Hybrid Programming and Performance for Beam Propagation Modeling

Misun Min,*‡ Jing Fu,‡ Azamat Mametjanov‡
‡Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, USA

Abstract

We examined hybrid parallel infrastructures in order to ensure performance and scalability for beam propagation modeling as we move toward extreme-scale systems. Using an MPI programming interface for parallel algorithms, we expanded the capability of our existing electromagnetic solver to a hybrid (MPI/shared-memory) model that can potentially use the computer resources on future-generation computing architecture more efficiently. As a preliminary step, we discuss a hybrid MPI/OpenMP model and demonstrate performance and analysis on the leadership-class computing systems such as the IBM BG/P, BG/Q, and Cray XK6. Our hybrid MPI/OpenMP model achieves speedup when the computation amounts are large enough to compensate the OMP threading overhead.

INTRODUCTION

Multicore architectures are current trends for gaining improvement in computing power, instead of increased clock speed. To achieve scalable performance with minimal parallelization overhead on such platforms, we have explored incorporating multithreading frameworks into our existing MPI-only code NekCEM for beam propagation modeling. Specifically, we are using hierarchical parallelization frameworks based on MPI/OpenMP schemes for intranode operations of MPI programs using OpenMP directives around time-consuming loops that do not contain data dependencies, while leaving the source code unchanged.

NekCEM [1, 2, 3] is a freely available, massively parallel, scalable high-order code for electromagnetic device simulations. NekCEM has great potential for meeting the future computational needs of experimental and theoretical research at exascale, by using a fast communication kernel for efficiency and body-fitted hexahedral meshes that allow significant gains in accuracy. We previously conducted wakefield calculations using the spectral-element discontinuous Galerkin (SEDG) scheme [4, 5] with fourth-order Runge-Kutta time stepping, with favorable results in comparison with those from low-order methods.

Future-generation supercomputing systems will be memory-limited relative to the raw computational performance. Currently, per processor memory requirements for NekCEM scale roughly as 600 8-byte words per allocated gridpoint. The total memory requirements are $n = EN^3$ points x 600 (words/point) x 8 (bytes/word). For example, with E=800K and N=16, the total memory requirements are $800K \times 16^3 \times 600 \times 8$. Assuming that 200 MB of memory per core are available to the user, one can run this simulation with P>786,432 cores by setting the number of local elements more than 10. In the current parallel context, however, there will be an increasing memory penalty associated with two variables, the maximum number of cores and the upper bound on the total number of elements, as the problem size becomes very large at extreme scale. A hybrid MPI/share-memory framework can reduce the memory dependency on these two parameters.

* mmin@mcs.anl.gov

IMPLEMENTATION

Ultrarelativistic beam propagations are governed by Maxwell's equations,

$$\mathbf{Q}\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{q}) = \mathbf{S}, \tag{1}$$

where we define the field vector $\mathbf{q} = [\mathbf{H}, \mathbf{E}]^T$ and the flux $\mathbf{F}(\mathbf{q}) = [\mathbf{F_H}, \mathbf{F_E}]^T$ with $\mathbf{F_H} = \mathbf{e_i} \times \mathbf{E}$ and $\mathbf{F_E} = -\mathbf{e_i} \times \mathbf{H}$, and the source term $\mathbf{S} = [\mathbf{0}, \mathbf{J}]^T$. The electric, magnetic, and current fields are represented by $\mathbf{E} = (E_x, E_y, E_z)^T$, $\mathbf{H} = (H_x, H_y, H_z)^T$, and $\mathbf{J} = (0, 0, J_z)^T$, respectively. The material properties are defined as $\mathbf{Q} = \text{diag}(\mu, \mu, \mu, \epsilon, \epsilon, \epsilon)$ with the free space permittivity ϵ and free space permeability μ. Initial fields in the presence of the Gaussian beam are obtained by solving the Poisson equation in transverse direction at the beam location in the longitudinal direction.

Numerical Approach

We consider the computational domain Ω with nonoverlapping hexahedral elements Ω^e such that $\Omega = \cup_{e=1}^{E} \Omega^e$, and we define a weak formulation, introducing the numerical flux $\mathbf{F}^*$ as in [4, 5]:

$$\left(\mathbf{Q}\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{q}) - \mathbf{S}, \phi\right)_{\Omega^e} = (\hat{n} \cdot [\mathbf{F} - \mathbf{F}^*], \phi)_{\partial\Omega^e}. \tag{2}$$

The local solutions of the fields can be written as

$$q^N(\mathbf{x}, t) = \sum_{i,j,k=1}^{N} q_{ijk}\psi_{ijk}(\mathbf{x}), \tag{3}$$

where q_{ijk} is the solution at $\mathbf{x}$=(x_i, y_j, z_k) on Ω^e and ψ_{ijk}=$l_i(r)l_j(s)l)l_k(t)$ using the one-dimensional Legendre Lagrange interpolation polynomial l_i of degree N-1 associated with the N Gauss-Lobatto-Legendre (GLL) quadra-

ture nodes [6]. Plugging (3) into (2) with a local discontinuous test function $\phi = \psi_{ijk}$ and applying the GLL quadrature for the spatial integration, we obtain a semi-discrete scheme with the mass and stiffness matrices defined as

$$\mathbf{M} = (\psi_{ijk}, \psi_{\hat{i}\hat{j}\hat{k}})_{\Omega^e}, \mathbf{D}_x = \left(\frac{\partial \psi_{ijk}}{\partial x}, \psi_{\hat{i}\hat{j}\hat{k}}\right)_{\Omega^e}, \quad (4)$$

$$\mathbf{D}_y = \left(\frac{\partial \psi_{ijk}}{\partial y}, \psi_{\hat{i}\hat{j}\hat{k}}\right)_{\Omega^e}, \mathbf{D}_z = \left(\frac{\partial \psi_{ijk}}{\partial z}, \psi_{\hat{i}\hat{j}\hat{k}}\right)_{\Omega^e} \quad (5)$$

and the surface integration on the right-hand side in (2) (we omit the detailed form in this paper).

Algorithms

The algorithm can be simplified as

$$U^{n+1} = U^n + \texttt{mxm}(U^n) + \texttt{comm}(F(U^n)), \quad (6)$$

where `mxm` and `comm` respectively represent the matrix-matrix product and communication operation for the surface integration, $F(U^n)$. The main operation count in our algorithm is dominated by the `mxm` routine: the curl operator takes 30% of total cost and 15% of total cost for communication. We focus on speedup by additional threading on the `mxm` routines.

Within each subdomain, derivatives are evaluated in a tensor product fashion by using a one-dimensional differentiation matrix on a reference domain $[-1, 1]^3$. A derivative of u^e ($e = 1, ..., E$) in a subdomain with respect to $(r, s, t) \in [-1, 1]^3$ is expressed by

$$\mathrm{u}_r^e \equiv D_r u^e = (I_t \otimes I_s \otimes \hat{D}) u^e = \Sigma_{l=1}^N \hat{D}_{il} u_{ljk}^e, \quad (7)$$

$$\mathrm{u}_s^e \equiv D_s u^e = (I_t \otimes \hat{D} \otimes I_r) u^e = \Sigma_{l=1}^N \hat{D}_{jl} u_{ilk}^e, \quad (8)$$

$$\mathrm{u}_t^e \equiv D_t u^e = (\hat{D} \otimes I_s \otimes I_r) u^e = \Sigma_{l=1}^N \hat{D}_{kl} u_{ijl}^e, \quad (9)$$

where I_t, I_s, and I_r are the $N \times N$ identity matrices and $\hat{D}$ is the one-dimensional differentiation matrix of $N \times N$ defined in [6].

D_x, D_y and D_z in Eq. (4) require three `mxm` operations for each. Thus the cost per timestep involving the `mxm` operations for a curl operator scales as $O(18EN^4)$ for six field components. For the MPI-only model, we use the `mxm` routine, written in Fortran and assembly code, which includes the inner-product dimension completely unrolled into a single statement, allowing a short-nested loop for more work per iteration and a hardcoded address increments into memory read instructions by the compiler [6]. We consider the OMP routine as an alternative for possible speedup.

OpenMP is a set of APIs for writing multithreaded programs on shared-memory machines. It can help the compiler parallize applications at the highest possible level through explicit compiler directives yet not involve application programmers in low-level details.

For our hybrid MPI/OpenMP design, we use the following instruction for the `mxm` routine:

```
c$OMP PARALLEL DEFAULT(PRIVATE)
      SHARED(A,B,C,N1,N2,N3)
c$OMP DO
      do j=1,N3
      do i=1,N1
         c(i,j) = 0
      do k=1,N2
         c(i,j) = c(i,j) + a(i,k)*b(k,j)
      enddo
      enddo
      enddo
c$OMP END DO
c$OMP END PARALLEL
```

PERFORMANCE

We performed our tests on the IBM BG/P and BG/Q and the Cray XK6 for different problem sizes with varying N on a hexahedral mesh with E=1152 for an undulator. The features of the systems are described below.

IBM BG/P The Blue Gene/P Intrepid consists of 40,960 compute nodes (40 racks and 1,024 nodes per rack, including 640 I/O nodes) with 850 MHz quad-core processor and 2 GB RAM per node, for a total of 163,840 cores, 80 TB of RAM, and a theoretical peak performance of 557 teraflops.

IBM BG/Q The Blue Gene/Q Mira consists of 49,152 compute nodes (48 racks and 1,024 nodes per rack, including 384 I/O nodes) with 1.6 GHz 16-core processor and 16 GB RAM per node, for a total of 786,432 cores, 786 TB of RAM, and a theoretical peak performance of 10 petaflops.

Cray XK6 The Cray XK6 Jaguar consists of 18,688 compute nodes. Each compute node consists of 16-core 2.2 GHz AMD Opteron processors and 32 GB of RAM, for a total of 299,008 cores, 598 TB of RAM, and a theoretical peak performance 2.63 petaflops.

We measured the CPU time and wallclock time for 100 timestep runs. For the CPU time measure, we used `dclock` and got an average time over the total number of MPI ranks. For the wallclock time, we used `mpi_wtime` and `omp_get_wtime` for MPI ranks and OMP threads, respectively. Here we demonstrate the CPU time using the average value over all MPI ranks.

Figure 1 shows the CPU time for a fixed number of MPI ranks (=1024) with an increasing number of OMP threads for the granularity per core n/P=576 and 4608. In the top figure, adding additional threads (i.e., more computing resources) decreases the performance because the overhead of creating threads offsets the benefit of parallelizing the loops for such coarse granularity in computation. As the work amount increases in the bottom figure, we observe more OMP threads for speedup. Compared with the MPI-only case, however, we do not gain much speedup.

Figure 2 demonstrates the CPU time for a fixed number of total threads (=1024) with varying numbers of MPI

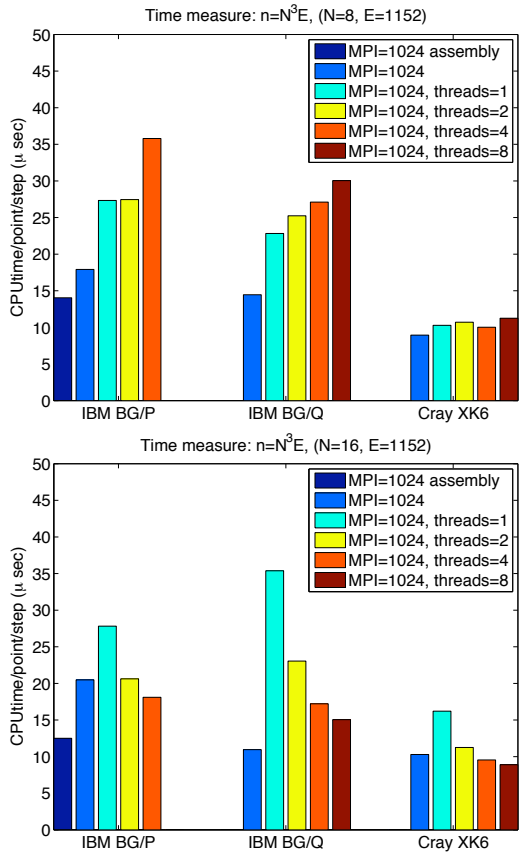

Figure 1: Scaling on MPI/multithreading algorithms on BG/P, BG/Q, and Cray XK6.

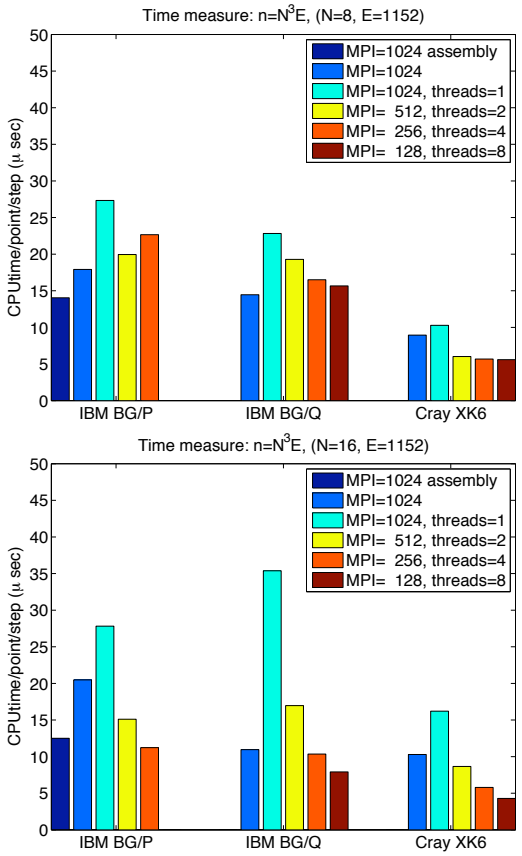

Figure 2: Scaling on MPI/multithreading algorithms on BG/P, BG/Q, and Cray XK6.

ranks and OMP threads. The granularity per core changes as n/P=576, 1152, 2304, 4608 with P =1024, 512, 256, 128, respectively, for n=0.58M (top) and n/P=4608, 9216, 18432, 36864 with P =1024, 512, 256, 128, respectively, for n=4.7M (bottom). By increasing the number of OMP threads but reducing communication with a smaller number of MPI ranks, the results show increased speedup consistently. However, superior performance compared with the MPI-only case is possible when there is enough work for the OMP loops, as shown in the bottom figure.

CONCLUSION

We have conducted performance studies on leadership-class computing systems and have demonstrated the speedup for a hybrid MPI/OpenMP model in comparison with the MPI_only case when the computation amounts are large enough to hide the OMP threading overhead. Future work includes expansion to MPI/GPU threading and comparison with further optimized MPI/OpenMP model at larger scale.

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.

REFERENCES

[1] "NekCEM," http://wiki.mcs.anl.gov/nekcem.

[2] M. S. Min, J. Fu, P. F. Fischer, "Performance analysis of the spectral-element discontinuous Galerkin method for electromagnetic modeling on the IBM BG/P and Cray XK6," Preprint ANL/MCS-P1802-0111.

[3] J. Fu, M. S. Min, R. Latham, C. D. Carothers, "I/O threads to reduce checkpoint blocking for an electromagnetics solver on Blue Gene/P and Cray XK6," in International Workshop on Runtime and Operating Systems for Supercomputers (ROSS), in conjunction with International Conference on Supercomputing (ICS), June 2012.

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

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

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