Science and Technology Facilities Council

An Introduction to PSyclone

Rupert Ford, **Andrew Porter**, Sergi Siso, STFC Hartree Centre
Iva Kavcic, Chris Maynard, UK Met Office
Joerg Henrichs, Australian Bureau of Meteorology

ESIWACE2 Summer School, 24th August 2020

Hartree Centre
Science & Technology Facilities Council

Met Office

Australian Government
Bureau of Meteorology

Overview

1. Motivation
2. PSyclone
   a. What it is and what it does
   b. Modes of Operation
   c. Levels of Abstraction
3. The LFRic Domain
4. The NEMO Domain
5. Other Features

ESiWACE2 has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 823988

Motivation

See previous talk on DSLs but essentially:

- 3P's : Performance, Portability and Productivity
  - Maintainable high performance software
  - Single-source science code
  - Performance portability
- Complex parallel code + Complex parallel architectures + Complex compilers = Complex optimisation space => unlikely to be a single solution
- Single-source optimised code is unlikely to be possible
- So … separate science specification/code from code optimisation

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

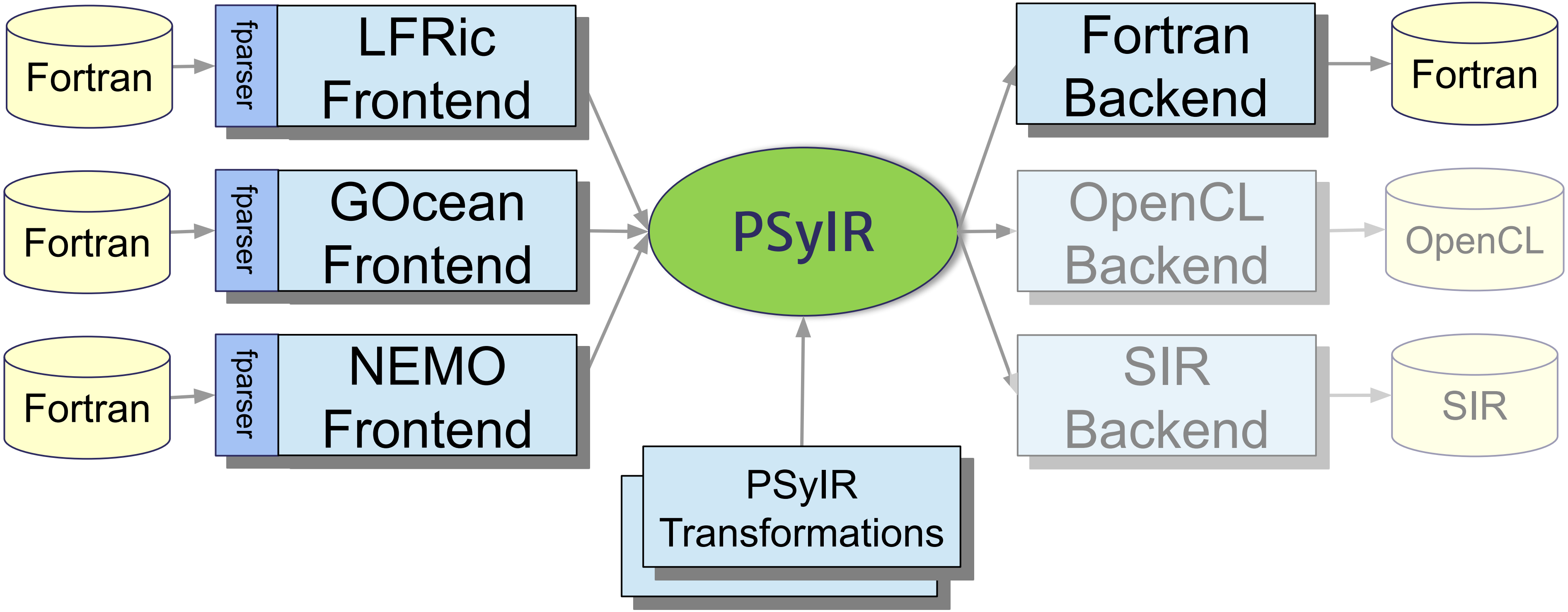
PSyclone

PSyclone 1.9.0
BSD 3-clause
https://github.com/stfc/PSyclone
https://psyclone.readthedocs.io
```
> pip install psyclone
```

- A domain-specific compiler for embedded DSL(s)
  - Configurable: FD/FV NEMO, GOcean, FE LFRic
  - Currently Fortran -> Fortran/OpenCL
  - Supports distributed- and shared-memory parallelism
  - Supports code generation and code transformation
- A tool for use by HPC experts
  - Hard to beat a human (debatable)
  - Work round limitations/bugs
  - Optimisations encoded as a 'recipe' rather than baked into the scientific source code
  - Different recipes for different computer architectures
  - Enables scriptable, whole-code optimisation

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

Basic Structure



ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

Fparser

- Pure Python Fortran parser
- Supports Fortran 2003 + some 2008
- Open source BSD3 licence
- Developed on GitHub
- Can fully parse UM, LFRic and NEMO source
- Work-in-progress to parse IFS source
- Used by PSyclone, Stylist, Loki

https://github.com/stfc/fparser
https://fparser.readthedocs.io/

```
> pip install fparser
```

```fortran
PROGRAM copy_stencil
  IMPLICIT NONE
  INTEGER, PARAMETER :: n = 10, np1 = 11
  INTEGER :: i, j, k
  REAL, DIMENSION(np1, n, n) :: out, in
  DO k = 1, n
    DO j = 1, n
      DO i = 1, n
        out(i, j, k) = in(i + 1, j, k)
```

```
child type =  <class 'fparser.two.Fortran2003.Execution_Part'>
  child type =  <class 'fparser.two.Fortran2003.Block_Nonlabel_Do_Construct'>
    child type =  <class 'fparser.two.Fortran2003.Nonlabel_Do_Stmt'>
      child type =  <class 'str'> 'DO'
      child type =  <class 'fparser.two.Fortran2003.Loop_Control'>
        child type =  <class 'NoneType'>
        child type =  <class 'tuple'>
        child type =  <class 'NoneType'>
    child type =  <class 'fparser.two.Fortran2003.Block_Nonlabel_Do_Construct'>
      child type =  <class 'fparser.two.Fortran2003.Nonlabel_Do_Stmt'>
        child type =  <class 'str'> 'DO'
        child type =  <class 'fparser.two.Fortran2003.Loop_Control'>
          child type =  <class 'NoneType'>
          child type =  <class 'tuple'>
          child type =  <class 'NoneType'>
      child type =  <class 'fparser.two.Fortran2003.Block_Nonlabel_Do_Construct'>
        child type =  <class 'fparser.two.Fortran2003.Nonlabel_Do_Stmt'>
          child type =  <class 'str'> 'DO'
```

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

PSyclone: Two Modes of Operation

Revolution

Process code written in a DSL

Currently two Domains supported:

- LFRic - Mixed finite elements, mesh unstructured in horizontal, structured in vertical, embedded in Fortran
- GOcean - DSL for 2D, finite difference, stretched, structured grid, embedded in Fortran

Evolution

Process existing code that follows strict coding conventions

Recognise certain code structures and construct higher-level Internal Representation

Transformations applied to this IR

In development for NEMO (plus associated models, e.g. SI3, MEDUSA). Also applied to ROMS.

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

Levels of Abstraction

Domain-specific: LFRic IR, NEMO IR, GOcean IR

DSLs

Language-independent: PSyIR

Not DSLs!

Language-specific: Fortran, C, … OpenMP, OpenACC, MPI, …

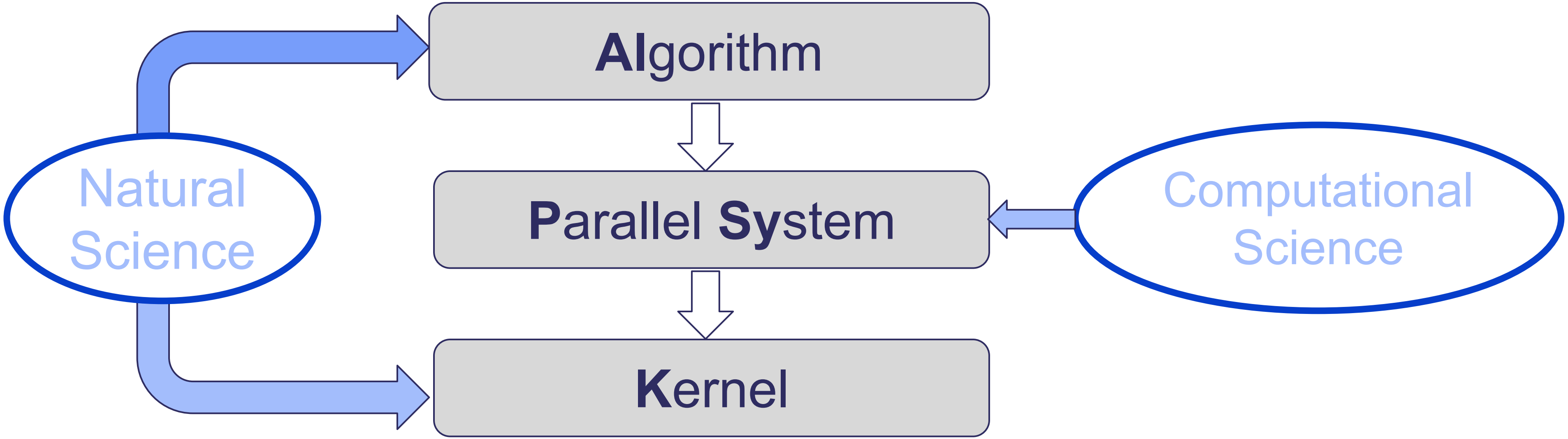
ESiWACE2 has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 823988

The LFRic Domain

(Revolution)

ESiWACE2 has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 823988

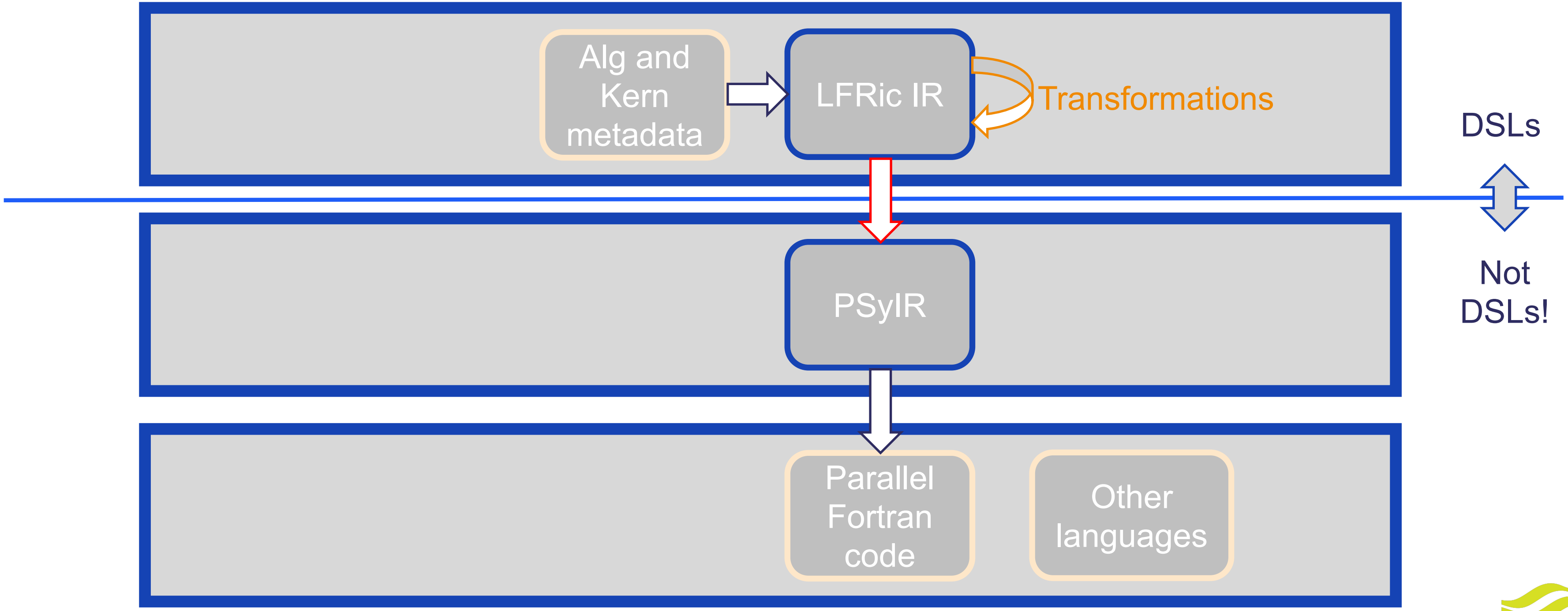
LFRic: Separation of Concerns



PSyKAl : Separate the Natural Science from the Computational Science (performance)

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

LFRic DSL PSy Layer



ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

LFRic DSL: Algorithm Layer Example

```
type(field_type) :: hb_inv
type(field_type), private :: grad_p
```

Logically-global field objects

```
call invoke( setval_c(grad_p, 0.0_r_def),                              &
             scaled_matrix_vector_kernel_type(grad_p, p, div_star,     &
                                    hb_inv),
             enforce_bc_kernel_type( grad_p ),
             apply_variable_hx_kernel_type(
                  Hp, grad_p, mt_lumped_inv, p,
                  compound_div, p3theta, ptheta2, m3_exner_star,
                  tau_t, timestep_term) )
```

Specify kernels to execute using an invoke()

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

LFRic DSL: Kernel Metadata Example

```
type, public, extends(kernel_type) :: apply_variable_hx_kernel_type
  private
  type(arg_type) :: meta_args(10) = (/                                   &
       arg_type(GH_FIELD,    GH_WRITE, W3),                              &
       arg_type(GH_FIELD,    GH_READ,  W2),                              &
       arg_type(GH_FIELD,    GH_READ,  ANY_SPACE_1),                     &
       arg_type(GH_FIELD,    GH_READ,  W3),                              &
       arg_type(GH_OPERATOR, GH_READ,  W3, W2),                          &
       arg_type(GH_OPERATOR, GH_READ,  W3, ANY_SPACE_1),                 &
       arg_type(GH_OPERATOR, GH_READ,  ANY_SPACE_1, W2),                 &
       arg_type(GH_OPERATOR, GH_READ,  W3, W3),                          &
       arg_type(GH_REAL,     GH_READ),                                   &
       arg_type(GH_REAL,     GH_READ)                                    &
       /)
  integer :: iterates_over = CELLS
contains
  procedure, nopass :: apply_variable_hx_code
end type
```

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

LFRic DSL: Vanilla PSy-layer Code

```
      DO df=1,undf_aspc1_grad_p
        grad_p_proxy%data(df) = 0.0_r_def
      END DO
      DO cell=1,grad_p_proxy%vspace%get_ncell()
        !
        CALL scaled_matrix_vector_code(nlayers, grad_p_proxy%data, p_proxy%data, div
_star_proxy%data, hb_inv_proxy%data, ndf_aspc1_grad_p, undf_aspc1_grad_p, map_aspc1_
grad_p(:,cell), ndf_aspc2_p, undf_aspc2_p, map_aspc2_p(:,cell), ndf_w3, undf_w3, map
_w3(:,cell))
      END DO
      DO cell=1,grad_p_proxy%vspace%get_ncell()
        !
        CALL enforce_bc_code(nlayers, grad_p_proxy%data, ndf_aspc1_grad_p, undf_aspc
1_grad_p, map_aspc1_grad_p(:,cell), boundary_dofs_grad_p)
      END DO
```

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

LFRic Transformation Example

(psyclone/examples/lfric/eg3)

Consider a simpler example where an invoke() contains a single, user-supplied kernel. Algorithm code:

```
type(field_type), intent(inout)    :: lhs
type(field_type), intent(in)       :: rhs
type(mesh_type),  intent(in)       :: mesh
type(field_type), intent(in)       :: chi(3)

integer(i_def),                 intent(in) :: solver_type
type(quadrature_type), optional, intent(in) :: qr
```

```
call invoke( w3_solver_kernel_type(lhs, rhs, chi, ascalar, qr) )
```

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

LFRic Transformation Example

Corresponding PSyIR representation:

```
InvokeSchedule[invoke='invoke_0_w3_solver_kernel_type', dm=False]
    0: Loop[type='', field_space='w3', it_space='cells', upper_bound='ncells']
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFINED>]
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFINED>]
        Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
        Schedule[]
            0: CodedKern solver_w3_code(lhs,rhs,chi,ascalar) [module_inline=False]
```

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

Transformation script:

```
def trans(psy):
    ''' PSyclone transformation script for the dynamo0p3 api to apply
    colouring and OpenMP generically.'''
    ctrans = Dynamo0p3ColourTrans()
    otrans = DynamoOMPParallelLoopTrans()

    # Loop over all of the Invokes in the PSy object
    for invoke in psy.invokes.invoke_list:

        schedule = invoke.schedule

        # Colour all of the loops over cells unless they are on
        # discontinuous spaces
        cschedule = schedule
        for child in schedule.children:
            if isinstance(child, Loop) \
               and child.field_space.orig_name \
               not in FunctionSpace.VALID_DISCONTINUOUS_NAMES \
               and child.iteration_space == "cells":
                cschedule, _ = ctrans.apply(child)
        # Then apply OpenMP to each of the colour loops
        schedule = cschedule
        for child in schedule.children:
            if isinstance(child, Loop):
                if child.loop_type == "colours":
                    schedule, _ = otrans.apply(child.loop_body[0])
                else:
                    schedule, _ = otrans.apply(child)
```

ESiWACE2 has received funding from t

LFRic Transformation Example

Transformed PSyIR representation:

```
InvokeSchedule[invoke='invoke_0_w3_solver_kernel_type', dm=False]
    0: Directive[OMP parallel do]
        Schedule[]
            0: Loop[type='', field_space='w3', it_space='cells', upper_bound='ncells']
                Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFINED>]
                Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFINED>]
                Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
                Schedule[]
                    0: CodedKern solver_w3_code(lhs,rhs,chi,ascalar) [module_inline=False]
```

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

LFRic Transformation Example

Generated Fortran (PSy layer):

```
      !
      !$omp parallel do default(shared), private(cell), schedule(static)
      DO cell=1,lhs_proxy%vspace%get_ncell()
        !
        CALL solver_w3_code(nlayers, lhs_proxy%data, rhs_proxy%data, chi_proxy(1)%dat
a, chi_proxy(2)%data, chi_proxy(3)%data, ascalar, ndf_w3, undf_w3, map_w3(:,cell), ba
sis_w3_qr, ndf_wchi, undf_wchi, map_wchi(:,cell), diff_basis_wchi_qr, np_xy_qr, np_z_
qr, weights_xy_qr, weights_z_qr)
      END DO
      !$omp end parallel do
      !
```

Transformed Algorithm code:

```
CALL invoke_0_w3_solver_kernel_type(lhs, rhs, chi, ascalar, qr)
```

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

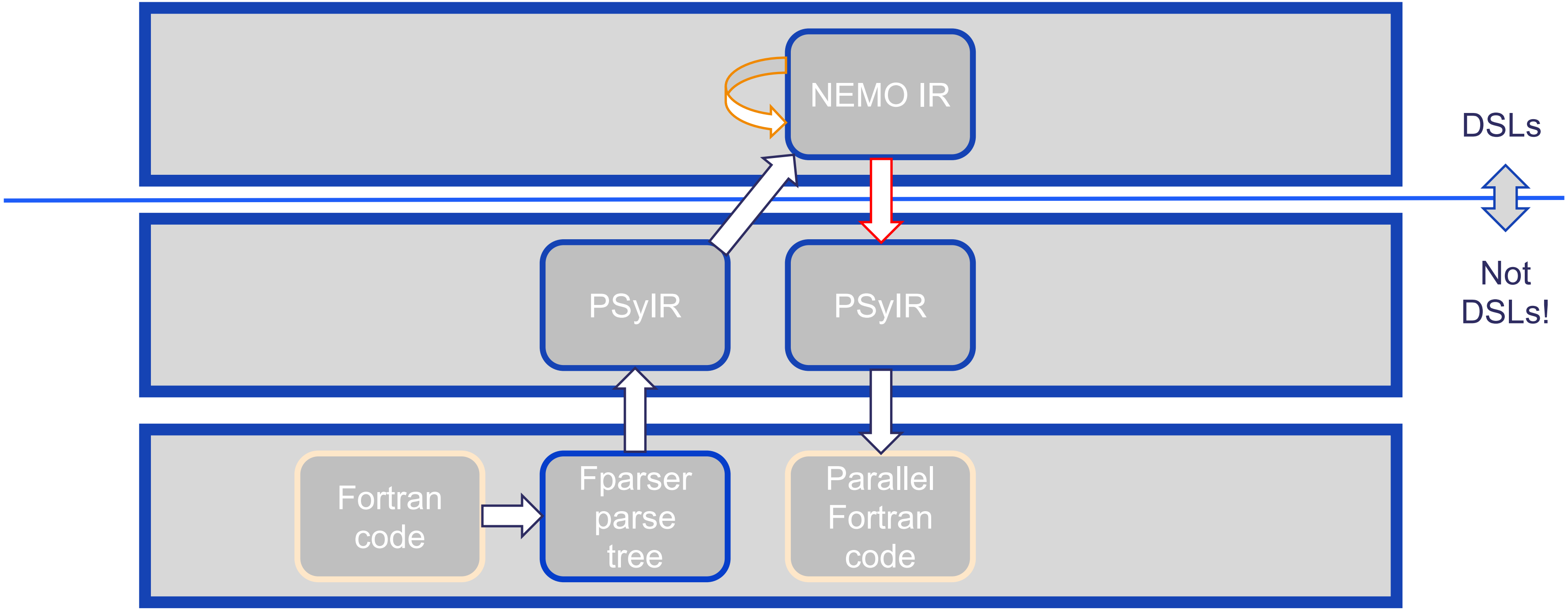
The NEMO Domain

(Evolution)

ESiWACE2 has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 823988

NEMO DSL

Construct high-level representation of existing source code:



ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

NEMO Transformation Example

(psyclone/examples/nemo/eg2)

Original code (tral_ldf_iso routine):

```
!                                                            ! ===========
DO jn = 1, kjpt                                              ! tracer loop
   !                                                         ! ===========
   !
   !!----------------------------------------------------------------------
   !!   I - masked horizontal derivative
   !!----------------------------------------------------------------------
: bug.... why (x,:,:)?   (1,jpj,:) and (jpi,1,:) should be sufficient....
   zdit (1,:,:) = 0._wp     ;     zdit (jpi,:,:) = 0._wp
   zdjt (1,:,:) = 0._wp     ;     zdjt (jpi,:,:) = 0._wp
   !!end

   ! Horizontal tracer gradient
   DO jk = 1, jpkm1
      DO jj = 1, jpjm1
         DO ji = 1, jpim1   ! vector opt.
            zdit(ji,jj,jk) = ( ptb(ji+1,jj  ,jk,jn) - ptb(ji,jj,jk,jn) ) * umask(ji,jj,jk)
            zdjt(ji,jj,jk) = ( ptb(ji  ,jj+1,jk,jn) - ptb(ji,jj,jk,jn) ) * vmask(ji,jj,jk)
         END DO
      END DO
   END DO
   IF( ln_zps ) THEN      ! botton and surface ocean correction of the horizontal gradient
      DO jj = 1, jpjm1                ! bottom correction (partial bottom cell)
```

PSyIR constructed by PSyclone:

```
4: Loop[type='levels', field_space='None', it_space='None']
    Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
    Reference[name:'jpkm1']
    Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
    Schedule[]
        0: Loop[type='lat', field_space='None', it_space='None']
            Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
            Reference[name:'jpjm1']
            Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
            Schedule[]
                0: Loop[type='lon', field_space='None', it_space='None']
                    Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
                    Reference[name:'fs_jpim1']
                    Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
                    Schedule[]
                        0: InlinedKern[]
                            Schedule[]
                                0: Assignment[]
                                    ArrayReference[name:'zdit']
                                        Reference[name:'ji']
                                        Reference[name:'jj']
                                        Reference[name:'jk']
                                    BinaryOperation[operator:'MUL']
```

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

NEMO Transformation Script

```
def trans(psy):
    ''' Transform a specific Schedule by making all loops
    over levels OpenMP parallel.

    :param psy: the object holding all information on the PSy layer \
                to be modified.
    :type psy: :py:class:`psyclone.psyGen.PSy`

    :returns: the transformed PSy object
    :rtype:  :py:class:`psyclone.psyGen.PSy`

    '''
    from psyclone.psyGen import TransInfo
    from psyclone.nemo import NemoKern
    # Get the Schedule of the target routine
    sched = psy.invokes.get('tra_ldf_iso').schedule
    # Get the transformation we will apply
    ompt = TransInfo().get_trans_name('OMPParallelLoopTrans')
    # Apply it to each loop over levels containing a kern
    for loop in sched.loops():
        kernels = loop.walk(NemoKern)
        if kernels and loop.loop_type == "levels":
            sched, _ = ompt.apply(loop)
    # Return the modified psy object
    return psy
```

Parallelises all loops over vertical levels using OpenMP

mme under grant agreement No 823988

Transformed PSyIR:

```
Literal[value: '0.', Scalar<REAL, wp: <Scalar<INTEGER, UNDEFINED>, Unresolved>>]
4: Directive[OMP parallel do]
    Schedule[]
        0: Loop[type='levels', field_space='None', it_space='None']
            Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
            Reference[name:'jpkm1']
            Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
            Schedule[]
                0: Loop[type='lat', field_space='None', it_space='None']
                    Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
                    Reference[name:'jpjm1']
                    Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
                    Schedule[]
                        0: Loop[type='lon', field_space='None', it_space='None']
                            Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
                            Reference[name:'fs_jpim1']
                            Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
                            Schedule[]
                                0: InlinedKern[]
                                    Schedule[]
                                        0: Assignment[]
```

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

Generated Fortran with OpenMP directives added

```
DO jn = 1, kjpt
  zdit(1, :, :) = 0._wp
  zdit(jpi, :, :) = 0._wp
  zdjt(1, :, :) = 0._wp
  zdjt(jpi, :, :) = 0._wp
  !$OMP parallel do default(shared), private(ji,jj,jk), schedule(static)
  DO jk = 1, jpkm1
    DO jj = 1, jpjm1
      DO ji = 1, fs_jpim1
         zdit(ji, jj, jk) = (ptb(ji + 1, jj, jk, jn) - ptb(ji, jj, jk, jn)) &
                  * umask(ji, jj, jk)
         zdjt(ji, jj, jk) = (ptb(ji, jj + 1, jk, jn) - ptb(ji, jj, jk, jn)) &
                  * vmask(ji, jj, jk)
      END DO
    END DO
  END DO
  !$OMP end parallel do
```

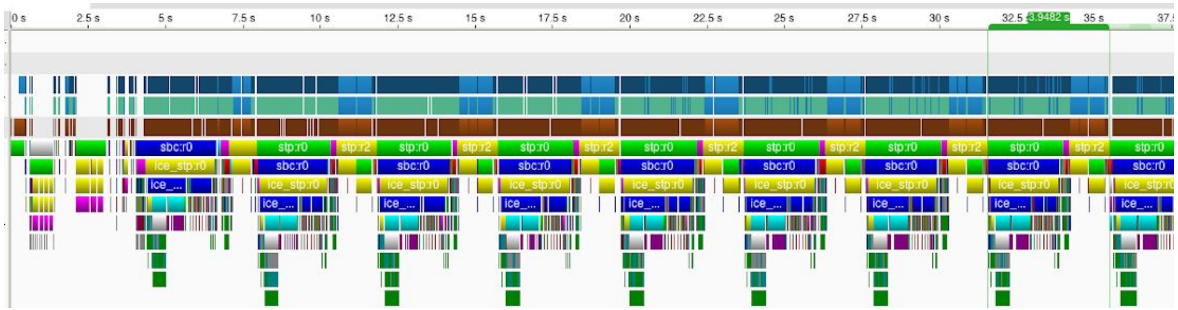
ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

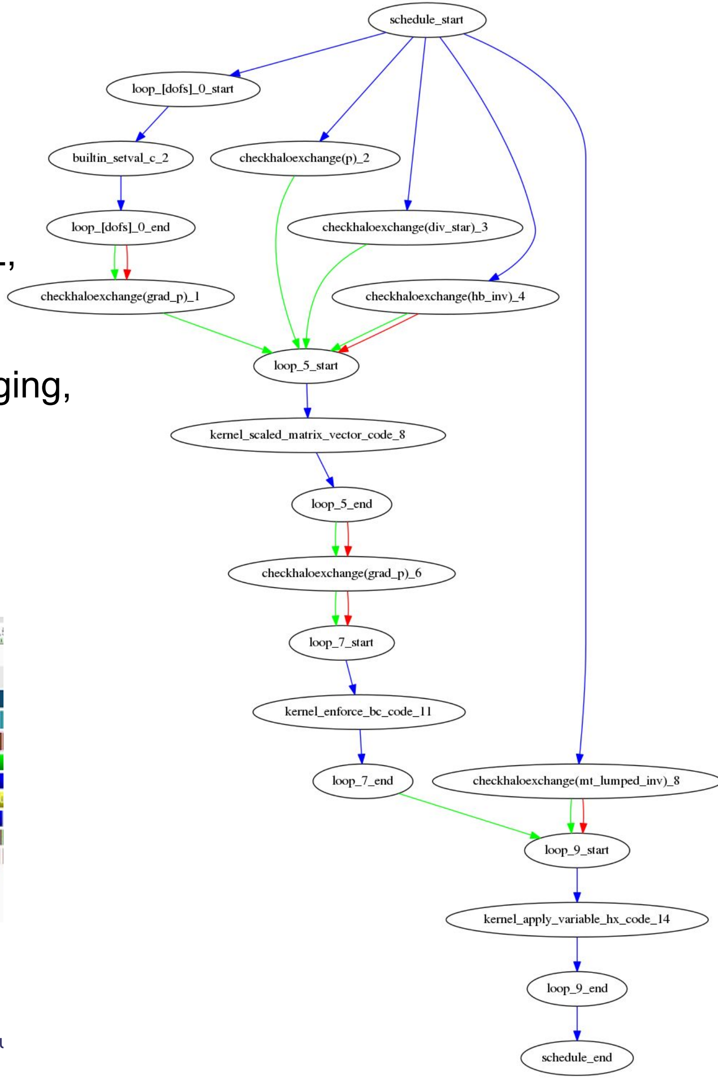
Other Features

Available transformations (loop fusion, OpenMP, OpenACC, OpenCL, asynchronous halo exchanges, redundant computation)

PSyData API - allows calipers to be inserted for e.g. profiling, debugging, validation, kernel (benchmark) extraction, on-line visualisation etc.

DAG view of PSy-layer Schedules





ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme u

Summary

- PSyclone is a Domain-Specific Compiler for use with both DSLs and existing code
- Intended as a tool for use by an HPC expert
- Initially developed in support of the MO LFRic Model (revolution)
- Extended to tackle existing finite difference code (evolution)
- Constructs a PSyclone Internal Representation of supplied code
- User transforms this representation using Python scripts
- Generates Fortran (or OpenCL) for the transformed PSyIR

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

Science and Technology Facilities Council

Thank you

User, Developer and Reference Guides are available:

psyclone[-dev,-ref].readthedocs.io

For more information please contact:

rupert.ford@stfc.ac.uk
andrew.porter@stfc.ac.uk

PSyclone

Extras

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988