University of BRISTOL

Investigating the Performance Portability Capabilities of OpenMP 4.0, Kokkos and RAJA

Using TeaLeaf and other mini-apps to assess the performance portability of modern parallel programming models

Matt Martineau - UoB (m.martineau@bristol.ac.uk)

Simon McIntosh-Smith - UoB (cssnmis@bristol.ac.uk)

Wayne Gaudin – UK Atomic Weapons Establishment

University of BRISTOL

Agenda

- Which mini-apps?
- How do you program with each model?
- Which models perform best?
- Conclusions

University of BRISTOL

Mini-apps

TeaLeaf (2d)

Implicit, sparse, matrix-free solvers for heat conduction equation on structured grid, memory bandwidth bound

Solvers: Conjugate Gradient (CG), Chebyshev, Preconditioned Polynomial CG (PPCG)

CloverLeaf (2d)

Lagrangian-Eulerian hydrodynamics – explicit solver on structured grid, memory bandwidth bound

Bristol University Docking Engine (BUDE)

Molecular docking benchmark that uses an Evolutionary Monte Carlo technique, compute bound

University of BRISTOL

The Porting Process

- New ports with emerging parallel programming models:
  - **TeaLeaf**: *Kokkos, RAJA, OpenMP 4.0, OpenACC*
  - **CloverLeaf**: *OpenACC, OpenMP 4.0*
  - **BUDE**: *OpenACC, OpenMP 4.0*

- Developed or utilised existing ports to gauge performance bounds:
  - OpenCL, CUDA, OpenMP 3.0

University of BRISTOL

CUDA Code Sample

```
// CUDA kernel for CG solver
__global__ void cg_calc_p(/*...*/)
{
    // Get global id of thread
    int gid = threadIdx.x + blockIdx.x*blockDim.x;
    int col = gid % x;
    int row = gid / x;

    // Exclude halo region from computation
    if(col >= pad && col < x-pad && row >= pad && row < y-pad)
    {
        p[index] = r[index] + beta*p[index];
    }
}

// Execute with n blocks of 128 threads
cg_calc_p<<< n, 128 >>>(/*...*/);
```

University of BRISTOL

OpenMP 4.0 Code Sample

```
// Setup device data environment
#pragma omp target data \
    map(to: r[:r_len]) map(tofrom: p[:p_len])
{
    // Offload calculation using resident data
    #pragma omp target teams distribute
    for(int jj = pad; jj < y-pad; ++jj)
    {
        for(int kk = pad; kk < x-pad; ++kk)
        {
            int index = jj * x + kk;
            p[index] = beta * p[index] + r[index];
        }
    }
} // Only p is read back from device
```

University of BRISTOL

OpenMP 4.0 Alternatives

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

// Intel KNC targeting
#pragma omp target
#pragma omp parallel for
for(/*...*/) {}

// Clang GPU targeting
#pragma omp target teams distribute
#pragma omp parallel for
for(/*...*/) {}
```

University of BRISTOL

RAJA Code Sample

```
// Global execution policy for IndexSets
typedef RAJA::IndexSet::ExecPolicy<
    RAJA::seq_segit, RAJA::omp_parallel_for_exec> policy;

// Custom box segment taking box coords
RAJA::BoxSegment box(/*...*/)
RAJA::IndexSet inner_domain_list;
inner_domain_list.push_back(box);

// Halo region excluded by virtue of the BoxSegment
RAJA::forall<policy>(inner_domain_list, [=] RAJA_DEVICE (int index)
{
    p[index] = beta*p[index] + r[index];
});
```

University of BRISTOL

Kokkos Code Sample

```
// Define the target device type
#define DEVICE Kokkos::OpenMP

// Globally initialise the execution space and views
Kokkos::Initialize();
Kokkos::View<double*, DEVICE> p("p", x*y);

Kokkos::parallel_for(x*y, KOKKOS_LAMBDA (int index)
{
    int kk = index % x;
    int jj = index / x;

    // Exclude halo region from computation
    if(kk >= pad && kk < x - pad && jj >= pad && jj < y - pad)
    {
        p(index) = beta*p(index) + r(index);
    }
});
```

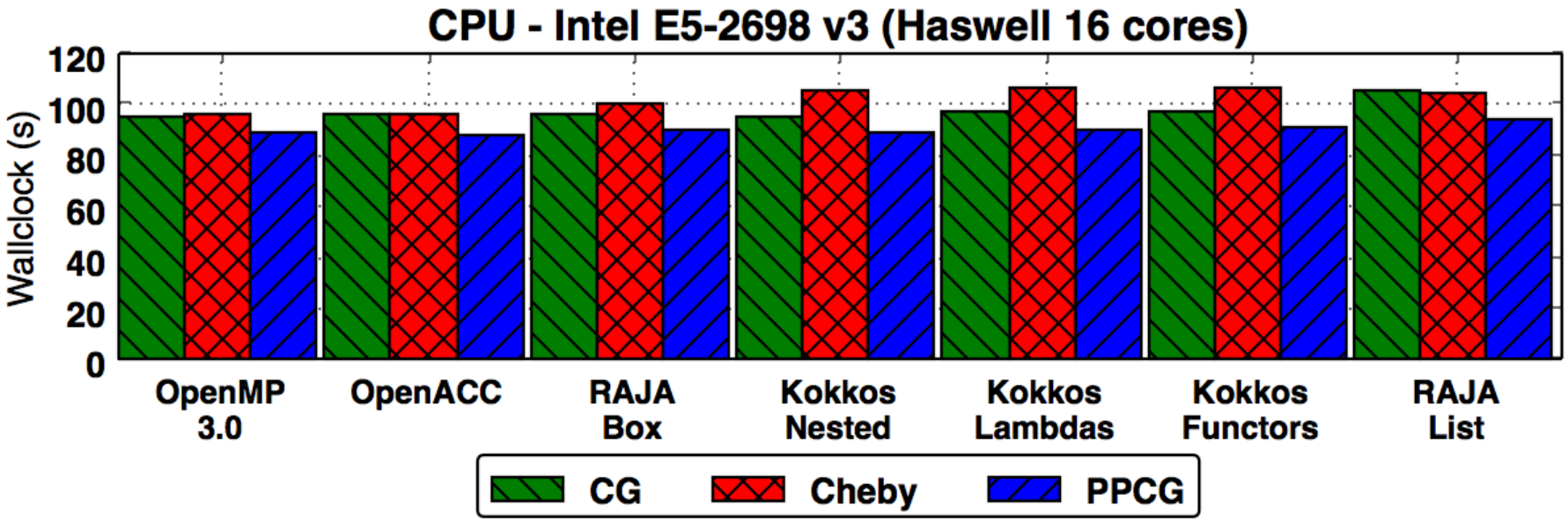
University of BRISTOL

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, whilst aiming for performance portability
- Solved *4096x4096* problem, the point of mesh convergence, for *single iteration*

University of BRISTOL

TeaLeaf - CPU

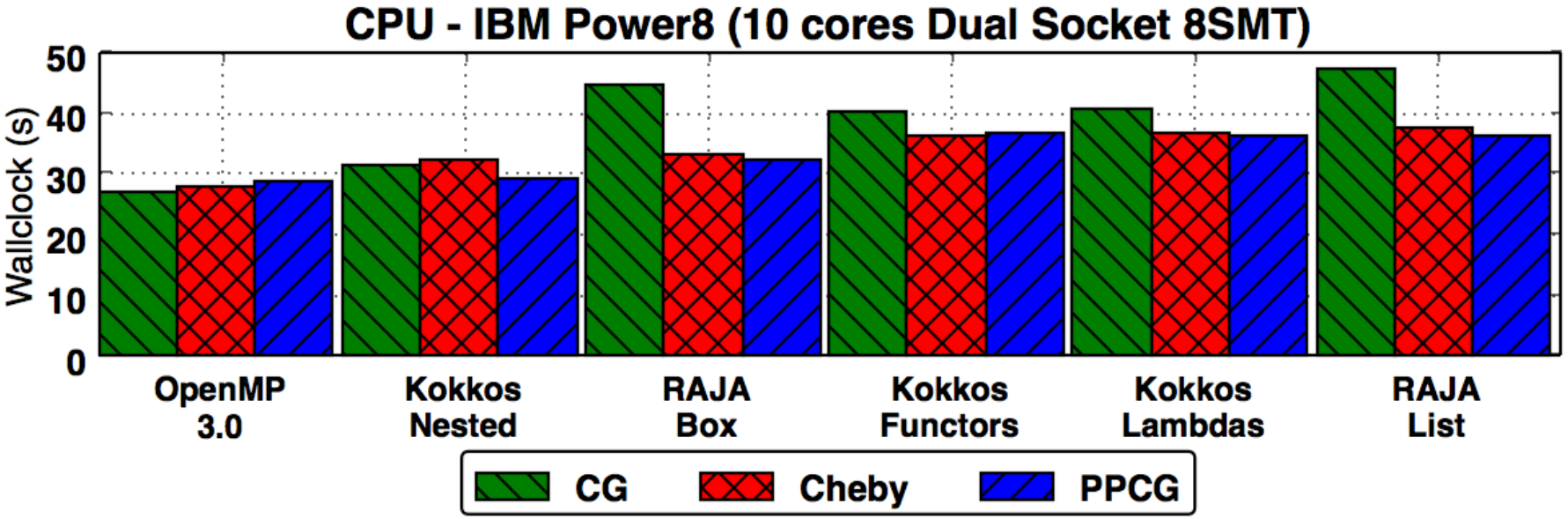


Intel Compilers 16.0.1 (OpenMP, Kokkos, RAJA), and PGI Compilers 15.10 (OpenACC)

At most 12% runtime penalty for modern Intel CPU, nearly identical relative performance seen with Ivy Bridge

University of BRISTOL

TeaLeaf – Power8

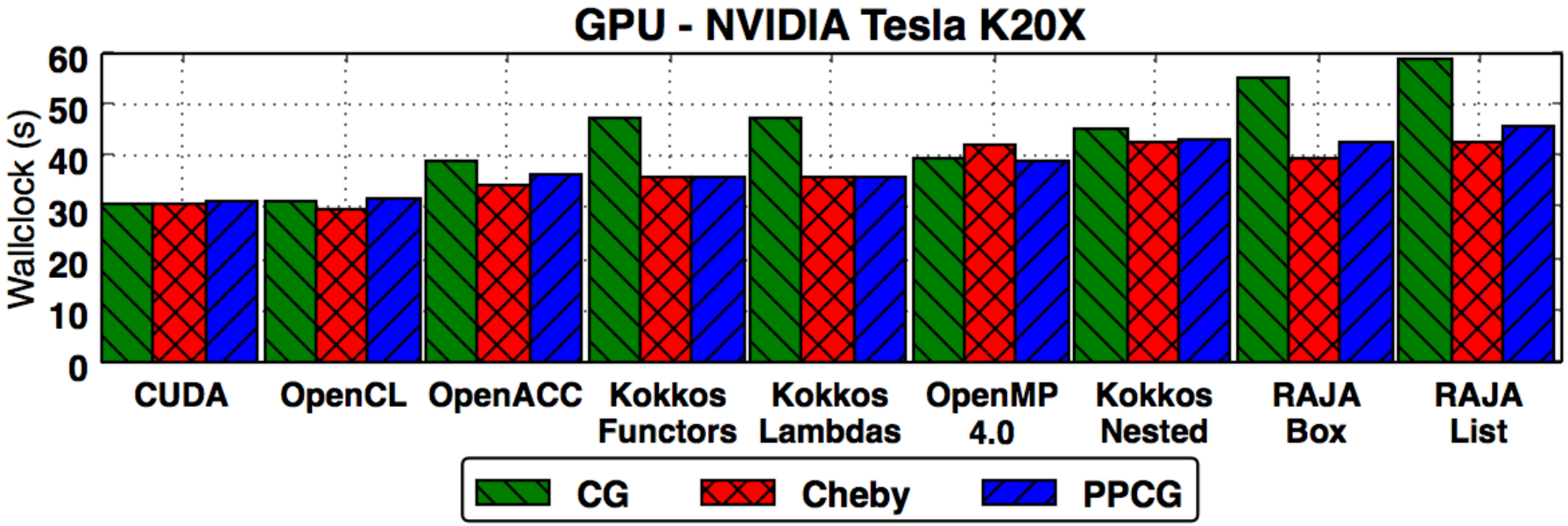


GCC 4.9.1 (OpenMP, Kokkos, RAJA)

Results generally good, and particularly for optimised Kokkos Nested and RAJA Box versions

University of BRISTOL

TeaLeaf – GPU



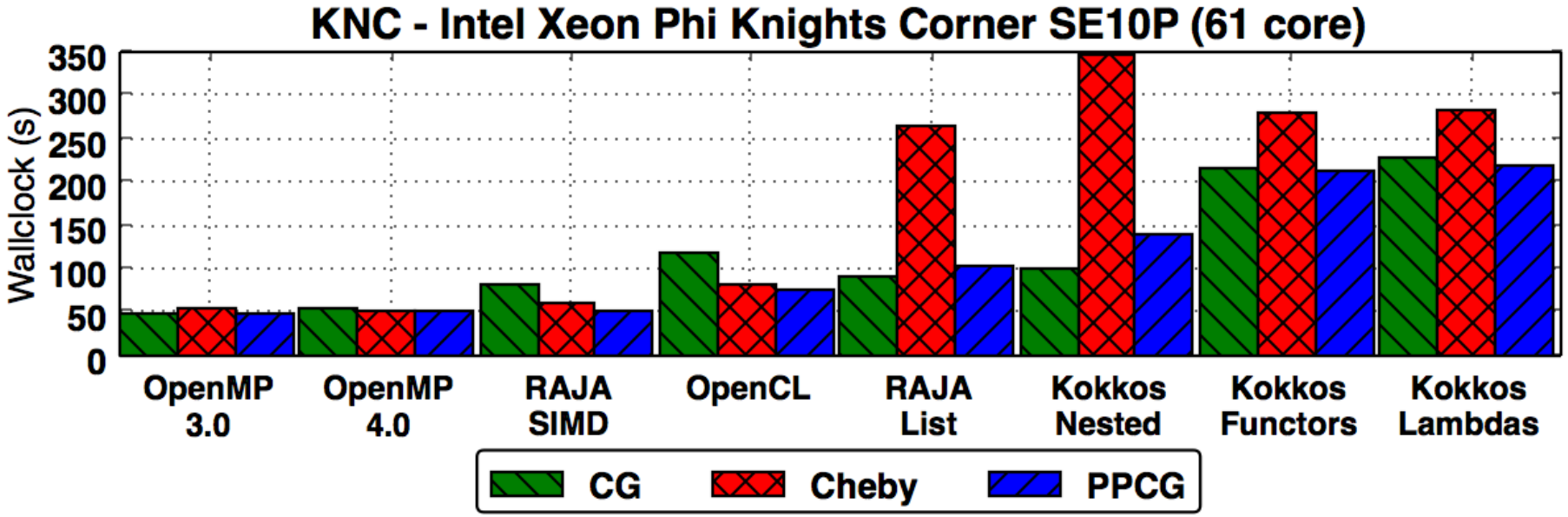
CUDA Toolkit 7.5 (All), PGI 15.10 (OpenACC), CCE 8.4.4 (OpenMP 4.0)

Performance bug with CG again present in some cases

Shows that all portable models have a small runtime penalty

University of BRISTOL

TeaLeaf – KNC



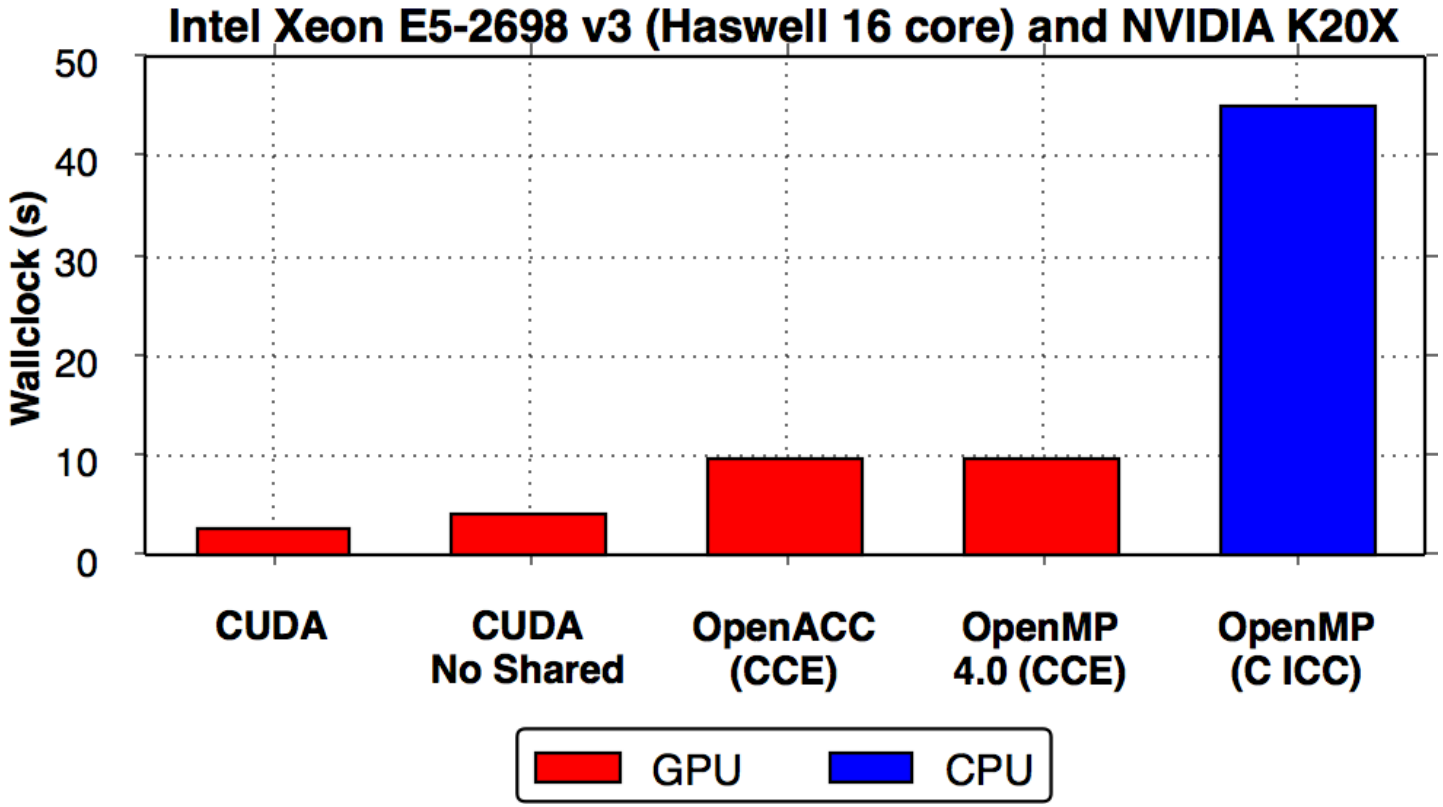
Intel Compilers 16.0.1 (OpenMP, RAJA, Kokkos), Intel OpenCL (OpenCL)

Achieving good performance is more challenging on KNC

Vectorisation was a very important factor

University of BRISTOL

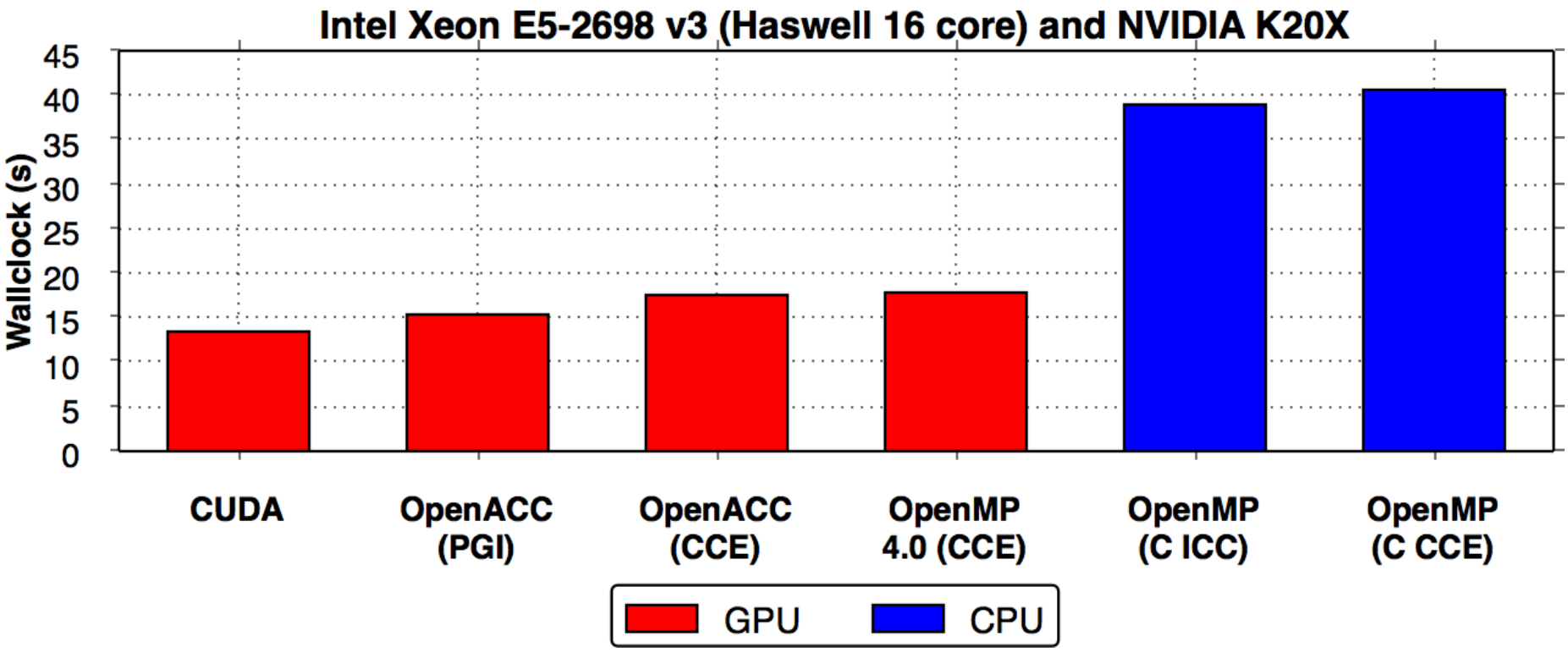
BUDE – CPU & GPU



Required at least 2.2x runtime, shared optimisation increased this even further

University of BRISTOL

CloverLeaf – CPU & GPU



At most 1.3x performance penalty, re-emphasises good performance of directive-based models for such problems

University of BRISTOL

Conclusions

- There are already lots of maturing models
- You can balance performance and complexity
- The performance portable models can achieve performance that is close to low-level APIs
- **Your** best choice might depend on:
  - Productivity and potential for tuning
  - The dominant language of your applications

University of BRISTOL

Acknowledgements

- David Beckingsale, Jeff Keasler, Rich Hornung, LLNL, for help with RAJA port
- Carter Edwards, Christian Trott, Sandia, for help with Kokkos port
- The Intel Parallel Computing Center (IPCC) at Bristol
- Alistair Hart and Cray Inc. for support and provision of the Cray XC40 Swan supercomputer
- EPSRC for funding the research

University of BRISTOL

Any Questions?

Mini-apps including TeaLeaf, CloverLeaf, SNAP (for GPUs), and BUDE
https://github.com/UK-MAC/
https://github.com/UoB-HPC/

Assessing the Performance Portability of Modern Parallel Programming Models using TeaLeaf
Martineau, M., McIntosh-Smith, S. & Gaudin, W.
Submitted to Concurrency and Computation: Practice and Experience (April 2016)

Optimising Sparse Iterative Solvers for Many-Core Computer Architectures
Boulton, M., McIntosh-Smith S., Gaudin, W., & Garrett, P. Presented at UKMAC'14, Cambridge, December 2014

Evaluating OpenMP 4.0's Effectiveness as a Heterogeneous Parallel Programming Model
Martineau, M., McIntosh-Smith, S. & Gaudin, W.
Presenting at HIPS Workshop (IPDPS), Chicago, May 2016

University of BRISTOL

Extra Slides

University of BRISTOL

SNAP

- High dimensionality and sweep presents an interesting difficulty for many models
- We were not able to accelerate with CCE on GPU because of use of indirection arrays
- OpenACC using PGI does successfully achieve within 50% runtime of hand-optimised CUDA implementation