Evaluating OpenMP's Effectiveness in the Many-Core Era

Prof Simon McIntosh-Smith

HPC Research Group

simonm@cs.bris.ac.uk

University of BRISTOL

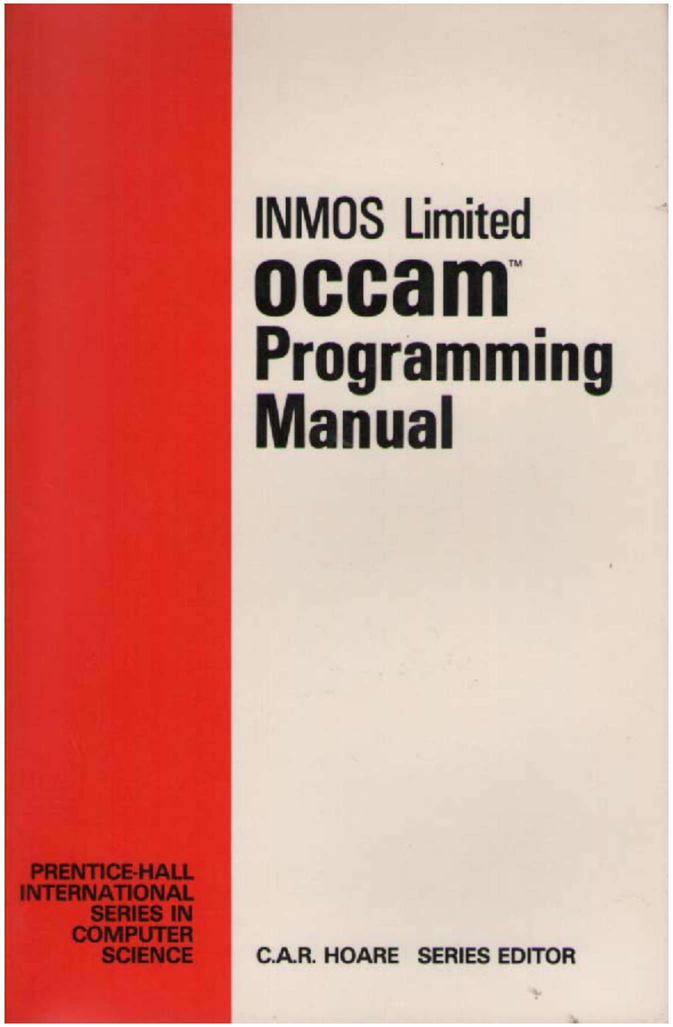
Inverness
Aberdeen
SCOTLAND
Dundee
Edinburgh
Glasgow
United Kingdom
NORTHERN IRELAND
Isle of Man
Galway
Ireland
Dublin
Limerick
Cork
Liverpool
Manchester
ENGLAND
WALES
Cambridge
Oxford
London
Bristol
Cardiff
Brighton
Southampton
Plymouth
English Channel
Google
Guernsey

Bristol, UK

- 10th largest city in UK
- Aero, finance, chip design
- HQ for Cray EMEA
- 100 miles west of London

University of BRISTOL

University of BRISTOL



INMOS Limited

occam™

Programming Manual

PRENTICE-HALL INTERNATIONAL SERIES IN COMPUTER SCIENCE

C.A.R. HOARE SERIES EDITOR

University of BRISTOL

Bristol's long HPC history

inmos

meiko

ClearSpeed

PIXELFUSION

Quadrics

A Finmeccanica Company

gnodal

CRAY

THE SUPERCOMPUTER COMPANY

University of BRISTOL

What does my group do?

- **Performance portability**
  - Programming model evaluations
  - Code design strategies
  - Hardware evaluations
  - "Cross-X", where X = vendor, language, …

- **Fault tolerance**
  - Application-based fault tolerance
  - Reliable computing on unreliable hardrware

PERFORMANCE PORTABILITY

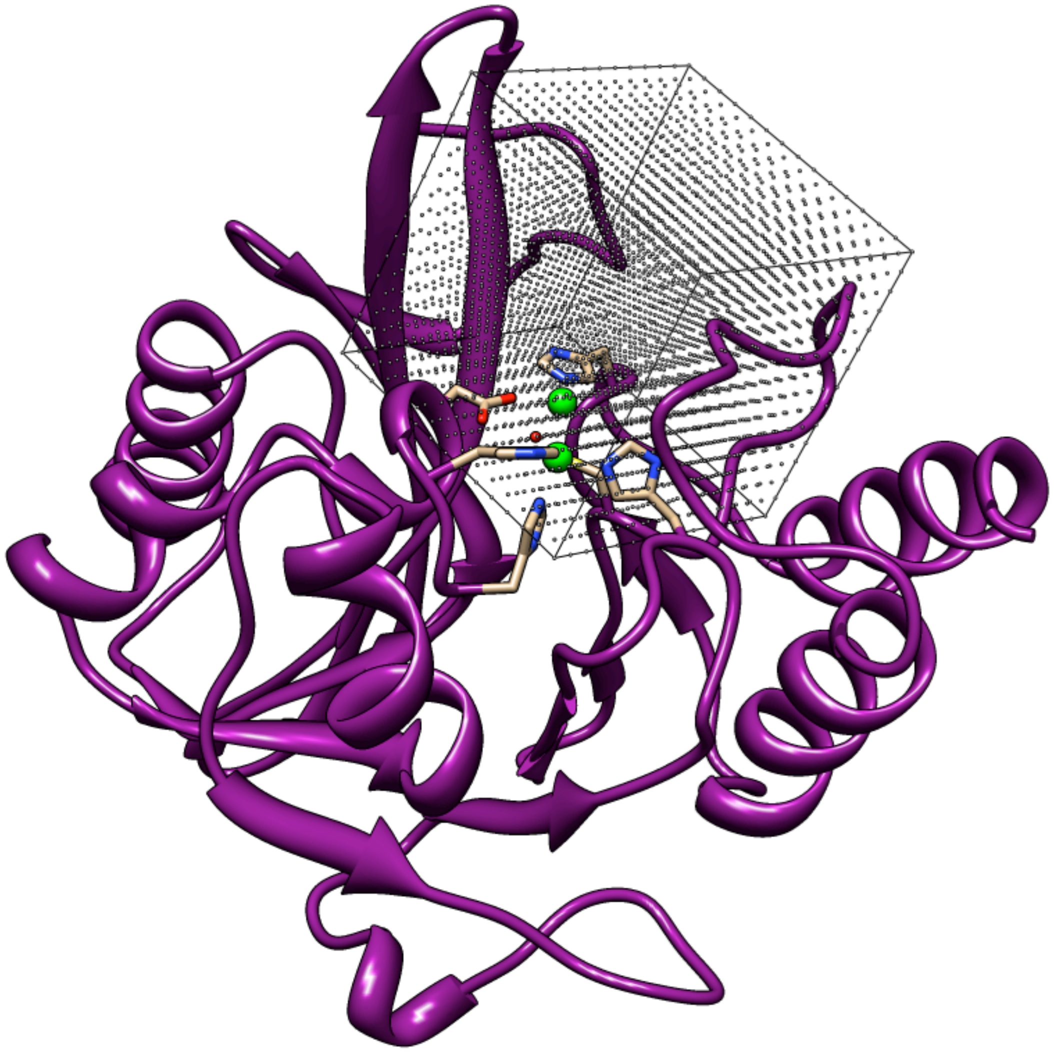
BUDE – MOLECULAR DOCKING (2013)

What is BUDE?

- **B**ristol **U**niversity **D**ocking **E**ngine
  - Dr Richard Sessions, PI (Biochemistry)
- *In silico* virtual drug screening by docking
- Employs a genetic algorithm-based search of the six degrees of freedom in the arrangement of the protein and drug molecules to reduce the search space

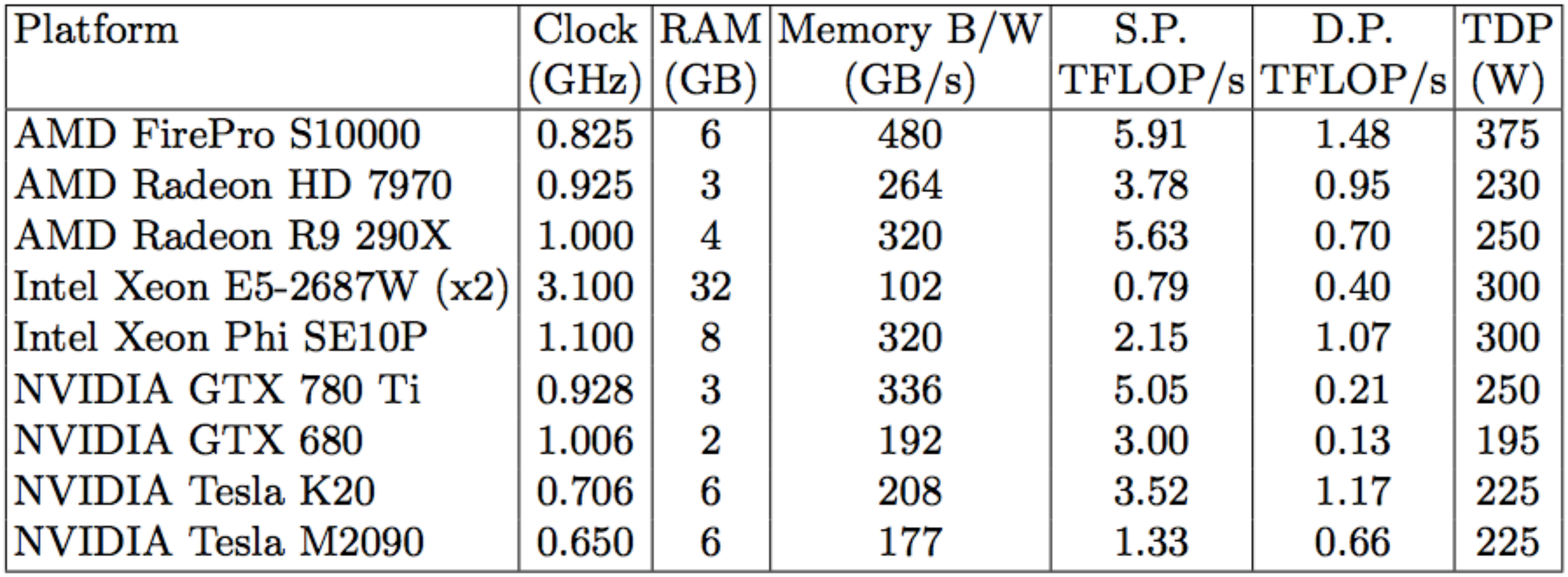
University of BRISTOL

BUDE protein-ligand docking



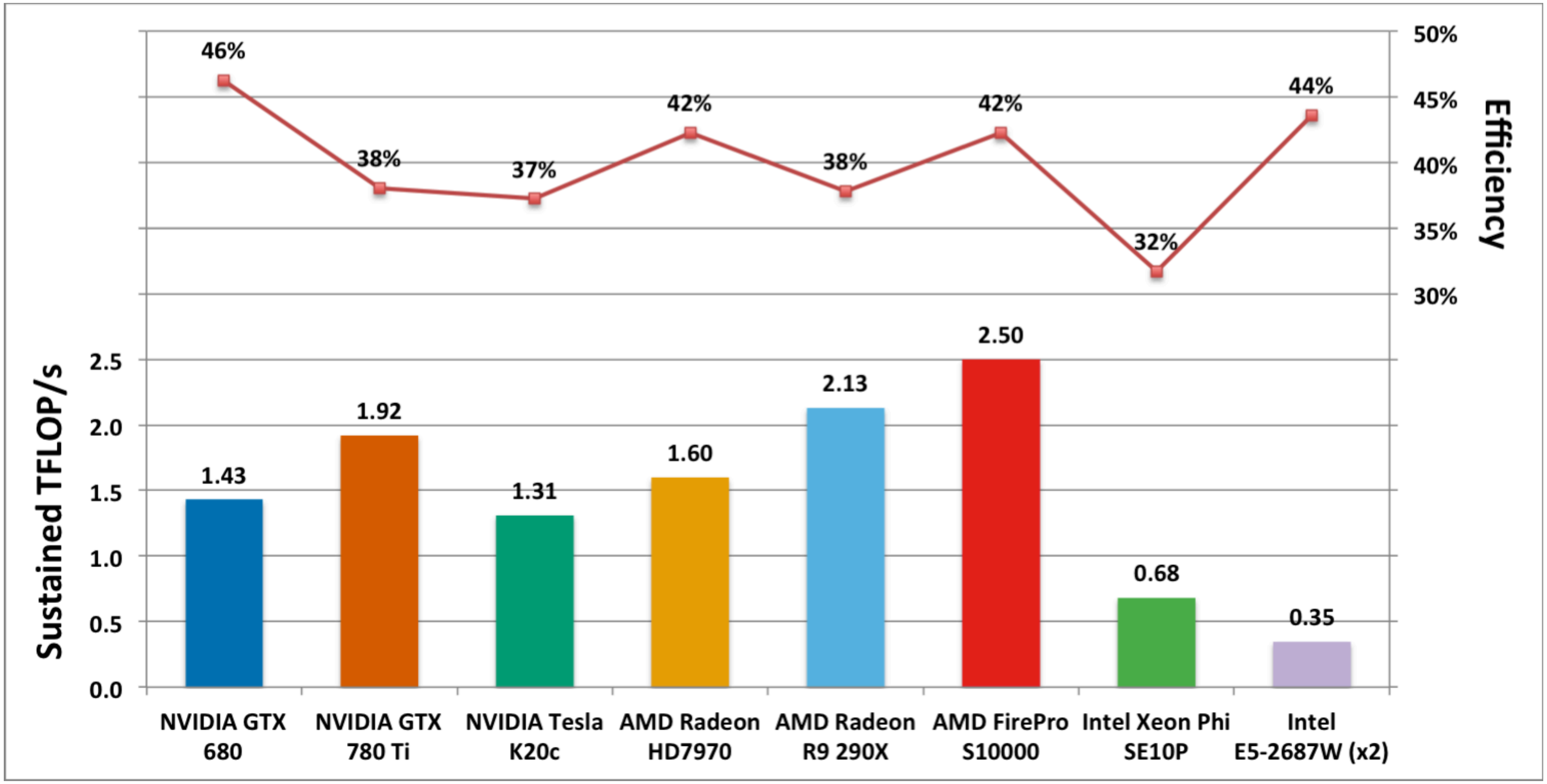
University of BRISTOL

Target hardware



Platform	Clock (GHz)	RAM (GB)	Memory B/W (GB/s)	S.P. TFLOP/s	D.P. TFLOP/s	TDP (W)
AMD FirePro S10000	0.825	6	480	5.91	1.48	375
AMD Radeon HD 7970	0.925	3	264	3.78	0.95	230
AMD Radeon R9 290X	1.000	4	320	5.63	0.70	250
Intel Xeon E5-2687W (x2)	3.100	32	102	0.79	0.40	300
Intel Xeon Phi SE10P	1.100	8	320	2.15	1.07	300
NVIDIA GTX 780 Ti	0.928	3	336	5.05	0.21	250
NVIDIA GTX 680	1.006	2	192	3.00	0.13	195
NVIDIA Tesla K20	0.706	6	208	3.52	1.17	225
NVIDIA Tesla M2090	0.650	6	177	1.33	0.66	225

BUDE results

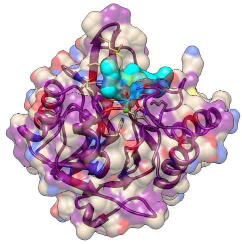


"High Performance *in silico* Virtual Drug Screening on Many-Core Processors", S. McIntosh-Smith, J. Price, R.B. Sessions, A.A. Ibarra, IJHPCA 2014
DOI: 10.1177/1094342014528252

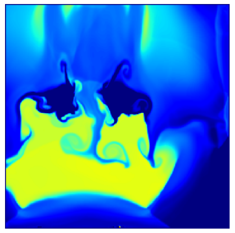
University of BRISTOL

Performance portability

- **BUDE**'s OpenCL implementation proved to be highly performance portable
  - Compute intensive, N-body / Monte Carlo



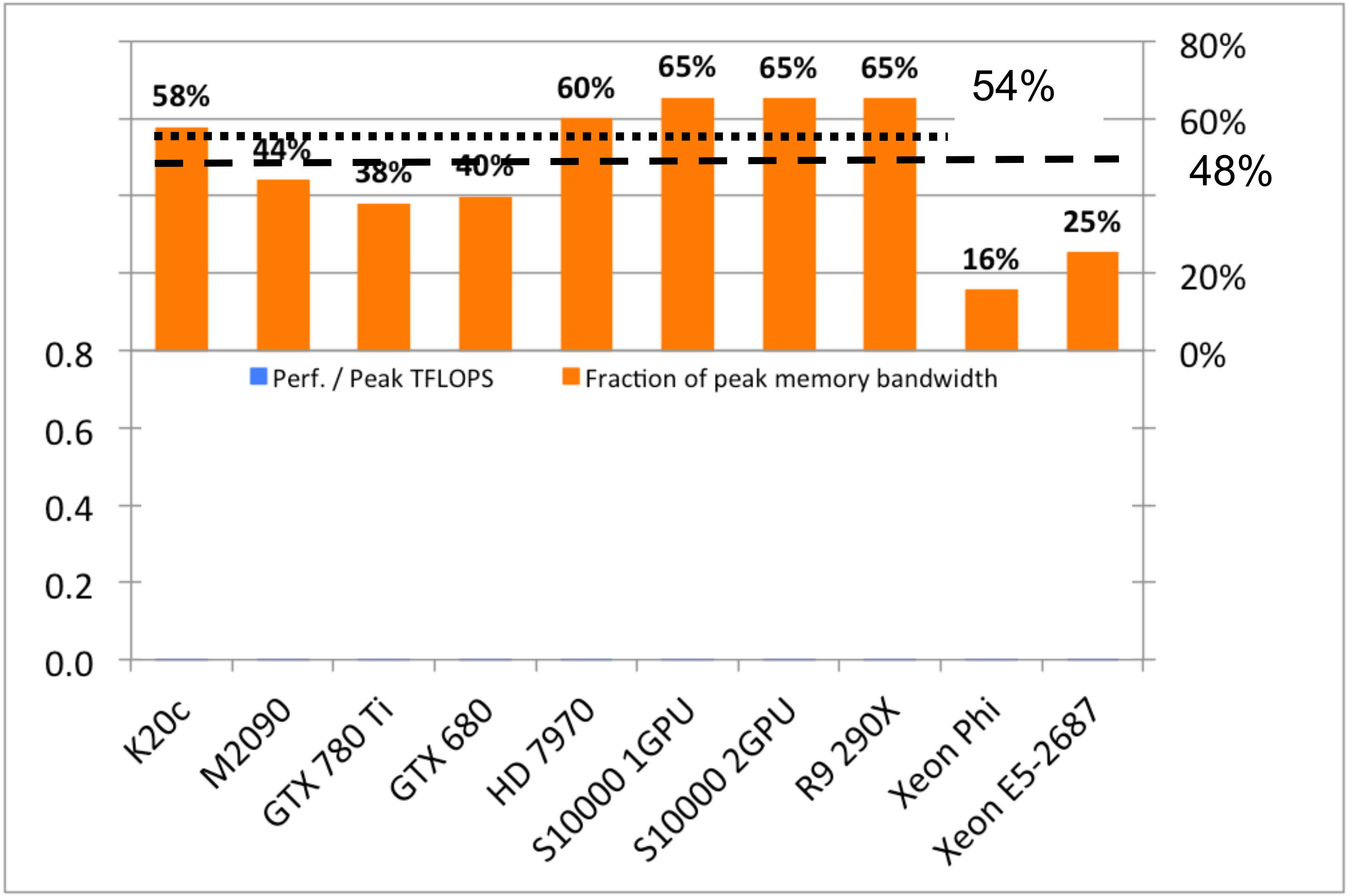
- Looked at bandwidth intensive codes next, such as the **CloverLeaf** structured grid hydrodynamics mini-app



CloverLeaf: A Lagrangian-Eulerian hydrodynamics benchmark

- Solves the compressible Euler equations, which describe the conservation of energy, mass and momentum in a system
- These equations are solved on a Cartesian grid in 2D with second-order accuracy, using an explicit finite-volume method
- Optimised parallel versions exist in **OpenMP, MPI, OpenCL, OpenACC, CUDA and Co-Array Fortran**

Results – sustained bandwidth



S.N. McIntosh-Smith, M. Boulton, D. Curran, & J.R. Price, “On the performance portability of structured grid codes on many-core computer architectures”, ISC, Leipzig, June 2014. DOI: 10.1007/978-3-319-07518-1_4

(Ninja level) performance portability techniques

- Use a platform portable parallel language
- Aim for 80-90% of optimal
- Avoid platform-specific optimisations
- ***Most*** optimisations make the code faster on ***most*** platforms
- This was only possible in OpenCL in 2014…

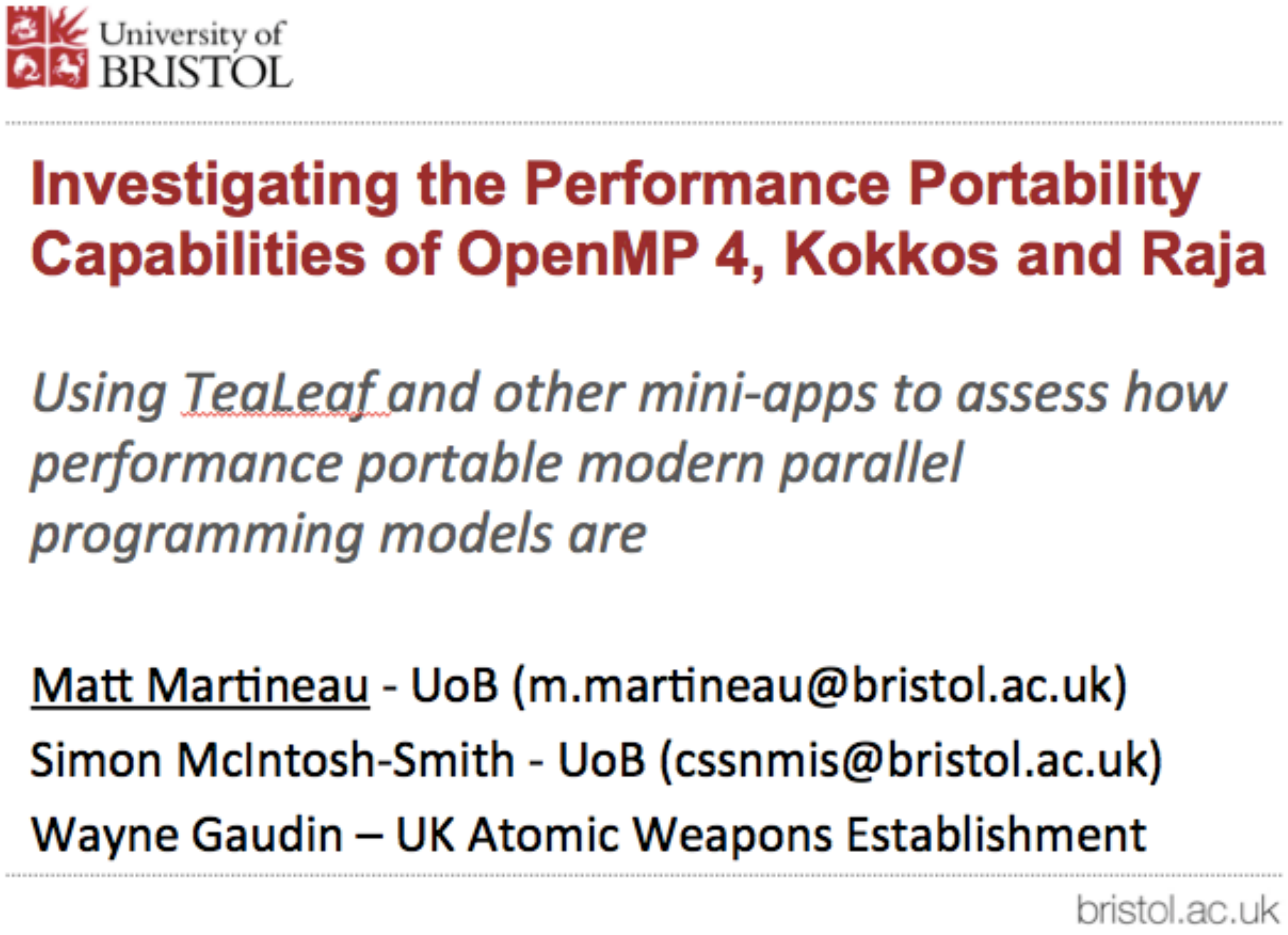
HIGHER-LEVEL PERFORMANCE PORTABILITY (2014-)

Moving on up

- Low-level programming in OpenCL or CUDA is all very well …
- ... But we don't expect most scientific codes to be re-written in these languages

- What are the emerging options?
  - Directive-based: **OpenMP 4.x, OpenACC, OmpSs, ...**
  - C++ based: **RAJA, Kokkos, SYCL, ...**

University of BRISTOL



Investigating the Performance Portability Capabilities of OpenMP 4, Kokkos and Raja

Using TeaLeaf and other mini-apps to assess how performance portable modern parallel programming models are

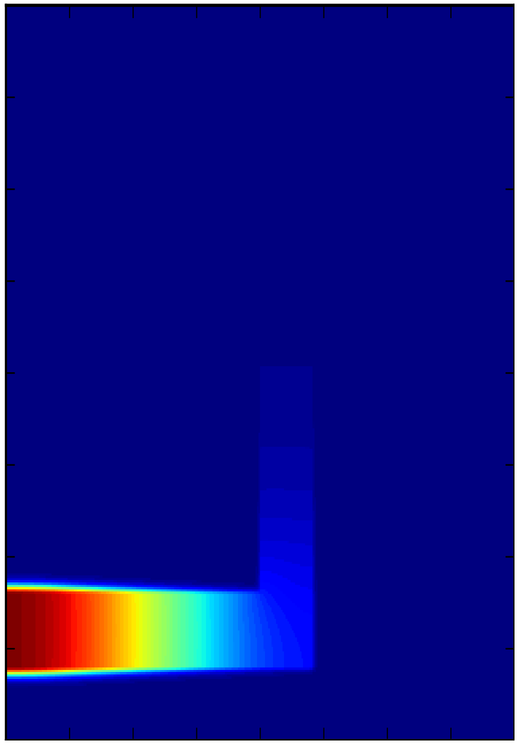
Matt Martineau - UoB (m.martineau@bristol.ac.uk)
Simon McIntosh-Smith - UoB (cssnmis@bristol.ac.uk)
Wayne Gaudin – UK Atomic Weapons Establishment

bristol.ac.uk

DOE performance portability workshop, Arizona, April 2016.

TeaLeaf – Heat Conduction

- Mini-app from Mantevo suite of benchmarks



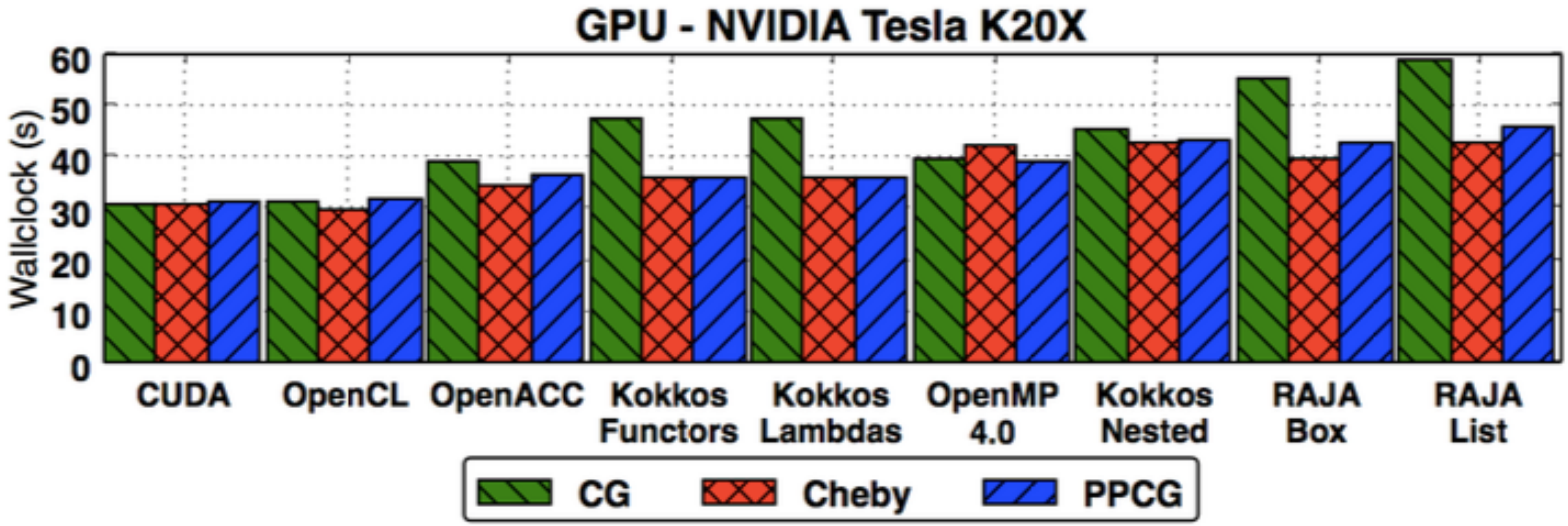
- Implicit, sparse, matrix-free solvers on structured grid
  - Conjugate Gradient (CG)
  - Chebyshev
  - Preconditioned Polynomial CG (PPCG)
- Memory bandwidth bound
- Good strong and weak scaling on Titan & Piz Daint

The Performance Experiment

- Performance tested on **CPU**, **GPU**, and **KNC**
- Single node only (multi-node scaling proven)
- All ports were optimised as much as possible, while ensuring performance portability
- Solved *4096x4096* problem, the point of mesh convergence, for *single iteration*

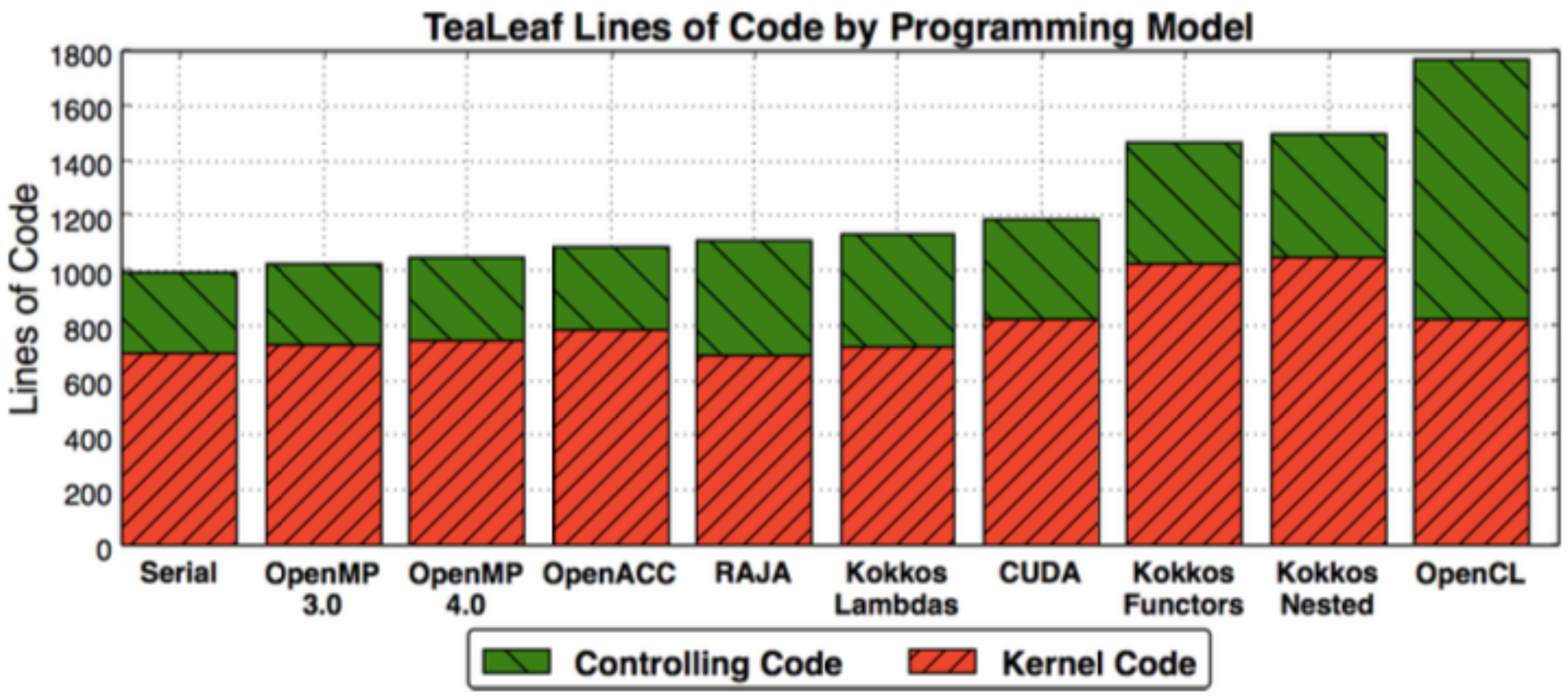
University of BRISTOL

TeaLeaf – GPU



All the programming models get to performance within 25% of OpenCL / CUDA hand-optimised code

TeaLeaf lines of code



Martineau, M., McIntosh-Smith, S. Gaudin, W., *Assessing the Performance Portability of Modern Parallel Programming Models using TeaLeaf, 2016, CC-PE*

TeaLeaf conclusions

- RAJA and Kokkos both looking promising
  - For GPU (NVIDIA) and CPU (Intel, IBM)
  - What about other architectures though?
    - AMD GPUs, ARM CPUs, …
  - **Big question is**: *who maintains these in the long-term?*

- **OpenMP 4.x** also looking good for GPUs
  - Still lots of Fortran out there

University of BRISTOL

What we did next

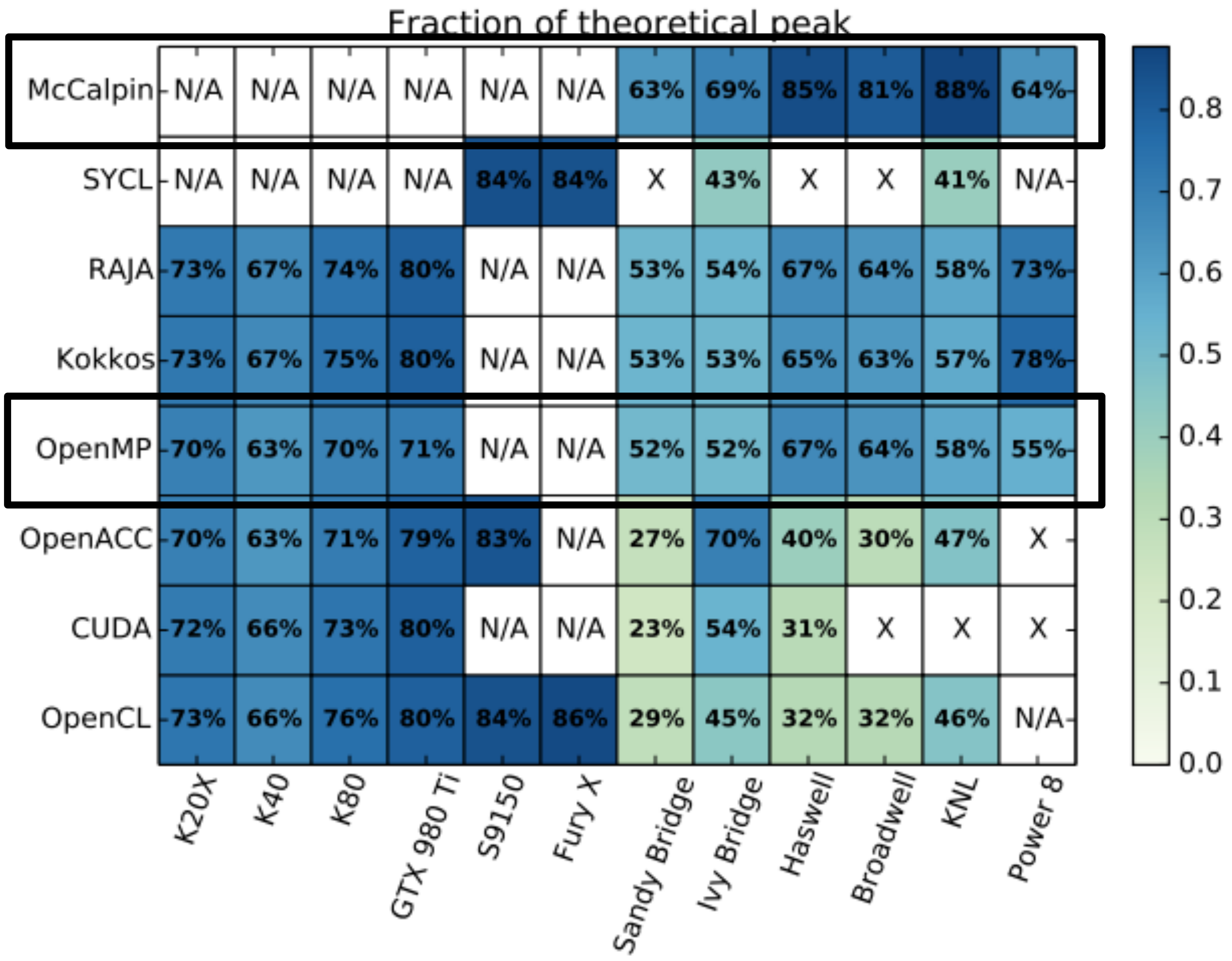
- Based on these early successes we decided to do a ***detailed assessment of OpenMP 4.x compiler implementations***
- Started with something simple – a modern port of STREAM to OpenMP 4.x and other parallel programming languages
- Then looked at a range of codes and mini-apps
- See **Matt Martineau**'s talk on Friday for all the details: **"Pragmatic Performance Portability with OpenMP 4.x"**

OpenMP 4: STREAM Triad

```
template <class T>
void OMP45Stream<T>::triad()
{
  const T scalar = 0.3;

  unsigned int array_size = this->array_size;
  T *a = this->a;
  T *b = this->b;
  T *c = this->c;
  #pragma omp target teams distribute parallel for simd \
    map(to: a[0:array_size], b[0:array_size], c[0:array_size])
  for (int i = 0; i < array_size; i++)
  {
    a[i] = b[i] + scalar * c[i];
  }
}
```

GPU-STREAM 2 performance



	K20X	K40	K80	GTX 980 Ti	S9150	Fury X	Sandy Bridge	Ivy Bridge	Haswell	Broadwell	KNL	Power 8
McCalpin	N/A	N/A	N/A	N/A	N/A	N/A	63%	69%	85%	81%	88%	64%
SYCL	N/A	N/A	N/A	N/A	84%	84%	X	43%	X	X	41%	N/A
RAJA	73%	67%	74%	80%	N/A	N/A	53%	54%	67%	64%	58%	73%
Kokkos	73%	67%	75%	80%	N/A	N/A	53%	53%	65%	63%	57%	78%
OpenMP	70%	63%	70%	71%	N/A	N/A	52%	52%	67%	64%	58%	55%
OpenACC	70%	63%	71%	79%	83%	N/A	27%	70%	40%	30%	47%	X
CUDA	72%	66%	73%	80%	N/A	N/A	23%	54%	31%	X	X	X
OpenCL	73%	66%	76%	80%	84%	86%	29%	45%	32%	32%	46%	N/A

OpenMP 3.0

On those supported target architectures,

OpenMP 4.x

performance

Deakin, T., Price, J., Martineau, M., McIntosh-Smith, S., *GPU-STREAM v2.0 Benchmarking the achievable memory bandwidth of many-core processors across diverse parallel programming models, P^{3}MA, ISC'16*

Current Compiler Support

- **Intel** began support for OpenMP 4.0 targeting the Xeon Phi coprocessors in 2013.
- **Cray** provided the first vendor implementation targeting NVIDIA GPUs in late 2015. Now supports OpenMP 4.0, and a subset of OpenMP 4.5.
- **IBM** has recently completed a compiler implementation using Clang, that fully supports OpenMP 4.5. This is being introduced into the Clang main trunk.
- **GCC 6.1** introduced full support for OpenMP 4.5, and can target Intel Xeon Phi, or HSA enabled AMD GPUs. Still very immature

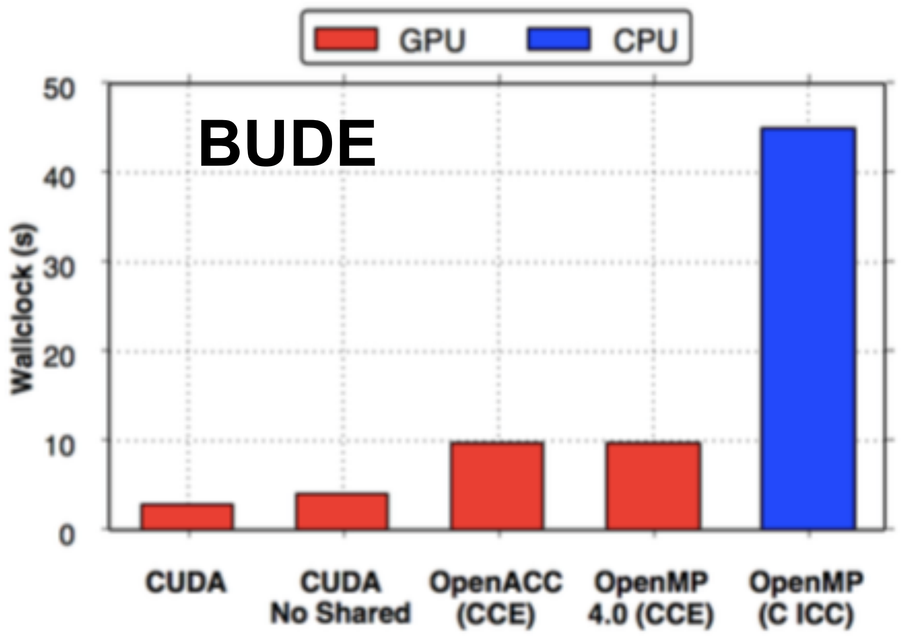
University of BRISTOL

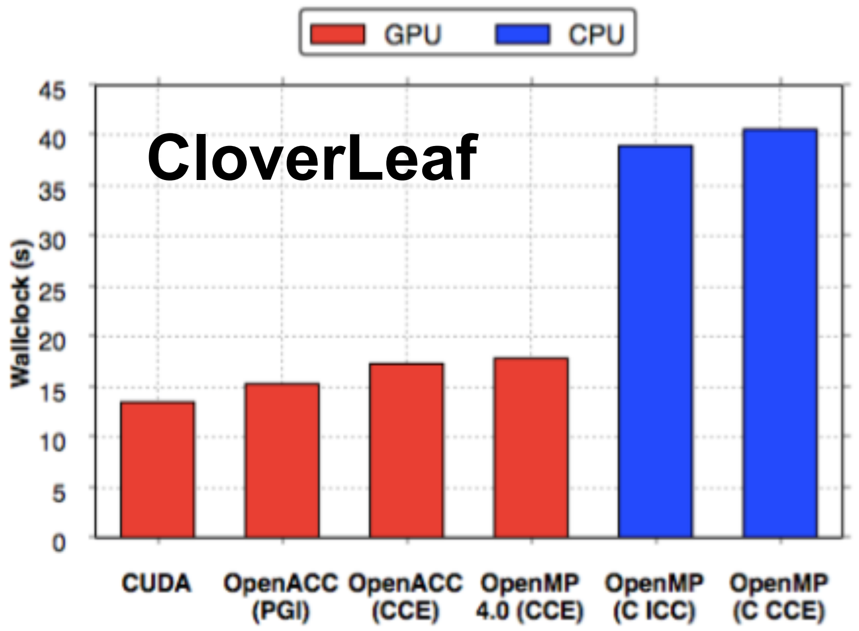
Performance?

- To test performance we use a mixture of synthetic benchmarks and mini-apps.
- We compare against device-specific code written in **OpenMP 3.0** and **CUDA**.
- We use OpenMP 4.x to run on *a diverse range of currently supported architectures*.
- Our initial expectations were low for GPUs…

* CCE 8.4.3, ICC 15.0.3, PGI 15.01, CUDA 7.0 on an NVIDIA® K20X, and Intel® Xeon® Haswell 16 Core Processor (E5-2698 v3 @ 2.30GHz)

Performance?





Really need to target local/shared memory for BUDE – OpenMP 5?

OpenMP 4.0 nearly as fast as hand-optimised CUDA with CCE

Martineau, M., McIntosh-Smith, S. Gaudin, W., *Evaluating OpenMP 4.0's Effectiveness as a Heterogeneous Parallel Programming Model, 2016, HIPS'16*

University of BRISTOL

Achieving good OpenMP 4.x performance on GPUs

- Our findings so far:
  - *You* ***can*** *achieve good performance with OpenMP 4.x.*

- Achieve this by:

  - Keeping data resident on the device for the greatest possible time.

  - Collapsing loops with the **collapse** clause, creating a large enough iteration space to saturate a device such as a GPU.

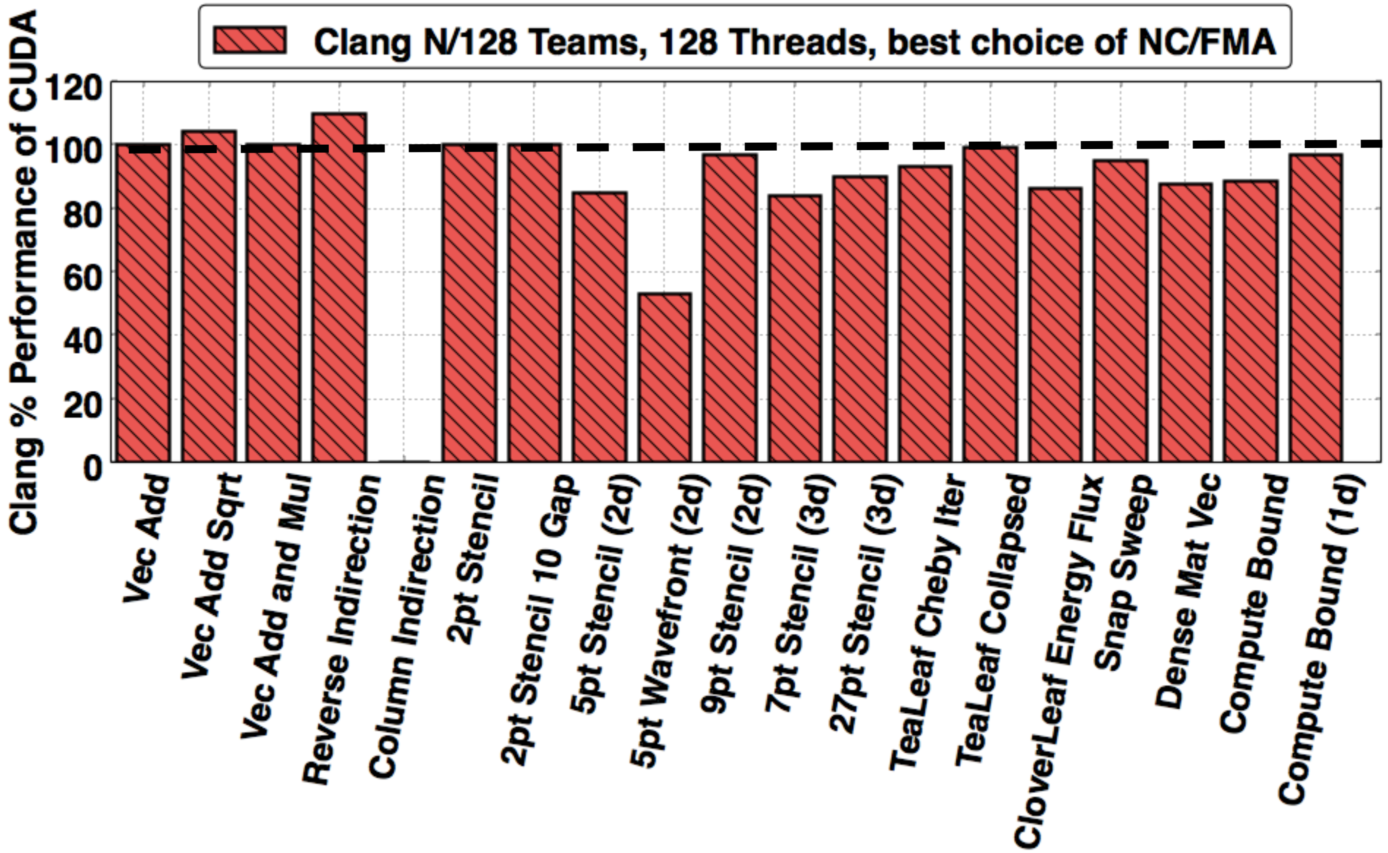
  - Using the **simd** directive to vectorize inner loops.

  - Using **schedule(static, 1)** for coalescence (obsolete).

  - Using profilers such as *nvprof*.

* Clang copy https://github.com/clang-ykt, CUDA 8.0, NVIDIA K40m

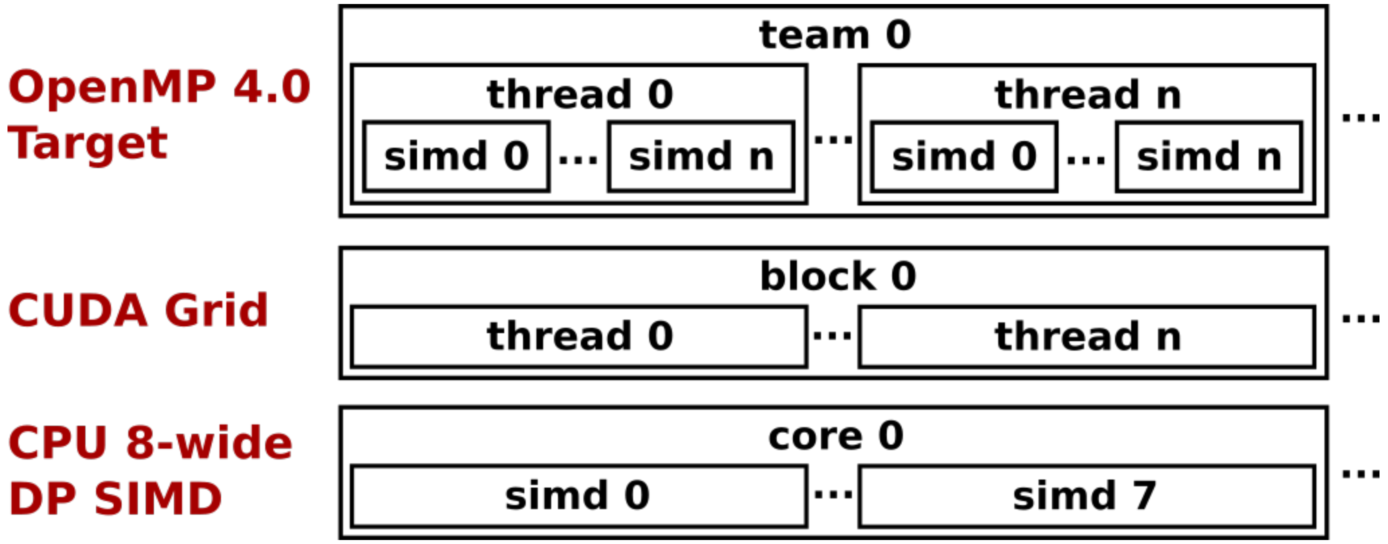
Can we do even better?



Through extensive tuning of the clang/llvm compiler implementation we achieved CloverLeaf performance within 9% of hand optimized CUDA code…

Martineau, M., Bertolli, C., McIntosh-Smith, S., et al. *Broad Spectrum Performance Analysis of OpenMP 4.5 on GPUs, 2016, PMBS'16*

An important observation



OpenMP can express more levels of parallelism than we often need

- Leaves ambiguity when not all levels needed
- How the "simd" directive is implemented now and in the future will have a big impact on success…

Pragmatic portability

```
// CCE targeting NVIDIA GPU
#pragma omp target teams distribute simd
for(...) {
}

// Clang targeting NVIDIA GPU
#pragma omp target teams distribute parallel for schedule(static, 1)
for(...) {
}

// GCC 6.1 target AMD GPU
#pragma omp target teams distribute parallel for
for(...) {
}

// ICC targeting Intel Xeon Phi
#pragma omp target if(offload)
#pragma omp parallel for simd
for(...) {
}
```

Four different styles of pragma for the same kernel is not what anyone wants … (See Martineau's talk for the answer!)

University of BRISTOL

What are the main issues?

- Dealing with the host-device data movement got a lot easier in OpenMP 4.5
- We want to write **performance portable** OpenMP
  - Ideally the same set of pragmas on all hardware platforms (CPUs, GPUs, …)
- **C++** still causing all sorts of performance problems
- Need to be able to exploit the emerging **memory hierarchies** (HBM etc)
- Then there's still multi-device support, heterogeneous computing, dynamic load balancing across devices...

University of BRISTOL

Summary

- Quite a few OpenMP 4.x implementations now emerging (Intel, Cray, PGI, gcc, clang/llvm…)
- Levels of maturity are quite mixed
- Demonstrated OpenMP 4.x can achieve GPU performance similar to OpenCL/CUDA
- Can also achieve a reasonable degree of performance portability, although need to jump through hoops with the pragmas
- ***The signs are promising for OpenMP 4.x!***

University of BRISTOL

Performance portability refs

- **On the performance portability of structured grid codes on many-core computer architectures**
  S.N. McIntosh-Smith, M. Boulton, D. Curran, & J.R. Price
  ISC, Leipzig, June 2014. DOI: 10.1007/978-3-319-07518-1_4
- **Assessing the Performance Portability of Modern Parallel Programming Models using TeaLeaf**
  Martineau, M., McIntosh-Smith, S. & Gaudin, W.
  Concurrency and Computation: Practice and Experience (April 2016), to appear
- **GPU-STREAM v2.0: Benchmarking the achievable memory bandwidth of many-core processors across diverse parallel programming models**
  Deakin, T. J., Price, J., Martineau, M. J. & McIntosh-Smith, S. N.
  First International Workshop on Performance Portable Programming Models for Accelerators (P^3MA), ISC 2016
- https://github.com/UoB-HPC/

University of BRISTOL

For related software and papers

See: http://uob-hpc.github.io

GPU-STREAM:
 https://github.com/UoB-HPC/GPU-STREAM
CloverLeaf:
 https://github.com/UoB-HPC/CloverLeaf-OpenMP4
TeaLeaf:
 https://github.com/UoB-HPC/TeaLeaf
SNAP:
 https://github.com/UoB-HPC/SNAP_MPI_OpenCL

ADDITIONAL MATERIAL

Abbreviation	System details
K20X	Cray® XC40, NVIDIA® K20X GPU, Cray compilers version 8.5, gnu 5.3, CUDA 7.5
K40	Cray® CS cluster, NVIDIA® K40 GPU, Cray compilers version 8.4, gnu 4.9, CUDA 7.5
K80	Cray® CS cluster, NVIDIA® K40 GPU, Cray compilers version 8.4, gnu 4.9, CUDA 7.5
S9150	AMD® S9150 GPU. Codeplay® copmputeCpp compiler 2016.05 pre-release. AMD-APP OpenCL 1.2 (1912.5)drivers for SyCL. PGI® Accelerator)TM) 16.4 OpenACC
GTX 980 Ti	NVIDA® GTX 980 Ti. Clang-ykt fork of Clang for OpenMP. PGI® Accelerator™ 16.4 OpenACC. CUDA 7.5
Fury X	AMD® Fury X GPU (based on the Fiji architecture).
Sandy Bridge	Intel® Xeon® E5-2670 CPU. Intel® compilers release 16.0. PGI® Accelerator)TM) 16.4 OpenACC and CUDA-x86. Intel® OpenCL runtime 15.1. Codeplay® copmputeCpp compiler 2016.05 pre-release
Ivy Bridge	Intel® Xeon® E5-2697 CPU. Gnu 4.8 for RAJA and Kokkos, Intel® compiler version 16.0 for stream, Intel® OpenCL runtime 15.1. Codeplay® copmputeCpp compiler 2016.05 pre-release.
Haswell	Cray® XC40, Intel® Xeon® E5-2698 CPU. Intel® compilers release 16.0. PGI® Accelerator)TM) 16.3 OpenACC and CUDA-x86. Gnu 4.8 for RAJA and Kokkos
Broadwell	Cray® XC40, Intel® Xeon® E5-2699 CPU. Intel® compilers release 16.0. PGI® Accelerator)TM) 16.3 OpenACC and CUDA-x86. Gnu 4.8 for RAJA and Kokkos
KNL	Intel® Xeon® Phi™7210 (knights landing) Intel® compilers release 16.0. PGI® Accelerator)TM) 16.4 OpenACC with target specified as AVX2.
Power 8	IBM® Power 8 processor with the XL 13.1 compiler.

University of

Deakin, T., Price, J., Martineau, M., McIntosh-Smith, S., *GPU-STREAM v2.0 Benchmarking the achievable memory bandwidth of many-core processors across diverse parallel programming models, ISC'16*