Science & Technology Facilities Council

工合海洋 GOcean

Applying the GungHo Computational Framework to Ocean Modelling

Mike Ashworth, Rupert Ford, Andrew Porter,
STFC Daresbury Laboratory

Jason Holt, Hedong Liu,
National Oceanography Centre

Graham Riley, University of Manchester & STFC

GOcean Project Overview

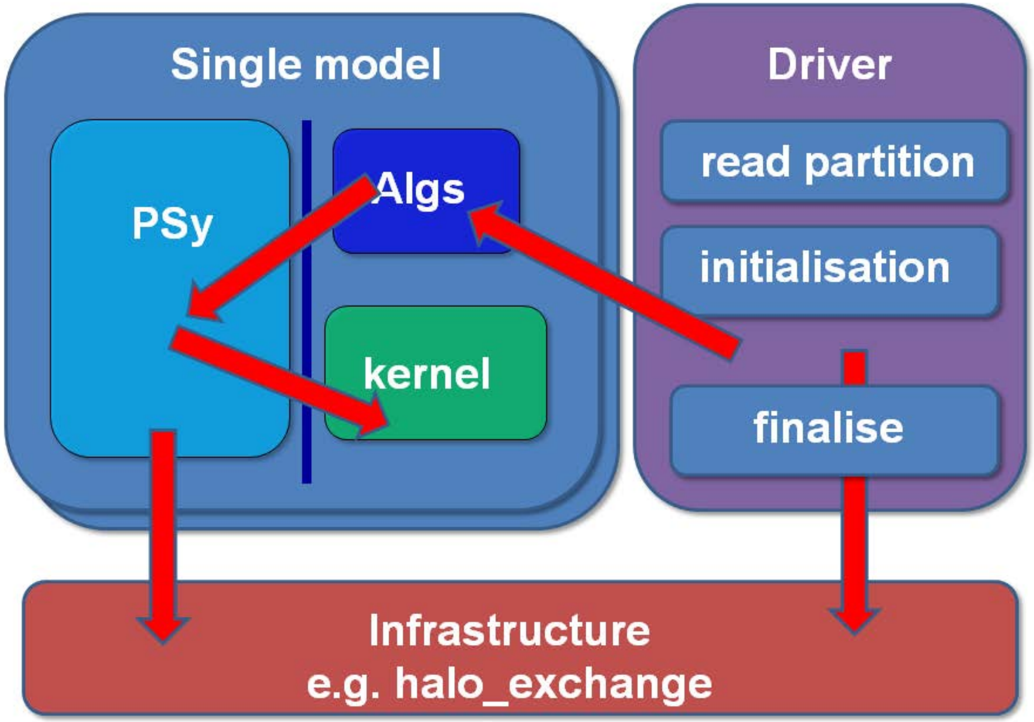
- Investigate the feasibility of applying the GungHo approach to ocean modelling
- The GungHo Project is tackling atmospheric modelling and has adopted an unstructured mesh to work around the pole problem
- Ocean models can put the poles over land
  $\Rightarrow$ can retain a latitude-longitude grid
- Extend the developing GungHo infrastructure to support lat-long grids
- NOC and STFC joint project: 1.4 FTEs
- NERC Technology Proof of Concept Fund

Science & Technology
Facilities Council

Support for finite difference in GungHo

- Could use GungHo indirect addressing model as-is
- Investigate extending GungHo to support direct addressing
  - Simpler, more intuitive kernels?
  - Better performance?

Separation of Concerns in GungHo



Separation of Concerns…

- Oceanographers writes the algorithm and kernel layers, following certain rules
- A code-generation system (PSyClone) generates the PSy middle layer
  - parses the Fortran code using fparser from the f2py project
  - glues the algorithm and kernels together
  - incorporates all code related to parallelism

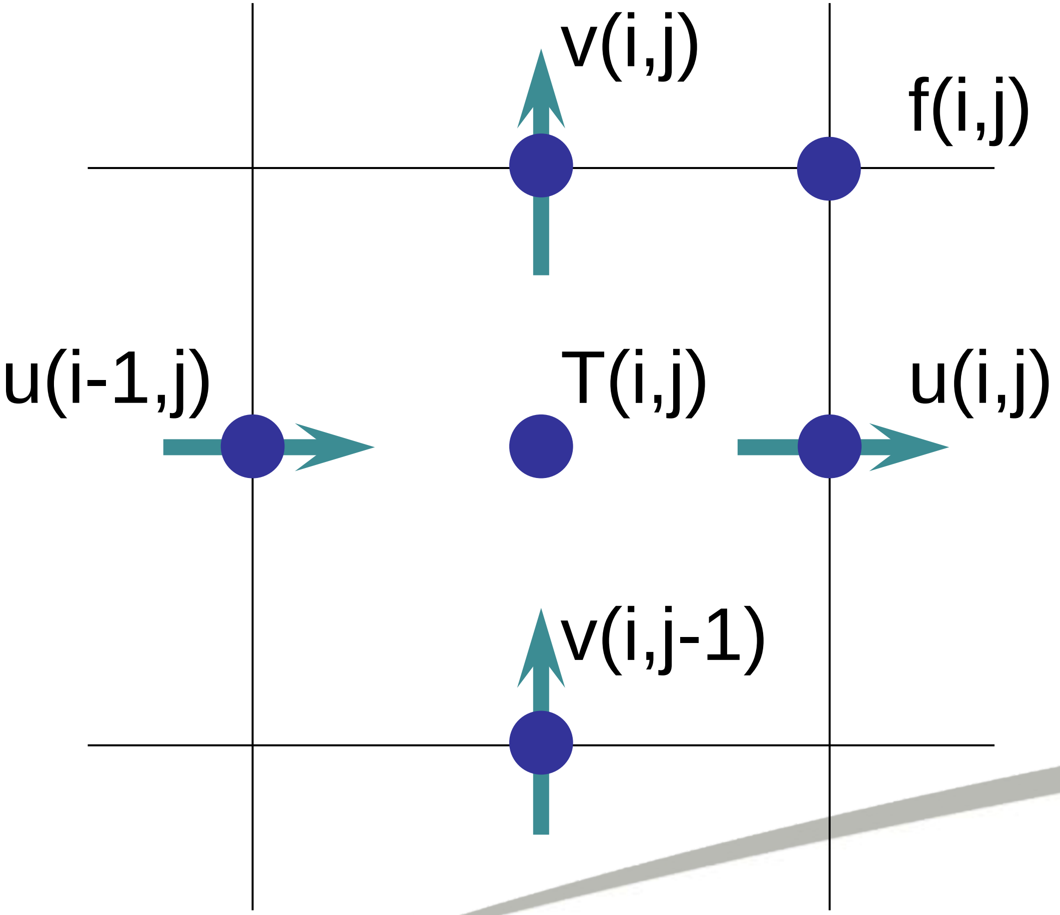
Two Test Codes for Gocean…

- GOcean project is applying GungHo approach to two codes:
  - ‘shallow:’ originally written by Swarztrauber, NCAR
  - ‘Gocean2D:’ 2D, free surface extracted from NEMO
- Both use Finite Differences on an Arakawa C grid
- But there are important differences:
  - Boundary conditions
  - Indexing of variables
- Understanding and expressing these differences is essential for correct code generation

Science & Technology Facilities Council

Placing variables...

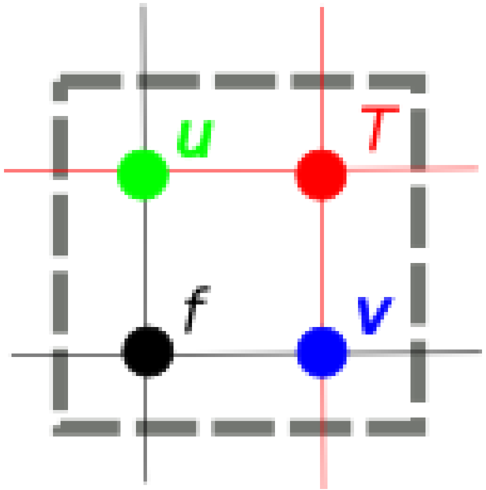
- Variable placement for the Arakawa C-grid with NEMO indexing convention:



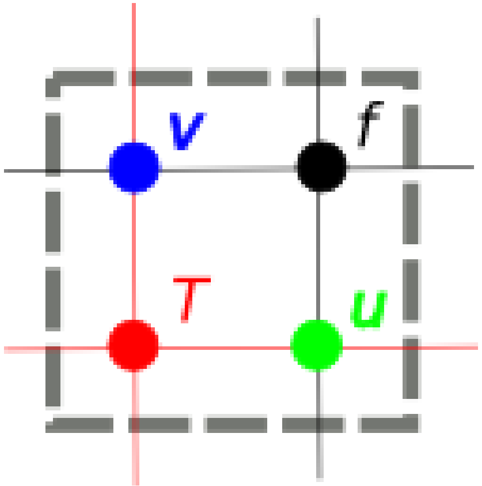
T: scalars (density, salinity etc.)
u,v: velocity components
f: vorticity

Indexing 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:



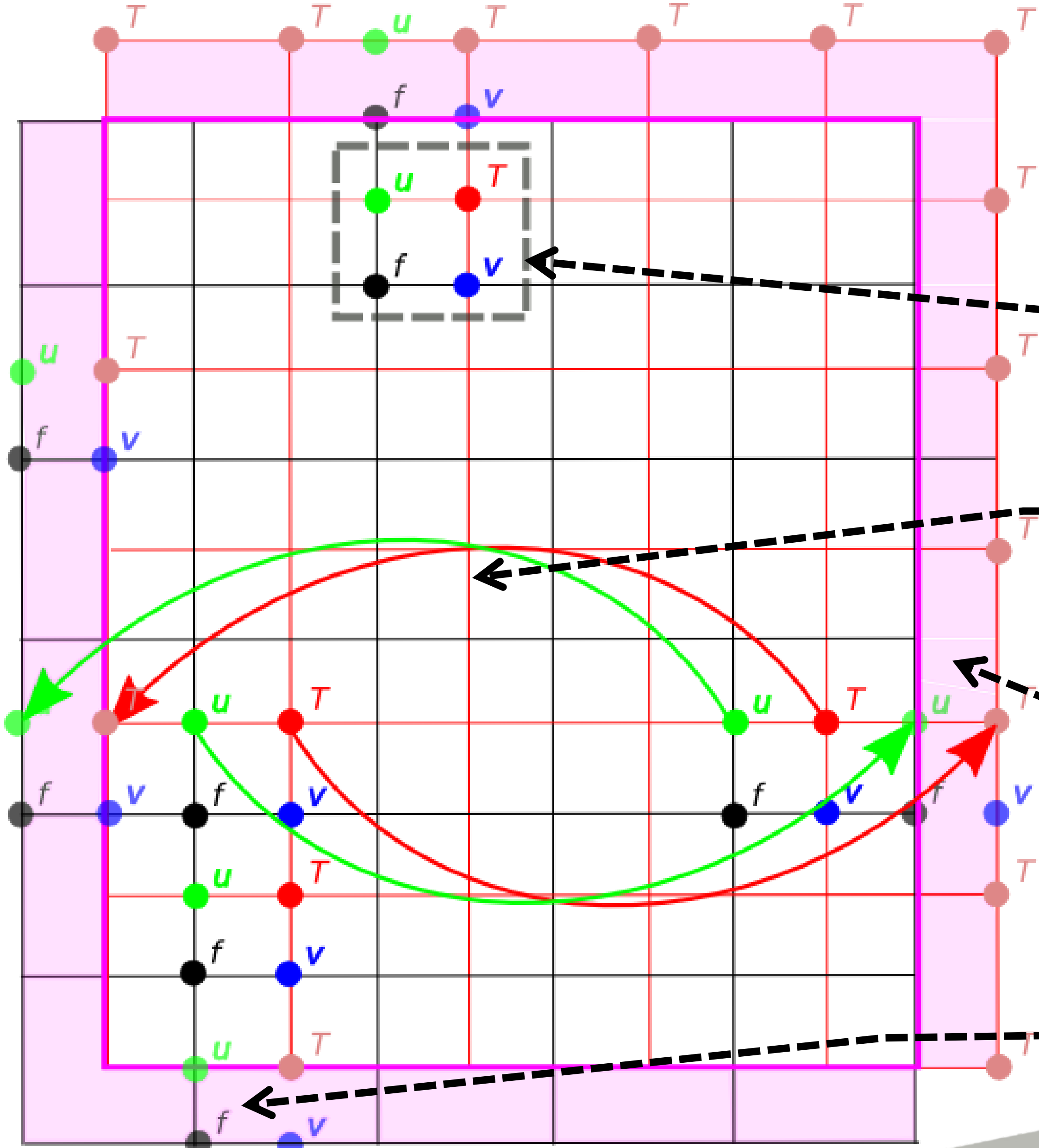
shallow



NEMO

We call this choice the 'indexing' of the grids

shallow: SW indexing with Periodic Boundaries



Grid points with same (i,j)

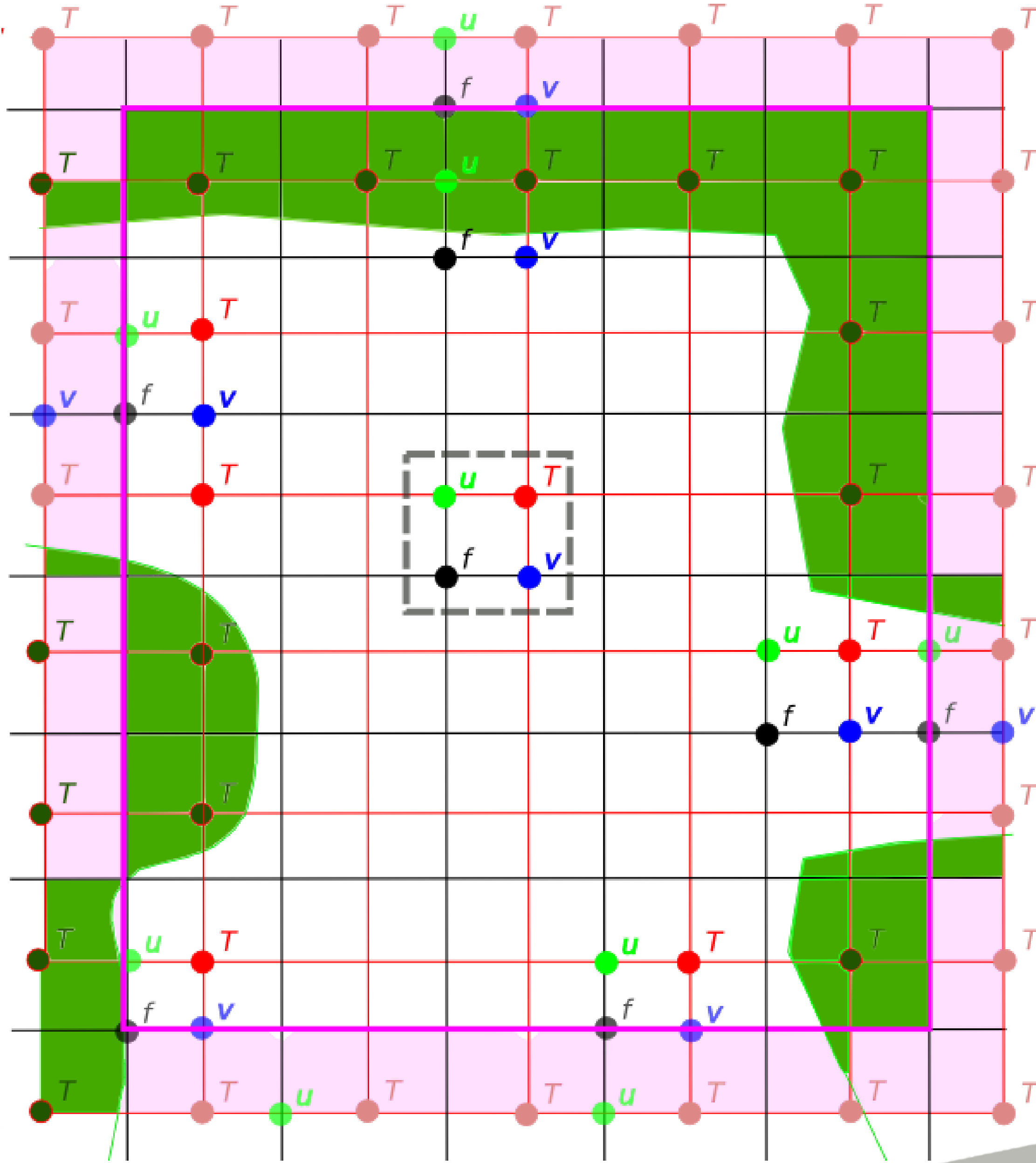
Data movement for periodic BC update

Boundary regions

External/boundary grid points

Science & Technology Facilities Council

shallow: SW indexing 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

PSyKAl-ification...

Original 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
```

Science & Technology Facilities Council

Re-structure following GungHo rules

Time-stepping loop at algorithm level becomes….

```
!  ** 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) )
```

a 'block' of code carrying out a specific function is replaced by 'call invoke' with four objects

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
```

the kernel is expressed as a simple one-point calculation on (i,j)

PSyclone generated code 1

```
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)
```

the algorithm layer remains mostly unchanged with a simple call to the PSy

PSyclone generated code 2

```
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) …
```

loop ranges express DM sub-domains and/or tiling

calls to kernels can be readily inlined by the compiler

But what about performance?

Some initial 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*:	0.38	0.37	0.32	0.41
Manual†:	3.1	0.62	0.53	0.66
Generated‡:	6.4	0.61	0.53	0.65

* Unmodified shallow code
† Manual PSyKAl version
‡ Generated PSyKAl version

Science & Technology Facilities Council

- Initial performance results encouraging…
- Re-structuring has ‘only’ slowed-down the code by 60-70%
  - (Apart from Gnu compiler where is 10x slower)
- Slow-down primarily due to splitting-up loops that are fused in original version:

```
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)
    Z(I+1,J+1)=(FSDX*(V(I+1,J+1)-V(I,J+1))-…
    H(I,J) = P(I,J)+.25*(U(I+1,J)*U(I+1,J)+…
  END DO
END DO
```

Science & Technology Facilities Council

Code generation to the rescue...

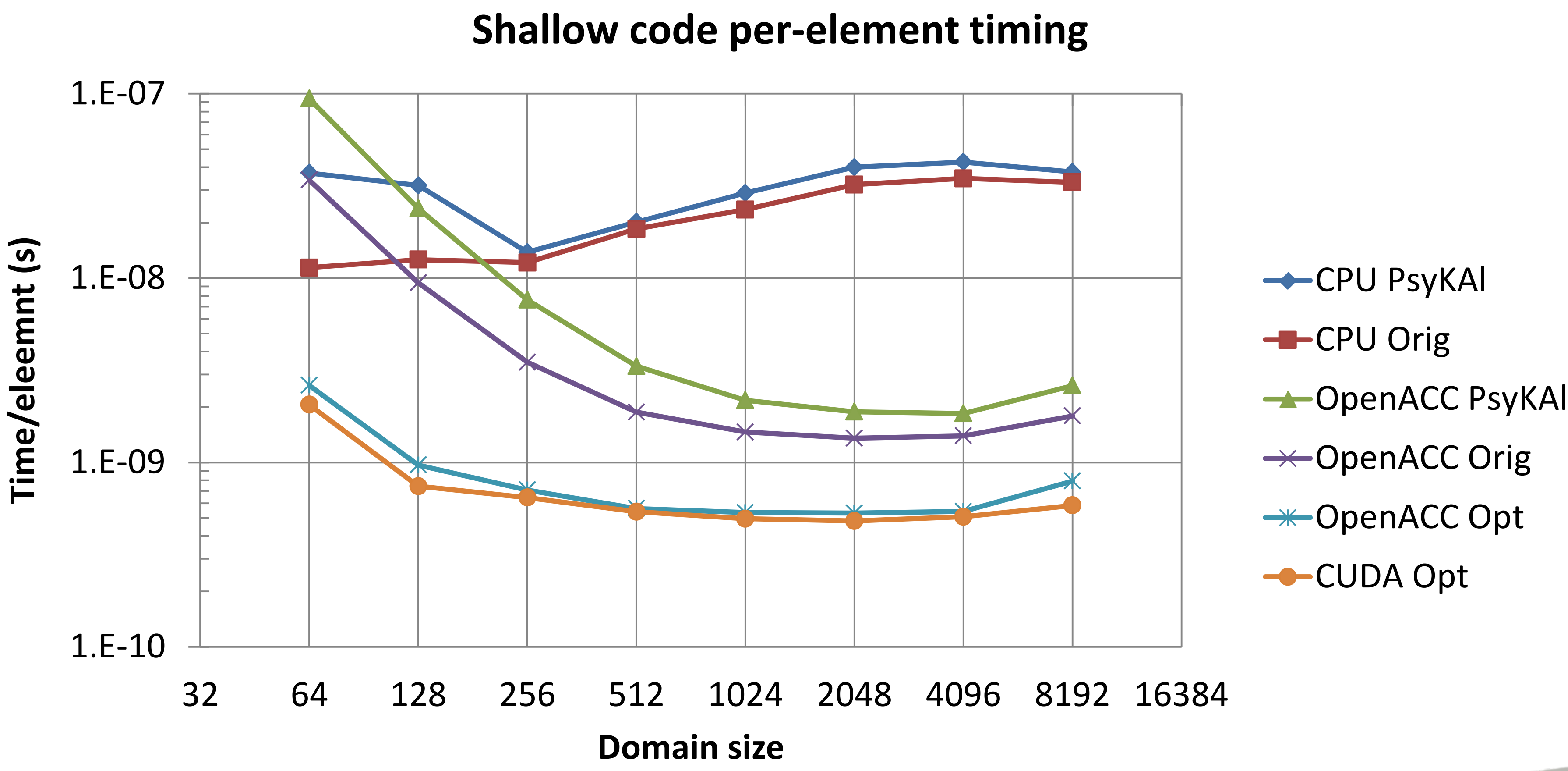
- PSyclone currently has rudimentary support for loop-fusion (and the addition of OpenMP)
- *e.g*. for the time-smoothing section of the code where all 3 loops have same bounds:

```
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

PSyKAl-ised code on GPUs



from Jeremy Appleyard, NVIDIA

GPU: NVIDIA Kepler K40, ECC on, Boost max.

CPU: Intel Xeon CPU E5-2650 v2 @ 2.60GHz

Science & Technology Facilities Council

Next steps…

- Supply vendors with original and PSyKAl-ised versions and let them optimise
  - Use lessons learned to improve PSyclone
  - Started already with NVIDIA
- PSyclone development
  - Continue work on Loop Fusion and OpenMP transforms
  - Add OpenACC transforms
- Three dimensions
  - Current test cases are two-dimensional.
  - Full models are a mixture of 2D and 3D…

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 run up to Exascale
- Introduces flexibility needed to achieve performance on different architectures
  - e.g. potentially enables index ordering to be swapped 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 as well as FE for GungHo
- Basic code generation is working
  - Support for loop fusion and OpenMP transformations
- Performance
  - Working in collaboration with hardware vendors to understand what loses us performance and how to regain it

Thank you!

Mike Ashworth, Rupert Ford, Andrew Porter, Jason Holt, Hedong Liu, Graham Riley

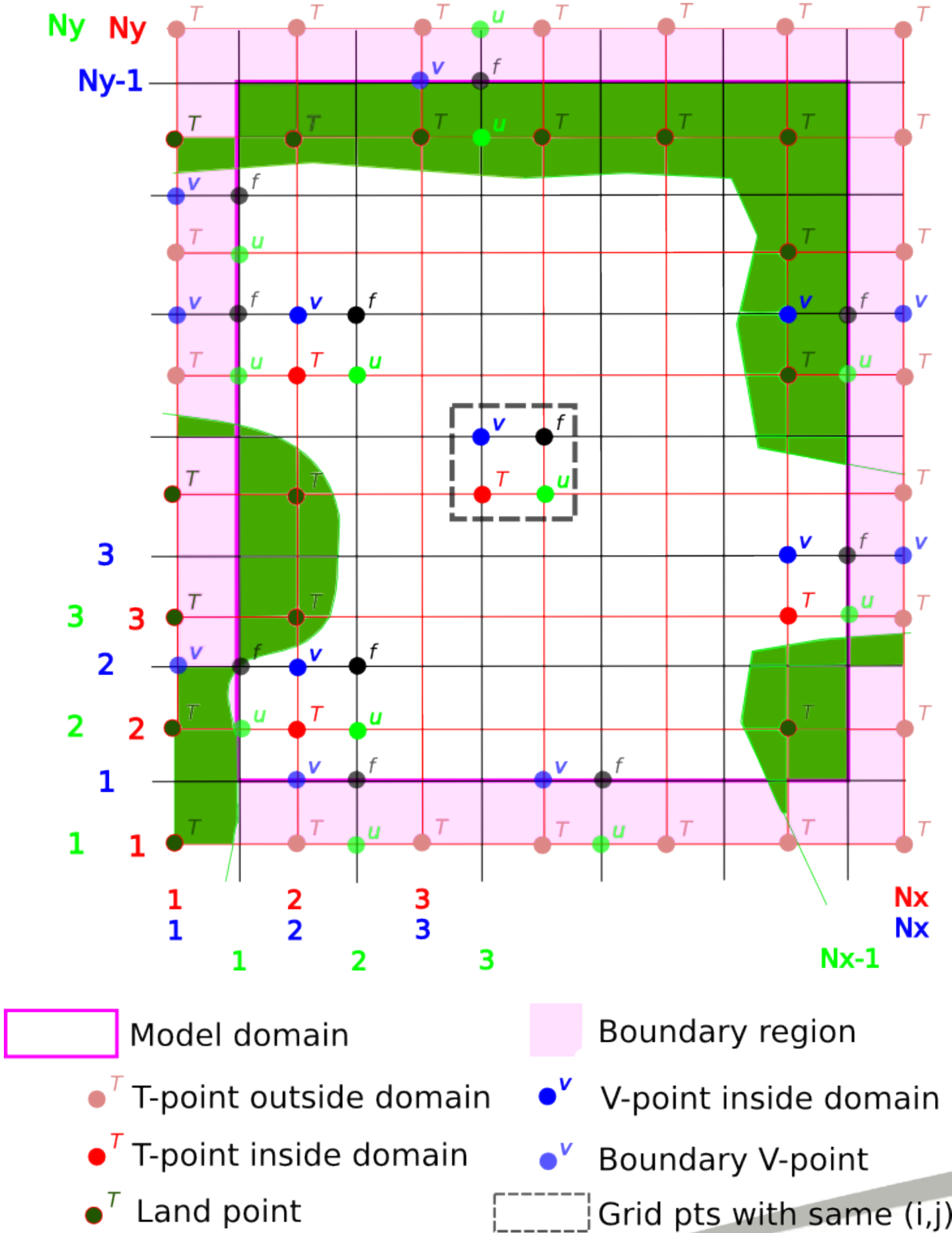
mike.ashworth@stfc.ac.uk

http://www.stfc.ac.uk/scd

GOcean is funded through grants NE/L01209X/1 and NE/L012111/1 from the Natural Environment Research Council

Extras…

NEMO: NE indexing with in-place boundary conditions



Science & Technology Facilities Council