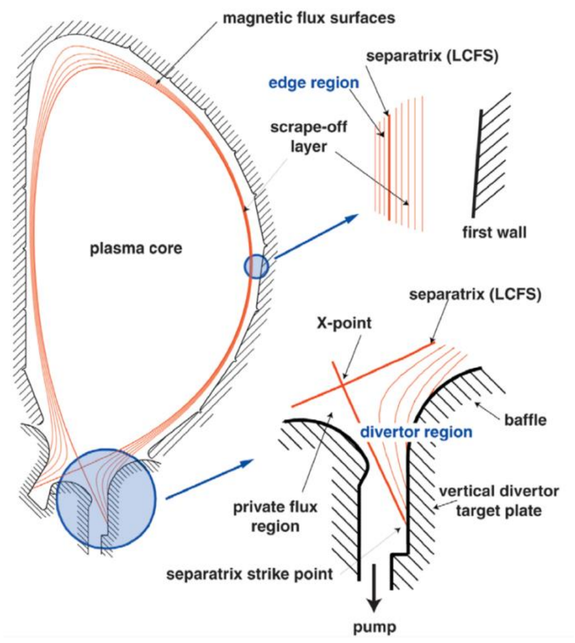
Approaches to Performance Portability for Fusion Applications

Steven Wright[1], Ben Dudson[1], Peter Hill[1], David Dickinson[1], Edward Higgins[1], Gihan Mudalige[2], Ben McMillan[2] and Tom Goffrey[2]
[1]University of York, [2]University of Warwick

Project NEPTUNE

NEPTUNE (*NEutrals & Plasma TUrbulence Numerics for the Exascale*) is the *Fusion Modelling System* use case for the ExCALIBUR Programme.

The goal is to develop a code that can make efficient use of current Petascale and future Exascale hardware in order to draw insights from ITER, and to guide and optimise the design of the UK demonstration Nuclear Fusion Power Plant (STEP).

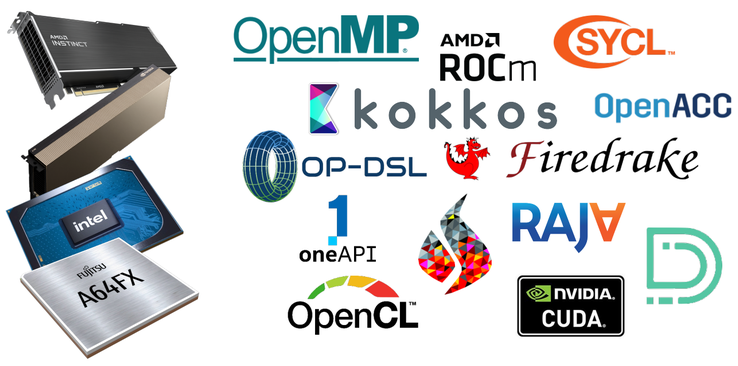


The initial focus is on simulation of the edge and divertor regions (or the “exhaust system”).

The **Support and Coordination** work package is focussed on establishing a series of best practices in engineering such an Exascale-ready simulation code.

Context

Hardware is diversifying. There is a proliferation of hardware and programming models. Almost all pre- and post-Exascale systems will be heterogenous.



Most of the FLOP/s will be provided by GPU accelerators from NVIDIA, AMD, and Intel. These will sit alongside x86_64 and ARM CPU architectures from Intel, AMD, NVIDIA, Fujitsu, and possibly others.

Each architecture might require a specific parallel programming model for optimal performance (e.g. CUDA for NVIDIA).

Avoiding vendor-lock-in requires an approach that is portable between architectures but also performant.

Evaluation

Our evaluation is based on mini-apps that are available in a range of parallel programming models that implement algorithms that are of interest to NEPTUNE.

In particular, we have been periodically evaluating applications that implement **fluid methods** and **particle methods**. We evaluate the performance portability of each available implementations with the Pennycook metric [1].

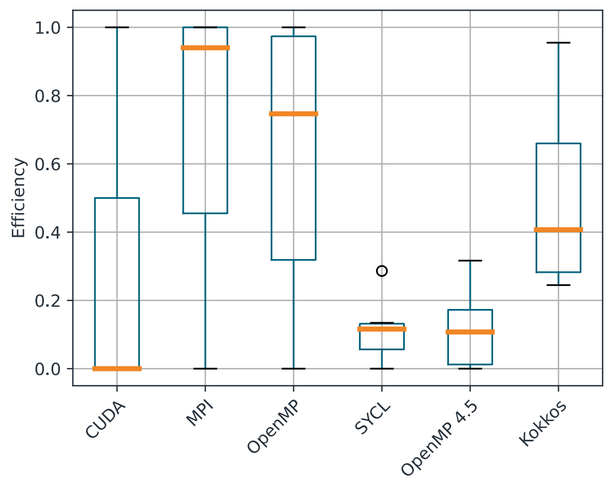
$$\Phi(a,p,H) = \begin{cases} \dfrac{|H|}{\sum_{i \in H} \dfrac{1}{e_i(a,p)}} & \text{if } i \text{ is supported } \forall i \in H \\ 0 & \text{otherwise} \end{cases}$$

Since this may hide useful insights, we visualise portability with the box plot and cascade plot visualisations from Sewall et al. [2].

Fluid Methods

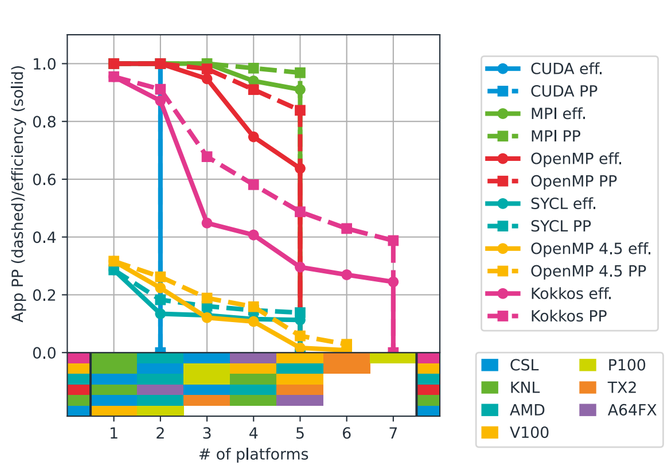
It is often sufficient to treat plasma as a fluid, using various computational schemes. Our evaluation is currently based finite difference (TeaLeaf), finite element (miniFE) and high-order finite element (Laghos) schemes.

miniFE from the Mantevo suite implements a heat diffusion problem in 3D on an unstructured finite element mesh.



Native implementations (CUDA, MPI, OpenMP) achieve the highest levels of performance on their respective platforms. Kokkos is the only fully portable solution, but may lead to half the performance or worse.

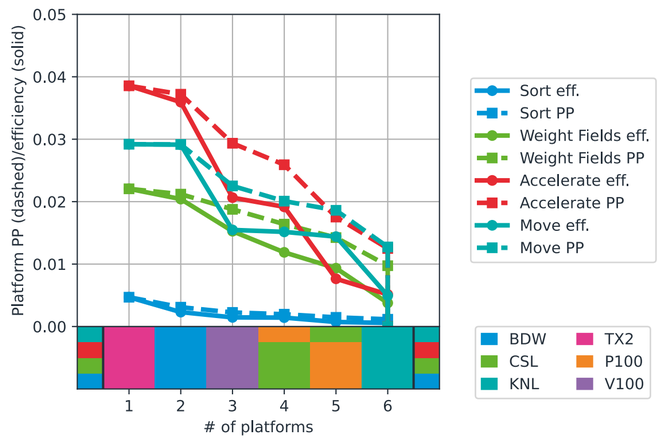
OpenMP 4.5 and SYCL both perform poorly (likely due to immature compiler support). The cascade plot better shows how performance changes as platforms are added to the evaluation set.



Particle Methods

The particle-in-cell (PIC) method is typically used when a fluid model is insufficient. Our evaluation of particle methods is based on three Kokkos-based PIC codes (CabanaPIC, VPIC and EMPIRE-PIC [3]).

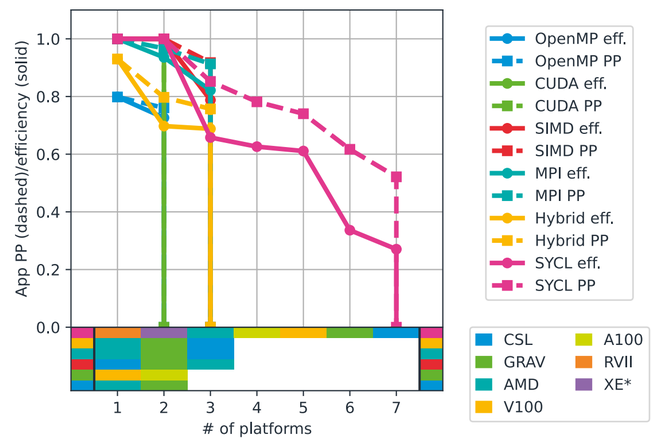
As these applications are only available in a single programming model, we can only evaluate each kernel with respect to the platform’s peak performance.



Across each of the kernels (except sort, which has a low arithmetic intensity), we typically achieve a fraction of peak performance (1-4%), and this approximately correlates with each platform’s available memory bandwidth.

Highlights

- Pragma-based approaches are the easiest to implement and can offer good portability. Achieving high performance on accelerators often requires different directives (and therefore multiple implementations).
- Template-based programming models offer good portability and programmability. SYCL should offer similar performance as compilers mature (see data from MG-CFD below [4]).



- Higher-level DSLs (e.g. Firedrake, UFL) may allow scientists to be more expressive, and can code-gen to a secondary programming model. Thus, they may be able to provide better portability and productivity, but require complex back-end development.

References

[1] S.J. Pennycook, et al. Implications of a metric for performance portability. Future Generation Computer Systems, 92:947 –958, 2019.

[2] J.D. Sewall, et al. Interpreting and visualizing performance portability metrics. In 2020 P3HPC Workshop, pages 14–24, 2020.

[3] M.T. Bettencourt, et al. EMPIRE-PIC: A Performance Portable Unstructured Particle-in-Cell Code. Comms. in Computational Physics, 30(4):1232–1268, August 2021

[4] I.Z. Reguly, et al. Under the Hood of SYCL – An Initial Performance Analysis With an Unstructured-mesh CFD Application, International Supercomputing Conference (ISC 2021), June 2021

University of York
Warwick The University of Warwick
Met Office
UK Research and Innovation
Department of Computer Science
University of York
york.ac.uk

ExCALIBUR 10