Science & Technology Facilities Council

PSyKAl: Code Generation for Performance Portability

Mike Ashworth, Rupert Ford, Andrew Porter
STFC Scientific Computing Department
STFC Daresbury Laboratory

Chris Maynard, Tom Melvin
The Met Office

United Kingdom

This work was funded by the STFC Hartree Centre and by the NERC

GungHo Project

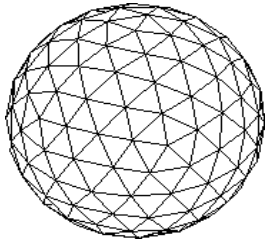
Globally
Uniform
Next
Generation
Highly
Optimized

工合

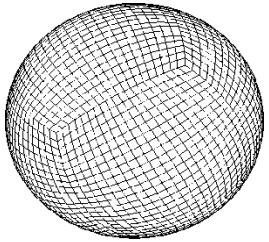
"Working together harmoniously"

Targets a brand new dynamical core for weather forecasting and climate research

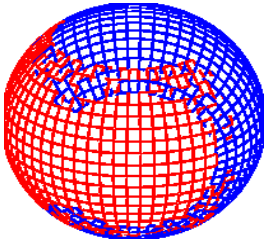
- Scalability – choose a globally uniform grid which has no poles
- Speed – maintain performance at high & low resolution and for high & low core counts
- Accuracy – need to maintain standing of the model
- Space weather implies a 600km deep model
- Five year project 2011-2015
- Operational weather forecasts beyond 2020



Triangles



Cube-sphere



Yin-Yang

Met Office

NATURAL ENVIRONMENT RESEARCH COUNCIL

Science & Technology Facilities Council

工合海洋 GOcean

- NERC-funded Technology Proof of Concept fund
- 12-months funding 03/2014 – 02/2015
- Collaboration between National Oceanography Centre, Liverpool (NOC) and STFC
- Investigate the feasibility of applying technology from the GungHo project to ocean modelling
- Extend the developing GungHo infrastructure to support finite difference on regular, lat-lon grids

Science & Technology Facilities Council

Industry Collaborations

- IBM / Hartree Centre
  - HC to continue development of PSyclone after GungHo
  - GungHo + IBM → stencil mapping, Gocean
  - IBM collaboration → OpenPower + NVIDIA
  - Expand collaboration → data analytics
- Intel PCC
  - Year 2
  - 0.5 FTE UM performance on Xeon Phi (radiation)
  - 0.5 FTE on dynamo optimisation for Xeon Phi
- NVIDIA collaboration
  - Optimisation effort
    (GOcean moving into GungHo)

Hartree Centre
Science & Technology Facilities Council

IBM

intel
Intel Parallel Computing Centers

NVIDIA

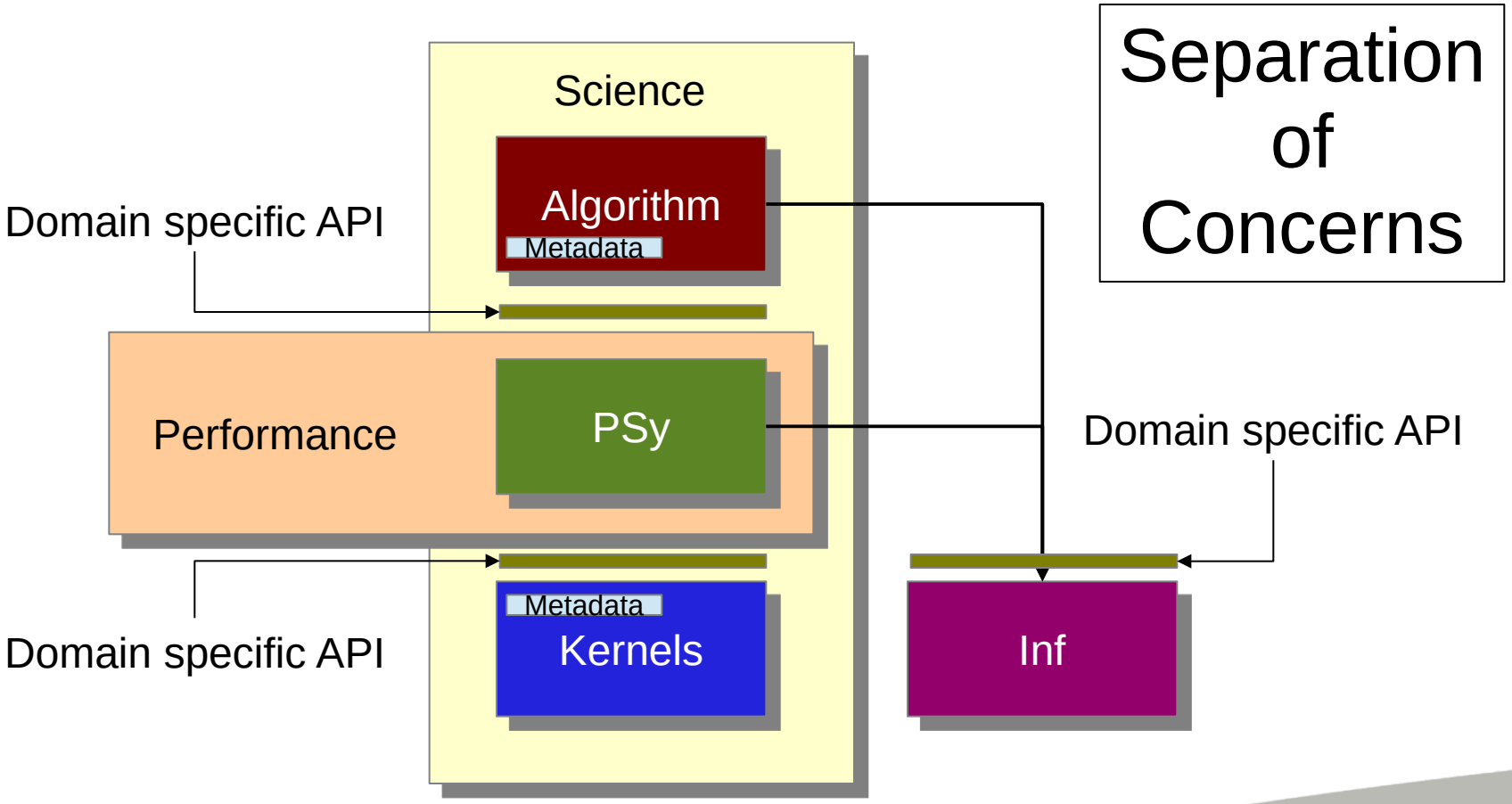
Science & Technology Facilities Council

Motivation

Proposition:

- It is impossible to produce a single source code which performs optimally across a range of modern parallel computer architectures (processors, compilers etc.)
  - Multiple versioning e.g. using #ifdef is unmaintainable
  - This is unlikely to improve in next 5-10 years

What is PSyKAl?



Science & Technology Facilities Council

What is PSyclone?

- Configurable domain specific compiler for domain specific languages (DSLs) using the PSyKAl approach
  - PSyKAl developed in the GungHo project
  - Configurable to related, defined APIs – implementations for unstructured FE (dynamo APIs) and structured FD (GOcean APIs)
  - Designed from the outset for performance

Science & Technology Facilities Council

Difficult to continue as we are

Science & Technology Facilities Council

Don't move too far

'I'm looking for something that's more than irrational but less than psychotic.'

Science & Technology Facilities Council

DSL in Fortran

- Coded Algorithm layer written in Fortran
- Coded Kernels written in Fortran
- Generated pointwise routines
- PSy layer generated in Fortran

Science & Technology Facilities Council

Algorithm code example

```
...
call invoke(                                                    &
            continuity(ssha_t, sshn_t, sshn_u, sshn_v,          &
                       hu, hv, un, vn, rdt),                    &
            momentum_u(ua, un, vn, hu, hv, ht,                  &
                       ssha_u, sshn_t, sshn_u, sshn_v),         &
            ...
            copy(sshn_t, ssha_t),                               &
            next_sshu(sshn_u, sshn_t),                          &
            next_sshv(sshn_v, sshn_t)                           &
           )
...
```

```
 ...
 USE psy_gocean2d, ONLY: invoke_0
 ...
 CALL invoke_0(ssha_t, sshn_t, sshn_u, sshn_v, hu, hv, un, vn,
rdt, ua, ht, ssha_u, va, ssha_v, istp)
 ...
```

Generated vanilla PSy code example (gocean1.0 API)

```
  MODULE psy_gocean2d
    ...
    SUBROUTINE invoke_0(ssha_t, sshn_t, sshn_u, sshn_v, hu, hv,
un, vn, rdt, ua, ht, ssha_u, va, ssha_v, istp)
      ...
      DO j=ssha_t%internal%ystart,ssha_t%internal%ystop
        DO i=ssha_t%internal%xstart,ssha_t%internal%xstop
          CALL continuity_code(i, j, ssha_t%data, sshn_t%data,
sshn_u%data, sshn_v%data, hu%data, hv%data, un%data, vn%data,
rdt, sshn_t%grid%area_t)
        END DO
      END DO
      ...
    END SUBROUTINE invoke_0
  END MODULE psy_gocean2d
```

Science & Technology Facilities Council

Kernel code example (gocean1.0 API)

```
module continuity_mod
  ...
  type, extends(kernel_type) :: continuity
     type(arg), dimension(10) :: meta_args =         &
          (/ arg(WRITE, CT,       POINTWISE),        & ! ssha
             ...
             arg(READ,  CU,       POINTWISE),        & ! un
             arg(READ,  CV,       POINTWISE),        & ! vn
             arg(READ,  R_SCALAR, POINTWISE),        & ! Time-step
             arg(READ,  GRID_AREA_T)                 &
           /)
     integer :: ITERATES_OVER = INTERNAL_PTS
     integer :: index_offset = OFFSET_NE
  contains
    procedure, nopass :: code => continuity_code
  end type continuity
  ...
  subroutine continuity_code(ji, jj,                      &
                             ssha, sshn, sshn_u, sshn_v, &
                             hu, hv, un, vn, rdt, e12t)
    ...
  end subroutine continuity_code
end module continuity_mod
```

Science & Technology Facilities Council

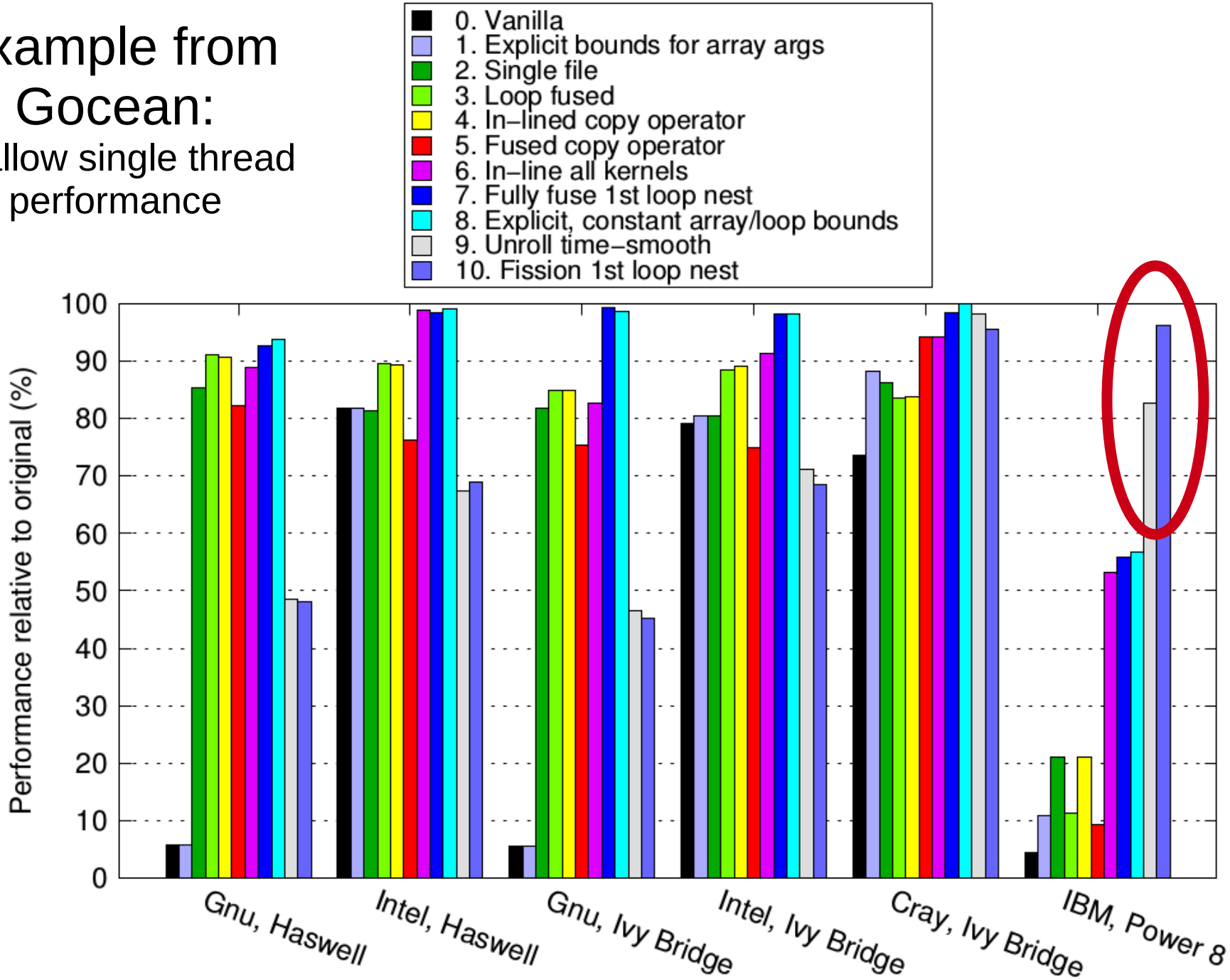
PSyclone Code Generation

- Generate the PSy layer, modify the algorithm layer
- Code generation requires, Alg API, Kern API, Inf API, Kernel metadata
- Code generation can help with
  - Generation of boilerplate code
  - Testing for correctness
  - Changes in interfaces
  - Flexible implementation of optimisations (not hard coded)
    - Single code base
    - Portable performance

Science & Technology Facilities Council

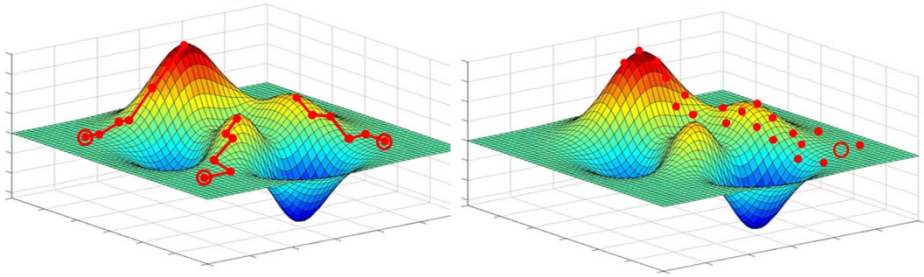
Example from Gocean:

Shallow single thread performance



Performance Portability?

- Complex parallel Fortran code
  - Complex optimization space



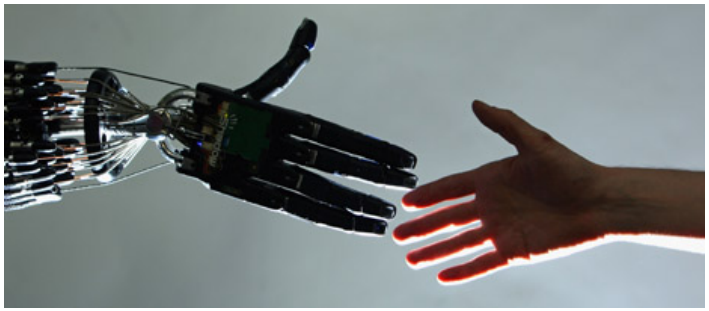
- Complex parallel architectures
  - Different minima

- Complex software stack (compilers etc.)
  - Different minima

Science & Technology Facilities Council

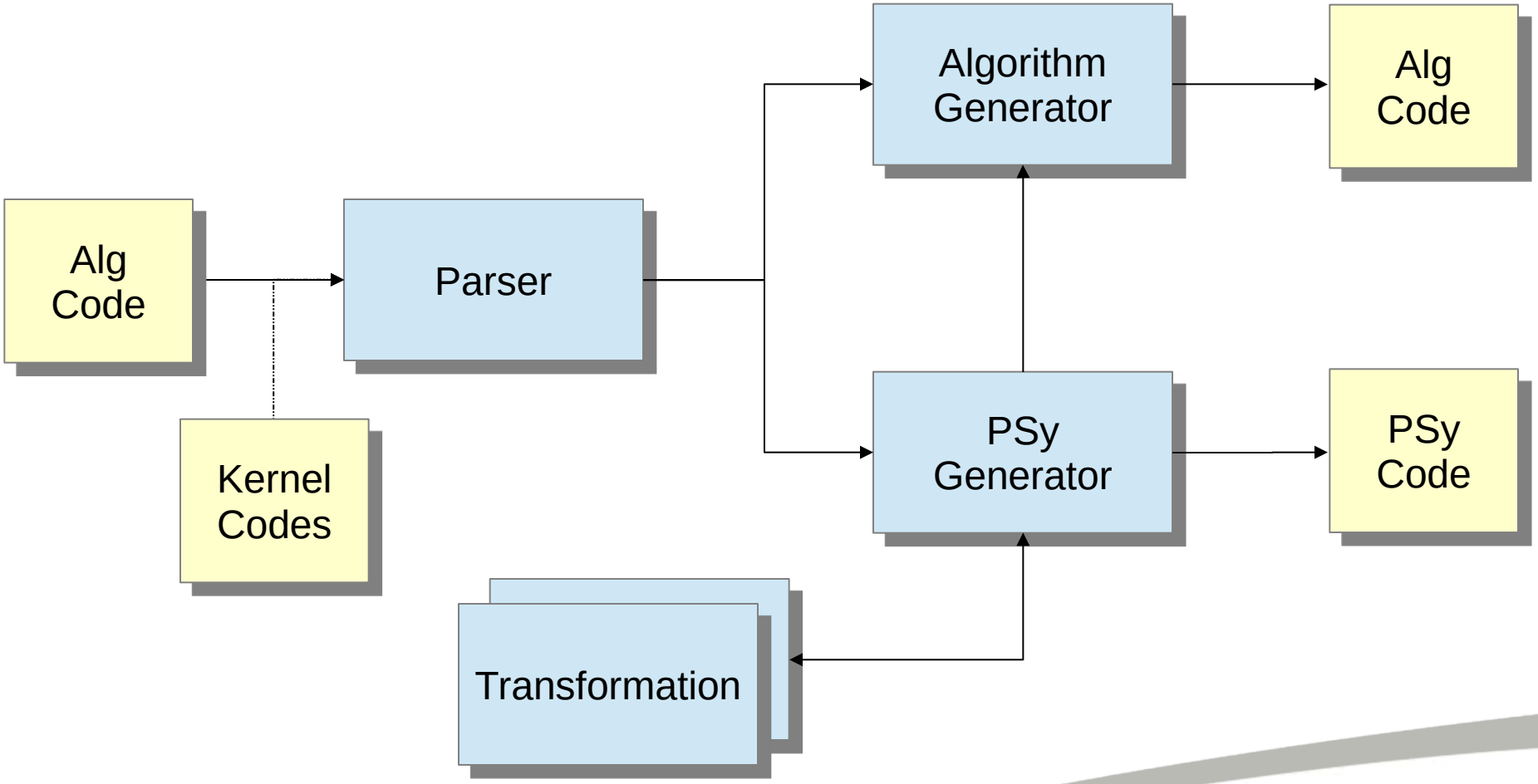
PSyclone Code Generation

- Psyclone takes a user-specified optimisation approach to support the expert
- Can optimisations be automated?
  - Perhaps for “reasonable” performance
  - To search the optimisation space?
- Compile time optimisation (static analysis) vs. runtime (dynamic)



Science & Technology Facilities Council

PSyclone



Science & Technology Facilities Council

Using PSyclone

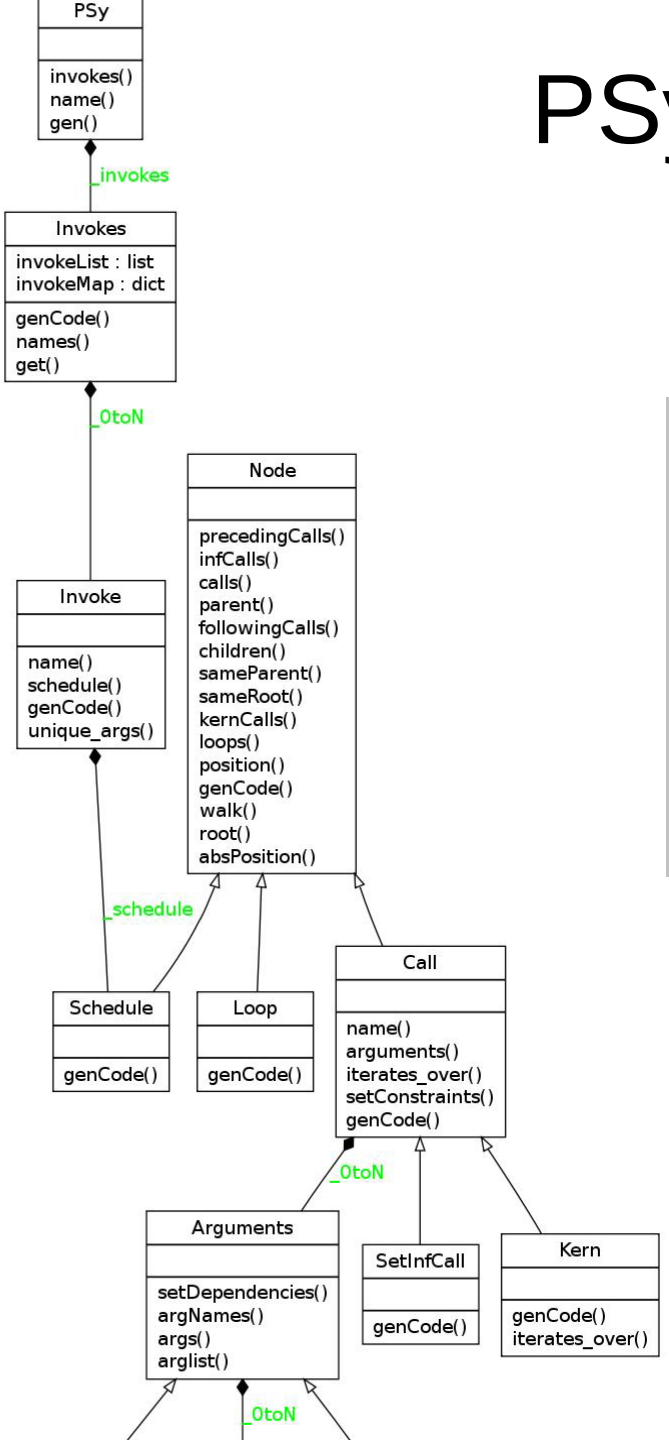
```
$ python generator.py [-h][-oalg ALG][-opsy PSY][-api API][-s SCRIPT][-d
DIR] [-l] file
```

```
>>> from generator import generate
>>> psy, alg = generate(“example.f90”, api=”dynamo0.3”,
kernel_path=”path” script_name=”scr.py”, line_length=False)
>>> print psy.gen
>>> print alg.gen
```

```
>>> from algGen import Alg
>>> from parser import parse
>>> from psyGen import PSyFactory
>>> ast, info = parse(“example.f90”, api=”dynamo0.3”, kernel_path=”path”,
line_length=False)
>>> psy = PSyFactory(“dynamo0.3”).create(info)
>>> alg = Alg(ast, psy)
>>> print psy.gen
>>> print alg.gen
```

Science & Technology Facilities Council

PSy schedule

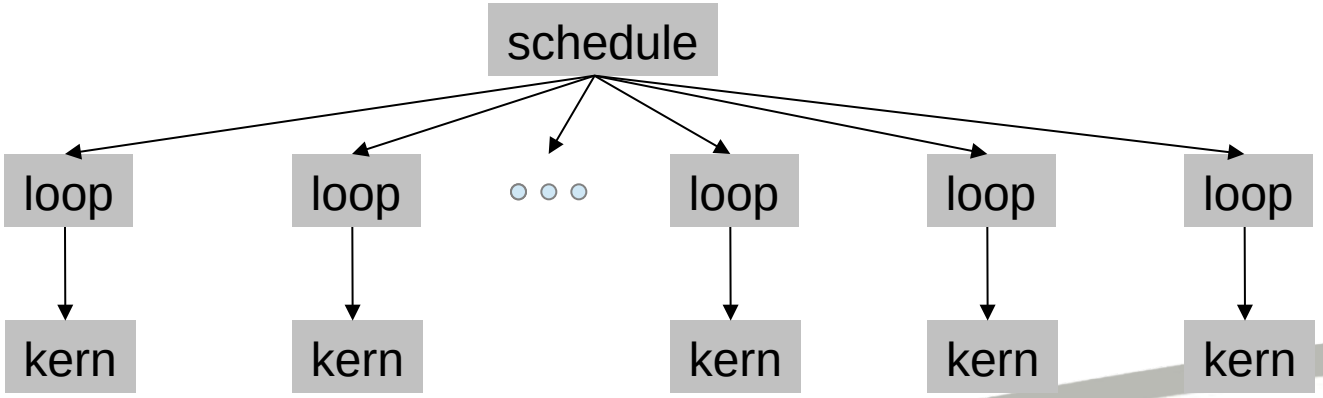



```
>>> psy = PSyFactory("dynamo0.3").create(info)
>>> invokes = psy.invokes
>>> invokes.names
>>> invoke = invokes.get("name")
>>> schedule = invoke.schedule
>>> schedule.view()
```

Science & Technology Facilities Council

Schedule internal tree representation

```
...
call invoke(                                                    &
            continuity(ssha_t, sshn_t, sshn_u, sshn_v,          &
                       hu, hv, un, vn, rdt),                    &
            momentum_u(ua, un, vn, hu, hv, ht,                  &
                       ssha_u, sshn_t, sshn_u, sshn_v),         &
            ...
            copy(sshn_t, ssha_t),                               &
            next_sshu(sshn_u, sshn_t),                          &
            next_sshv(sshn_v, sshn_t)                           &
           )
...
```



Science & Technology Facilities Council

Status

- Generates correct sequential code for dynamo0.1, dynamo0.3, gocean0.1 and gocean1.0 APIs
- Transformations: Loop fusion, OpenMP (parallel, do, parallel do), Colouring, Module inlining, Constant loop bounds
- Used by the Dynamo build system
- Initial Dynamo OpenMP parallel results
- Kernel Stub generator
- Trunk: 8,667 lines of Python, ~50% generic
- Trunk: 273 pytest tests, 6,232 lines of Python

Performance Results

- Compare hand-coded PSyKAl implementation to PSyclone generated implementation
- NEMOLite2D benchmark (developed by NOC in the GOcean project)
- Archer (Intel Ivy Bridge) and Intel 15.0.0.9, Cray 8.3.3 and Gnu 4.9.1 compilers
- Concentrate on OpenMP performance (because this is implemented!)
- Chosen a Static schedule with default block size in all cases (as that gave the best performance for the hand-optimised version).
- Codes the same except : Field copy subroutine vs field copy kernel

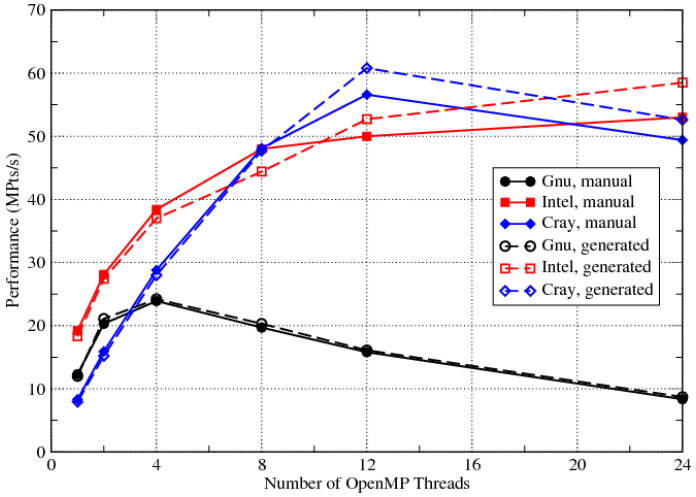
NEMOLite2D – parallel do script

```
$ python ${PSYCLONEHOME}/src/generator.py -api gocean1.0 -oalg
alg.f90 -opsy psy.f90 nemolite2d_alg.f90

$ cat ./script.py
def trans(psy):
    schedule = psy.invokes.get('invoke_0').schedule
    ltrans = GOceanOMPParallelLoopTrans()
    for child in schedule.children:
        newschedule, _ = ltrans.apply(child)
        schedule = newschedule
    psy.invokes.get('invoke_0').schedule = schedule
    return psy

$ python ${PSYCLONEHOME}/src/generator.py -api gocean1.0 -oalg
alg.f90 -opsy psy.f90 -s ./script.py nemolite2d_alg.f90
```

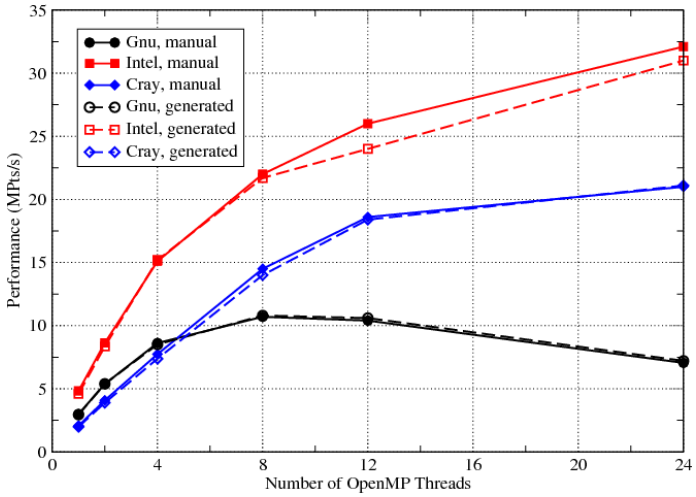
NEMOLite2D – OMP parallel do

Solid lines – manual OpenMP
Dashed lines – PSyclone-generated

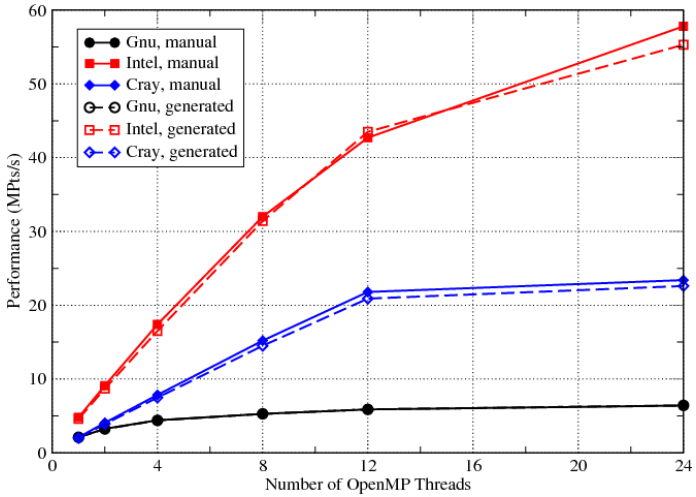
32



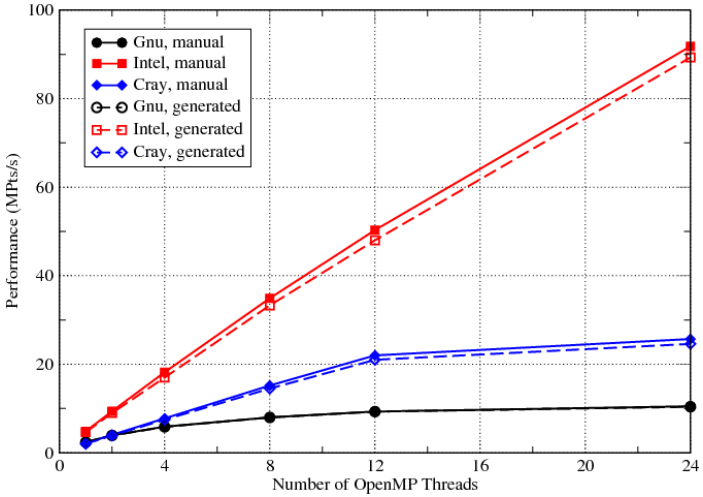
64



128



256



NEMOLite2D – single parallel region script + internal representation

```
$ cat ./script2.py
def trans(psy):
    schedule = psy.invokes.get('invoke_0').schedule
    ltrans = GOceanOMPLoopTrans()
    rtrans = OMPParallelTrans()
    for child in schedule.children:
        newschedule, _ = ltrans.apply(child)
        schedule = newschedule
    newschedule, _ = rtrans.apply(schedule.children)
    psy.invokes.get('invoke_0').schedule = newschedule
    return psy

$ python ${PSYCLONEHOME}/src/generator.py -api gocean1.0 -oalg
alg.f90 -opsy psy.f90 -s ./script2.py nemolite2d_alg.f90
```

```
Schedule[invoke='invoke_0']
    Directive[OMP Parallel]
        Directive[OMP do]
            Loop[type='outer',field_space='ct',it_space='internal_pts']
                Loop[type='inner',field_space='ct',it_space='internal_pts']
                    Call continuity_code(ssha_t,sshn_t,sshn_u,...)
        ...
```

Science & Technology Facilities Council

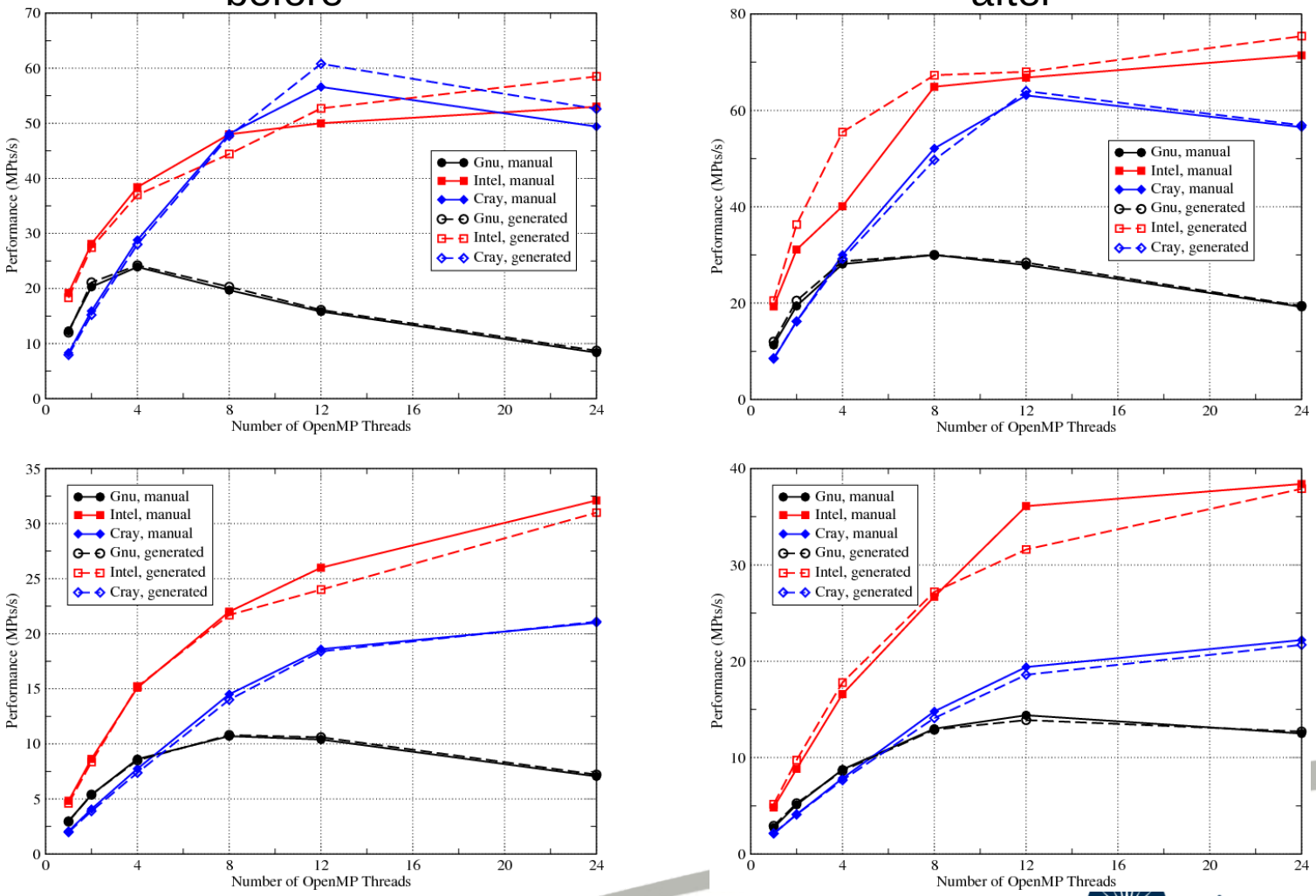
NEMOLite2D – single parallel region

before

after

32

64



Science & Technology Facilities Council

Summary

- PSyclone is a code generation and optimisation system for the “separation-of-concerns” PSyKAl approach
- PSyclone is part of the Dynamo build system for the GungHo dynamical core
- PSyclone has been used to generate parallel OpenMP Dynamo code
- For the NEMOLite2D benchmark there is no overhead using PSyclone generated OpenMP code compared with hand written OpenMP code

Future Work

- Continue to support API extensions for the Met Office Dynamo
- Support MPI parallel
- Extend scalar transformations and complete OpenACC transformation support
- Extend benchmarks (Dynamo test cases) and architectures (Xeon Phi and GPU performance analysis)
- Search the optimisation space
- Kernel optimisations

If you have been …

… thank you for listening

Mike Ashworth

mike.ashworth@stfc.ac.uk

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