

Performance portability, fault tolerance, and accelerators - oh my!

Simon McIntosh-Smith

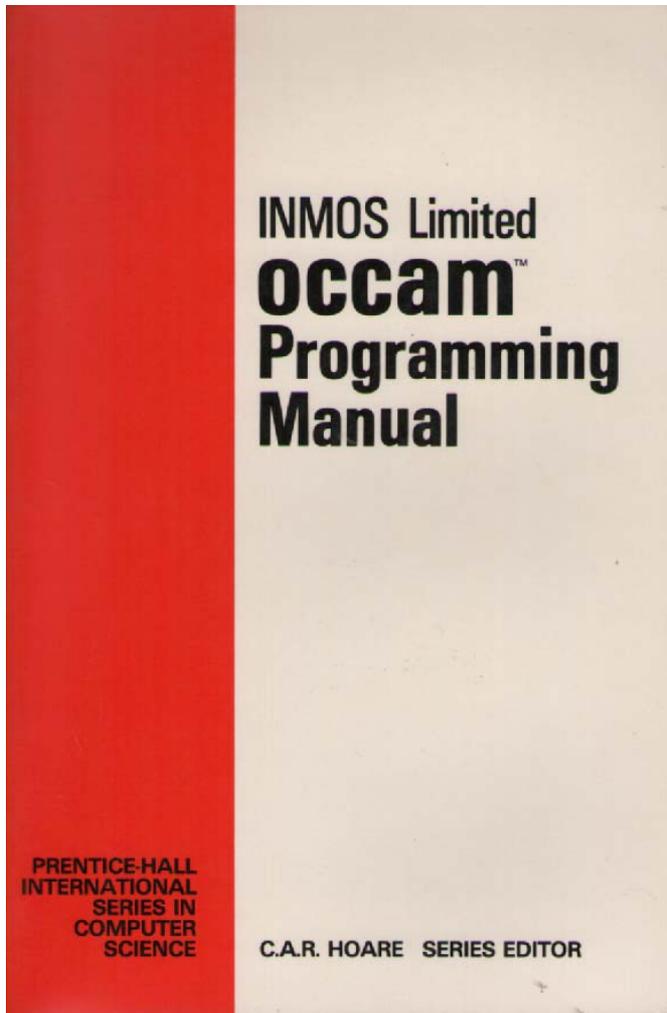
HPC Research Group

simonm@cs.bris.ac.uk







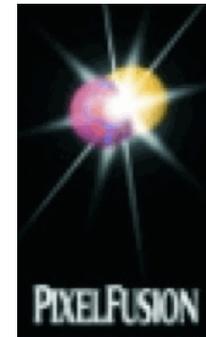


🔥 Bristol's long HPC history

inmos

meiko

ClearSpeed



CRAY
THE SUPERCOMPUTER COMPANY

🔥 What does my group do?

- **Performance portability**

- Programming model evaluations
- Code design strategies
- Hardware evaluations
- "Cross-X", where X = vendor, language, ...

- **Fault tolerance**

MONT-BLANC

- Application-based fault tolerance
- Reliable computing on unreliable hardware

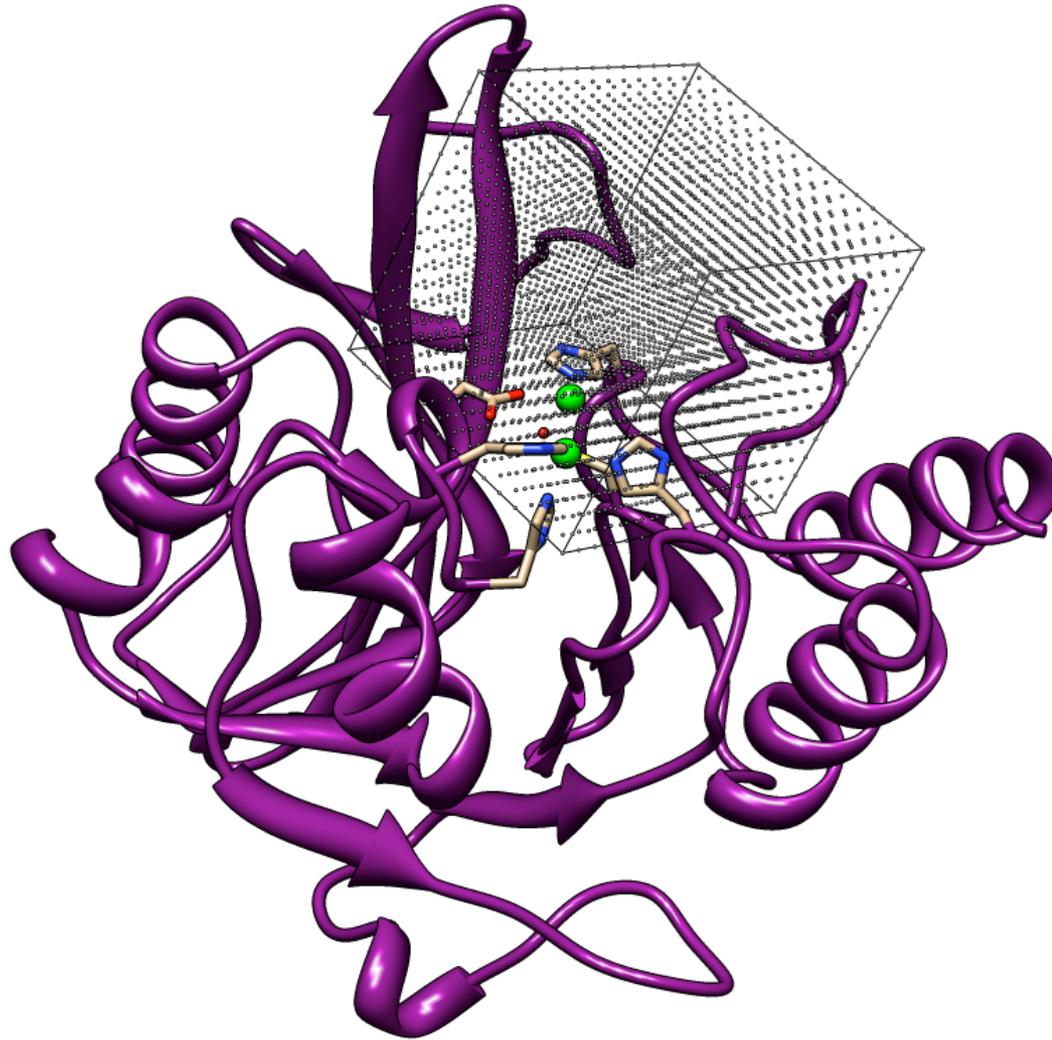
PERFORMANCE PORTABILITY

BUDE – MOLECULAR DOCKING (2013)

What is BUDE?

- **Bristol University Docking Engine**
 - Dr Richard Sessions, PI (Biochemistry)
- Designed for true *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
- Uses a tuned empirical free-energy forcefield for predicting the binding pose and energy of the ligand with the target protein

BUDE protein-ligand docking



What did we do?

- Started with **OpenCL**
 - Supported by all the major vendors (even Nvidia!)
 - Wasn't anything else that let us try cross-vendor, cross-hardware at the time
 - **Ninja level programming**
- Optimised initially for the most parallel device we had
- Kept checking that the optimisations weren't making things worse on the other devices

🔥 More specifically...

- Ported all the code to the accelerator
- Helped the compiler turn all the conditional branches into straight-line, predicated code
 - Involved eyeballing the generated PTX
- Did all the usual things to optimise memory accesses
 - Alignment, padding, coalescence etc.
- Chose sensible problem/work-group sizes

Optimising conditional branches

Conditional execution

```
// Only evaluate expression
// if condition is met
if (a > b)
{
    acc += (a - b*c);
}
```

Corresponding PTX

```
setp.gt.f32 %pred, %a, %b
@!%pred bra $endif
mul.f32 %f0, %b, %c
sub.f32 %f1, %a, %f0
add.f32 %acc, %acc, %f1
$endif:
```

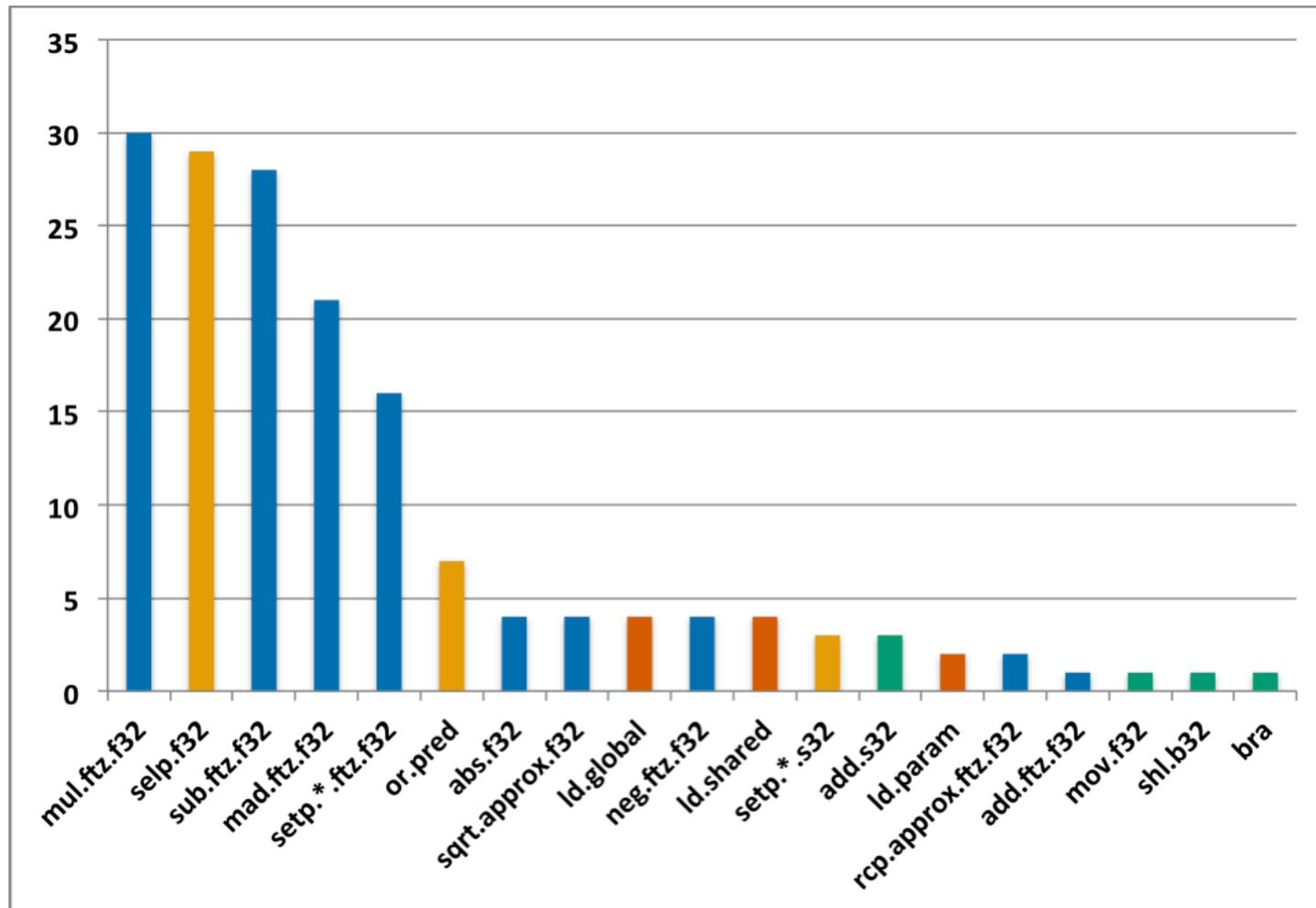
Selection and masking

```
// Always evaluate expression
// and mask result
temp = (a - b*c);
mask = (a > b ? 1.f : 0.f);
acc += (mask * temp);
```

Corresponding PTX

```
mul.f32 %f0, %b, %c
sub.f32 %temp, %a, %f0
setp.gt.f32 %pred, %a, %b
selp.f32 %mask, %one, %zero, %pred
mad.f32 %acc, %mask, %temp, %acc
```

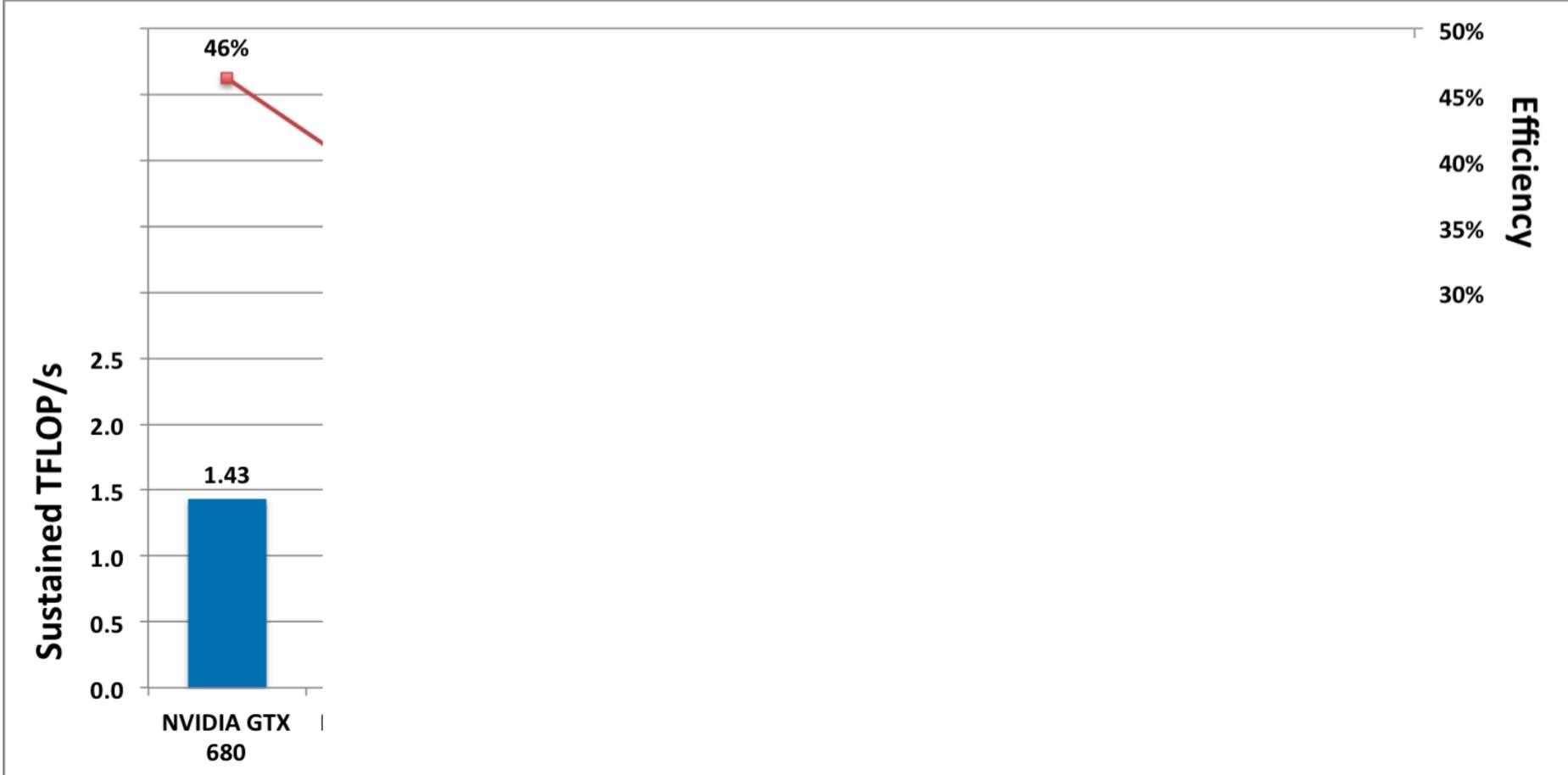
🌿 Instruction mix



Target hardware

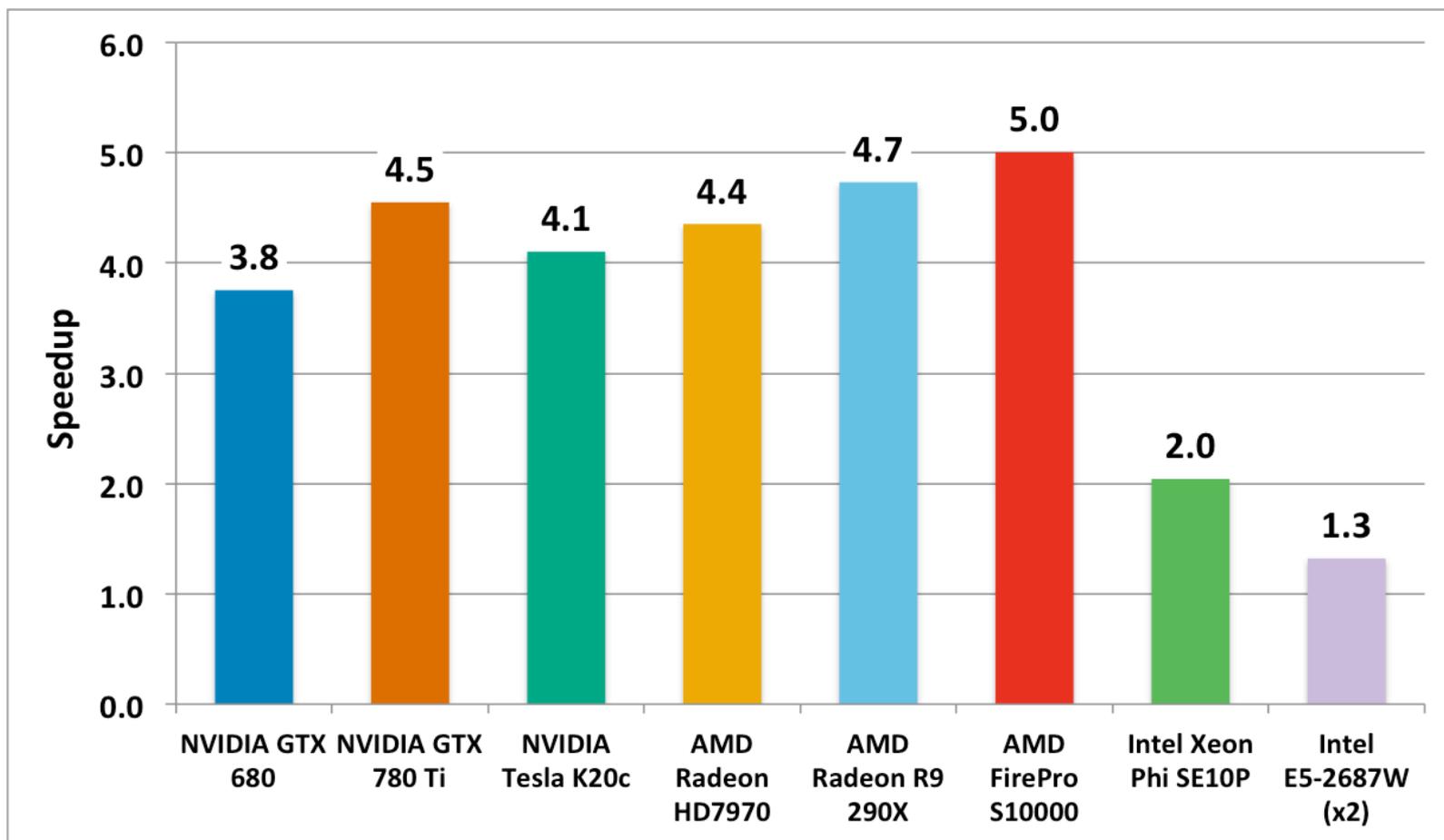
Platform	Clock (GHz)	RAM (GB)	Memory B/W (GB/s)	S.P. TFLOP/s	D.P. TFLOP/s	TDP (W)
AMD FirePro S10000	0.825	6	480	5.91	1.48	375
AMD Radeon HD 7970	0.925	3	264	3.78	0.95	230
AMD Radeon R9 290X	1.000	4	320	5.63	0.70	250
Intel Xeon E5-2687W (x2)	3.100	32	102	0.79	0.40	300
Intel Xeon Phi SE10P	1.100	8	320	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

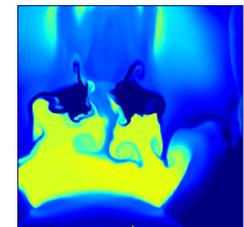
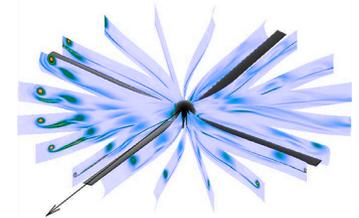
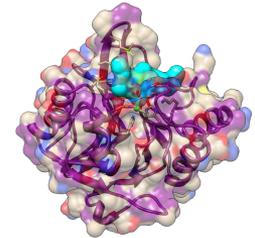
🔥 How much the optimisations helped



"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

🌿 Performance portability

- BUDE was highly performance portable
 - Compute intensive, N-body / Monte Carlo
- Bandwidth intensive codes next
 - Structured grid codes:
 - **CloverLeaf (hydrodynamics)**
 - ROTORSIM (CFD)
 - Lattice Boltzmann



STRUCTURED GRID CODES **(2013)**

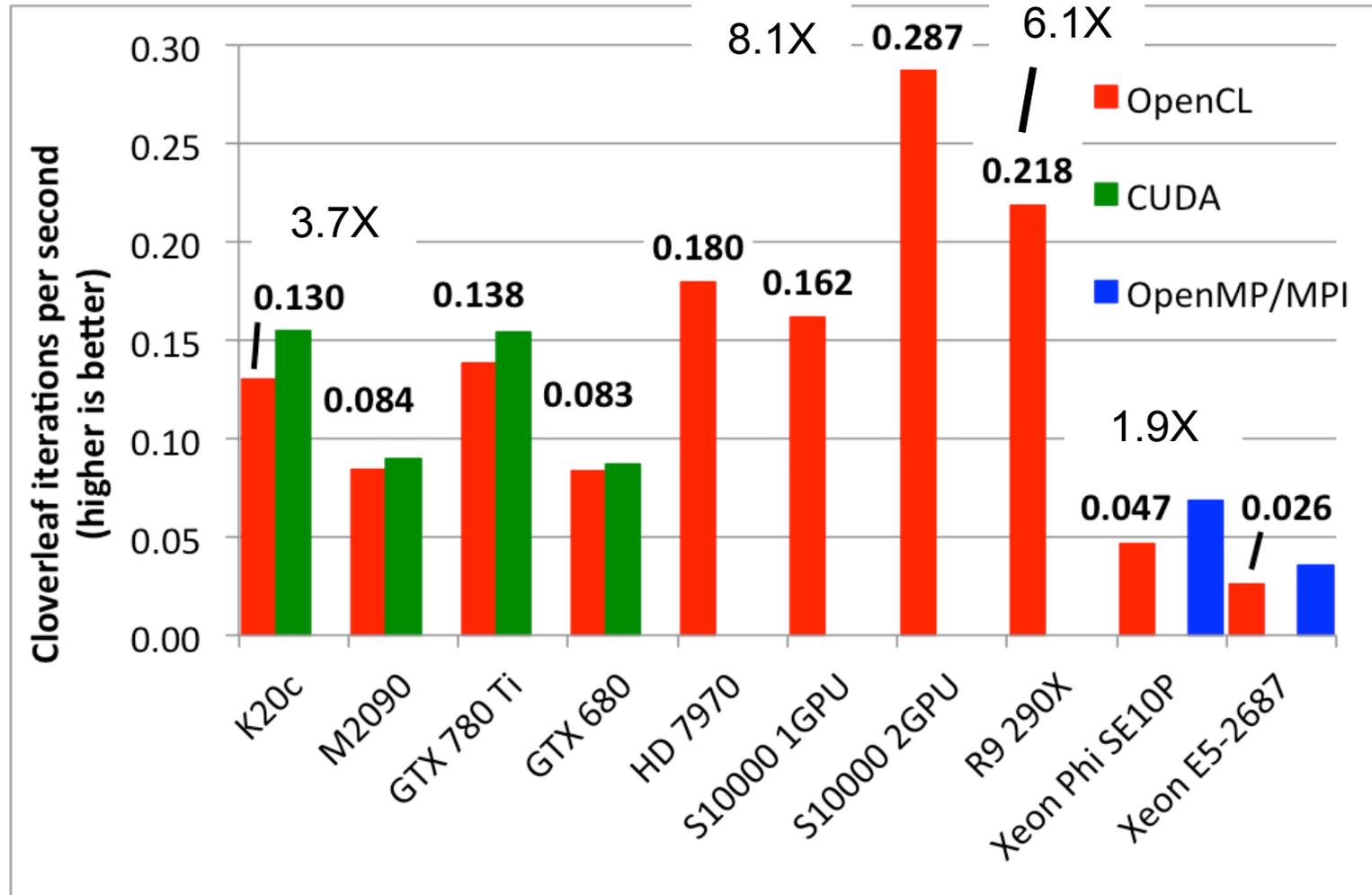
CloverLeaf: A Lagrangian- Eulerian hydrodynamics benchmark

- A collaboration between AWE, Warwick & Bristol
- CloverLeaf is a bandwidth-limited, structured grid code and part of Sandia's "Mantevo" benchmarks
- 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

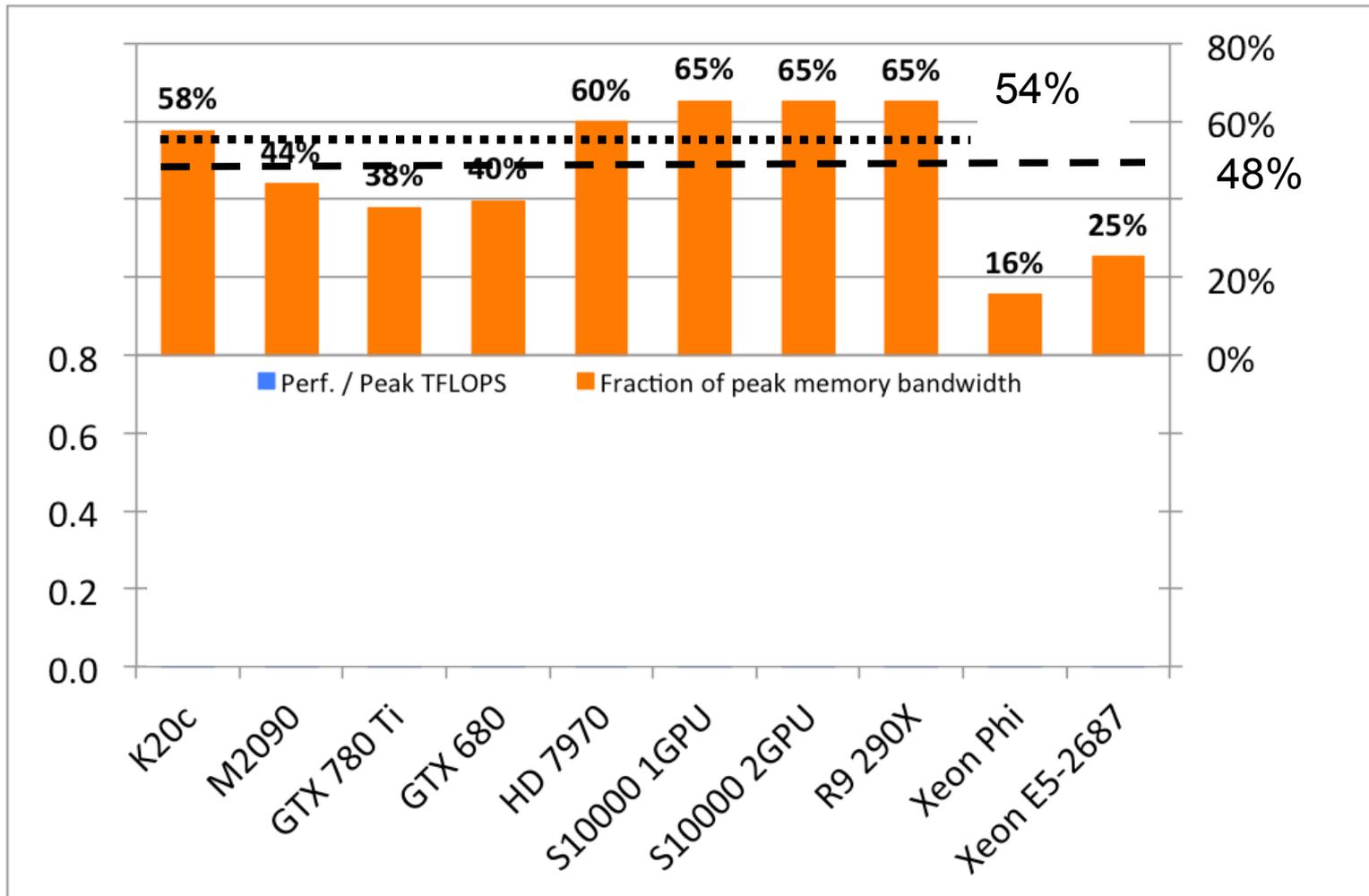
CloverLeaf benchmark parameters

- Double precision grid of size 1920×3840
 - ~7.4m grid points, 25 values per grid point
→ ~1.5 Gbytes in working dataset
- The OpenCL and CUDA parallelisations were performed in an identical manner
 - One work-item/thread for each grid point
 - Identical arrangements for work-group sizes and layouts
 - E.g. 2D work-groups of (128, 1) for OpenCL

Results – performance



🔥 Results – sustained bandwidth



Performance portability isn't what we expect

But 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

(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

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, ...**

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)

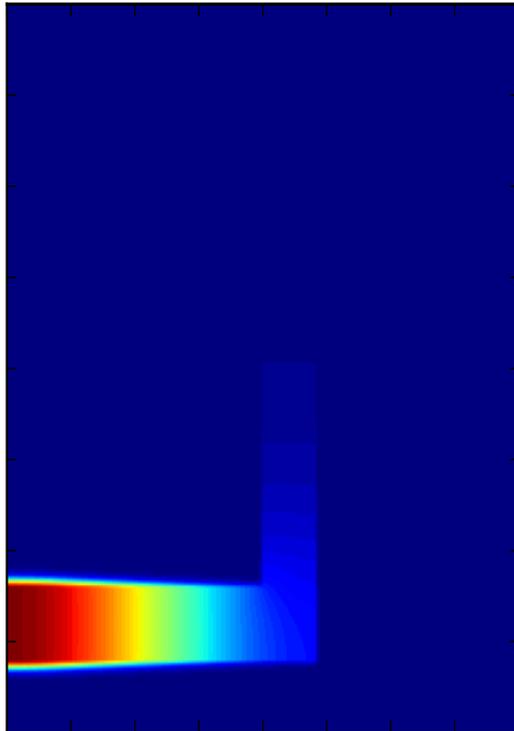
Wayne Gaudin – UK Atomic Weapons Establishment

bristol.ac.uk

DOE performance portability workshop, Arizona, April 2016.

🌿 TeaLeaf – Heat Conduction

- Mini-app from Mantevo suite of benchmarks

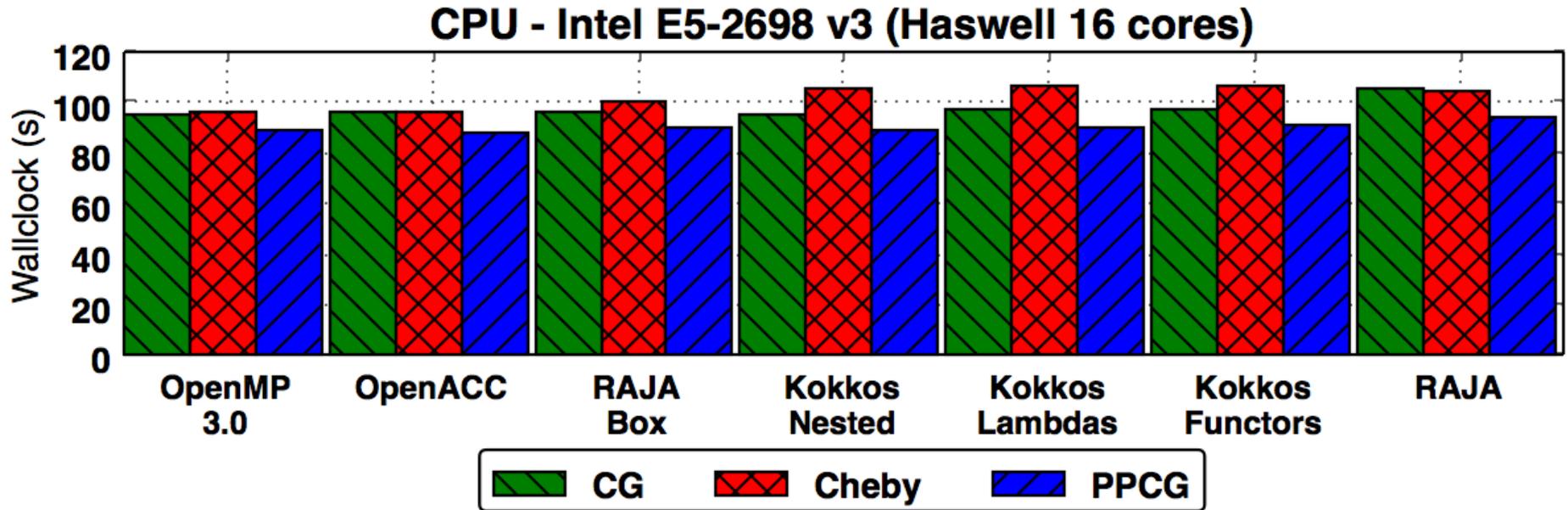


- Implicit, sparse, matrix-free solvers on structured grid
 - Conjugate Gradient (CG)
 - Chebyshev
 - Preconditioned Polynomial CG (PPCG)
- Memory bandwidth bound
- Good strong and weak scaling on Titan & Piz Daint

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 4096×4096 problem, the point of mesh convergence, for *single iteration*

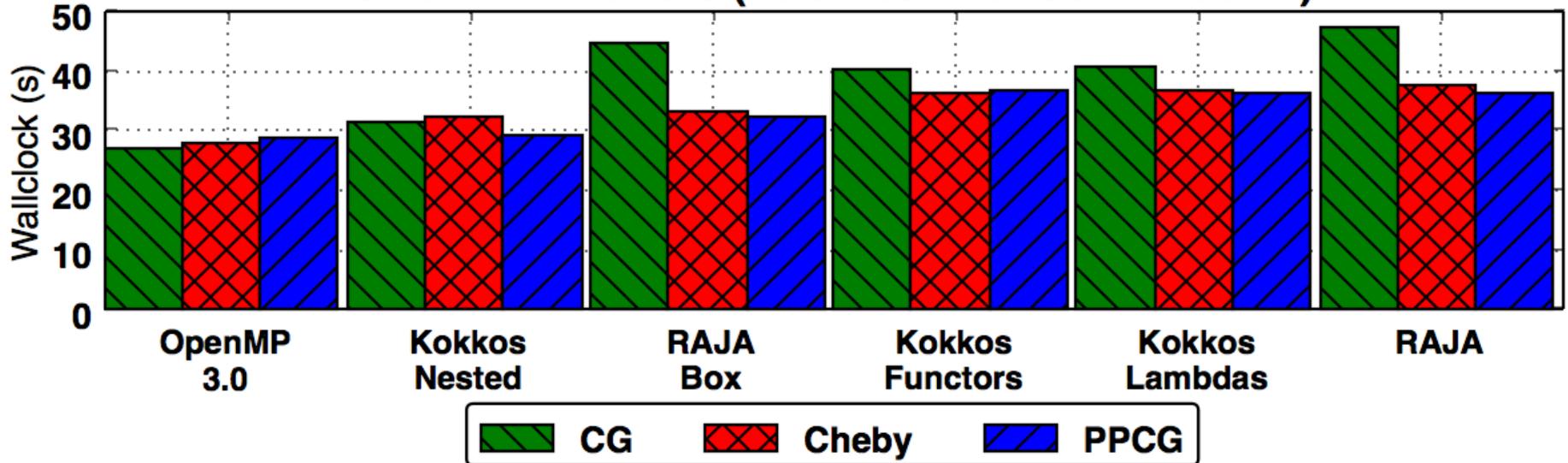
TeaLeaf - CPU



At most 12% runtime penalty for modern Intel CPU

TeaLeaf – Power8

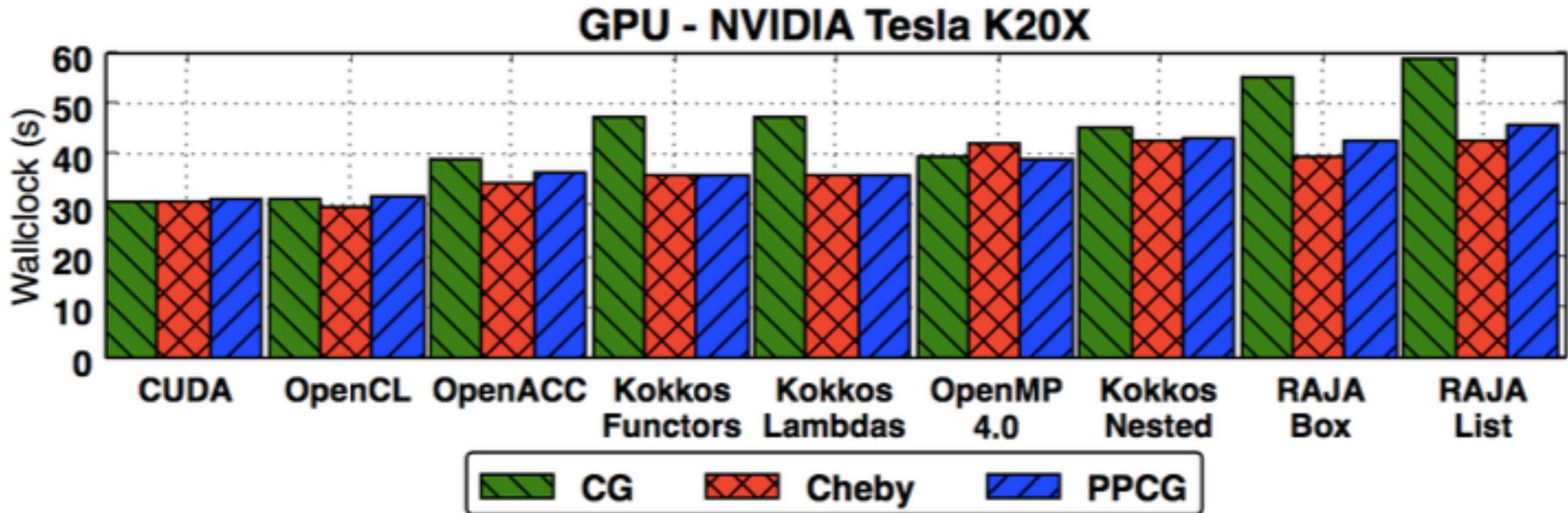
CPU - IBM Power8 (10 cores Dual Socket 8SMT)



Performance bug often seen with CG solver

Results generally good, particularly for optimised versions

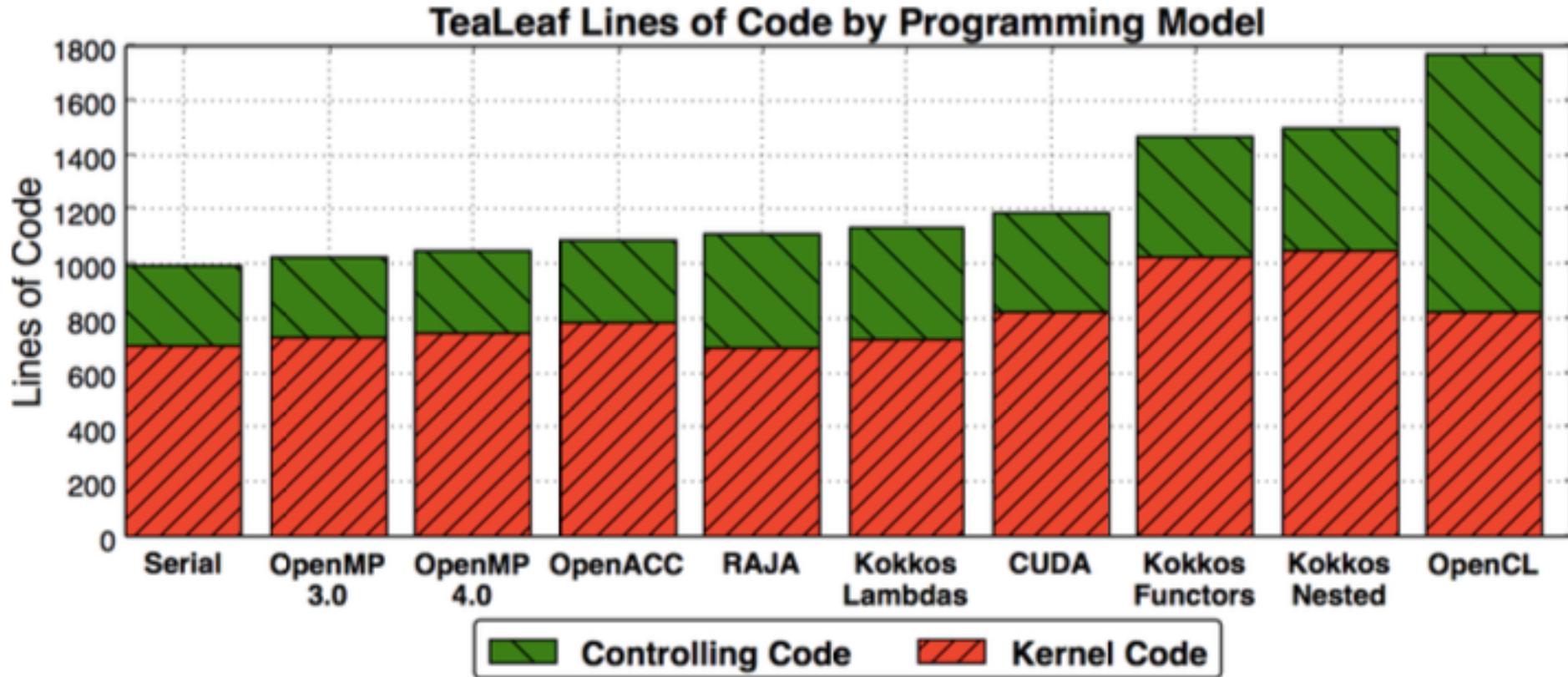
TeaLeaf – GPU



Performance bug with CG again present in some cases

All models get to within 25% of OpenCL / CUDA

🌿 TeaLeaf lines of code



🌿 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

HARDER APPLICATIONS: TRANSPORT (2014-)

Transport: SNAP mini-app

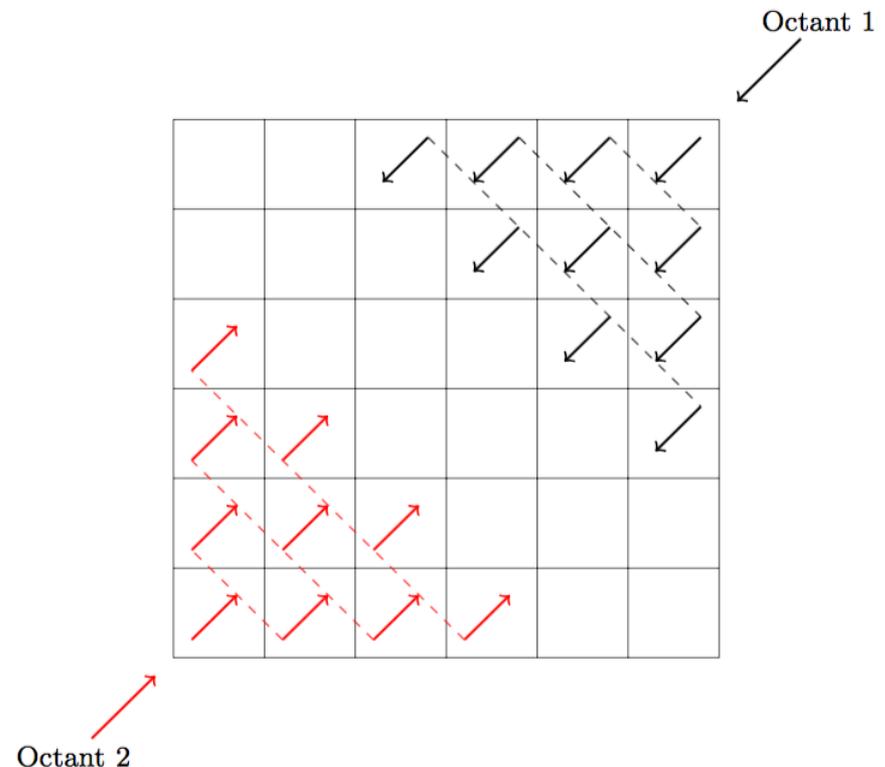
- Neutral particle transport code from LANL
- Performance proxy for production code PARTISN
- Discretization
 - Finite difference in time
 - Finite difference in space
 - Multi-group in energy
 - Discrete ordinates in angle
- Solution
 - Matrix free, structured grid
 - Simple iterations on scattering source (RHS of equation)
 - Jacobi iterations in energy
- Generates **a sweep across the spatial mesh**
 - Can't compute in all cells all at once like your other favourite PDEs!

Making SNAP fast

- Weren't any really fast GPU ports of SNAP
- The wavefront dependencies make it hard to parallelise

🌟 A Many-core Implementation

- Need to solve for each cell, for each angle and for each energy group
- We exposed all the different levels of parallelism in one node:
 - Cells in the wavefront/hyperplane
 - Angles in the octant
 - Energy groups
- This is the first time GPUs outperformed CPUs for deterministic transport

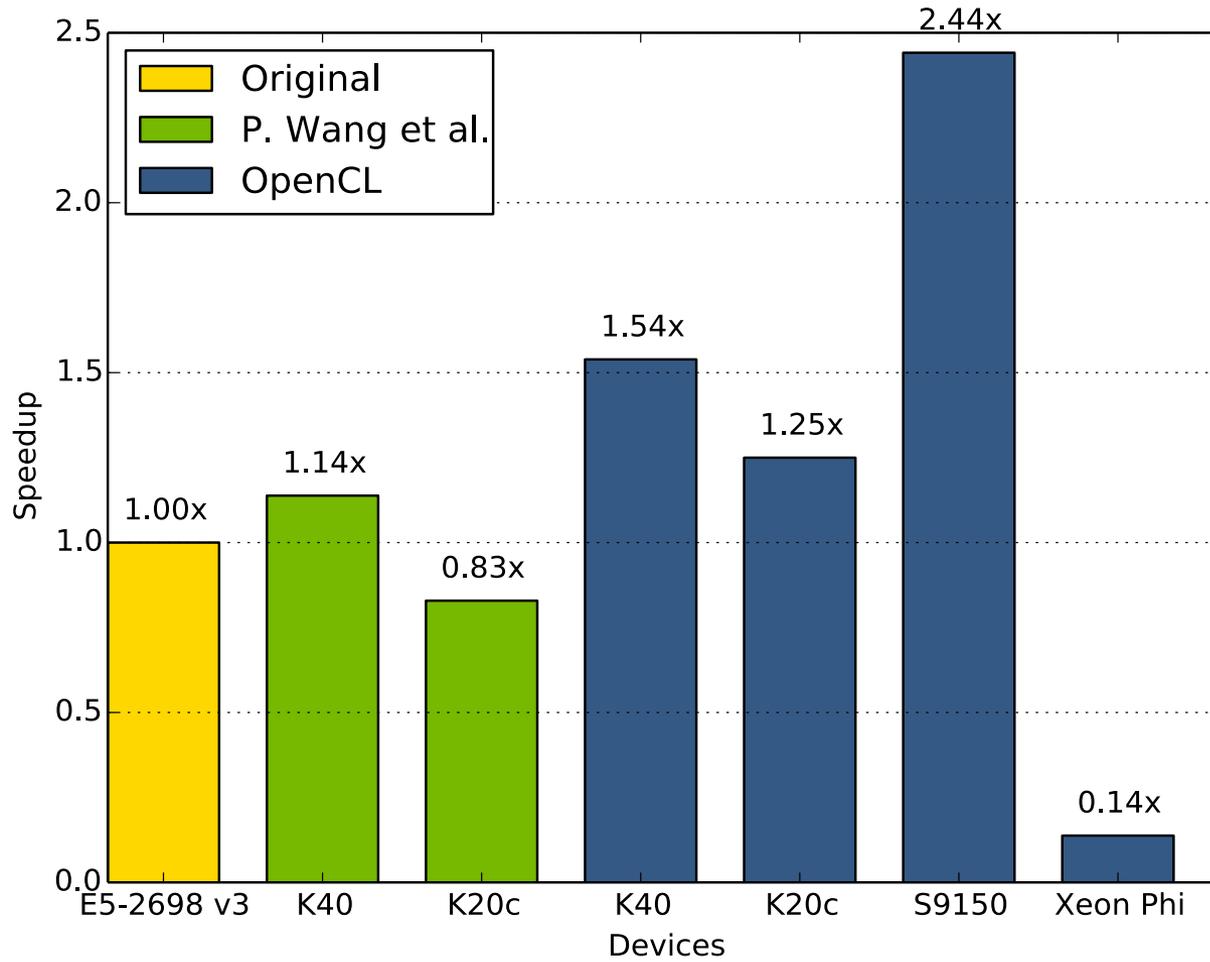


Expressing Parallelism on Many-Core for Deterministic Discrete Ordinates Transport.

Deakin, T., McIntosh-Smith, S., & Gaudin, W. (2015).

International Workshop on Representative Applications (WRAP), IEEE Cluster, Chicago, USA.

New SNAP results

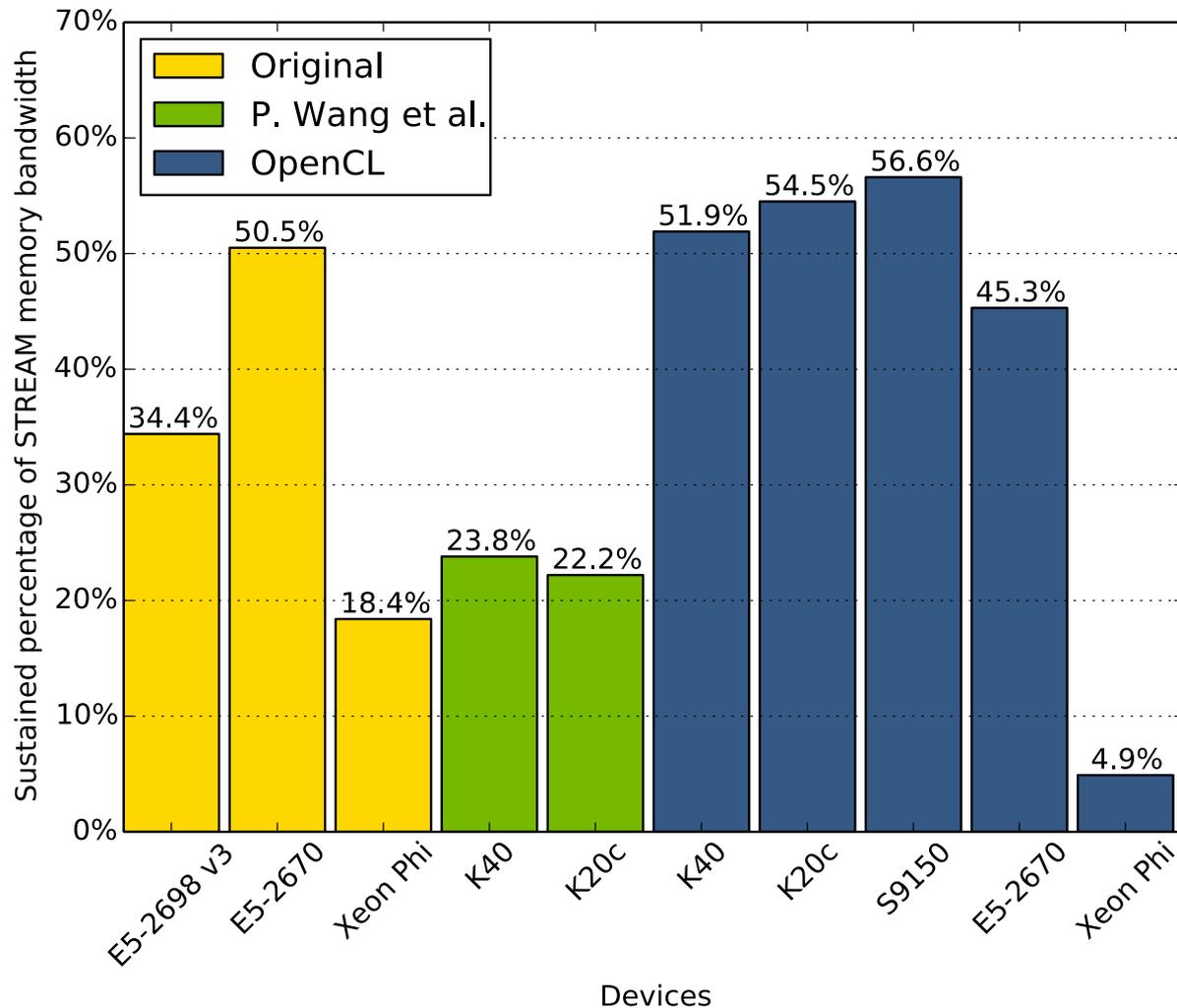


Expressing Parallelism on Many-Core for Deterministic Discrete Ordinates Transport.

Deakin, T., McIntosh-Smith, S., & Gaudin, W. (2015).

International Workshop on Representative Applications (WRAP), IEEE Cluster, Chicago, USA.

Performance portability too

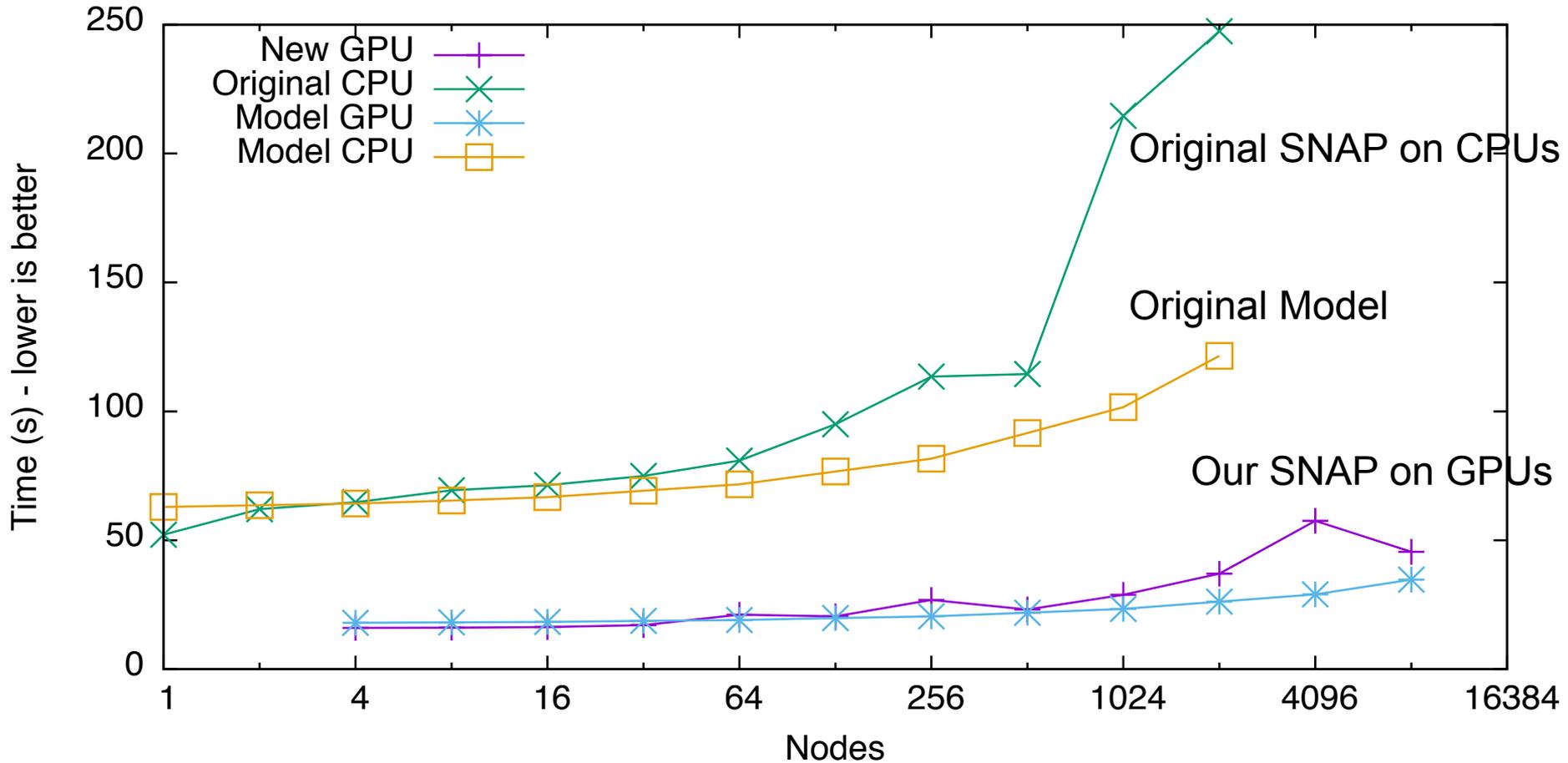


Expressing Parallelism on Many-Core for Deterministic Discrete Ordinates Transport.

Deakin, T., McIntosh-Smith, S., & Gaudin, W. (2015).

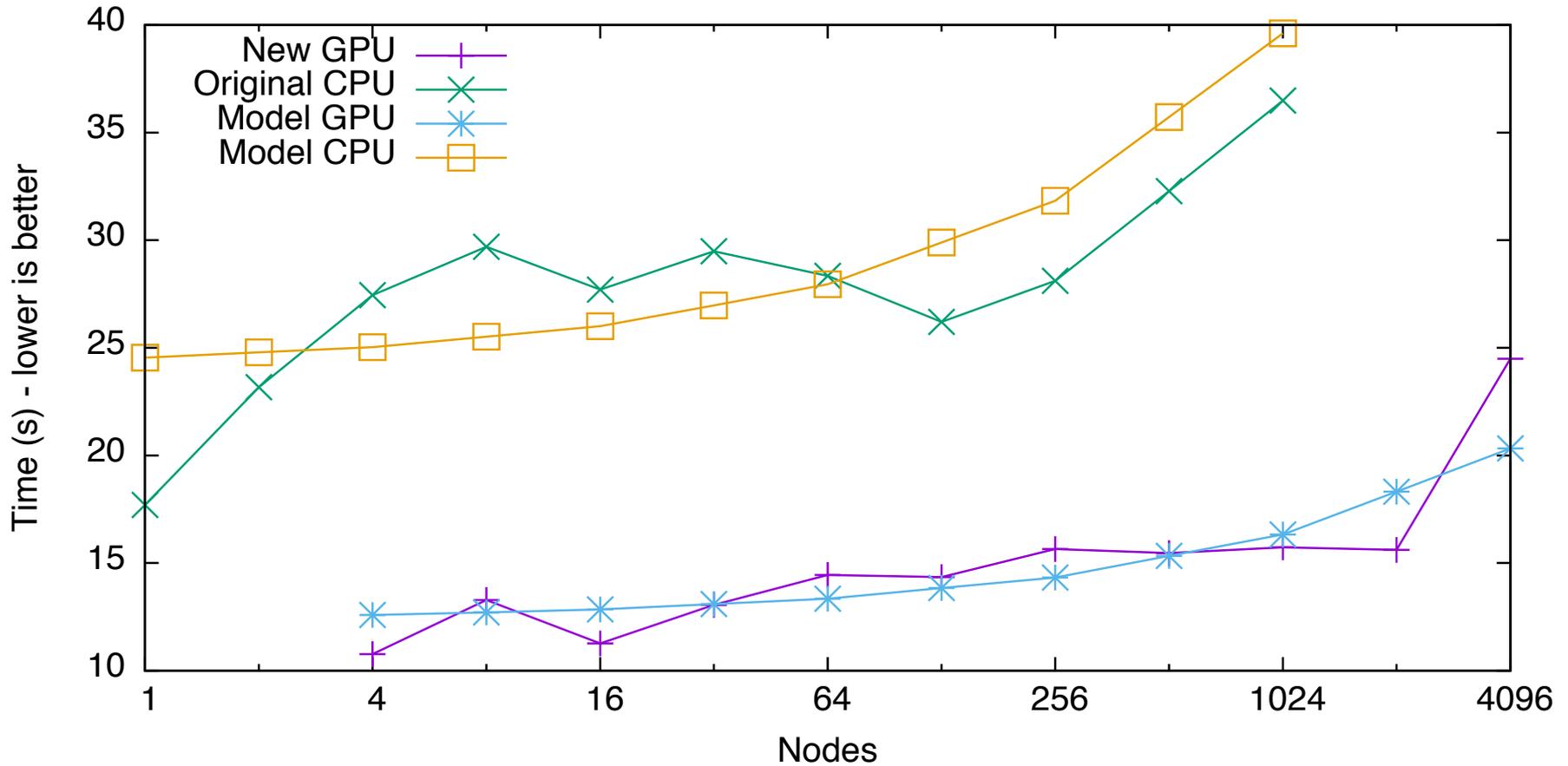
International Workshop on Representative Applications (WRAP), IEEE Cluster, Chicago, USA.

Titan
Weak scaling 4x4x400 cells per MPI rank
CPU: 4 ranks per node; GPU: 1 rank per node



- 4X speedup using the GPUs
- Models line up well with experimental results: within 20%
- Titan network struggling: 84% time in communication at 8192 MPI ranks

Piz Daint
Weak scaling 4x4x400 cells per MPI rank
CPU: 2 ranks per node; GPU: 1 rank per node



- 2X speedup using the GPUs - CPUs much faster than in Titan
- Models line up well with experimental results: within 12%
- Better network helps transport: less time in comms than Titan's Gemini torus

Performance portability summary

- It is possible to achieve performance portability for hydrodynamics, diffusion and transport
 - At least for current CPUs and GPUs
 - Harder for the next generation though
 - Memory hierarchy, degree of parallelism per node, ...
- Performance portability should be getting easier to achieve with higher-level abstractions
 - RAJA, Kokkos all looking promising
 - OpenMP 4.x also delivering good results

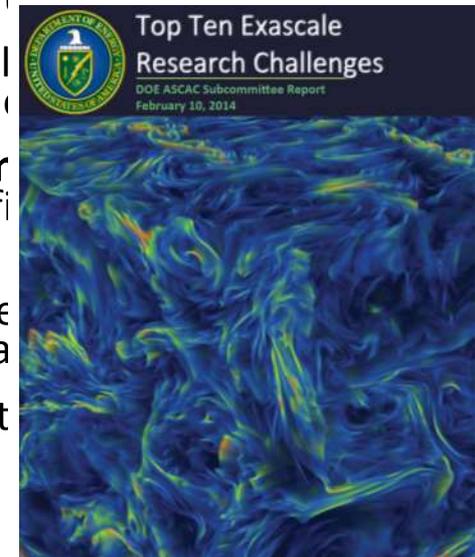
Performance portability refs

- **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
- **Expressing Parallelism on Many-Core for Deterministic Discrete Ordinates Transport**
Deakin, T., McIntosh-Smith, S., & Gaudin, W.
International Workshop on Representative Applications (WRAp), IEEE Cluster, Chicago, United States. Sep 2015.
DOI: 10.1109/CLUSTER.2015.127
- <https://github.com/UoB-HPC/>

FAULT TOLERANCE

🔥 Top 10 Exascale challenges

1. **Energy efficiency:** Creating more energy-efficient circuit, power, and cooling technologies.
2. **Interconnect technology:** Increasing the performance and energy efficiency of data movement.
3. **Memory technology:** Integrating advanced memory technologies to improve both capacity and bandwidth.
4. **Scalable system software:** Developing scalable system software that is power- and **resilience-aware**.
5. **Programming systems:** Inventing new programming environments that express massive parallelism, data locality, and **resilience**.
6. **Data management:** Creating data management software that can handle the volume, velocity and diversity of data that is and
7. **Exascale algorithms:** Reformulating science problems or reinventing, their solution algorithms for exascale
8. **Algorithms for discovery, design, and decision making:** Mathematical optimization and uncertainty quantification for discovery, design, and decision making.
9. **Resilience and correctness:** Ensuring correct science in the face of faults, reproducibility, and algorithm verification
10. **Scientific productivity:** Increasing the productivity of scientists with new software engineering tools and



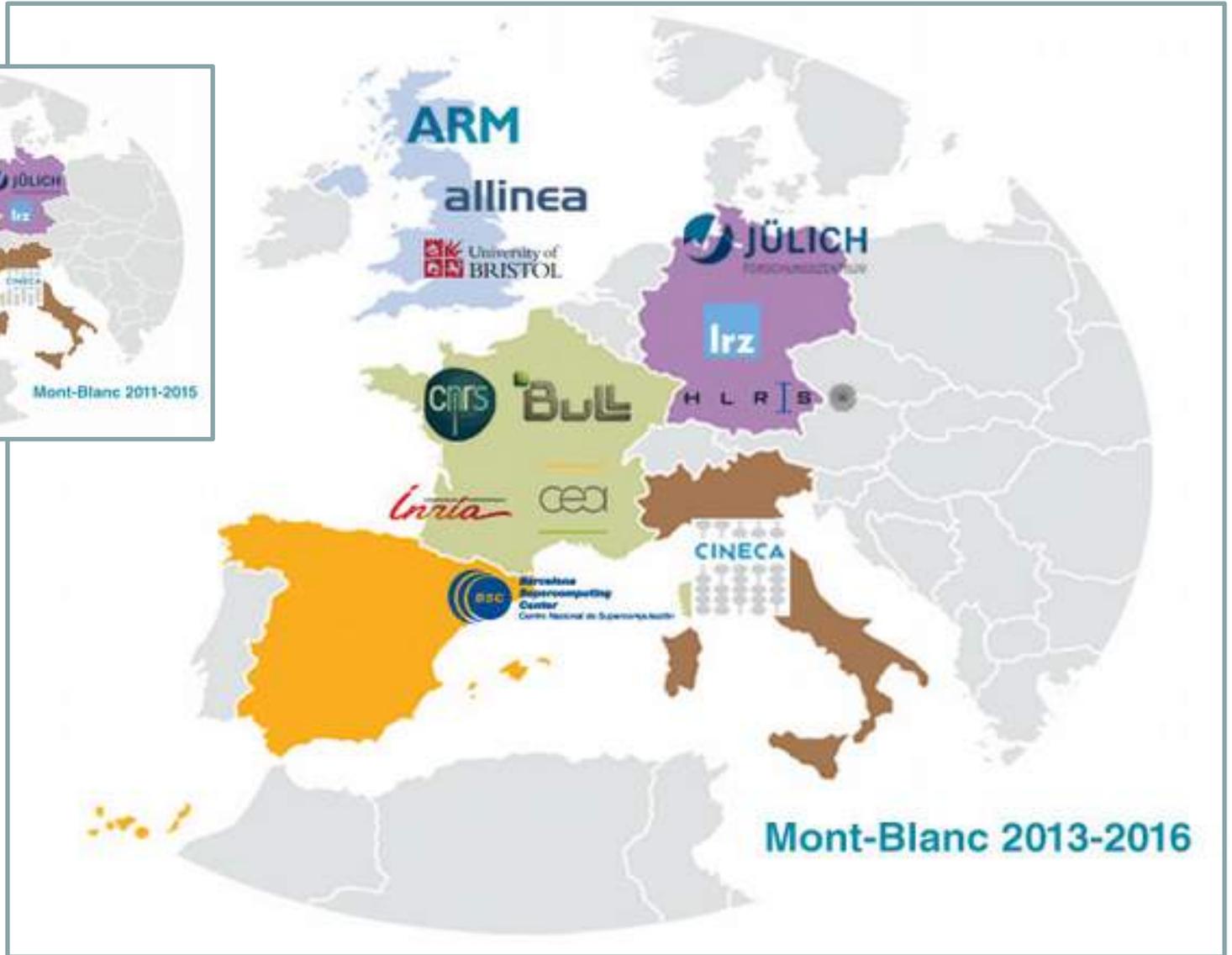
FP7 Mont Blanc project

MONT-BLANC

Build a supercomputer from mobile processor technology – is it more energy efficient?

- Project running since Oct 2011
- €16m total EU funding for 6 years
- 14 project partners
- Several prototype machines so far

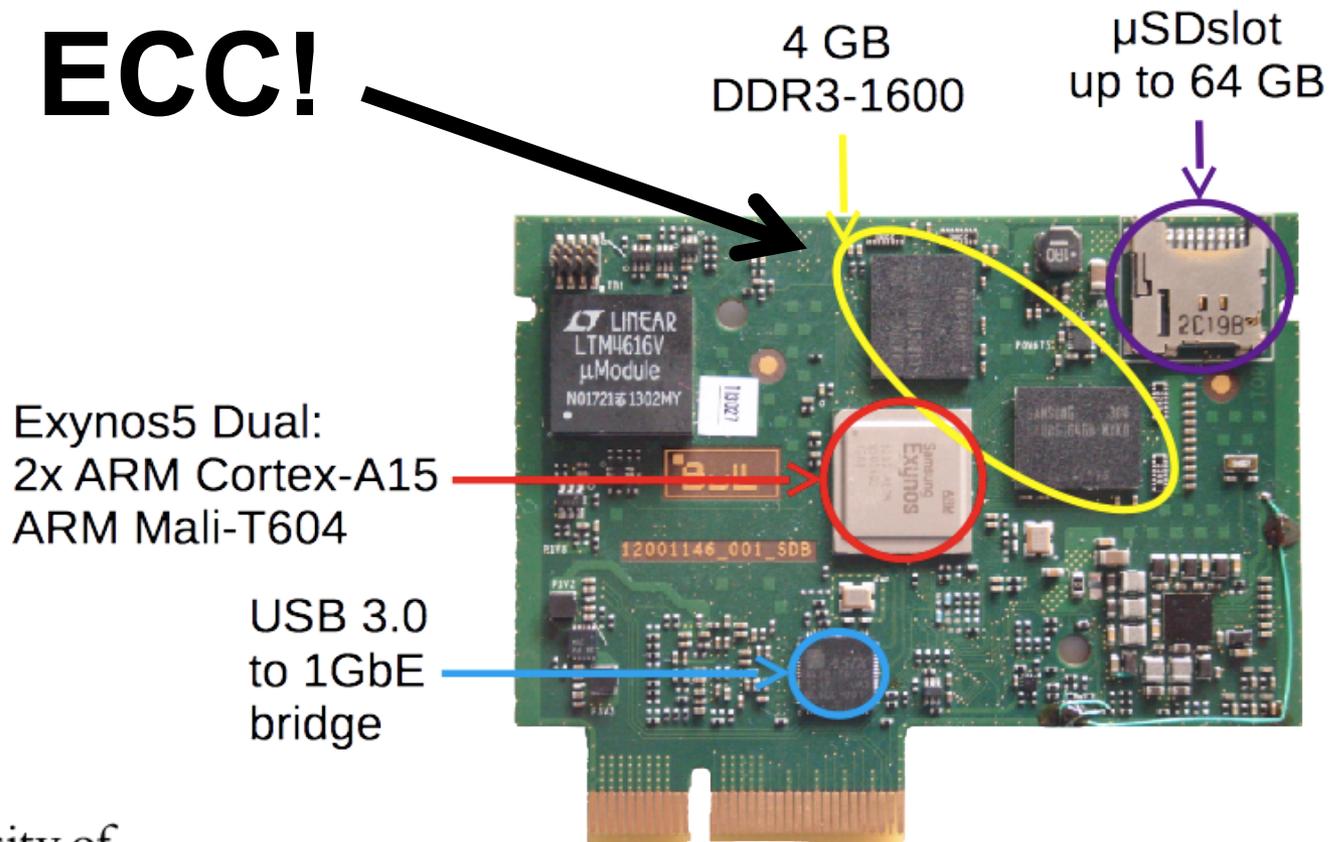
<http://www.montblanc-project.eu/>



🔥 Mont Blanc compute card

CPU + GPU + DRAM + storage + network
all in a compute card just 8.5 x 5.6 cm

No ECC!

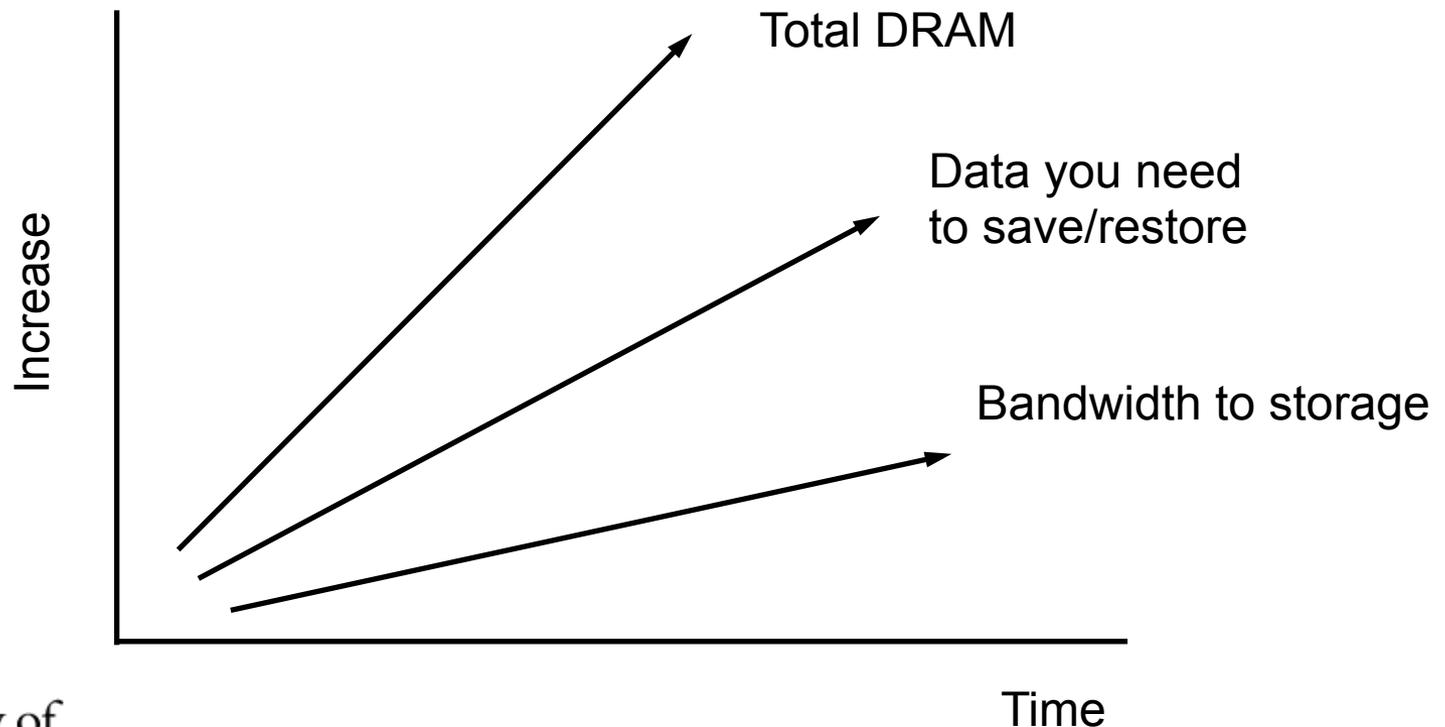


🔥 Fault tolerance & resilience

- All handled by the hardware, right?
- No free lunch anywhere anymore...
- Consider "simple" ECC on external DRAM
 - Single Error Correct Double Error Detect
 - Uses an extra 8 parity bits per 64 data bits
 - 12.5% more memory, bandwidth, power, ...
 - Expensive, slow mechanism when an uncorrectable error occurs
 - Invoke OS, checkpoint/restart sequence, ...

🔥 Checkpoint/restart

- But if ECC doesn't catch the error, checkpoint/restart solves the problem, right?



Application-Based Fault Tolerance (ABFT)

Lots of good progress being made in:

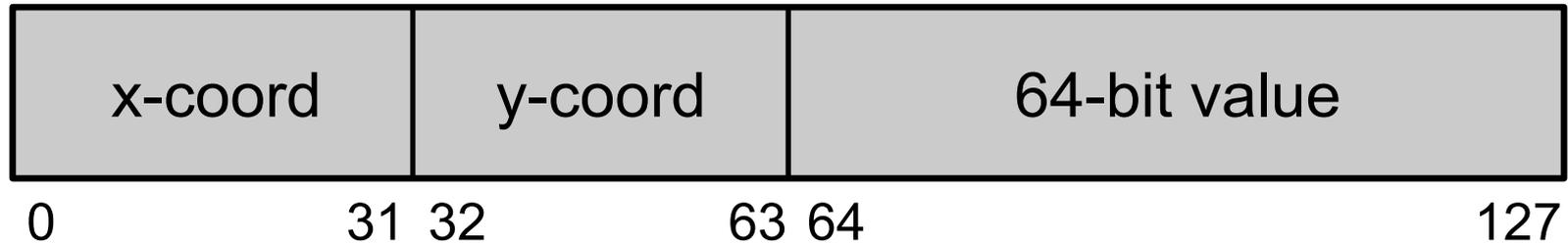
- **Sparse linear algebra ([this work](#))**
- Dense linear algebra
- Monte Carlo
- Structured / Unstructured grids
- N-body
- Spectral (FFT)

🔥 Sparse matrix compressed formats

- Sparse matrices are typically mostly 0
- E.g. in the University of Florida sparse matrix collection (~2,600 real, floating point examples), the median fill of non-zeros is just **~0.24%**
- Therefore stored in a **compressed format**, such as COOrdinate format (COO) and Compressed Sparse Row (CSR)



🔥 COO sparse matrix format

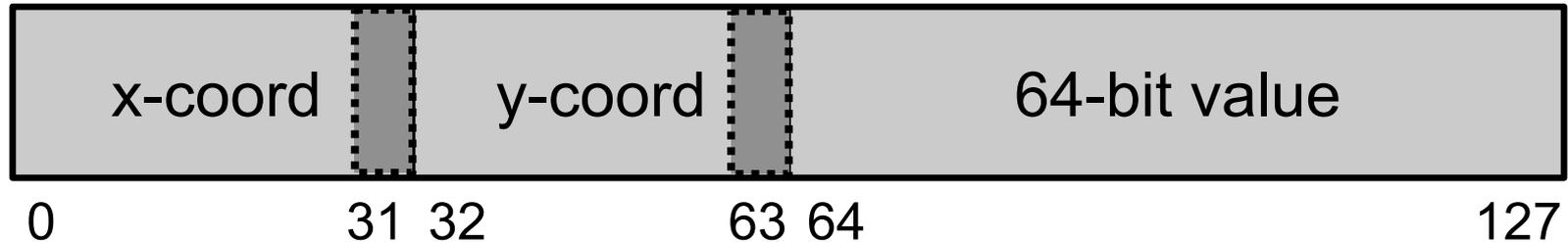


- Conceptually think of each sparse matrix element as a 128-bit structure:
 - Two 32-bit unsigned coordinates (x,y)
 - One 64-bit floating point data value

Software ECC protection for sparse matrix elements

- Remember that most sparse matrices don't use all their index bits
- **Observation:** *This leave index bits that could be "repurposed" for a software ECC scheme*
- A software ECC scheme might save considerable energy, performance and memory (all in the region of 10-20%)

🔥 COO sparse matrix format



- Using 8 bits of the 128-bit compound element would allow a full single error correct, double error detect (SECDED) scheme in software
- Use e.g. 4 unused bits from the top of each index
 - Limits their size to "just" $0..2^{27}$ ($0..134M$)
 - With 64-bit indices, even more spare bits we can use
- Requires no more bandwidth, just more compute
 - Actually saves the 12.5% ECC bandwidth...

Software-based ECC Results



This is looking very promising!

Scheme	Bits needed	x86 overhead	ARM32 overhead
Constraints	0	1.00x	1.07x
SED	1	1.01x	1.28x
SEC 7-bit	7	2.61x	4.27x
SEC 8-bit	8	1.04x	1.48x
SECDED	8	2.49x	4.79x

- Next, implement on a GPU
- Compare with/without ECC hardware enabled

🔥 Fault tolerance conclusions

- Fault tolerance / resilience is set to become a **first-order concern for Exascale scientific software**
- Application-based fault tolerance (**ABFT**) is one promising technique to address this issue
- ABFT can be applied at the ***library-level*** to help protect ***large-scale sparse matrix operations***

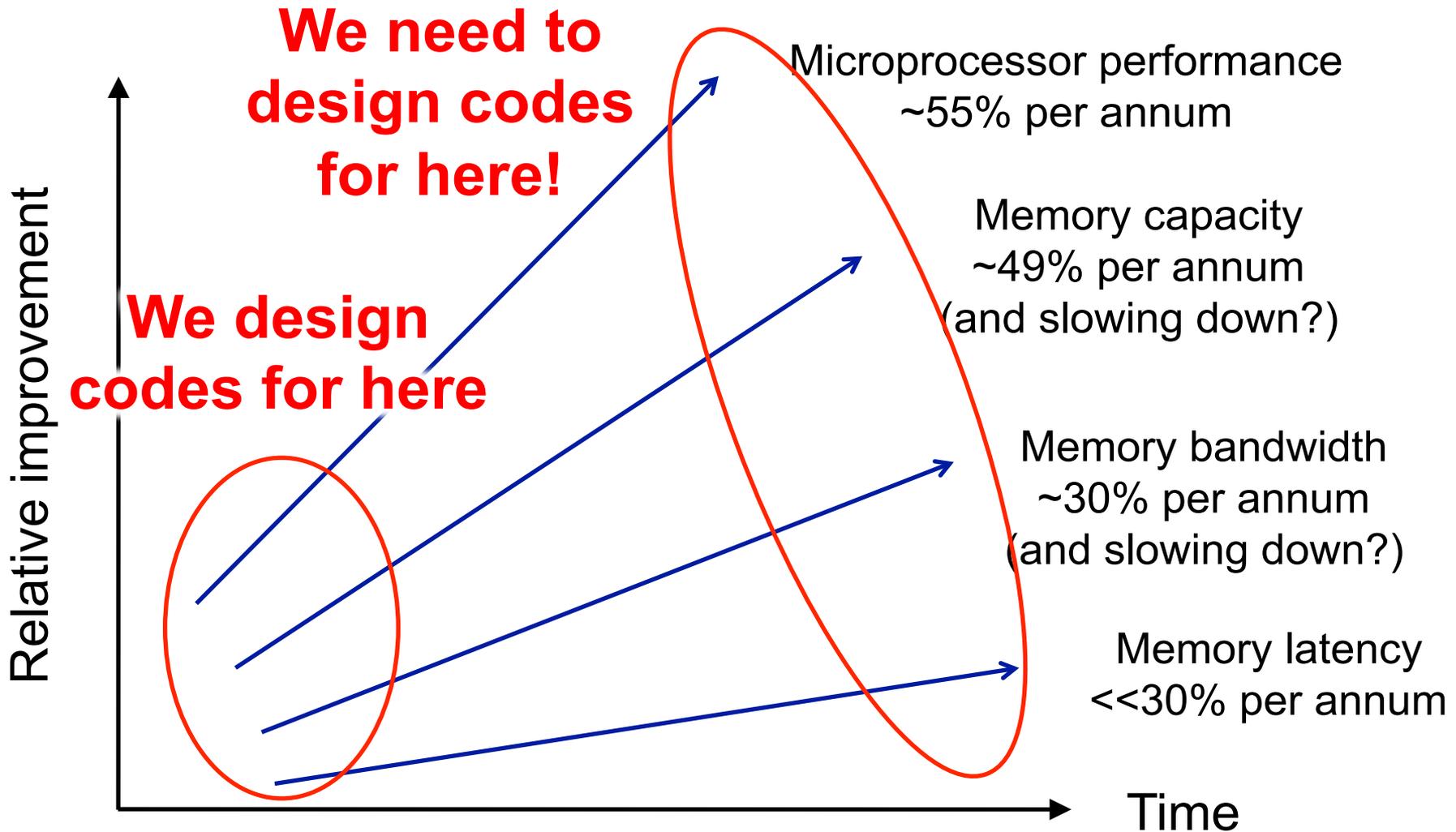
References

Exploiting Spatial Information in Datasets To Enable Fault Tolerant Sparse Matrix Solvers, R. Hunt and S. McIntosh-Smith, FTS, IEEE Cluster, Chicago, Sep 8th 2015

Application-Based Fault Tolerance Techniques for Sparse Matrix Solvers, S. McIntosh-Smith, R. Hunt, J. Price and A. Vesztry, to appear in IJHPCA, 2016

A parting thought

🌟 Long-term fundamental trends



🔥 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-OpenSrc>

SNAP:

https://github.com/UoB-HPC/SNAP_MPI_OpenCL