Science & Technology
Facilities Council

工合海洋 GOcean

An update…

Andrew Porter, Rupert Ford & Graham Riley
STFC

Hedong Liu & Jason Holt
NOC

Project Overview

- NERC funded, 03/2014 – 02/2015
- Collaboration between NOC and STFC
- Existing NEMO code structure needs updating
  - 20 yrs old and MPI only, vertical level index last
- Investigate the feasibility of applying the GungHo approach to ocean modelling
- Ocean models can avoid pole problem
  - put poles over land; can retain a lat-lon grid
- **Look to extend the developing Gung-Ho infrastructure to support finite difference on lat-lon grids**

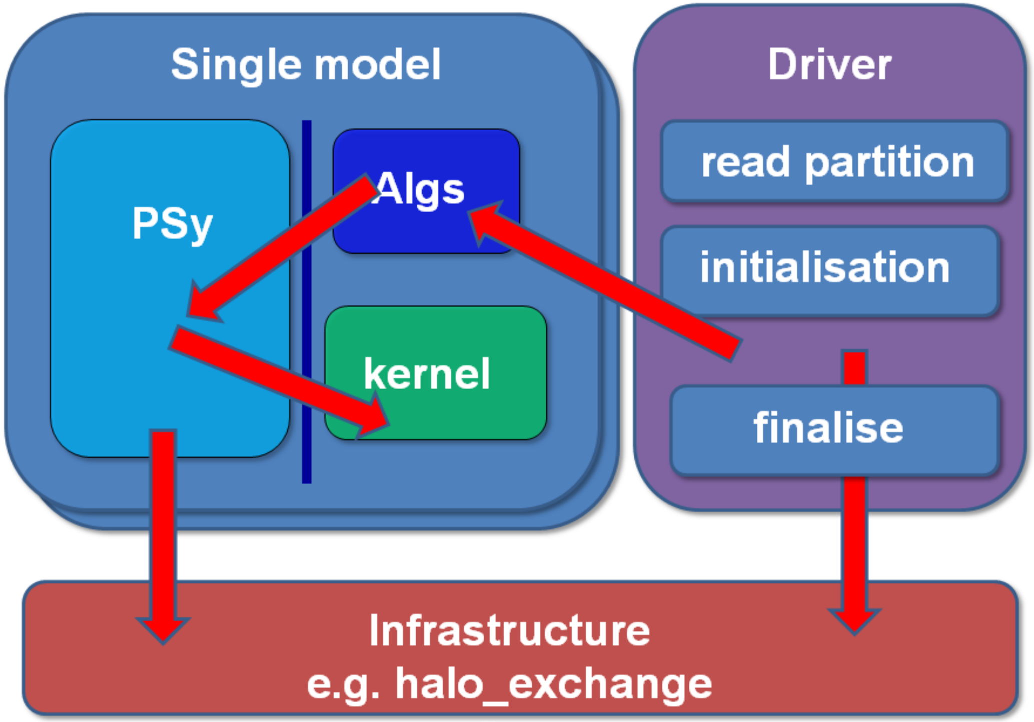
Science & Technology Facilities Council

The Plan…

- Obtain finite-difference, shallow-water model(s)
- Re-structure following PSyKAl approach
- Extend kernel meta-data to capture necessary information
  - Finite difference, lat-lon, direct-addressed
- Extend PSyclone to generate the Parallel System (middle layer)
  - Support for multi/many-core/GPU etc. but not MPI
- Investigate performance implications and feedback to PSyclone
- Feedback to the NEMO system team

Science & Technology Facilities Council

Separation of Concerns in Gung-Ho (recap)



The Parallel System, Kernel, Algorithm (PSyKAl) Approach…

- Oceanographer writes the algorithm and kernel layers, following certain rules
  - no need to worry about relative indexing of various fields
  - no need to worry about parallelism
- A code-generation system (PSyclone) generates the PSy middle layer
  - glues the algorithm and kernels together
  - incorporates all code related to parallelism

Science & Technology Facilities Council

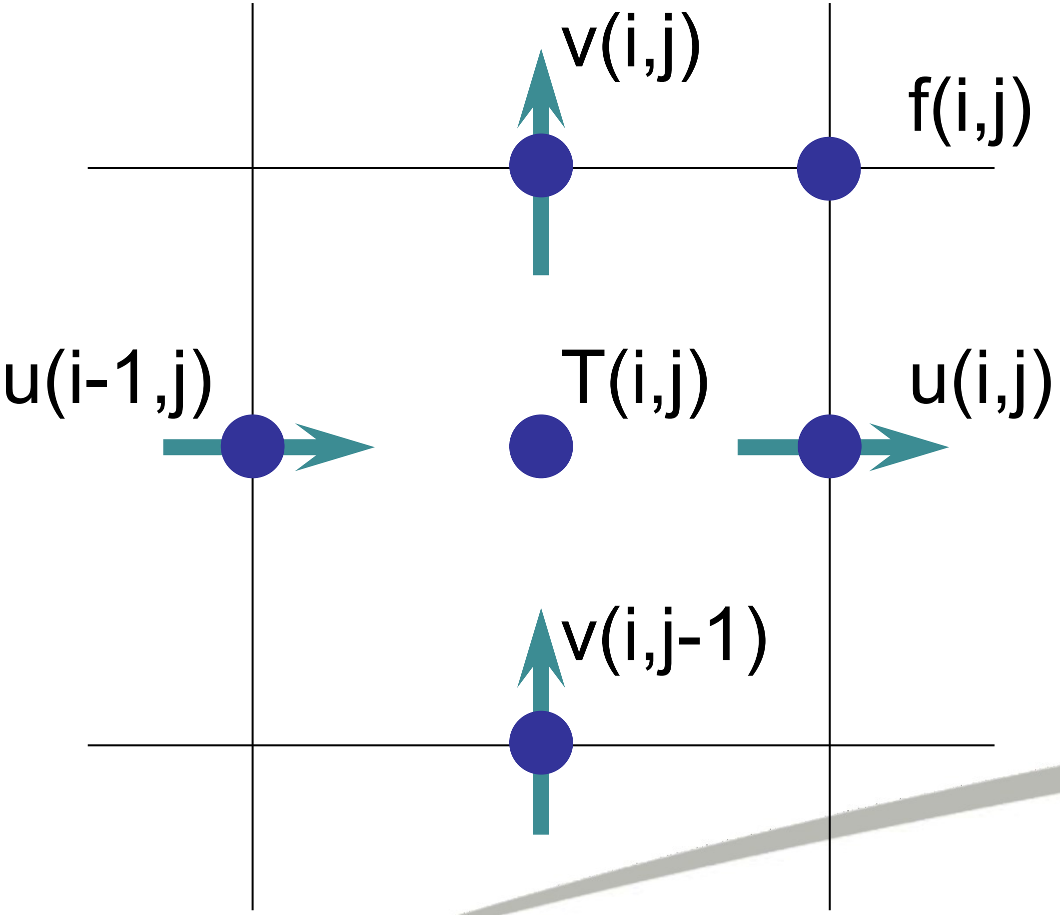
Two shallow-water codes…

- We are applying PSyKAl approach to two codes:
  - ‘shallow:’ originally written by Swarztrauber, NCAR
  - ‘NEMOLite2D:’ 2D, free surface part of NEMO extracted by NOC
- Both use Finite Difference on Arakawa C grid
- But there are important differences:
  - Boundary conditions (periodic vs. forced/closed)
  - Relative indexing of variables on the grid
- Understanding and expressing these differences is essential for correct code generation

Science & Technology Facilities Council

Placing variables...

- Variable placement for the Staggered Arakawa C-grid with NEMO indexing convention:

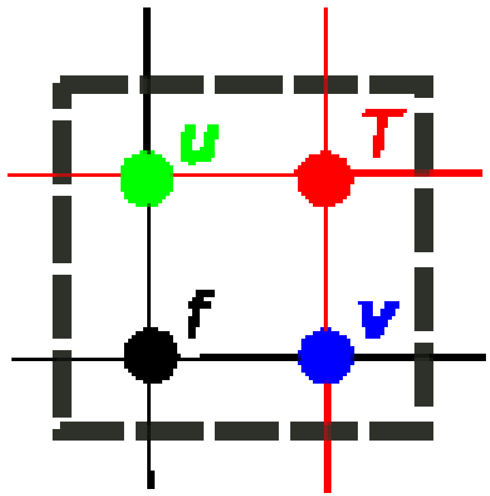


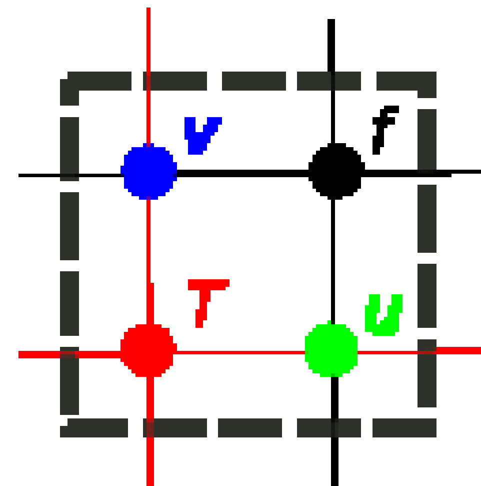
T: scalars (density, salinity etc.)
u,v: velocity components
f: vorticity

Science & Technology Facilities Council

Offset choice

- A *developer* can choose how the different grid-point types are indexed relative to T
- shallow defines {u,v,f} points to the South and West of the T point to have same (i,j) index while NEMO uses those to the North and East:





- We call this choice the ‘offset’ of the grids
- Specified in kernel meta-data

Science & Technology Facilities Council

PSyKAl-ification...

Take shallow-water code, e.g.:

```
DO ncycle=1,itmax   ! Time-stepping loop
! COMPUTE CAPITAL U, CAPITAL V, Z AND H
  DO J=1,N
    DO I=1,M
      CU(I+1,J) = &
         .5*(P(I+1,J)+P(I,J))*U(I+1,J)
      CV(I,J+1) = &
         .5*(P(I,J+1)+P(I,J))*V(I,J+1)
      ...
    END DO
  END DO
```

Construct (point-wise) kernels, *e.g.*:

```
SUBROUTINE compute_cu_code(i, j, &
                           cu, p, u)
  INTEGER,  INTENT(in) :: i, j
  REAL(wp), INTENT(in) :: p(:,:), u(:,:)
  REAL(wp), INTENT(inout):: cu(:,:)

  CU(I,J) = .5*(P(I,J)+P(I-1,J))*U(I,J)

END SUBROUTINE compute_cu_code
```

Science & Technology Facilities Council

Re-structure following Gung-Ho rules (all computation must be done in a kernel):

Time-stepping loop at algorithm level becomes….

```
DO ncycle=1,itmax ! Time-stepping loop

  ! COMPUTE CAPITAL U, CAPITAL V, Z AND H
  CALL compute_cu(CU, P, U)
  CALL compute_cv(CV, P, V)
  ...
  CALL invoke(compute_cu(CU, P, V), &
              compute_cv(CV, P, V), &
              ...)
```

Science & Technology Facilities Council

Translation/generation…

```
PROGRAM shallow
  USE psy_shallow, ONLY: invoke_0
  …
  !   ** Start of time loop **
  DO ncycle=1,itmax

    ! COMPUTE CAPITAL U, CAPITAL V, Z, H
    CALL invoke_0(cu, p, u, cv, v, z, h)
```

Science & Technology Facilities Council

```
SUBROUTINE invoke_0(cu_1,p,u,cv_1,v,z,h)
  USE compute_cv_mod, ONLY: compute_cv_code
  USE compute_cu_mod, ONLY: compute_cu_code
  USE topology_mod,   ONLY: cu, cv
  REAL,intent(inout),dimension(:,:) :: cu_1,p,u,cv_1,v,z,h
  DO i=cu%istart,cu%istop
    DO j=cu%jstart,cu%jstop
      CALL compute_cu_code(i, j, cu_1, p, u)
    END DO
  END DO
  DO i=cv%istart,cv%istop
    DO j=cv%jstart,cv%jstop
      CALL compute_cv_code(i, j, cv_1, p, v)
    …
```

Kernel meta-data

- Kernels make use of several grid-related quantities, *e.g*. area of cell around a T point, T-point mask etc.

```
subroutine next_sshu_code(ji,jj, sshn_u, sshn, &
                          tmask,e12t,e12u)
  implicit none
  integer,                    intent(in)    :: ji, jj
  integer,  dimension(:,:), intent(in)    :: tmask
  real(wp), dimension(:,:), intent(in)    :: e12t, e12u
  real(wp), dimension(:,:), intent(inout) :: sshn_u
  real(wp), dimension(:,:), intent(in)    :: sshn
```

- The algorithm layer should not/cannot supply these:

```
call invoke( next_sshu(sshn_u, sshn), &
             next_sshv(sshn_v, sshn) )
```

Science & Technology Facilities Council

- Extend meta-data to specify what quantities a kernel requires from the infrastructure:

```
type, extends(kernel_type) :: next_sshu
   type(arg), dimension(5) :: meta_args =  &
        (/ arg(READWRITE, CU, POINTWISE),  &
           arg(READ,      CU, POINTWISE),  &
           arg(READ,      GRID_MASK_T),    &
           arg(READ,      GRID_AREA_T),    &
           arg(READ,      GRID_AREA_U)     &
         /)
```

- PSyclone must then supply these quantities from the generated middle layer (*e.g*. by accessing a grid objected pointed to by a field object)

Science & Technology Facilities Council

What about performance?

Some timings for 2000 time steps of 128x128 domain on a single core:

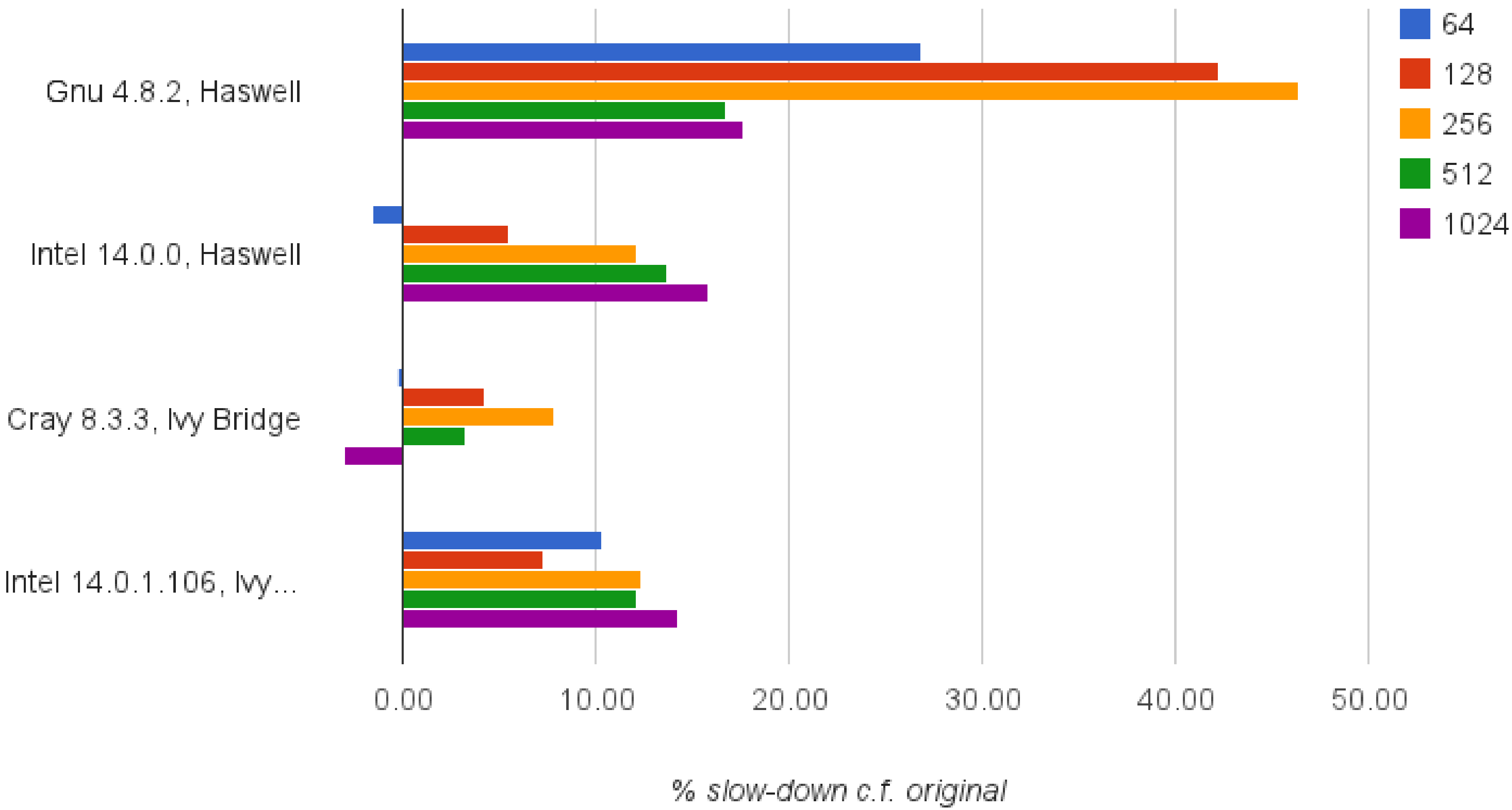
Compiler:	Gnu 4.8.2	Intel 14.0.0	Cray 8.2.6	Intel 14.0.1
CPU:	Xeon E5-1620 v3, 3.7 GHz (Haswell)	Xeon E5-1620 v3, 3.7 GHz (Haswell)	Xeon E5-2697, 2.7 GHz (Ivy Bridge)	Xeon E5-2697, 2.7 GHz (Ivy Bridge)
Original*:	0.37	0.37	0.29	0.40
Manual, vanilla†:	6.30	0.42	0.41	0.49
Manual, opt'd‡:	0.53	0.39	0.31	0.43

* Unmodified shallow code
† Manual PSyKAl version
‡ As † but with middle layer optimised

Science & Technology Facilities Council

All compilers are not created equal

Performance of best (so far) PSyKAI version compared to original, unmodified 'shallow':



Next steps…

- Supply vendors (IBM, NVIDIA) with original and PSyKAl'ised versions and let them optimise
  - Use lessons learned to continue to improve PSyclone
- Gain fuller understanding of cost (if any) of introducing layered structure
- Currently not exploring the optimisation space
  - Only attempting to recover original code structure
- Three dimensions
  - Current test cases are two-dimensional
  - Full models are a mixture of 2D and 3D…
  - NOC working on introducing some 3D aspects to NEMOLite

Science & Technology Facilities Council

Summary I

- Separation of Concerns: A practical approach to marrying the requirements of oceanographers with the requirements for performance in the (pre-) exascale era.
- Introduces flexibility needed to achieve performance on different architectures
  - potentially enables *e.g*. OpenMP or OpenACC to be used, depending on target hardware
- Very dependent on middle layer to retrieve the performance that we've thrown out

Science & Technology Facilities Council

Summary II

- Framework now supports two distinct shallow-water models
- Basic code generation functional
  - Support for loop fusion and OpenMP transformations
- Working in collaboration with hardware vendors to understand what loses us performance and how to regain it
- Potential to explore optimisation space that has been opened by the flexibility provided by code transformation & generation

Extras…

Science & Technology Facilities Council

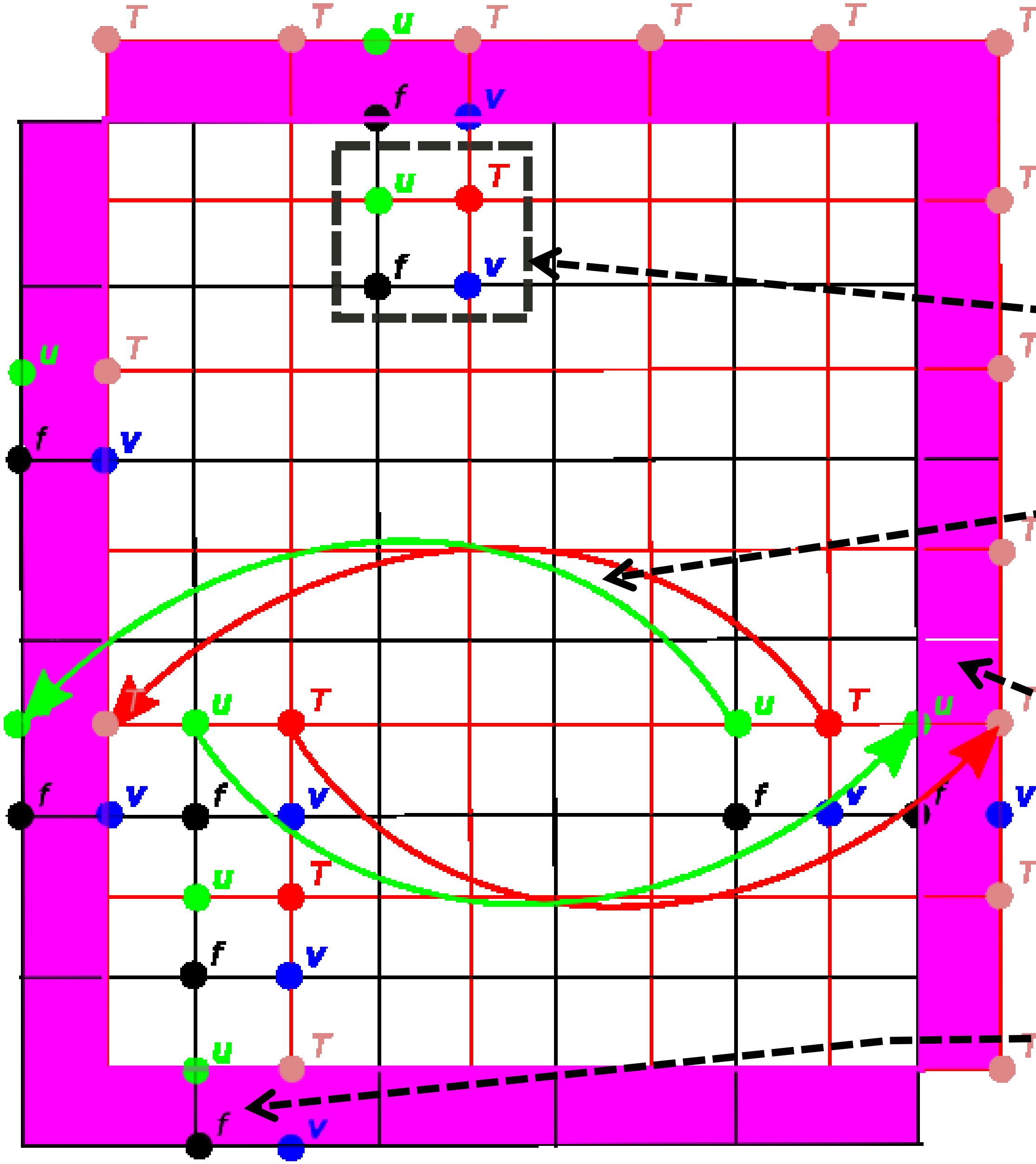
Middle layer is generated…

```
!   ** Start of time loop **
DO ncycle=1,itmax

  ! COMPUTE CAPITAL U, CAPITAL V, Z, H
  call invoke(compute_cu_type(CU, P, U),   &
              compute_cv_type(CV, P, V),   &
              compute_z_type(z, P, U, V), &
              compute_h_type(h, P, U, V) )
```

Science & Technology
Facilities Council

SW offset with Periodic Boundaries



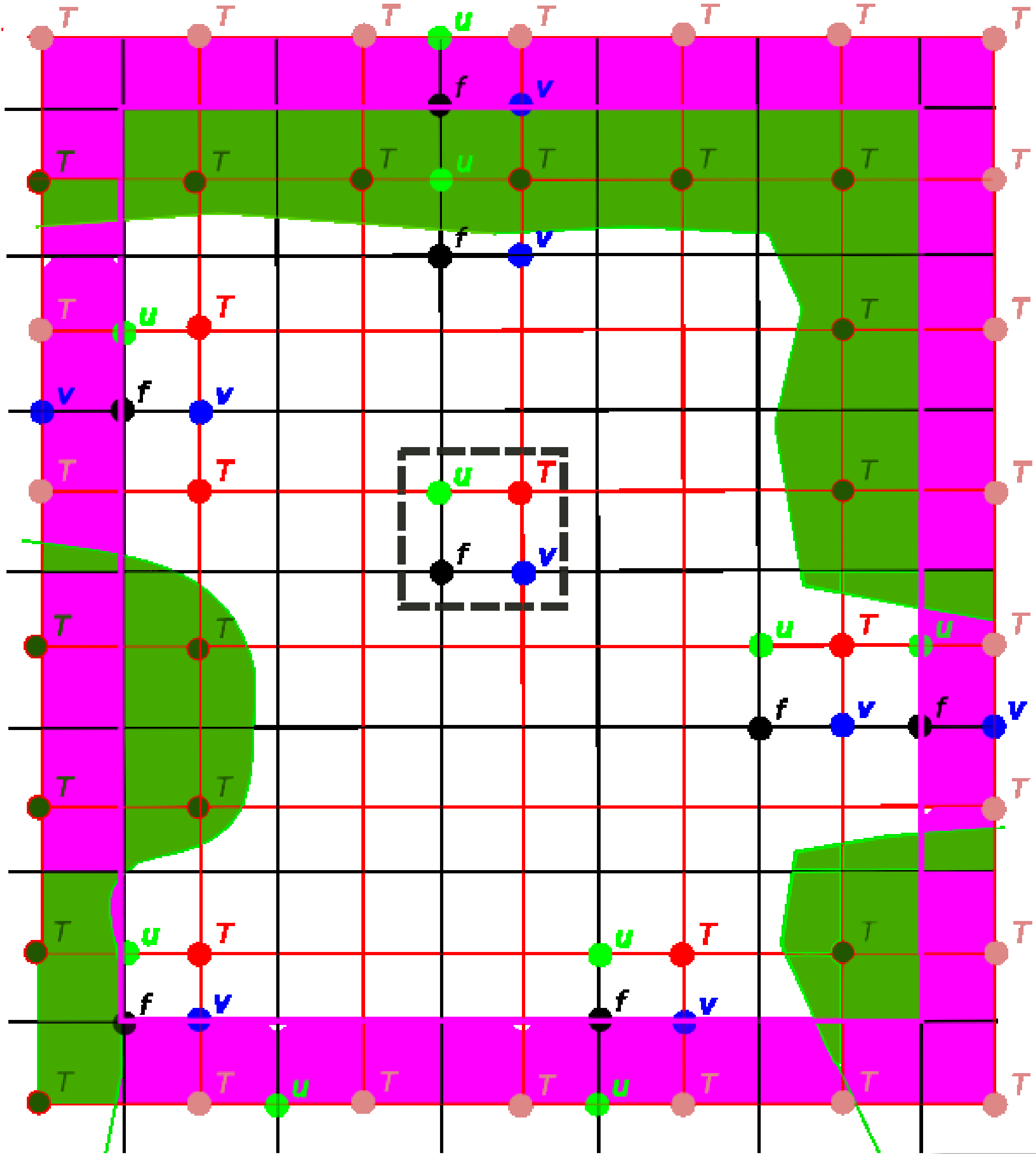
Grid points with same (i,j)

Data movement for PBC update

Boundary region

External/boundary grid points

Science & Technology Facilities Council

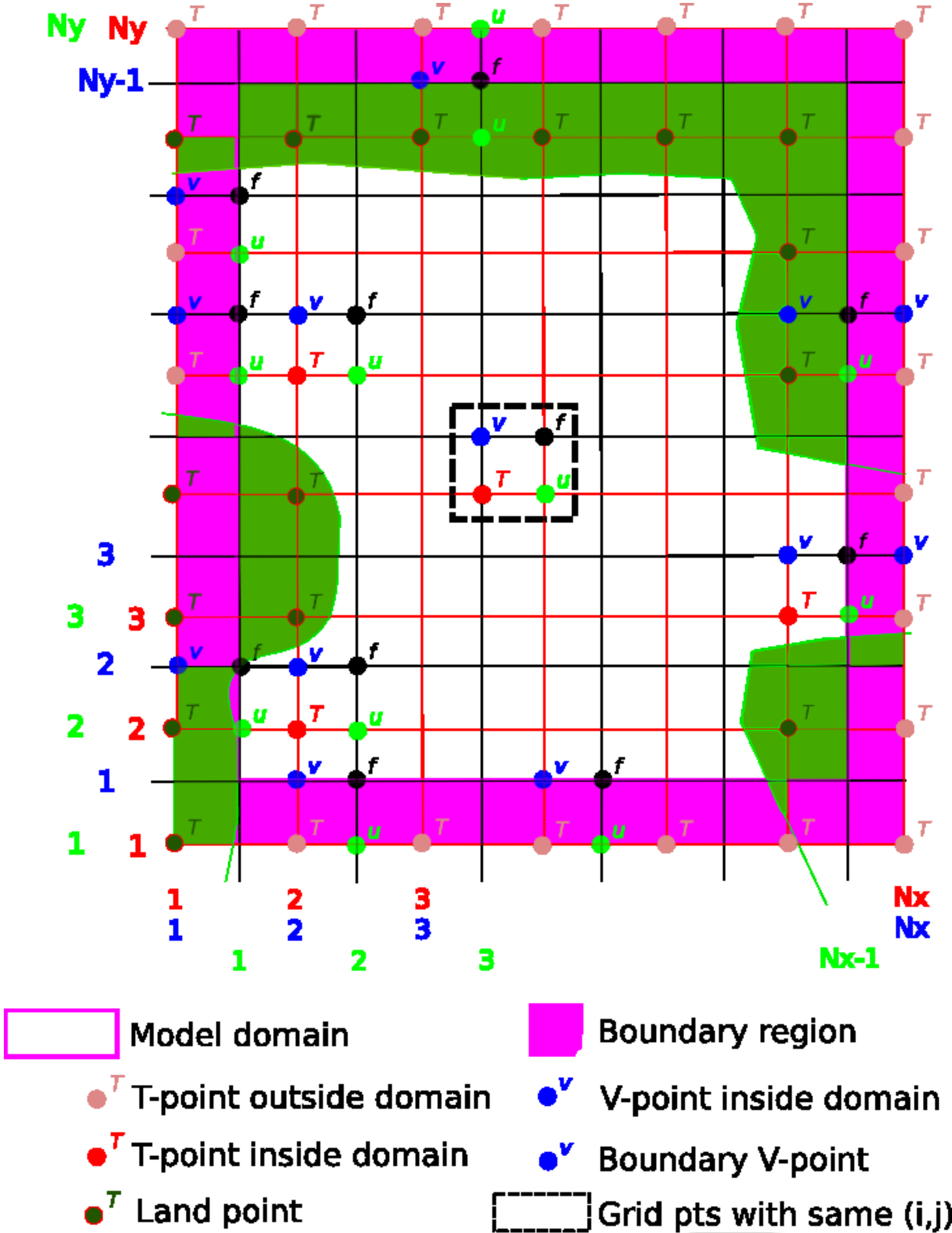


SW offset with open and closed boundaries

- User defines domain in terms of T points
- Definition *includes* the boundary points
    - Serial implementation doesn't need halos

Science & Technology Facilities Council

NE offset with in-place boundary conditions



Code generation to the rescue…

- PSyclone currently has rudimentary support for loop-fusion (and the addition of OpenMP)
- *e.g.* for the time-smoothing section of the code where all 3 loops have same bounds:

```
DO j=1,SIZE(uold,2)
  DO i=1,SIZE(uold,1)
    CALL tsmooth_code(i,j,u…)
DO i=1,SIZE(vold,1)
  DO j=1,SIZE(vold,2)
    CALL tsmooth_code(i,j,v…)
DO i=1,SIZE(pold,1)
  DO j=1,SIZE(pold,2)
    CALL tsmooth_code(i,j,p…)
```

⇒

```
DO j=1, SIZE(uold, 2)
  DO i=1, SIZE(uold, 1)
    CALL tsmooth_code(i,j,u…)
    CALL tsmooth_code(i,j,v…)
    CALL tsmooth_code(i,j,p…)
  END DO
END DO
```

Science & Technology Facilities Council