Hartree Centre
Science & Technology Facilities Council

PSyclone

Rupert Ford, Andrew Porter
Iva Kavcic
Joerg Henrichs
[Matthew Hambley, Peter Vitt]
+ many from LFRic

5th ENES HPC Workshop 17th – 18th May 2018

PSyclone

Hartree Centre
Science & Technology Facilities Council

Motivation

- Maintainable high performance software
  - Single source science code
  - Performance portability

cf. Paul Messina ECP yesterday

- Complex parallel code + Complex parallel architectures + Complex compilers = Complex optimisation space => no single solution
- "Single source optimised code is not attainable"

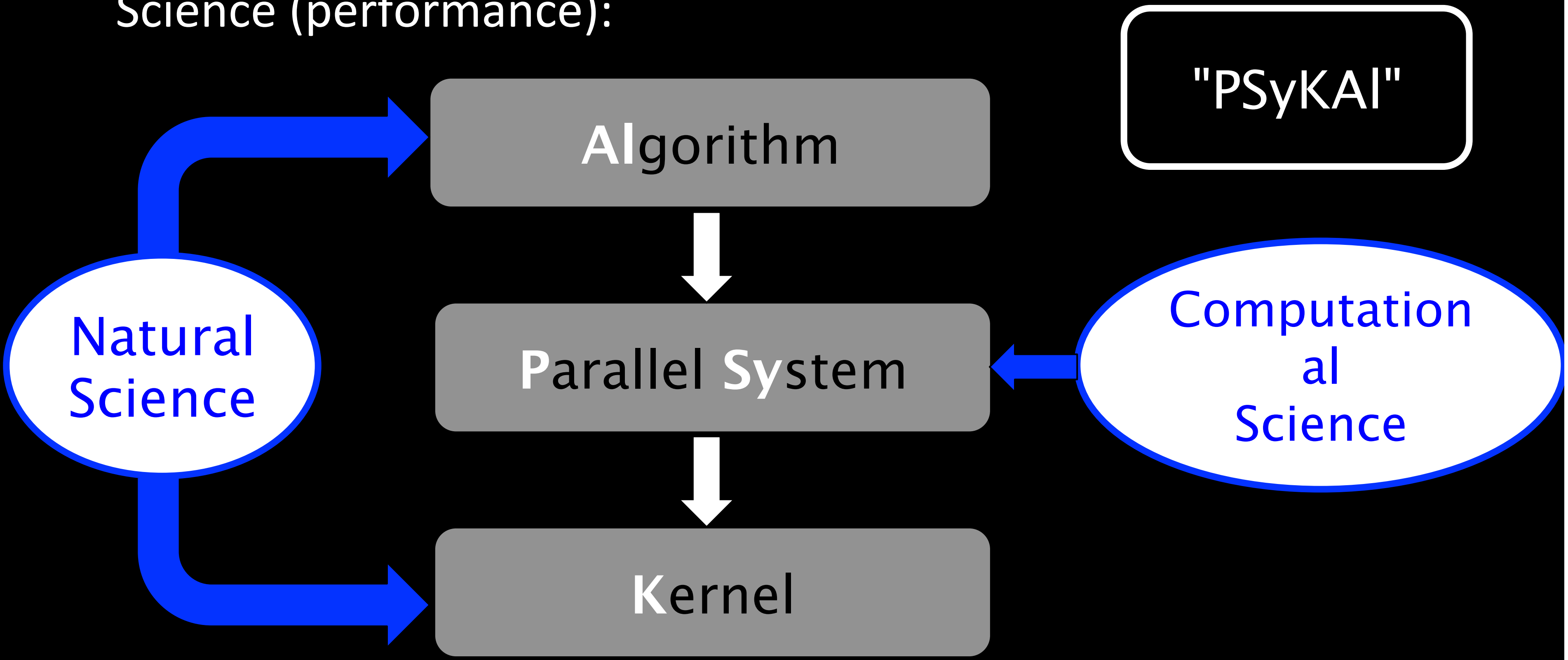
Maintenance issue with directives – Thomas Schulthess and Hisashi Yashiro yesterday

- So …
- Separate science specification/code from code optimisation

Hartree Centre
Science & Technology Facilities Council

Separation of Concerns

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



Hartree Centre
Science & Technology Facilities Council

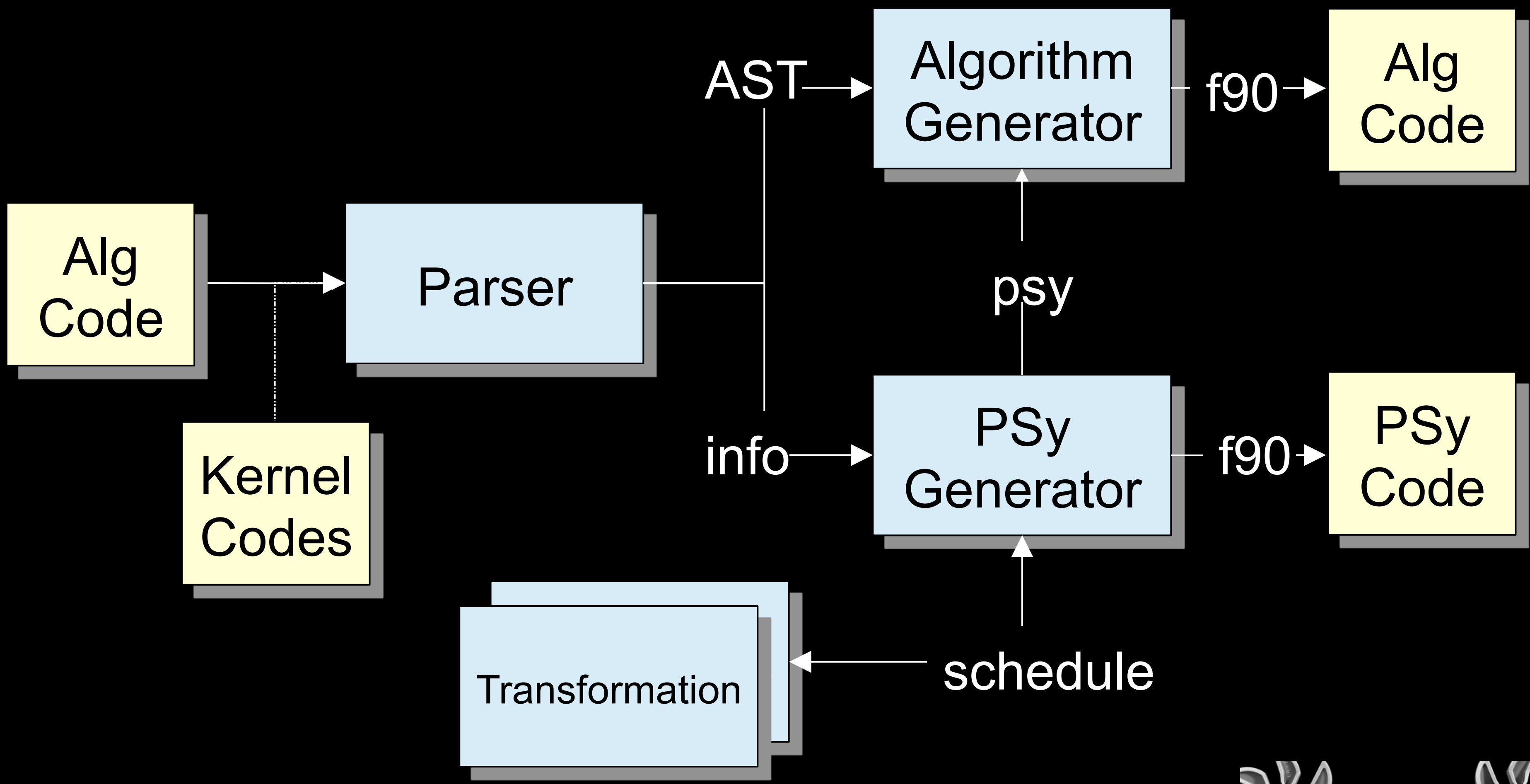
Generating the PSy Layer

PSyclone

- A domain-specific compiler for embedded DSL(s)
  - Finite Difference/Volume, Finite Element
  - Currently Fortran -> Fortran
  - Supports distributed- and shared-memory parallelism
- Should reduce programmer errors (both correctness and optimisation) – but more evidence needed
- A tool for use by HPC experts
  - Hard to beat a human (debatable)
  - Optimisations encoded as a ‘recipe’ rather than baked into the scientific source code
  - Different recipes for different architectures

Hartree Centre
Science & Technology Facilities Council

PSyclone Architecture

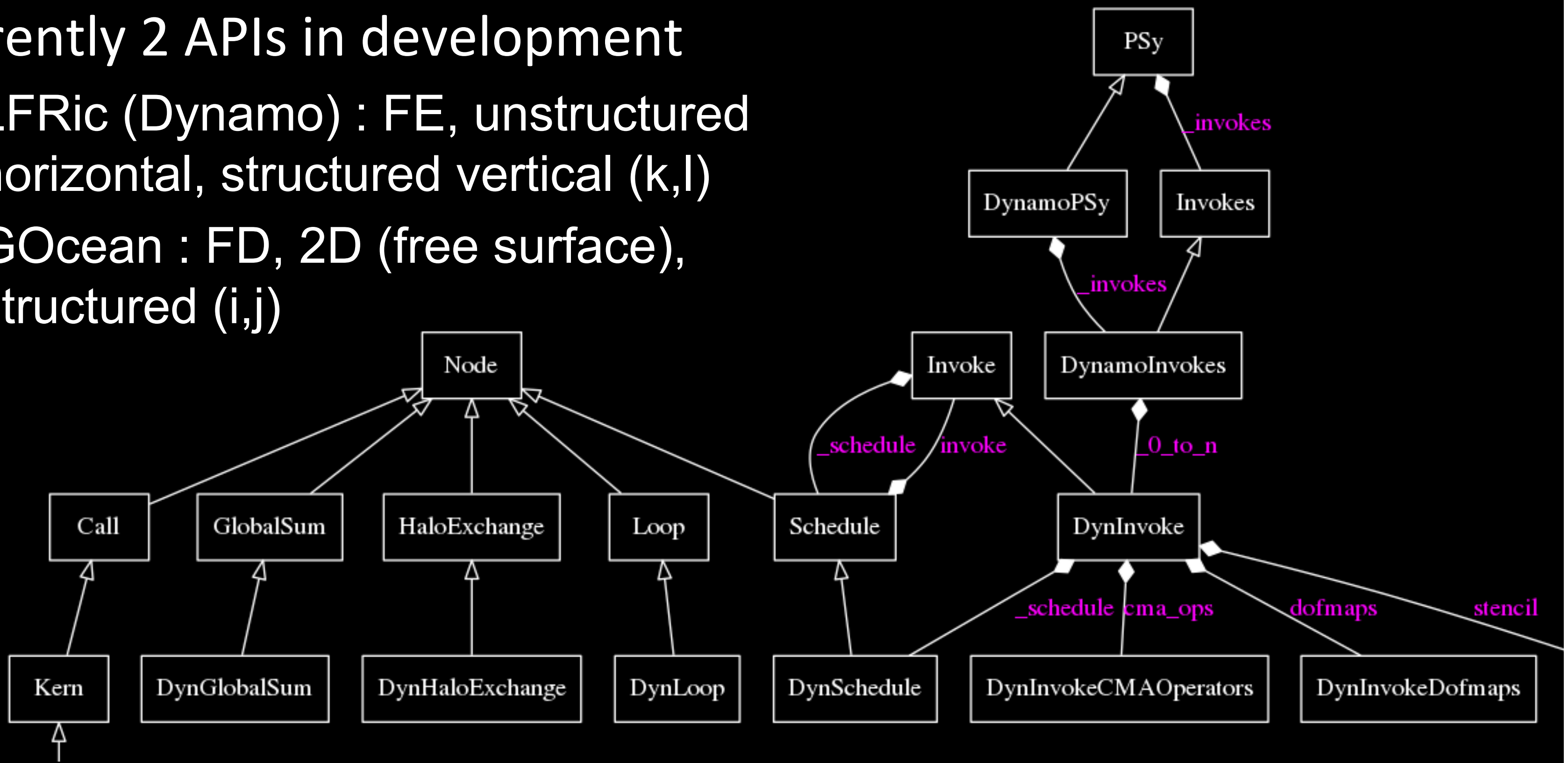


PSyclone

Hartree Centre
Science & Technology Facilities Council

PSyclone APIs

- Generic base classes
- Overide for a particular API
- PSyclone API + API Infrastructure == DSL
- Currently 2 APIs in development
  - LFRic (Dynamo) : FE, unstructured horizontal, structured vertical (k,l)
  - GOcean : FD, 2D (free surface), structured (i,j)



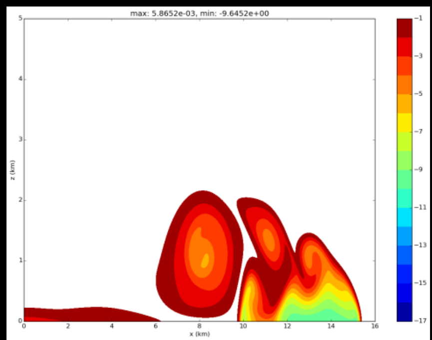
Hartree Centre
Science & Technology Facilities Council

Users

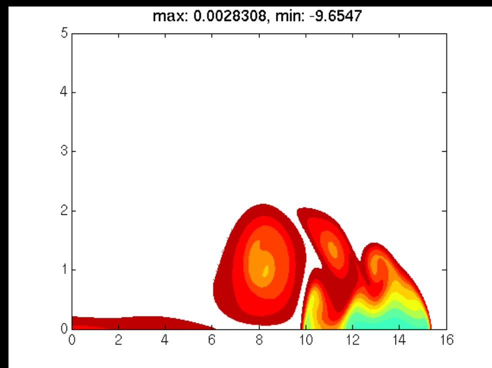
- Met Office LFRic Model
  - Prototype next generation atmosphere model
  - PSyclone integrated into LFRic build system in September 2015
  - Went (MPI + OpenMP) parallel in March 2016. No change to science code.
  - Used for new science since then, including Physics (but there are overrides)

Cold air bubble on Cartesian domain

GungHo



ENDGame



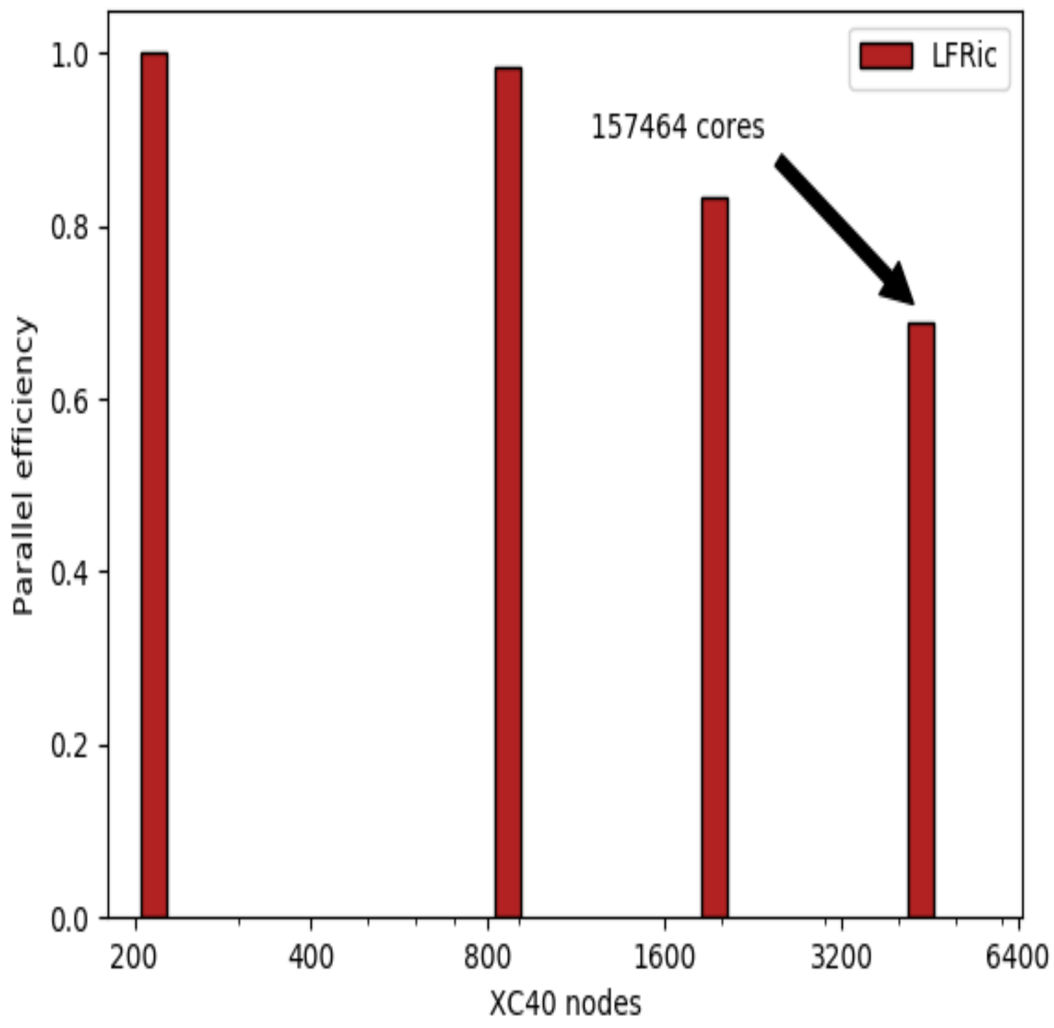
Strong scaling

Strong Scaling.
C1944 (1994*1944*6)
30 levels ~ 5km resolution
Global model (orography)
Baroclinic wave
Intel17 compiler
Cray XC40 dual 18-core socket Broadwell Xeon
2 MPI ranks per socket
9 OpenMP threads per rank
Parallel efficiency scaled

Courtesy of Chris Maynard

Met Office

Dromedary 2018 LFRic parallel efficiency versus node count

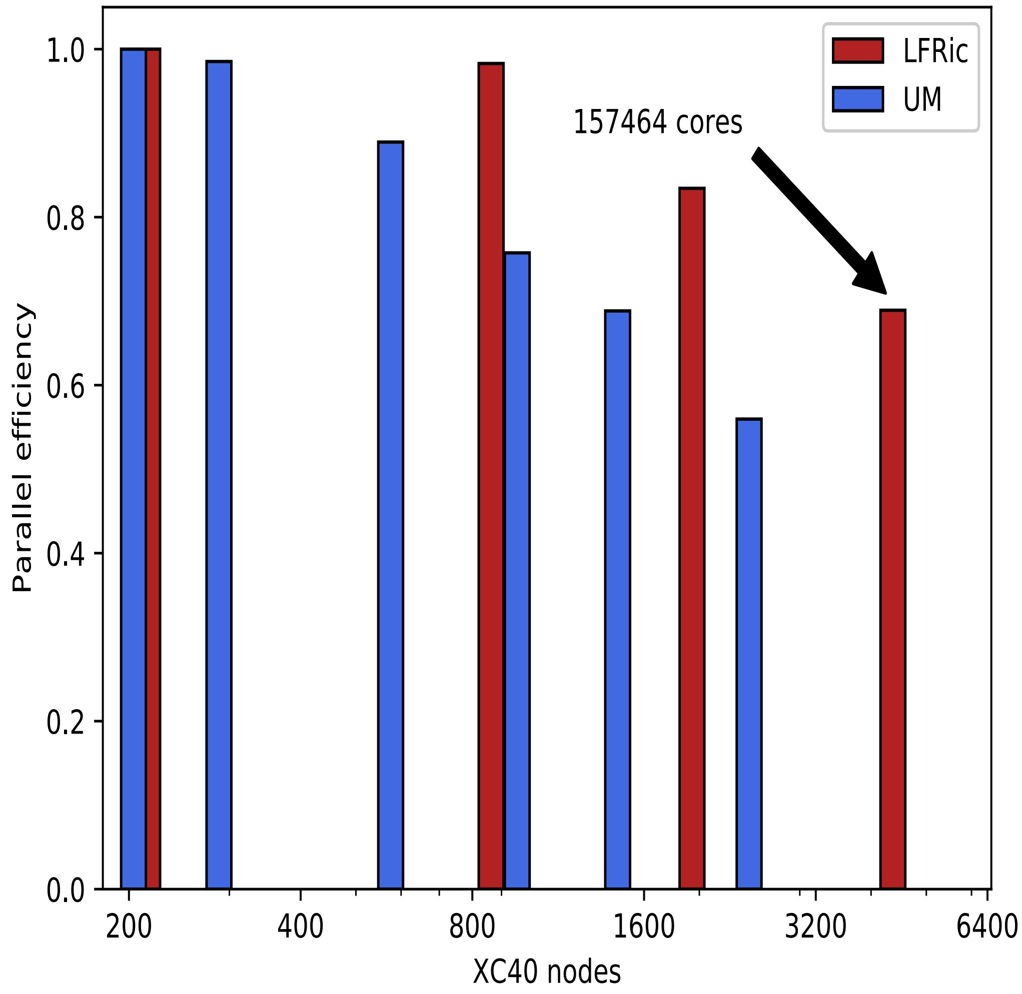


Comparison with UM

Hartree Centre
Science & Technology Facilities Council

Indirect comparison
Global N2048 , UM10.4
70 levels ~ 6.5km resolution
Intel17 compiler
Cray XC40 dual 18-core socket Broadwell Xeon
6 MPI ranks per socket
3 OpenMP threads per rank
Parallel efficiency scaled

Dromedary 2018 LFRic versus 10.4 (Q4 2016 UM)



Courtesy of Chris Maynard

Met Office

Hartree Centre
Science & Technology Facilities Council

Users

- Under evaluation by BOM for their in-house Tsunami code (MOST)
  - Implementation by Joerg Henrichs (UM Partners collab)
  - Using GOcean API
  - "Once the first kernel was done, the rest was very quick"
  - "Surprisingly, first version working within 15 minutes" (OpenMP parallelisation)
  - "Subjective: PSyclone code is easier to understand"
  - "It can meet hand-tuned performance"
  - 24 threads : OpenMP PSyclone Intel 31.0s, PSyclone Cray 32.7s Hand-optimised Intel C 35.3s

Australian Government
Bureau of Meteorology

Hartree Centre
Science & Technology Facilities Council

Algorithm Example

- Taken from LFRic subroutine apply_helmholtz_lhs
- Logically global (whole field operations)
- Types not arrays (no access to data)
- Supports multiple invoke's in a subroutine
- Multiple kernels in an invoke (as many as possible)
- Mixed builtin's and coded kernels

```
! For building rhs
type(field_type), private :: r_u, r_p, u_term, &
                             flux, theta_adv_term, m2_u
! For applying lhs
type(field_type), private :: grad_p, l_p
```

```
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) )
```

Hartree Centre
Science & Technology Facilities Council

Kernel Code Example

- Taken from LFRic subroutine apply_variable_hx_kernel
- Single column
- Fortran arrays (direct access to data)
- Maps the values of a field in one function space to a field in another function space

```fortran
! Compute Pt2 * u
do k = 0, nlayers-1
  do df = 1, ndf_w2
    x_e(df) = x(map_w2(df)+k)
  end do
  ik = (cell-1)*nlayers + k + 1

  t_e = matmul(pt2(:,:,ik),x_e)
  do df = 1,ndf_wt
    t(map_wt(df)+k) = t(map_wt(df)+k) + t_e(df)
  end do
end do
```

Kernel Metadata Example

- Taken from LFRic subroutine apply_variable_hx_kernel
- Information read by PSyclone
- Description of argument types, access, function spaces, assumed iteration space and code subroutine name

```fortran
type, public, extends(kernel_type) :: apply_variable_hx_kernel_type
  private
  type(arg_type) :: meta_args(9) = (/                                   &
       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)                                   &
       /)
  integer :: iterates_over = CELLS
contains
  procedure, nopass ::apply_variable_hx_code
end type
```

Transformation Example

- Same example as before
- Computationally costly
- Slightly modified (replaced operator).
- Apply redundant computation to remove halo exchanges and reorder code (using PSyclone 1.6.0)

```
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) )
```

Hartree Centre
Science & Technology Facilities Council

Transformation Example

PSyclone's internal representation.
A schedule.

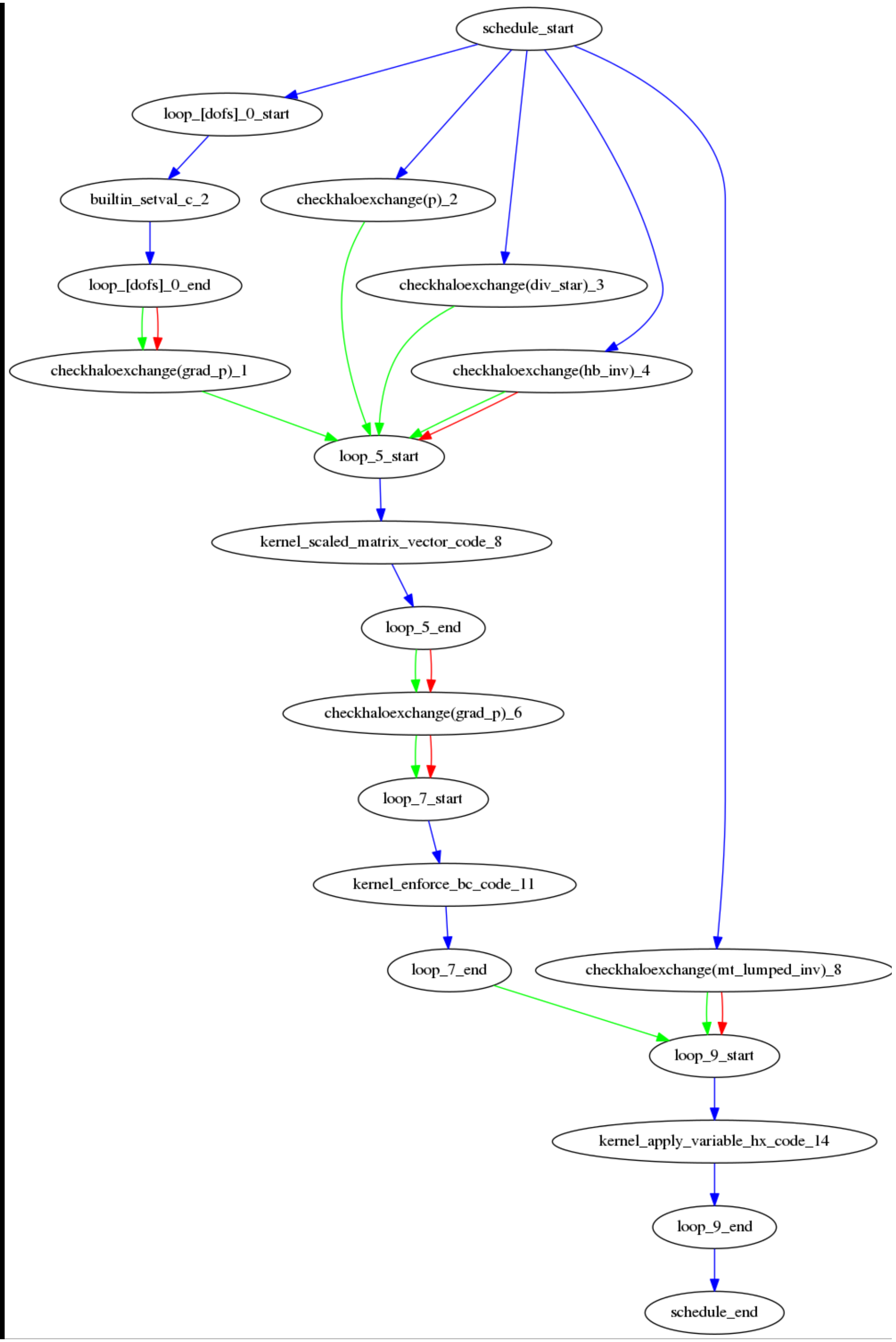
```
Schedule[invoke='invoke_0' dm=True]
    Loop[type='dofs',field_space='any_space_1',it_space='dofs', upper_bound='ndofs']
        Call setval_c(grad_p,0.0_r_def)
    HaloExchange[field='grad_p', type='region', depth=1, check_dirty=False]
    HaloExchange[field='p', type='region', depth=1, check_dirty=True]
    HaloExchange[field='div_star', type='region', depth=1, check_dirty=True]
    HaloExchange[field='hb_inv', type='region', depth=1, check_dirty=True]
    Loop[type='',field_space='any_space_1',it_space='cells', upper_bound='cell_halo(1)']
        KernCall scaled_matrix_vector_code(grad_p,p,div_star,hb_inv) [module_inline=False]
    HaloExchange[field='grad_p', type='region', depth=1, check_dirty=False]
    Loop[type='',field_space='any_space_1',it_space='cells', upper_bound='cell_halo(1)']
        KernCall enforce_bc_code(grad_p) [module_inline=False]
    HaloExchange[field='mt_lumped_inv', type='region', depth=1, check_dirty=True]
    Loop[type='',field_space='w3',it_space='cells', upper_bound='ncells']
        KernCall apply_variable_hx_code(hp,grad_p,mt_lumped_inv,p,compound_div,p3theta,pthe
ta2,m3_exner_star,tau_t,timestep_term) [module_inline=False]
```

Hartree Centre
Science & Technology Facilities Council

Transformation Example

PSyclone's dependency view.
A task graph (DAG).

Natural fit with task-based approaches as advocated by Jesus



Transformation Example

Hartree Centre
Science & Technology Facilities Council

Example-specific redundant computation script.

```
> psyclone -oalg a.f90 -opsy p.f90 -s ./script.py alg.f90
```

```python
def trans(psy):
    '''removes the grad_p halo exchanges by redundant computation then
    moves the remaining halo exchanges to the beginning of the invoke
    call'''
    from psyclone.transformations import Dynamo0p3RedundantComputationTrans, \
        MoveTrans
    rc_trans = Dynamo0p3RedundantComputationTrans()
    m_trans = MoveTrans()
    invoke = psy.invokes.invoke_list[0]
    schedule = invoke.schedule

    schedule.view()
    schedule.dag("dag1", file_format="png")

    # redundant computation to remove grad_p halo exchanges
    schedule, _ = rc_trans.apply(schedule.children[5], depth=2)
    schedule, _ = rc_trans.apply(schedule.children[0], depth=2)

    schedule.view()
    schedule.dag("dag2", file_format="png")

    # move remaining (potential) halo exchanges to start of the invoke
    schedule, _ = m_trans.apply(schedule.children[0], schedule.children[4])
    schedule, _ = m_trans.apply(schedule.children[6], schedule.children[0])

    schedule.view()
    schedule.dag("dag3", file_format="png")

    return psy
```

Hartree Centre
Science & Technology Facilities Council

Transformation Example

Schedule after redundant computation applied. grad_p halo exchanges no longer required but some other halo exchanges require greater depth

```
Schedule[invoke='invoke_0' dm=True]
    Loop[type='dofs',field_space='any_space_1',it_space='dofs', upper_bound='dof_halo(2)']
        Call setval_c(grad_p,0.0_r_def)
    HaloExchange[field='p', type='region', depth=2, check_dirty=True]
    HaloExchange[field='div_star', type='region', depth=2, check_dirty=True]
    HaloExchange[field='hb_inv', type='region', depth=2, check_dirty=True]
    Loop[type='',field_space='any_space_1',it_space='cells', upper_bound='cell_halo(2)']
        KernCall scaled_matrix_vector_code(grad_p,p,div_star,hb_inv) [module_inline=False]
    Loop[type='',field_space='any_space_1',it_space='cells', upper_bound='cell_halo(1)']
        KernCall enforce_bc_code(grad_p) [module_inline=False]
    HaloExchange[field='mt_lumped_inv', type='region', depth=1, check_dirty=True]
    Loop[type='',field_space='w3',it_space='cells', upper_bound='ncells']
        KernCall apply_variable_hx_code(hp,grad_p,mt_lumped_inv,p,compound_div,p3theta,pthe
ta2,m3_exner_star,tau_t,timestep_term) [module_inline=False]
```

Transformation Example

Schedule after applying redundant computation and re-ordering of halo exchanges.
First halo exchanges then computation.

```
Schedule[invoke='invoke_0' dm=True]
    HaloExchange[field='mt_lumped_inv', type='region', depth=1, check_dirty=True]
    HaloExchange[field='p', type='region', depth=2, check_dirty=True]
    HaloExchange[field='div_star', type='region', depth=2, check_dirty=True]
    HaloExchange[field='hb_inv', type='region', depth=2, check_dirty=True]
    Loop[type='dofs',field_space='any_space_1',it_space='dofs', upper_bound='dof_halo(2)']
        Call setval_c(grad_p,0.0_r_def)
    Loop[type='',field_space='any_space_1',it_space='cells', upper_bound='cell_halo(2)']
        KernCall scaled_matrix_vector_code(grad_p,p,div_star,hb_inv) [module_inline=False]
    Loop[type='',field_space='any_space_1',it_space='cells', upper_bound='cell_halo(1)']
        KernCall enforce_bc_code(grad_p) [module_inline=False]
    Loop[type='',field_space='w3',it_space='cells', upper_bound='ncells']
        KernCall apply_variable_hx_code(hp,grad_p,mt_lumped_inv,p,compound_div,p3theta,ptheta2,m3_exner_star,
tau_t,timestep_term) [module_inline=False]
```

Hartree Centre
Science & Technology Facilities Council

Ongoing work

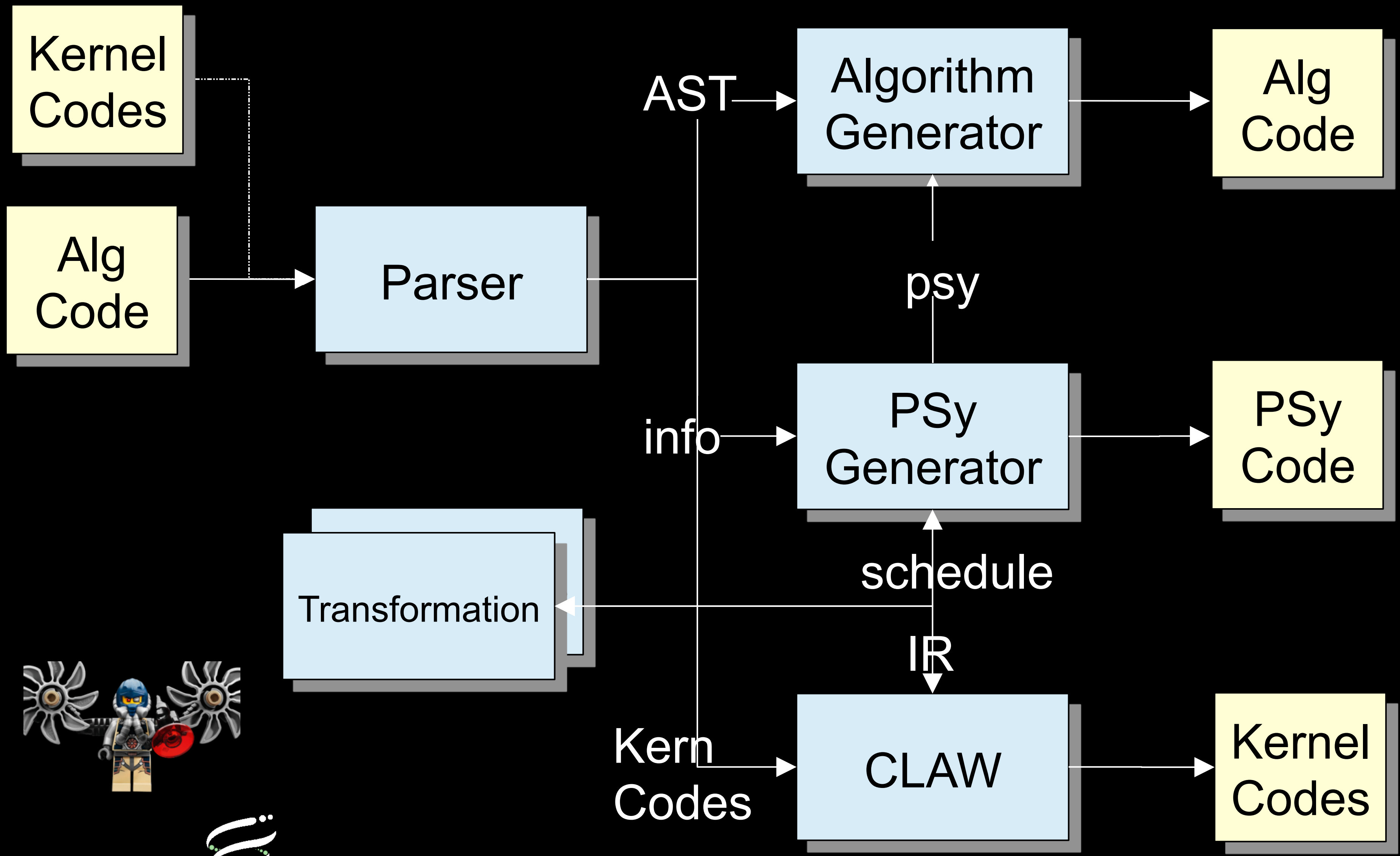
- Support for FPGA's via EuroExa Project
- Kernel transformations (integrate CLAW)
- LFRic-api extensions
- LFRic optimisations (e.g. asynchronous halo exchange, GPU directives)
- NEMO api
- Search the optimisation space

EuroExa

- Building an Exascale demonstrator
- ARM processors + FPGA's
- NEMO benchmark
- Use PSyclone to generate code
- OpenCL, OmpSs, …
- [Manchester LFRic benchmark – use PSyclone?]
- [ECMWF Physics, collaborate on CLAW]

Hartree Centre
Science & Technology Facilities Council

CLAW in PSyclone



PSyclone

Hartree Centre
Science & Technology Facilities Council

Thanks for listening

PSyclone 1.6.0

BSD 3-clause

https://github.com/stfc/PSyclone

https://psyclone.readthedocs.io

> sudo pip install psyclone

fparser 0.0.7

BSD 3-clause

https://github.com/stfc/fparser

https://fparser.readthedocs.io

> sudo pip install fparser

EURO EXA

Met Office

intel
Intel® Parallel Computing Center

is-enes
INFRASTRUCTURE FOR THE EUROPEAN NETWORK FOR EARTH SYSTEM MODELLING

Hartree Centre
Science & Technology Facilities Council