Evaluation of a performance portable lattice Boltzmann code using OpenCL

Simon McIntosh-Smith
Dan Curran
Computer Science
University of Bristol

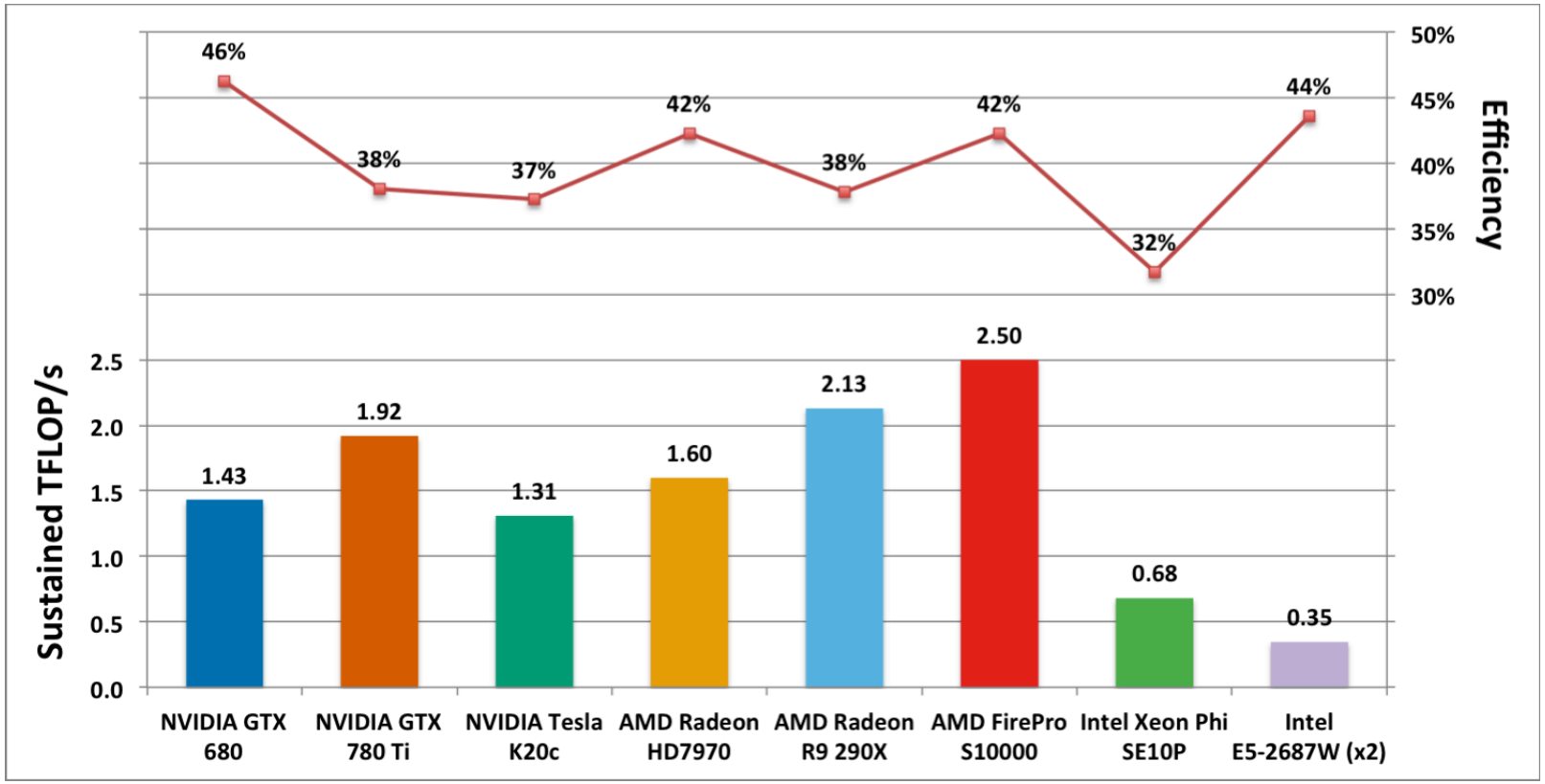
UOB HPC

University of BRISTOL

Twitter: @simonmcs

Motivation

Our BUDE molecular docking code turned out to show strong performance portability:



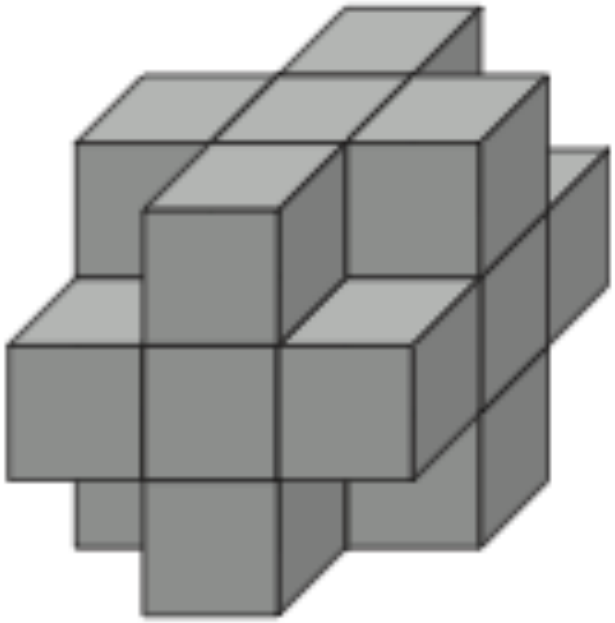
University of BRISTOL

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

Lattice Boltzmann (LBM)

- A versatile approach for solving incompressible flows based on a simplified gas-kinetic description of the Boltzmann equation (used for CFD et al)
- A structured grid algorithm
- Usually **memory bandwidth limited**
- Ports well to most parallel architectures
- We targeted the most widely used variant, D3Q19-BGK

D3Q19-BGK LBM



- To update a cell, need to access 19 of the 27 surrounding cell values in the 3D grid

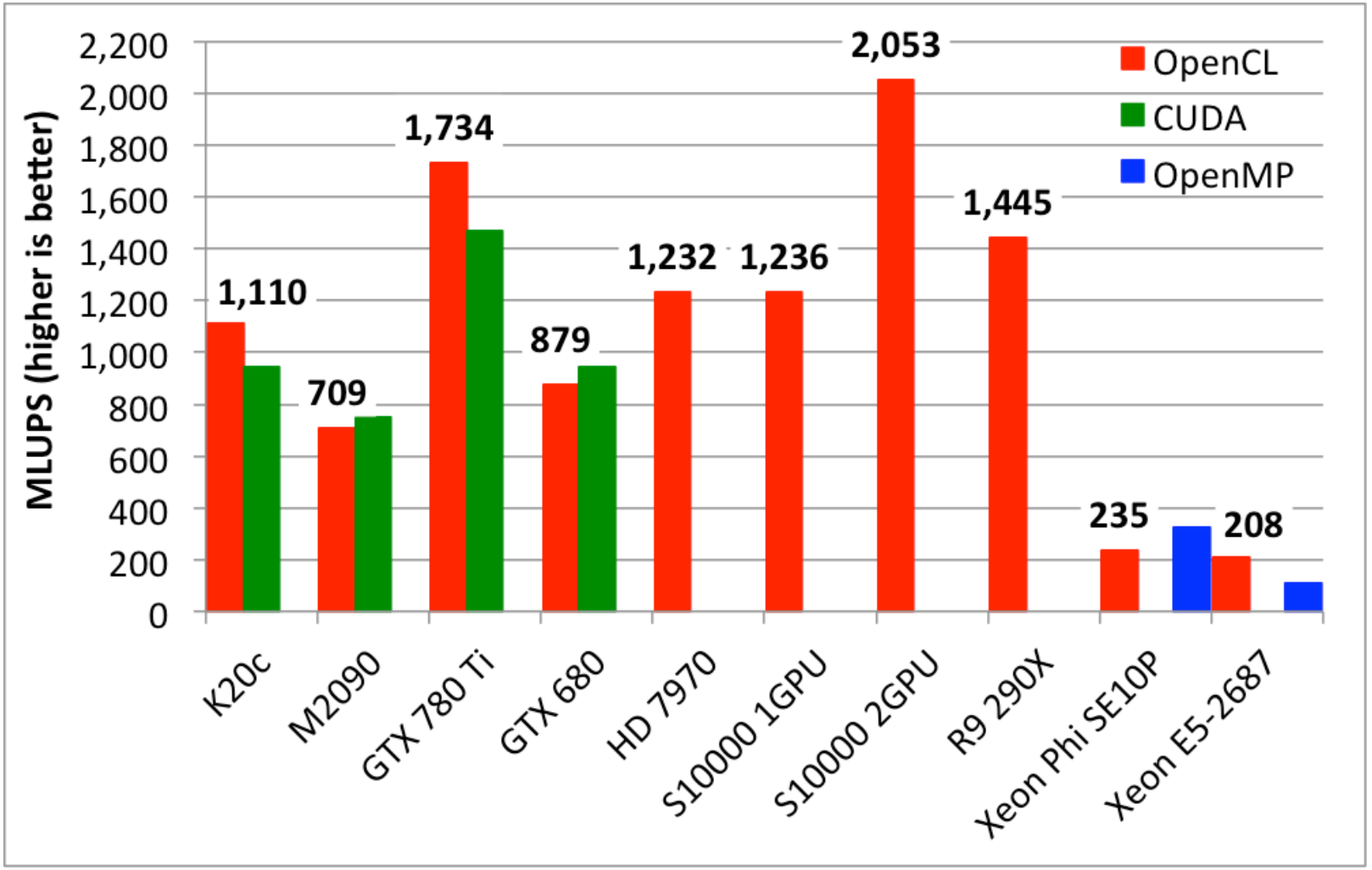
Target platforms

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

Methodology

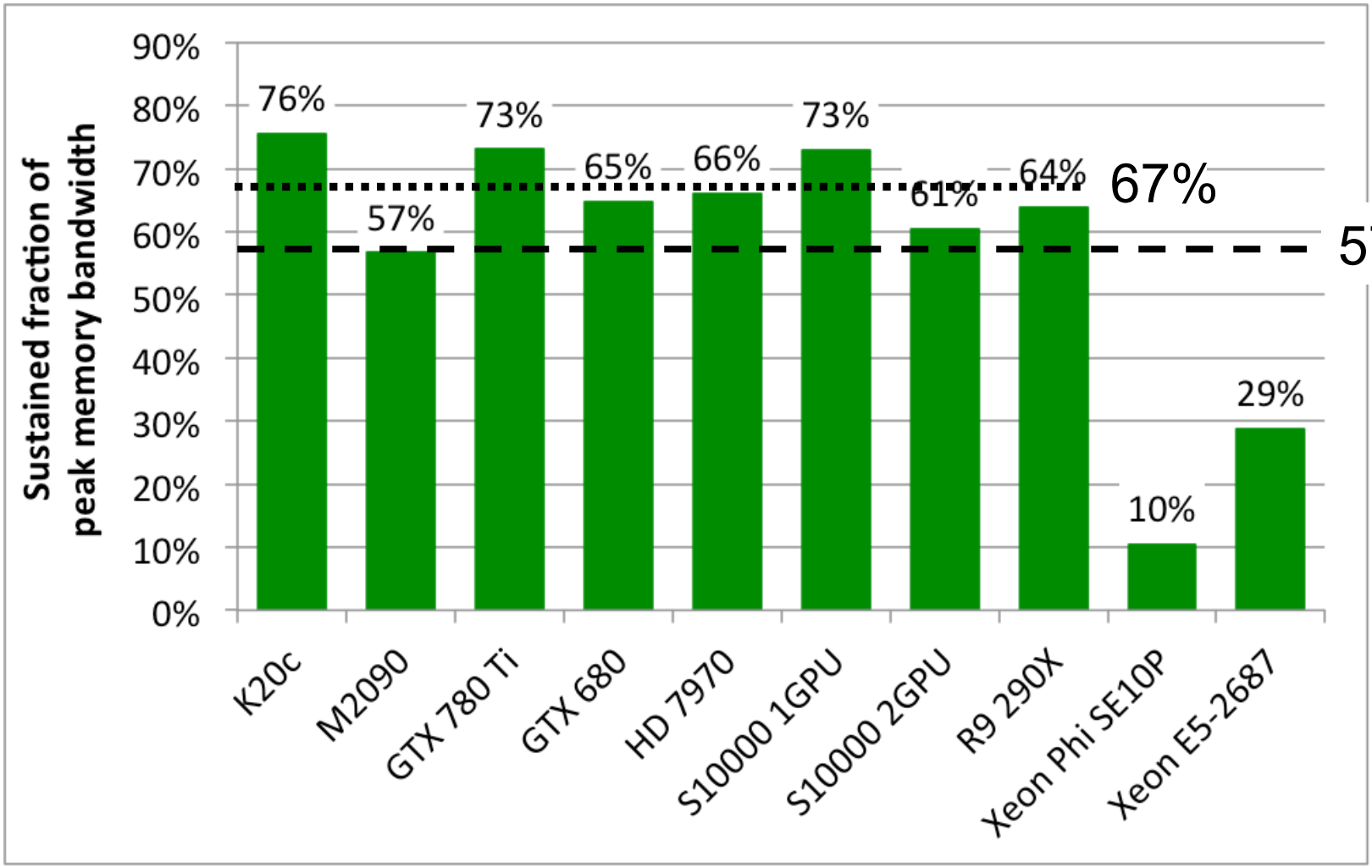
- Code was extremely efficient but not over complicated
- "Identical" versions in OpenCL and CUDA
  - Single precision grid 128^3 (~2m grid points, 304 MBytes)
  - The OpenCL three dimensional work-group size was fixed at (128,1,1) for all OpenCL runs on all devices.
  - The CUDA thread grouping was arranged in exactly the same way as the OpenCL execution, with a blocksize of (128,1,1).
- The OpenMP code was as close as possible to the OpenCL/CUDA versions
- Made sure the OpenMP code was being vectorised

Performance results for 128^3



Single precision results

Performance results for 128^3

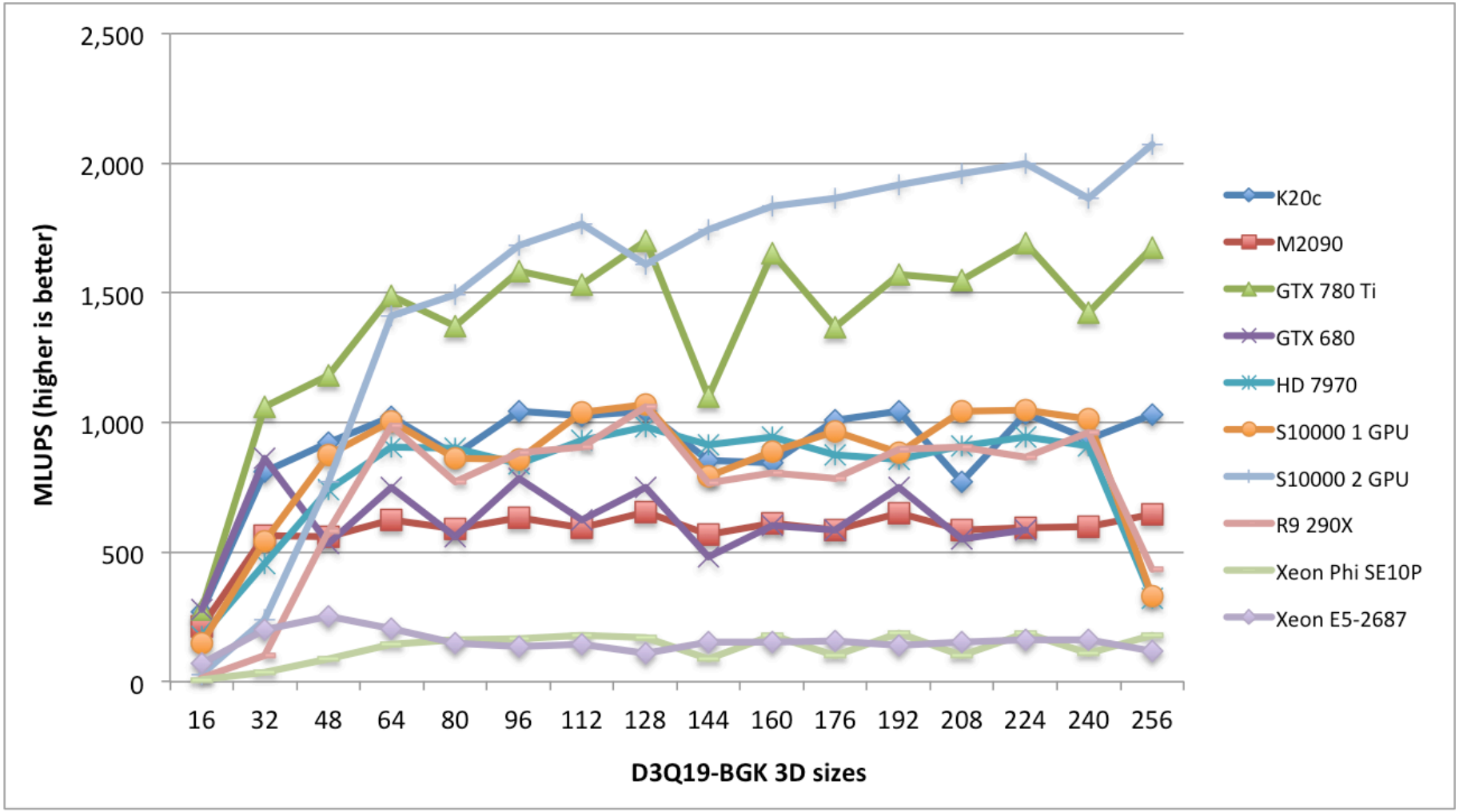


OpenCL single precision results

So perf. portable, but is it fast?

- On an Nvidia K20, our best 128^3 single precision performance in OpenCL was 1,110 MLUPS
- In the literature, the fastest quoted results are ~1,000 MLUPS (Januszewski and Kostur's *Sailfish* program) and 982 MLUPS (Mawson and Revell)
- Our results are a 13% improvement over Mawson-Revell and a 10% improvement over Januszewski-Kostur

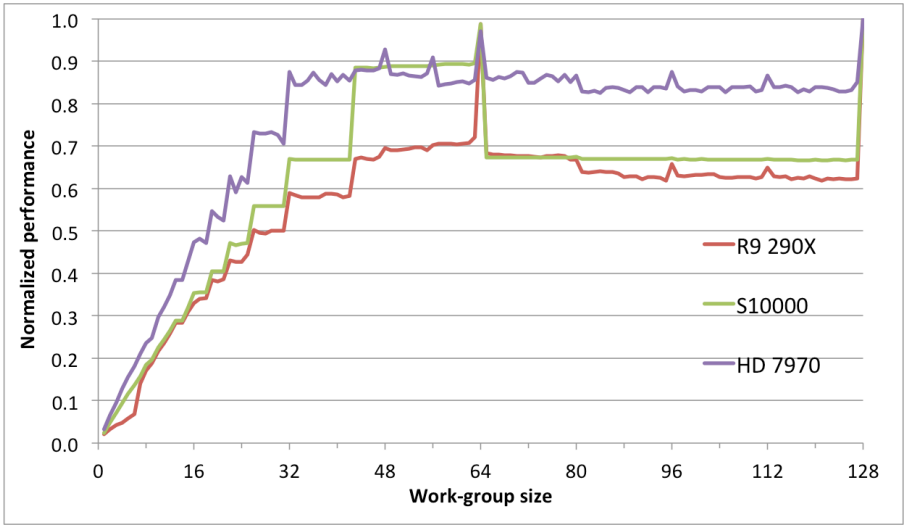
Other grid sizes



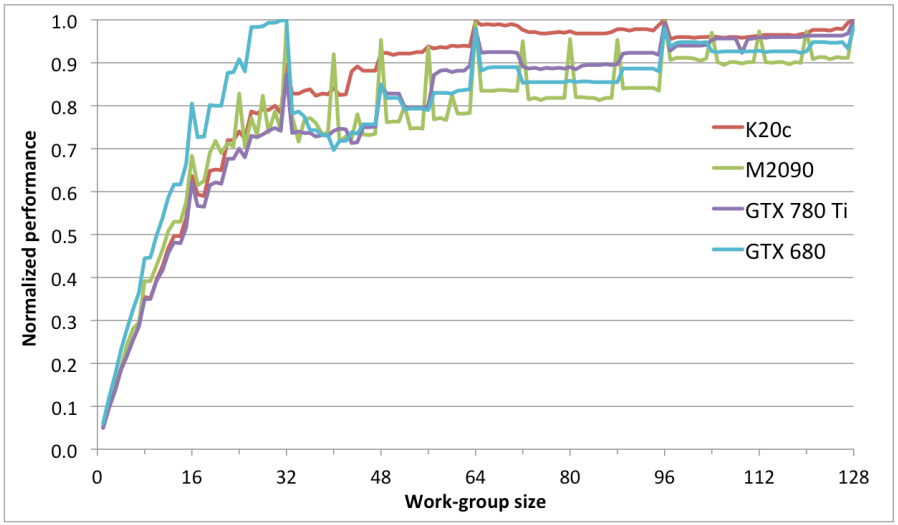
University of BRISTOL

OpenCL single precision results

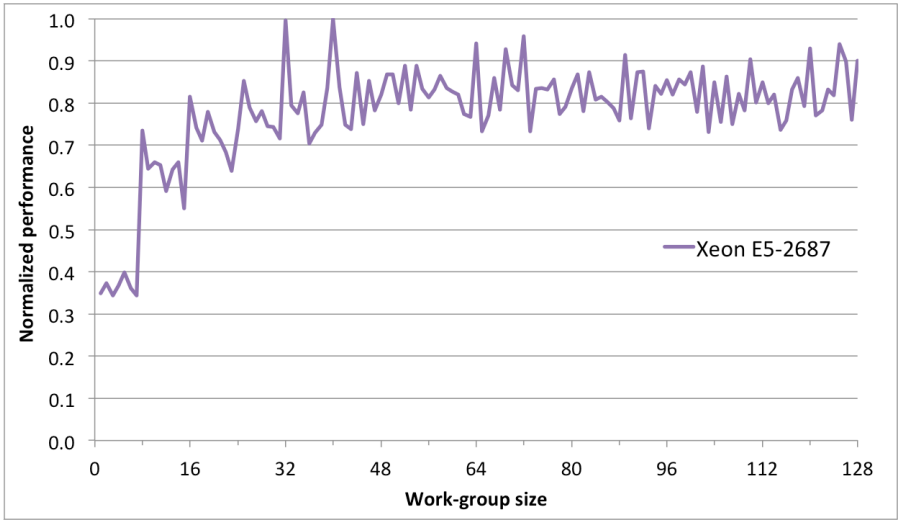
Impact of work-group sizes



AMD GPUs



NVIDIA GPUs



Intel CPU

University of BRISTOL

OpenCL single precision results

Performance portability isn't what we expect

… is it?

Why not?

Why don't we expect perf. portability?

- Historical reasons
  - Started with immature drivers
  - Started with immature architectures
  - Started with immature applications

- But things have ***changed***
  - Drivers now mature / maturing
  - Architectures now mature / maturing
  - Applications now mature / maturing

Performance portability techniques

- Aim for 80-90% of optimal
  - Then easier to get this on many platforms
  - Aiming for ~100% on a specific platform often results in slower code on other platforms
- Avoid platform-specific optimisations
- ***Most*** optimisations make the code faster on ***most*** platforms

Conclusions

- 2D structured grid codes such as lattice Boltzmann can benefit from significant performance improvements on many-core accelerators such as GPUs and Xeon Phi
- **OpenCL** can straightforwardly enable a much better degree of performance portability than most people expect

Related Publications

- "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

- "On the performance portability of structured grid codes on many-core computer architectures", S.N. McIntosh-Smith, M. Boulton, D. Curran and J.R. Price. To appear, International Supercomputing, Leipzig, June 2014.

- "Accelerating hydrocodes with OpenACC, OpenCL and CUDA", Herdman, J., Gaudin, W., McIntosh-Smith, S., Boulton, M., Beckingsale, D., Mallinson, A., Jarvis, S. In: High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:. (Nov 2012) 465–471.

University of BRISTOL