WHERE DOMAIN SPECIFIC ABSTRACTIONS ARE DELIVERING PORTABILITY AND PERFORMANCE

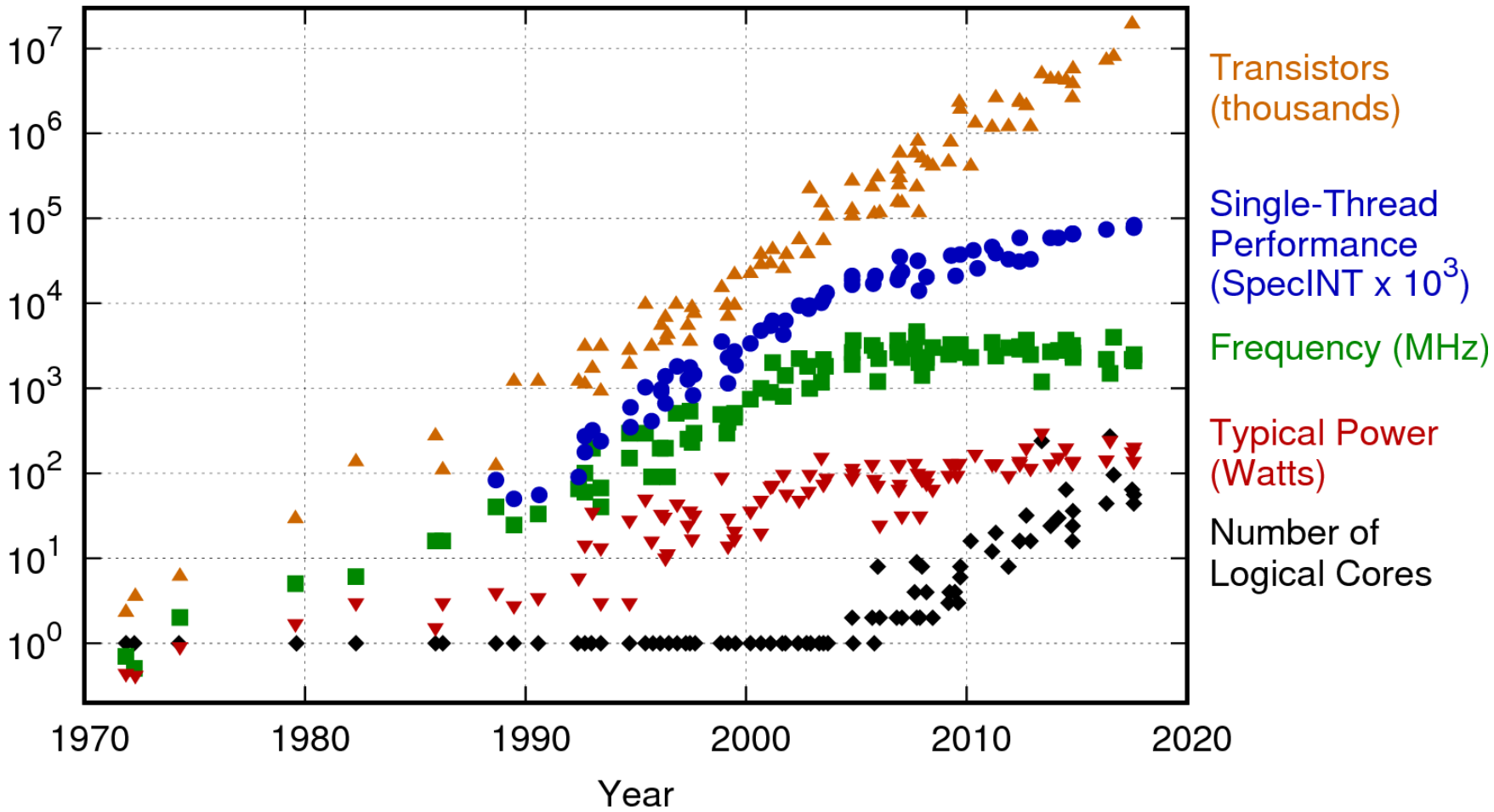
Gihan Mudalige
Royal Society Industry Fellow
Associate Professor, Department of Computer Science, University of Warwick
g.mudalige@warwick.ac.uk

Joint work with:

Istvan Reguly, Attila Sulyok, Dániel Balogh (PPCU), Mike Giles (Oxford), Carlo Bertolli (IBM Research),
Sathya Jammy, Christian Jacobs and Neil Sandham (Southampton),
Paul Kelly, Adam Betts, Fabio Luporini (Imperial),
Serge
Rolls Royce plc., NAG, UCL, STFC and many more.

WARWICK
THE UNIVERSITY OF WARWICK

SINGLE THREAD SPEEDUP IS DEAD



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

THE HAIL MARY PASS !

http://www.theemike.com/mikes-free-football-comic-book-hail-mary-pass/

"The semiconductor industry threw the equivalent of a Hail Mary pass when it switched from making microprocessors run faster to putting more of them on a chip - doing so without any clear notion of how such devices would in general be programmed."

David Patterson, University of California - Berkeley 2010

Diverse Hardware Landscape

- Traditional CPUs
  - Intel, AMD, IBM, ARM
  - multi-core (> 20 currently)
  - Deep memory hierarchy (cache levels and RAM)
  - longer vector units (e.g. AVX-512)
- GPUs
  - NVIDIA, AMD, and now Intel (Xe GPUs announced)
  - Many-core (> 1024 simpler SIMT cores)
  - CUDA cores, Tensor cores
  - Cache, Shared memory, HMB (3D stacked DRAM)
- XeonPhi (discontinued)
  - Many-core – based on simpler x86 cores
  - MCDRAM (3D stacked DRAM)
  - Have we seen the last of this ?
- Heterogeneous Processors
  - NVIDIA Volta + POWER9 + NVLink
  - AMD APUs
- FPGAs
  - Various vendors / configurations
  - Low-level language
- DSP Processors
  - e.g. The Chinese Matrix2000 GPDSP accelerators (Top500 news 29/01/2018)
- Quantum ?

But .. Even more diverse ways to programming them !

OpenMP, SIMD, CUDA, OpenCL,
OpenMP4.0, OpenACC,
SYCL/OneAPI,
MPI, PGAS
ROCm,
And more ….

Software Challenge – A Moving Target

- Each new platform requires new performance tuning effort
  - Deeper cache hierarchies and/or shared-memory (non-coherent)
  - Multiple (heterogeneous) memory spaces (device memory/host 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

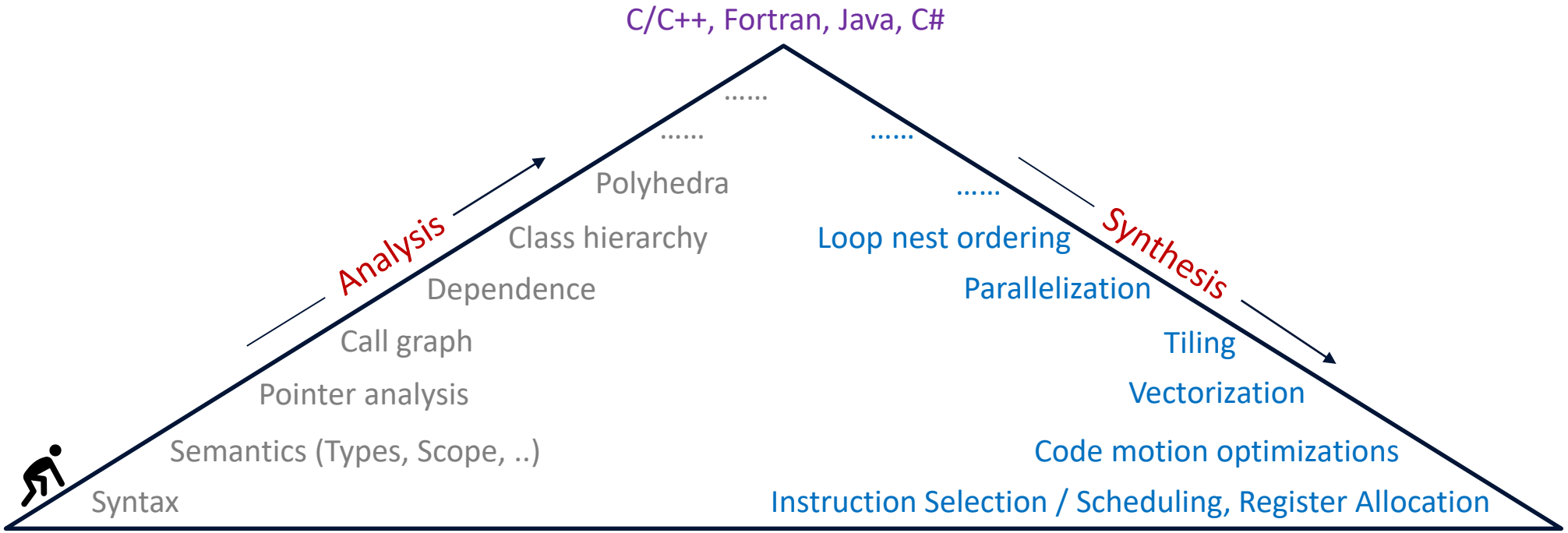
Software Challenge – Data Movement

- Computing is cheap and massively parallel
- Data movement dominates performance costs
  - Bandwidth is the main bottleneck
  - Reduce communications to reduce energy
- Current programming environments
  - Not designed to take account of the cost of communication
  - Simply rely on the hardware cache coherency to virtualize data movement
  - Difficult to express data locality and affinity
  - Difficult to describe how to decompose and layout data in the memory
- Need easier ways to program for optimized data movement
  - Express information about data locality/affinity
  - A data-centric programming model ?

OUTLINE

- ❑ Motivation ✓
- ❑ Raising the Level of Abstraction
- ❑ Oxford Parallel Libraries – OP2 and OPS
- ❑ Codes and Projects using OP2/OPS
- ❑ Problems / Challenges
- ❑ Lessons Learnt and Conclusions

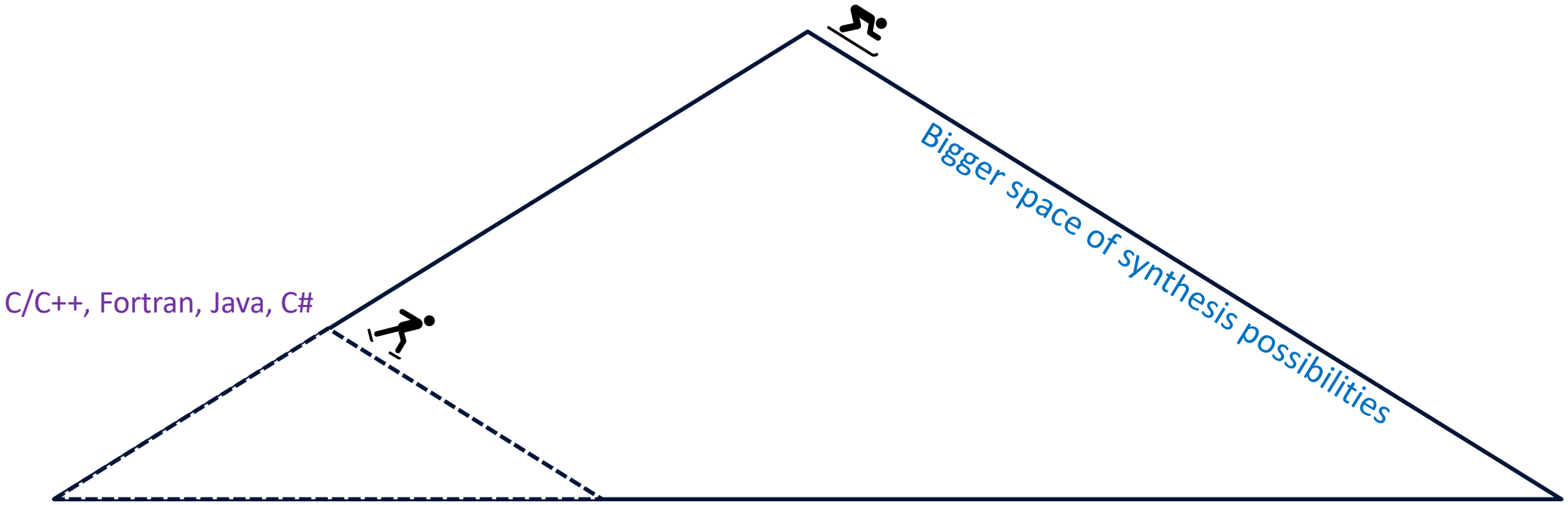
The Level of Abstraction – Climbing the Analysis Hill and Generating Code



- The higher you can get to (in analysis) the bigger the space of code synthesis possibilities
- If you start at a lower level – climbing higher is a struggle
  - Difficult to ensure optimizations are safe (e.g. data races, pointer aliasing)
  - Sometimes, impossible to extract richer information (e.g. data partitioning/layouts, memory spaces)
  - Limits the optimizations possible
- Compounding the issue - the way code is written by (most) people will not be easy to analyse !

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)

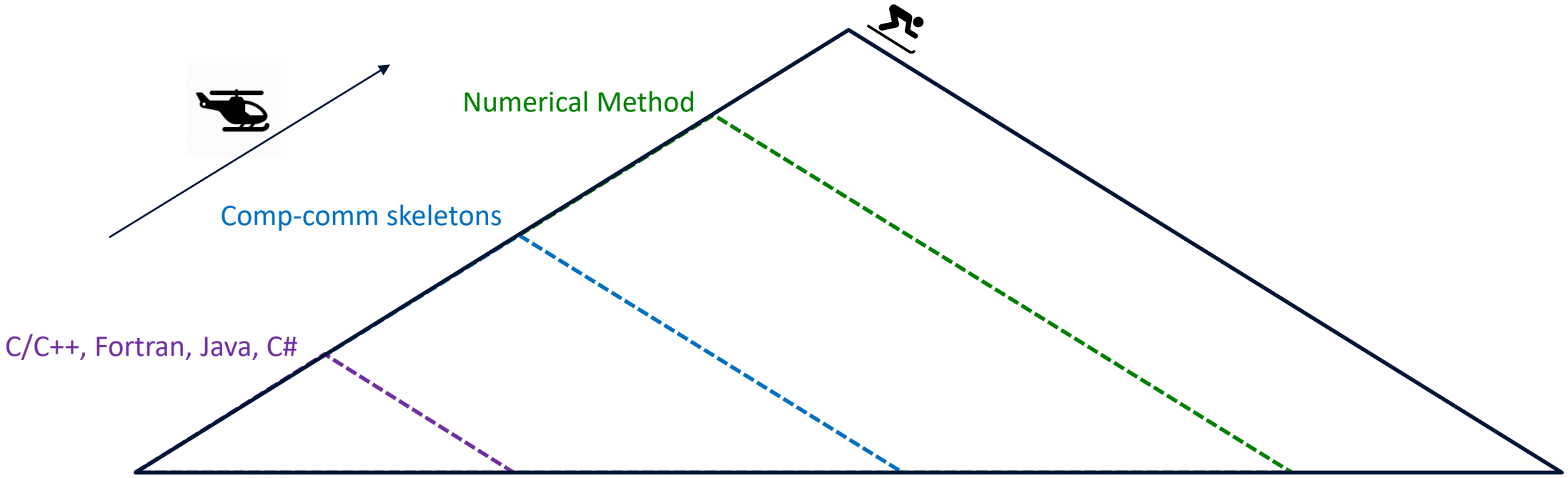
The Level of Abstraction



- If you can start higher
  - Results in a bigger space of code synthesis possibilities
  - Could they give the same (or better) performance as code written by hand ?
  - Could these possibilities include targeting different (parallel) architectures ?
- How can you start higher ?

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)

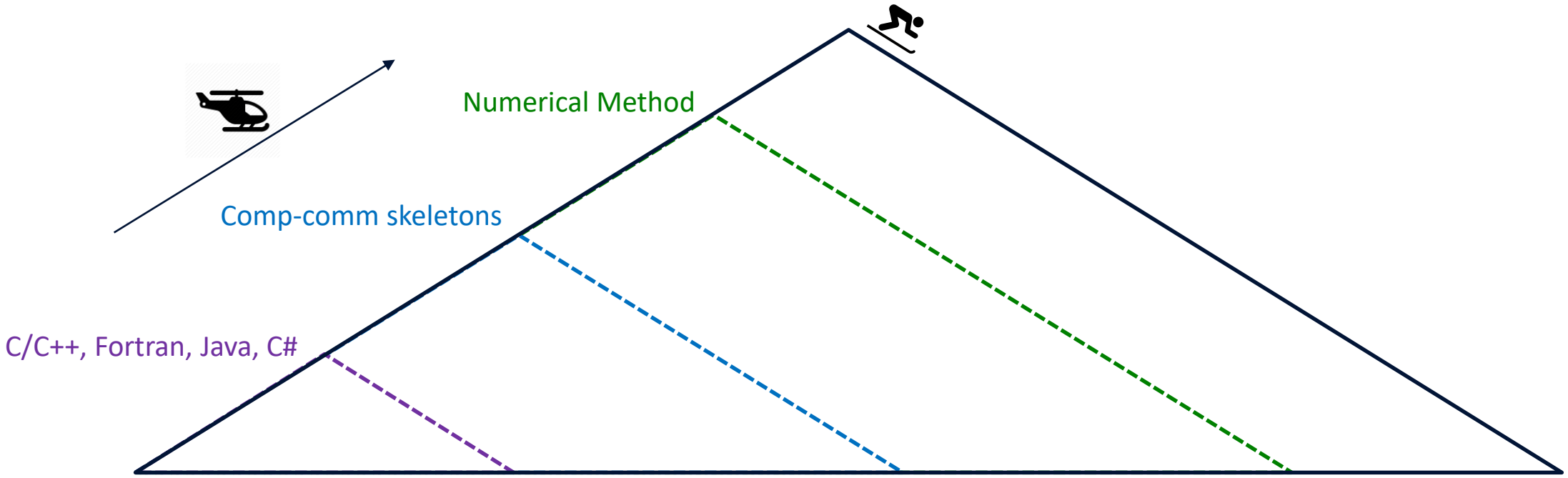
Domain Specific Abstractions



- 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

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)

Domain Specific Abstractions



- If you get the abstraction right, then:
  - Can isolate numerical methods from mapping to hardware
  - Can reuse a body of optimizations/code generation expertise/techniques for this class (or numerical method) to match target hardware

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)

How Do We Raise The Level of Abstraction ?

- 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 straightjacket" – Mike Giles

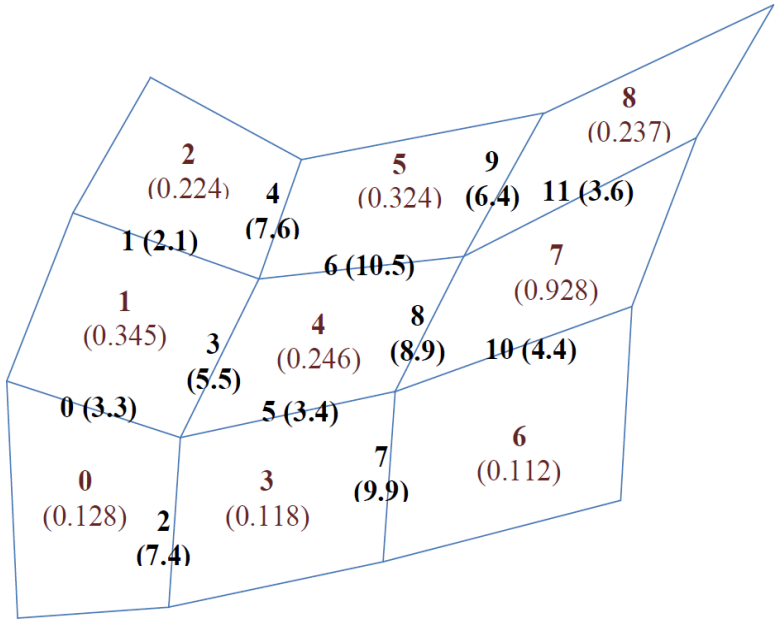
- Implementation of the API left to a lower level
  - Target implementation to hardware – can use best optimizations

OP2 for Unstructured-mesh Applications

```
//sets
op_set nodes  = op_decl_set(nnode,  "nodes");
op_set edges  = op_decl_set(nedge,  "edges");
op_set cells  = op_decl_set(ncell,  "cells");


//mapping between sets
op_map pedge   = op_decl_map(edges, nodes, 2, edge,  "pedge");
op_map pecell  = op_decl_map(edges, cells, 2, ecell, "pecell");


//data on sets
op_dat p_x     = op_decl_dat(nodes , 2, "double", x    ,"p_x");
op_dat p_q     = op_decl_dat(cells , 4, "double", q    ,"p_q");
op_dat p_adt   = op_decl_dat(cells , 1, "double", adt  ,"p_adt");
op_dat p_res   = op_decl_dat(cells , 4, "double", res  ,"p_res");
```



OP2 for Unstructured-mesh Applications

```
//elemental kernel
void res_calc(const double* x1, const double* x2,
const double* q, double* res1, double* res2){
  //computations such as:
  res1[0] += q[0]*(x1[0]-x2[0]);
  ...
  ...
}

//Parallel loop
op_par_loop(res_calc,"residual_calculation", edges,
       op_arg_dat(p_x, 0, pedge, 2, "double", OP_READ),
       op_arg_dat(p_x, 1, pedge, 2, "double", OP_READ),
       op_arg_dat(p_q, -1,OP_ID, 4, "double", OP_READ),
       op_arg_dat(p_res, 0, pecell, 4, "double", OP_INC),
       op_arg_dat(p_res, 1, pecell, 4, "double", OP_INC));
```

Directly accessed

Iteration set

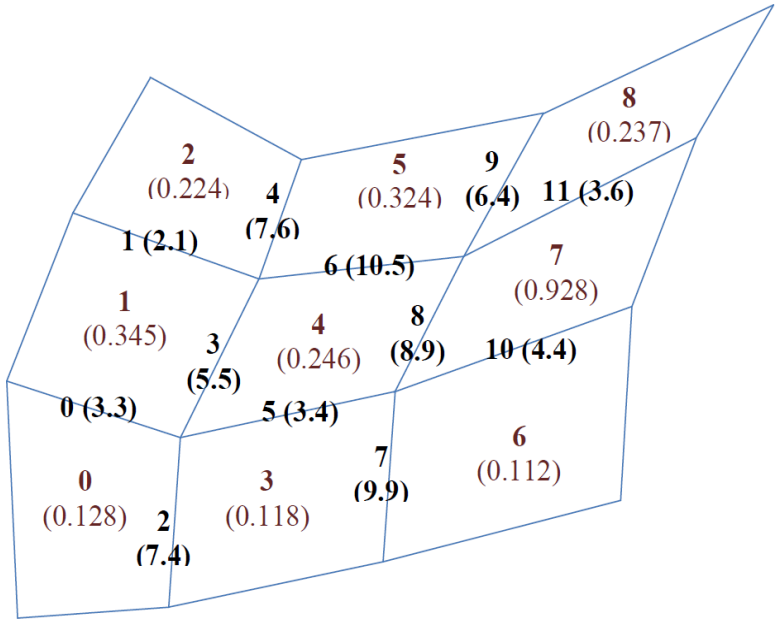
Access descriptors

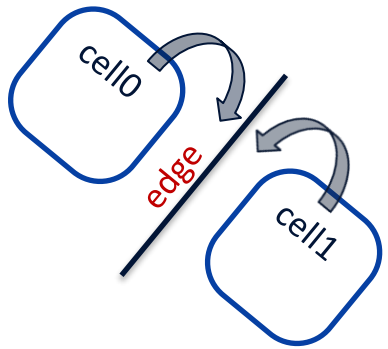
Indirection index

Indirectly accessed via mapping

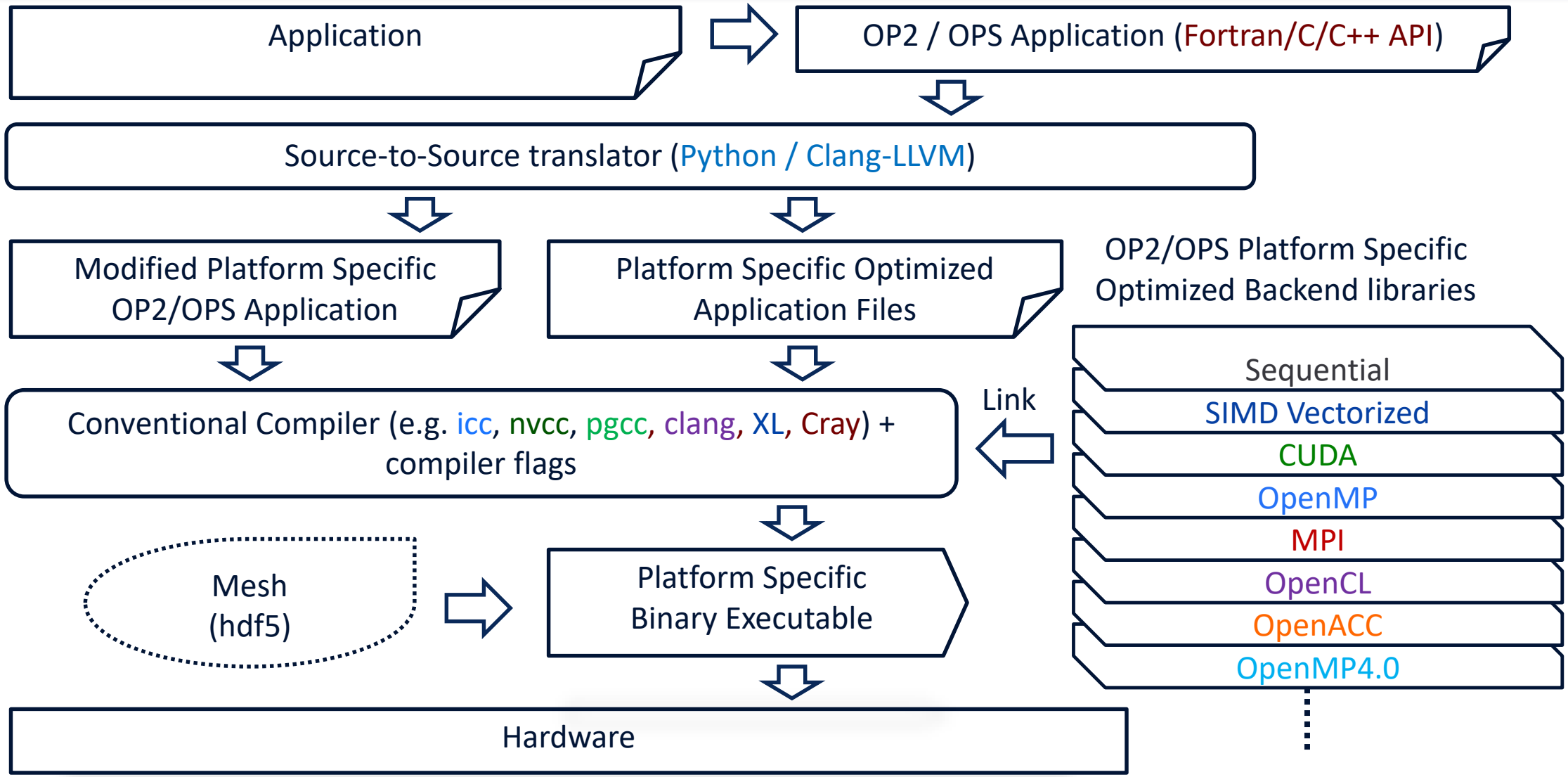
Data dimension

Data type (for checks)





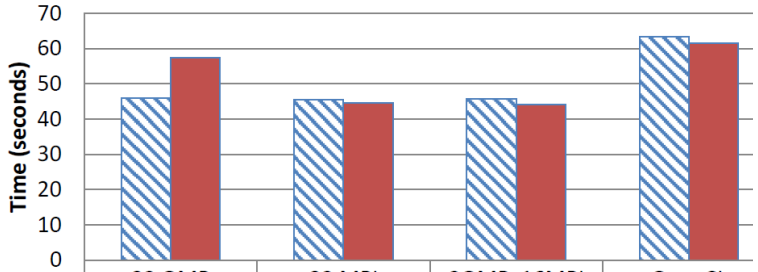
Application Development



Codes and Current Projects – UK MAC miniApps

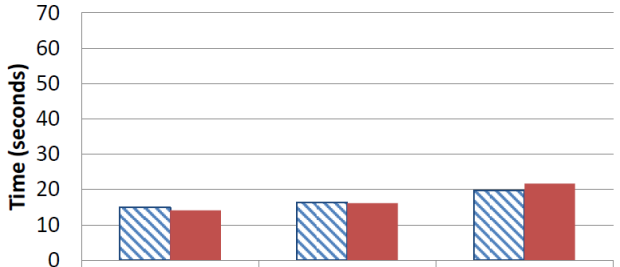
- AWE - CloverLeaf2D

2 x 8-core Intel Xeon E5-2680 2.70GHz (Sandy bridge)



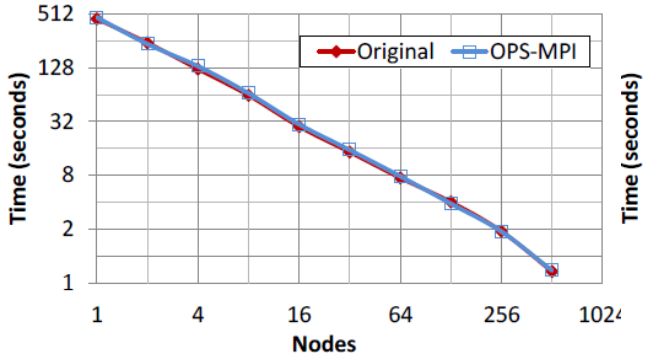
	32 OMP	32 MPI	2OMPx16MPI	OpenCL
OPS	45.92	45.55	45.82	63.35
Original	57.39	44.60	44.22	61.54

2 x NVIDIA Tesla K20c

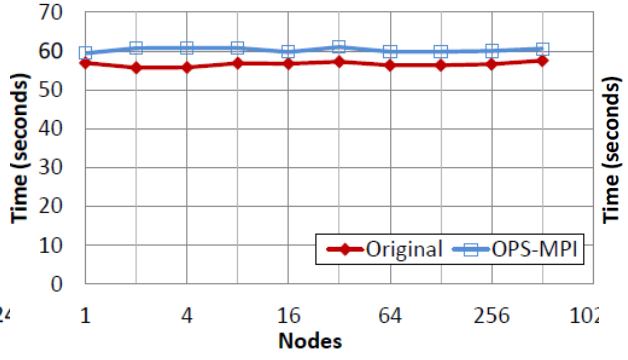


	CUDA	OpenCL	OpenACC
OPS	15.01	16.27	19.82
Original	14.14	16.19	21.67

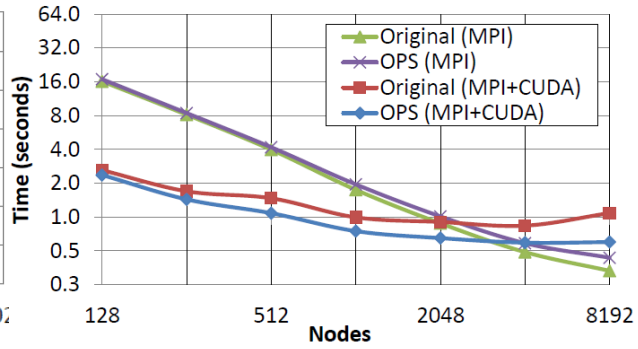
- AWE CloverLeaf – ~6k LoC originally written in Fortran 90
- Original code - Multiple manually parallelized versions
- Re-engineered to use OPS – uses OPS C/C++ API
- Results demonstrate <15% overhead due to abstraction



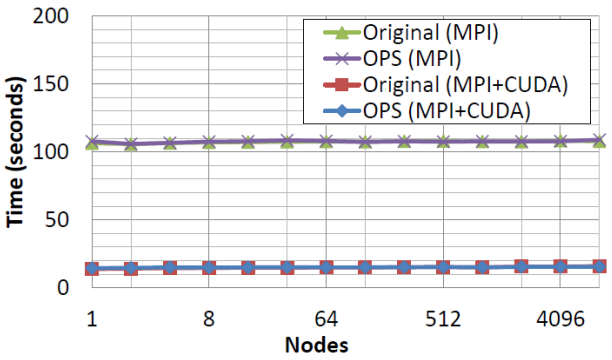
Cray XC30 (ARCHER) - Strong scaling 15360^2 mesh



Weak scaling 2x 3840 x 3840 mesh per node



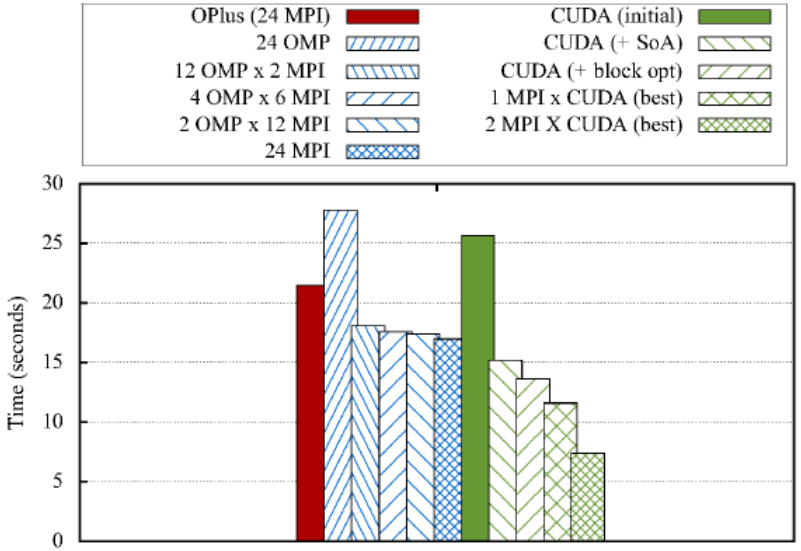
Cray XK7 (TITAN)- Strong scaling 15360^2 mesh



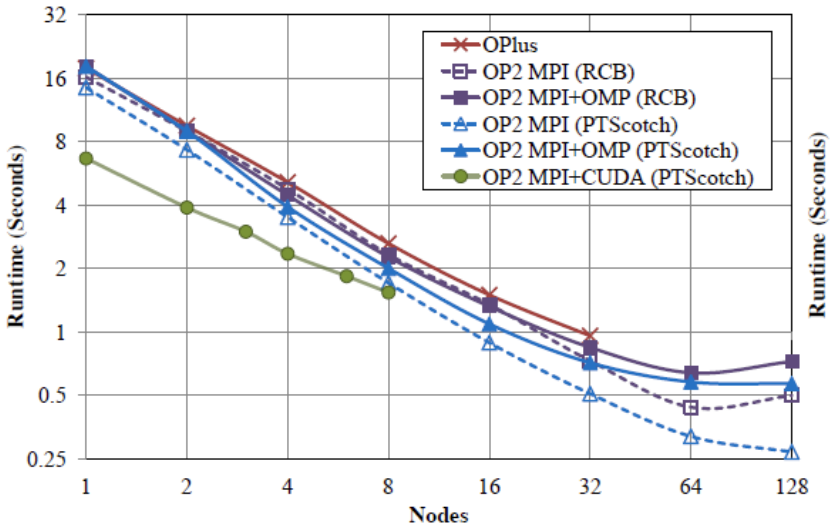
Weak scaling 2x 3840 x 3840 mesh per node

- ARCHER (Cray XC30) 2×12-core Intel Xeon E5-2697 2.70GHz (Ivy Bridge)
- Titan (Cray XK7) – AMD Opteron 6274 (16 core) + NVIDIA K20X

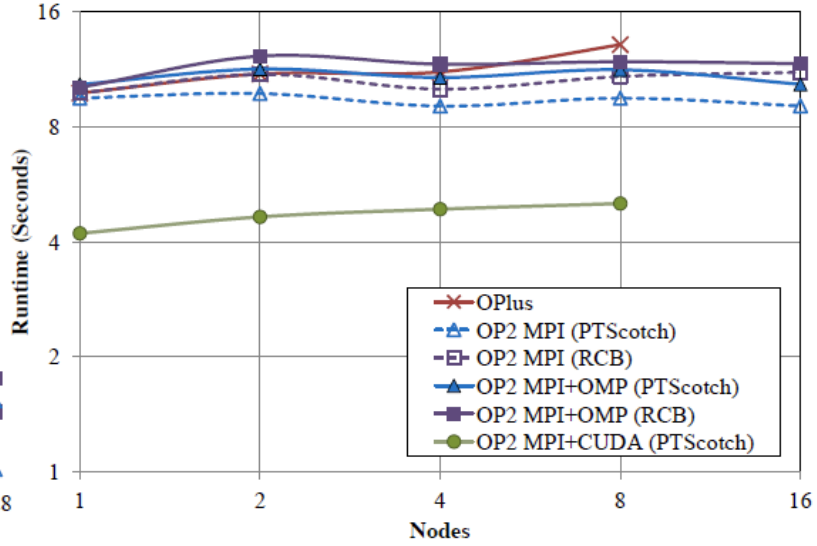
Royal Society Fellowship and ASIMOV with Rolls-Royce plc.



OP2 Hydra NASA Rotor 37, Multi-/Many-core performance
2xTesla K20c GPUs + 2x6-core Intel Xeon E5-2640 2.50GHz
(2.5M edges, 20 iterations)



Strong Scaling (2.5M mesh edges)



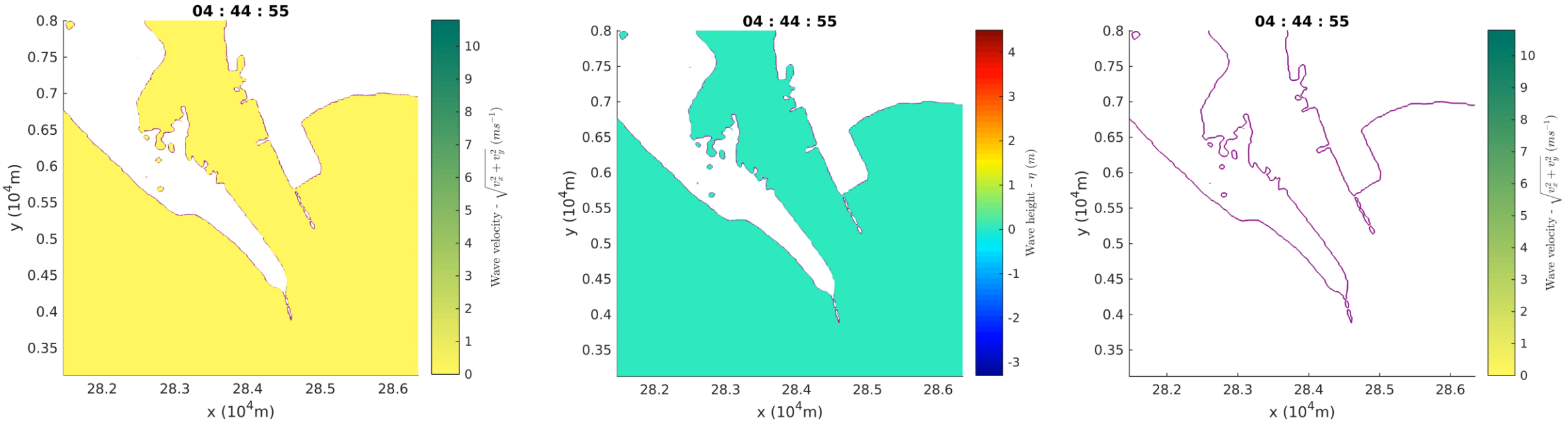
Weak Scaling (0.5M mesh edges per node)

OP2 Hydra NASA Rotor 37, Scaling on HECToR (MPI, MPI+OpenMP) and Jade (MPI+CUDA) : 20 iterations
HECToR (Cray XE6) – 2 x 16-core AMD Opteron 6276 (Interlagos) 2.3GHz
Jade (NVIDIA GPU Cluster) – 2 x Tesla K20m GPUs + Intel Xeon E5-1650 3.2GHz

- Rolls-Royce Hydra – ~100k LoC originally written in Fortran 77 , over 300 parallel loops
- Re-engineered to use OP2 – uses OP2 Fortran API
- Automatically Parallelized with OP2 – MPI + (OpenMP 3.0, CUDA, OpenACC) , OpenMP 4.0 (experimental)
- Royal Society Industrial Fellowship (2018) - Moving OP2-Hydra to production

I. Z. Reguly, G. R. Mudalige , C. Bertolli, M. B. Giles, A. Betts, P. H. J. Kelly, and D. Radford. *Acceleration of a Full-scale Industrial CFD Application with OP2* . (2015) IEEE Transactions on Parallel and Distributed Systems, doi:10.1109/TPDS.2015.2453972

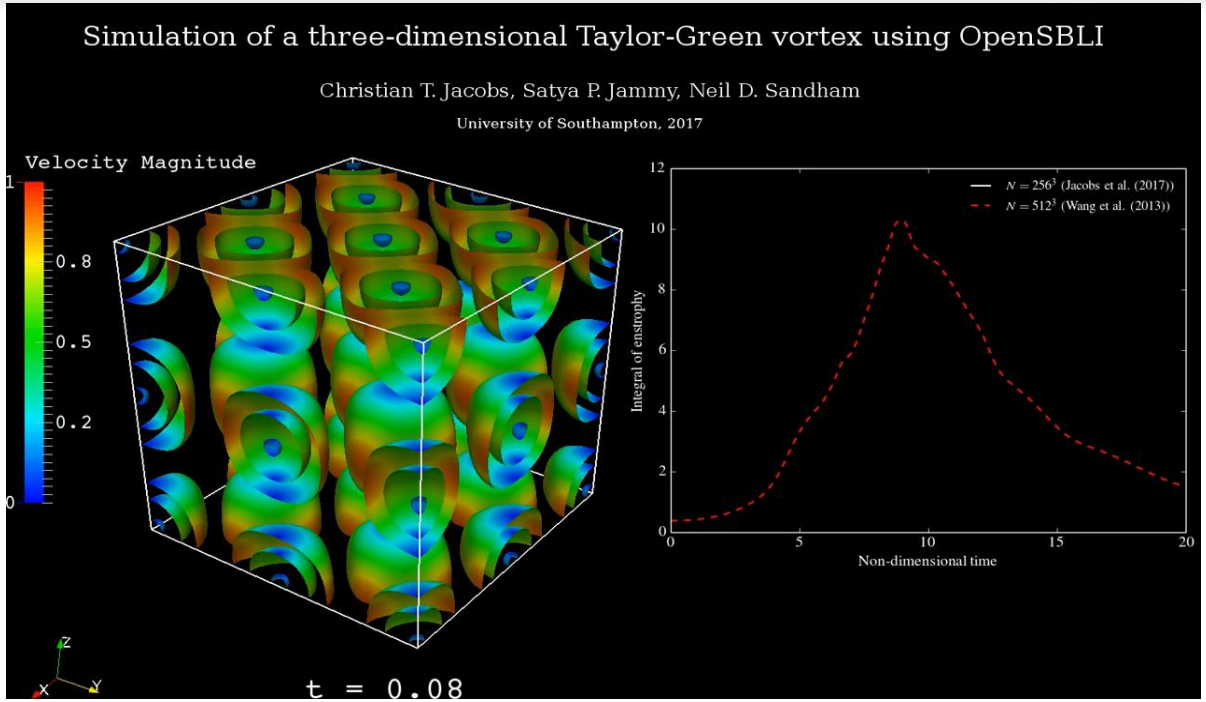
Codes and Current Projects - Volna (Uncertainty Quantification @ the ATI London)

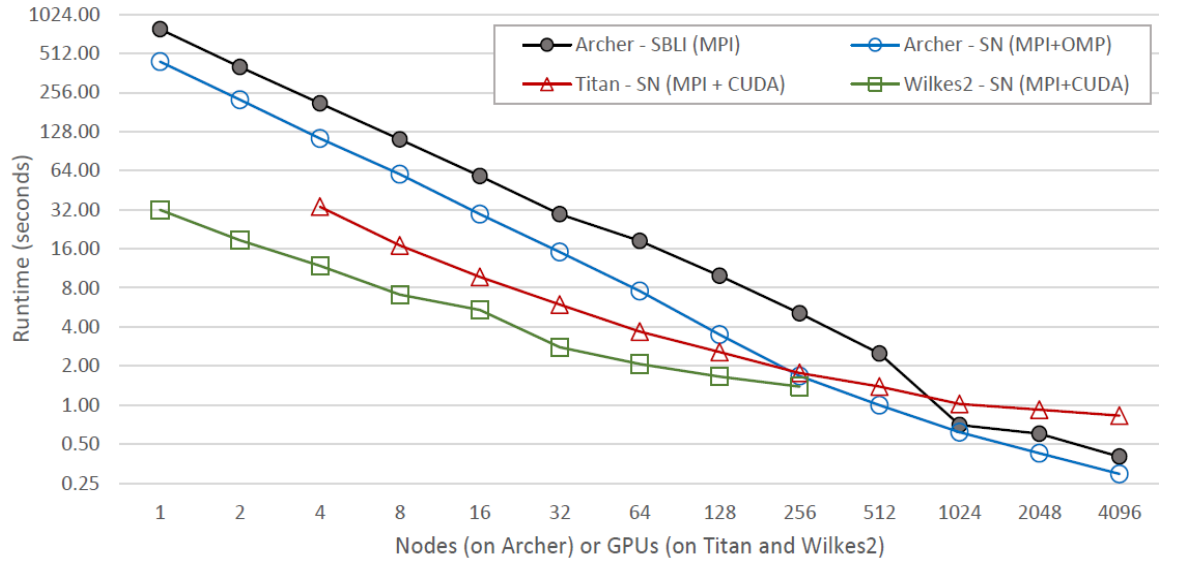


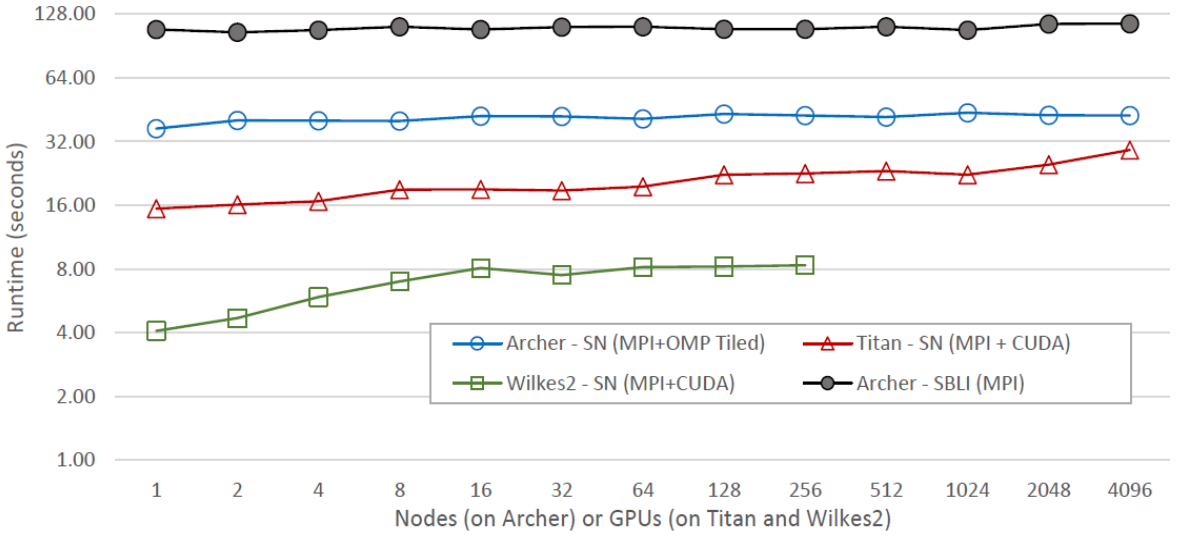
- Uncertainty quantification of multi-scale and multi-physics computer models – ATI Project with Dr. Serge Guillas
- Coupling 3D Earthquake simulation (SPECFM3D) with Tsunami simulation (Volna-OP2)
- Utilizing GPUs for both applications, Single Precision.
- Volna Speedup ~6.9x on 32 GPUs (NVIDIA P100) vs 32 CPU nodes (dual-socket Xeon Gold 6142 CPUs, 32 cores / node)

Reguly, I. Z., Giles, D., Gopinathan, D., Quivy, L., Beck, J. H., Giles, M. B., Guillas, S., and Dias, F.: The VOLNA-OP2 tsunami code (version 1.5), Geosci. Model Dev., 11, 4621-4635, https://doi.org/10.5194/gmd-11-4621-2018, 2018.

Codes and Current Projects – OpenSBLI







- OpenSBLI - framework for the automated derivation of finite difference solvers from high-level problem descriptions
- Generates OPS code from high-level mathematical description
- Only GPU-ready UK Turbulence Consortium code
- OpenSBLI is now an ARCHER benchmark

G.R. Mudalige, I.Z. Reguly, S.P. Jammy, C.T. Jacobs, M.B. Giles and N.D. Sandham, (2019) Large-scale performance of a DSL-based multi-block structured-mesh application for direct numerical simulation. Journal of Parallel and Distributed Computing. doi:10.1016/j.jpdc.2019.04.019

Codes and Current Projects – NAG PDE Toolkit

- The Numerical Algorithms Group (NAG) developing a new PDE solver package based on OPS
- Initially targeting financial industry with further plans for Oil and Gas
- Initial phase OPS only, latter phases may include OP2
- Better software engineering / testing / maintenance

OTHER USERS

- ETH Zurich – BASEMENT (Basic Simulation Environment for Computation of Environmental Flows and Natural Hazard Simulations) – Version 3
  - Flood forecast and mitigation, River morphodynamics, Design of hydraulic structures
  - Finite volume discretisation, cell centred
  - Targeting OP2 for GPU and multi-core parallelisation

- STFC – HiLeMMS project (High-Level Mesoscale Modelling System):
  - high-level abstraction layer over OPS for the solution of the Lattice Boltzmann method
  - Adaptive mesh refinement - Chombo (Lawrence Berkeley National Labs)

- Imperial College London and University of Nottingham – CFD code development with OPS
  - Simulation of Turbomachinery flows
  - Implicit solvers using OPS's (experimental) Tridiagonal Solver API

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 ?
  - Difficult to convince users to use new code

  - Incremental conversion – loop by loop
  - Simpler than CUDA, but more difficult than OpenACC/OpenMP
  - Automated conversion ?

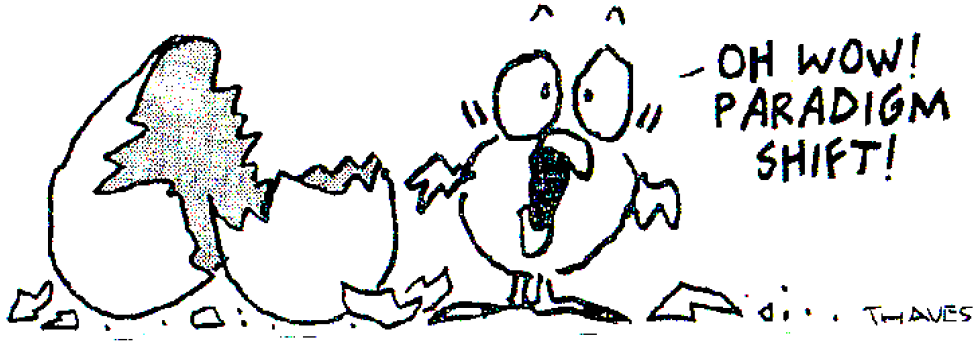
- Code-generation
  - Tools not entirely mature – currently source-to-source with Python
  - Pushing clang/LLVM source-to-source to do what we want - but what about Fortran (may be F18/Flang ?)
  - User kernel modification, Vectorization
  - Maintainable/long term source-to-source technologies (not the ROSE compiler !!)

- Maintenance
  - Currently purely done via academic and (small/short term) industrial funding
  - Long term funding – once established probably will not be different to any other classical library
  - Will require compiler expertise to maintain code generation tools

Other Major High-level Abstractions Frameworks

- FEniCS - PDE solver package - https://fenicsproject.org/
- Firedrake - automated system for the portable solution of PDEs using the finite element method (FEM) - https://www.firedrakeproject.org/ (Imperial College and others)
- 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 (Imperial College)
- Psyclone from the GungHO project Weather modelling codes STFC (and Metoffice)
- STELLA and GridTools – DSL for stencil codes, for solving PDEs - Metro Swiss
- Kokkos – C++ template library – SNL (included in Trilinos) – https://github com/kokkos/kokkos
- RAJA - C++ template libraries – LLNL - https://github com/LLNL/RAJA

Lessons Learnt and Conclusions



- 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

Downloads and More Information

- GitHub Repositories
  - OP2 – https://github.com/OP-DSL/OP2-Common
  - OPS – https://github.com/OP-DSL/OPS
  - OPS Tutorial - https://github.com/OP-DSL/OPS/tree/master/apps/c/laplace2d_tutorial
  - 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.
- Gihan Mudalige was supported by the Royal Society Industrial Fellowship Scheme (INF/R1/180012)
- CloverLeaf re-engineering research is funded by the UK AWE plc. under project "High-level Abstractions for Performance, Portability and Continuity of Scientific Software on Future Computing Systems".
- CloverLeaf development is supported by the UK AWE plc. under grants CDK0660 (The Production of Predictive Models for Future Computing Requirements) and CDK0724 (AWE Technical Outreach Programme).
- 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
- Research presented used the UK National Supercomputing Service – ARCHER 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.
- We acknowledge PRACE for awarding us access to resource Marconi based in Italy at Cineca.
- This research utilized the systems provided by Advanced Research Computing (ARC) at the University of Oxford in carrying out this work.