HIGH-PERFORMANCE BEAM SIMULATOR FOR THE LANSCE LINAC*

X. Pang†, L. Rybarcyk, and S. A. Baily
Los Alamos National Laboratory, NM, 87545, USA

Abstract

A high performance multiparticle tracking simulator is currently under development at Los Alamos. The heart of the simulator is based upon the beam dynamics simulation algorithms of the PARMILA code, but implemented in C++ on Graphics Processing Unit (GPU) hardware using NVIDIA's CUDA platform. Linac operating set points are provided to the simulator via the EPICS control system so that changes of the real time linac parameters are tracked and the simulation results updated automatically. This simulator will provide valuable insight into the beam dynamics along a linac in pseudo real-time, especially where direct measurements of the beam properties do not exist. Details regarding the approach, benefits and performance are presented.

INTRODUCTION

The LANSCE accelerator complex is a multi-beam, multi-user facility that provides high-intensity H+ and H- particle beams for a variety of user programs. At the heart of the facility is a room temperature linac that is comprised of a 100-MeV drift tube linac and an 800-MeV coupled cavity linac. During routine operation of the facility, due to the limited number of diagnostic tools, very limited information can be obtained about the beam dynamics inside of the accelerator. Linac operational set points are adjusted with a goal of maintaining minimal beam loss. However, without the detailed knowledge of the beam distribution and the effect of the adjustments on it, these changes can potentially lead to degradation in beam quality and higher losses downstream. A more desirable situation would be one where knowledge of the beam distribution along the linac is available in pseudo real time to aid in the optimization of the linac operation and beam performance.

For these reasons we are pursuing a high performance beam dynamics simulator that when linked to the accelerator control system will track changes to system parameters and in rapid response provide valuable insight into the beam motion and quality. The core of this simulator is based upon the multi-particle beam dynamics code PARMILA [1], but implemented in C++ using NVIDIA's CUDA [2] technology for GPU. The GPU hardware was chosen for its superior parallel computing performance and cost efficiency. Beam dynamics algorithms are recast to maximize the benefit of GPU architectures and a substantial speedup was obtained. So far, the simulator has demonstrated its capabilities of soliciting real-time operational parameters through the accelerator control system, quickly updating the simulation and displaying user-requested beam quality metrics at any location along the accelerator.

* Work supported U.S. Dept. of Energy, NNSA under contract DE-AC52-06NA25396

† xpang@lanl.gov

CODE STRUCTURE

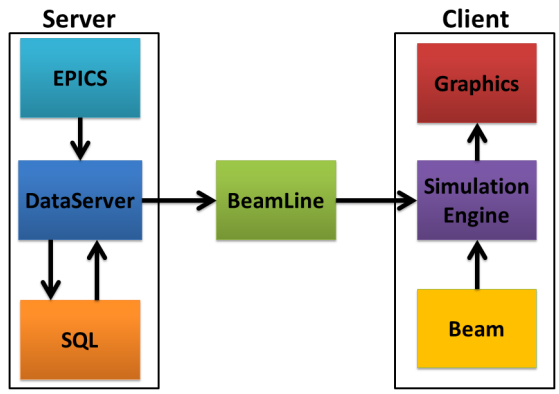

Figure 1: Top level code structure and data flow.

Shown in Fig. 1 is the top level code structure of the simulator. On the server side, a *DataServer* reads in the real-time linac set points from the EPICS control system and pipes the data into a SQL database to convert the set points into model parameters that can directly be used by the *SimulationEngine* on the client side. *BeamLine* serves as a bridge between server and client. It resides in the mapped pinned memory on GPU and CPU to achieve the highest memory bandwidth through PCIe bus and facilitates information sharing. OpenGL graphics rendering was handled by the *Graphics* module on the client side. Fig. 2 is a screen shot of the simulation outputs shown side by side with the EPICS control channels for quadrupole mangets

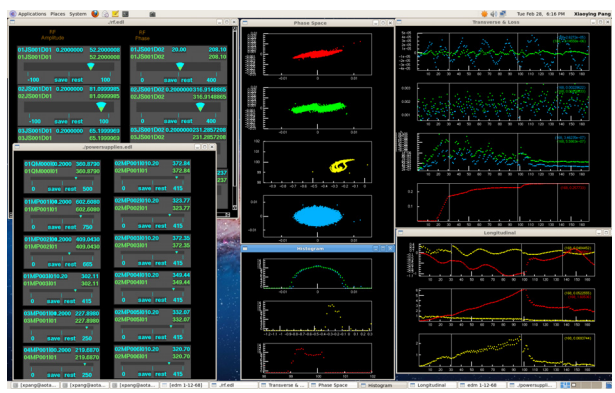

Figure 2: Left: EPICS control sliders. Right: simulation outputs including phase spaces, beam centroids, sizes, emittances, distribution histograms and loss.

ISBN 978-3-95450-115-1

and RF phases and amplitudes. The output quantities include beam centroid, size, emittance, phase space, distribution and loss. The output plots can be updated rapidly in response to any change to the EPICS control channels.

BEAM DYNAMICS ALGORITHM

The beam dynamics algorithms of this simulator are based on the algorithms used in PARMILA, but recast to maximize the benefits of the GPU hardware. Transfer maps were used to transport beam through different linac elements. Along the linac, space charge kicks are applied to the beam at appropriate locations. The space charge routines which were implemented in PARMILA are 2D SCHEFF [3] and 3D PICNIC [4]. In SCHEFF, beam distributions are assumed to be cylindrically symmetric. And the mesh is set up in the radial and longitudinal directions. A SCHEFF routine follows four steps to calculate the space charge kicks for particles: computing the electric field table for the entire mesh based on the relative positions of every two individual cells, putting particles on the mesh and calculating the charge density distribution, calculating the electric field at every node on the mesh, interpolating and applying the electric force on every particle. The PICNIC routine follows a similar procedure, however implemented using a 3D mesh. This routine is currently not included in the simulator.

PERFORMANCE

To compare the performance of the codes running on GPU and CPU, we picked a simple beam line that consists of two quadrupole magnets separated by a RF accelerating gap as a test case. A space charge kick is applied at the center of the RF gap using the 2D SCHEFF space charge routine. The performance tests were conducted on a Super-Micro Workstation running a Scientific Linux 6.1 OS. It is equipped with an Intel Xeon E5520 quad-core CPU, 32GB of RAM, a NVIDIA Quadro NVS 290 GPU, and a GeForce GTX 580 GPU. The GTX580 GPU which has 512 cores, 3GB of global memory, and 48KB of configurable shared memory was used for the performance tests. The programming platform adopted was CUDA 4.0.

Overall

Figure 3 shows the dramatic performance difference between GPU and CPU. Codes on GPU run significantly faster than their counterparties on CPU. From Fig. 4 one can see the substantial speedup gained by the GPU code for both the space charge and the transport routines. The overall speedups of the GPU code range from 20 to almost 100. With more particles being used in the simulation, more speedup can be achieved through the thread level parallelism (TLP). However, the speedups of the space charge routine are less than those of the transport due to the fact that the electric field table calculations which count for a substantial part of the space charge routine do not depend

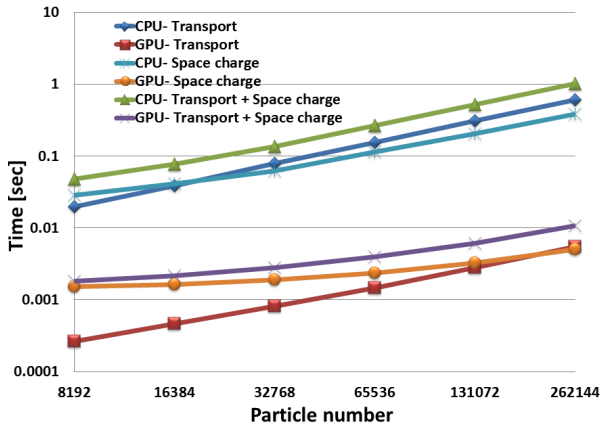

Figure 3: Semi-log plot of the computing times on CPU and GPU.

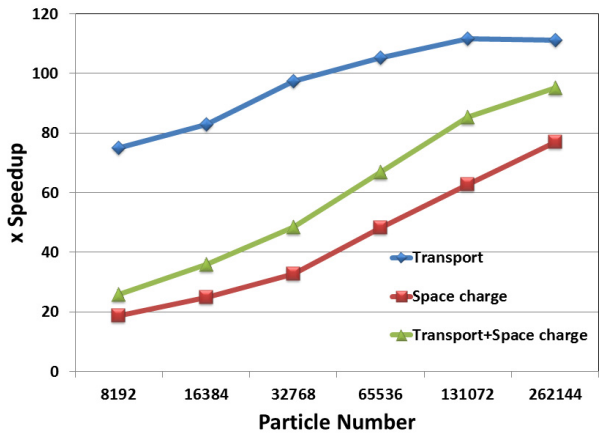

Figure 4: Speedup gained by GPU.

on the number of particles being used for the simulation. This is clearly shown in Fig. 5 where the computation times of four major kernels in the SCHEFF space charge routine are compared. As particle number increases, the particle number dependent kernels of the space charge routine take a larger portion of the whole computing time, therefore boost the speedup of the entire space charge routine as indicated by Fig. 4.

It only takes the simulator on the GPU 0.422 second to push 32K H- particles through the LANSCE 100 MeV DTL which consists of 165 RF gap (space charge kick at every gap), 135 quadrupole magnets and 37 drifts.

Space Charge Routines

SCHEFF The performance of a space charge routine can be critical for the performance of the overall program. The SCHEFF space charge routine claims 38% to 60% of the entire computing time on CPU, and 47% to 83% of the time on GPU. A SCHEFF routine on the GPU consists of four major kernel functions which correspond to four calculation steps of the SCHEFF routine. Depicted in Fig. 5, the kernel that calculates the final electric field table for

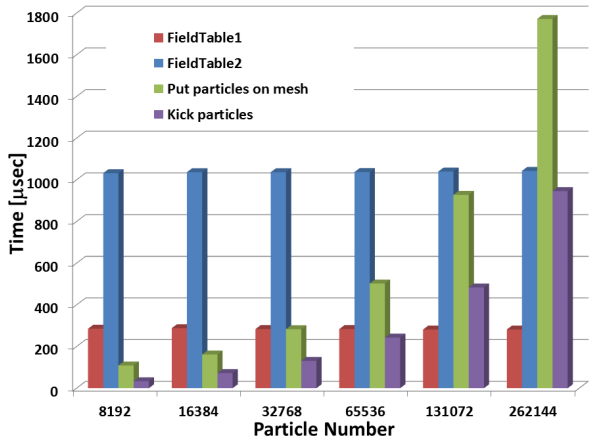

Figure 5: Computing times of SCHEFF kernels.

every node on the mesh (the blue bar) and the one that calculates the charge density (the green bar) can potentially be the bottlenecks of the entire routine. Data prefeching from GPU's global memory into the fast shared memory was employed to help prevent the bandwidth limit in the final electric field table calculation. However, the kernel is instruction limited due to the great amount of double precision instructions issued. Algorithm modification might be necessary in order to further improve the performance. In the charge distribution calculation, the use of atomic addition could potentially hurt its performance especially with condensed beam distributions. As an alternative, data decomposition and block reduction techniques are currently under test to get more consistent performance from this kernel.

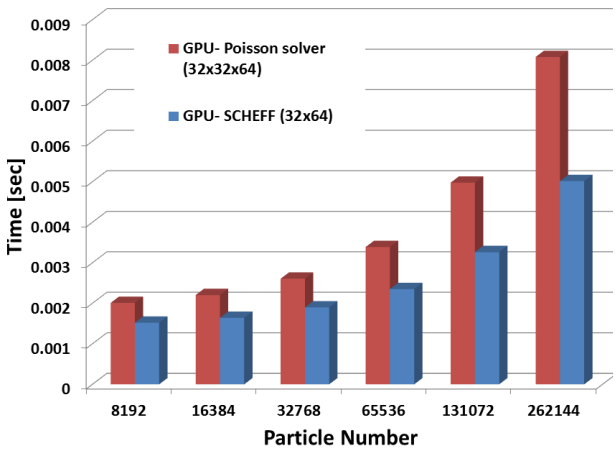

Figure 6: Computing times for 3D Poisson solver and SCHEFF.

Poisson Solver For more general beam distributions, space charge routines based on Poisson solvers lead to more accurate simulation results. Therefore we also implemented a space charge routine using a 3D Poisson solver.

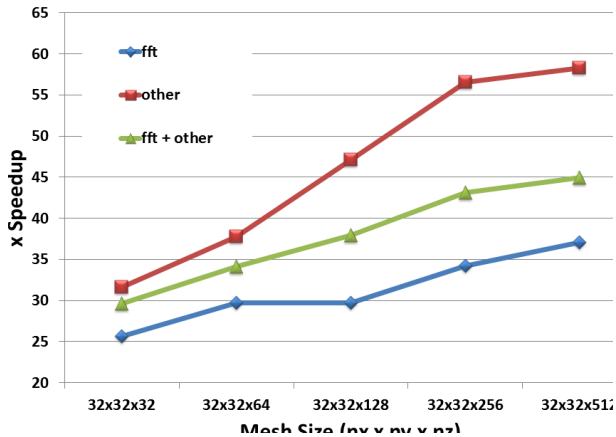

Figure 7: Speedup gained by GPU for 3D Poisson solver.

In Fig. 6, we compare the performance of a SCHEFF routine (mesh size 32×64) with that of a 3D Poisson solver (mesh size $32 \times 32 \times 64$) using GPU for varying particle numbers. The computing time increments were limited by 50% in our test cases. Further investigation is needed to decide the tradeoff between accuracy and computing time. At the heart of the Poisson solvers are the FFT routines. On GPU, the CUFFT library [5] was employed while on CPU, FFTW library [6] was used for the performance tests. Figure 7 compares the speedups achieved by the FFT routine, non-FFT part of the solver, and the entire solver. As the size of the mesh increases, the overall speedup of the solver goes up ranging from 30 to 45. It is eventually limited by the speedup obtained on the FFT routine.

CONCLUSIONS

A high performance multi-particle simulation tool was developed on GPU hardware using NVIDIA's CUDA platform. It can track any real-time operational parameter change in an accelerator, update simulations rapidly and display user-requested beam quality metrics. A substantial speedup was obtained by using the GPU hardware. This simulator will provide valuable insights into the beam dynamics inside an ion linac especially where direct beam measurements are not available.

REFERENCES

[1] H.Tadeda, J. Billen, *PARMILA*, LANL publication, LA-UR-98-4478, 2005

[2] NVIDIA, *NVIDIA CUDA C Programming Guide, Version 4.0*, 2011

[3] P. Lapostolle et al., Nucl. Instrum. Methods Phys. Res., Sect. A 379, 21 (1996)

[4] N. Pichoff et al., Proceedings of the International LINAC Conference, Chicago, IL, 1998, p. 141.

[5] NVIDIA, *CUFFT Library Programming Guide*, 2011

[6] M. Frigo, S.G. Johnson, *FFTW for version 3.3.2*, 2012

ISBN 978-3-95450-115-1