A Fast Poisson Solver for 3D Space Charge Calculations in a CPU+GPU Heterogeneous Routine

Dawei Zheng, Ursula van Rienen, *Member, IEEE*

University of Rostock, Faculty of Computer Science and Electrical Engineering, D-18059 Rostock, Germany

Space charge calculations play a key role in beam dynamics studies for particle accelerators and various other physical and technical fields. The major work is to solve Poisson's equation for numerous time steps. Therefore, it is important to solve Poisson's equation in a short time. In this paper, we present a CPU+GPU based heterogeneous parallel computation routine with the CUDA platform in a normal PC. The new framework benefits from a novel efficient discrete cosine transform study using the classical Hockney's convolution routine rather than the former one with discrete Fourier transform. A model problem has been studied, which showed an efficiency improvement with the new heterogeneous routine.

***Index Terms*—Poisson's equation, FFT, beam dynamics, space charge.**

I. Introduction

PARTICLE accelerators offer a great variety of research applications nowadays in fields like particle physics, material sciences, chemistry and biology. In these applications, the beam of bunches, a large amount of particles, influences the performance of the accelerator facility. In the beam dynamics study regarding the space charge effects of particles, it is necessary to solve Poisson's equation with proper boundary conditions, usually free boundary conditions. A common method to solve Poisson's equation:

$$-\Delta\varphi = \frac{\rho}{\epsilon_0}, \tag{1}$$

where ρ is the charge density, φ is the electrical potential and ϵ_0 is the permittivity of vacuum, is to use the convolution of the density of charged particles and Green's function in free space [1]. After the zero padding of the charge density and the extensions of Green's function, we process a double-sized fast Fourier transform (FFT) in all dimensions. The electrical potential is easily achieved by multiplying the two and transferring back with an inverse FFT. This method is known as Green's function (GF) method (Hockney's routine) [1] [2] [3].

For huge beam dynamics simulation projects, the Poisson solvers are usually implemented in a parallel routine by pure MPI or OpenMP+MPI hybrid programming. The programs are implemented in supercomputer, cluster or MIC (Many Integrated Core) architecture. Alternatively the NVIDIA has been releasing its GPUs with computational acceleration since 2006. The CUDA (Compute Unified Device Architecture) platform provides the possibility to have computations made by a large number of GPU cores. For pure GPU cluster and MPI+GPU routines, some results have already been presented, e.g. [4]. For other simulation projects using a single workstation or PC, the GPU+CPU acceleration fits the aim to speed up calculation dramatically. However, most codes focus originally on the efficient implementation with supercomputers or clusters, even though these codes would be applicable for PCs as well.

In this paper, we introduce a heterogeneous parallel routine by OpenMP+CUDA framework. In detail, we emphasize the PC implementation, i.e. both CPU and GPU are sufficiently exploited. To fulfill this purpose, we present the cutting reduced integrated Green's function (CRIGF) method, which speeds up the pure CPU time around 15% - 25%. Furthermore, we use a 3-D discrete cosine transform to replace the commonly used 3-D discrete Fourier transform for GF values using CPU computation. Simultaneously, the FFT is applied after the above-mentioned zero-padded charge density is proceeded by using CUDA CUFFT computation on GPU.

In principle, this CPU threads with GPU acceleration should perform better than pure CPU or pure GPU computation in PC-level simulation, that can be seen from our numerical examples.

II. CRIGF with discrete cosine transform routine

The solution of Poisson's equation (1) in free space reads as:

$$\varphi(\mathbf{r}) = \frac{1}{4\pi\varepsilon_0}\cdot\iiint \rho\left(\mathbf{r}'\right)\frac{1}{\|\mathbf{r}-\mathbf{r}'\|}dx'dy'dz', \tag{2}$$

where the vectors are expressed as $\mathbf{r} = (x, y, z)$, $\mathbf{r}' = (x', y', z')$ in Cartesian coordinate. The $\frac{1}{\|\mathbf{r}-\mathbf{r}'\|}$ is defined as Green's function (GF) $G(\mathbf{r}, \mathbf{r}')$:

$$G(\mathbf{r}, \mathbf{r}') = \frac{1}{\sqrt{(x-x')^2 + (y-y')^2 + (z-z')^2}}. \tag{3}$$

With the summation of integrals for Green's function over each equidistant grid cell, the integrated Green's function (IGF) formula is given by

$$\varphi(\mathbf{r_l}) \approx \frac{1}{4\pi\varepsilon_0}\cdot\sum_{\mathbf{l}\;=(1,1,1)}^{(N_x,N_y,N_z)} \rho(\mathbf{r'}_{\mathbf{l}'})\tilde{G}_{\text{IGF}}(\mathbf{r_l}, \mathbf{r}'_{\mathbf{l}'}), \tag{4}$$

where $\mathbf{l} = (i, j, k)$, $\mathbf{l}' = (i', j', k')$ and $\tilde{G}_{\text{IGF}}(\mathbf{r_l}, \mathbf{r}'_{\mathbf{l}'})$ (or simplified $\tilde{G}_{\text{IGF}}(x_i, y_j, z_k)$) is calculated for the Green's function as :

$$\tilde{G}_{\text{IGF}}(\mathbf{r_l}) = \iiint_{\mathbf{r'}_{\mathbf{l}'}-\mathbf{h}/2}^{\mathbf{r_l}-\mathbf{h}/2} G(\mathbf{r}_l, \mathbf{r'}_{l'})\mathrm{d}V', \tag{5}$$

where the $\mathbf{h} = (h_x, h_y, h_z)$ is the discrete stepsizes, $\mathrm{d}V' = \mathrm{d}x'\mathrm{d}y'\mathrm{d}z'$. This integral's expression can be found in [3].

The CRIGF (Cutting Reduced Integrated Green's Function) integral is defined as follows:

$$\tilde{G}_{\text{CRIGF}}(\mathbf{r}_{(i,j,k)}) = \begin{cases} \tilde{G}_{\text{IGF}}(\mathbf{r}_{(i,j,k)}) & 1 \le w \le R_w; \\ h_x h_y h_z G(\mathbf{r}_{(i,j,k)}) & R_w \le w \le C_w; \\ 0 & \text{otherwise}; \end{cases}$$

for w in $\{x, y, z\}$. C_w and R_w are offered to switch between different numerical integrals. More details about how to choose them are discussed in our former work [5].

Normally, after the tilde Green's function calculations, we previously extend the GF values and take the 3-D Fourier transform. However, in this case we provide a discrete cosine transform-I (DCT-I) of tilde GF to replace the extended 3-D Fourier transform, which has been shown in [6].

Theorem II.1. *Suppose a vector* $g = \{g_0, g_1, \cdots g_n\}$, *the real even symmetric extension of* g *is* $\tilde{g}$, *i.e.* $\tilde{g} = \{g_0, g_1, \cdots g_n, g_{n-1}, g_{n-2}, \cdots g_1\}_{2n}$. *If the vector* $\tilde{y} = \frac{1}{2}\mathfrak{F}_{2n}\tilde{g}$ *and* $y = DCT_n(g)$, *where DCT is the first kind of discrete cosine transform (DCT-I), then* $\tilde{y}$ *is the real even symmetric extension of* y.

III. CPU+GPU Heterogeneous parallel implementation

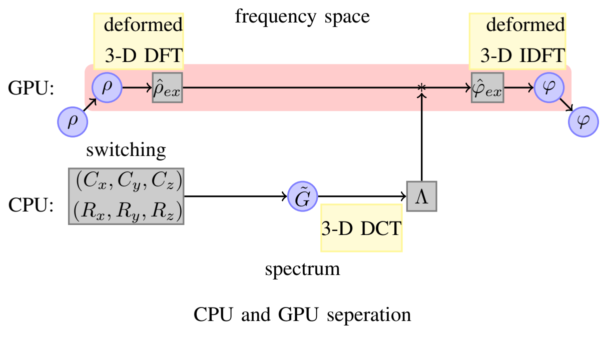

Fig. 1. The schematic plot of the heterogeneous Poisson solver routine with CPU and GPU separation.

The algorithm improvement indicates a heterogeneous implementation of its routine. Firstly, the calculation of Green's function and related 3-D DCT is separated from the calculation of the deformed DFT of the charge density. Secondly, the size of 3-D DCT of the Green's function is smaller than the size of 3-D deformed FFT of the charge density. Thirdly, the calculation time of FFT on a GPU is much lower than that on a CPU with the same size. However, a GPU does not have a strong capability to calculate complicated formulas like IGF values. The above facts give an insight into performing a CPU+GPU heterogeneous routine of our Poisson solver. We split the calculations into two portions by the CPUs' threads as shown in Figure 1: the CPU computing portion and the GPU computing portion. For the CPU computing portion, we compute the tilde Green's function values G and the following 3-D DCT. Simultaneously, we copy the charge density data ρ to the GPU and start the deformed 3-D DFT. After a synchronization between CPU threads, we copy the spectrum Λ, which is the transformed Green's function values, to the GPU to multiply with $\hat{\rho}_{ex}$ to achieve $\hat{\varphi}_{ex}$. The following deformed 3-D IDFT is performed on $\hat{\varphi}_{ex}$ to gain the electrical potential φ. Finally, the calculation results are copied back to RAM for further usage.

IV. Example and Conclusion

Firstly, the algorithm is applied to a charged ellipsoid with a longitudinal-transverse ratio of 30. This example is used for the model of a bunch of electrons with charge, $Q = -1nC$. The relative errors, which we used are defined as follows:

$$\eta_\varphi(i,j,k) := \frac{|\varphi_{i,j,k} - \varphi_{true_{i,j,k}}|}{\max_{i,j,k}|\varphi_{true_{i,j,k}}|}, \text{ and } \hat{\eta}_\varphi : \max_{i,j,k}(\eta_\varphi(i,j,k)).$$

Here, the notations are, $\eta_\varphi(i,j,k)$, $\hat{\eta}_\varphi$, $\varphi_{i,j,k}$ and $\varphi_{true_{i,j,k}}$ as the relative error of the potential at index (i,j,k), the global relative error of the potential, the computed potential at index (i,j,k) and the true potential for the same index, respectively. The comparison is shown in Table I. For the CPU routine, the GF values are calculated by the IGF method and we use a FFT of the extended $\tilde{G}$ values. For the CPU+GPU heterogeneous routine, the GF values are calculated by the RIGF method and we use the 3-D DCT of the GF values.

TABLE I
Elapsed time comparison of pure CPU routine and heterogeneous routine with increasing grids resolution.

N+1	CPU	$\hat{\eta}_\varphi$	CPU+GPU	$\hat{\eta}_\varphi$
32	0.035 s	0.0239	0.0194 s	0.0239
64	0.450 s	0.00317	0.104 s	0.00316
128	4.223 s	0.00125	0.760 s	0.00126

In this paper, we presented a new CPU+GPU heterogeneous routine for the Poisson solver in the calculation of the space charge effect in particle accelerators. The first simulation result showed that, the new routine has a higher efficiency than the former CPU routine. In future, the programming routine will be optimized to fully exploit the CPU+GPU heterogeneous efficiency.

Acknowledgement

Dawei Zheng would like to thank the China Scholarship Council for the support of his study in this work.

References

[1] R. W. Hockney and J. W. Eastwood, *Computer simulation using particles*. CRC Press, 2010.

[2] J. Qiang, S. Lidia, R. Ryne, and C. Limborg-Deprey, "Three-dimensional quasistatic model for high brightness beam dynamics simulation," *Phys. Rev. ST Accel. Beams*, vol. 9, no. 4, Apr. 2006. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevSTAB.9.044204

[3] ——, "Erratum: Three-dimensional quasistatic model for high brightness beam dynamics simulation [phys. rev. ST accel. beams 9, 044204 (2006)]," *Phys. Rev. ST Accel. Beams*, vol. 10, no. 12, Dec. 2007. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevSTAB.10.129901

[4] Q. Lu and J. Amundson, "Synergia CUDA: GPU-accelerated accelerator modeling package," in *Journal of Physics: Conference Series*, vol. 513, no. 5. IOP Publishing, 2014, p. 052021.

[5] D. Zheng, G. Pöplau, and U. van Rienen, "On several Green's function methods for fast Poisson solver in free space," *SCEE, 2014*.

[6] ——, "Efficiency optimization of fast Poisson solver in beam dynamics simulation," *will be submitted to Computer Physics Communications*.