Separation of Concerns for Performance Portability and the wider context of the UK ExCALIBUR Programme

Royal Society Industry Fellow
Reader in High Performance Computing
Department of Computer Science, University of Warwick
g.mudalige@warwick.ac.uk

Joint work with:

Istvan Reguly @ PPCU

Kamalavasan Kamalakannan, Arun Prabhakar, Archie Powell and others at the HPSC group @ Warwick

Neil Sandham and team @ Southampton, Dario Amirante @ Surrey

Mike Giles @ Oxford, Sylvain Laizet, Paul Kelly and many more @ Imperial College London

Rolls-Royce plc., NAG, UCL, STFC, IBM and many more.

WARWICK
THE UNIVERSITY OF WARWICK

- Traditional CPUs
  - Intel, AMD, ARM, IBM
  - multi-core (> 20 currently)
  - Deep memory hierarchy (cache levels and RAM)
  - longer vector units (e.g. AVX-512)
- GPUs
  - NVIDIA (A100), AMD (MI200) , Intel (Xe GPUs)
  - Many-core (> 1024 simpler SIMT cores)
  - CUDA cores, Tensor cores
  - Cache, Shared memory, HBM (3D stacked DRAM)
- Heterogeneous Processors
  - Different core architectures over the past few years
  - ARM *big.LITTLE*
  - NVIDIA *Grace.Hopper*
- XeonPhi (discontinued)
  - Many-core – based on simpler x86 cores
  - MCDRAM (3D stacked DRAM)
- FPGAs
  - Xilinx (AMD) and Intel
  - Various configurations
  - Low-level language / HLS tools for programming
  - Significant energy savings
- DSP Processors
  - Matrix 2000+ (MTP) DSP accelerator
  - [Yet to be announced Chinese Exascale system ?]
- TPUs (e.g. from Google), IPUs …

… Custom ASICs driven by AI ... in the cloud.

- Domain specific Hardware …
- 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) ….

- Open standards (e.g OpenMP, SYCL)
  - So far have not been agile to catch up with changing architectures
- Proprietary models (e.g. CUDA, OpenACC, ROCm, OneAPI)
  - Restricted to narrow vendor specific hardware
- 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 !

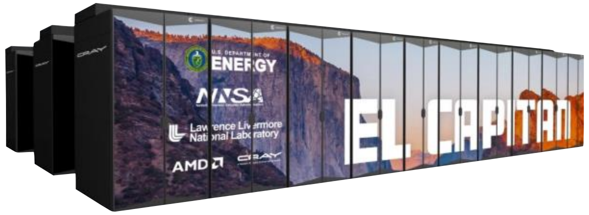
❑ What would an Exa-scale machine architecturally look like ?

- Perlmutter - Over 100 PFLOP/s - AMD EPYC CPUs (Milan) with NVIDIA A100 GPUs
- Aurora - 1 EFLOP Intel Xeon CPUs (Sapphire Rapids) with Intel Xe GPUs
- Frontier - 1.5 EFLOP/s AMD EPYC CPUs (Milan) with AMD Instinct GPUs
- El Capitan - 2 EFLOP/s AMD EPYC CPUs (Genoa) with AMD Instinct GPUs

---

- LUMI - 0.5 EFLOP/s AMD EPYC CPUs with AMD Instinct GPUs
- LEONARDO - 0.3 EFLOP/s - Intel Xeon CPUs (Sapphire Rapids) with NVIDIA A100 GPUs
- MareNostrum5 - 2 distinct 100+ PFLOP/s systems possibly based on ARM/RISC-V

- ARCHER2- 28 PFLOP/s AMD EPYC CPUs (Rome)
- Many Tier-2 systems in the UK - Isambard-2 – ARM A64FX | Baskerville - NVIDIA A100 GPUs

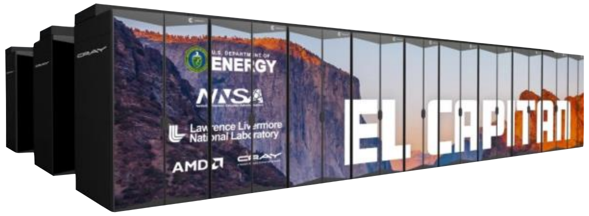


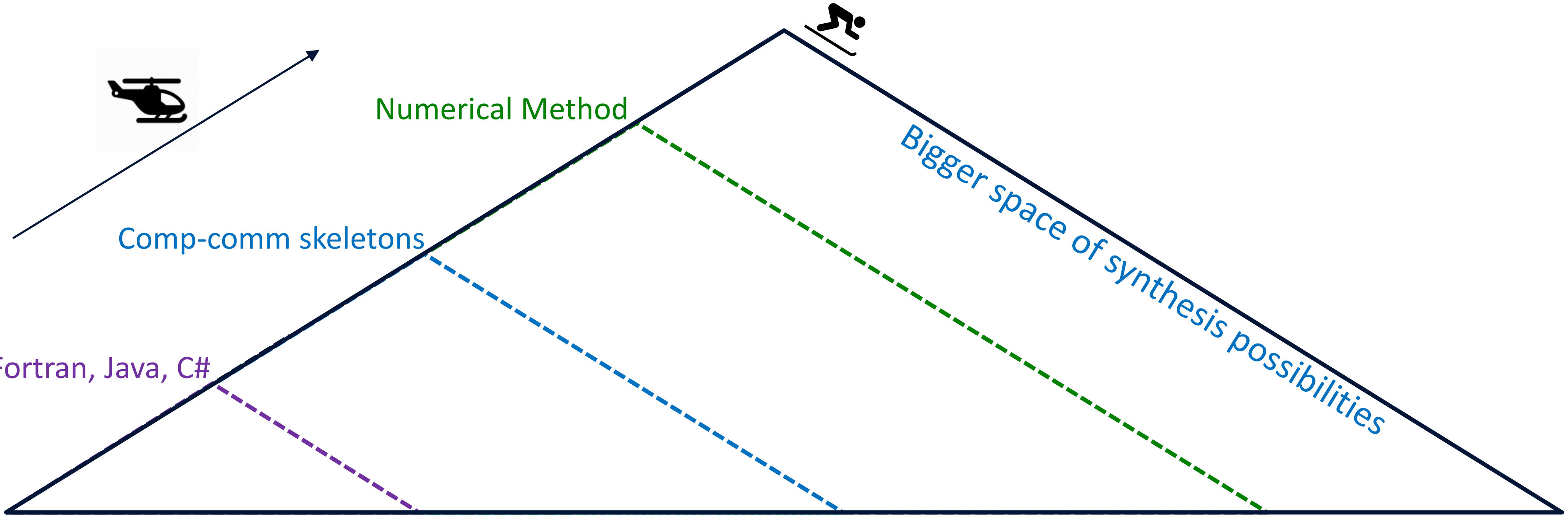
SOFTWARE CHALLENGE – A MOVING TARGET

- ❑ Each new platform requires new performance tuning effort
  - Deeper memory/cache hierarchies and/or shared-memory (including non-coherent)
  - Multiple (heterogeneous) memory spaces (device memory/host memory/near-chip memory)
  - Complex programming skills set needed to extract best performance on the newest architectures

- ❑ Not clear which architectural approach is likely to win in the long-term
  - Cannot be re-coding applications for each new type of architecture or parallel system
  - Nearly impossible for re-writing legacy codes

- ❑ Need to <u>future-proof</u> applications for their continued performance and portability
  - If not – significant loss of investment
  - Applications will not be able to make use of emerging architectures



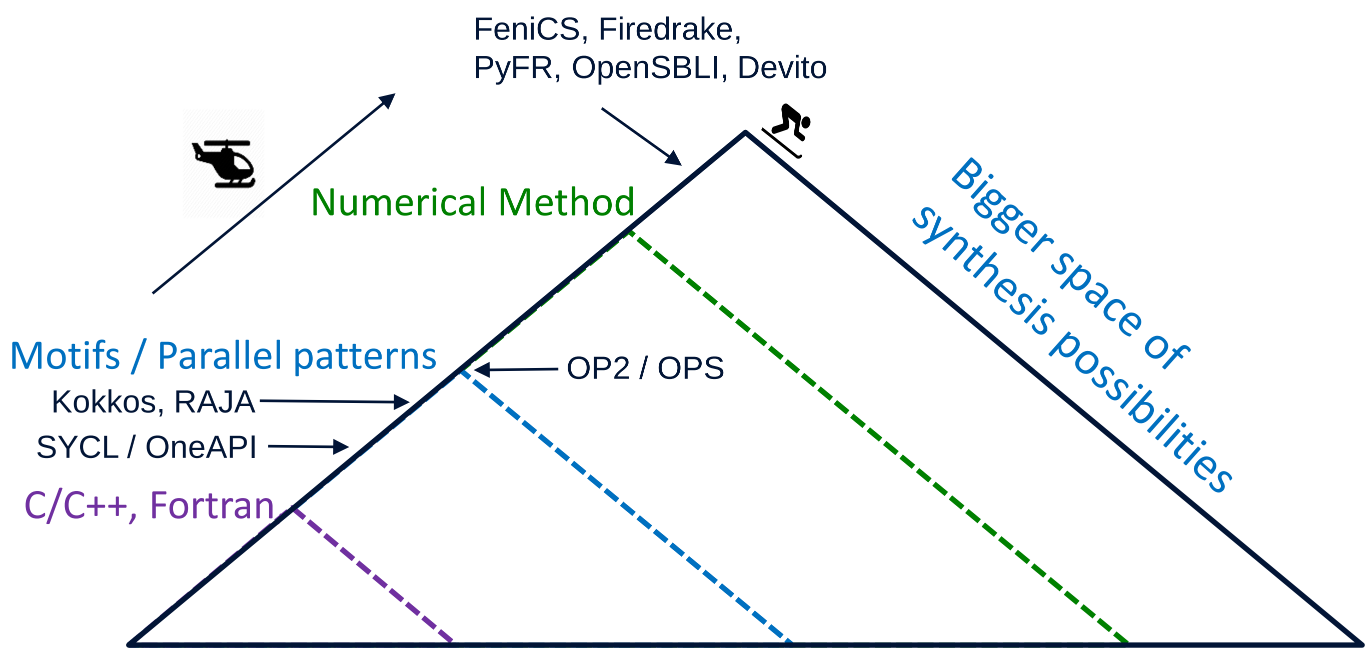


Adapted from: *Synthesis versus Analysis: What Do We Actually Gain from Domain-Specificity?* Keynote talk at the LCPC 2015. Paul H. J. Kelly (Imperial College London)

- Rise the abstraction to a specific domain of variability
- Concentrate on a narrower range (class) of computations
  - Computation-Communications skeletons - Structured-mesh, Unstructured-mesh, … 7 Dwarfs [Colella 2004] ?
  - (higher) Numerical Method - PDEs, FFTs, Monte Carlo …
  - (even higher) Specify application requirements, leaving implementation to select radically different solution approaches

DSLs / High-level Abstractions gaining traction

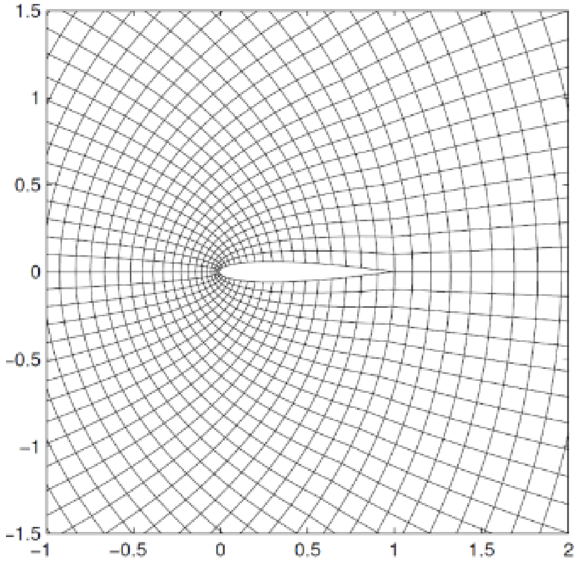
- FEniCS - PDE solver package - https://fenicsproject.org/
- Firedrake - automated system for the portable solution of PDEs using the finite element method https://www.firedrakeproject.org/
- PyFR - Python based framework for solving advection-diffusion type problems on streaming architectures using 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
- GungHO project - Weather modelling codes (MetOffice)
- STELLA – DSL for stencil codes(Metro Swiss)
- Liszt – Stanford University : DSL for solving mesh-based PDEs - http://graphics.stanford.edu/hackliszt/
- Kokkos – C++ template library – SNL
- RAJA - C++ template libraries - LLNL

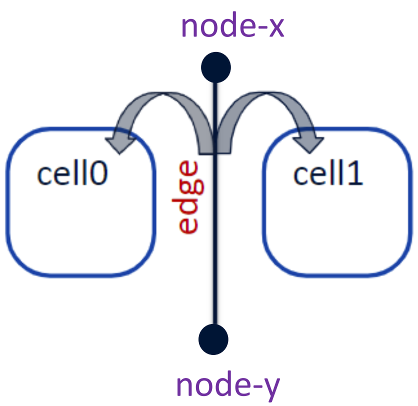


Adapted from: *Synthesis versus Analysis: What Do We Actually Gain from Domain-Specificity?* Keynote talk at the LCPC 2015. Paul H. J. Kelly (Imperial College London)

OP2 – Unstructured-mesh Applications Domain

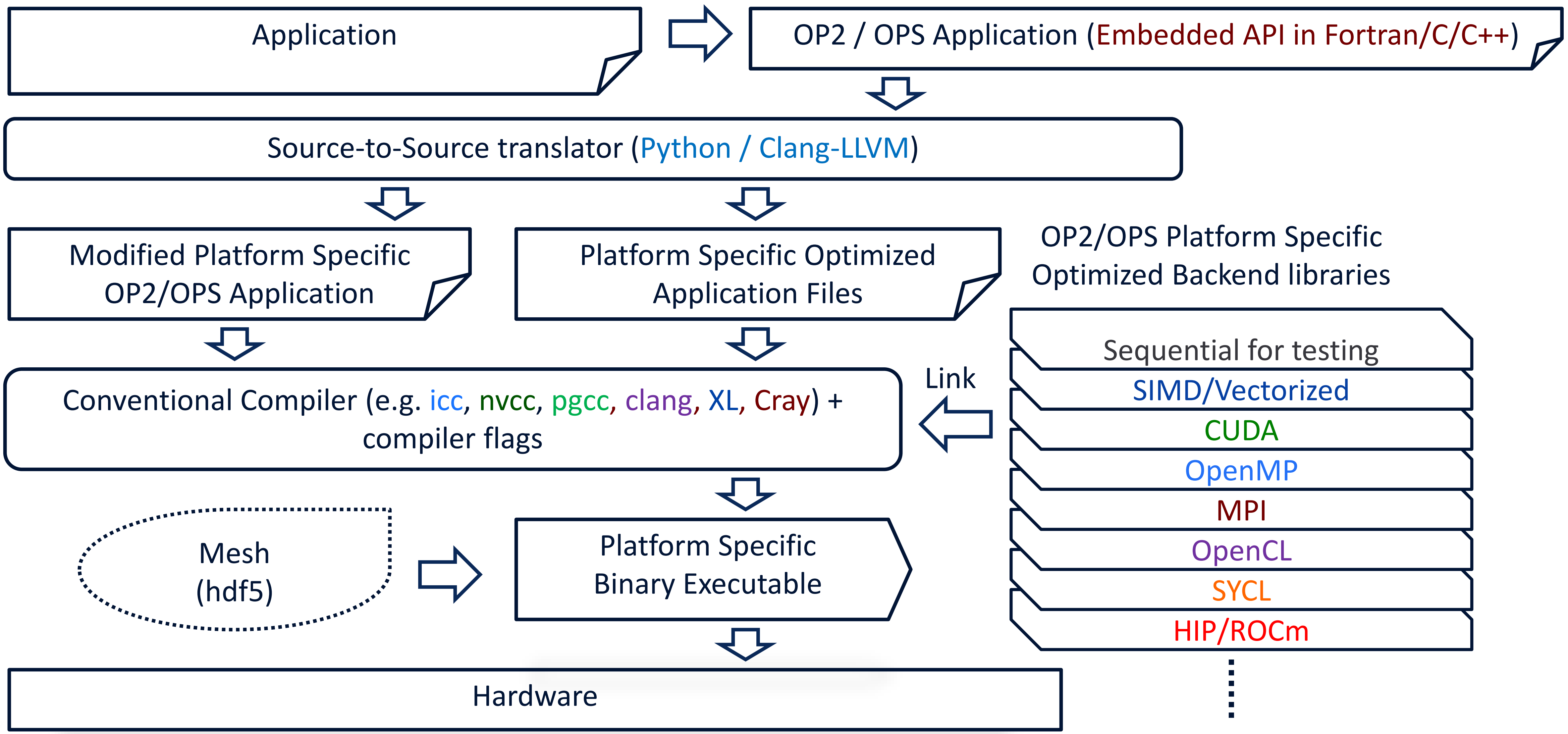
```
! Declaring the mesh with OP2
! sets
call op_decl_set(nnode,nodes,'nodes')
call op_decl_set(nedge,edges,'edges')
call op_decl_set(ncell,cells,'cells')
! maps
call op_decl_map(edges,nodes,2,edge ,pedge ,'pedge' )
call op_decl_map(edges,cells,2,ecell,pecell,'pecell')
! data
call op_decl_dat(nodes,2,'real(8)',x,p_x,'p_x')
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
  ...
! 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 ))
```

OP2 – Application development workflow



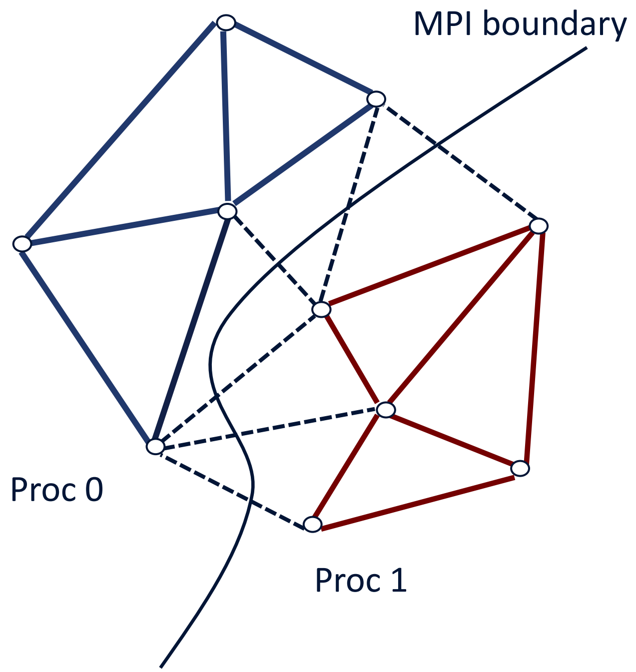
OP2 – Generated Code - CPU

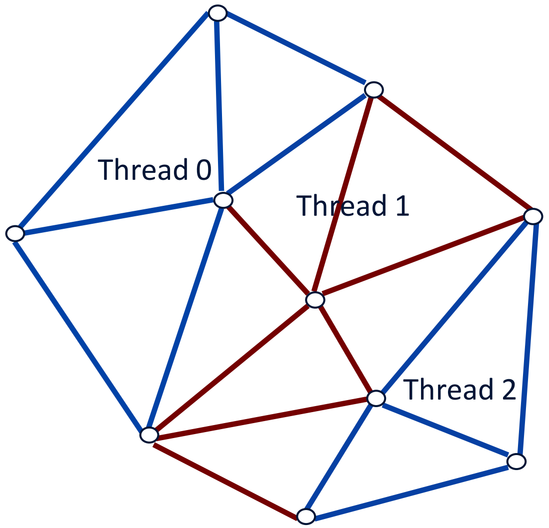
```fortran
! elemental kernel
SUBROUTINE res_calc(x1,x2,q1,q2,adt1,adt2,res1,res2)
...
END SUBROUTINE

! wrapper function - calls elemental kernel
SUBROUTINE op_wrap_res_calc( ... )
...
...
  DO i1 = bottom, top-1, 1
    IF (mod(i1,testfreq).eq.0) THEN
      call op_mpi_test_all(argc,args)
    END IF
    map1idx = opDat1Map(1 + i1 * opDat1MapDim + 0)+1
    map2idx = opDat1Map(1 + i1 * opDat1MapDim + 1)+1
    map3idx = opDat3Map(1 + i1 * opDat3MapDim + 0)+1
    map4idx = opDat3Map(1 + i1 * opDat3MapDim + 1)+1
    ! kernel call
    CALL res_calc(
         opDat1Local(1,map1idx), opDat1Local(1,map2idx),
         opDat3Local(1,map3idx), opDat3Local(1,map4idx),
         opDat5Local(1,map3idx), opDat5Local(1,map4idx),
         opDat7Local(1,map3idx), opDat7Local(1,map4idx))
    END DO
END SUBROUTINE
```

```fortran
! host function - setting up pointers and indirect accesses
SUBROUTINE res_calc_host( userSubroutine, set, opArg1, &
& opArg2, & opArg3, & opArg4, opArg5, opArg6, opArg7, opArg8)

  ! MPI halo exchanges
  n_upper = op_mpi_halo_exchanges(...)
  ...
  ...
  ! set up c to Fortran pointers
  CALL c_f_pointer(opArg1%data,opDat1Local, ...)
  CALL c_f_pointer(opArg1%map_data,opDat1Map, ...)
  ...
  ...
  ! compute over core iterations/elements
  CALL op_wrap_res_calc( opDat1Local, opDat3Local, &
                       & opDat5Local, opDat7Local, &
                       & opDat1Map, opDat1MapDim,  &
                       & opDat3Map, opDat3MapDim,  &
                       & 0, opSetCore%core_size, &
                       & numberOfOpDats,opArgArray,testfreq)

  ! wait for Halos to be received
  CALL op_mpi_wait_all(numberOfOpDats,opArgArray)

  ! compute over halo (redundant) iterations/elements
  CALL op_wrap_res_calc( opDat1Local, opDat3Local, &
  & opDat5Local, opDat7Local, &
  & opDat1Map, opDat1MapDim, opDat3Map, opDat3MapDim, &
  & opSetCore%core_size, n_upper,numberOfOpDats,&
  & opArgArray,2147483647)

  IF ((n_upper .EQ. 0) .OR. &
  &   (n_upper .EQ. opSetCore%core_size)) THEN
    CALL op_mpi_wait_all(numberOfOpDats,opArgArray)
  END IF

  ! mark halos dirty
  CALL op_mpi_set_dirtybit(numberOfOpDats,opArgArray)
  ....
  ....
END SUBROUTINE
END MODULE
```

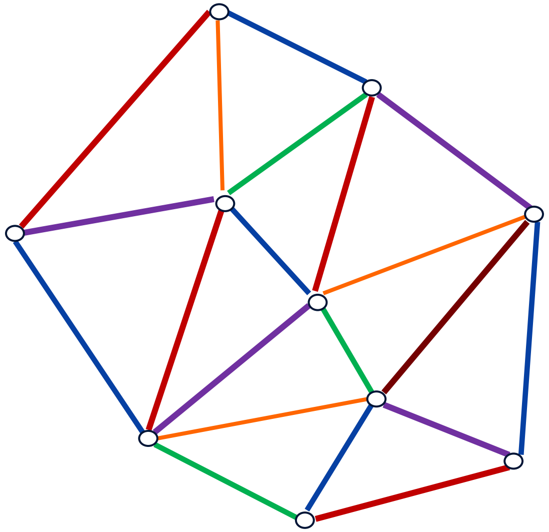
https://github.com/OP-DSL/OP2-APPS/blob/main/apps/fortran/airfoil/airfoil_hdf5/res_calc_seqkernel.F90

- ❑ Distributed memory parallelization
  - Mesh partitioning
  - Standard halo exchange methods
  - Redundant computation
- ❑ Single node – Inter-thread-block
  - Coloring
  - No two blocks of the same color update the same memory location
- ❑ Single node – Intra-thread block
  - Coloring
  - No two edges of the same colour update the same node
  - Use atomics





Threads 0 and 2 can run in parallel



OP2 – Generated Code – GPU with CUDA

```
SUBROUTINE res_calc_gpu(x1,x2,q1,q2,adt1,adt2,res1,res2)
  IMPLICIT NONE
  integer*4 istat
  REAL(kind=8)  :: x1(2)
  REAL(kind=8)  :: x2(2)
  REAL(kind=8), INTENT(IN) :: q1(4)
  REAL(kind=8), INTENT(IN) :: q2(4)
  REAL(kind=8), INTENT(IN) :: adt1
  REAL(kind=8), INTENT(IN) :: adt2
  REAL(kind=8) :: res1(4)
  REAL(kind=8) :: res2(4)
  REAL(kind=8) :: dx,dy,mu,ri,p1,vol1,p2,vol2,f
  dx = x1(1) - x2(1)
  dy = x1(2) - x2(2)
  ri = 1.0 / q1(1)
  p1 = 0.4 * (q1(4) - 0.5 * ri * (q1(2) * q1(2) + &
           & q1(3) * q1(3)))
  vol1 = ri * (q1(2) * dy - q1(3) * dx)
```

```
  ri = 1.0 / q2(1)
  p2 = 0.4 * (q2(4) - 0.5 * ri * (q2(2) * q2(2) + &
           & q2(3) * q2(3)))
  vol2 = ri * (q2(2) * dy - q2(3) * dx)
  mu = 0.5 * (adt1 + adt2) * 0.05
  f = 0.5 * (vol1 * q1(1) + vol2 * q2(1)) + &
         & mu * (q1(1) - q2(1))
  istat = atomicAdd(res1(1),+ f)
  istat = atomicAdd(res2(1),- f)
  f = 0.5 * (vol1 * q1(2) + p1 * dy + &
         & vol2 * q2(2) + p2 * dy) + mu * (q1(2) - q2(2))
  istat = atomicAdd(res1(2),+ f)
  istat = atomicAdd(res2(2),- f)
  f = 0.5 * (vol1 * q1(3) - p1 * dx + &
         & vol2 * q2(3) - p2 * dx) + mu * (q1(3) - q2(3))
  istat = atomicAdd(res1(3),+ f)
  istat = atomicAdd(res2(3),- f)
  f = 0.5 * (vol1 * (q1(4) + p1) + vol2 * (q2(4) + p2)) + &
         & mu * (q1(4) - q2(4))
  istat = atomicAdd(res1(4),+ f)
  istat = atomicAdd(res2(4),- f)
END SUBROUTINE
```

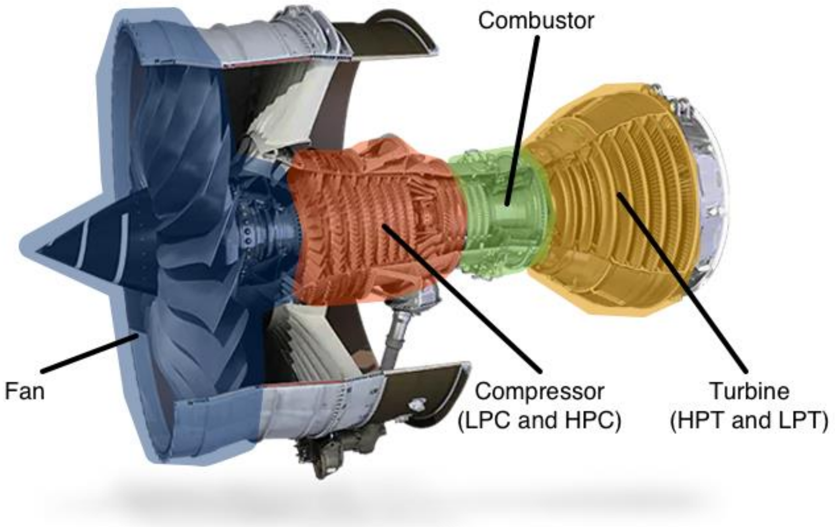
```
! CUDA kernel function
attributes (global) SUBROUTINE op_cuda_res_calc( &
& opDat1Deviceres_calc, opDat3Deviceres_calc,
& opDat5Deviceres_calc, opDat7Deviceres_calc,
& opDat1Map, opDat3Map, start, end, setSize)
...
...
  i1 = threadIdx%x - 1 + (blockIdx%x - 1) * blockDim%x
  IF (i1+start<end) THEN
    i3 = i1+start
    map1idx = opDat1Map(1 + i3 + setSize * 0)
    map2idx = opDat1Map(1 + i3 + setSize * 1)
    map3idx = opDat3Map(1 + i3 + setSize * 0)
    map4idx = opDat3Map(1 + i3 + setSize * 1)
    ! kernel call
    CALL res_calc_gpu( &
    & opDat1Deviceres_calc(1+map1idx*(2):map1idx*(2)+2), &
    & opDat1Deviceres_calc(1+map2idx*(2):map2idx*(2)+2), &
    & opDat3Deviceres_calc(1+map3idx*(4):map3idx*(4)+4), &
    & opDat3Deviceres_calc(1+map4idx*(4):map4idx*(4)+4), &
    & opDat5Deviceres_calc(1+map3idx), &
    & opDat5Deviceres_calc(1+map4idx), &
    & opDat7Deviceres_calc(1+map3idx*(4):map3idx*(4)+4), &
    & opDat7Deviceres_calc(1+map4idx*(4):map4idx*(4)+4))
  END IF
END SUBROUTINE
```

https://github.com/OP-DSL/OP2-APPS/blob/main/apps/fortran/airfoil/airfoil_hdf5/res_calc_kernel.CUF

EVOLVING PRODUCTION CODES – ROLLS-ROYCE HYDRA TO OP2-HYDRA

- Virtual certification of Gas Turbine Engines – EPSRC Prosperity Partnership (ASIMOV)
  - Main consortium with partners – EPCC, Warwick, Oxford, Cambridge, Bristol and Rolls-Royce plc.

- 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 7 days

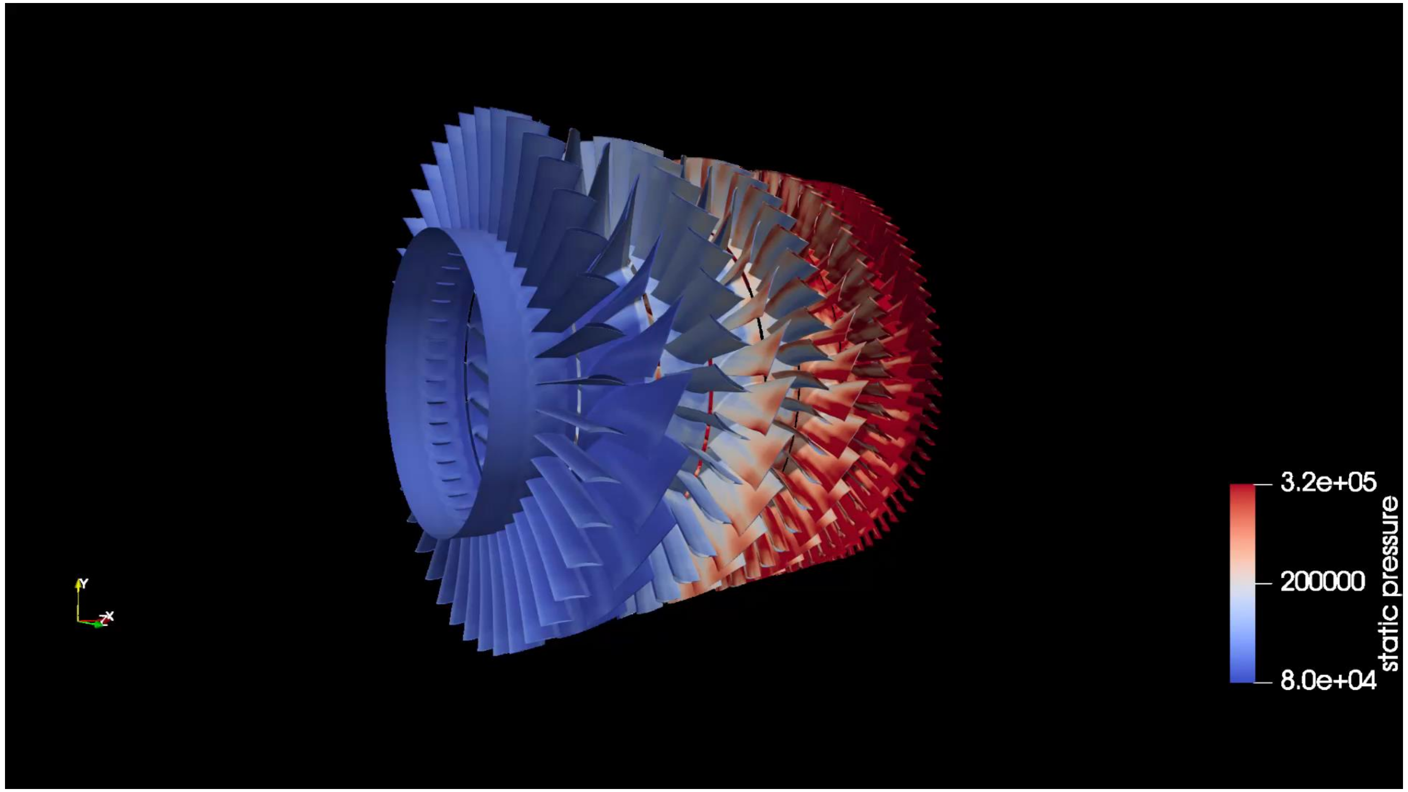


- Setup
  - Moving rotor-stator – sliding planes interfaces
  - Rotors and Stators modelled with Hydra CFD suite – URANS (360 degree models)
  - 10 rotor-stator interfaces
  - Code coupling for sliding planes – move from current monolithic (Hydra only) production code to coupling

- 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)

OP2-Hydra Performance *

System	ARCHER2 HPE Cray EX [6]	Cirrus SGI/HPE 8600 GPU Cluster [4]
Processor	AMD EPYC 7742 @ 2.25 GHz	Intel Xeon Gold 6248 (Cascade Lake) @ 2.5 GHz + NVIDIA Tesla V100-SXM2-16GN 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`	
Power/node	660W	≈ 900W



OP2-Hydra Performance *

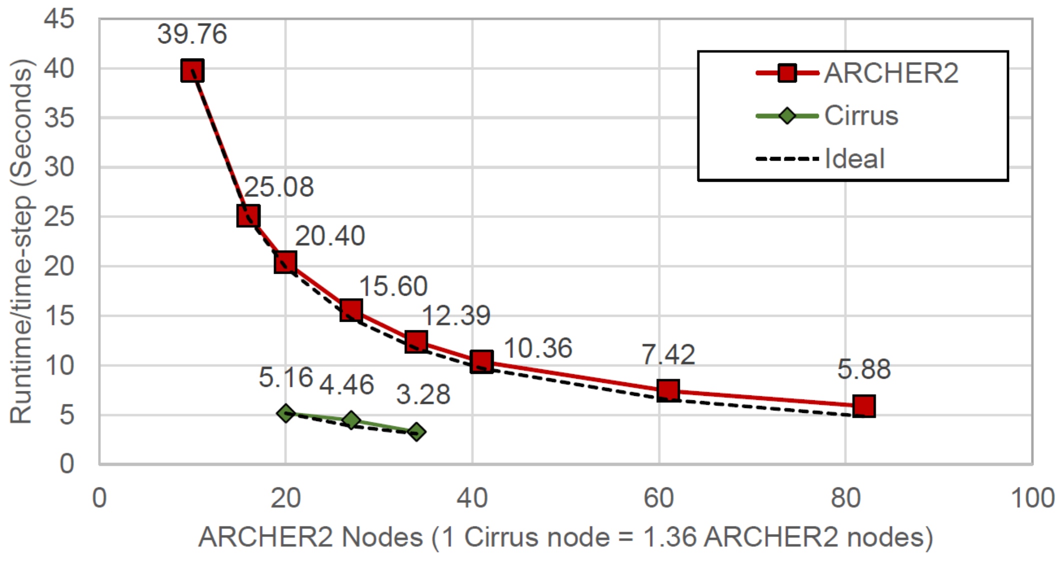
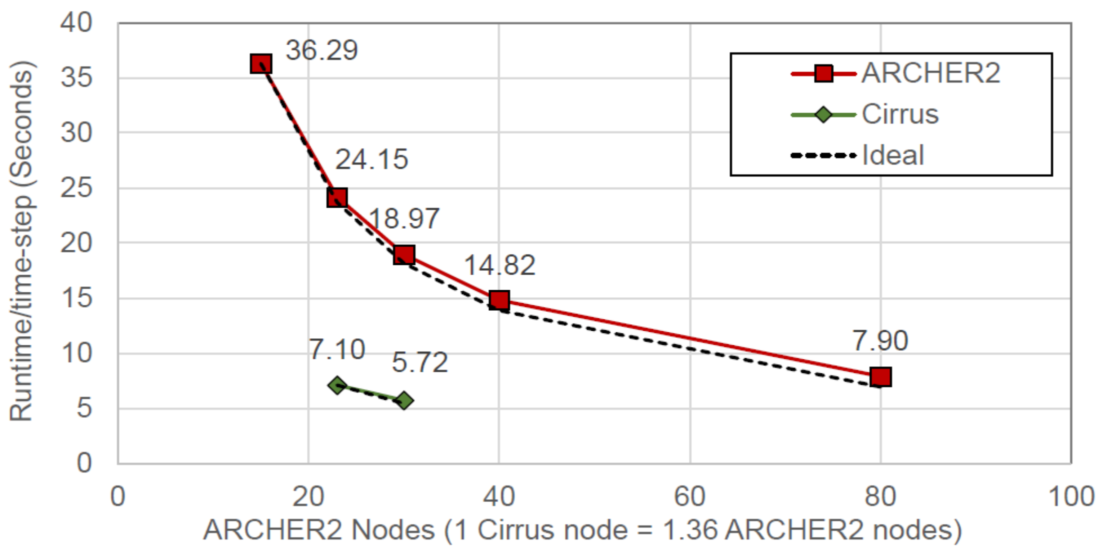


Figure 7: Rig250 $1 - 10_{430M}$ Mesh Runtime



Rig250 $1 - 2_{653M}$ Mesh Runtime

- 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

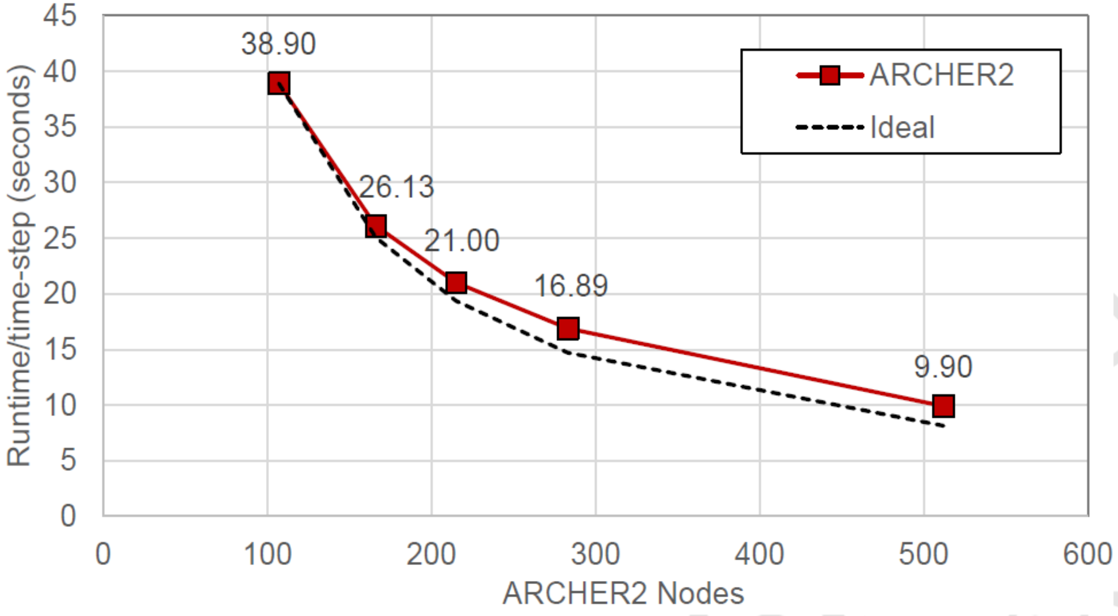
3.7- 4x speedup

- ARCHER2 @ 80 nodes
  - 88% parallel efficiency
  - 8% coupling overhead
- Cirrus @ 22 nodes
  - 94% parallel efficiency
  - 12% coupling overhead
- Cirrus @ 22 nodes
  - 94% parallel efficiency
  - 12% coupling overhead

3.3 - 3.4x speedup

* Results under review

OP2-Hydra Performance *



Rig250 $1-10_{4.58B}$ Mesh Runtime

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

Achieved (A) and *Projected (P)* times to solution (hours) : Rig250, 1 revolution

Rig250 Problem	ARCHER2 Runime	ARCHER2 #nodes	Cirrus Runtime	Cirrus #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)

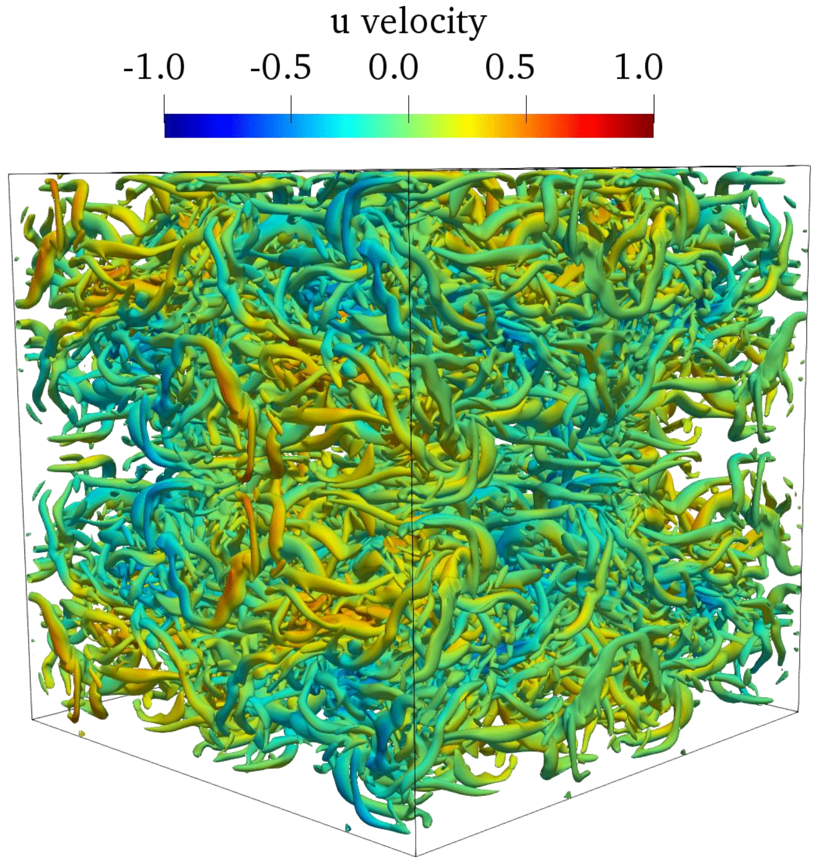
* Results under review

Production Apps – OpenSBLI (Uni. of Southampton)

- Compressible Navier-Stockes 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

OpenSBLI

https://opensbli.github.io/



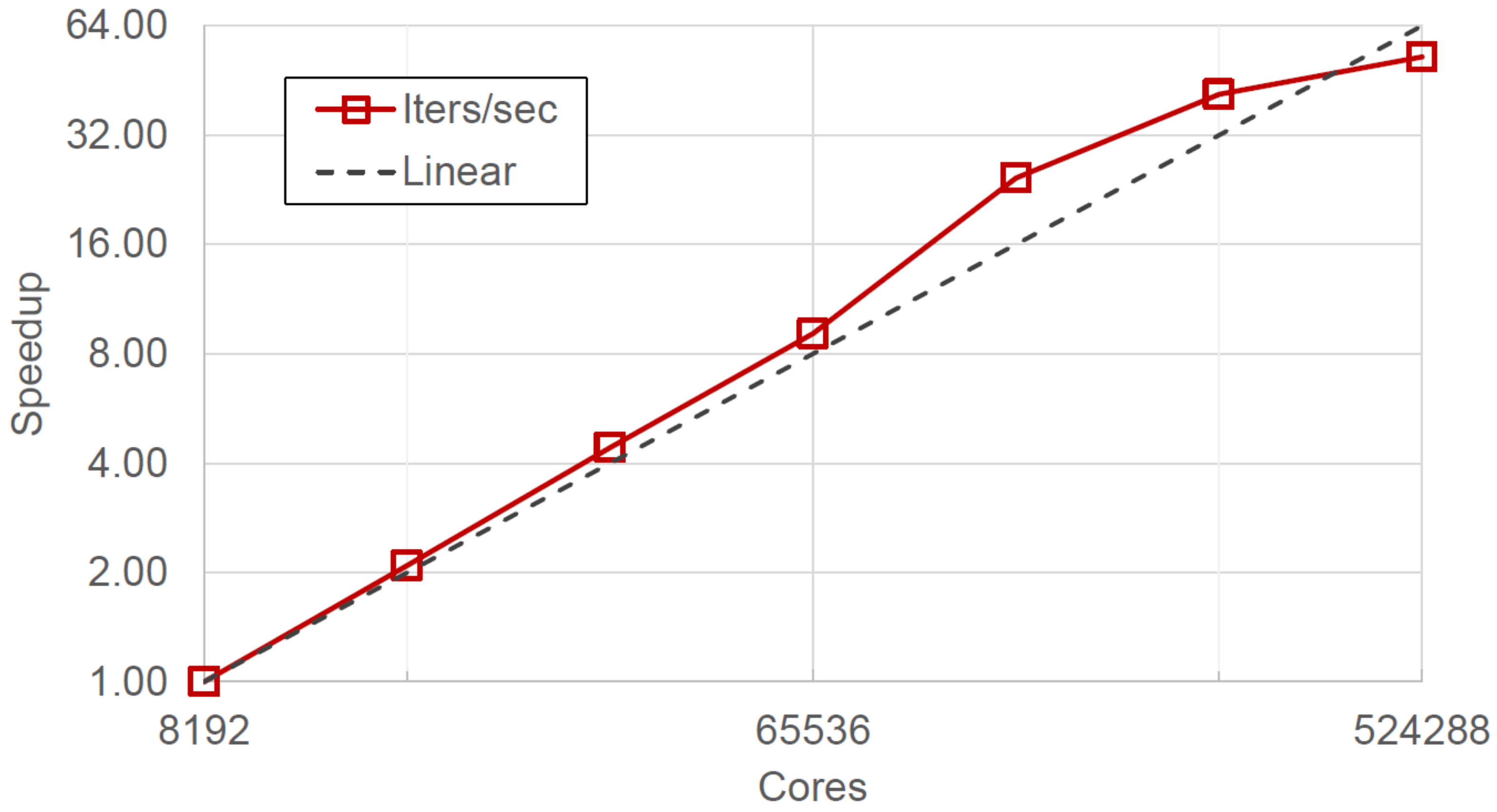

```
ndim = 3
sc1 = "**{\'scheme\':\'Teno\'}"
# Define the compresible Navier-Stokes equations in Einstein notation.
mass = "Eq(Der(rho,t), - Conservative(rhou_j,x_j,%s))" % sc1
momentum = "Eq(Der(rhou_i,t) , -Conservative(rhou_i*u_j + KD(_i,_j)*p,x_j , %s) + Der(tau_i_j,x_j) )" % sc1
energy = "Eq(Der(rhoE,t), - 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)
```

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 - 1024^3 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 and Current Projects

❑ CCP – Turbulence

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

UK TURBULENCE CONSORTIUM

UK CTRF UK CONSORTIUM ON TURBULENT REACTING FLOWS

❑ ExCALIBUR Phase 1B – Turbulence at the Exascale (one of 3 funded, £2.6M)

- 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

ExCALIBUR

❑ UK AEA Mini-Apps Project

- Collaboration with University of York
- Developing Prototype miniApps for UKAEA workload
- Investigate / advise on performance portability techniques and current state-of-the-art.

UK Atomic Energy Authority

Wider ExCALIBUR Projects of Interest – Separation of Concerns

- xDSL Project - Efficient Cross-Domain DSL Development for Exascale [https://xdsl.dev/index] : Tobias Grosser, Paul Kelly, Gerard Gorman et al.
  - A common ecosystem for DSL development
  - Funded as the ExCALIBUR Crosscutting research for exascale software and algorithms (2021)
  - Aims to offer software ecosystem for DSL building based on MLIR and LLVM

- Integrated Simulation at the Exascale: Coupling, Synthesis and Performance : Garth Wells and David Emerson et al.
  - Devlop mathematical and software tools to enable coupled simulations
  - Funded as the ExCALIBUR Crosscutting research for exascale software and algorithms (2021)
  - Coupled simulation of fusion modelling,
  - Strong coupling of electromagnetic, thermal, mechanical and fluid processes – electric propulsion

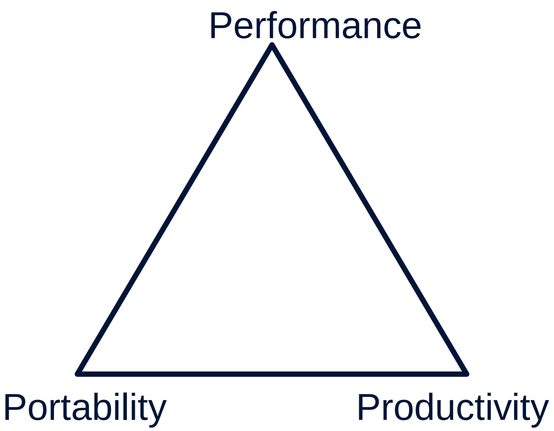
Wider ExCALIBUR Projects of Interest – Separation of Concerns

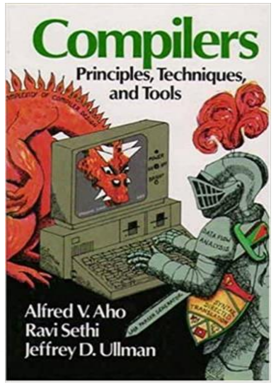
- SysGenX: Composable software generation for system-level simulation at exascale : Garth Wells, David Ham et al.
  - Funded as one of the 3 ExCALIBUR Phase 1b projects (£2.5M)
  - Developments for Firedrake, FEniCS and Bempp
  - Automatic code generation for very high-level problem description
  - Partners include Culham Centre for Fusion Energy Nvidia and Codeplay Software.

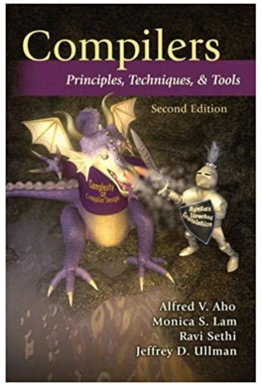
- Particles At eXascale on High Performance Computers (PAX-HPC) : group led by UCL
  - Developing exascale software for "efficiently calculating the interacting particles on vast numbers of computer cores"
  - The final one of the 3 ExCALIBUR Phase 1b projects (£3M)
  - Again, a core aim seems to be moving established software for exascale systems and extream scaling

- ExCALIBUR NEPTUNE Project – see Setven Wright's Talk on this later today.

- Utilizing domain knowledge will expose things that the compiler does not know
  - Iterating over the same mesh many times without change
  - Mesh is partitioned and colourable
- Compilers are conservative
  - Force it to do what you know is right for your code !
- Let go of the conventional wisdom that higher abstraction will not deliver higher performance
  - Higher abstraction leads to a bigger space of code synthesis possibilities
  - We can automatically generate significantly better code than what (most) people can (reasonably) write
  - Do not destroy performance portability by (hand-) tuning at a very low level to a specific platform







"Fundamentals and abstractions have more staying power than the technology of the moment"
Alfred Aho and Jeffrey Ullman (Turing Award Recipients 2020)

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.

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

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

- 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.

- UK National Supercomputing Service – ARCHER2 and resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.