T/NA086/20
Code structure and coordination

2047358-TN-02

Identification of Testbed Platforms and Applications

Steven Wright, Ben Dudson, Peter Hill, and David Dickinson

University of York

Gihan Mudalige

University of Warwick

July 9, 2021

1 Introduction

In our previous report we provided a state-of-the-art review of available hardware and software for the development of post-Exascale applications. The report highlights two key observations:

- The hardware landscape is diversifying. At Exascale there will likely be a diverse range of hardware available, and many of the largest systems will employ heterogeneous/hierarchical parallelism, characterised predominantly by the use of GPU accelerators.
- There is a wide range of approaches to software development which allow for portability between architectures. However, whether these approaches enable us to obtain the best performance on each architecture (performance portability) without significant manual modifications (productivity) is a key question.

The focus of this project is to establish best practice for developing a new plasma-fusion application that might achieve the trinity of *Performance*, *Portability* and *Productivity*.

In this project we seek to evaluate a number of hardware platforms and associated software development methodologies. This report outlines how we will assess these factors for the remainder of the project. We will also identify a number of representative applications, in the form of mini-applications that implement algorithms of interest with similar computational/communication patterns to those likely to be present in the NEPTUNE codebase.

1.1 Evaluating Performance Portability

The basis for this investigation will be to analyse the *performance portability* of a number of software development methodologies using the established metric introduced by Pennycook et al. [1] and restated in Equation 1.

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

The performance portability ($\mathbf{\Phi}$) of an application a, solving problem p, on a given set of platforms H, is calculated by finding the harmonic mean of an applications performance efficiency ($e_i(a,p)$). The performance efficiency for each platform can be calculated by comparing achieved performance against the best recorded (possibly non-portable) performance on each individual target platform (i.e. *the application efficiency*, or by comparing the achieved performance against the theoretical maximum performance achievable on each individual platform (i.e. *the architectural efficiency*). Should the application fail to run on one of the target platforms, a performance portability score of 0 is awarded.

The work by Pennycook et al. highlights a number of alternative measures of performance portability, highlighting their shortcomings at providing actionable insights [1]. They outline five criteria a useful metric should aspire to, and then demonstrate how their metric meets each of these criteria. Specifically, a useful metric should: (i) be measured specific to a set of platforms of interest H; (ii) be independent of the absolute performance across H; (iii) be zero if a platform in H is unsupported, and approach zero as the performance of platforms in H approach zero; (iv) increase if performance increases on any platform in H; and (v) be directly proportional to the sum of scores across H.

Since publication of this metric, it has been used extensively to assess the performance portability of a number of applications and programming models [2, 3, 4, 5, 6, 7]. This project aims to replicate this effort with a focus on applications and algorithms of interest to the plasma fusion community.

While a single, numeric metric has a number of advantages, there are also some shortcomings. For example, if a particular application fails to run on one platform then it will score 0, even if the application is performant on all other platforms. To overcome issues such as this, Sewall et al. have proposed a number of methods for visualising performance portability metrics [8].

The first of these is box plots, as demonstrated on synthetic data in Figure 1.

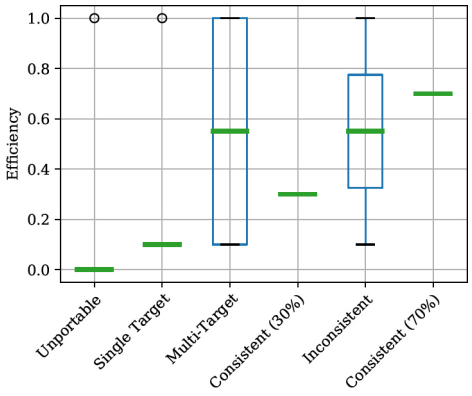

Figure 1: Visualising performance portability with box plots [8]

In both Figure 1 and Figure 2: an *unportable* application does not run on one or more of the available platforms; a *single target* solution runs well on a single platform, and achieves 10% performance on all other platforms; a *multi-target* application runs at 100% on half of the platforms, and 10% on all other platforms; an *inconsistent* solution performs increasingly better on each platform (from 10% to up 100%); and the *consistent* applications perform at 30% and 70% on all platforms, respectively.

In Figure 1, we can see that although the unportable application scores 0, there is an outlier showing that the application is performant on some of the platforms – information that is lost when evaluating based on a single numeric metric. For the other synthetic datasets, we see the performance portability ($\mathbf{\Phi}$), along with information about the range of efficiencies across all of the platforms. For some applications or kernels, it may be the case that a performance portability profile like that of the multi-target solution is acceptable, whereas for others, a consistent 70% might be more appropriate. Figures such as this can help us make these assessments, without relying on a single piece of information.

The second visualisation technique proposed by Sewall et al. is cascade plots, as shown in Figure 2. In these plots, the target platforms are labelled A-J, and plotted beneath the graph. Each application is profiled based on an increasing set of platforms (ordered from most efficient to least for each application), where the filled lines plot the platform efficiencies, and the dotted lines show

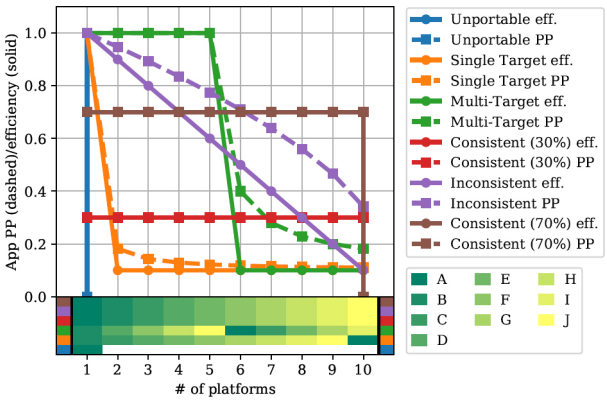

Figure 2: Visualising performance portability with cascade plots [8]

the corresponding performance portability as the platform set grows. Again, visualising performance portability data using these visual heuristics allows a developer to make a reasoned assessment about what might be acceptable for a particular application or kernel.

2 Proxy Applications

The exploratory stage of NEPTUNE includes a number of projects that are investigating the behaviour of plasmas through proxy applications. The applications currently being used broadly fall in to two categories, *fluid models* and *particle models*. In particular, T/NA078/20 is using Nektar++ to explore the performance of spectral elements, T/NA083/20 is focused on building a fluid referent model in both Bout++ and Nektar++, and T/NA079/20 is exploring the use of particle methods with the EPOCH particle-in-cell (PIC) code. It is therefore likely that the resultant NEPTUNE software stack will be a fluid model, based on the output of T/NA078/20 and T/NA083/20, coupled with a particle model, based on the output of T/NA079/20.

The three aforementioned applications are the result of many years of development and typically consist of many thousands of lines of C/C++ or Fortran. They are already widely used by the UK's scientific computing community on a diverse range of problems.

Prior to the development of the NEPTUNE software stack, it is prudent to assess the wide range of available technologies, without the associated burden of redeveloping these mature simulation applications into new programming frameworks. In this project, we have therefore decided to identify a series mini-applications that implement key computational algorithms that are similar to those used by the NEPTUNE proxy applications. These mini-applications are typically limited to a few thousand lines of code and are often available implemented in a wide range of programming frameworks already.

Notable collections of such mini-applications includes Rodinia [9], UK-MAC [10], the NAS Parallel Benchmarks [11], the ECP Proxy Apps [12] and the SPEC benchmarks [13]. In this section, we will discuss the applications we have identified from these benchmark suites, that may be relevant to our performance investigations.

2.1 Fluid Models

As previously noted, the fluid modelling aspects of the NEPTUNE project are largely focused on the use of **Bout++** [14, 15] and **Nektar++** [16]. Bout++ is a framework for writing fluid and plasma simulations in curvilinear geometry, implemented using a finite-difference method, while Nektar++ is a framework for solving computational fluid dynamics problems using the spectral element method.

Both applications are large C++ applications designed primarily for execution across homogeneous clusters. Parallelisation across a cluster in both applications is achieved using MPI, with Bout++ additionally capable of on-node parallelism with OpenMP. GPU acceleration is under development in both applications, through RAJA and HYPRE in Bout++, and through OpenACC in Nektar++ [17].

Rather than redevelop these applications, this project has instead identified a series of mini-applications that implement similar computational schemes. Specifically, we have identified a finite difference mini-app, two finite element mini-apps and one spectral element mini-app, each of which are implemented in a range of programming models for rapid evaluation of approaches to performance portability.

TeaLeaf

TeaLeaf is a finite difference mini-app that solves the linear heat conduction equation on a regular grid using a 5-point stencil. It has been used extensively in studying performance portability already [7, 2, 18, 19], and is available implemented using CUDA, HYPRE, OpenCL, PETSc and Trilinos[1].

miniFE

miniFE is a finite element mini-app, and part of the Mantevo benchmark suite [20, 21, 22, 23]. It implements an unstructured implicit finite element method and is available implemented using CUDA, Kokkos, OpenMP and OpenMP with offload[2].

Laghos

Laghos is a mini-app that is part of the ECP Proxy Applications suite [24, 25, 23]. It implements a high-order curvilinear finite element scheme on an unstructured mesh. It uses HYPRE for parallel linear algebra, and is additionally available in CUDA, RAJA and OpenMP implementations[3].

Nekbone

Nekbone is a mini-app that is representative of one of the core kernels of the incompressible Navier-Stokes solver Nek5000, from Argonne National Laboratory [26, 27, 28, 23]. Like Nek5000, it uses a high-order spectral element discretisation. The mini-app is available implemented using OpenMP, and with accelerator support via CUDA and OpenACC[4].

2.2 Particle Methods

The optimal use of particles in NEPTUNE is currently being explored using the EPOCH particle-in-cell code [29], and its associated mini-app, minEPOCH [30][5]. EPOCH is a PIC code that runs on a structured grid, using a finite differencing scheme and an implementation of the Boris push. Like Bout++ and Nektar++, EPOCH is a mature software package that is used widely by the UK science community, and thus is difficult to evaluate in alternative programming

[1] `http://uk-mac.github.io/TeaLeaf/`
[2] `https://github.com/Mantevo/miniFE`
[3] `https://github.com/CEED/Laghos`
[4] `https://github.com/Nek5000/Nekbone`
[5] `https://github.com/ExCALIBUR-NEPTUNE/minepoch`

models without a significant redevelopment effort. Furthermore, EPOCH (and minEPOCH) is developed in Fortran, making it increasingly difficult to adapt to many new programming models that are heavily based on C++.

The mini-app variant of EPOCH, minEPOCH, is likewise developed in Fortran and thus not appropriate for this study. However, there are a number of particle-based mini-apps that may be of interest to this project, that implement similar particle schemes, backed by a variety of electric/magnetic field solvers.

CabanaPIC

CabanaPIC is a structured PIC code built using the CoPA Cabana library for particle-based simulations [23]. Through the CoPA Cabana library, the application can be parallelised using Kokkos for on-node parallelism and GPU use, and with MPI for off-node parallelism[6].

VPIC/VPIC 2.0

Vector Particle-in-Cell (VPIC) is a general purpose PIC code for modelling kinetic plasmas in one, two or three dimensions, developed at Los Alamos National Laboratory [31]. VPIC is parallelised on-core using vector intrinsics, on-node through pthreads or OpenMP and off-node using MPI. VPIC 2.0 [32] adds support for heterogeneity by using Kokkos to optimise the data layout and allow execution on accelerator devices[7].

EMPIRE-PIC

EMPIRE-PIC is the particle-in-cell solver central the the ElectroMagnetic Plasma In Realistic Environments (EMPIRE) project [33]. It solves Maxwell's equations on an unstructured grid using a finite-element method, and implements the Boris push for particle movement. EMPIRE-PIC makes extensive use of the Trilinos library, and uses Kokkos as its parallel programming model [34, 35].

Each of the three particle-based mini-apps identified implement a PIC algorithm that is similar to that found in EPOCH. However, one weakness of this evaluation set is that all three applications are parallelised on-node through the Kokkos performance portability layer. Currently, we are unaware of any SYCL/DPC++-based PIC codes, however Kokkos has a range of backends

[6] `https://github.com/ECP-copa/CabanaPIC`
[7] `https://github.com/lanl/vpic`

including OpenMP target, and preliminary support for SYCL/DPC++ code generation.

Alongside the minEPOCH mini-app, there was a C++ mini-app port called miniEPOCH that is now orphaned [36], but may prove a useful evaluation vehicle should time allow a porting exercise.

Beyond the PIC method, there are other particle-based applications that we may consider as part of our evaluation, such the molecular dynamic mini-application, miniMD [37]. The evaluation set will be re-evaluated as the project progresses.

3 Evaluation Platforms

The primary focus of this project is to provide an assessment of the options available when developing Exascale-capable software. In the previous section we identified a series of mini-applications that have been implemented using a range of techniques that we believe will be important to developing future-proofed fusion simulations.

Many of these applications have already been evaluated on various platforms by others, and this project does not seek to re-run these experiments. Instead, wherever possible we will seek to collect performance data from existing studies and apply the metrics and visualisation techniques described in Section 1. Where evaluations do not already exist, we will look to evaluate performance on UK-based platforms.

Broadly speaking, we can divide UK-based HPC platforms into two categories, *homogeneous systems* and *heterogeneous systems*. The UK's only Tier-1 system, ARCHER2, is an homogeneous system with an estimated peak performance of 28 PFLOP/s. Across the UK's Tier-2 systems, there is a significant degree of diversity, offering a wide range of homogeneous and heterogeneous platforms/-partitions.

Where we require additional evaluation of these applications, we will be able to leverage access to a number of systems in the UK, such as those listed below. Evaluation on other UK or US based systems may also be possible, through existing collaborations.

3.1 Homogeneous Systems

ARHCER2

The national supercomputer, ARCHER2, is installed at the Edinburgh Parallel Computing Centre (EPCC). ARCHER2 is a Cray Shasta system interconnected with Cray Slingshot fabric. It consists of 5,848 nodes, each with two AMD EPYC Rome CPUs.

Avon

Avon will be a homogeneous cluster of 180 nodes, containing dual Intel Xeon Cascade Lake CPUs installed at the University of Warwick (expected mid-2021). It will be interconnected with Infiniband.

Isambard

The Isambard Tier-2 service is predominantly composed of Marvell ThunderX2 ARM cores, connected by a Cray Aries interconnect. Beside the ThunderX2 cabinet, Isambard also contains a cabinet of Fujistu A64FX processors.

Viking

Viking is a large Linux compute cluster supporting research needs at the University of York. It consists of approximately 170 compute nodes, each with Intel Xeon Skylake CPUs, connected via Infiniband.

Cirrus

The Cirrus cluster, installed at EPCC, consists of 280 compute nodes, each with dual Intel Xeon Broadwell processors. The cluster is connected via Infiniband fabric.

3.2 Heterogeneous Systems

Viking

The Viking cluster, at the University of York, is further bolstered by two GPU nodes, providing a small heterogeneous compute capability. The two GPU nodes each contain four NVIDIA V100 GPUs.

Bede

The Bede system, installed at the University of Durham, has an architecture similar to that found on Summit and Sierra. Bede is a single cabinet of IBM POWER9 CPUs each supporting four NVIDIA V100 GPUs.

Isambard

Alongside the two ARM-based partitions on the Isambard system is the Multi-Architecture Comparison System (MACS). MACS contains four nodes each with two NVIDIA P100 GPUs, and four nodes each with an NVIDIA V100 GPU. It also contains four nodes with AMD EPYC Rome CPUs, and four nodes with Intel Xeon Cascade Lake CPUs. Finally, there are also eight Intel Xeon Phi nodes, and two IBM Power9 nodes with NVIDIA V100 GPUs.

CSD3

The Cambridge Service for Data Driven Discovery (CSD3) provide two supercomputers under EPSRC Tier-2. Peta4 is a system comprising predominantly of Intel Xeon Skylake CPUs, with a small number of Intel Xeon Phil nodes. Wilkes2 provides the largest GPU enabled system in the UK, comprising of 90 nodes each with four NVIDIA P100 GPUs.

Baskerville

The Baskerville system will be the University of Birmingham's Tier-2 cluster. There are 46 compute nodes, each with four NVIDIA A100 GPUs alongside Intel Xeon Ice Lake CPUs.

4 Conclusions

This report has identified a number of mini-applications that implement similar numerical methods to those of interest in the NEPTUNE project. These applications will be our focus for the remainder of this project, using these applications to evaluate approaches to developing performance portable fusion applications.

Our next report will gather performance data from available sources, and will begin the process of evaluating performance across a range of architectures using performance portability metrics and visualisation techniques. Where data is not available it will be gathered from the systems we have at our disposal.

This analysis will allow us to make comparisons between differing programming models and in turn make well reasoned recommendations for the NEPTUNE programme.

References

[1] S.J. Pennycook, J.D. Sewall, and V.W. Lee. Implications of a metric for performance portability. *Future Generation Computer Systems*, 92:947 – 958, 2019.

[2] R. O. Kirk, G. R. Mudalige, I. Z. Reguly, S. A. Wright, M. J. Martineau, and S. A. Jarvis. Achieving Performance Portability for a Heat Conduction Solver Mini-Application on Modern Multi-core Systems. In *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 834–841, Sep. 2017.

[3] Daniela F. Daniel and Jairo Panetta. On applying performance portability metrics. In *2019 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC)*, pages 50–59, 2019.

[4] S. L. Harrell, J. Kitson, R. Bird, S. J. Pennycook, J. Sewall, D. Jacobsen, D. N. Asanza, A. Hsu, H. C. Carrillo, H. Kim, and R. Robey. Effective performance portability. In *2018 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC)*, pages 24–36, Nov 2018.

[5] T. R. Law, R. Kevis, S. Powell, J. Dickson, S. Maheswaran, J. A. Herdman, and S. A. Jarvis. Performance portability of an unstructured hydrodynamics mini-application. In *2018 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC)*, pages 0–12, Nov 2018.

[6] Simon McIntosh-Smith. Performance Portability Across Diverse Computer Architectures. In *P3MA: 4th International Workshop on Performance Portable Programming models for Manycore or Accelerators*, 2019.

[7] Tom Deakin, Simon McIntosh-Smith, James Price, Andrei Poenaru, Patrick Atkinson, Codrin Popa, and Justin Salmon. Performance portability across diverse computer architectures. In *2019 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC)*, pages 1–13, 2019.

[8] Jason Sewall, S. John Pennycook, Douglas Jacobsen, Tom Deakin, and Simon McIntosh-Smith. Interpreting and visualizing performance portabil-

ity metrics. In *2020 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC)*, pages 14–24, 2020.

[9] Shuai Che, Michael Boyer, Jiayuan Meng, David Tarjan, Jeremy W. Sheaffer, Sang-Ha Lee, and Kevin Skadron. Rodinia: A benchmark suite for heterogeneous computing. In *2009 IEEE International Symposium on Workload Characterization (IISWC)*, pages 44–54, 2009.

[10] UK Mini-App Consortium. Uk-mac. `http://uk-mac.github.io` (accessed April 20, 2021), 2021.

[11] David H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, Horst D. Simon, V. Venkatakrishnan, and S. K. Weeratunga. The NAS Parallel Benchmarks. *International Journal of High Performance Computing Applications*, 5(3):63–73, 1991.

[12] Exascale Computing Project. ECP Proxy Applications. `https://proxyapps.exascaleproject.org/` (accessed April 20, 2021), 2021.

[13] Guido Juckeland, William Brantley, Sunita Chandrasekaran, Barbara Chapman, Shuai Che, Mathew Colgrove, Huiyu Feng, Alexander Grund, Robert Henschel, Wen-Mei W. Hwu, Huian Li, Matthias S. Müller, Wolfgang E. Nagel, Maxim Perminov, Pavel Shelepugin, Kevin Skadron, John Stratton, Alexey Titov, Ke Wang, Matthijs van Waveren, Brian Whitney, Sandra Wienke, Rengan Xu, and Kalyan Kumaran. SPEC ACCEL: A Standard Application Suite for Measuring Hardware Accelerator Performance. In Stephen A. Jarvis, Steven A. Wright, and Simon D. Hammond, editors, *High Performance Computing Systems. Performance Modeling, Benchmarking, and Simulation*, pages 46–67. Springer International Publishing, 2015.

[14] Benjamin Daniel Dudson, Peter Alec Hill, David Dickinson, Joseph Parker, Adam Dempsey, Andrew Allen, Arka Bokshi, Brendan Shanahan, Brett Friedman, Chenhao Ma, David Schwörer, Dmitry Meyerson, Eric Grinaker, George Breyiannia, Hasan Muhammed, Haruki Seto, Hong Zhang, Ilon Joseph, Jarrod Leddy, Jed Brown, Jens Madsen, John Omotani, Joshua Sauppe, Kevin Savage, Licheng Wang, Luke Easy, Marta Estarellas, Matt Thomas, Maxim Umansky, Michael Løiten, Minwoo Kim, M Leconte, Nicholas Walkden, Olivier Izacard, Pengwei Xi, Peter Naylor, Fabio Riva,

Sanat Tiwari, Sean Farley, Simon Myers, Tianyang Xia, Tongnyeol Rhee, Xiang Liu, Xueqiao Xu, and Zhanhui Wang. BOUT++, 10 2020.

[15] B D Dudson, M V Umansky, X Q Xu, P B Snyder, and H R Wilson. BOUT++: A framework for parallel plasma fluid simulations. *Computer Physics Communications*, 180:1467–1480, 2009.

[16] C.D. Cantwell, D. Moxey, A. Comerford, A. Bolis, G. Rocco, G. Mengaldo, D. De Grazia, S. Yakovlev, J.-E. Lombard, D. Ekelschot, B. Jordi, H. Xu, Y. Mohamied, C. Eskilsson, B. Nelson, P. Vos, C. Biotto, R.M. Kirby, and S.J. Sherwin. Nektar++: An open-source spectral/hp element framework. *Computer Physics Communications*, 192:205–219, 2015.

[17] Jan Eichstädt. Implementation of High-performance GPU Kernels in Nektar++, 2020.

[18] Matthew Martineau, Simon McIntosh-Smith, and Wayne Gaudin. Assessing the performance portability of modern parallel programming models using tealeaf. *Concurrency and Computation: Practice and Experience*, 29(15):e4117, 2017.

[19] Simon McIntosh-Smith, Matthew Martineau, Tom Deakin, Grzegorz Pawelczak, Wayne Gaudin, Paul Garrett, Wei Liu, Richard Smedley-Stevenson, and David Beckingsale. TeaLeaf: A Mini-Application to Enable Design-Space Explorations for Iterative Sparse Linear Solvers. In *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 842–849, 2017.

[20] Richard Frederick Barrett, Li Tang, and Sharon X. Hu. Performance and Energy Implications for Heterogeneous Computing Systems: A MiniFE Case Study. 12 2014.

[21] Alan B. Williams. Cuda/GPU version of miniFE mini-application. 2 2012.

[22] Meng Wu, Can Yang, Taoran Xiang, and Daning Cheng. The research and optimization of parallel finite element algorithm based on minife. *CoRR*, abs/1505.08023, 2015.

[23] David F. Richards, Yuri Alexeev, Xavier Andrade, Ramesh Balakrishnan, Hal Finkel, Graham Fletcher, Cameron Ibrahim, Wei Jiang, Christoph Junghans, Jeremy Logan, Amanda Lund, Danylo Lykov, Robert Pavel,

Vinay Ramakrishnaiah, et al. FY20 Proxy App Suite Release. Technical Report LLNL-TR-815174, Exascale Computing Project, September 2020.

[24] J. C. Camier. Laghos summary for CTS2 benchmark. Technical Report LLNL-TR-770220, Lawrence Livermore National Laboratory, March 2019.

[25] Robert Anderson, Julian Andrej, Andrew Barker, Jamie Bramwell, Jean-Sylvain Camier, Jakub Cerveny, Veselin Dobrev, Yohann Dudouit, Aaron Fisher, Tzanio Kolev, Will Pazner, Mark Stowell, Vladimir Tomov, Ido Akkerman, Johann Dahm, David Medina, and Stefano Zampini. Mfem: A modular finite element methods library. *Computers & Mathematics with Applications*, 81:42–74, 2021. Development and Application of Open-source Software for Problems with Numerical PDEs.

[26] Ilya Ivanov, Jing Gong, Dana Akhmetova, Ivy Bo Peng, Stefano Markidis, Erwin Laure, Rui Machado, Mirko Rahn, Valeria Bartsch, Alistair Hart, and Paul Fischer. Evaluation of parallel communication models in nekbone, a nek5000 mini-application. In *2015 IEEE International Conference on Cluster Computing*, pages 760–767, 2015.

[27] Stefano Markidis, Jing Gong, Michael Schliephake, Erwin Laure, Alistair Hart, David Henty, Katherine Heisey, and Paul Fischer. Openacc acceleration of the nek5000 spectral element code. *The International Journal of High Performance Computing Applications*, 29(3):311–319, 2015.

[28] Jing Gong, Stefano Markidis, Erwin Laure, Matthew Otten, Paul Fischer, and Misun Min. Nekbone performance on gpus with openacc and cuda fortran implementations. *The Journal of Supercomputing*, 72(11):4160–4180, 2016.

[29] T D Arber, K Bennett, C S Brady, A Lawrence-Douglas, M G Ramsay, N J Sircombe, P Gillies, R G Evans, H Schmitz, A R Bell, and C P Ridgers. Contemporary particle-in-cell approach to laser-plasma modelling. *Plasma Physics and Controlled Fusion*, 57(11):113001, sep 2015.

[30] Michael Bareford. minEPOCH3D Performance and Load Balancing on Cray XC30. Technical Report eCSE03-1, Edinburgh Parallel Computer Centre, 2016.

[31] K. J. Bowers, B. J. Albright, B. Bergen, L. Yin, K. J. Barker, and D. J. Kerbyson. 0.374 Pflop/s Trillion-Particle Kinetic Modeling of Laser Plasma

Interaction on Roadrunner. In *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*, SC '08. IEEE Press, 2008.

[32] Robert Bird, Nigel Tan, Scott V Luedtke, Stephen Harrell, Michela Taufer, and Brian Albright. VPIC 2.0: Next Generation Particle-in-Cell Simulations. *IEEE Transactions on Parallel and Distributed Systems*, pages 1–1, 2021.

[33] Matthew T. Bettencourt and Sidney Shields. EMPIRE Sandia's Next Generation Plasma Tool. Technical Report SAND2019-3233PE, Sandia National Laboratories, March 2019.

[34] Matthew T. Bettencourt, Dominic A. S. Brown, Keith L. Cartwright, Eric C. Cyr, Christian A. Glusa, Paul T. Lin, Stan G. Moore, Duncan A. O. McGregor, Roger P. Pawlowski, Edward G. Phillips, Nathan V. Roberts, Steven A. Wright, Satheesh Maheswaran, John P. Jones, and Stephen A. Jarvis. EMPIRE-PIC: A Performance Portable Unstructured Particle-in-Cell Code. *Communications in Computational Physics*, x(x):1–37, March 2021.

[35] Dominic A.S. Brown, Matthew T. Bettencourt, Steven A. Wright, Satheesh Maheswaran, John P. Jones, and Stephen A. Jarvis. Higher-order particle representation for particle-in-cell simulations. *Journal of Computational Physics*, 435:110255, 2021.

[36] Robert F Bird, Patrick Gillies, Michael R Bareford, Andy Herdman, and Stephen Jarvis. Performance Optimisation of Inertial Confinement Fusion Codes using Mini-applications. *The International Journal of High Performance Computing Applications*, 32(4):570–581, 2018.

[37] S. J. Pennycook and S. A. Jarvis. Developing performance-portable molecular dynamics kernels in opencl. In *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*, pages 386–395, 2012.