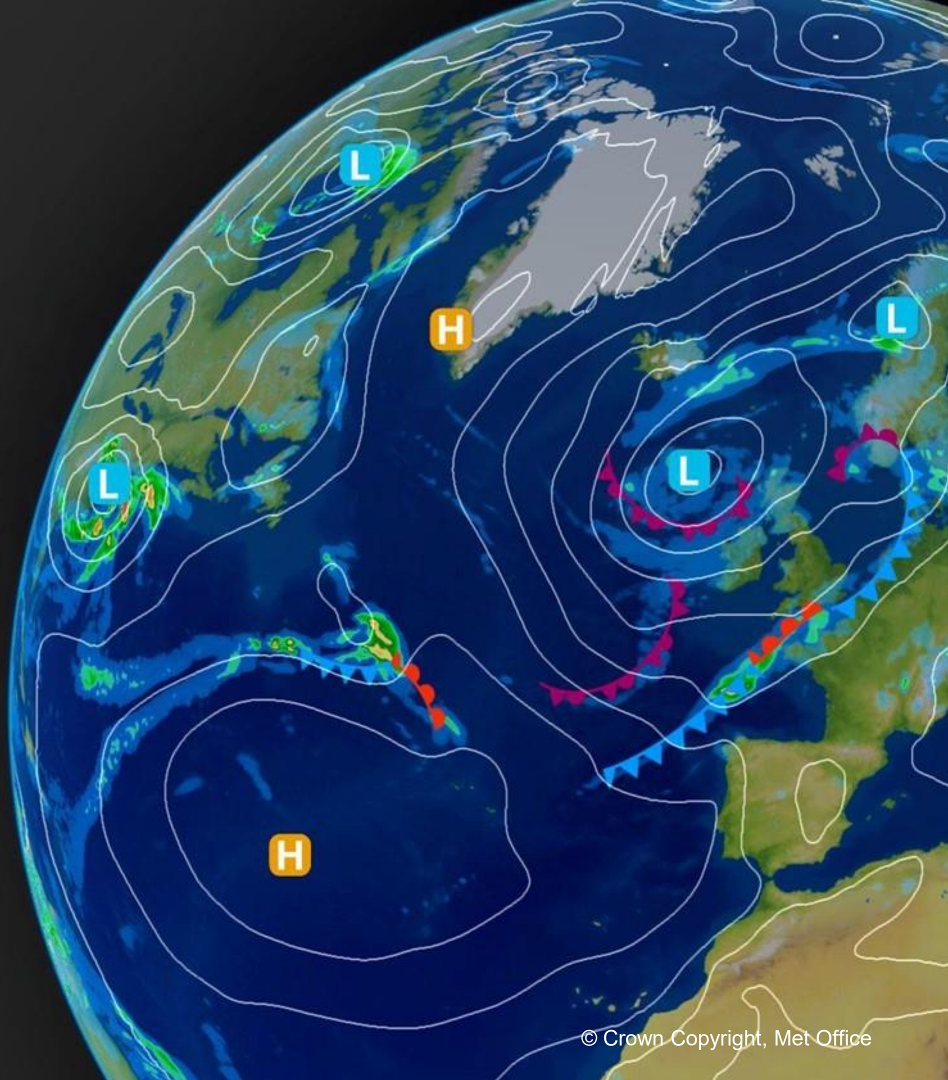
Met Office

Dynamical cores for the Met Office's Unified Modelling system: Past, present and future

Nigel Wood, Dynamics Research

With acknowledgments to many colleagues over many years



Outline of talk

- Past: Some history of the Unified Model
- Present: The challenge for the Unified Model
- Future:
    - GungHo
    - LFRic

The Unified Model: Past & Present

Some history

Year	Equation Set	Levels	ΔX (km)	Notes
<1990	Hydrostatic	15	150	Different NWP & Climate models
1991	Unified Model Deep, Quasi-Hydrostatic	20	90	Unified NWP & Climate global models
2002	Deep, Non-Hydrostatic ***New Dynamics***	38/50/70/85	60/40/25 (4/1.5)	Non-hydrostatic: Unified global & regional models
2014	Deep, Non-Hydrostatic ***ENDGame***	70/85	17 (4/1.5)	Improved stability, scalability, accuracy
2025	Deep, Non-Hydrostatic ***GungHo***	>100	<10 (<1)	Quasi-uniform grid

(ENDGame = Even Newer Dynamics for General atmospheric modelling of the environment)

What is the Unified Model?

- Operational forecasts
  - Global (resolution approx. 10km)
  - Regional (resolution approx. 1.5km)

- Seasonal predictions
  - Resolution approx. 60km

> 25 years old

- Global and regional climate predictions
  - Global resolution around 120km
  - Regional around 4-1.5km
  - Run for 10-100-… years

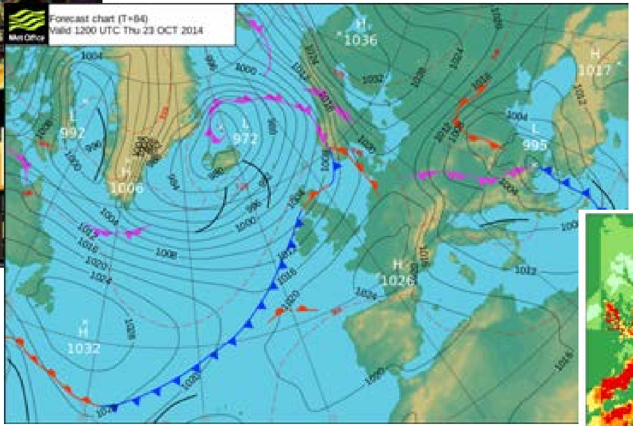
- Research mode
  - Resolution 1km - 10m

Brown et al. (2012) BAMS

The consequence of unification

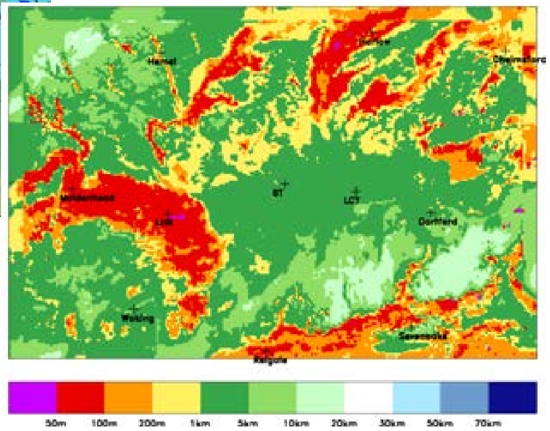
300 km

30 km



A factor of 1000 between these

300 m

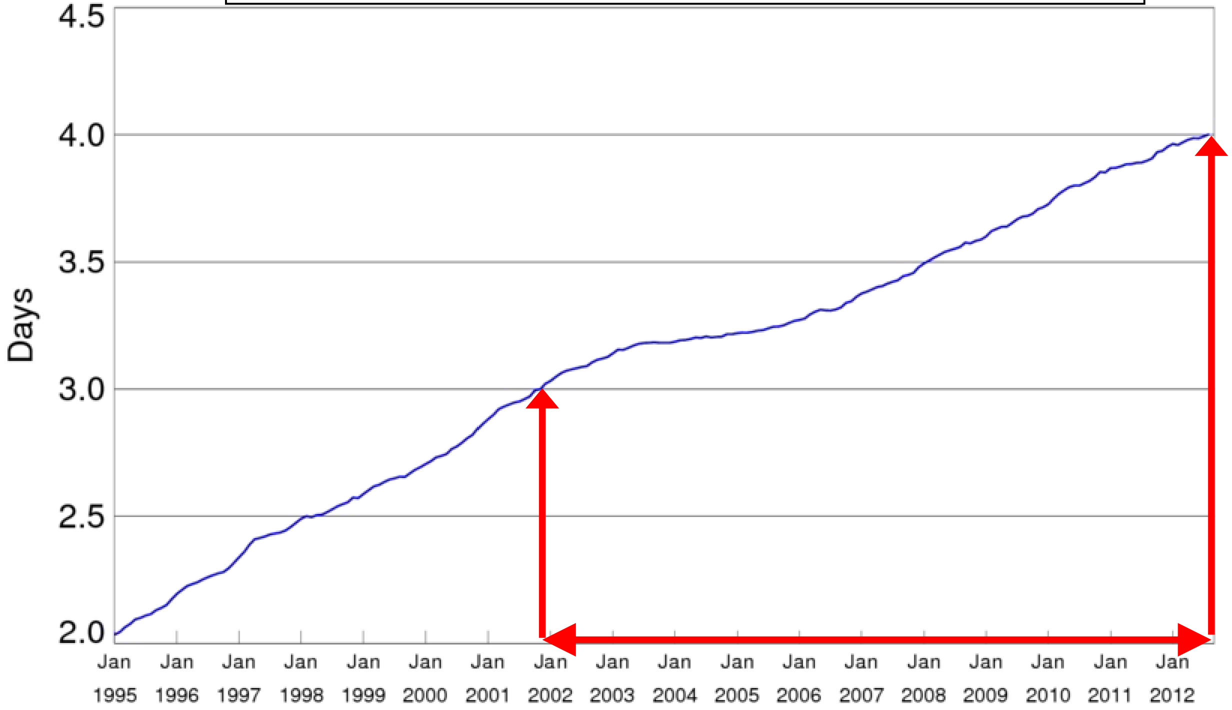


…the same scheme has to continue to work

“The quiet revolution”*: 1 day a decade

“…impact of NWP among greatest of any area of science… comparable to simulation of human brain and evolution of early universe”*

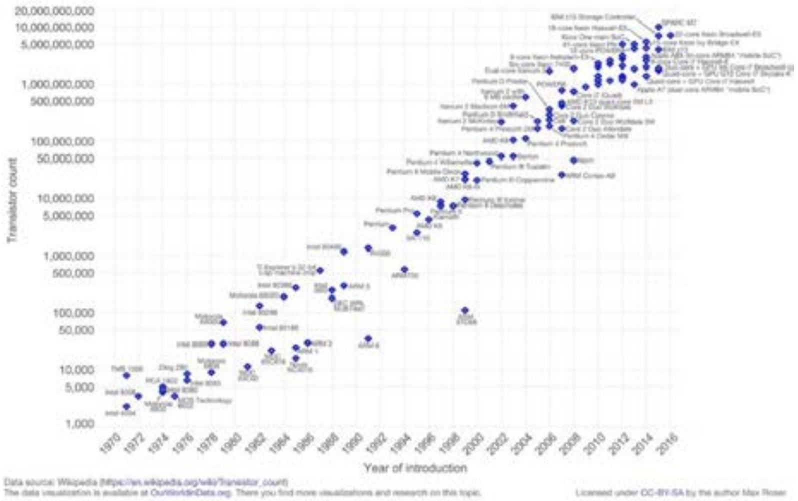
Forecast length where surface pressure RMSE over N. Atlantic and Europe matches one day forecast in 1980



*Bauer, Thorpe, Brunet (2015) Nature

The Challenge for the Unified Model

HPC key but HPC landscape is changing



Moore's law slowing but has struggled on: >10,000,000,000 transistors in 2016

But Dennard Scaling ceased 2006

- On 8 June 2018 U.S. DoE's ORNL announced Summit
- Peak performance of 200 petaflops.
- Size of 2 tennis courts; 4,000 gallons water a minute for cooling; 10 petabytes of memory; 250 petabytes file system

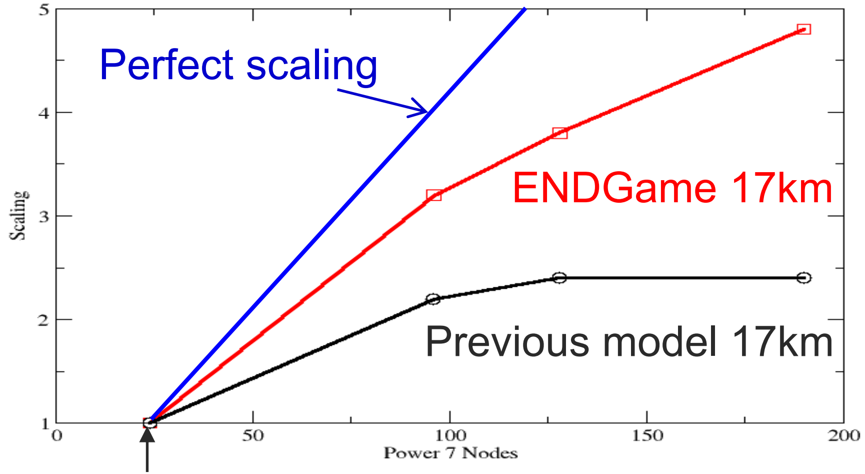
"Crossing the chasm" Lawrence et al (2018) GMD

Met Office

Scalability = critical element

T_{24}/T_N

N768



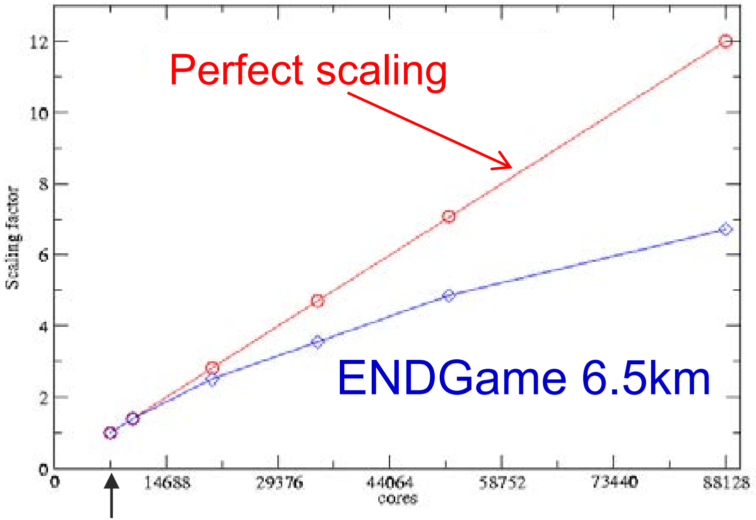
24 nodes

Nodes

(1 node=32 cores)

T_{7344}/T_N

N2048



7344 cores

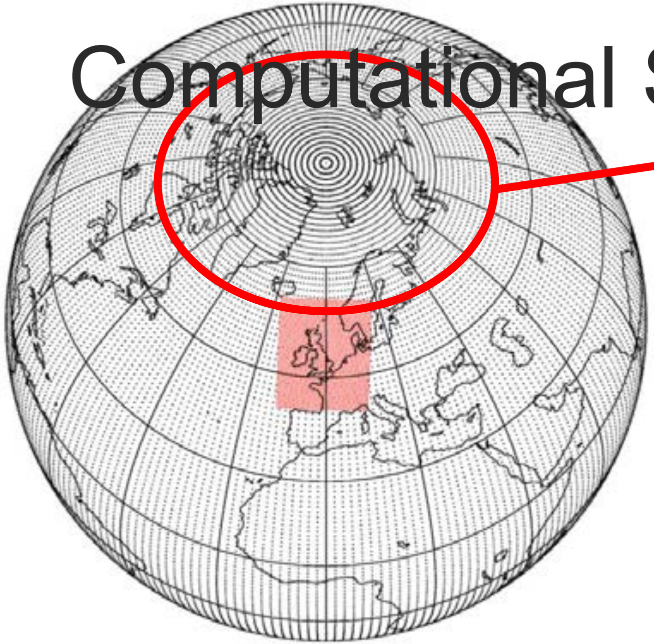
Cores

(1 node=36 cores)

Andy Malcolm, Paul Selwood

What are the barriers?

Computational Science



- At 10 km spacing near poles is 13 m
- At 1 km it reduces to 13 **cm**!
- **Communications** increase dramatically

Natural Science

⇒ Seek uniform mesh

- Project = GungHo

(GungHo = Globally Uniform next generation Highly optimized)

Met Office

What are the barriers?

Computational Science

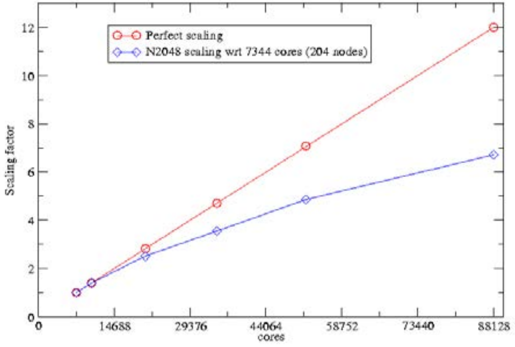
- No one knows just what these beasts are going to look like
- But we do know that the current code will not be efficient
- How do we write the new code?
- Project = LFRic

  (after L.F. Richardson)

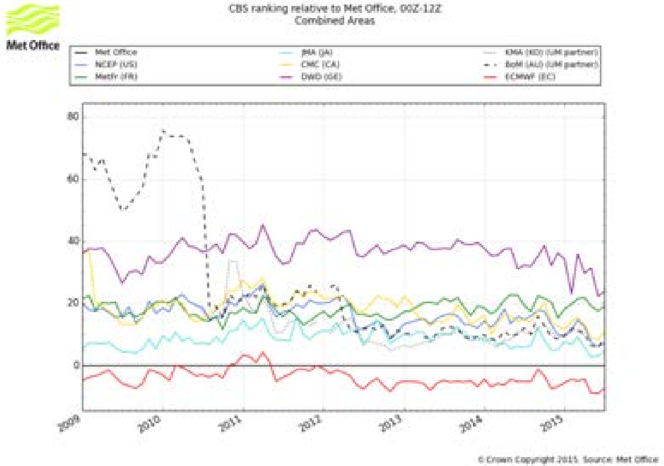
The Future: GungHo

Met Office

The challenge



+



+

"To research, design and develop a new dynamical core suitable for operational, global and regional, weather and climate simulation on massively parallel computers of the size envisaged over the coming 20 years."
[Targeting mid-2020's HPC upgrade]

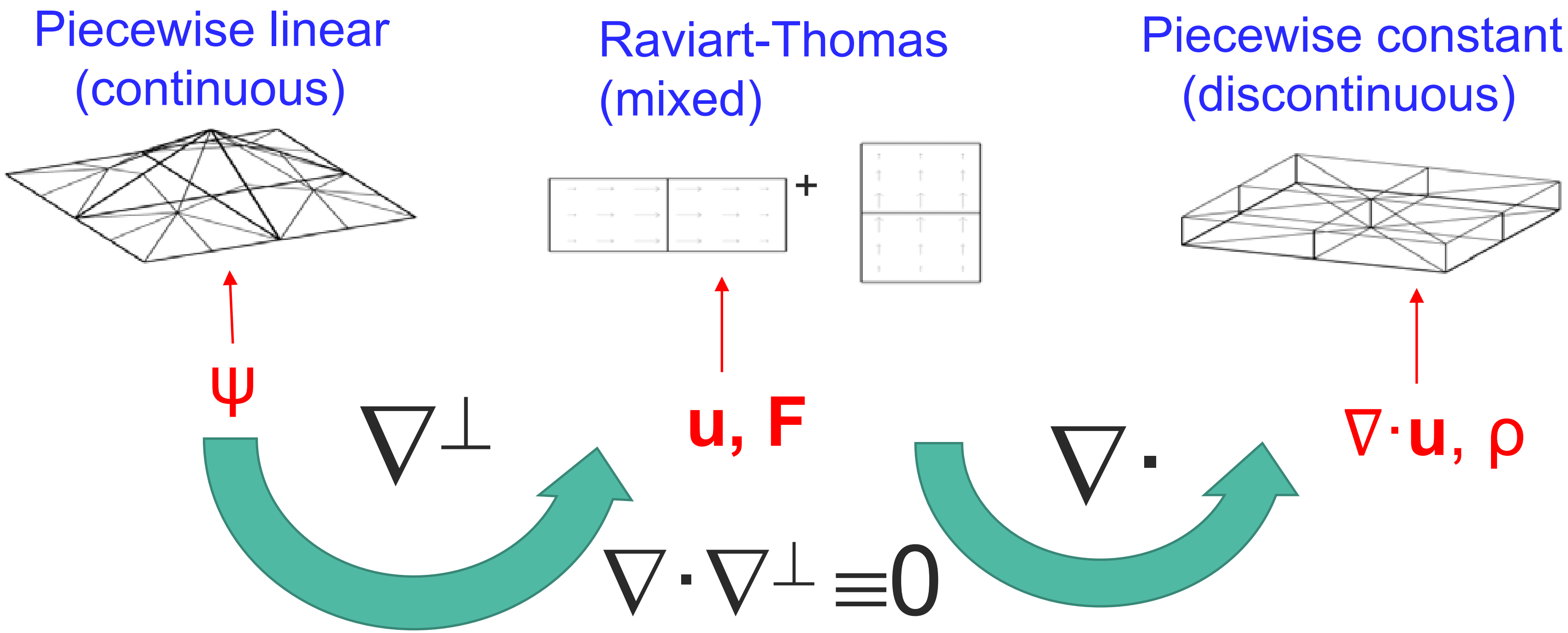
Met Office

GungHo Issues

- How to maintain accuracy of current model on a GungHo grid?
  Staniforth & Thuburn (2012)
- Principal points about current grid are:
  - Orthogonal, Quadrilateral, C-grid
- These allow good numerical aspects:
  - Lack of spurious modes
  - Mimetic properties
  - Good dispersion properties

Met Office

A way forward: mixed finite-element

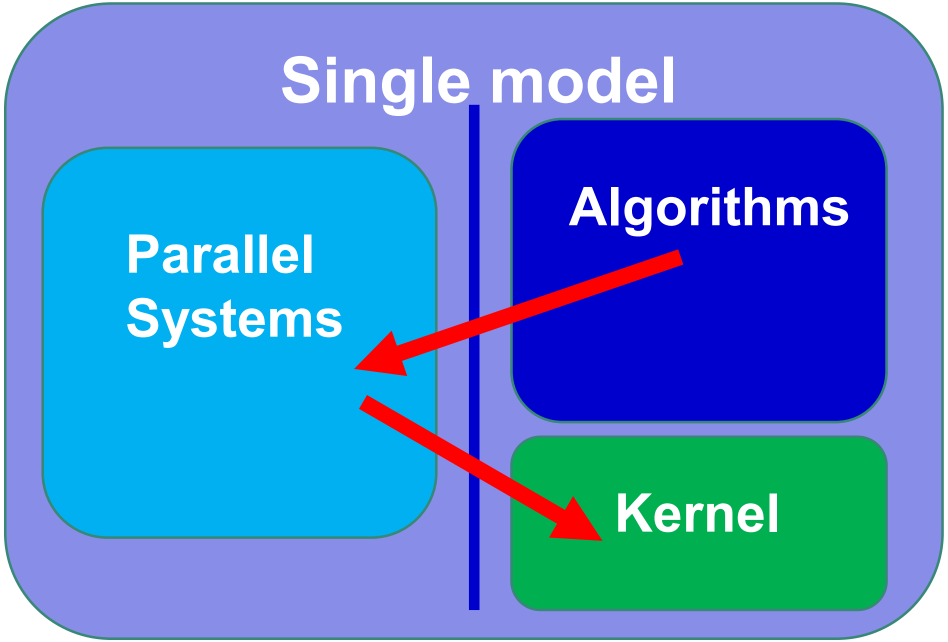


Exploiting ideas from discrete exterior calculus & differential geometry

Cotter (Imperial College) & Thuburn (Exeter)

The Future: LFRic

The separation of concerns: PSyKAl



- Indirect addressing for horizontal
- Vertical loop inner most
- F2003
- Code auto generation

Parallel Systems Kernel Algorithm = PSyKAl

Current code

(Coriolis terms $2\boldsymbol{\Omega} \times \boldsymbol{u}$)

All written by scientist

- Science code
- Horizontal looping
- Shared memory parallelism
- Distributed memory parallelism

```
! Compute work1 = (<vstar>^xi2)*f3_star - (<wstar>^eta)*f2_star

!$OMP PARALLEL DO DEFAULT(NONE) SCHEDULE(STATIC) PRIVATE(i,j,k)     &
!$OMP& SHARED(model_levels,pdims,vstar,f3_star,wstar,f2_star,work1)
  DO k=1, model_levels
    DO j=pdims%j_start, pdims%j_end
      DO i=pdims%i_start, pdims%i_end
        work1(i,j,k) = 0.5*(vstar(i,j,k)+vstar(i,j-1,k))*          &
                        f3_star(i,j,k) - 0.5*(wstar(i,j,k)+wstar(i,j,k-1))*     &
                        f2_star(i,j,k)
      END DO
    END DO
  END DO
!$OMP END PARALLEL DO

CALL swap_bounds(work1,                                              &
                 pdims_s%i_len - 2*pdims_s%halo_i,                   &
                 pdims_s%j_len - 2*pdims_s%halo_j,                   &
                 pdims_s%k_len,                                      &
                 pdims_s%halo_i, pdims_s%halo_j,                     &
                 fld_type_p,swap_field_is_vector, do_west_arg=.TRUE.)

!$OMP PARALLEL DO DEFAULT(NONE) SCHEDULE(STATIC) PRIVATE(i,j,k)     &
!$OMP& SHARED(model_levels,udims,u,r_u,beta_u_dt,work1,h1_xi1_u,dxi1_u)
  DO k=1, model_levels
    DO j=udims%j_start, udims%j_end
      DO i=udims%i_start, udims%i_end
        r_u(i,j,k) = u(i,j,k) + r_u(i,j,k) + beta_u_dt *              &
                     0.5*(work1(i,j,k)+work1(i+1,j,k)) /              &
                     ( h1_xi1_u(i,j,k)*dxi1_u(i) )
      END DO
    END DO
  END DO
!$OMP END PARALLEL DO
```

Met Office

LFRic code

Algorithm layer:
written by scientist

```
...
if ( rotating ) call invoke( rotation_kernel_type( rhs(igh_u), u, chi, qr ) )
...
```

Note:

- Only global fields referenced
- “Invoke” never actually called

Met Office

LFRic code

Kernel layer: written by scientist

- Loop over vertical only
- Rest is all science (loops over quadrature points and degrees of freedom, accessed by indirect addressing)
- No horizontal looping/no parallelism
- Where does that get done?

```
do k = 0, nlayers-1
  do df = 1, ndf_chi
    loc = map_chi(df) + k
    chi_e(:,df) = (/ chi_1( loc ), chi_2( loc ), chi_3( loc ) /)
  end do

  ! Calculate rotation vector Omega = (0, 2*cos(lat), 2*sin(lat)) and Jacobian
  call rotation_vector_sphere(ndf_chi, nqp_h, nqp_v, chi_e,      &
                              chi_basis, rotation_vector)
  call coordinate_jacobian(ndf_chi, nqp_h, nqp_v, chi_e,  &
                           chi_diff_basis, jac, dj)

  ! Compute the rotation component of RHS integrated over one cell
  do qp2 = 1, nqp_v
    do qp1 = 1, nqp_h
      u_at_quad(:) = 0.0_r_def
      do df = 1, ndf_w2
        u_at_quad(:) = u_at_quad(:) &
                     + u(map_w2(df) + k)*w2_basis(:,df,qp1,qp2)
      end do
      ! Rotation term
      jac_u = matmul(jac(:,:,qp1,qp2),u_at_quad)
      omega_cross_u = cross_product(rotation_vector(:,qp1,qp2), jac_u)
      do df = 1, ndf_w2
        v   = w2_basis(:,df,qp1,qp2)
        jac_v = matmul(jac(:,:,qp1,qp2),v)/dj(qp1,qp2)

        coriolis_term = dot_product(jac_v,omega_cross_u)

        r_u( map_w2(df) + k ) = r_u( map_w2(df) + k ) &
                              -  wqp_h(qp1)*wqp_v(qp2)*coriolis_term
      end do
    end do
  end do

end do

end subroutine rotation_code
```

The magic behind the curtain

PSy layer: autogenerated

- Shared memory parallelism
- Distributed memory parallelism

```
...
!
! Call kernels and communication routines
!
IF (rhs_tmp_proxy%is_dirty(depth=1)) CALL rhs_tmp_proxy%halo_exchange(depth=1)
IF (u_proxy%is_dirty(depth=1))       CALL u_proxy%halo_exchange(depth=1)
IF (chi_proxy(1)%is_dirty(depth=1))  CALL chi_proxy(1)%halo_exchange(depth=1)
IF (chi_proxy(2)%is_dirty(depth=1))  CALL chi_proxy(2)%halo_exchange(depth=1)
IF (chi_proxy(3)%is_dirty(depth=1))  CALL chi_proxy(3)%halo_exchange(depth=1)
!
! Look-up colour map
!
CALL rhs_tmp_proxy%vspace%get_colours(ncolour, ncp_colour, cmap)
!
DO colour=1,ncolour
  !$omp parallel default(shared), private(cell)
  !$omp do schedule(static)
  DO cell=1,ncp_colour(colour)
    !
    CALL rotation_code(nlayers, rhs_tmp_proxy%data, u_proxy%data, &
                       chi_proxy(1)%data, chi_proxy(2)%data, chi_proxy(3)%data, &
                       ndf_w2, undf_w2, map_w2(:,cmap(colour, cell)), basis_w2, &
                       ndf_any_space_9_chi, undf_any_space_9_chi, &
                       map_any_space_9_chi(:,cmap(colour, cell)), basis_any_space_9_chi, &
                       diff_basis_any_space_9_chi, nqp_h, nqp_v, wh, wv)
  END DO
  !$omp end do
  !$omp end parallel
END DO
!
! Set halos dirty for fields modified in the above loop
!
CALL rhs_tmp_proxy%set_dirty()
!
...
```

Summary

- Aim: keep the "quiet revolution" going
- Challenge: how to do that when hitting fundamental limitations with computers?
- The "free lunch" is over!

1. Need to expose as much parallelism of problem as possible (refactor codes/algorithms) (e.g. GungHo & similar)
2. Need to be flexible (separate concerns) to be able to use whatever HPC's of the future look like (e.g. LFRic & similar)

Met Office

Thank you! Any questions?

"It would appear that we have reached the limits of what is possible to achieve with computer technology, although one should be careful with such statements, as they tend to sound pretty silly in five years"

John von Neumann, 1949