Daresbury Laboratory
Science & Technology
Facilities Council

High-level Abstraction for Block Structured Applications: A lattice Boltzmann Exploration

Oxford e-Research Centre

University of BRISTOL

UNIVERSITY OF Southampton

Jianping Meng, Xiao-Jun Gu, **David R. Emerson**, Gihan Mudalige, István Reguly and Mike B Giles

Scientific Computing Department

STFC Daresbury Laboratory

Daresbury, Warrington, WA4 4AD, UK

david.emerson@stfc.ac.uk

EMiT 2016 Emerging Technology Conference

2-3 June 2016, Barcelona

Outline of presentation

- Today’s challenge for scientific software
- The solution concept
- OPS brief background and history
- High level abstraction
- Performance and early results
- Summary and conclusions

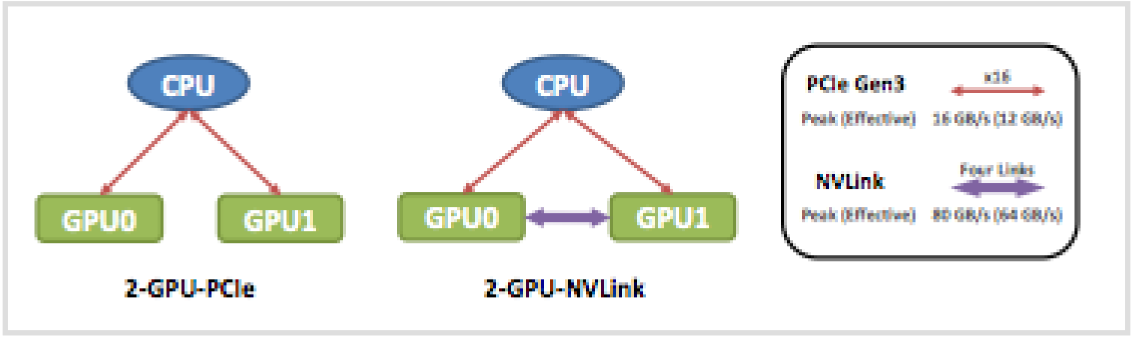
Challenge for scientific software

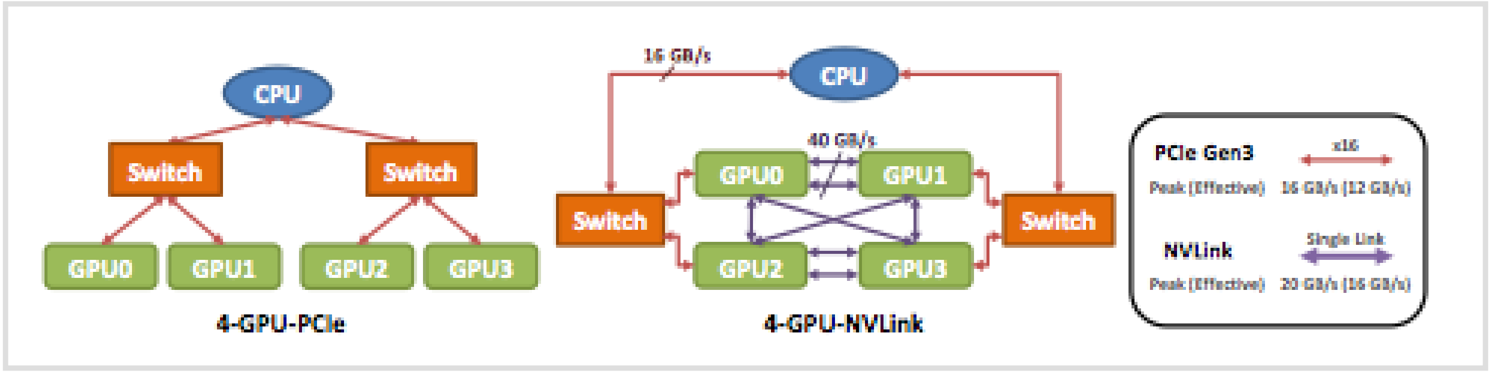
Parallel hardware: GPU, Intel Phi, GPDSP, FPGA…

Software framework: MPI, OpenMP, CUDA, OpenCL, OpenACC…

1999	OpenMP released
2001	Became practical and popular, matrix multiplication tested
2006	NVIDIA launched CUDA (November 2006)
2009	OpenCL 1.0 released with Mac OS X Snow Leopard
2010	Mathematica 8 introduced systematical GPGPU features
2011	OpenACC released
2013	Ansys Fluent 15 3D AMG coupled pressure-based solver

Challenge for scientific software





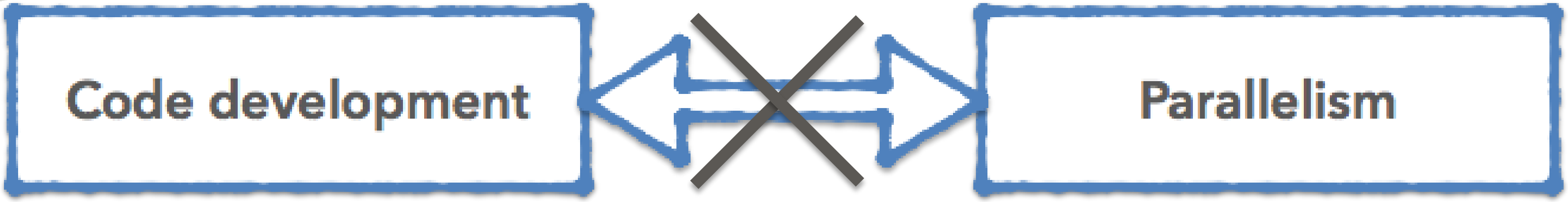
For scientific application developers,

It is a **time-consuming** and **risky** task to adapt to an emerging hardware!

But….next-gen hardware (e.g. Summit/Sierra) will have integrated GPUs

Also…TOP500 > 100 systems using accelerators – growing at ~25/yr

OPS solution concept……



- The basic idea is for the scientists and engineers to stay focused on their software development and allow OPS to provide the parallelism through abstracting all of the message passing and conversion to accelerators. This is particularly attractive to industry where software investment and software lifetime are key factors in making critical engineering decision. Complexity of today's hardware gets in the way and is making industry face mission-critical decisions in where to invest for future applications.

- The OPS (Oxford Parallel Structured software) project is developing an **open-source** framework for the execution of multi-block structured mesh applications on clusters of GPUs or multi-core CPUs and accelerators. Although OPS is designed to look like a conventional library, the implementation uses source-source translation to generate the appropriate back-end code for the different target platforms.

- For structured grid problems, OPS offers a potential way forward.

High-level abstraction

Scientific applications share many similarities:

- Data: Vector, matrix, global (sum, max, min…)
- Loop: Operations over each point and time marching…
- Stencils: Information from neighbouring points
- Common algorithms: linear algebra solvers, MG,…..

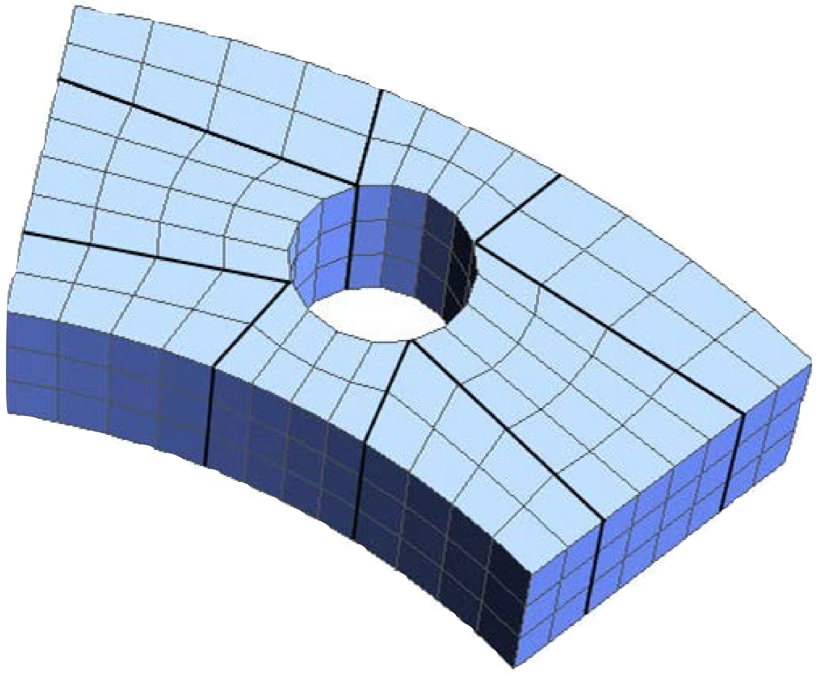
So, we may be able to:

- Build up a high-level framework on top of these similarities
- Provide clear abstractions with reasonable performance
- Develop applications with hidden hardware details
- Example: PETSc library

Higher level abstraction for CFD

For typical applications using multi-block structured meshes:

- Blocks: complex geometry
- Data: defined on block
- Loop: over block
- Stencil: FD, FV
- Halos: exchange



Domain specific language!

Brief review of OP development

OPLUS

1994 Fortran PVM Unstructured

- OPlus developed as a flexible parallel library for unstructured grids on distributed memory systems using PVM (see Crumpton & Giles)

OP2

2011 C/C++ MPI CUDA… Unstructured

- OP2 is second iteration of OPlus library but now handles C/C++ and use of Nvidea GPUs

OPS

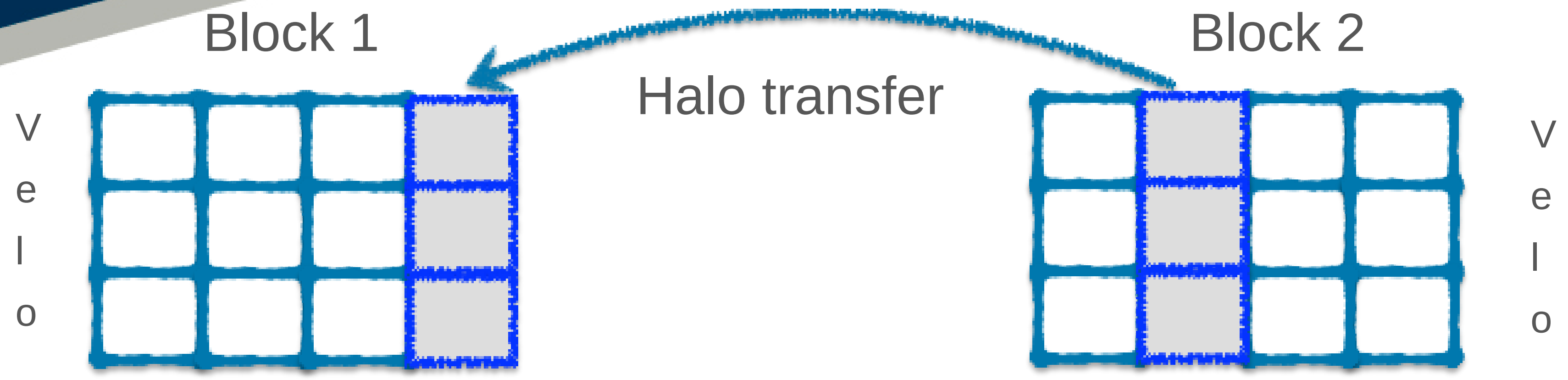
2013 C/C++ MPI CUDA… Structured

- OPS takes the abstraction idea and applies it to structured grid problems

Multigrid aircraft computations using the Oplus parallel library,
P.I. Crumpton and M.B. Giles, Parallel CFD 1995, Kyoto, Japan, pp. 339-346

Daresbury Laboratory
Science & Technology
Facilities Council

Ops: Data abstraction

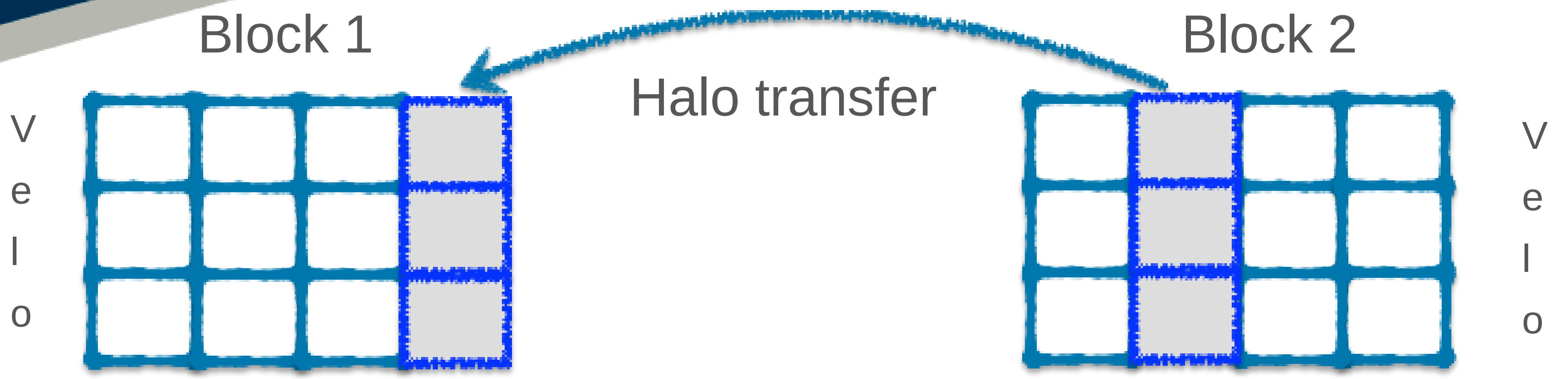


```
ops_block* grid2D;
grid2D = new ops_block[block_num];
grid2D[ib] = ops_decl_block(2, “grid2D’ib’”);
```

```
ops_dat* velo;
velo = new ops_dat[block_num];
velo[ib]=ops_decl_dat(grid2D[ib],2, size,base,d_m,d_p,
                          (Real*)temp, “double”, “velo’ib’”);
```

```
ops_halo* halos;
ops_halo_group f_halos;
halos[1] = ops_decl_halo(velo[0], velo[1], halo_iter, base_from, base_to, dir, dir);
f_halos = ops_decl_halo_group(halo_num, halos);
ops_halo_transfer(f_halos);
```

OPS: Parallel loop and stencil



```
void collision(const Real *f, const Real *feq, Real* fcoll, const Real*
omega) {
    for (int l = 0; l < nc; l++) {
        fcoll[OPS_ACC_MD2(l, 0, 0)] = (1 - (*omega)) * f[OPS_ACC_MD0(l,
        0, 0)] + (*omega) * feq[OPS_ACC_MD1(l, 0, 0)];
    }
}
```

```
ops_par_loop(collision, "collision", grid2D[0], 2, iter_range_coll,
                ops_arg_dat(f[0], nc, S2D_00, "double", OPS_READ),
                ops_arg_dat(feq[0], nc, S2D_00, "double", OPS_READ),
                ops_arg_dat(fcoll[0], nc, S2D_00, "double", OPS_WRITE),
                ops_arg_gbl(&omega, 1, "double", OPS_READ));
```

Work flow

Writing code using OPS API

↓ Use provided function calls and then test code in non-optimised serial and MPI mode

OPS source-to-source translator

↓ Platform specified and optimised code

Conventional compiling

↓ NVCC, mpicxx …

Run…?!

Lattice Boltzmann model

$$\frac{\partial f_\alpha}{\partial t} + c_{\alpha,i}\frac{\partial f_\alpha}{\partial x_i} = -\frac{1}{\tau}\left(f_\alpha - f_\alpha^{+}\right) + g_\alpha$$

Stream-Collision scheme

Easy to understand

Easy to parallelise

Particularly suitable for NS equations

Other schemes can be implemented

Collision

$$f_\alpha(\boldsymbol{x}, t+dt) = f_\alpha(\boldsymbol{x}, t) + \frac{1}{\tau}\left[f_\alpha^{eq}(\boldsymbol{x}, t) - f_\alpha(\boldsymbol{x}, t)\right]$$

Stream

$$f_\alpha(\boldsymbol{x} + \mathbf{c}_\alpha dt, t+dt) = f_\alpha(\boldsymbol{x}, t+dt)$$

Performance and early results

Problem: 2D Taylor-Green vortex

Grid size: 4096×4096

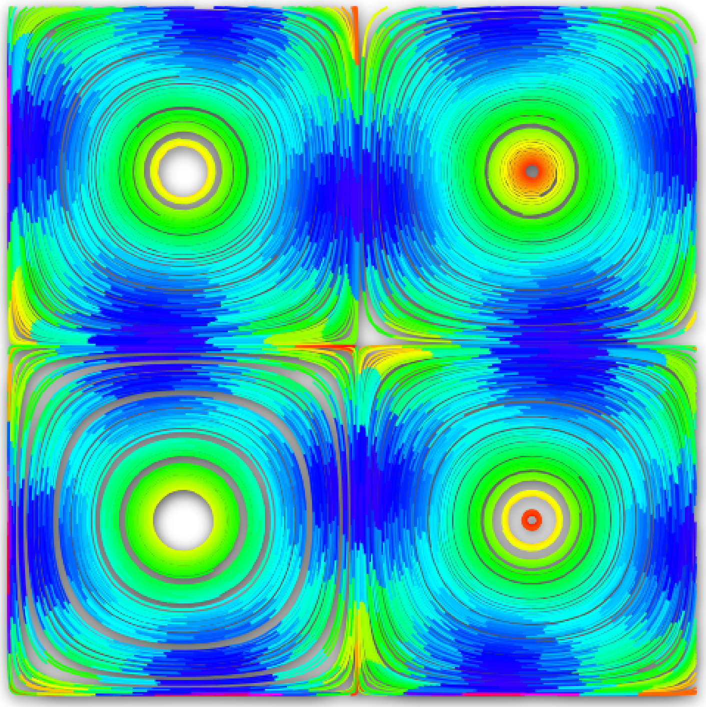
Total time step : 500

Lattice: D2Q9

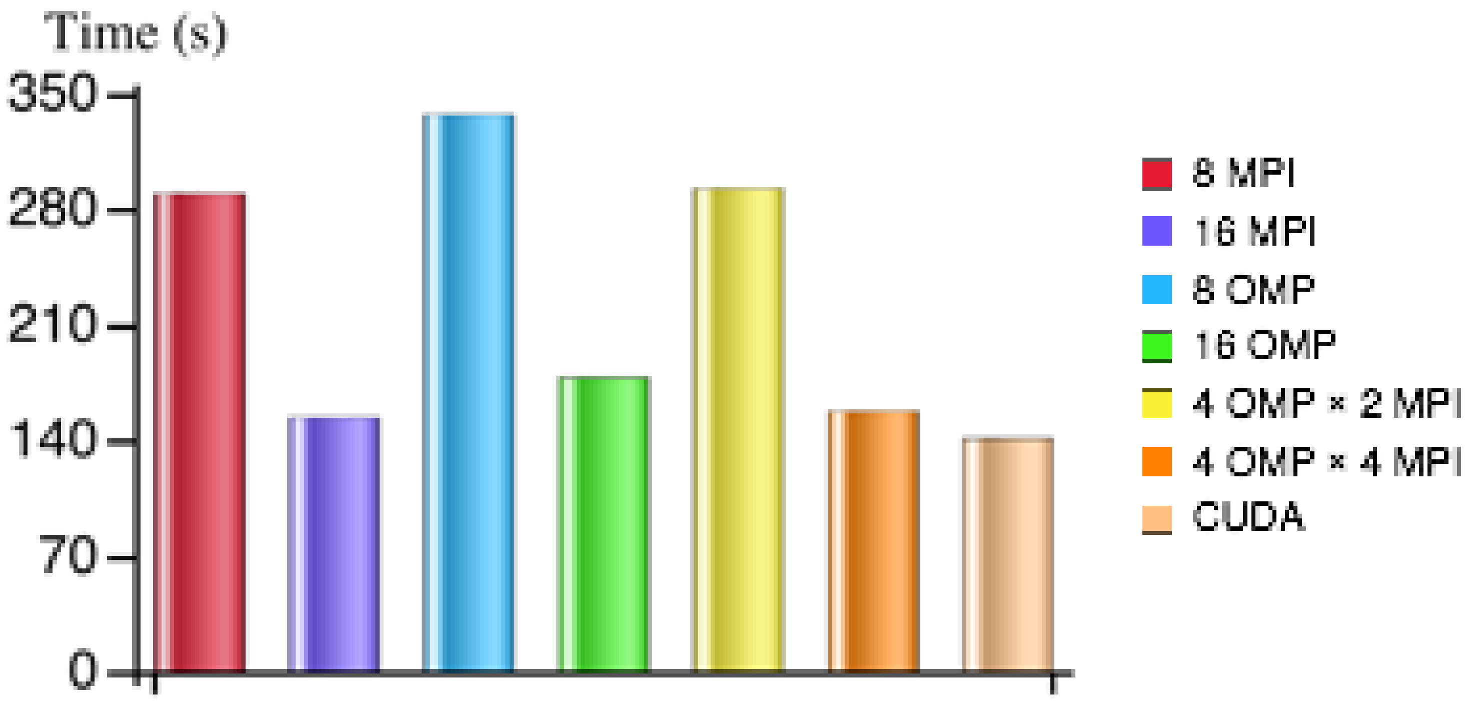
Hardware

Ivy Bridge E5 (iDataplex)

Power8



Performance and early results



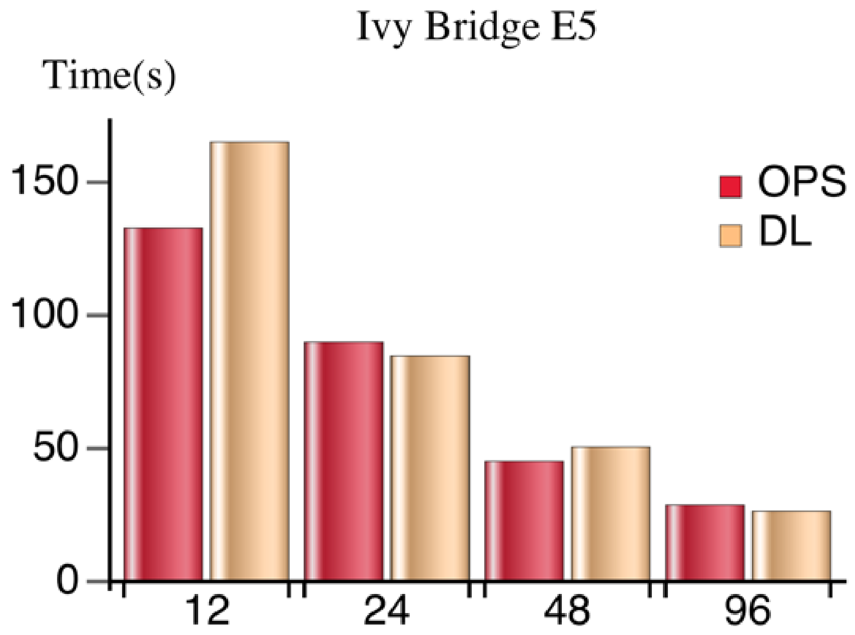
Power8

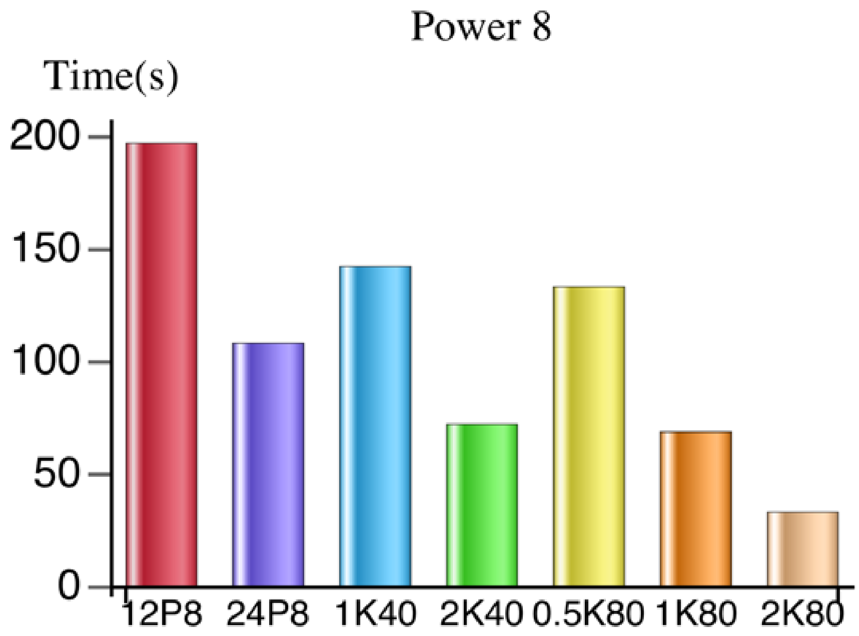
Compiler: IBM XL C/C++ for Linux, V13.1.2 (5725-C73, 5765-J08)

CUDA compilation tools, release 7.0, V7.0.27

MPI: MPICH-3.1.4, built with GCC

Performance and early results





iDataplex

Intel C/C++ 15.2.164

Intel MPI 5.03

Power8

IBM XL C/C++ 13.1.2

Cuda 7.0, V7.0.27 (K40)

Cuda 7.5, V7.5.21 (K80)

OpenMPI

Performance and early results

Energy consumption

Cores/GPU	DL (KWh)	OPS (KWh)
12	0.0072	0.0061
24	0.00397	0.00477
48	0.0028	0.00371
1 K40	-	0.00931
1 K80		0.00459

CPU consumption is measured on the Intel Ivy Bridge E5 system with the consideration of idle cores

Summary and conclusions

OPS provides a concise high-level abstraction

OPS can reduce complexity and speed up parallel application development

OPS shows decent performance

Users need a general understanding of CUDA, OpenACC,…

To adapt to a new hardware, we mainly need to update the translator

Knowledge of Python and regular expression is necessary

Although not a new concept, modern h/w complexity makes OPS or similar methodology look attractive…….but…..

Acknowledgements

- This work was supported through the U.K. Engineering and Physical Sciences Research Council (EPSRC) under Grant EP/K038494/1 “Future-proof massively-parallel execution of multi-block applications”

- For more information on this work, please visit http://oerc.ox.ac.uk/projects/ops

Daresbury Laboratory
Science & Technology
Facilities Council

High-level Abstraction for Block Structured Applications: A lattice Boltzmann Exploration

Oxford e-Research Centre

Jianping Meng, Xiao-Jun Gu, **David R. Emerson**, Gihan Mudalige, István Reguly and Mike B Giles

University of BRISTOL

UNIVERSITY OF Southampton

Scientific Computing Department
STFC Daresbury Laboratory
Daresbury, Warrington, WA4 4AD, UK

david.emerson@stfc.ac.uk

EMiT Emerging Technology Conference 2016

2-3 June 2016, Barcelona

Parallel Computational Fluid Dynamics

University of Strathclyde Glasgow

29th International Conference on Parallel Computational Fluid Dynamics

15-17 May, 2017

Technology and Innovation Centre
University of Strathclyde, Glasgow
www.parcfd.org

Parallel CFD is the annual conference series dedicated to the discussion of recent developments and applications of parallel computing in the field of CFD and related disciplines

Chairman: Professor Dimitris Drikakis
Co-chairmen: Professor David Emerson and Dr. Marco Fossati

Extended abstract submission deadline:
5 February 2017

Supported by:

Intelligent Light   NVIDIA

GLASGOW CITY MARKETING BUREAU