Science and
Technology
Facilities Council

PSyclone LFRic single node support

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

ESIWACE2 training course on Domain-specific Languages in Weather and Climate, 23rd-27th November 2020

Hartree Centre
Science & Technology Facilities Council
Met Office
Australian Government
Bureau of Meteorology

Single node vs shared memory

- Distributed memory vs Shared memory
- Multi-node vs Single node

- Usually distributed memory ⇔ multi-node

- But, single node may have accelerators so perhaps not "shared memory"

- Want to get performance from single node
    - Parallelism (cores), utilise accelerators, memory optimisations

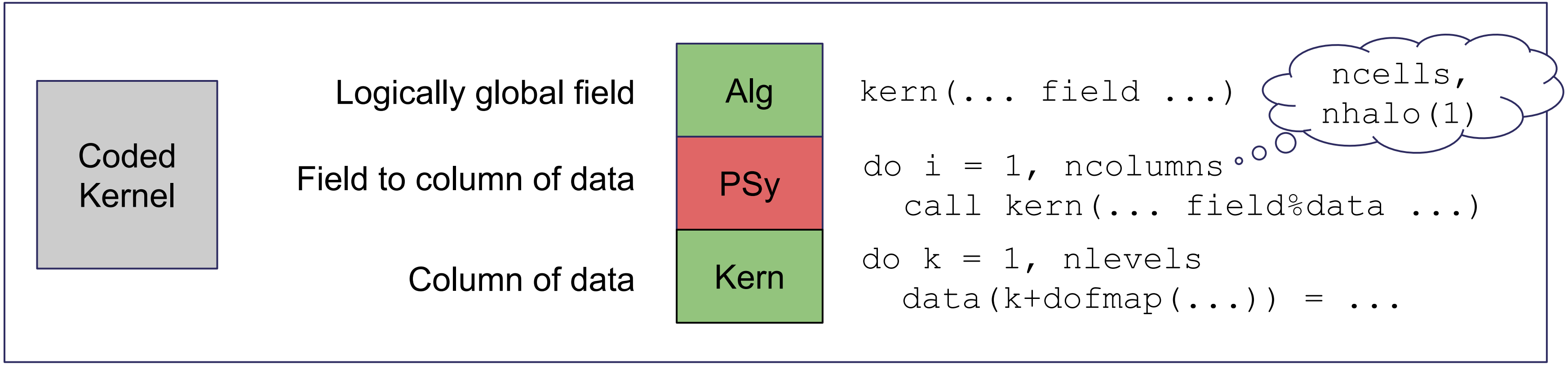
ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

Overview

- 90 minute session
- Introduction then hands on tutorials
- Introduction
  - ~25 minutes
- Hands on
  - ~65 minutes
  - 3 parts
    - OpenMP
    - OpenACC
    - Sequential optimisations
  - Any issues/questions on the slack channel

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

Loops in LFRic PSy-layer



Coded Kernel

Logically global field — Alg

```
kern(... field ...)
```

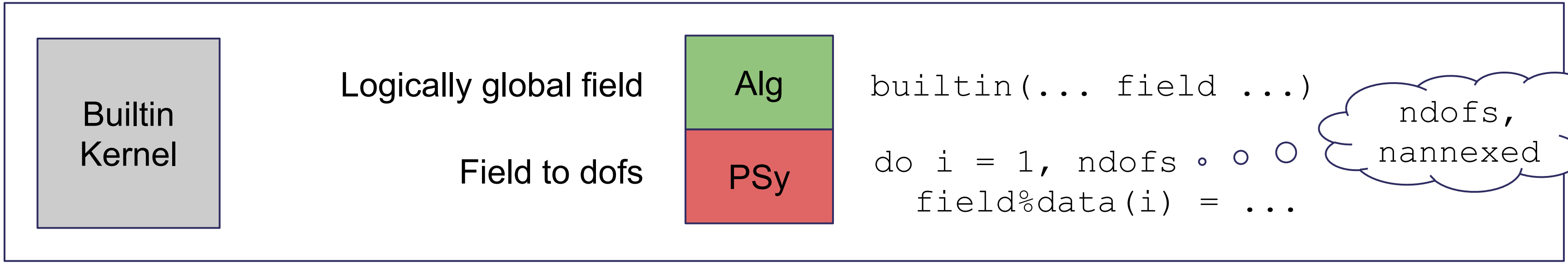
Field to column of data — PSy

```
do i = 1, ncolumns
  call kern(... field%data ...)
```

ncells,
nhalo(1)

Column of data — Kern

```
do k = 1, nlevels
  data(k+dofmap(...)) = ...
```



Builtin Kernel

Logically global field — Alg

```
builtin(... field ...)
```

Field to dofs — PSy

```
do i = 1, ndofs
  field%data(i) = ...
```

ndofs,
nannexed

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

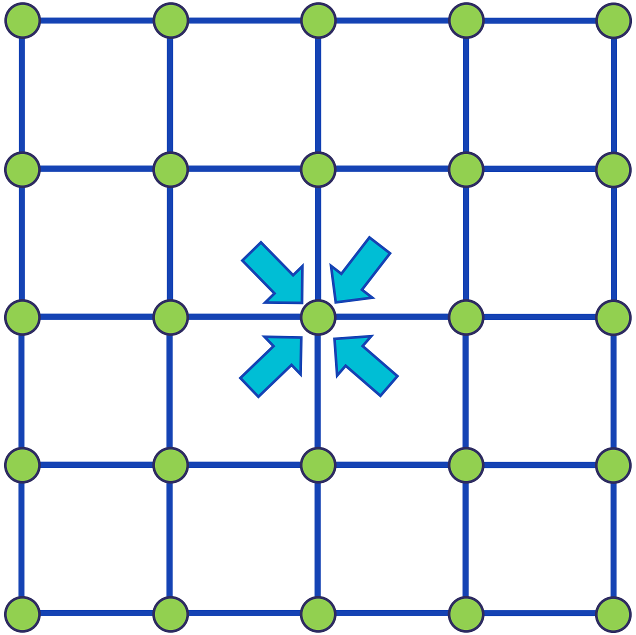
Loops in LFRic PSy-layer

- Exploit PSy-layer loop parallelism
  - not functional parallelism at the moment

- Loop types: key features
  - loop over cells
    - If dist mem and continuous then halo(1) else ncells
    - If continuous then inc access
      - Loop iterations not independent
    - If discontinuous then independent
  - loop over dofs
    - Loop iterations are independent (except reductions)
    - If dist mem and continuous then annexed dofs

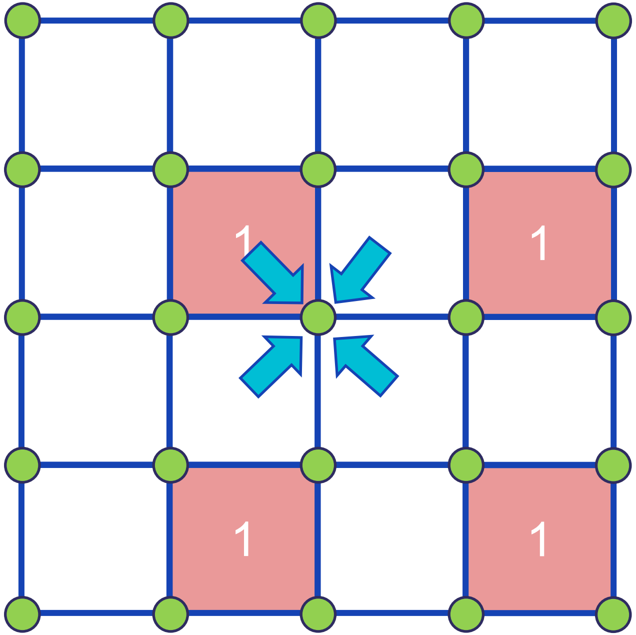
ESiWACE2 has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 823988

Colouring



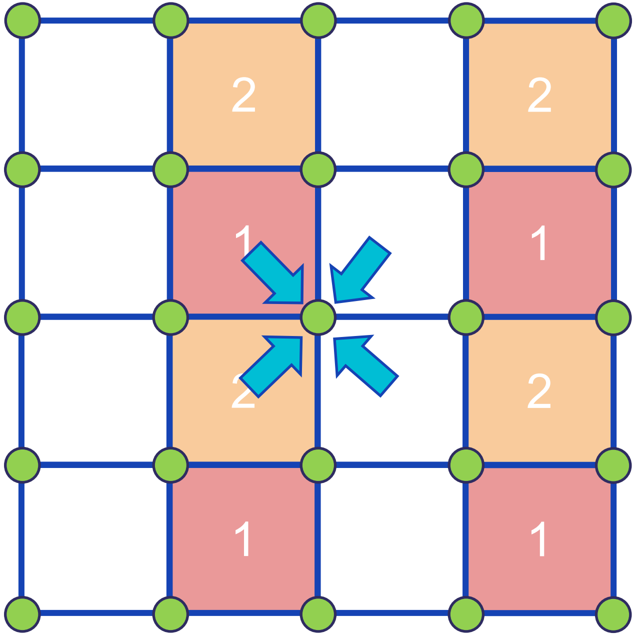
ESiWACE2 has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 823988

Colouring



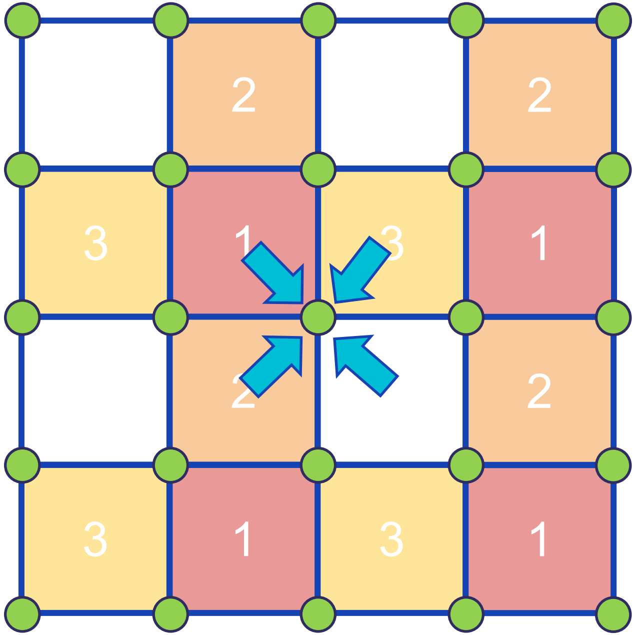
ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

Colouring



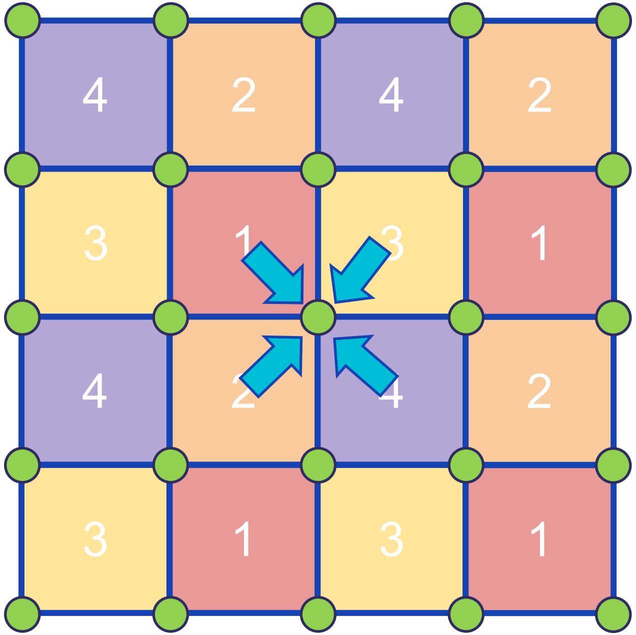
ESiWACE2 has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 823988

Colouring



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

Colouring

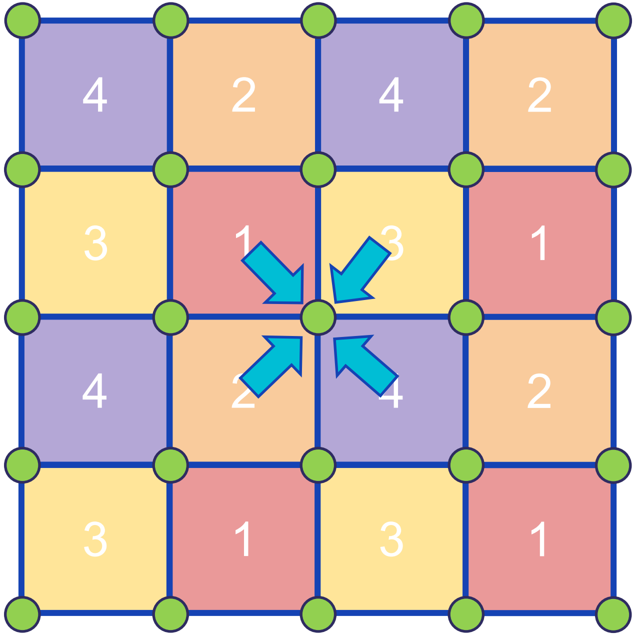


4	2	4	2
3	1	3	1
4	2	4	2
3	1	3	1

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

Colouring

```
do i = 1, ncells
   call kern ( … )
```



sequential

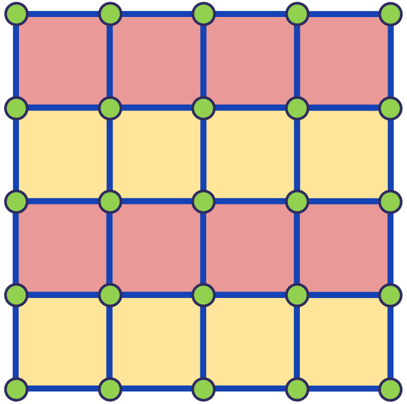
```
do colour = 1, ncolours
   do i = 1, ncolour(colour)
     call kern ( … )
```

parallel

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

Colouring

Two threads
Threads 0 and 1



- Future possibilities
  - Blocked colouring
  - Locks instead of colouring
  - Loop over vertices (for lowest order)

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

Reductions in loops

```
do i = 1, n
  a = a + data(i)
end do
```

Partial sums,
Locking, ...

- Reproducibility
  - Summing floating point numbers in different orders can produce different results
  - Considered important to be able to have reproducible reductions
    - Same code, same environment, same results
    - Testing
    - Debugging
    - Requirement for some operational configurations

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

Modifying kernels

- PSyclone creates PSyIR for the PSy-layer
- This is modified to add directives etc to improve performance
- Code is then output

- Might want to also modify existing kernel code
  - Restructure for performance
  - Add directives for parallelisation
- PSyclone also translates kernel code to PSyIR
- This can then be transformed
- Modified code can then be output
- This is relatively recent functionality

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

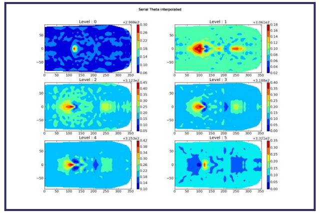
OpenMP

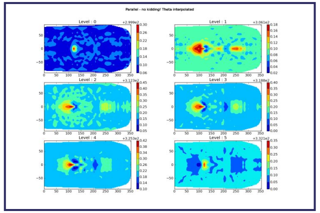
- Parallelisation for multi or many core CPU(s)
- Well used and supported
- Mixed mode / MPI + X
  - X = OpenMP

- OpenMP directives used (add via PSyclone transformations)
  - PARALLEL - declares a parallel region of code
  - DO - says to run a loop in parallel
  - PARALLEL DO -> PARALLEL + DO
  - DO and PARALLEL DO
    - support reductions - not guaranteed to be reproducible

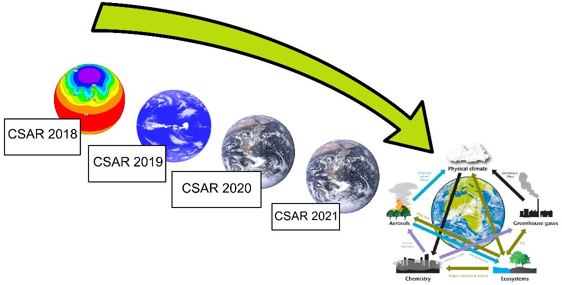
ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

OpenMP in practice





- PSyclone integrated into LFRic build system in September 2015 - serial
- LFRic went parallel (MPI + OpenMP) in March 2016
  - Switch was essentially immediate (but took 1 week in practice due to simple PSyclone OpenMP bug for reductions)
  - No change to science code from serial to parallel
- Science development has continued since then (including adding Physics)



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

OpenACC

- Parallelisation for GPU(s)
- Well used but only supported by one vendor
- Works for Fortran, help from NVIDIA
- Mixed mode / MPI + X
  - X = OpenACC
- Early days!
  - Partial functionality and bugs in implementations
- Other solutions?
  - Plan to support OpenMP GPU directives as well
  - Also see Sergi Siso’s presentation for what else we’re working on

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

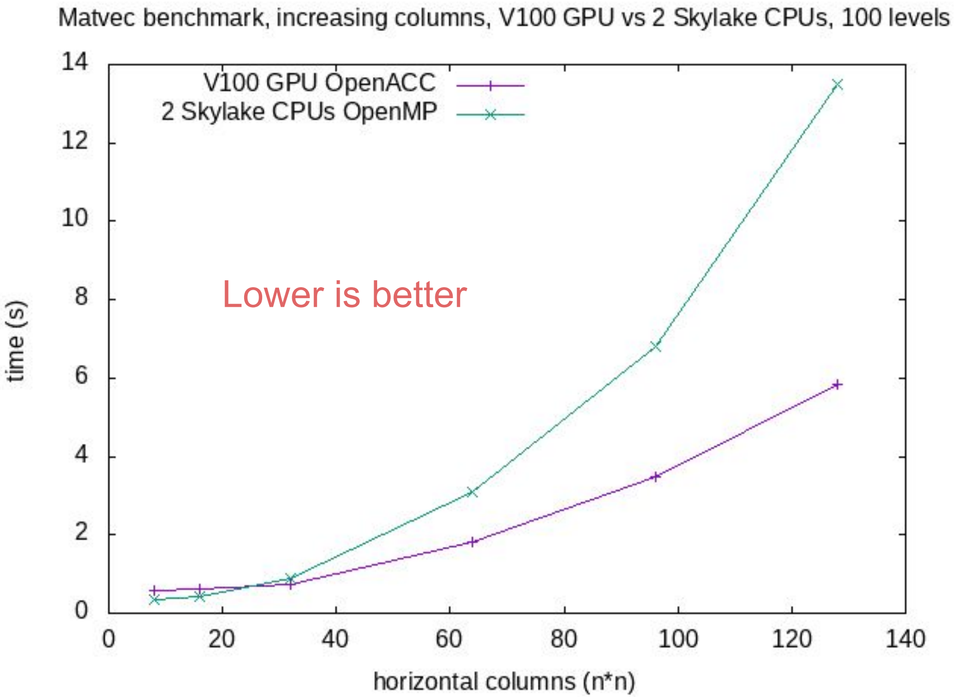
OpenACC

- OpenACC directives used (via PSyclone transformations)
  - KERNELS - GPU region, compiler responsible to parallelise
  - PARALLEL - GPU region, user responsible to parallelise
  - LOOP - specify a loop is parallel (within KERNELS or PARALLEL)
    - INDEPENDENT
  - ENTER DATA - specify data to copy from CPU to GPU (only copy data that is not already on GPU)
    - Avoids copying data between KERNELS and/or PARALLEL regions
  - ROUTINE - specify subroutine (kernel) will be run on GPU

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

OpenACC

- **Manual** matvec kernel results
- Time vs number of columns
- 2*16 core Skylake vs V100 GPU
- Green is optimised OpenMP which is 2* faster than current implementation
- Purple is optimised OpenACC which is up to 2* faster than optimised OpenMP



Lower is better

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

Sequential optimisations

- PSyclone transformation examples:

- Loop fusion in PSy-layer
  - Only if loop bounds are the same
  - Inc access can also stop fusion
- Constant values in kernels
  - e.g. nlayers, ndofs
  - Can help the compiler
  - Optimised for a particular configuration
- Fortran intrinsics: matmul (matvec)
  - Not available in other languages/representations
  - Expose the looping to allow restructuring - GPU opt

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

Hands on

- PSyclone transformation examples:
- ~65 minutes
- `cd <psyclone_home>/tutorial/practicals/LFRic/single_node`
- 3 parts `1_openmp, 2_openacc, 3_sequential`
- No compilation, just code generation
- Follow `README.md` in each directory
  - A browser will display `README.md` files nicely
  - https://github.com/stfc/PSyclone/tree/master/tutorial/practicals/LFRic/single_node
- Any issues/questions on slack
  - Use the psyclone channel
  - Please use threads for replies

Have fun!

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