PERFORMANCE OPTIMIZATION OF MULTI-PARTICLE BEAM DYNAMICS CODE IMPACT-Z ON NVIDIA GPGPU

Z. He[*], G. Shen, Y. Yamazaki. FRIB, Michigan State University, USA
X. Wang, ICER, Michigan State University, USA

Abstract

Facility for Rare Isotope Beams is designed using a multi-particle tracking code IMPACT-Z. IMPACT-Z is originally for the purpose of accelerator design, so it is precise, however, quite time consuming, therefore usually not suitable for on-line beam tuning applications. IMPACT-Z is originally boosted using Message Passing Interface (MPI) technology. For single node mode, performance of IMPACT-Z is usually bounded by CPU performance, and for multi-node mode, communication between MPI processes would become bottleneck. However, new emerging High Performance Computing (HPC) technology, like general-purpose graphics processing unit (GPGPU), brings new possibility in accelerating IMPACT-Z, so that the speed of IMPACT-Z satisfies for on-line beam tuning applications. This paper presents the efforts in exploring the capability of Nvidia GPGPU and the results of speed up of IMPACT-Z.

INTRODUCTION

Modern accelerator facility, Facility for Rare Isotope Beams (FRIB) [1] for example, usually has complicated lattice structures, so that it usually needs an online model for beam commissioning. The linac segment of FRIB is designed by a multi-particle tracking code IMPACT-Z [2]. IMPACT-Z is originally written for the purpose of accelerator design, so it is precise, but too time consuming when applying to online beam tuning, even though its speed has already been boosted by MPI-based HPC technology.

A new FRIB-specific envelope tracking based light-weight linear surrogate model is under development [3, 4] to cover most of the challenges in FRIB on-line beam tuning task such as cavity tuning and orbit correction during low power beam commissioning. However, because the limitation of the linear model, its ability to guide FRIB all the way up to full power operation is questionable. As a result, a boosted version of IMPACT-Z for high power beam commissioning is proposed.

IMPACT-Z is using MPI technology to boost its performance on both single-node multi-cores and multi-nodes infrastructure. Previous study has pointed out that new emerging HPC technology, like GPGPU, would possibly add more computing power with reasonable cost [5]. This paper aims at searching for optimized performance boost of IMPACT-Z provided by NVidia GPGPU. First, The performance of IMPACT-Z is baselined with FRIB lattice. Then, the strategy of rewriting IMPACT-Z into a GPU code is described. After that, the performance and precision of IMPACT-Z GPU version is discussed and compared with original version.

[*] hez@frib.msu.edu

BASELINE OF IMPACT-Z ON CPU

In order to compare the performance of IMPACT-Z on CPU and GPU, we first need to set up a baseline case and use it as standard test case for our following study. Linac segment 1 plus folding segment 1 of FRIB is used as the test lattice, and the input particle number is 50K. 20 integration steps for RF cavities and 4 integration steps for the rest elements are used. The GNU mpif90 compiler with optimization level O3 is used to compile the code and the code is run on an Intel Xeon CPU E5-2640 v2 @ 2.00GHz with number of MPI worker set to 1. And the running time of this standard test case is 446.35 s.

DESIGN STRATEGY FOR IMPACT-Z GPU VERSION

After tens of years of development, GPU has become a standard tool in scientific HPC technology. Previous work reported that GPU is high performance, cost effective, and its parallelization model is applicable to accelerator simulation [5]. The possibility of boosting the performance of IMPACT-Z using GPU to benefit FRIB commissioning is worth investigation. The following study is performed on NVidia Tesla K20 GPU card provided by Institute for Cyber-Enabled Research (ICER), MSU. The configuration of NVidia Tesla K20 GPU card can be seen in Table 1 [6].

Table 1: The Configuration of NVidia Tesla K20 GPU Card

Item	Value
Tesla Products	K20
Core clock (GHz)	0.706
Number of multiprocessors	13
Single-precision cores per multiprocessor	192
Total single-precision cores	2496
Total single-precision (Gflops)	3524
Total global memory (GB)	5

Parallelization Model of IMPACT-Z on GPU

GPU is using a fine-grain parallel model: create many threads at the same time with each piece of thread executes exactly the same piece of code. In order to make full utilization of GPU computing power, it is preferable to create as many threads as reasonable. For a particle tracking code like IMPACT-Z, it is straight forward to utilize GPU parallel mode by assigning a thread to each particle just by replacing the particle tracking sudo-code:

Do n=1, Ntot; Track(particle(n)); End do

Into GPU kernel call:

*Call TrackKernel<<<NumB, tpB>>>(*particle)*

where Ntot means total number of particles, NumB means number of block, tpB means threads per block where $NumB * tpB \geq Ntot$ is satisfied.

The original MPI-based parallel model used by IMPACT-Z is known as coarse grain parallel model. For each coarse grain, we can still divide workload into finer pieces and utilize GPU fine-grain parallel model. As a result, the original parallel model used by IMPACT-Z doesn't conflict with GPU model, so we can keep the basic structure of IMPACT-Z code. For our following work, we mainly focus on GPU parallelization by setting number of MPI worker equals one.

Beam dynamics study has demonstrated that space charge effect can be neglected in FRIB linac. Therefore, no space charge effect is assumed for the following study.

NVidia has been working with PGI to create a compiler for CUDA-Fortran [6, 7]. After installing PGI CUDA compiler, just by adding -Mcuda flag when compiling IMPACT would automatically include cuda function.

Performance Tuning of IMPACT-Z on GPU

A naïve implementation of IMPACT-Z code on GPU would already bring great performance boost. Simply by rewriting all particle tracking subroutine into GPU core function, around 15 s running time can be achieved. However, further tuning of the code is needed to get better performance. The CUDA command line profiler is used as profiling tool.

PCI Data Traffic Optimization a common bottleneck for GPU computation is frequent data traffic between host and GPU device. A simple solution would be putting all particle distribution data to GPU side and eliminate back-and-forth particle data transformation between CPU and GPU. This simple improvement would boost the running time of the standard test case by around 7 s.

Precision control for a NVidia Tesla K20 GPU card, there are 2496 single-precision cores which add up to 3524 Gflops. At the same time, there are also 832 double-precision cores which add up to 1175 Gflops. It is obvious that using single-precision calculation instead of double-precision calculation contributes to performance boost. IMPACT-Z code is originally using double-precision calculation, switching to single-precision calculation decreases computational time by around 2.5 s. However, the side-effect is a decrease in calculation precision. Benchmark of computation precision is discussed in the following section, benchmark results suggest that using single-precision is acceptable for on-line beam tuning purpose.

Memory access pattern tuning the global memory access pattern needs to be tuned for better performance. Because when doing 2D memory organization, Fortran is using column first mode. As a result, the better way to declare a 6D phase space array of N particles should be particle(N,6) instead of particle(6,N) to avoid stride memory access pattern. This adjustment of memory access pattern results in 0.5 s decrease in calculation time. It has also been found out that constantly allocating and deallocating device memory would result in severe speed decrease. Declare a piece of device memory once and reuse it whenever possible turns out to be a good practice, which saves 0.3 s from total calculation time.

Thread number and block number tuning Choice of number of threads per block (TPB) would also affect calculation speed. For NVidia Tesla K20 GPU, maximum number of TPB is 1024. Because 32 threads are organized in a warp as the minimum unit that gets calculated in single-instruction, so at least 32 threads are needed in a block. Too small TPB would result in insufficient number of warp for work scheduling inside a block, and too large TPB would decrease number of active block on a multi-processor which is bounded by memory resources. After careful tuning, we found out that for our specific case, the optimum TPB equals 128, which results in 0.4 s decrease in calculation time.

Diagnostic function optimization for each integration step, the particle distribution data is processed and beam parameters are calculated and outputted. The naïve implementation of transferring particle distribution data from device to host each step and processing by the host is too slow. To optimize this process, a device-host hybrid data processing procedure is proposed. Unlike on CPU, particle data processing, mostly reduction operation, is not trivial on GPU, because threads can only be synchronized within a block, and block level synchronization can only be achieved using separate kernel call. Data can be shared among all threads within a block using shared on-chip memory. Because the size of shared memory of NVidia Tesla K20 GPU is at most 48K, so a proper number of particle accommodated is 256. Because IMPACT-Z calculates 34 different beam parameters for output, and all of them can be calculated in parallel. We can take advantage of this level of parallelization to increase threads used in a block to 1024. After that, the binary tree is used to do reduction operation [8]. After first round of block synchronization, the remaining data for reduction is usually too small in scale for second round GPU-based reduction. The best choice is to transfer the remaining data to CPU to finish reduction. Each kernel call of this optimized diagnostic function takes about 180 us, and total time spent is about 1 s.

After optimization, the running time for the standard test case is 5.2 s. Though the running time can vary according to the situation, we can still get a general insight on the relative importance among each step.

IMPACT-Z GPU VERSION PROFILING

In this section, the performance of the optimized IMPACT-Z GPU version is profiled and compared with the original

ISBN 978-3-95450-147-2

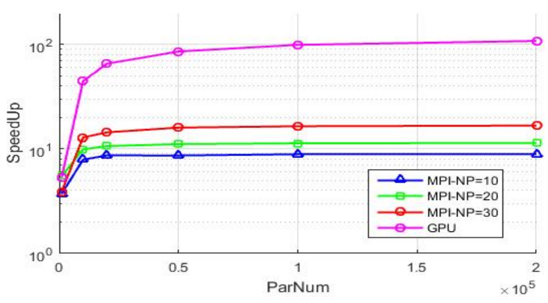


Figure 1: Speed up factor for IMPACT GPU and MPI version vs. particle number. Blue, green, red line means speed up factor for original IMPACT-Z on a 32 threads Intel Xeon CPU with MPI processes to be 10, 20 and 30 vs. particle number. Magenta line means speed up factor for IMPACT-Z GPU version vs. particle number.

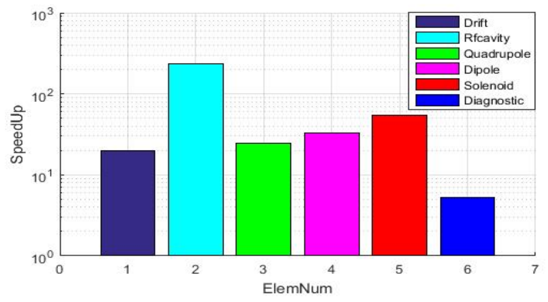


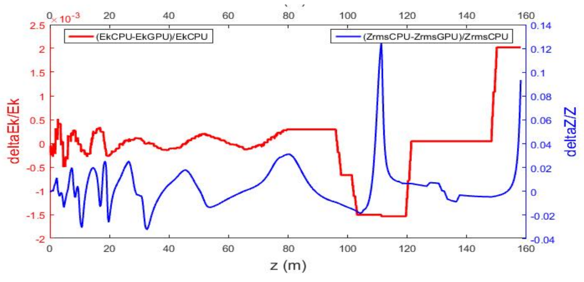
Figure 2: Speed up factor for IMPACT GPU version vs. lattice element type. Rfcavity performs much better because of its higher arithmetic intensity.

version. Figure 1 shows the result of speed up factor in relation with particle number. The standard running case is treated as the baseline. Its running time is then divided by the running time of GPU version to get the speed up factor. Because the baseline CPU is a multicore CPU with 32 threads, and the speed up factor using original MPI with worker number 10, 20 and 30 is also calculated. It can be easily seen that, with particle number increasing, the speed up factor of the original IMPACT-Z MPI version saturate quickly around 18, however, the speed up factor of the IMPACT-Z GPU version doesn't saturate that fast, and can arrive almost 100 for 100K particles.

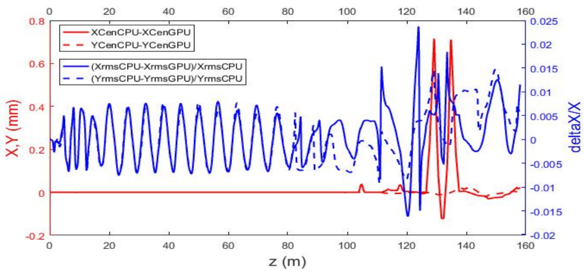
Speed up factor by different kind of lattice element is recorded in Fig. 2 with the standard testing case. It can be seen that RF cavity, which has largest arithmetic intensity, shows best performance boost. On the other hand, for diagnostics, which mostly involves reduction operation and has the lowest arithmetic intensity, shows least performance boost. We can arrive at the conclusion that using GPU would be beneficial when dealing with computational intensive problem instead of memory-bandwidth bounded problem.

IMPACT-Z GPU VERSION BENCHMARK

Using single precision data boosts calculation speed at the cost of calculation precision. Figure 3 shows the benchmark result.Calculation result shows good agreement. The



(a)



(b)

Figure 3: Benchmarked of IMPACT-Z GPU version with the original version (a) left axis: kinetic energy error, right axis: longitudinal rms size error (b) left axis: transverse beam orbit error, right axis: transverse rms size error

average error is 0.013 MeV/u for kinetic energy, 0.10 mm for transverse beam orbit, 0.028 mm for transverse rms size and 3.68e-4 rad for longitudinal rms size. The results confirms that the precision of IMPACT-Z GPU version can meet the precision requirement for on-line beam tuning application.

CONCLUSION

It has been confirmed that NVidia GPGPU is effective to boost the calculation speed of particle tracking-based accelerator simulation code IMPACT-Z. After a series of optimization including PCI data traffic optimization, precision control, memory access pattern tuning, thread number and block number tuning, diagnostic function optimization, the total speed up of around 100 times is expected. Further study reflects that GPU is more effective when handling problem with larger scale and more arithmetic intensity. Benchmark of IMPACT-Z GPU version against original version ensures its precision for on-line beam tuning application. GPU-boosted IMPACT-Z simulator can be useful as a supplement for envelope-based linear model, especially under high power beam operation scenario.

ACKNOWLEDGMENT

The work is supported by the U.S. National Science Foundation under Grant No. PHY-11-02511, and the U.S. Department of Energy Office of Science under Cooperative Agreement DE-SC0000661. This work is also supported in part by Michigan State University through computational resources provided by the Institute for Cyber-Enabled Research.

REFERENCES

[1] Jie Wei et al. *FRYBA3, NA-PAC*, 13, 2013.

[2] Ji Qiang et al. In *Proceedings of the 1999 ACM/IEEE conference on Supercomputing*, page 55. ACM, 1999.

[3] Zhengqi He et al. Tupb058. In *Proc. of LINAC*, volume 12, 2012.

[4] G Shen and Z He. Development status of a thin lens model for frib online model service. In *Proc. of ICAP2015*, 2015.

[5] X Pang and L Rybarcyk. Gpu accelerated online multi-particle beam dynamics simulator for ion linear particle accelerators. *Computer Physics Communications*, 185(3):744–753, 2014.

[6] Greg Ruetsch and Massimiliano Fatica. Cuda fortran for scientists and engineers. *NVIDIA Corporation*, 2701, 2011.

[7] M Wolfe et al. Cuda fortran programming guide and reference. *The Portland Group, Release*, 2012.

[8] Mark Harris et al. Optimizing parallel reduction in cuda. *NVIDIA Developer Technology*, 2(4), 2007.