PERFORMANCE ANALYSIS ON THE IBM BLUE GENE/P FOR WAKEFIELD CALCULATIONS

Misun Min*‡ and Paul Fischer‡
‡Mathematics and Computer Science, Argonne National Laboratory, Argonne, IL 60439, USA

Abstract

Accurate and efficient simulations will significantly reduce the cost and the risk in the design process for various applications in accelerator design. We improved capability of the Argonne-developed high-fidelity wakefield simulation code, NekCEM, by upgrading pre-setup and communication subroutines for high-performance simulations beyond petascale. We present a detailed study of parallel performance of NekCEM on the IBM Blue Gene/P at Argonne. We demonstrate strong scaling up to P=131,072 cores using up to more than 1.1 billion grid points with the total number of elements up to E=273,000 and N=15 which gives 75% efficiency at 8,530 grid points per core compared to the base case of P=16,384 cores.

INTRODUCTION

During the past decade, discontinuous Galerkin (DG) methods have emerged as a powerful algorithmic tool for the numerical solution of PDEs and led to a dramatic expansion for applications based on electromagnetics, acoustics, elasticity, shallow water, plasma physics, gas dynamics, and flow problems. For time-dependent, wave-dominated problems, the DG method reflects the underlying dynamics more flexibly than does the continuous Galerkin method, by carefully designing the numerical flux to ensure stability.

It is well known that high-order methods such as spectral, spectal element, and spectral element discontinuous Galerkin (SEDG) methods converge exponentially for smooth solutions with a faster rate of convergence than for any fixed-order method. The spectral rate of convergence of the SEDG method in NekCEM enables a converged solution with a minimum 5 grid points per wavelength, with minimal numerical dispersion after a long time integration [1, 2]. One can achieve engineering-level accuracy, approximately 1e-3∼1e-5, by using high-order approximation N>10, using 6–8 grid points per wavelength.

Another important feature of a high-order scheme is that the number of degrees of freedom to obtain a given accuracy also decreases as the order of the approximation grows. The motivation to consider high-order SEDG methods can be summarized as follows: *fewer grid points per wavelength* required to achieve a given accuracy; efficiency, with *CPU time depending on the degrees of freedom*, not the order of approximation and geometric flexibility with *body-conforming* meshes.

Our solution approach based on the DG framework solves the Maxwell's equations for wakefield calculations [6, 7]. The computational domain Ω is decomposed into nonoverlapping hexahedral elements Ω^e such that $\Omega = \cup_{e=1}^{E}\Omega^e$. In the DG framework, the Maxwell equations are written in a conservation form by defining a flux function F as in q [8] and a weak formulation is obtained by multiplying a test function ϕ to the the governing equation and integrating it by parts twice:

$$\left(Q\frac{\partial q}{\partial t} + \nabla\cdot F(q) - S, \phi\right)_{\Omega^e} = (\hat{n}\cdot[F - F^*], \phi)_{\partial\Omega^e} \quad (1)$$

where we define Q = diag$\{\mu, \mu, \mu, \epsilon, \epsilon, \epsilon\}$, the field vector q = $[H, E]^T$, the flux $F(q)$ = $[F_H, F_E]^T$ with F_H = $-e_i\times E$ and F_E = $e_i\times H$ for the electric field E and magnetic field H, and the source term S = $[S_H, S_E]$ with S_H = $(0,0,0)^T$ and S_E = $(0,0,J)^T$. The current source J is defined for a Gaussian beam as in [6]. The numerical flux F^* is chosen as defined in [8]. A local solution q_N on each Ω^e can be defined as $q_N(\tilde{x},t) = \sum_{j=0}^{n} q_j(t)L_j(\tilde{x})$ where $q_j(t)$ is the solution at n grid points $\tilde{x}_j$ on Ω^e, and $L_j(\tilde{x})$ is the three-dimensional Legendre Lagrange interpolation polynomial of degree N associated with $n = E(N+1)^3$ nodes [1]. The local discontinuous test function is chosen to be $\phi = L_i(\tilde{x})$ and the Gauss-Lobatto quadrature is applied for the spatial integration. The Nth-order tensor-product brick elements are mapped to body-fitted coordinates, with weakly imposed boundary conditions. Such a grid basis forms *a local diagonal mass matrix*, resulting in no cost for inversion. For the time advancing, the code currently supports the five-stage, fourth-order Runge-Kutta and exponential time integration methods.

We performed wakefield calculations for an electron bunch length of 1 cm and demonstrate the wakepotential using five times larger grid spacings compared to the second-order finite-difference time-domain (FDTD)

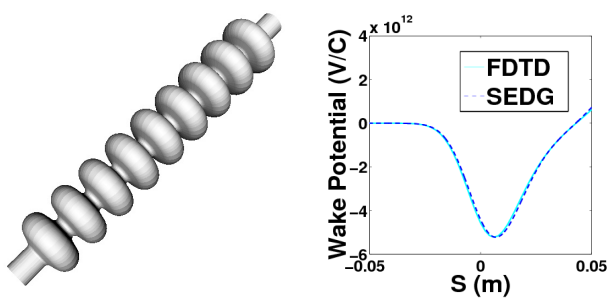

Figure 1: Wakepotential for a 9-cell TESLA cavity.

* mmin@mcs.anl.gov

case [9]. Figure 1 shows a good agreement between those methods for a 9-cell TESLA cavity in [6].

SOFTWARE DEVELOPMENT

NekCEM is open source and available at [3] for easy access to the current version. The package has a number of examples for testing the convergence and performance of the code for different problem configurations in electromagnetics. Current capabilities include wakefield and wakepotential calculations, waveguides, and electric potential calculations. NekCEM is written in Fortran and C. The code uses the core infrastructure of the incompressible Navier-Stokes solver Nek5000 [4], awarded the Gordon Bell prize in 1999. NekCEM uses the distributed-memory message-passing interface (MPI) programming model and the single-program, multidata (SPMD) model so that each processor independently executes a copy of the same program on distinct subsets of data.

NekCEM has an instruction for the following three tasks that are performed consecutively at run time: $presetup$, $solver$, and $checkpointing$. A general description on the code is summarized as follows. *Presetup* includes initialization of processors, setting compile-time data sizes, reading run-time parameters and global mesh data from input files, distributing mesh data to each processor, and assigning numbering for nodal points and coordinates for a geometry. *Solver* involves the SEDG scheme and time iterations. Depending on the applications, a Poisson solve to obtain the initial field, some auxiliary differential quations, or postprocessing routines are additional. *Checkpointing* generates output files for the global field data computed from the solver; these files can be used for restarting. Several options are currently available with five different parallel I/O approaches [5].

NekCEM has two input data files providing the information on global mesh (*.rea) data and global mapping (*.map) for vertices including processor distribution for each element. For simplicity, data files are kept in global format so that users are not required to deal with mesh partition before compile/runs with easier management for many different mesh configurations. Input files are auto-generated from meshing tools such as prex and genmap [4], which are included in the NekCEM package. Data files are read at run time, so it is important to save CPU cycles during this procedure before the actual solver runs. Reading the global data for a mesh takes from 7.5 seconds for 557 million grid points to 28 seconds for 2.2 billion grid points, with E=136K and 546K elements on P=32,768 and 131,072 cores of BG/P, respectively. Similarily, reading the global data for vertex mapping takes 2.6 seconds for 557 million grid points and 11 seconds 2.2 billion grid points with E=136K and 546K elements on P=32,768 and 131,072 cores on BG/P, respectively.

For the communication kernel for exchanging the values at the interfaces between neighboring elements, we use the general-purpose utility gather-scatter kernel [10], `gs()`, following the methodology by Fox et. al. in 1988, which enables significant improvements in scalability. The `gs()` kernel is for parallel matrix-vector products with a particularly lightweight interface. In a setup phase, gs_handle is invoked by `call gs_setup (gs_handle,glo_num,`n^f`,nekcomm,`P`)`, where `glo_num` is a local list of n^f integers containing a global identifier for each entry. For repeated entries in `glo_num` (locally or on other processors), `call gs_op_fields (gs_handle,`u`,`n^f`,`n^f`_stride,1,1,0)` returns the sum of the corresponding elements of u. With this strategy, NekCEM requires $\approx$ 0.5 seconds for `gs_setup` and $\approx$ 0.02 seconds for `gs_op_fields` for n^f=236 million grid points on the faces with actual total grid points of n=474 millions with E=274K elements on 65,536 cores on BG/P.

Parallel I/O is a critical component for proactively dealing with unexpected failures as well as for visualizing simulation tasks. NekCEM initially used the traditional 1 file POSIX I/O per processor (1PFPP) approach in which every processor has to write its own file output. With that approach, performance becomes severely limited when shared storage is accessed by hundreds of thousands processors simultaneously, generating huge overheads from excessive metadata traffic, and disk block locking. Efficient parallel I/O algorithms are necessary. Several I/O approaches has been developed for NekCEM [5], that utilize the MPI-IO libraries, demonstrating the I/O performance of NekCEM in weak scaling for write bandwidth. A 100$\times$ improvement in CPU time over the traditional 1PFPP approach for output file sizes up to 156 GB on up to 65,536 cores on the Argonne BG/P is shown in [5]. NekCEM uses a VTK legacy format for output files that can be directly read by postprocessing tools for visualization using ParaView or VisIt.

PERFORMANCE

We demonstrate parallel performance and scalability of NekCEM for the key algorithms in NekCEM using the number of elements E=136K and E=273K with the polynomial approximation orders N=5 and N=15, resulting in a total of 29 million to 1.1 billion grid points. In Figures 2-3, we observe that the efficiency and the speedup increase as the number of grid points per core increases. Specically, NekCEM achieves 75% epsciency on 131,072 cores for n/P=8,530 (still not a large amount) grid points per core, compared to a base case on 16,384 cores for n/P=68,250 grid points per core. Even with an unrealistically small number of grid points per core n/P=224, NekCEM achieves 58% epsciency on 131,072 cores, compared to a base case on 16,384 cores for n/P=1,793 grid points per core. CPU time per time step is d 0.13 seconds on 131K cores for the case of E=272K with 1.1 billion grid points. In Table 1, the detailed efficiencies for the case of E=273K with N =15 are demonstrated.

Figure 3 demonstrates performance in TFLOPS, showing approximately 10% of peak performance on BG/P.

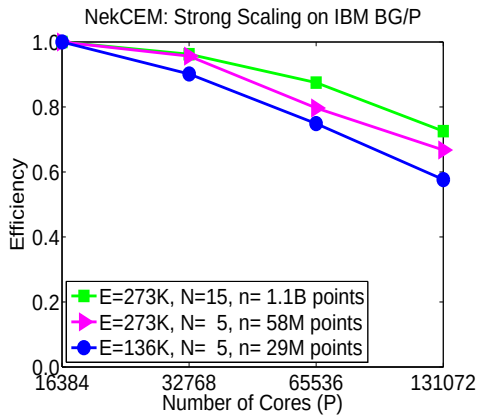

Figure 2: Efficiency on different problem sizes.

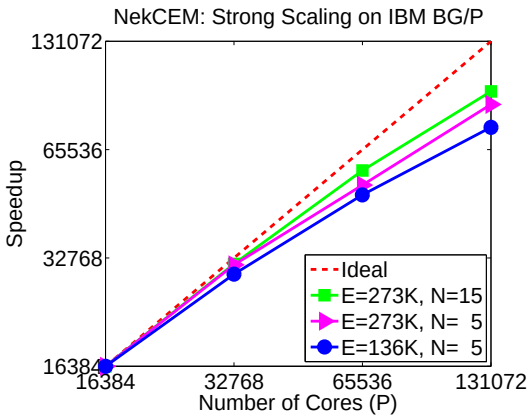

Figure 3: Speedup on different problem sizes.

Our future work includes further enhancement of NekCEM performance with double-hammer intensive coding. Currently, NekCEM uses an MPI programming interface for parallel algorithms. We will also expand this to a hybrid (MPI/shared-memory) programming approach. These and other advanced approaches will dramatically enhance NekCEM's capabilities for petascale and exascale simulations.

Table 1: Parallel efficiency with E=273,000, N=15, the total number of grid points n=$E(N+1)^3$ after running 100 timesteps for the different numbers of cores with the number of grid points per core n/P. Eff denotes the efficiency.

Proc (P)	n/P	CPU (sec)	Ideal(sec)	Eff
16,384	68,250	1.9130	1.9130	–
32,768	34,125	1.0610	0.9565	0.96
65,536	17,063	0.6388	0.4783	0.87
131,072	8,531	0.4146	0.2391	0.72

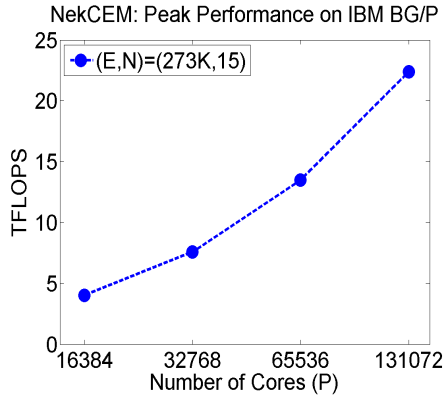

Figure 4: Performance in TFLOPS.

ACKNOWLEDGMENT

This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357. The computational resources are partially from the INCITE allocations from the Nek5000 project led by Paul Fischer and partially from the Director's Discretionary allocations on the Argonne Blue Gene/P Intrepid for NekCEM project.

REFERENCES

[1] M.O. Deville, P.F. Fischer, and E.H. Mund, "High Order Methods for Incompressible Fluid Flow," Cambridge University Press (2002).

[2] J.S. Hesthaven, S. Gottlieb, and D. Gottlieb, "Spectral methods for time-dependent problems," Cambridge University Press (2007).

[3] M.S. Min and P.F. Fischer, "NekCEM," Mathematics and Computer Science Division, Argonne National Laboratory. https://svn.mcs.anl.gov/repos/NEKCEM.

[4] https://nek5000.mcs.anl.gov/

[5] J. Fu, M.S. Min, R. Latham, and C. Carothers, "Parallel I/O performance for application-level checkpointing on the Blue Gene/P system," Preprint ANL/MCS-P1804-1010, 2011.

[6] M.S. Min, P.F. Fischer, Y.C. Chae, "Wake fields for TESLA cavity structures: Spectral element discontinuous Galerkin simulations," Proc. of SRF07, TUP34, (2007).

[7] M.S. Min, P.F. Fischer, "Spectral-element discontinuous Galerkin simulations with a moving window algorithm for wakefield calculations", Proc. of PAC09, TH5PFP03, (2009).

[8] J.S. Hesthaven and T. Warburton,"Nodal High Order Methods on Unstructured Grids - I. Time-Domain Solution of Maxwell's Equations," J. Comp. Physics, 101 (2002), p. 186.

[9] A. Taflov and S.C. Hagness, "Computational Electromagnetics: The Finite-Difference Time Domain Method," Artech House, 3rd ed. (2005).

[10] P. Fischer, J. Lottes, D. Pointer and A. Siegel, "Petascale algorithms for reactor hydrodynamics," Proc. of SciDAC, (2008).