FERMILAB-CONF-16-271-AD-CD

FEL SIMULATIONS USING DISTRIBUTED COMPUTING

J. Einstein[1], S.G. Biedron[†2], H.P. Freund, S.V. Milton, P.J.M. van der Slot[3],Colorado State University, Fort Collins, Colorado, USA
G. Bernabeu, Fermi National Accelerator Laboratory, Batavia, Illinois, USA
[1] also at Fermi National Accelerator Laboratory, Batavia, Illinois, USA
[2] also at Faculty of Electrical Engineering, University, Ljubljana, Slovenia
[3] also at Laser Physics and Nonlinear Optics, Mesa and Institute for Nanotechnology, University of Twente, Enschede, the Netherlands

Abstract

While simulation tools are available and have been used regularly for simulating light sources, including Free-Electron Lasers, the increasing availability and lower cost of accelerated computing opens up new opportunities. This paper highlights a method of how accelerating and parallelizing code processing through the implementation and use of COTS software interfaces can be of great benefit to our community.

INTRODUCTION

Massive parallelization of algorithms is a significant challenge that needs to be overcome to enable real-time optimization and simulation of any system. Free Electron Laser (FEL) simulation is particularly computationally intensive, but has the benefit of being able to be processed in discrete units. Present FEL simulation tools rely on OpenMP or MPI (Open Message Processing or Message Processing Interface) for inter-processing unit (PU) communication, which is currently limited to discrete cores as the smallest PU.

Several tools have recently become available that allow for code development for parallel processing in many different hardware environments [1,2,3]. Combining the benefits of OpenMPI with the speed-ups available from OpenCL (Open Computing Language) or CUDA® code allows for significant time saving in simulation.

The OpenCL is a program-writing framework that executes across heterogeneous platforms consisting of central processing units (CPUs), graphics processing units (GPUs), digital signal processors (DSPs), field-programmable gate arrays (FPGAs) and other processors or hardware accelerators [4].

CUDA® is a parallel computing platform and application programming interface (API) model created by NVIDIA. It allows software developers to use a CUDA®-enabled graphics processing unit (GPU) for general purpose processing – an approach known as GPGPU [5].

Arrayfire is an open source toolkit that allows for abstracting interfaces to both OpenCL and CUDA without significant changes to the application code [3].

Here we identify and describe one of several FEL codes that we can further modify and also describe the tests that we envision, including those with Arrayfire.

MEDUSA/OPC

MEDUSA is a three-dimensional simulation code that includes time-dependence, harmonics, and start-up from noise [6,7]. It models helical and planar wigglers and the optical field is represented as a superposition of Gaussian modes. Electron trajectories are integrated using the three-dimensional Lorentz force equations in the combined magnetostatic and optical fields. No wiggler average orbit analysis is used. Models for quadrupoles and dipoles are included. The time-dependence is treated in either of two ways. First, the electron beam and the optical mode are described by an ensemble of temporal slices where each slice is advanced from $z \rightarrow z + \Delta z$ as in steady-state simulations, after which the field is allowed to slip relative to the electrons. Second, an explicit polychromatic expansion of the fields can be employed. These two algorithms are equivalent; however, the former is simpler to employ and is used here. Note that the first time-dependence algorithm can be combined with a polychromatic harmonic representation to treat the evolution of the fundamental and harmonics in the time-domain.

MEDUSA has been coupled with the Optics Propagation Code (OPC) in order to simulate complete FEL-based oscillators. OPC propagates the optical field using either the Fresnel diffraction integral or the spectral method in the paraxial approximation [8,9] using fast discrete Fourier transforms (FFT). A modified Fresnel diffraction integral is also available and allows the use of FFTs in combination with an expanding grid on which the optical field is defined. This method is often used when diffraction of the optical beam is large. Currently, OPC includes mirrors, lenses, phase masks, and round and rectangular diaphragms. Several optical elements can be combined to form more complex optical components, *e.g.*, by combining a mirror with a hole element, extraction of radiation from a resonator through a hole in one of the mirrors can be modeled. Phase masks can be used to model mirror distortions or to create non-standard optical components like a cylindrical lens.

In a typical resonator configuration, OPC handles the propagation from the end of the gain medium to the first optical element, applies the action of the optical element to the optical field and propagates it to the next optical element until it reaches the entrance of the gain medium.

† biedron@engr.colostate.edu

Copyright © 2016 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-147-2

Operated by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the United States Department of Energy

Diagnostics can be performed at the planes where the optical field is evaluated. Some optical elements, specifically diaphragms and mirrors allow forking of the optical path. For example, the reflected beam of a partial transmitting output mirror forms the main intracavity optical path, while the transmitted beam is extracted from the resonator. When the intracavity propagation reached the output mirror, this optical propagation can be temporarily suspended, and the extracted beam can be propagated to a diagnostic point for evaluation. Then the intracavity propagation (main path) is resumed.

Typically, the simulation of an FEL oscillator starts with the FEL gain model, in this case with MEDUSA, that will initialize the optical field and propagate it together with the electron beam through the wiggler. Then at the position of the wiggler exit, the optical field, *i.e.*, the complex phasor of the field, is handed over to OPC to propagate it towards the downstream mirror. The portion of the optical mode that is reflected is then propagated to the upstream mirror (which is a high reflector) by OPC, and then back to the wiggler entrance where the optical field is handed back to the gain model. The complex phasor is reduced to an ensemble of Gaussian modes that are used as input for the next pass. Here we explore two examples of the code's present performance in comparison to experimental data.

LCLS FEL EXAMPLE

The Linac Coherent Light Source (LCLS) [10] is a SASE FEL user facility that became operational in 2009 and operates at a 1.5-Å wavelength. It employs a 13.64-GeV/250-pC electron beam with a flat-top temporal pulse shape of 83-fsec duration. The normalized emittance (x and y) is 0.4 mm-mrad and the rms energy spread is 0.01%. The undulator line consisted of 33 segments with a period of 3.0 cm and a length of 113 periods including one period each in entry and exit tapers. A mild down-taper in field amplitude of -0.0016 kG/segment starting with the first segment (which has an amplitude of 12.4947 kG and K_{rms} = 2.4748) and continuing from segment to segment was used to compensate for energy loss due to Incoherent Synchrotron Radiation (ISR). The electron beam was matched into a FODO lattice consisting of 32 quadrupoles each having a field gradient of 4.054 kG/cm and a length of 7.4 cm. Each quadrupole was placed a distance of 3.96 cm downstream from the end of the preceding undulator segment.

The LCLS produces pulses of about 1.89 mJ at the end of the undulator line, and saturation is found after about 60 m. A comparison between the measured pulse energies (green circles) and the simulation (blue) is shown in Fig. 1. The data is courtesy of P. Emma and H.-D. Nuhn at SLAC, and the simulation results represent an average over an ensemble of runs performed with different noise seeds. As shown in the figure, the simulations are in good agreement with the measurements and in the start-up and exponential growth regions. The simulation exhibits saturation where the exponential gain regime ends after 60 m with a pulse energy of 1.5 mJ. However, the pulse energy grows more slowly to about 1.92 mJ after 110 m, which is in good agreement with the measurements.

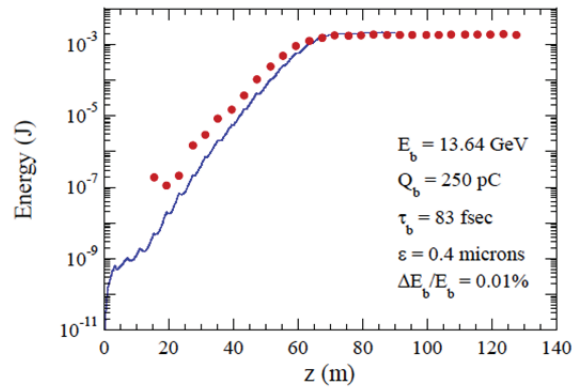

Figure 1: Comparison of the simulated pulse energy versus distance with the data from the LCLS experiment.

Simulation with MEDUSA/OPC has also been compared with the experimental results of the IR-Upgrade experiment at Jefferson Laboratory [11,12]. The experimental configuration is described in detail in ref. 7. The specific parameters to be used for the comparison are as follows. The electron beam has a kinetic energy of 115 MeV, a bunch charge of up to 115 pC, an energy spread of 0.3%, a pulse length of 390 fsec, and a pulse repetition frequency of 74.85 MHz. The normalized emittance is 9 mm-mrad in the wiggle-plane and 7 mm-mrad in the plane normal to the wiggle-plane. The beam is matched to the optical mode, so that it is focused to a spot near the center of the wiggler. The planar wiggler is 30 periods in length, has a peak on-axis amplitude of 3.75 kG, and a period of 5.5 cm. The FEL is tuned to a wavelength of 1.6 microns and the resonator is approximately 32 m long. For the experiments under consideration here, the Rayleigh range is 0.75 m. The downstream mirror is partially transmissive, and out-couples about 21% of the energy per pass.

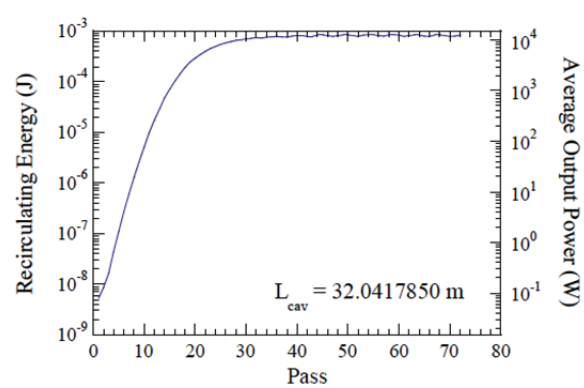

Figure 2: Simulation of the recirculating energy per pass through the resonator in The JLab IR-Upgrade.

Copyright © 2016 CC-BY-3.0 and by the respective authors

The optimal cavity length is found in simulation to be 32.0417085 m, and we plot the evolution of the circulating energy in each pulse and the average output power versus pass in Fig. 2. An equilibrium state is achieved after about 40 passes with a circulating pulse energy of about 0.8 mJ. Since 21% of the pulse energy is outcoupled per pass at a repetition rate of 74.85 MHz, this translates into an average output power of 12.3 kW. An output power of 14.3 ± 0.72 kW was observed in the experiment, so that the simulation result is only approximately 9% lower than the experimental observation.

MPI PARALLELIZATION RESULTS

At the present time, MEDUSA has been efficiently parallelized using MPI. A measure of the effectiveness of the MPI parallelization is shown in Fig. 3 where we plot the ratio of the run time using 1 CPU to that using *N* CPUs on the cluster GANGLIA at the Naval Postgraduate School that has 24 nodes with 4 CPUs per node. MEDUSA is parallelized for time-dependent simulations by parsing different temporal slices across a series of CPUs on multiple nodes. The example shown in Fig. 3 used 225 slices over a range of from 1 to 48 CPUs. For example, the best achievable parallelization would correspond to a reduction in the run time by a factor of *N* when the number of CPUs increases by the same amount. This would yield a slope of unity in Fig. 3. Of course, this is never achievable due to various overhead/communication issues between the nodes/CPUs. However, the curve for MEDUSA exhibits a slope of 0.77 that is quite good.

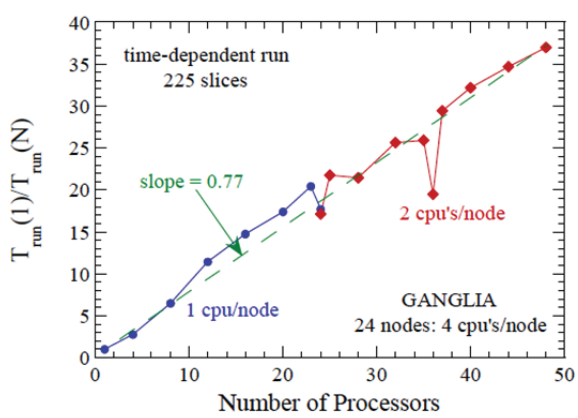

Figure 3: Measure of present, MPI-based parallelization efficiency with MEDUSA.

GOAL – DISTRIBUTED COMPUTING

Although MPI parallelization can provide a significant reduction in run time, this may not be sufficient to compensate the computational requirements for simulating x-ray FELs. For example, in the case of the LCLS, the electron pulse is about 170,000 radiation wavelengths long and the interaction length is over 100 m. This necessitates either extremely large super-clusters or extremely long run times, or both. As a result, the advent of very fast, and relatively inexpensive, GPUs and Many Integrated Core (MIC) architectures hold the potential for achieving dramatic reductions in run times for these simulations possible. In particular, these GPUs can be installed on large clusters so that they can be used in conjunction with MPI. Also, capable GPUs can now be found on many high-end laptops, as well. MIC architectures are present in many modern supercomputers like DoE's CORI (phase II) [13].

As such, the goal of this effort is to make MEDUSA compatible with both MPI- and GPU-based processing, which we expect will result in dramatic improvements in the modelling of x-ray FELs. This toolbox can also be extended for use with other light sources, in future work.

STEPS FOR DISTRIBUTED SIMULATION

Distributed computation requires the use of high-bandwidth nodes, each with on-board processing capabilities. Earlier sections of this paper discussed the scalability and resource usage of an MPI-based simulation code. GPU-based computation provides each local node with additional computation capabilities.

The first step in optimizing a distributed FEL simulation system is determining which computations are best optimized locally, i.e. single array operations or shared memory, and what is best distributed on separate nodes, i.e. independent processes.

In FEL simulation, each time slice can generally be treated as an independent process, with calculations of interaction and field best calculated locally. These can be optimized using BLAS (Basic Linear Algebra Subprograms) libraries [14] and identifying the array interactions.

CONCLUSION

Distributed simulation techniques provide significant benefits for design and operations of light sources. In particular, being able to verify experimental systems and identify problems early require faster simulation processes.

We have presented here a path forward for moving towards significantly faster simulation times in FEL codes. Modern software tools and packages provide opportunities that need to be taken to bring FEL design and operations in to a real-time space.

ACKNOWLEDGMENT

Fermilab is operated by the Fermi Research Alliance, LLC under Contract No. De-AC02-07CH11359 with the U.S. Department of Energy.

REFERENCES

[1] Matlab Parallel Computing toolbox http://www.mathworks.com/products/parallel-computing/

[2] National Instruments, Multicore Programming with LabVIEW Technical Resource Guide.

Copyright © 2016 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-147-2

ftp://ftp.ni.com/evaluation/labview/ekit/multicore_programming_resource_guide.pdf

[3] http://ww.arrayfire.org

[4] OpenCL - https://www.khronos.org/opencl/

[5] http://www.nvidia.com/object/cuda_home_new.html

[6] H.P. Freund, Phys. Rev. E **52**, 5401 (1995).

[7] H.P. Freund, S.G. Biedron, and S.V.Milton, IEEE J. Quantum Electron. **27**, 243 (2000).

[8] J.G. Karssenberg *et al*., J. Appl. Phys. **100**, 093106 (2006).

[9] http://lpno.tnw.utwente.nl/opc.html

[10] P. Emma *et al*., Nat. Photon. **176**, 1038 (2010).

[11] P.J.M. van der Slot, H.P. Freund, W.H. Miner, Jr., S.V.Benson. M. Shinn, and K.-J. Boller, Phys. Rev. Lett., **102**, 244802 (2009).

[12] G.R. Neil *et al*., Nucl. Instr. Meth. A **557**, 9 (2006).

[13] http://www.nersc.gov/users/computational-systems/cori/

[14] http://www.netlib.org/blas/

Copyright © 2016 CC-BY-3.0 and by the respective authors