Science and Technology Facilities Council

Hartree Centre

Transforming Weather and Climate Code with PSyclone

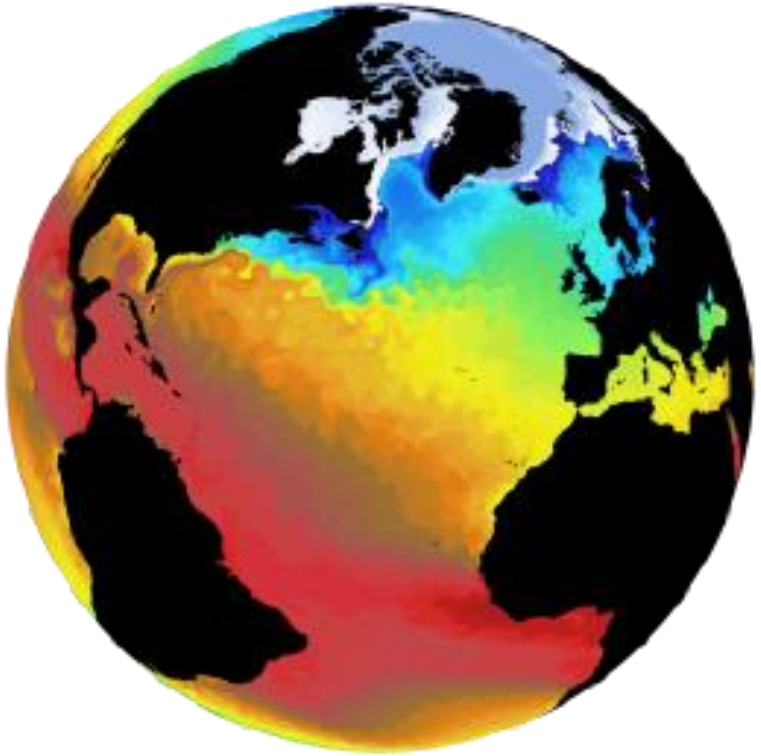
Rupert Ford[1], Andrew Porter[1], **Sergi Siso**[1], Aidan Chalk[1], Iva Kavcic[2], Chris Maynard[2], Joerg Henrichs[3], …

[1]STFC Hartree Centre, [2]UK Met Office,
[3]Australian Bureau of Meteorology

20th ECMWF workshop on high performance computing in meteorology - Bologna - Oct 23

Motivation

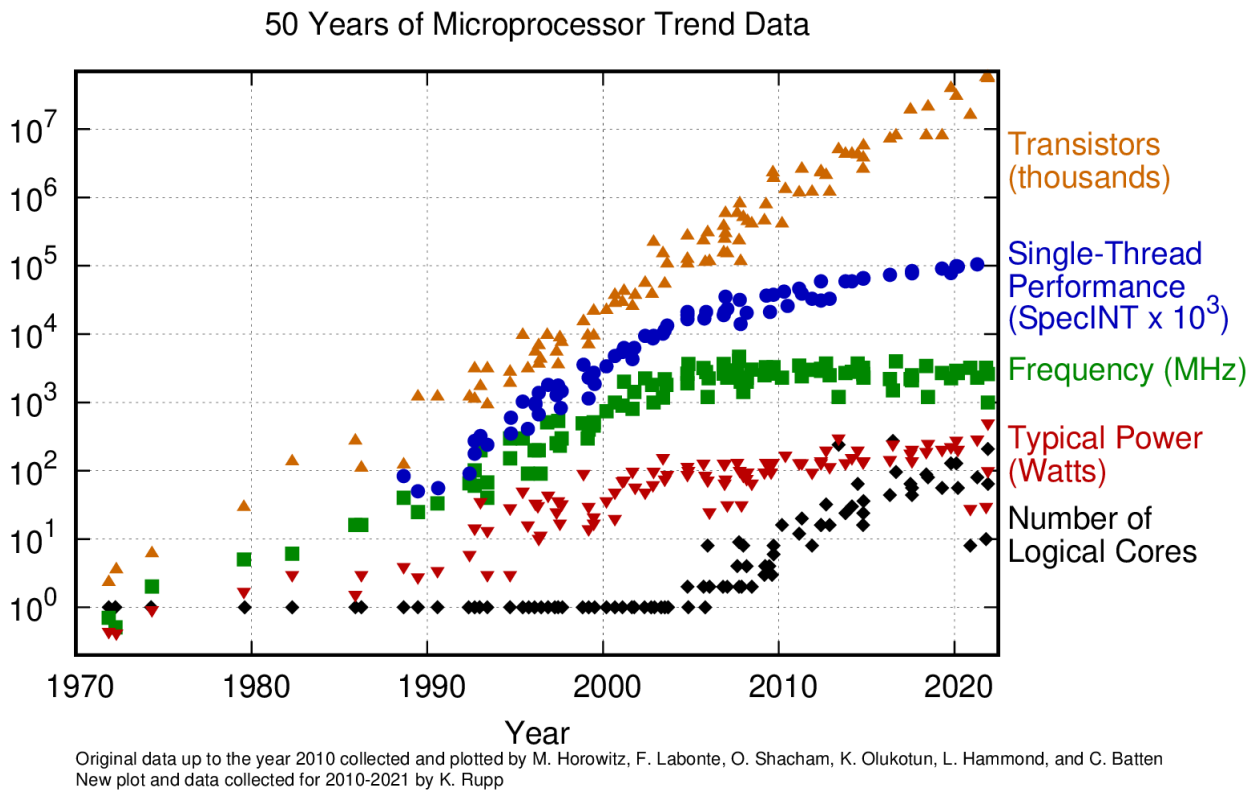
NEMO: Nucleus for **E**uropean **M**odelling of the **O**cean



UK MetOffice **LFRic** (replacement for UKMO UM)

Large FORTRAN codes that run on CPU-based HPC systems

The future (and present) of HPC is heterogeneous



Source: https://github.com/karlrupp/microprocessor-trend-data

Top 500 list November 2022

Position	Name	Processor	Linpack PFlop/s
#1	Frontier	AMD EPYC 64 cores AMD Instinct MI250X	1,102
#2	Fugaku	Fujitsu A64FX 48C	442
#3	Lumi	AMD EPYC 64 cores AMD Instinct MI250X	309
#4	Leonardo	Xeon Platinum 8358 32C Nvidia A100 SMX4	174
#5	Summit	IBM POWER9 22C Nvidia V100	148
#7	Taihulight	Sunway SW26010 260C	93
#10	Tianhe-2A	Intel Xeon E5-2692v2 12C MATRIX-2000	61

And upcoming Intel GPUs, Nvidia CPUs, RISC-V, FPGAs, …

Collaborative development

- Applications have many contributors from multiple institutions:
  - Run on different systems in each institution. Hard to maintain multiple implementations. It must be portable.
  - Contributors from different areas of expertise. Productivity, readability, maintainability are essential for the sustainability of the project.
  - Millions LOCs of FORTRAN (validated long-standing code)

-> Single source, with performance/parallelisation details abstracted

Fortran portability options

Fortran lacks the meta-programing mechanisms that C++ performance portability frameworks use.

- Pre-processor macros: Sometimes seen in HPC codes but impacts software readability.
- CUDA Fortran: Good performance but proprietary, single vendor support.
- Fortran do concurrent: Not widely adopted yet. Irregular compiler support.
- OpenMP4+ and OpenACC: Can be written in different styles. Compilers support different versions/features but getting better.

Portability
!=
Performance Portability

AoS vs SoA vs AoSoA

Redundant comp vs sharing

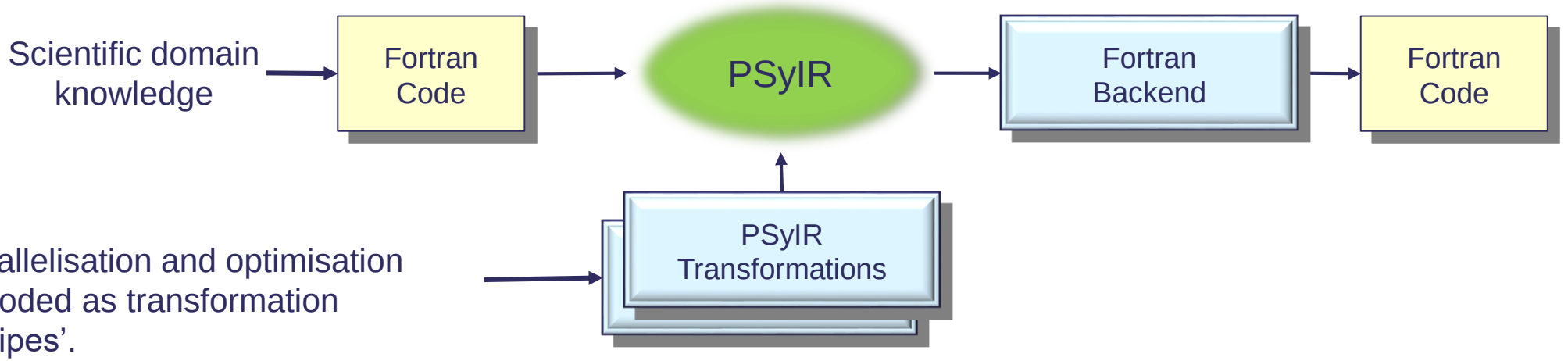
Colouring vs Atomics

SIMD vs SIMT

Can performance portability abstractions be achieved by source-to-source transformations?

PSyclone overview

PSyclone (BSD 3-clause)
https://github.com/stfc/PSyclone



Separation of concerns

Perf portability: diff. recipes for each architectures

Visible "*readable*" output Standard debugging/profiling

Standard output composable w. other tools and vendor compilers

Development strategy

PSyclone

- Existing science code
- Incremental development
- Towards separation of concerns
- Partial control of the parallelisation and optimisation

Rest of this talk

DSLs PSyclone

- Needs code rewriting
- High starting cost
- Strict separation of concerns
- Full control of the model parallelisation and optimisation

Previous talk by UKMO (LFRic)

Code transformation approach: NEMO example

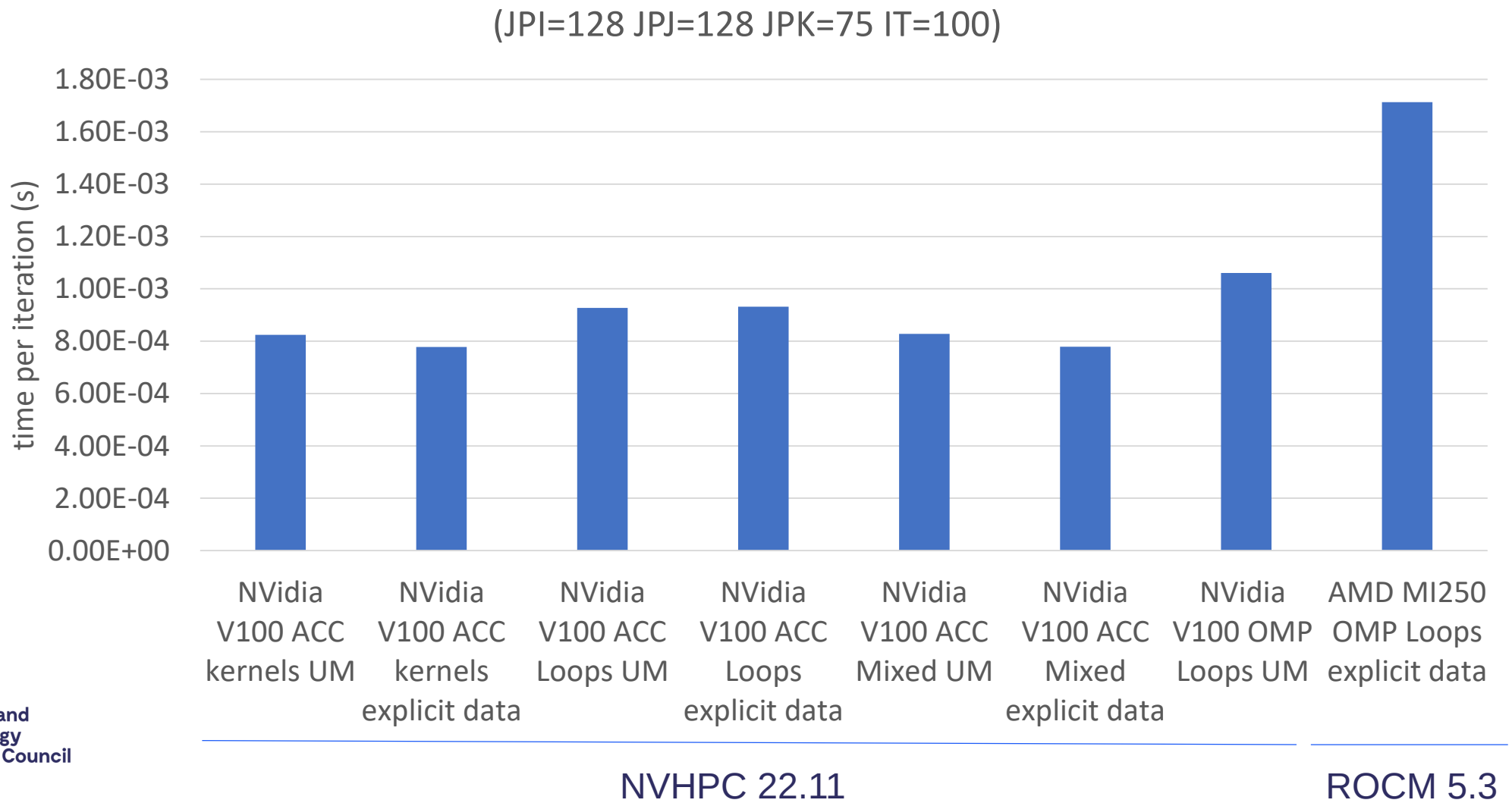
```
do jk = 1, jpkm1, 1
  zun(:,:,jk) = e2u(:,:) * e3u_n(:,:,jk) &
              & * (un(:,:,jk) + usd(:,:,jk))
enddo
```

```
for subroutine in psyir:

    for assign in subroutine.walk(Assignment):
        ArrayRange2LoopTrans.apply(assign)

    for loop in subroutine.walk(Loop):
        try:
            OMPLoopTrans().apply(loop)
            directive = loop.ancestor(Directive)
            OMPTargetTrans().apply(directive)
        except TransformationEror as err:
            print("Loop not accelerated:", err)
```

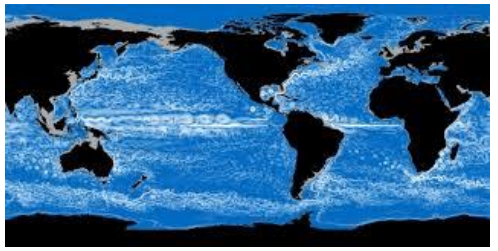
```
!$omp target
!$omp loop collapse(3)
do jk = 1, jpkm1, 1
  do idx_6 = LBOUND(zun, dim=2), UBOUND(zun, dim=2), 1
    do idx_7 = UBOUND(zun, dim=1),LBOUND(zun, dim=1), 1
      zun(idx_7,idx_6,jk) = e2u(idx_7,idx_6) * e3u_n(idx_7,idx_6,jk) &
                          & * (un(idx_7,idx_6,jk) + usd(idx_7,idx_6,jk))
    enddo
  enddo
enddo
!$omp end loop
!$omp end target
```

Dependency analysis
Compiler loop optimisations
+
Domain specific logic
Whole program optimisation

Porting the tracer_advection bench. to GPUs



NEMO psyclone-accelerated for GPUs

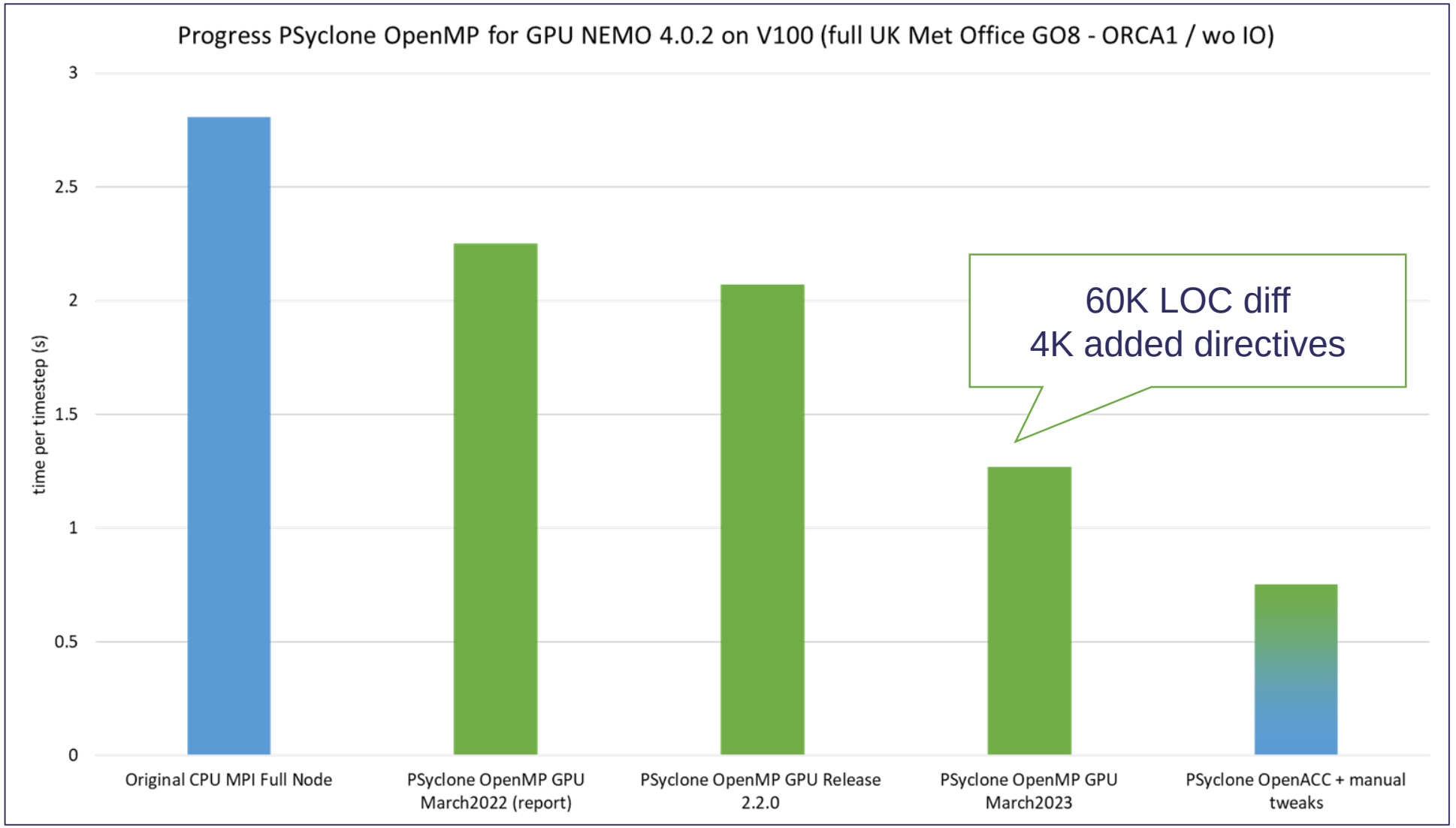


CPU:
Intel Xeon W-2133

GPU:
nvidia V100

Blue: manual opt
Green: psyclone opt

Transformation:
~200 LOCs



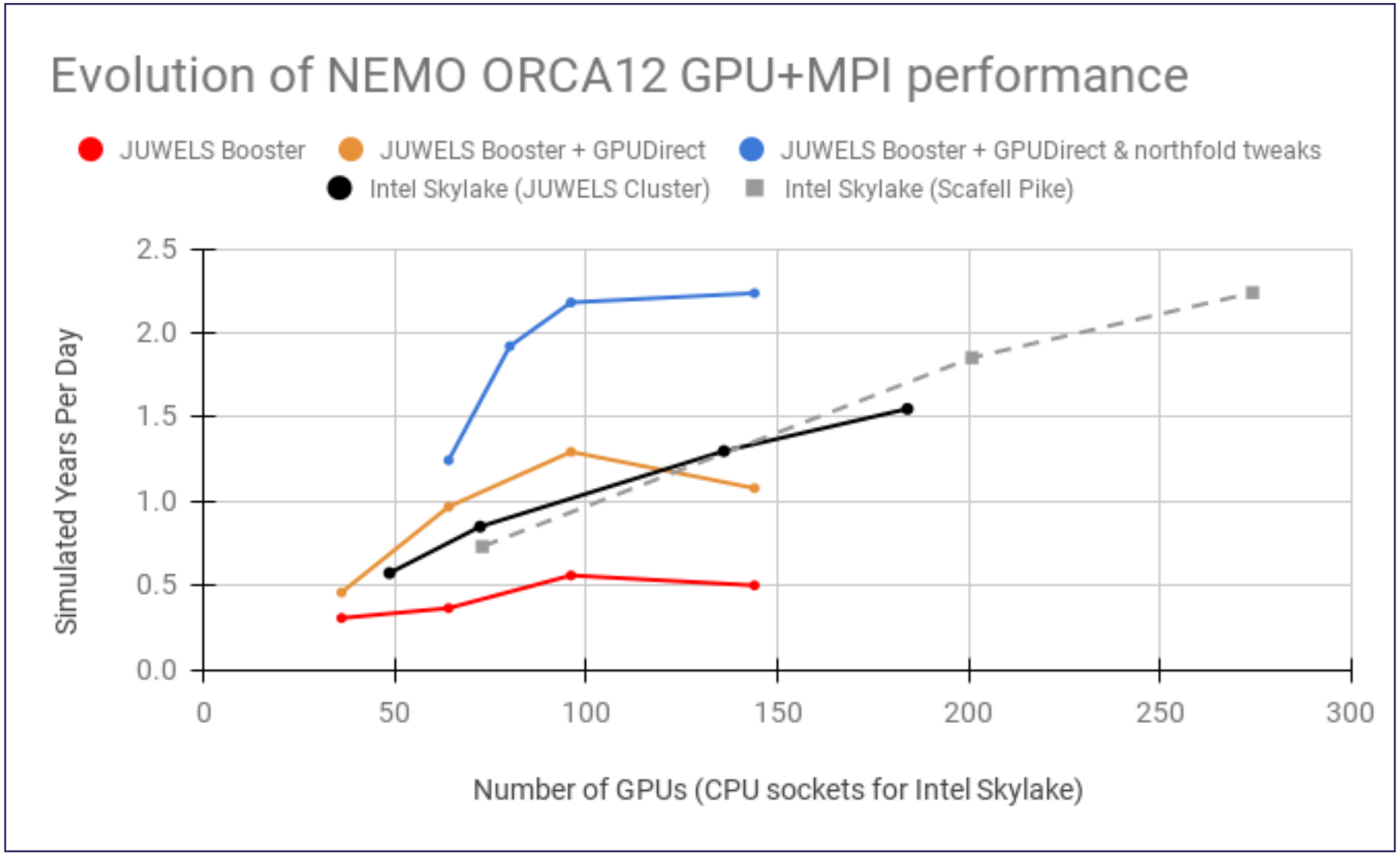
Science and Technology Facilities Council
Hartree Centre

NEMO (full UK Met Office GO8 configuration)

CPU:
Intel Xeon 8168
2× 24 cores

GPU:
nvidia A100

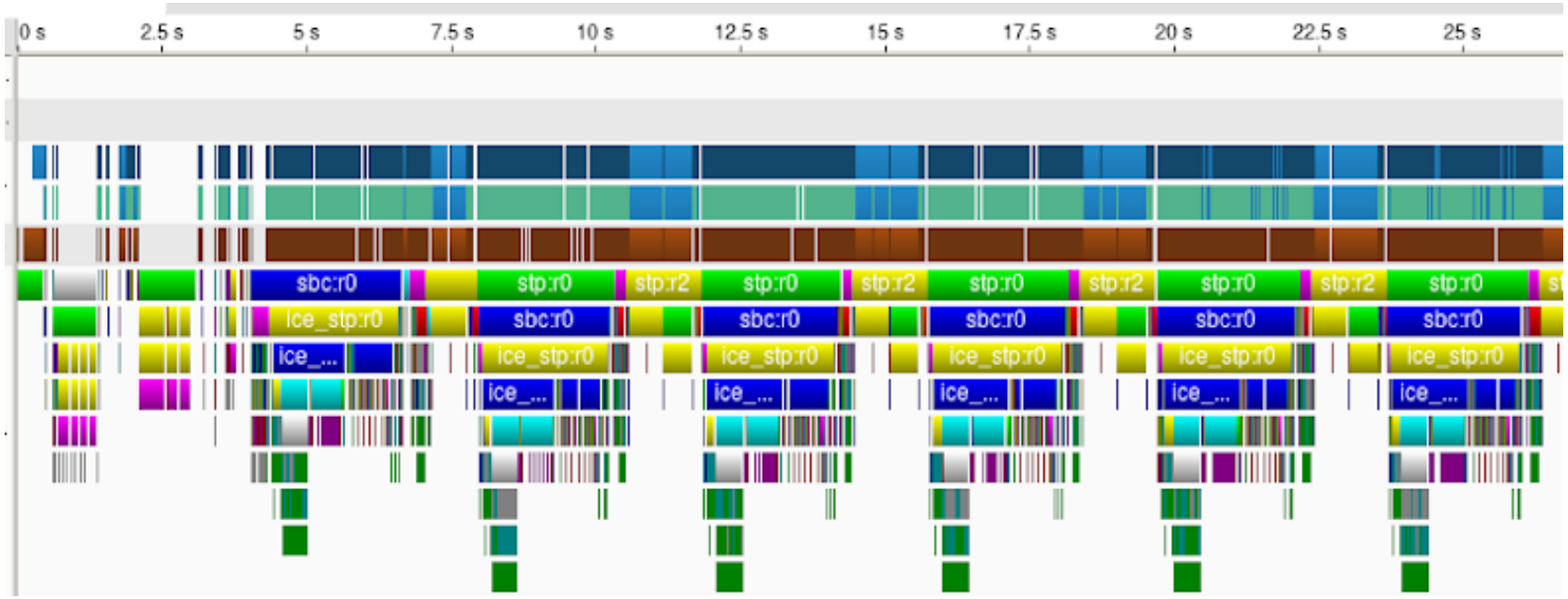
Transformation:
PSyclone OpenACC
+ manual tweaks
(by Chris Dearden)

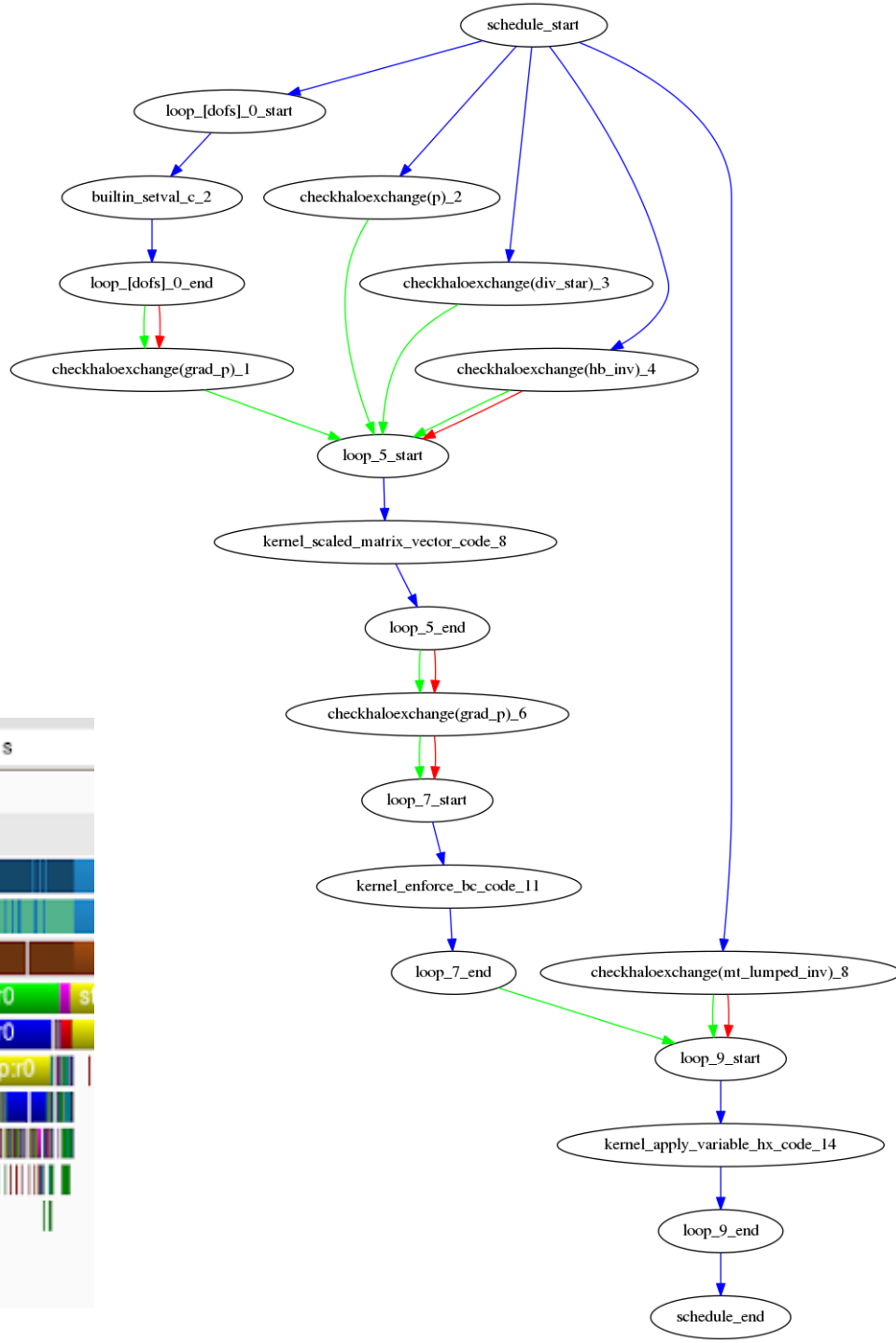


Additional PSyclone utilities

- DAG view of the Parallel System

- PSyData API - allows callipers to be inserted for e.g. profiling, debugging, validation, on-line visualisation etc.

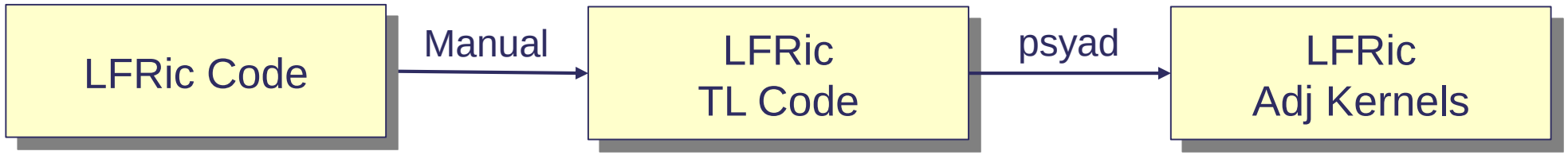




Additional PSyclone utilities

- PSyKE: PSyclone Kernel Extraction
  - Creates a standalone driver (single file with no dependencies)
  - Captures all data needed to execute the driver
  - Some limitations (e.g. private variables)

- PSyAD: PSyclone Adjoints for LFRic Data Assimilation



Extending it to other models

- NEMO:
  - PSyclone is now an optional part of NEMO build and test system (Simon Mueller @ NOC)
  - Applicability to other NEMO UKMO versions and configurations. (Daley Calvert @ UKMO)
  - PSyclone OpenMP CPU generation evaluated for ECMWF configuration of NEMO (Kristian Mogenson, Sam Hatfield @ ECMWF)
  - PSyclone is being evaluated for use with CMCC configurations of NEMO in ESiWACE3 (Italo Epicoco @ CMCC)
- MEDUSA: Good results inserting OpenACC (Simon Mueller @ NOC)
- NEMOVAR: Optimisation in progress by Met Office
- ROMS: Proof of concept PSyKAl DSL (Joerg Henrichs @ BOM)
- CROCO: OpenACC (Martin Schreiber and Sebastien Valat @ U. of Grenoble)
- NUMA3d: Initial exploration (collaboration with Frank Giraldo @ NPS)

Extending it to other models

- UKCA: Inserting OpenACC and optimisations for GPUs in progress by Met Office (Joe Abram, Joe Wallwork @ UKMO)
- Socrates (UM physical parametrisations): Inserting OpenMP for CPU is working, ongoing work to inject OpenMP offloading.

Research

- Automatic Fortran to OpenCL kernels transformation

Sergi Siso, Andrew R. Porter, and Rupert W. Ford. 2023. Transforming Fortran weather and climate applications to OpenCL using PSyclone. In Proceedings of the 2023 International Workshop on OpenCL (IWOCL '23), Article 10, 1–8. https://doi.org/10.1145/3585341.3585360

- ESiWACE IR interoperability

Fortran -> PSyIR -> SIR (gridtools) -> CUDA

- ExCALIBUR OpenMP tasking

PSyclone transformations to leverage asynchronous parallelism

- ExCALIBUR xDSL (Nick Brown @ University of Edinburgh)

Leveraging MLIR / LLVM ecosystem

Hartree Centre

Summary

- PSyclone is a FORTRAN source-to-source compiler for use with existing code and DSLs
- Separation of concerns and a tool for HPC experts
- Used with production/full configurations:
  - LFRic (parallelises next UK MetOffice atmospheric model)
  - NEMO (integrated in the build system and GPU demonstrator)
- Ongoing work:
  - Applying PSyclone to more applications
  - Improving OpenMP/ACC capabilities: memory movement, atomics, asynchronous kernels
  - LFRic offloading to GPUs

Science and Technology Facilities Council

Hartree Centre

Thank you

sergi.siso@stfc.ac.uk