Commun. Comput. Phys.
doi: 10.4208/cicp.OA-2020-0261

Vol. **30**, No. 4, pp. 1232-1268
October 2021

EMPIRE-PIC: A Performance Portable Unstructured Particle-in-Cell Code

Matthew T. Bettencourt[1], Dominic A. S. Brown[2,*], Keith L. Cartwright[1], Eric C. Cyr[1], Christian A. Glusa[1], Paul T. Lin[3], Stan G. Moore[1], Duncan A. O. McGregor[1], Roger P. Pawlowski[1], Edward G. Phillips[1], Nathan V. Roberts[1], Steven A. Wright[4], Satheesh Maheswaran[5], John P. Jones[6] and Stephen A. Jarvis[7]

[1] *Sandia National Laboratories, Albuquerque, NM.*
[2] *Department of Computer Science, University of Warwick, UK.*
[3] *Lawrence Berkeley National Laboratory, Berkeley, CA.*
[4] *Department of Computer Science, University of York, UK.*
[5] *Diamond Light Source Ltd, Diamond House, Harwell Science and Innovation Campus, Didcot, UK*
[6] *Atomic Weapons Establishment, Aldermaston, UK.*
[7] *College of Engineering and Physical Sciences, University of Birmingham, UK.*

Received 24 December 2020; Accepted (in revised version) 30 March 2021

Abstract. In this paper we introduce EMPIRE-PIC, a finite element method particle-in-cell (FEM-PIC) application developed at Sandia National Laboratories. The code has been developed in C++ using the Trilinos library and the Kokkos Performance Portability Framework to enable running on multiple modern compute architectures while only requiring maintenance of a single codebase. EMPIRE-PIC is capable of solving both electrostatic and electromagnetic problems in two- and three-dimensions to second-order accuracy in space and time. In this paper we validate the code against three benchmark problems – a simple electron orbit, an electrostatic Langmuir wave, and a transverse electromagnetic wave propagating through a plasma. We demonstrate the performance of EMPIRE-PIC on four different architectures: Intel Haswell CPUs, Intel's Xeon Phi Knights Landing, ARM Thunder-X2 CPUs, and NVIDIA Tesla V100 GPUs attached to IBM POWER9 processors. This analysis demonstrates scalability of the code up to more than two thousand GPUs, and greater than one hundred thousand CPUs.

AMS subject classifications: To be provided by authors

Key words: PIC, electrostatics, electromagnetics, HPC, performance portability.

*Corresponding author. *Email addresses:* `mbetten@sandia.gov` (M. Bettencourt), `Dominic.Brown@warwick.ac.uk` (D. A. S. Brown)

1 Introduction

High Performance Computing (HPC) provides huge benefit to the scientific community and has been especially useful within fields that require experiments that may be infeasible, and/or expensive to conduct physically. As a consequence of the continually rising computational performance of HPC systems, scientists are able to conduct research of ever increasing complexity. This improved capability will continue to grow with the move towards Exascale computing (the ability to carry out at least 10^{18} floating point operations per second), a major milestone in the field of HPC.

This significant improvement in performance has been accompanied by an increasing amount of diversity in modern compute architectures. For example, heterogeneous systems that make use of Graphics Processing Unit (GPU) accelerators or many-core CPUs are rapidly becoming more prevalent as we continue to move away from the traditional homogeneous cluster systems that were previously the norm [9]. As of June 2020, six out of the top ten supercomputers depend on heterogeneous architectures to achieve their compute performance [5]. The upcoming Exascale machines Aurora and Frontier will follow this same trend by using Intel and AMD GPUs respectively. As a result, it is now highly desirable for scientific codes to be able to perform well across a wide variety of systems, a concept often referred to as 'performance portability' [59]. However, this increase in architectural diversity brings greater difficulty in implementing production codes that are capable of fully exploiting the available hardware resources when compared to the traditional Single Program, Multiple Data (SPMD) MPI-based approach of the past. This problem is exacerbated by the fact that each architecture requires specific optimizations in order to achieve peak performance. Scientific codes must be able to adapt to this changing landscape without the difficulty of developing and maintaining several versions of the same application.

There are various standards that have been proposed to remedy this issue by providing directives to the compiler that sections of code should be run in parallel and/or on a given device – commonly an accelerator card. These include OpenMP [16] and OpenACC [31]. Another approach being considered to aid performance portability is the use of parallel programming frameworks or libraries. Examples include Kokkos [36], from Sandia National Laboratories (SNL), and RAJA [47], from Lawrence Livermore National Laboratory (LLNL), both of which make use of C++ template meta-programming to inject hardware-specific device code, targeting a system during compilation. Other notable examples of parallel programming frameworks include Khronos' OpenCL [2] and SYCL [42], and Intel's newly developed OneAPI [4].

HPC contributes to a variety of scientific fields, with the areas of fusion energy research, and the behavior of plasmas under various conditions being notable examples. Particle-in-Cell (PIC) [14, 32, 45, 55] codes are commonly used to carry out simulations of charged particles under the influence of electric and magnetic fields. Examples in fusion energy research include both Inertial Confinement Fusion (ICF) and Magnetic Confinement Fusion (MCF) devices. Such devices include the National Ignition Facil-