- Computational ‘work’ currently on whole columns
- Complicates I/O as the data needs restructuring into a ‘layer ordered’ format
- Also (dynamics) field dofs are in computational space for finite element method – needs projection to real world, and special treatment for vector fields



File formats

- UGRID NetCDF
- LFRic base input mesh is in 2D UGRID format
- Output diagnostic format is currently ‘3D layered’ UGRID
- Initial conds, Ancills / LBCs will be UGRID
- Challenges with analysis and visualisation of UGRID



```
dimensions:
nMesh2_node = 6 ; // nNodes
nMesh2_edge = 7 ; // nEdges
nMesh2_face = 2 ; // nFaces
nMaxMesh2_face_nodes = 4 ; // MaxNumNodesPerFace
Mesh2_layers = 10 ;

Two = 2 ;

variables:
// Mesh topology
integer Mesh2 ;
Mesh2:cf_role = "mesh_topology" ;
Mesh2:long_name = "Topology data of 2D unstructured mesh" ;
Mesh2:topology_dimension = 2 ;
Mesh2:node_coordinates = "Mesh2_node_x Mesh2_node_y" ;
Mesh2:face_node_connectivity = "Mesh2_face_nodes" ;
Mesh2:face_dimension = "nMesh2_face" ;
Mesh2:edge_node_connectivity = "Mesh2_edge_nodes" ;
Mesh2:edge_dimension = "nMesh2_edge" ;
Mesh2:edge_coordinates = "Mesh2_edge_x Mesh2_edge_y" ;
Mesh2:face_coordinates = "Mesh2_face_x Mesh2_face_y" ;
Mesh2:face_edge_connectivity = "Mesh2_face_edges" ;
Mesh2:face_face_connectivity = "Mesh2_face_links" ;
Mesh2:edge_face_connectivity = "Mesh2_edge_face_links"
integer Mesh2_face_nodes(nMesh2_face, nMaxMesh2_face_nodes) ;
Mesh2_face_nodes:cf_role = "face_node_connectivity" ;
Mesh2_face_nodes:long_name = "Maps every face to its corner nodes." ;
Mesh2_face_nodes:_FillValue = 999999 ;
Mesh2_face_nodes:start_index = 1 ;
integer Mesh2_edge_nodes(nMesh2_edge, Two) ;
Mesh2_edge_nodes:cf_role = "edge_node_connectivity" ;
Mesh2_edge_nodes:long_name = "Maps every edge to the two nodes that it connects." ;
Mesh2_edge_nodes:start_index = 1 ;
```

http://ugrid-conventions.github.io/ugrid-conventions/#3d-layered-mesh-topology

XIOS

- For our needs seemed the best of available parallel I/O frameworks
- Supports parallel read and write
- Proven on jobs ~10K cores in weather and climate domain
- Already in use in Met Office (NEMO ocean model)
- Works with OASIS coupler now and coupling functionality is being added
- Prior to 2016, no frameworks supported UGRID output
- We have collaborated with IPSL to add UGRID support for LFRic

LFRic and XIOS Progress

- **Implementation progress**
  - UGRID support added to XIOS late 2016
  - XIOS integrated with LFRic 2016 -2017
- **Current functionality**
  - LFRic writes UGRID diagnostics
  - LFRic reads/writes CF NetCDF checkpoints
- **Work in progress**
  - Proper treatment of diagnostic vector fields in UGRID (on edges / faces)
  - Reading UGRID initial conditions / start dumps
  - UGRID for checkpoint

Performance Results (2017)

Scaling

- Run out to 14k cores with little/no I/O penalty. (but nowhere near operational config)
- Tuning I/O servers. As expected generally more == better. Reduces client wait time and no increase in overall run time…..BUT…

Impacts of a Lustre file system?

- With appropriate striping, can achieve low client wait time with fewer I/O servers

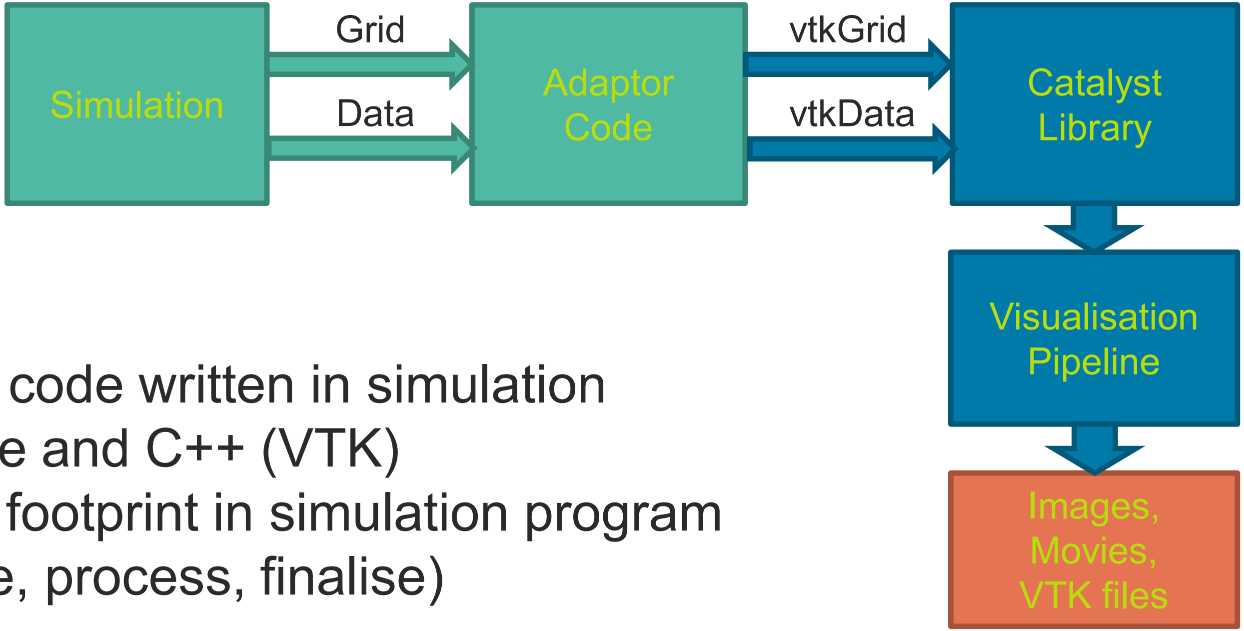
Diagnostic output loading

- With each 100 field (~112Gb) increase, approx +5% I/O penalty

In-situ Analysis and Visualisation

- Maybe we don’t always have to write full data to disk?!
- “FLOPS are free” – better to process data while it is still “hot” (near the processor)
- If scientists want a way to quickly look at results or debug a model run
- Avoids issues with specialist file formats like UGRID
- In collaboration with NIWA we have been prototyping with Paraview/Catalyst – a well-known HPC visualisation solution.

Paraview / Catalyst Workflow



- Adaptor code written in simulation language and C++ (VTK)
- Minimal footprint in simulation program (initialise, process, finalise)

Image courtesy of Wolfgang Hayek

In-situ Analysis and Visualisation

- From the end user point of view
- Python pipeline
  - Create standard scripts for end users to edit
  - End users can create their own from within Paraview
  - Minimal programming required (but need to learn some Paraview)
- Create a specialist 'adaptor' in C++. More programming skills required but have access to full power of Paraview.
- Paraview 'live' – Paraview running simultaneously with live model. Potentially great for debugging

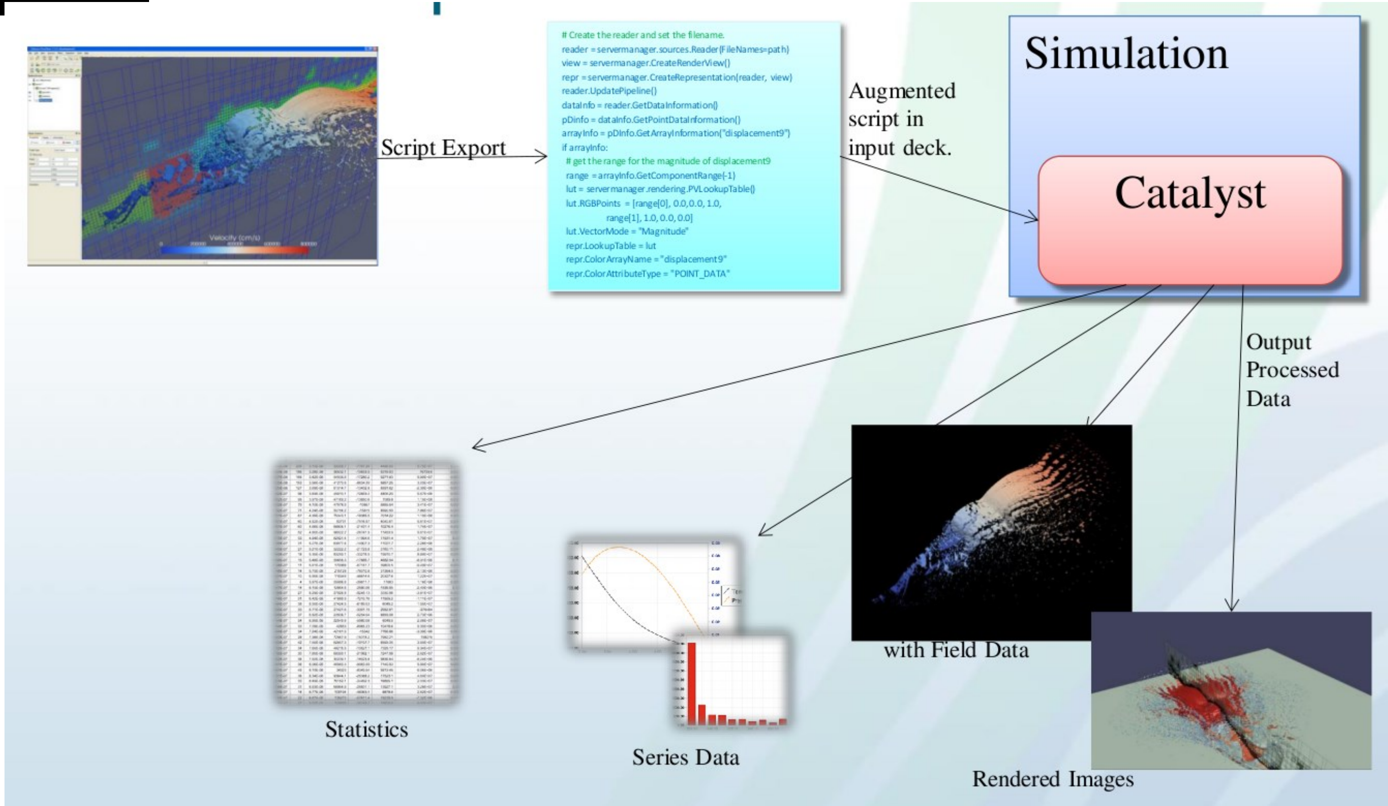
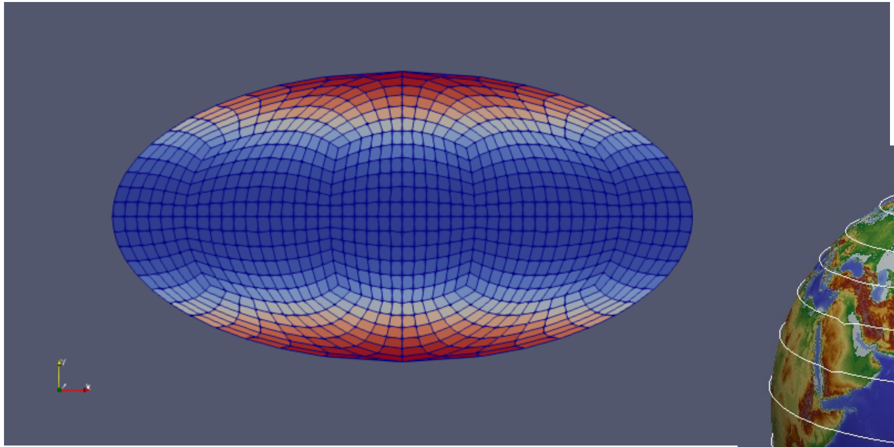


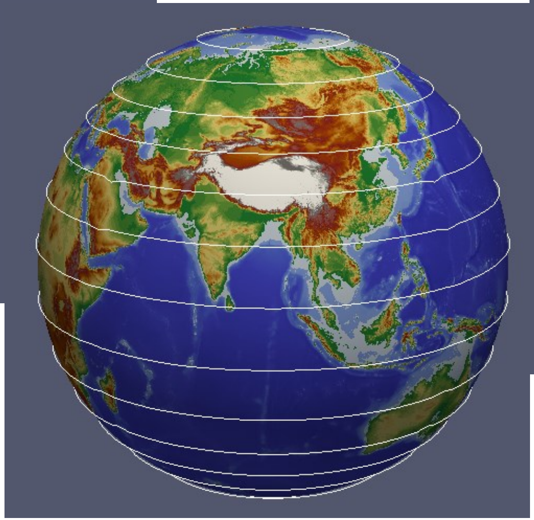
Image courtesy of Kitware Catalyst Tutorial

The visualisations were created using LFRic output and ParaView Catalyst, ParaView, and VTK.

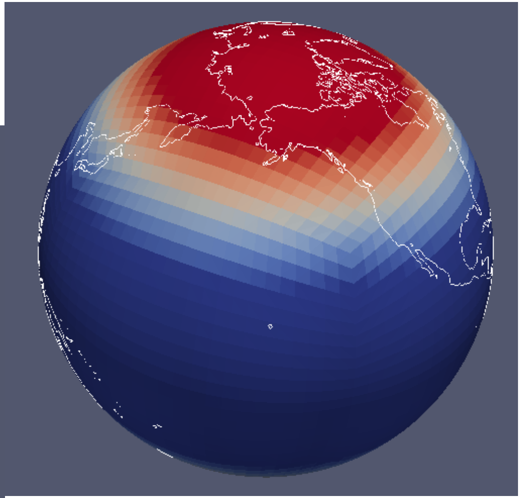
Images courtesy of Wolfgang Hayek, NIWA



Density field output to VTK, then post-processing with VTK-Python for the map projection



Direct render of density field to contours plus topography



Direct render of density field with coastline

Next Steps

- More work needed on both I and O parts of I/O!
- Proper treatment of vector fields. Currently project to a 'finite volume' cell centre equivalent
- Reading start dumps / initial conditions in UGRID format
- LFRic collaborations with UM partner NIWA
  - In situ visualisation (Wolfgang Hayek). Will deliver an LFRic mini-app in 2018/2019
  - Mimetic regridding / post-processing (Alex Pletzer). Will deliver an LFRic mini-app in 2018/2019

Met Office

Acknowledgements

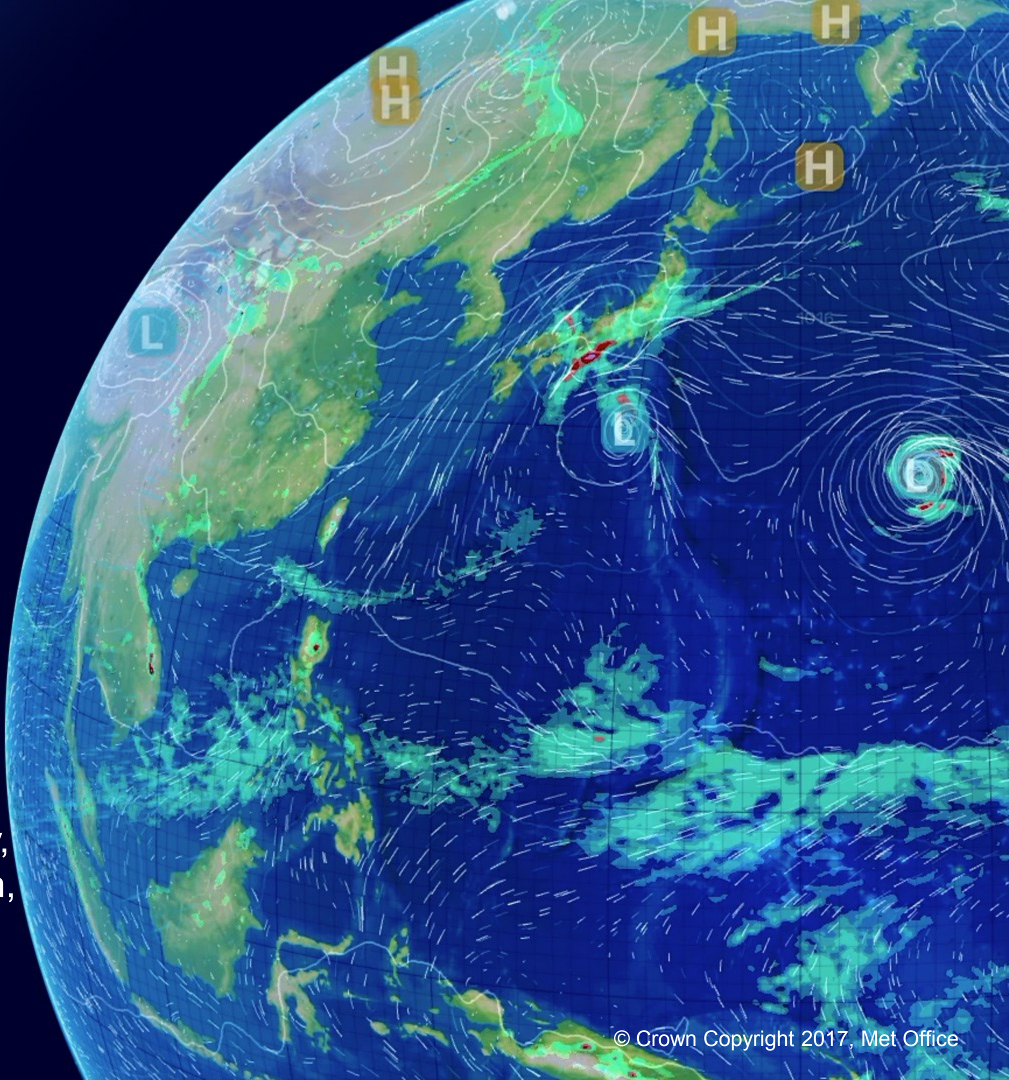
Monash University, Australia: Mike Rezny

IPSL (LSCE/CEA), France:
Olga Abramkina, Yann Meurdesoif

NIWA / NESI, NZ:
Wolfgang Hayek, Alex Pletzer

STFC (Hartree Centre), UK:
Rupert Ford, Andy Porter

Met Office UK LFRic team:
Sam Adams, Tommaso Benacchio, Matthew Hambley, Mike Hobson, Iva Kavcic, Chris Maynard, Tom Melvin, Steve Mullerworth, Stephen Pring, Steve Sandbach, Ben Shipway, Ricky Wong



Met Office

Thank You!

Questions?