Evolutionary Re-Engineering of an Industrial HPC Application with OP-DSL

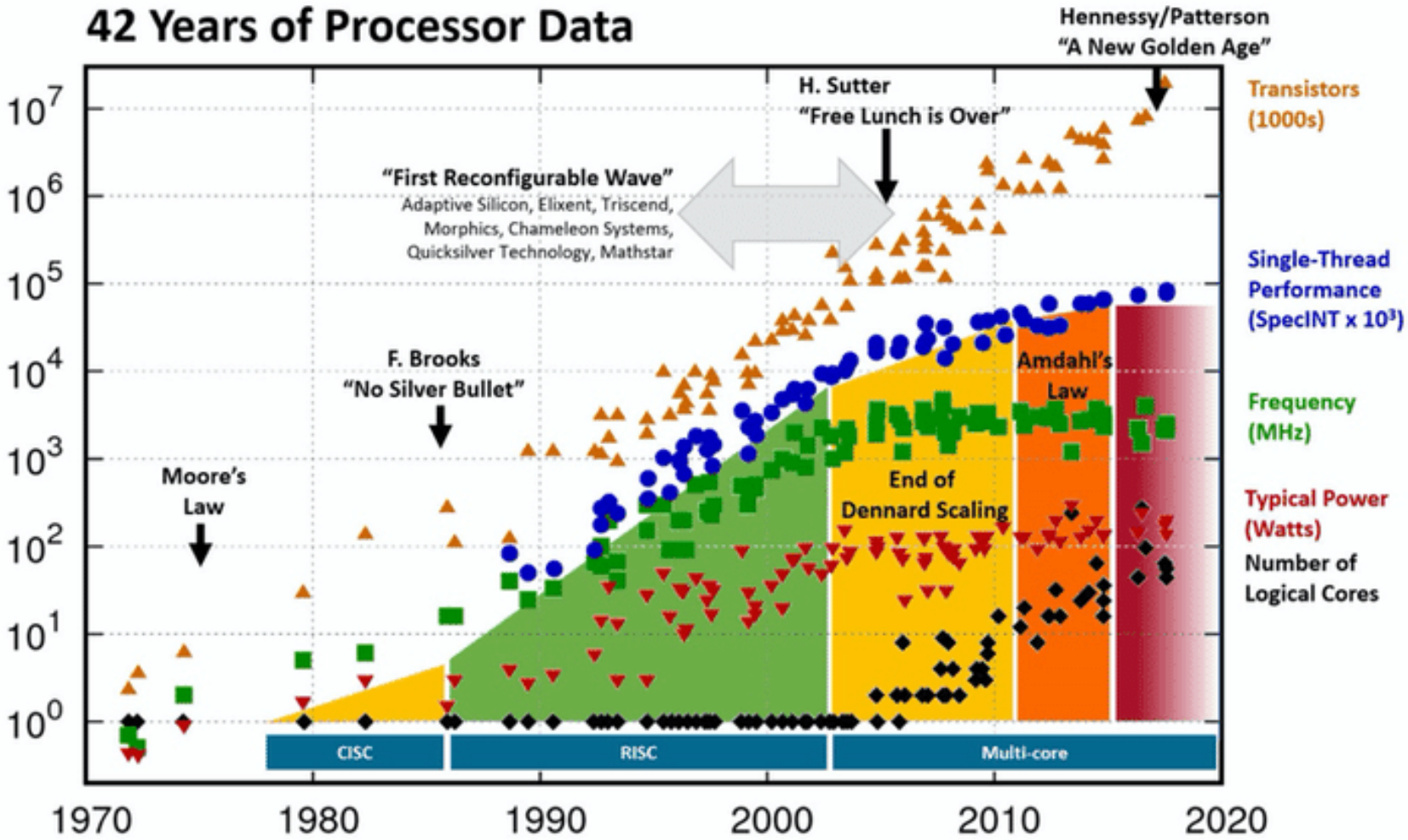
Gihan Mudalige
g.mudalige@warwick.ac.uk

Joint work with:
Istvan Reguly @ PPCU
Arun Prabhakar, Archie Powell and others at the HPSC group @ Warwick
Neil Sandham and team @ Southampton, Dario Amirante @ Surrey
Mike Giles @ Oxford, Paul Kelly and many more @ Imperial College London
Leigh Lapworth, Christopher Goddard, Paolo Adami and team @ Rolls-Royce plc. and Rolls-Royce Deutschland Ltd Co KG
NAG, UCL, STFC, IBM and many more industrial and academic collaborators.

22th August 2022 – EuroPar Workshop on Domain Specific Languages

WARWICK
THE UNIVERSITY OF WARWICK

Single thread speedup is dead – Must exploit parallelism



Hennessy and Patterson, Turing Lecture 2018, overlaid over "42 Years of Processors Data"
https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/;"First Wave" added by Les Wilson, Frank Schirrmeister
Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

Diverse Hardware Landscape – Compounded by the Race to Exascale !

- Traditional CPUs
  - Intel, AMD, ARM, IBM
  - multi-core (close to 100 cores)
  - Deep memory hierarchy (cache levels and RAM)
  - longer vector units (e.g. AVX-512)
- GPUs
  - NVIDIA (A100), AMD (MI200) , Intel (Xe GPUs)
  - Many-core (> 2k simpler SIMT cores)
  - CUDA cores, Tensor cores
  - Cache, Shared memory, HBM (3D stacked DRAM)
- Heterogeneous Processors
  - Different core architectures over the past years
  - ARM big.LITTLE
  - NVIDIA Grace.Hopper
- XeonPhi (discontinued)
  - Many-core – based on simpler x86 cores
  - MCDRAM (3D stacked DRAM)
- FPGAs
  - Dominated by Xilinx and Intel
  - Various configurations
  - Low-level language / HLS tools for programming
  - Significant energy savings
- DSP Processors
  - Phytium / The Chinese Matrix2000 GPDSP accelerator (Chinese Exascale systems)
- TPUs, IPUs ….
- Quantum ?

But .. Even more diverse ways to programming them !

OpenMP, SIMD,
CUDA, OpenCL, OpenMP4.0, OpenACC,
SYCL/OneAPI,
HIP/ROCm,
MPI, PGAS
Task-based (e.g Legion)
and others ….

- Open standards so far have not been agile to catch up with changing architectures : OpenMP, SYCL
- Proprietary programming models are restricted to narrow vendor specific hardware : CUDA, OpenACC, ROCm
- Need different code-paths/parallelization schemes to get the best performance
  - E.g. Coloring vs atomics vs SIMD vs MPI vs Cache-blocking tiling for unstructured mesh class of applications
- What about legacy codes ? There is a lot of FORTRAN code out there

OP-DSL

- Separation of Concerns (… back in 2010 !)
  - Specify the problem – not the implementation
  - Leverage the best implementation for the target context
  - Can be many contexts - hardware, programming model, parameters etc.

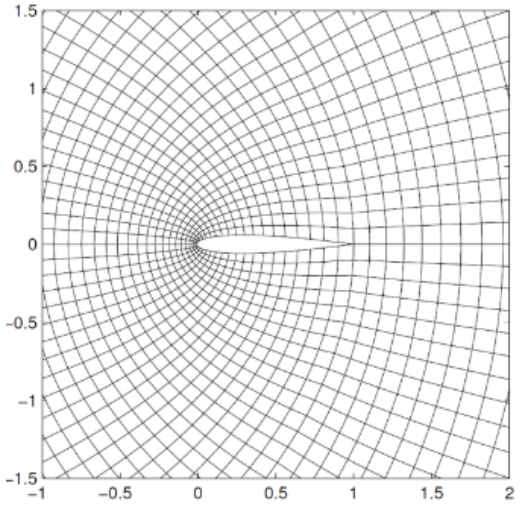
- Domain Specific API
  - Get application scientists to pose the solution using domain specific constructs – provided by the API
  - Handling data done only using API – contract with the user

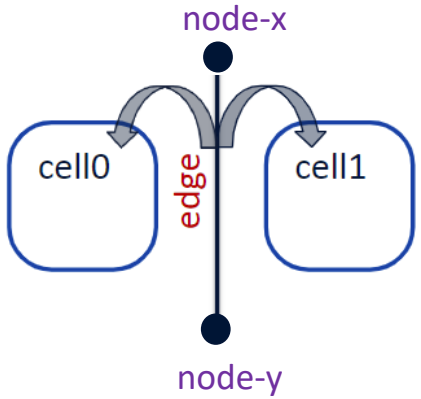
- Restrict writing code that is difficult (for the compiler) to reason about and optimize
  - "OP2 and OPS are a straitjacket" – Mike Giles
  - Build in safeguards so that user cannot write bad code !

- Implementation of the API left to a lower level
  - Target implementation to hardware – can use best optimizations
  - Automatically generate implementation from specification for the context
  - Exploit domain knowledge for better optimizations - reuse what we know is best for each context

OP2 API - EXAMPLE

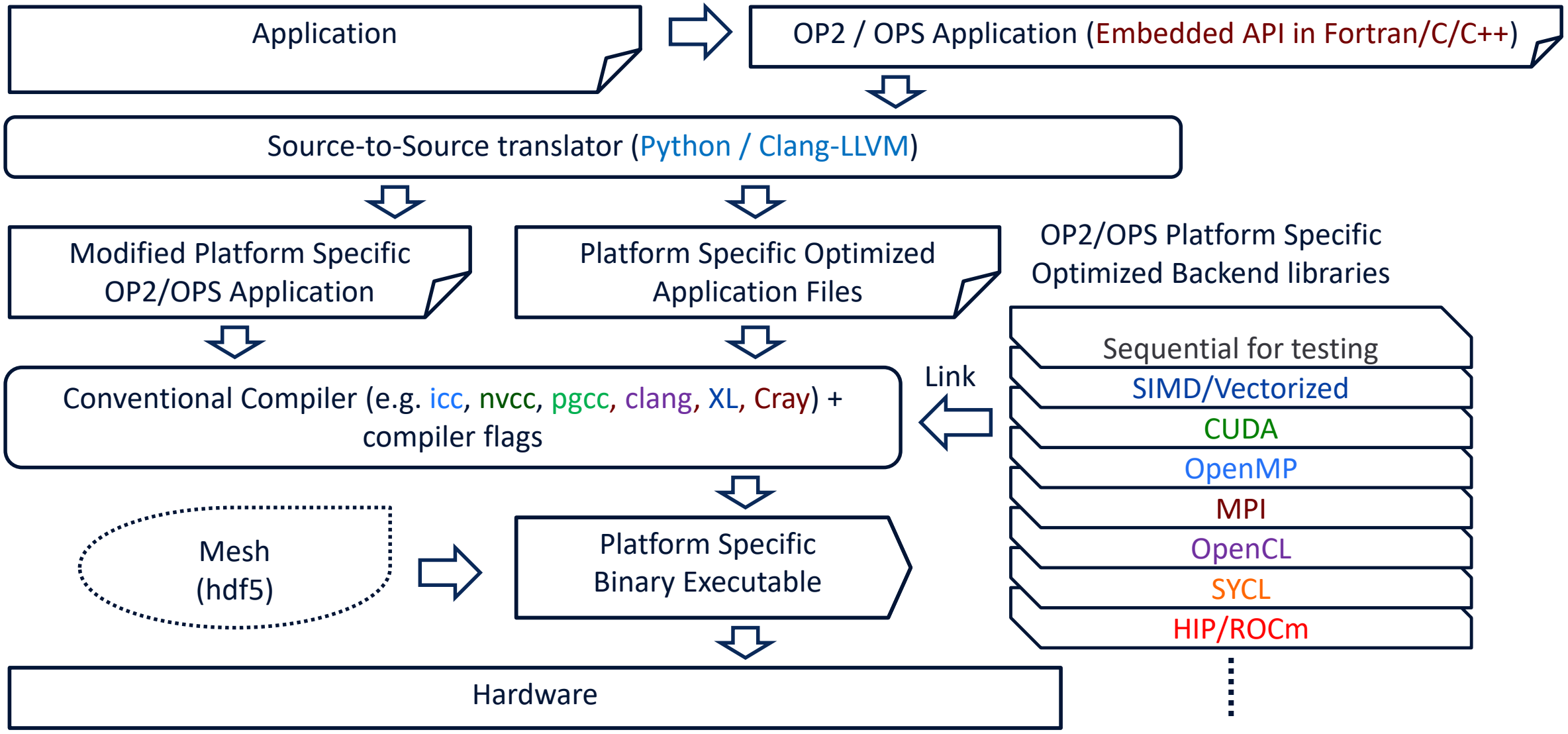
```
! Declaring the mesh with OP2
! sets
call op_decl_set(nnode,nodes,'nodes')
call op_decl_set(nedge,edges,'edges')
call op_decl_set(ncell,cells,'cells')
! maps
call op_decl_map(edges,nodes,2,edge ,pedge ,'pedge' )
call op_decl_map(edges,cells,2,ecell,pecell,'pecell')
! data
call op_decl_dat(nodes,2,'real(8)',x,p_x,'p_x')
call op_decl_dat(cells,4,'real(8)',q,p_q,'p_q')
call op_decl_dat(cells,1,'real(8)',adt,p_adt,'p_adt')
call op_decl_dat(cells,4,'real(8)',res,p_res,'p_res')

```



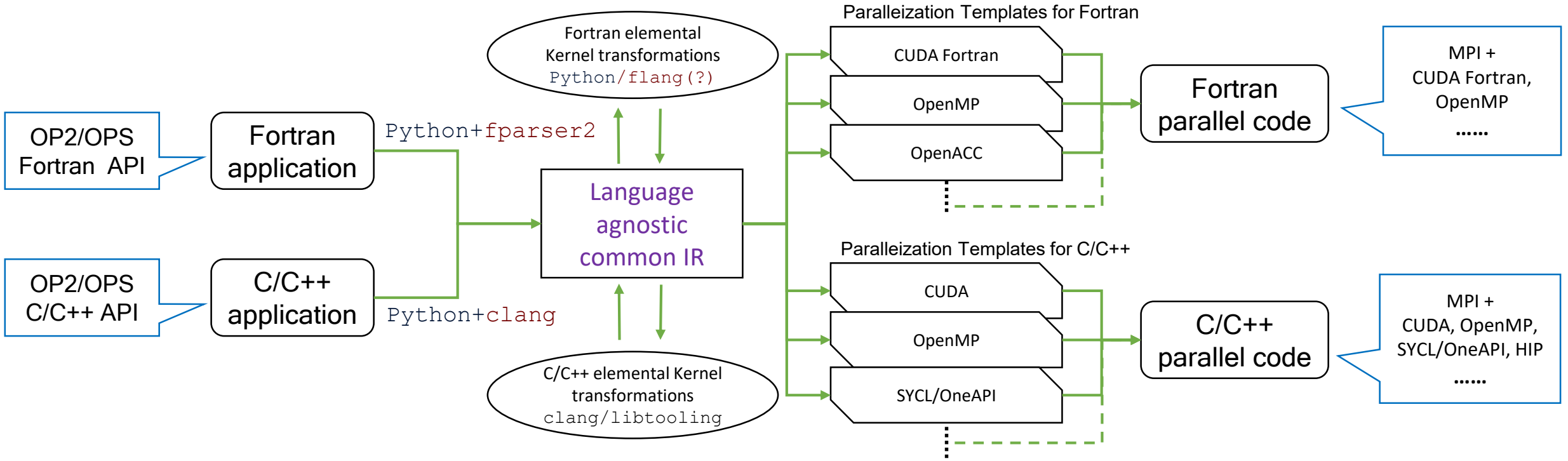



```
! Elemental kernel
subroutine res_calc(x1,x2,q1,q2,adt1,adt2,res1,res2)
  IMPLICIT NONE
  REAL(kind=8), DIMENSION(2), INTENT(IN) :: x1
  REAL(kind=8), DIMENSION(2), INTENT(IN) :: x2
  ...
  REAL(kind=8) :: dx,dy,mu,ri,p1,vol1,p2,vol2,f
  dx = x1(1) - x2(1)
  dy = x1(2) - x2(2)
  ...
  f = 0.5 * (vol1 * q1(1) + vol2 * q2(1)) + &
  &    mu * (q1(1) - q2(1))
  res1(1) = res1(1) + f
  res2(1) = res2(1) - f
  ...
end subroutine
! Calculate flux residual - parallel loop over edges
call op_par_loop_8 (res_calc, edges, &
& op_arg_dat(x,   1, edge,  2,"real(8)", OP_READ), &
& op_arg_dat(x,   2, edge,  2,"real(8)", OP_READ), &
& op_arg_dat(q,   1, ecell, 4,"real(8)", OP_READ), &
& op_arg_dat(q,   2, ecell, 4,"real(8)", OP_READ), &
& op_arg_dat(adt, 1, ecell, 1,"real(8)", OP_READ), &
& op_arg_dat(adt, 2, ecell, 1,"real(8)", OP_READ), &
& op_arg_dat(res, 1, ecell, 4,"real(8)", OP_INC ), &
& op_arg_dat(res, 2, ecell, 4,"real(8)", OP_INC ))
```

Evolutionary Application Development



OP2/OPS Code Generation

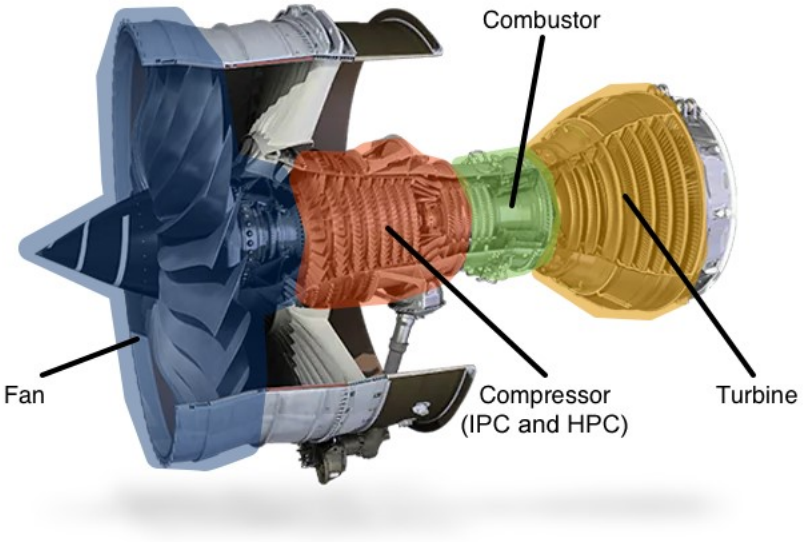


- Simplest Code generation / translation
  - Intermediate representation is simply the loop descriptions + elemental kernels
  - Generated parallel code can be viewed and understood by a human !
- Multi-layered – no opaque / black box layers
- Build with well supported / long-term technologies - Python, Clang/libtooling, [flang, mlir]

EPSRC Prosperity Partnership – ASIMoV [Hydra to OP2-Hydra]

- Virtual certification of Gas Turbine Engines
  - Main consortium with partners – Rolls-Royce plc., EPCC, Warwick, Oxford, Cambridge and Bristol

- Grand Challenge 1 – Sliding Planes model of Rig250 (DLR test rig compressor)
  - 4.5 stage rotor-stator (10-row full annulus) | 4.58B mesh nodes.
  - Need to obtain 1 revolution of compressor in less than 24 hours
  - Current production estimates at over 7 days



- Setup
  - Moving rotor-stator – sliding planes interfaces
  - Rotors and Stators modelled with Hydra CFD suite – URANS (360-degree models)
  - 10 rotor-stator interfaces
  - Code coupling for sliding planes – move from current monolithic (Hydra only) production code to coupling

- Challenges
  - Performance portability – run both CPUs and GPUs by multiple vendors
  - Preserve production code's scientific code and structure – cannot re-write, MUST "evolve" not overhaul !
  - Convince users to adopt ! (Ongoing for nearly 10 years now)

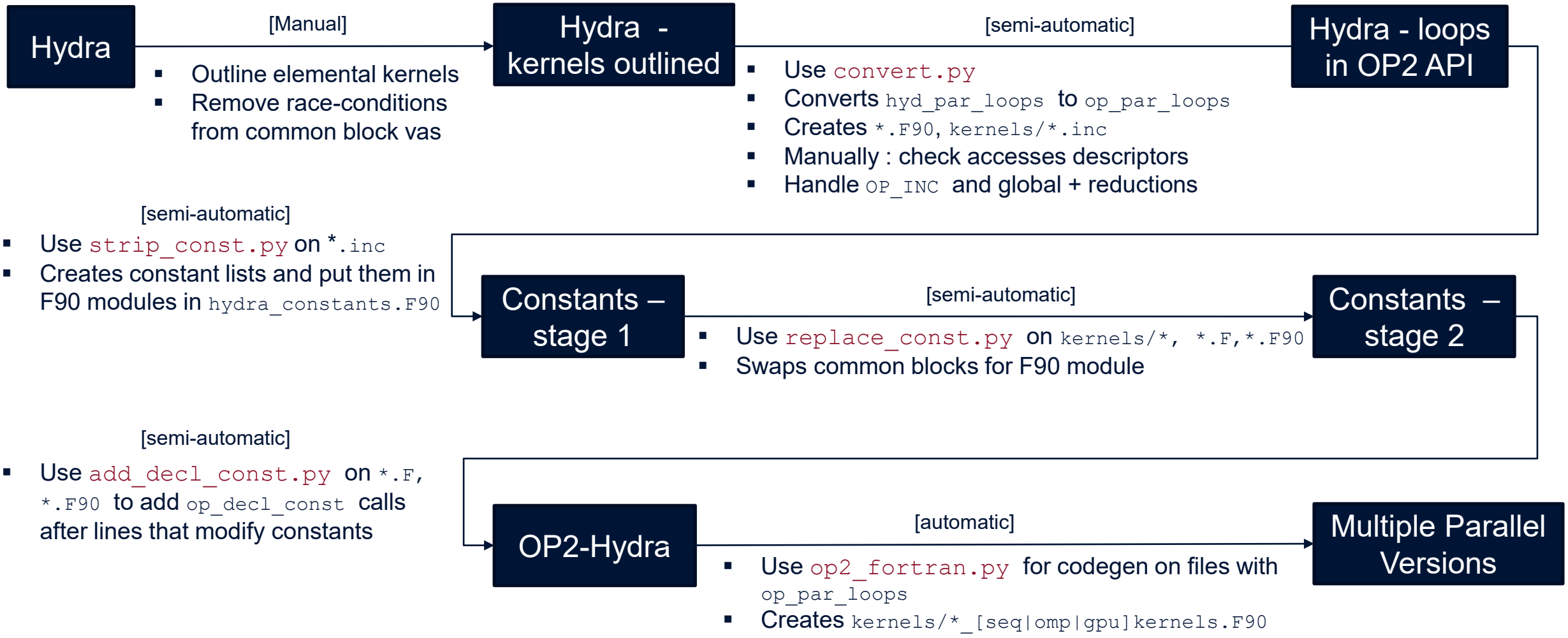
EVOLVING HYDRA TO OP2-HYDRA - PRELIMINARIES

- Declare problem using OP2 API [Manual]
  - Start with original code
  - Pass over Hydra data to OP2's **op_sets**, **op_maps** and **op_dats**
  - Needed to rely on in-house I/O libs to read mesh data – then hand these raw pointers to OP2
  - Still original loops accessing original blocks of memory– OP2 dev version does not move / **realloc** memory
  - Add OP2 header module + link with OP2 sequential developer backend : validate a small case
- Prepare loops to be with *outlined kernels* [Manual]

```
do while (hyd_par_loop (ncells, istart, iend))
 call hyd_access_r8('r',areac, 1, ncells, null,  0,0,1,1)
 call hyd_access_r8('r',arean, 1, nnodes, ncell, 1,1,1,3)
 do ic = istart, iend
   i1 = ncell (1,ic)
   i2 = ncell (2,ic)
   i3 = ncell (3,ic)
   area(i1) = arean(i1) + areac(ic)/3.0
   area(i2) = arean(i2) + areac(ic)/3.0
   area(i3) = arean(i3) + areac(ic)/3.0
 end do
end while
```

```
subroutine distr(areac,arean1,arean2,arean3)
  real(8), intent(in) :: areac
  real(8), intent(in) :: arean1, arean2, arean3
  arean1 = arean1 + areac/3.0
  arean2 = arean2 + areac/3.0
  arean3 = arean3 + areac/3.0
end subroutine

do while (hyd_par_loop (ncells, istart, iend))
  call hyd_access_r8('r',areac, 1, ncells, null,  0,0,1,1)
  call hyd_access_r8('r',arean, 1, nnodes, ncell, 1,1,1,3)
  do ic = istart, iend
    i1 = ncell (1,ic)
    i2 = ncell (2,ic)
    i3 = ncell (3,ic)
    call distr(areac(ic),arean(i1),arean(i2),arean(i3))
  end do
end while
```

Evolving Hydra to OP2-Hydra – Automating Loop conversion



Evolving Hydra to OP2-Hydra – Distributed Memory Challenges

❑ Using legacy / in-house I/O libraries

- Took a lot of time to get at the raw data read in from in-house I/O libraries – Hydra uses Cray pointers !
- Currently attempting to automate this process for any new version of Hydra

❑ Communications avoiding optimizations

- Partial MPI halo (PH) exchange – only exchange part of the MPI halo that has been modified
- Grouping of MPI halos (GH) – pack halos from different `op_dats` to single message | there is now a packing cost !
- GPU-side Gathers (GG) and scatter – gather/scatter data to be communicated to coupler using a GPU kernel, instead of on host

ARCHER2				
	$1-10_{430M}$		$1-10_{4.58B}$	
	10 nodes	27 nodes	107 nodes	283 nodes
Default	41.62	16.55	41.24	18.19
+PH	39.87	15.64	38.36	16.88

5 – 7 % gains

Cirrus			
	$1-10_{430M}$		$1-2_{653M}$
	15 nodes	20 nodes	17 nodes
Default	19.07	13.58	23.79
+GG +PH +GH	5.09	4.23	6.74

60 – 75 % gains

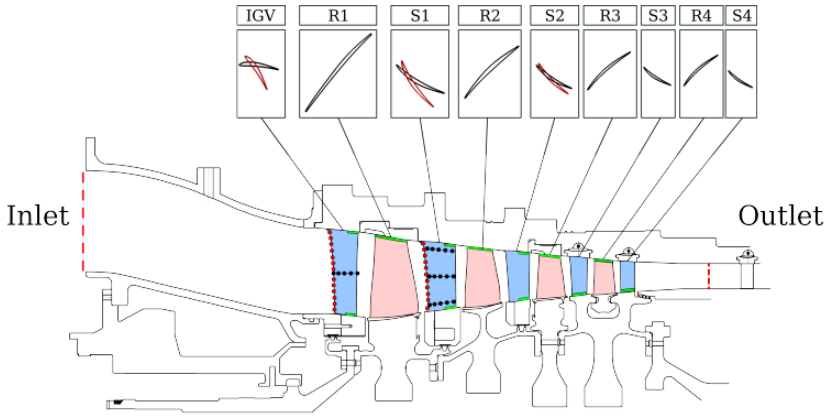
EVOLVING HYDRA TO OP2-HYDRA – GPU CHALLENGES

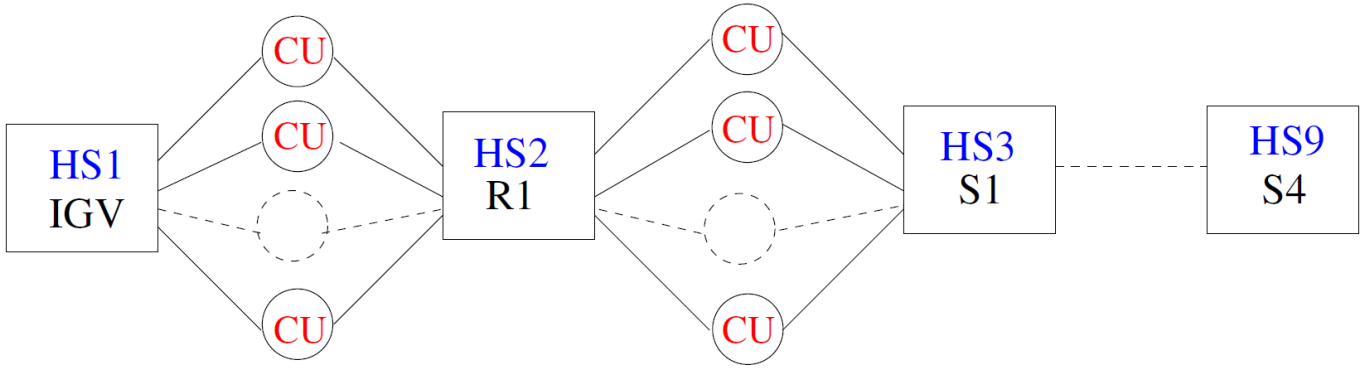
- Fortran 77 Common blocks declaring constants
  - Move to Fortran 90 – use modules
  - Declare constants with **`op_decl_const`** so that GPU code generation can add copies to device

- Elemental kernel calling external function
  - Need source of external functions - to code-gen SoA parameters for external function
  - No pointer aliasing

- Elemental kernel body
  - Exits in the kernel, `print` statements

EVOLVING HYDRA TO OP2-HYDRA – INTEGRATING WITH COUPLER

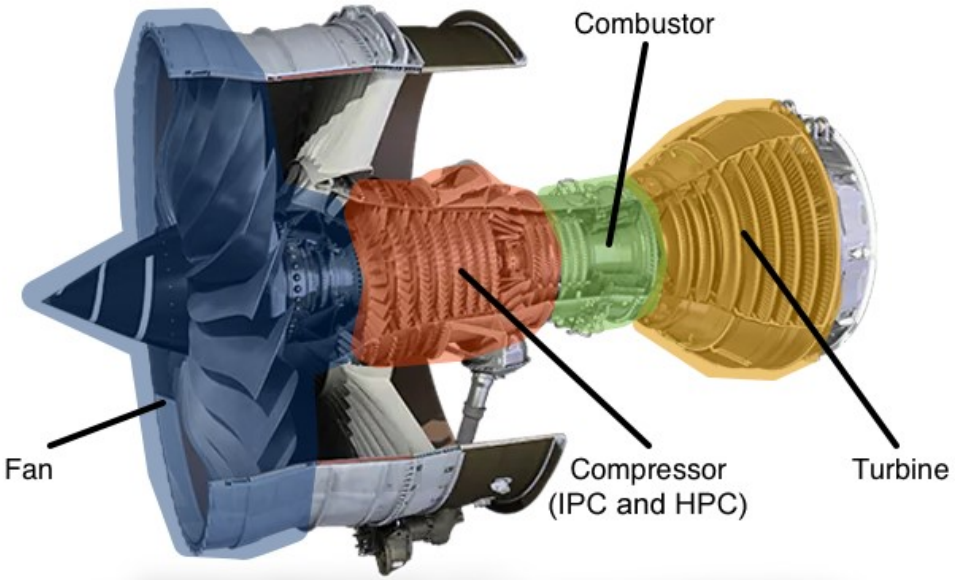




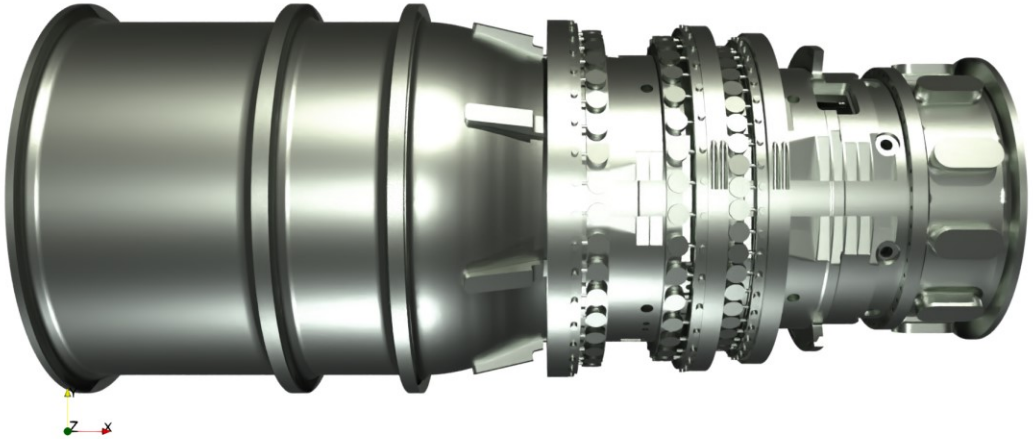
Reproduced with permission from : V. Marciniak, A. Weber, E. K¨uegler, Modelling transition for the design of modern axial turbomachines, in: 6th European Conference on Computational Fluid Dynamics, Barcelona, Spain, 2014, pp. 20–25

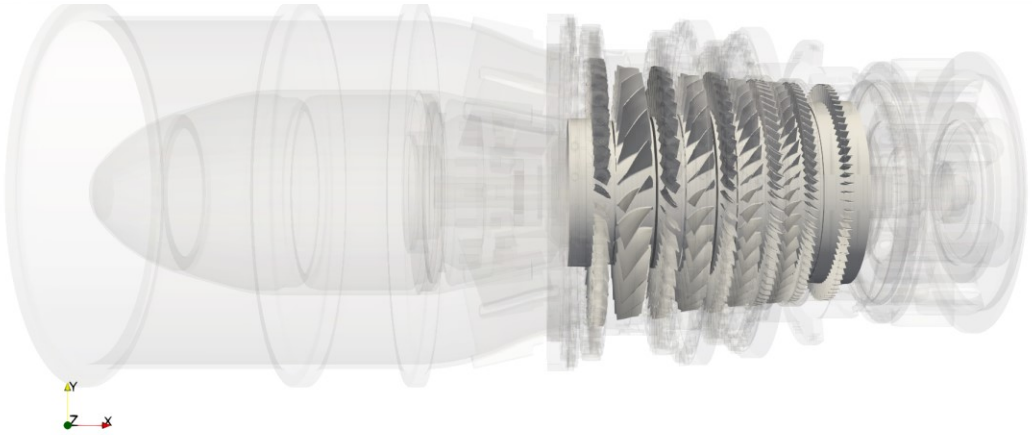
- Needed to interface with Rolls-Royce's in-house coupler - *JM76*
  - JM76 can couple arbitrary number of models – specialized treatment for fluid-solid and fluid-fluid interfaces
- For this problem it implements sliding-planes interfaces
  - Extrude mesh of Rotor and Stator to create overlapping mesh with adjacent zone
  - During execution – search and find correct source-donor element | interpolate flow vars from one zone | communicate to donor zone
- Key challenges
  - Give JM76 access to OP2 internal data – **`op_dat`**s and **`op_map`**s
  - Efficient communication between GPUs – GPU side gathers

Trent XWB Engine



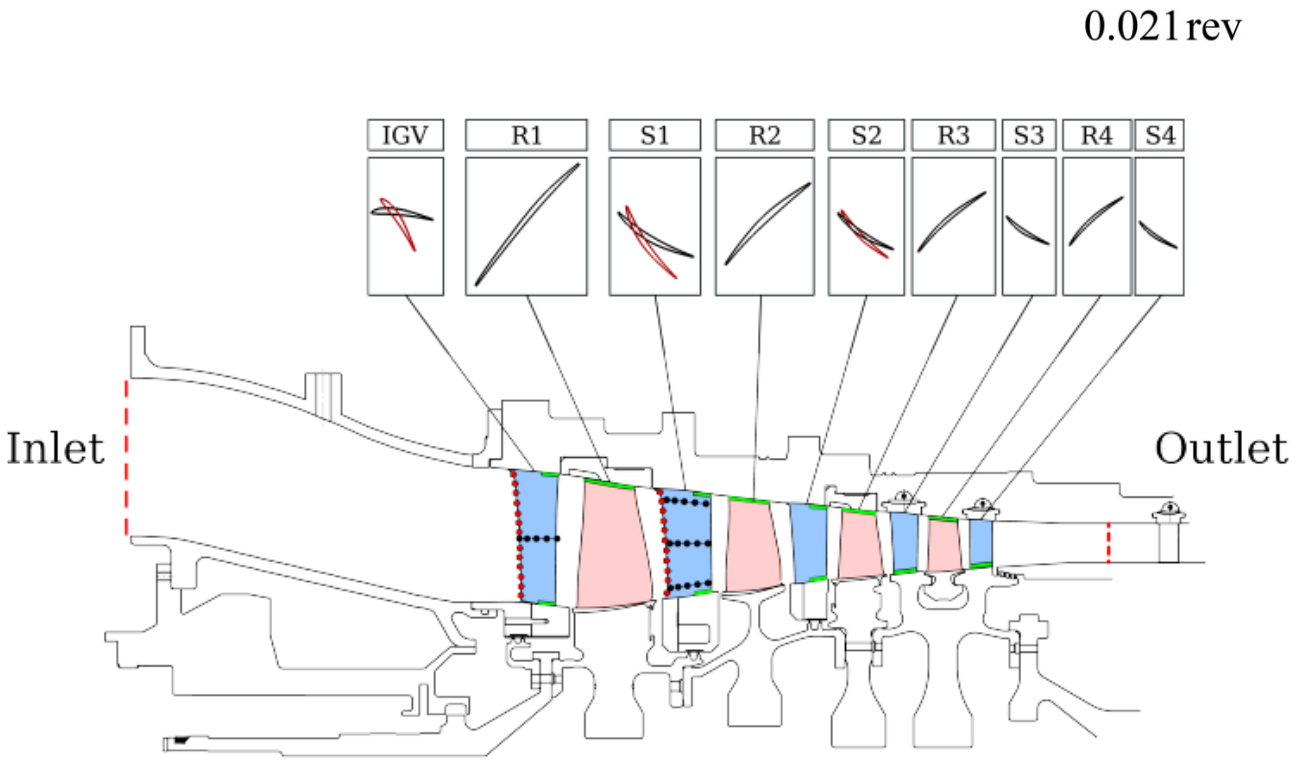
Rig250 Geometry and Engine



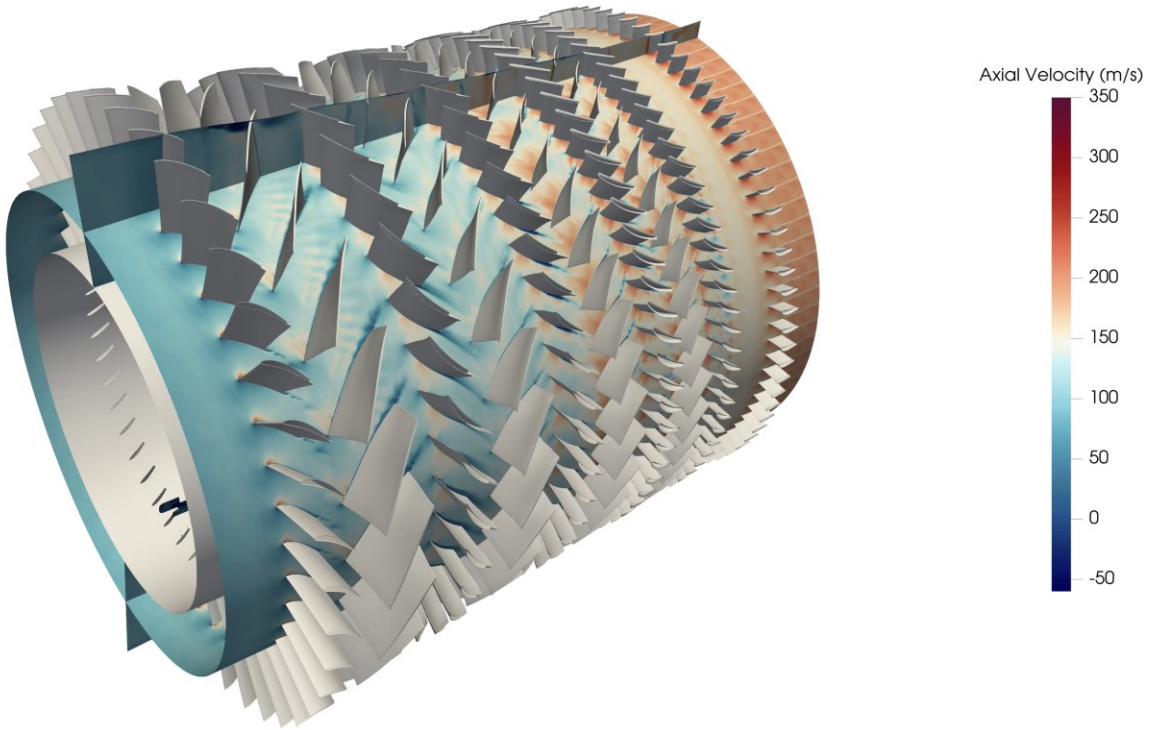


- Rig250 – 430M mesh – 10 rows [10_{430M}]
- Rig250 – 653M mesh – 2 rows [2_{653M}]
- Rig250 – 4.58B mesh – 10 rows [$10_{4.58B}$]

RIG250 – PROBLEM SETUP AND MESH



Reproduced with permission from : V. Marciniak, A. Weber, E. K¨uegler, Modelling transition for the design of modern axial turbomachines, in: 6th European Conference on Computational Fluid Dynamics, Barcelona, Spain, 2014, pp. 20–25



G.R. Mudalige, I.Z. Reguly, A. Prabhakar, D. Amirante, L. Lapworth, S.A. Jarvis, *Towards Virtual Certification of Gas Turbine Engines With Performance-Portable Simulations.* In 2022 **IEEE International Conference on Cluster Computing (CLUSTER)**, 2022

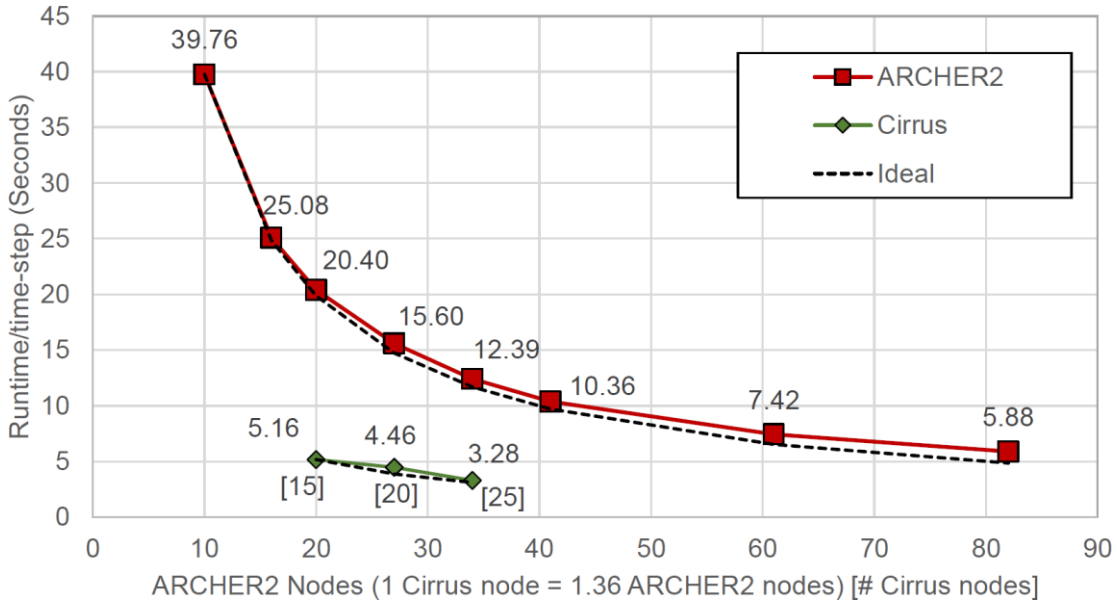
OP2-HYDRA PERFORMANCE - SYSTEMS

TABLE I: Systems specifications.

System	ARCHER2 HPE Cray EX	Cirrus SGI/HPE 8600 GPU Cluster
Processor	AMD EPYC 7742 @ 2.25 GHz	Intel Xeon Gold 6248 (Cascade Lake) @ 2.5 GHz + NVIDIA Tesla V100-SXM2-16GB GPU
(procs×cores) /node	2×64	2×20 + 4×GPUs
Memory/node	256 GB	384 GB + 40GB/GPU
Interconnect	HPE Cray Slingshot 2×100 Gb/s bi-directional/node	Infiniband FDR, 54.5 Gb/s
OS	HPE Cray LE (based on SLES 15)	Linux CentOS 7
Compilers	GNU 10.2.0	nvfortran (nvhpc 21.2)
Compiler Flags	`-O2 -eF -fPIC`	CUDA 11.6 and sm_70 -O2 -Kieee
Power/node	660W	≈ 900W

OP2-Hydra Scaling Performance

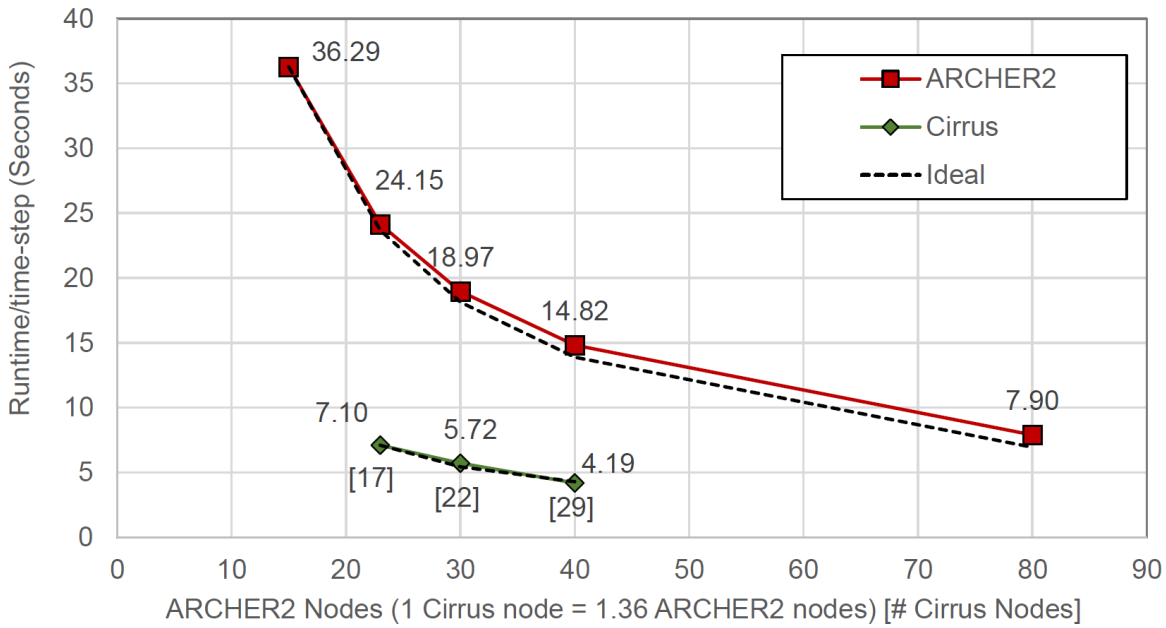
Rig250 – 430M mesh, 10 rows



- ARCHER2 @ 34 nodes
  - 94% parallel efficiency
  - 10% coupling overhead
- Cirrus @ 25 nodes
  - 94% parallel efficiency
  - 20% coupling overhead
- ARCHER2 @ 82 nodes
  - 82% parallel efficiency
  - 20% coupling overhead

3.7- 4x speedup

Rig250 – 653M mesh, 2 rows



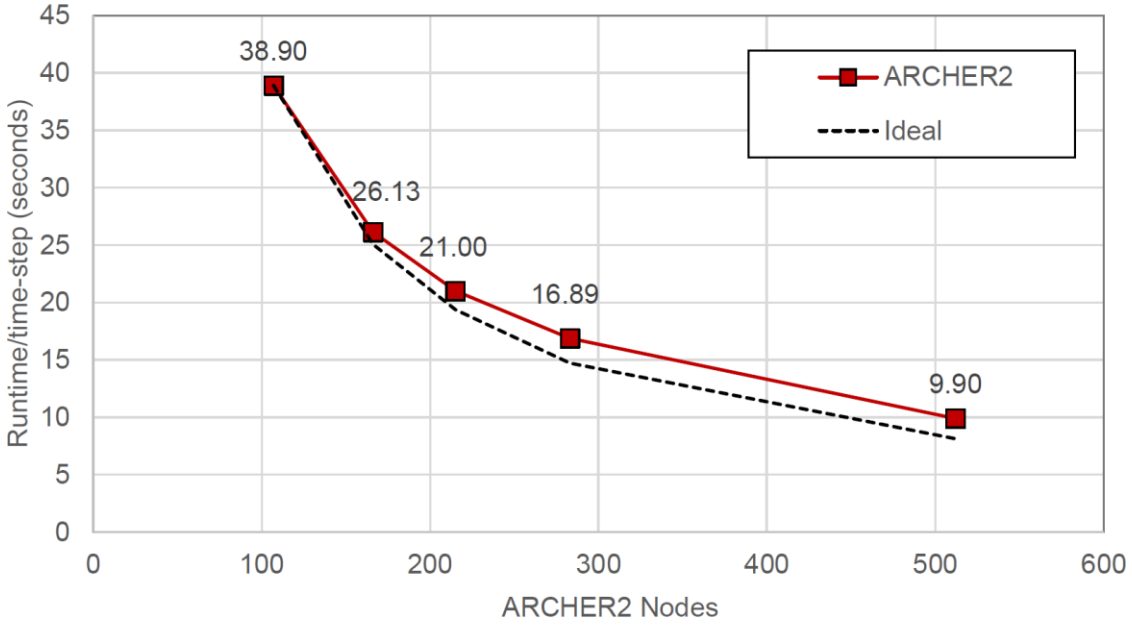
- ARCHER2 @ 80 nodes
  - 88% parallel efficiency
  - 8% coupling overhead
- Cirrus @ 29 nodes
  - 98% parallel efficiency
  - 12% coupling overhead

3.3 - 3.4x speedup

G.R. Mudalige, I.Z. Reguly, A. Prabhakar, D. Amirante, L. Lapworth, S.A. Jarvis, *Towards Virtual Certification of Gas Turbine Engines With Performance-Portable Simulations.* In 2022 **IEEE International Conference on Cluster Computing (CLUSTER)**, 2022

OP2-HYDRA SCALING PERFORMANCE - RIG250 4.58B MESH

Runtime per time-step (seconds)



- ARCHER2 @ 512 nodes:
  - 82% parallel efficiency (vs 107 node run)
  - 15% coupling overhead

Time to solution (hours) – Achieved (A), *Projected* (*P*)

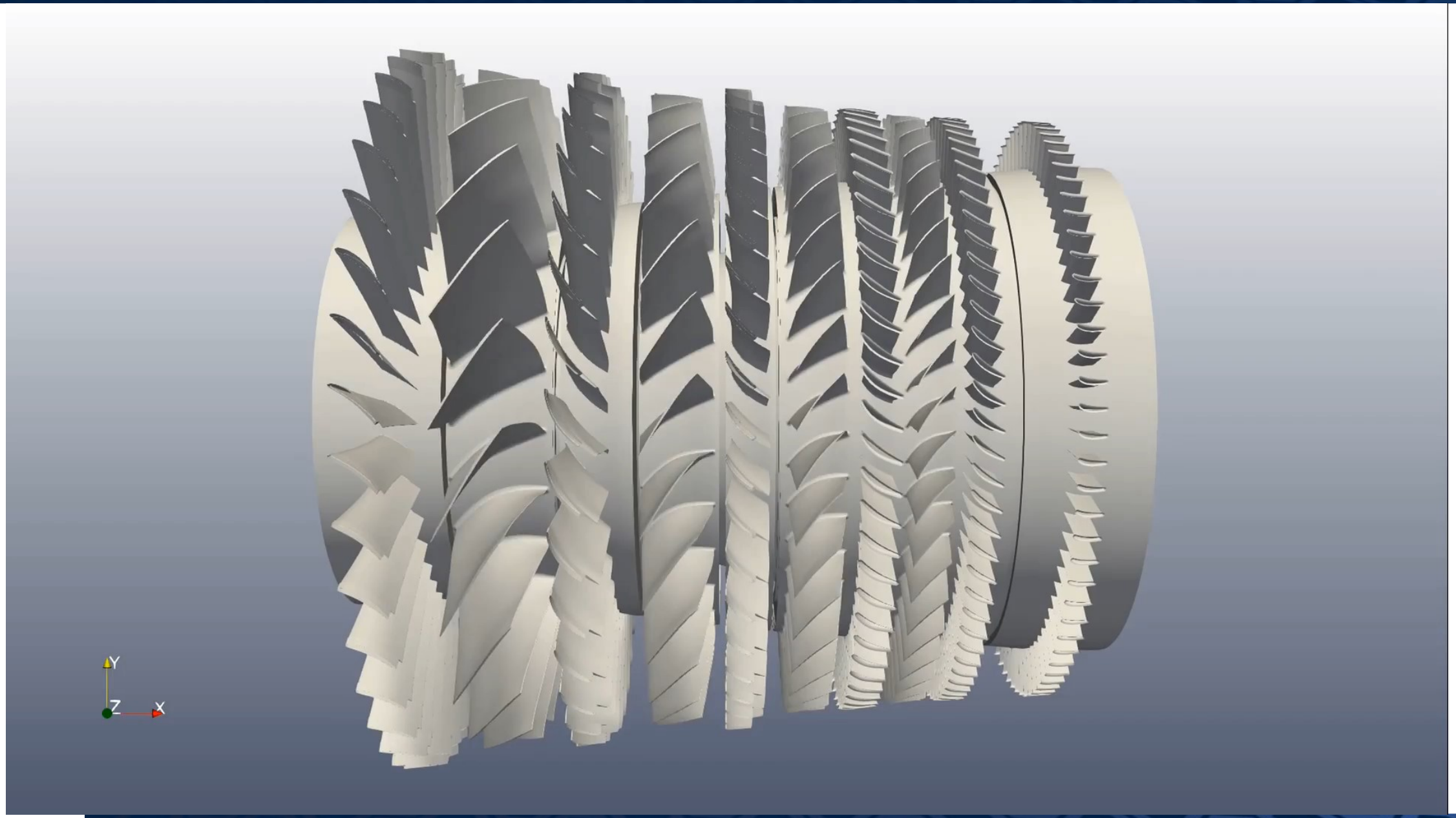
1 revolution of Rig250

Rig250 Problem	ARCHER2		Cirrus	
	Runime	#nodes	Runtime	#nodes
$1-10_{430M}$ - Monolithic	*93.0 (P)*	*8*		
$1-10_{430M}$ - Coupled	*85.0 (P)*	*8*	*2.9 (P)*	*15*
$1-10_{430M}$ - Coupled	*3.3 (P)*	*80*	*1.8 (P)*	*25*
$1-2_{653M}$ - Monolithic	*110.0 (P)*	*8*		
$1-2_{653M}$ - Coupled	*40.0 (P)*	*8*	*3.9 (P)*	*17*
$1-2_{653M}$ - Coupled	*8.2 (P)*	*40*	*3.2 (P)*	*22*
$1-10_{4.58B}$ - Coupled	*14.5 (A)*	*166*	*4.7 (P)*	*122*
$1-10_{4.58B}$ - Coupled	*9.4 (A)*	*256*		
$1-10_{4.58B}$ - Coupled	*5.5 (A)*	*512*		

- 122 Cirrus nodes is power equivalent to 166 ARCHER2 nodes
- ARCHER2 needs just over 3x more number of *power equivalent* nodes (512) to match Cirrus's runtime (4.7 hours)

G.R. Mudalige, I.Z. Reguly, A. Prabhakar, D. Amirante, L. Lapworth, S.A. Jarvis, *Towards Virtual Certification of Gas Turbine Engines With Performance-Portable Simulations.* In 2022 **IEEE International Conference on Cluster Computing (CLUSTER)**, 2022

RIG250 – FLOW SOLUTION



Other Projects – OpenSBLI (Uni. Of Southampton)

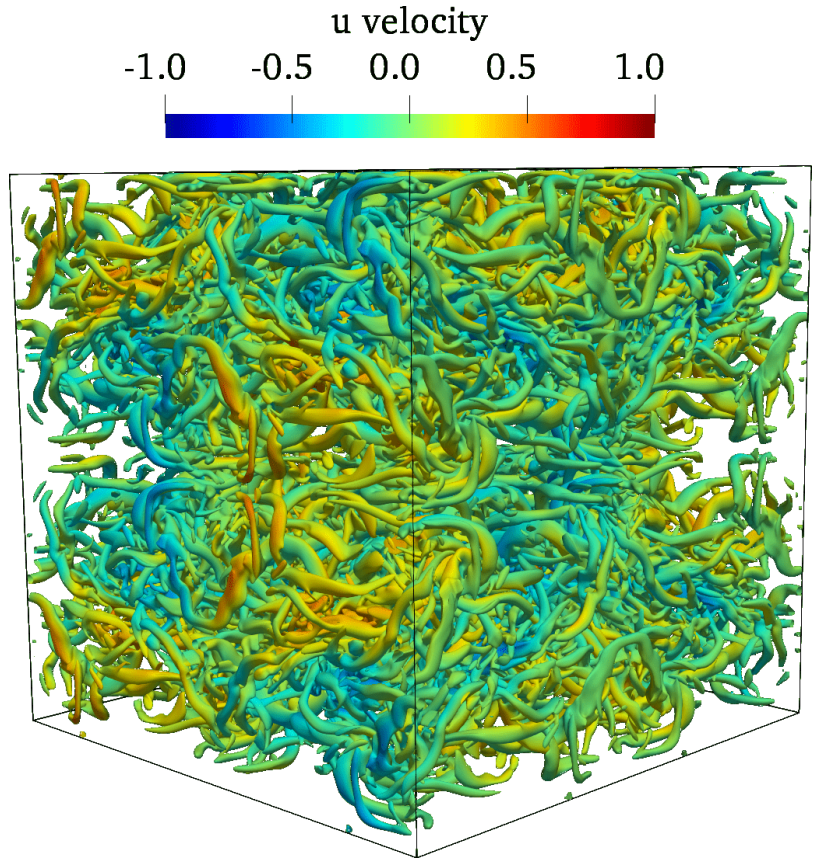
- Compressible Navier-Stokes solver
  - With shock capturing WENO/TENO
  - 4th order Finite Difference
  - Single/double precision

- OpenSBLI is a Python framework
  - Write equations in SymPy expressions
  - OPS code generated

```
ndim = 3
sc1 = "**{\'scheme\':\'Teno\'}"
# Define the compresible Navier-Stokes equations in Einstein notation.
mass = "Eq(Der(rho,t), - Conservative(rhou_j,x_j,%s))" % sc1
momentum = "Eq(Der(rhou_i,t) , -Conservative(rhou_i*u_j + KD(_i,_j)*p,x_j , %s) + Der(tau_i_j,x_j) )" % sc1
energy = "Eq(Der(rhoE,t), - Conservative((p+rhoE)*u_j,x_j, %s) - Der(q_j,x_j) + Der(u_i*tau_i_j ,x_j) )" % sc1
stress_tensor = "Eq(tau_i_j, (mu/Re)*(Der(u_i,x_j)+ Der(u_j,x_i) - (2/3)* KD(_i,_j)* Der(u_k,x_k)))"
heat_flux = "Eq(q_j, (-mu/((gama-1)*Minf*Minf*Pr*Re))*Der(T,x_j))"
# Numerical scheme selection
Avg = RoeAverage([0, 1])
LLF = LLFTeno(teno_order, averaging=Avg)
cent = Central(4)
rk = RungeKuttaLS(3, formulation='SSP')
# Specifying boundary conditions
boundaries[direction][side] = IsothermalWallBC(direction, 0, wall_eqns)
# Generate a C code
alg = TraditionalAlgorithmRK(block)
OPSC(alg)
```

OpenSBLI

https://opensbli.github.io/

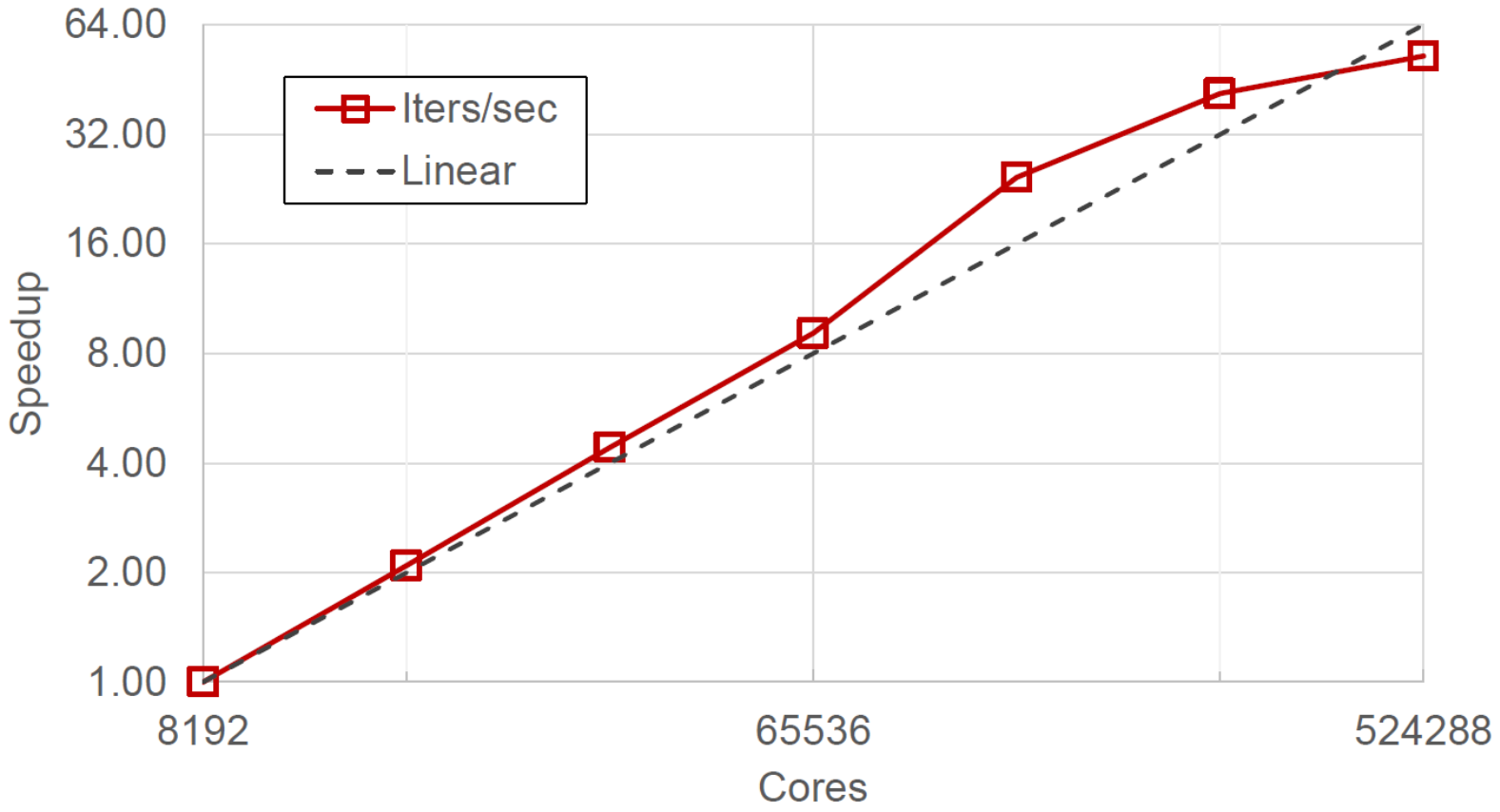


Jacobs, C. T., Jammy, S. P., Sandham N. D. (2017). OpenSBLI: A framework for the automated derivation and parallel execution of finite difference solvers on a range of computer architectures. *Journal of Computational Science*, 18:12-23, DOI: 10.1016/j.jocs.2016.11.001

OpenSBLI on ARHCER2

- Taylor – Green Vortex Problem – ARCHER2 benchmark
  - Strong Scaling - 10243 Mesh
  - Double precision
  - Speedup calculated from 1000 iterations – includes start up time.

From recent benchmarking runs done by Andrew Turner and the ExCALIBUR Benchmarking team (Oct 2021)



ExCALIBUR Project - Turbulent Flow Simulations at the Exascale

- ExCALIBUR Phase 1B – Turbulence at the Exascale
  - Imperial, Warwick, Newcastle, Southampton, Cambridge, STFC collaboration | UKTC and UKCTRF Communities
  - Xcompact3D and Wind Energy, OpenSBLI and Green Aviation, uDALES and Air Quality, SENGA+ and Net-Zero Combustion
  - Extending OPS capability – robust code-gen tools and parallel transformations | support future-proof code development
  - UQ, I/O, Coupling and Visualization
  - Machine Learning Algorithms for Turbulent Flow

- CCP – Turbulence
  - Direct solver libraries – Tri-, penta-, 7-, 9-, 11 diagonal, multi-dimensional solvers
  - Integrate direct-solver libraries to be called within OPS
  - OpenSBLI type high-level (Python) framework for XCompact3D – High Order FD framework

Challenges – Cost / Effort of Conversion

- Converting legacy code is time consuming
  - Large code base,
  - Defunct 3rd party libs,
  - Fortran 77 or older !
- Difficult to validate code
  - New code giving the same accurate scientific output ?
  - What code should I certify ? High-level code/generated code ?
  - Difficult to convince users to use new code - fear of an opaque compiler / intermediate representation / black box !
- Incremental conversion – loop by loop
  - Simpler than CUDA, but more difficult than OpenACC/OpenMP
  - Automated conversion ?
- Changing user requirements
  - Wanting to use a DSL for doing things beyond what it was intended for !
  - Asking for “back-doors” / “escape hatches” -- leads to poor performance

Challenges – Cost / Effort of Conversion

- Currently purely done via academic and (small/short term) industrial funding

- Long term funding and maintenance
  - Once established probably will not be different to any other classical library
  - Will require compiler expertise to maintain code generation tools

- What DSL to choose ?
  - Re-use technologies / DSLs – especially code-gen tools (best not to reinvent !)

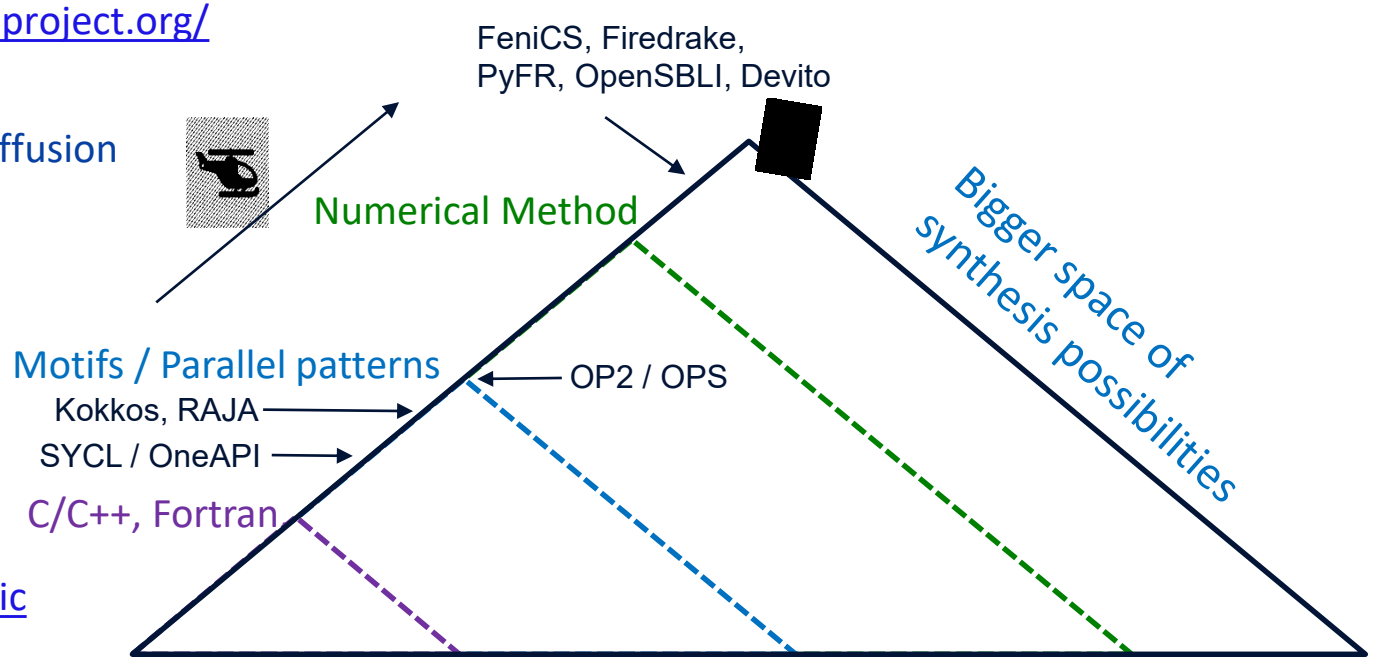
- Skills Gap
  - Program in C/C++/Fortran (at a minimum)
  - Knowledge of compilers / code-generation
  - Compete for applicants – Communicate what we do better | impact of HPC / Computational Sciences
  - [In the UK] Salary
  - [In the UK] Contracts

DSLs / High-Level Abstractions Gaining Traction

- FEniCS - PDE solver package - https://fenicsproject.org/
- Firedrake - automated system for the portable solution of PDEs using the finite element method https://www.firedrakeproject.org/

- PyFR - Python based framework for solving advection-diffusion type problems on streaming architectures using the Flux Reconstruction approach - http://www.pyfr.org/

- Devito - prototype DSL and code generation framework based on SymPy for the design of highly optimised finite difference kernels for use in inversion methods - http://www.opesci.org/devito-public



Adapted from: *Synthesis versus Analysis: What Do We Actually Gain from Domain-Specificity?* Keynote talk at the LCPC 2015. Paul H. J. Kelly (Imperial College London)

- GungHO/PSyclone project - Weather modelling codes (MetOffice)
- STELLA – DSL for stencil codes, for solving PDEs (Metro Swiss)

- Kokkos – C++ template library – SNL
- RAJA - C++ template libraries - LLNL

Downloads and More Information

- GitHub Repositories
  - OP2 – https://github.com/OP-DSL/OP2-Common
  - OPS – https://github.com/OP-DSL/OPS
- OP-DSL Webpage - https://op-dsl.github.io/
- Contact
  - Gihan Mudalige (Warwick) - g.mudalige@warwick.ac.uk
  - Istvan Reguly (PPCU – Hungary) - reguly.istvan@itk.ppke.hu

ACKNOWLEDGEMENTS

- OP2 was part-funded by the UK Technology Strategy Board and Rolls-Royce plc. through the SILOET project, and the UK EPSRC projects EP/I006079/1, EP/I00677X/1 on Multi-layered Abstractions for PDEs.

- OPS was part-funded by the UK Engineering and Physical Sciences Research Council projects EP/K038494/1, EP/K038486/1, EP/K038451/1 and EP/K038567/1 on "Future-proof massively-parallel execution of multi-block applications" and EP/J010553/1 "Software for Emerging Architectures" (ASEArch) project.

- This research is supported by Rolls-Royce plc., and by the UK EPSRC (EP/S005072/1) Strategic Partnership in ComputationalScience for Advanced Simulation and Modelling of Engineering Systems (ASiMoV).

- Gihan Mudalige was supported by the Royal Society Industrial Fellowship Scheme (INF/R1/180012)
- Research was part-supported by the Janos Bolyai Research Scholarship of the Hungarian Academy of Sciences.

- The research has been carried out within the project Thematic Research Cooperation Establishing Innovative Informatic and Info-communication Solutions, which has been supported by the European Union and co-financed by the European Social Fund under grant number EFOP-3.6.2-16-2017-00013.

- OpenSBLI was part-funded by EPSRC grants EP/K038567/1 and EP/L000261/1, and European Commission H2020 grant 671571 "ExaFLOW: Enabling Exascale Fluid Dynamics Simulations