Parallelization of a beam dynamics code and first large scale radio frequency quadrupole simulations

J. Xu, B. Mustapha, V. N. Aseev, and P. N. Ostroumov

Physics Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, Illinois 60439, USA

(Received 10 July 2006; published 3 January 2007)

The design and operation support of hadron (proton and heavy-ion) linear accelerators require substantial use of beam dynamics simulation tools. The beam dynamics code TRACK has been originally developed at Argonne National Laboratory (ANL) to fulfill the special requirements of the rare isotope accelerator (RIA) accelerator systems. From the beginning, the code has been developed to make it useful in the three stages of a linear accelerator project, namely, the design, commissioning, and operation of the machine. To realize this concept, the code has unique features such as end-to-end simulations from the ion source to the final beam destination and automatic procedures for tuning of a multiple charge state heavy-ion beam. The TRACK code has become a general beam dynamics code for hadron linacs and has found wide applications worldwide. Until recently, the code has remained serial except for a simple parallelization used for the simulation of multiple seeds to study the machine errors. To speed up computation, the TRACK Poisson solver has been parallelized. This paper discusses different parallel models for solving the Poisson equation with the primary goal to extend the scalability of the code onto 1024 and more processors of the new generation of supercomputers known as BlueGene (BG/L). Domain decomposition techniques have been adapted and incorporated into the parallel version of the TRACK code. To demonstrate the new capabilities of the parallelized TRACK code, the dynamics of a 45 mA proton beam represented by 10^8 particles has been simulated through the 325 MHz radio frequency quadrupole and initial accelerator section of the proposed FNAL proton driver. The results show the benefits and advantages of large-scale parallel computing in beam dynamics simulations.

DOI: 10.1103/PhysRevSTAB.10.014201 PACS numbers: 29.17.+w, 29.27.−a, 41.75.−i

I. INTRODUCTION

The beam dynamics code TRACK [1] has been developed at ANL over the past few years. TRACK is a ray-tracing code that was originally developed to fulfill the special requirements of the rare isotope accelerator (RIA) accelerator systems [2]. It is based on serial beam dynamics codes for designing and commissioning of various medium energy high-intensity accelerators [3–6]. The TRACK code has become a general beam dynamics code for hadron linacs and has found wide applications worldwide [7–10]. The most recent version of TRACK supports an extensive number of different types of beam line elements with 3D fields and realistic fringe fields. 3D space charge forces for intense beams are calculated by solving the Poisson equation after every tracking step. It also includes the simulation of all possible sources of error, beam monitoring tools, corrective transverse steering, and longitudinal corrections as well as automatic longitudinal and transverse tuning of single and multiple charge state beams. They are particularly important as a basis for the realization of the concept of the ''model driven accelerator'' to make the code useful in the three stages of a linear accelerator project, namely, the design, commissioning, and operation of the machine. Reference [11] contains a brief description of the code. For more details and specific applications of TRACK, see Ref. [12] for error simulations and beam loss analysis in the RIA driver linac and Ref. [13] for automatic longitudinal tuning of a multiple charge state heavy-ion beam.

Parallel computing has become an important tool in most fields of science, speeding up existing calculations and making more challenging and previously unreachable ones possible. To take advantage of the ever-expanding computing capabilities, we started this effort of parallelizing the beam dynamics code TRACK. This will extend the scope of its applications and enable using its unique features with large-scale and fast parallel computing.

In the large-scale error simulations [12], we implemented and used a simple parallel version of the code TRACK, where separate processors simulated the same accelerator with different randomly generated misalignment and radio frequency (RF) errors. While this approach is very useful to study error tolerances and beam loss analysis for a given design, where simulating $\sim 10^6$ particles is sufficient, it cannot be used to simulate a given accelerator setup with a very large number of particles (10^7 or more) especially when including space charge forces for high-intensity beams. By simulating large numbers of beam particles, more of the available phase space is filled which usually results in more realistic simulations and permits studies of beam halo formation. For these reasons we have developed a fully parallel version of the code TRACK capable of performing such large-scale calculations including all realistic effects.

We demonstrate the capabilities of the new parallel version of TRACK by an important application, which is a large-scale simulation of the radio frequency quadrupole (RFQ) and the following medium energy beam transport (MEBT) and initial section of the proposed Fermi National Lab-Proton Driver (FNAL-PD) linac. This is a very special and critical section of the linac, which forms RF bunches and provides the initial acceleration. The RFQ takes a continuous beam from the ion source, bunches it, accelerates it, and focuses it to fit the acceptance of the following accelerator system. The MEBT is a complex transition section which provides a space to house a beam chopper. Space charge simulations in RFQs are very computation intensive. So far, no existing code could perform a large-scale simulation of an RFQ. Existing codes are either serial and could not simulate very large number of particles, such as PARMTEQ [14], or parallel but do not support RFQ simulations, such as IMPACT [15].

In this paper, we present the new parallel version of the beam dynamics code TRACK and its first application for large-scale accelerator simulations. The parallel code TRACK has been used to simulate the initial section of the proposed FermiLab proton driver linac [16], which includes a 325 MHz RFQ [17], a MEBT, and a room temperature linac. The 45 mA beam at the entrance of the RFQ was represented by 10^8 particles in a bunch. The number of particles is close to the actual number of particles per bunch and beam halo formation in both transverse and longitudinal phase spaces can be clearly observed. The choice of the number of particles is dictated not only by the space charge considerations but also by the beam behavior in combined external and space charge fields.

In the following section, we describe the parallelization approach implemented in TRACK and discuss how it compares to methods used in other parallel codes such as IMPACT [15]. In Sec. III, we present the parallel Poisson solver developed for TRACK and how it benchmarks on different large-scale computing platforms. The implementation of the Poisson solver into the code TRACK is discussed in Sec. IV along with some performance and consistency tests relative to the serial version of the code. Finally, Sec. V gives a summary and discusses potential future applications of the parallel version of TRACK.

II. CHOICE OF THE PARALLELIZATION SCHEME

In the TRACK code, a particle is tracked through a beam line element using a step-by-step integration of the equation of motion involving the corresponding electromagnetic (EM) fields. The integration step size is usually dictated by the variation of the element's EM fields. For accuracy, high-order Runge-Kutta (RK) integration (4th) is usually used in every tracking step. In the presence of external EM fields only, the beam particles could be transported independently from end to end like in RAYTRACE [18]. This is valid only for low intensity beams where space charge forces or internal EM fields of the beam are negligible. For high-intensity beams, particles can no longer be considered independent but feel forces from each other due to the electric charge they carry. This is usually treated in an effective way by defining the space charge distribution of the beam and solving the corresponding Poisson equation in order to extract the self generated EM fields of the beam. This is a dynamic effect because the particle distribution in the 3D space changes continuously while the beam is being transported. These variations in internal EM fields are usually slower than the variations in external EM fields. Hence, we may justify the use of an integration step size based on external fields. In the presence of beam space charge, the split-operator integration method is applied as was proposed in [15] but unlike Ref. [15] where the maps are applied to transport particles in the external fields, TRACK uses RK method for particle tracking over short steps. The most distributed versions of the TRACK code use the z-coordinate as an independent variable (z-based code). The results presented in this paper were obtained with z-based code too. As was discussed in Ref. [19], the z-based code produces the same results as t-based code in most accelerator physics applications. Similar comparison for the serial TRACK code has been reported in Ref. [20]. The parallelization methods discussed below are exactly the same for both z- and t-based codes.

From this discussion it is clear that there are two components in a beam dynamics calculation; a single or an individual particle component, which is tracking, and a collective component which is computing the space charge distribution and solving the corresponding Poisson equation to extract the effective internal EM fields. This method of calculating space charge forces is known as the particle in cell (PIC) method [21]. Considering individual particle interactions with each other, known as the direct summation method, would be prohibitively expensive in computing time. The PIC method incorporates a computational grid to represent the space charge distribution of the beam. Each beam particle (macroparticle) deposits a fraction of its charge on the closest grid nodes, thus resulting in a coarse-grained discretized space charge distribution. The EM potential associated with such discretized charge distribution is computed by solving the Poisson equation on the grid. Finally, the forces acting on each individual particle are computed by interpolation from the discretized potential on the grid.

While the tracking component could be easily parallelized, by sharing the beam particles over the computing processors, the space charge component requires a special parallel algorithm. We investigated two possible methodologies of parallelizing the code TRACK. The first option is to use domain decomposition for both the tracking and space charge parts following the model used in the parallel

code IMPACT [15]. For the reasons given below, we have decided to use domain decomposition only for the space charge calculation. In this model, only the space charge calculation requires communications between different processors. The particles are simply distributed among the processors but not the field information. This means that the complete EM fields and space charge grid information are on each processor. Among the advantages of this model are:

(i) It is easy to implement in the versatile serial code which supports extensive functions for the accelerator design, beam dynamics simulations and accelerator tuning;

(ii) It involves communications only when solving the Poisson equation and not for particle tracking;

(iii) Field interpolation does not require additional interprocessor communication.

The main limitation of this model, however, is the memory required on each processor to hold the global EM field and space charge grid information, which may limit the maximum possible size for the space charge grid depending on the total memory available to each processor. However, with the fast development of the supercomputer, the bottleneck of memory and speed will diminish in the near future. Definitely, for the most practical problems being solved currently, there is no memory limitation. There is another consideration to implement the proposed parallelization scheme. Beam halo formation and beam losses in the linac are caused not by the space charge forces only. The machine errors can produce a significant impact on the beam quality and beam losses. The study of combined effects of machine errors in external and space charge fields requires the simulation of hundreds of machine error sets generated from different random number seeds. Usually, $\sim$300 seeds are necessary to extract statistically significant information about the machine error effects on the beam. A timely large statistics error simulation (10^8 particles per seed) for accurate beam loss analysis may require 1024 processors per seed on BG/L type machines [22]. A total of $\sim 3 \times 10^5$ processors will be needed for the full simulation of 300 seeds. Such a large number of processors will be available at the future petascale machines.

In conclusion, a parallel Poisson solver has to be developed and implemented into the original code without significant modifications of the rest of the code.

III. THE PARALLEL POISSON SOLVER

The Poisson solver routine used in TRACK takes beam particle distributions as input and produces the EM fields of the beam on a predefined 3D space charge (SC) grid as output. Since TRACK is a z-based code, all beam particles have the same z coordinate at a given tracking step but have different arrival times (or phases) to that same z. Therefore, the first step in this calculation is to generate the 3D space distribution of the beam at the moment where the beam center is at z from the arrival time information of the beam particles. The next step is to transform the particle distributions to the rest frame of the beam and perform the deposition of the electric charges carried by the beam particles (macroparticles). This is done using the so-called ''cloud in cell'' method where depending on distance a particle deposits a fraction of its charge on the closest 8 nodes of the SC grid defining the SC cell the particle belongs to. At the end of this step the beam is represented by a space charge distribution on the SC grid. The next step consists of solving the corresponding Poisson equation for the electric potential U. In the current solver, we use Cartesian coordinates with rectangular boundary conditions in the transverse directions and periodic boundary conditions in the longitudinal direction:

$$\Delta U = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial z^2} = -\frac{\rho}{\varepsilon_0}$$

with the boundary conditions

$$U(0, y, z) = U(L_x, y, z) = 0,$$
$$U(x, 0, z) = U(x, L_y, z) = 0, \qquad U(x, y, 0) = U(x, y, L_z),$$

where L_x, L_y, and L_z are the x, y, and z dimensions of the SC grid, respectively. The solution is performed using fast Fourier transforms (FFT); sine transforms in the transverse directions and real transforms in the longitudinal direction. Once the potential U is determined on the SC grid, it is straightforward to derive the induced electric field in the rest frame of the beam. By boosting back to the laboratory frame, the EM fields could be determined on each node of the SC grid. A second order interpolation method is used to get the (E, B) fields in the location of a given particle in the next tracking step.

A. Models for the parallel Poisson solver

In a practical computation, the mesh of the SC grid used to solve the Poisson equation is fixed. Typical grid sizes are $N_x \times N_y \times N_z = 32 \times 32 \times 64$, $64 \times 64 \times 128$, $128 \times 128 \times 256$, where N_x, N_y, and N_z are the number of grid points along the x, y, and z directions, respectively. The size of the grid imposes limitations on both the parallelization options and the maximum number of processors to use such that the parallelization efficiency cannot be increased when increasing the number of processors beyond this limit. We here try to work around these limitations and design an efficient parallel model to solve the Poisson equation. For this purpose we have investigated several parallel models. An explanation and comparison of the three most promising models are given in the following subsections.

1. 1D Domain decomposition in the Z direction

In this model the 3D SC grid is decomposed only in the longitudinal Z direction. Each slice in Z is assigned to a

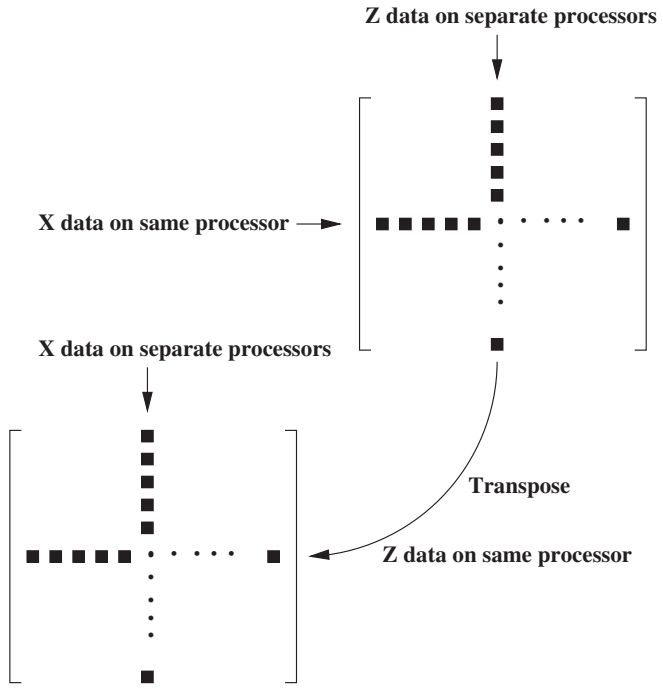

FIG. 1. Transposition of the SC data in the (x, z) plane represented by a simple transposition of a 2D matrix. After transposition the Z data for a given X originally shared among N processor will be on the same processor, so a FFT in the Z direction is easily performed.

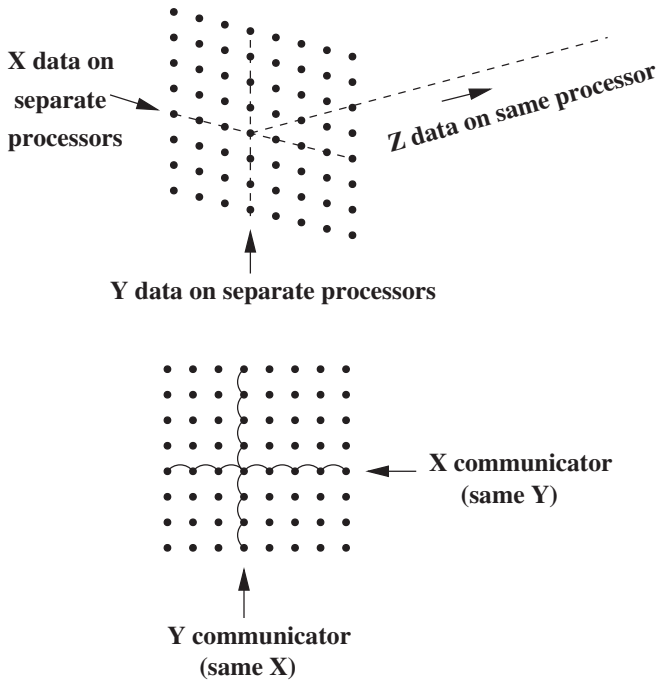

FIG. 2. Definition of two communication groups, one in X and one in Y.

single processor. Since each processor has the whole SC data in the (x, y) plane, the Fourier transform can be directly performed in these two directions. In order to perform the Fourier transform in the longitudinal direction, a transpose of the (x, z) plane data is applied, as shown in Fig. 1. The data are transposed back to its original position after the Fourier transform. This model has the major drawback of limiting the maximum number of processors to the number of SC grid nodes along the Z direction (typically less than 256), making it useful only for a relatively small number of processors. For some applications, the simulation of very large number of beam particles (10^7 to 10^9) may be necessary. For this purpose, the code should be scalable to thousands of processors and 1D domain decomposition is not sufficient.

2. 2D Domain decomposition in X and Y directions

In this model we decompose the SC grid in the transverse plane in both X and Y directions. We choose the decomposition in X and Y directions because they are equivalent from beam dynamics point of view, which leads to a more symmetric data flow between processors. The decomposition could very well be done in (x, z) or (y, z) plane. In fact, the decomposition in (x, z) or (y, z) would benefit from a larger number of processors because the number of SC grid nodes in the Z direction is often double the number used in the X and Y direction. The merit of 2D domain decomposition is that it can easily be used with thousands of processors.

In the case of (x, y) decomposition, FFT can be directly performed in the longitudinal direction Z as the data are located on the same processor. In order to perform FFT in the X and Y directions, we need to transpose the corresponding data to be on the same processor. For this we define two separate processor communication groups. One is composed of processors with the same X location called "X communicator" and the other contains processors with the same Y location named "Y communicator" as shown in Fig. 2. We first apply a global data transposition in the X group, that means in all (x, z) planes, to have complete X data on individual processors of the X group and perform the sine FFT in the X direction. Second, we apply a global transposition in the Y group, in all (y, z) planes, and perform sine FFT in the Y direction. Last, we perform a third global transposition to bring the data to its original position. This method is proved to be the most efficient one for the Poisson solver.

3. 3D Domain decomposition in X, Y, and Z directions

The 1D and 2D domain decomposition models have been implemented by some researchers before [15], in order to use an even larger number of processors; we have implemented a third model. In this model we decompose the SC grid in all three directions. Since the mesh is divided in all three directions, the data transposition required before FFT is performed in the X, Y, and Z directions. Three separate processor communication groups are

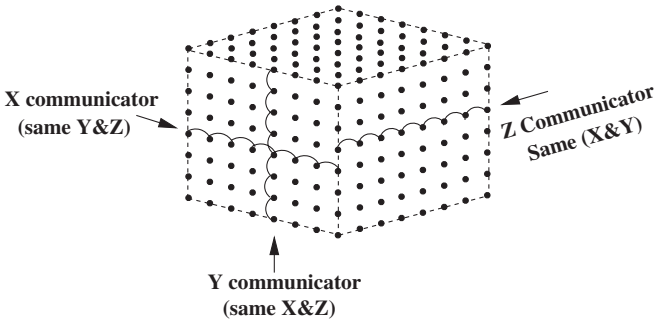

FIG. 3. Definition of three communication groups, one in X, one in Y, and one in Z.

defined. In addition to "X communicator" and "Y communicator", the "Z communicator" is defined as shown in Fig. 3. First, a global data transposition in the X group is performed to have complete X data on individual processors of the X group and perform FFT in the X direction. After performing FFT, the data must be transposed back to its original position. The same operation is performed in the Y and Z communicators. Since more communications between processors are required (twice as much as the 2D case), the 3D domain decomposition may not have good scaling on a small grid size. The advantage of 3D domain decomposition is the possibility to use larger number of processors compared to 2D domain decomposition. For example, a mesh of 32^3 can only be run on 1024 processors with 2D domain decomposition, but can be run on 32 768 processors with 3D domain decomposition. Since BG/L has 3D torus structure networks for communications [22], 3D domain decomposition would be more appropriate. 3D domain decomposition has the potential to assign each processor to a relevant node in a BG/L network, which may result in very high performance of the code. However, additional studies are required for this model.

4. Performance tests of the parallelization methods

We tested the performance of all parallel models on the same platform, which is the IBM BG/L machine at ANL.

Figure 4 shows the speedup factor $F_{\text{speed}} = \text{time}(1)/\text{time}(N)$ defined as the ratio of the computing time obtained with a single processor, time(1), to the one obtained with N processors, time(N). As we can see in Fig. 4, the speedup becomes saturated and levels off at a maximum of about 10 for the 1D domain decomposition model. The 3D model is better but it scales badly at 64 processors; whereas, using 2D domain decomposition we can reach a speedup of about 64 for the same number of processors. The reason for the better performance of the 2D model over the 1D and 3D models on the same platform is due to the smaller size of data that needs to be transferred between different processors. For example, using the 1D model for the grid mesh $128 \times 128 \times 256$ with 4 processors, the total amount of data that needs to be transposed by

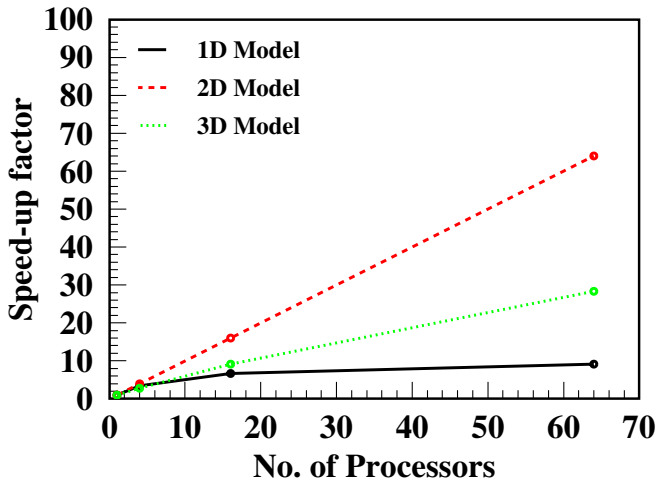

FIG. 4. (Color) Performance test results for 1D, 2D, and 3D domain decomposition models of the parallel Poisson solver on BG/L. Shown is the speedup factor as function of the number of processors. The calculations were performed on the same $128 \times 128 \times 256$ SC grid.

all processors is $128 \times 128 \times 256 \times 8B = 33.5MB$. While using the 2D model with $2 \times 2 = 4$ processors, 2 in each communication group, the total amount of data that needs to be transposed in each group is only $64 \times 128 \times 256 \times 8B = 16.8MB$. That is communication of only half the data across only half of the processors, whereas the 1D model involves communication of all the data across all the processors. In addition, as we increase the number of processors in the 2D model, the amount of data to be transposed will decrease linearly as the number of processors in each communication group increases; whereas the 3D model, as we explained earlier, requires about twice as many communications as the 2D model. Considering the performance and scalability of the 2D domain decomposition model, we choose to use it for our parallel Poisson solver to be implemented into the parallel version of TRACK.

B. Validation of the parallel Poisson solver

In order to validate the newly developed parallel Poisson solver, based on the 2D domain decomposition model, we have compared its results to cases with exact analytical solutions.

We consider the following analytical solution:

$$\Delta U(x, y, z) = -\rho(x, y, z),$$
$$\rho(x, y, z) = 3 \times \sin(x)\sin(y)\sin(z)$$

to be solved in a $L_x = L_y = L_z = 2\pi$ box. This equation has the analytical solution $U(x, y, z) = \sin(x)\sin(y)\sin(z)$. Figure 5 shows the comparison of the solution obtained using the parallel Poisson solver to the exact analytical solution. Other cases with exact analytical solutions were

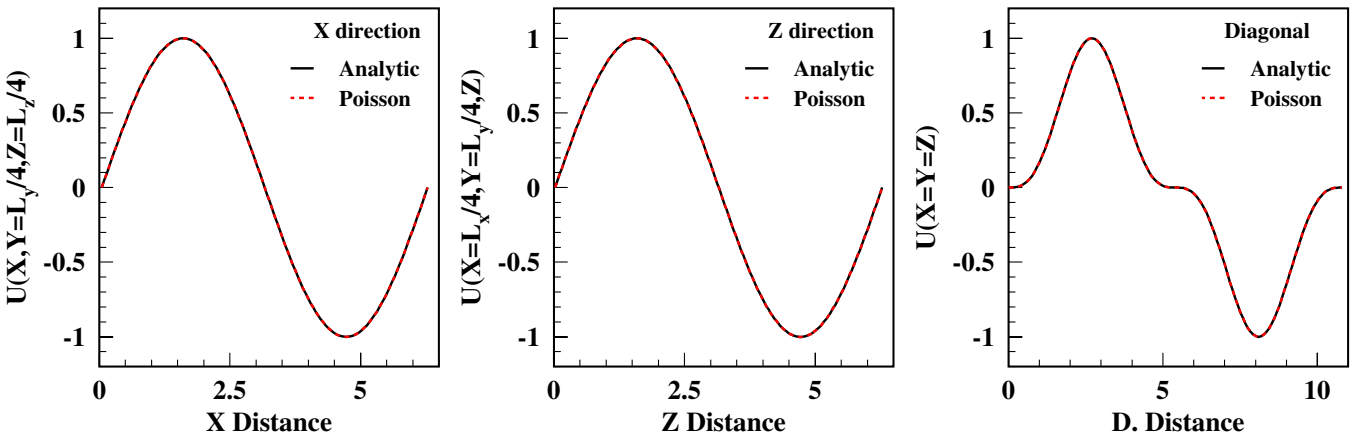

FIG. 5. (Color) Comparison of the results from the parallel Poisson solver to the analytical solution $U(x, y, z) = \sin(x)\sin(y)\sin(z)$ of the equation: $\Delta U(x, y, z) = 3 \times \sin(x)\sin(y)\sin(z)$ in a $2\pi \times 2\pi \times 2\pi$ box.

successfully tested. In all cases the agreement is within round-off error.

C. Performance tests on different platforms

We tested the performance of the parallel Poisson solver based on the 2D domain decomposition model on several platforms, namely, the Linux cluster Jazz at ANL, the IBM SP3 supercomputer Seaborg at NERSC, Iceberg at the Arctic Region Supercomputing Center (ARSC), Lemieux at Pittsburgh Supercomputing Center (PSC), and the IBM BG/L machine at ANL.

The performance is quite different on the different platforms, as shown in Fig. 6. This is due to the different network communication speeds. Among all of them, BG/L has the fastest communication speed, and Lemieux at Pittsburgh Supercomputer Center (PSC) has the slowest communication speed. So on Lemieux, the maximum speedup is only 25, while BG/L can speed up to about 700 on 1024 processors. From this comparison it is clear that what limits the scaling of a Poisson solver is not always the algorithm but it is the network connecting the processors on a given machine.

Since there is no convenient software available to measure the speed of code on BG/L, the tests of the parallel code were performed on Seaborg and Iceberg. The parallel code can achieve 100 MFLOPS/CPU on Seaborg, and 400 MFLOPS/CPU on Iceberg.

IV. IMPLEMENTATION OF THE PARALLEL VERSION OF TRACK

As discussed in Sec. II, the beam dynamics simulation has two components; particle tracking and space charge calculation. For TRACK, we choose to use domain decomposition only for the SC component of the calculation. At the beginning of the calculation, the internally generated or read-in initial particle distribution is equally shared among all the processors. Each processor has the information of full external fields for the beam line or accelerator element being simulated. The SC grid is also defined on all the processors. Before every tracking step, the internal SC fields of the beam have to be computed and combined with the external fields. The first step in the calculation of SC fields is the particle charge deposition on the nodes of the SC grid. This has been done locally, which means

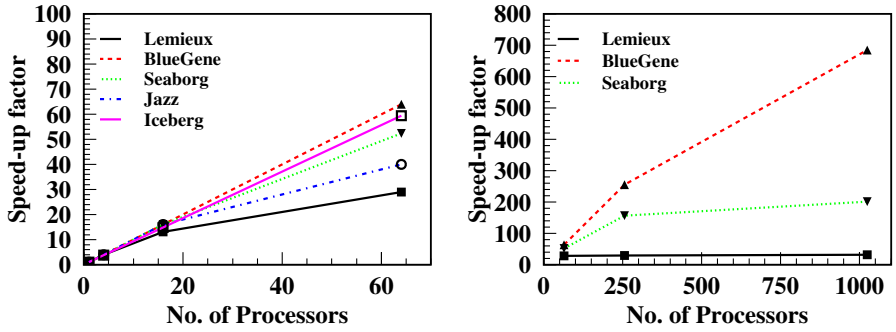

FIG. 6. (Color) Performance test results of the parallel Poisson solver (based on the 2D model) on different Platforms. The left plot shows the speedup factor up to 64 processors on 5 machines and the right one shows the speedup factor up to 1024 processors on 3 machines.

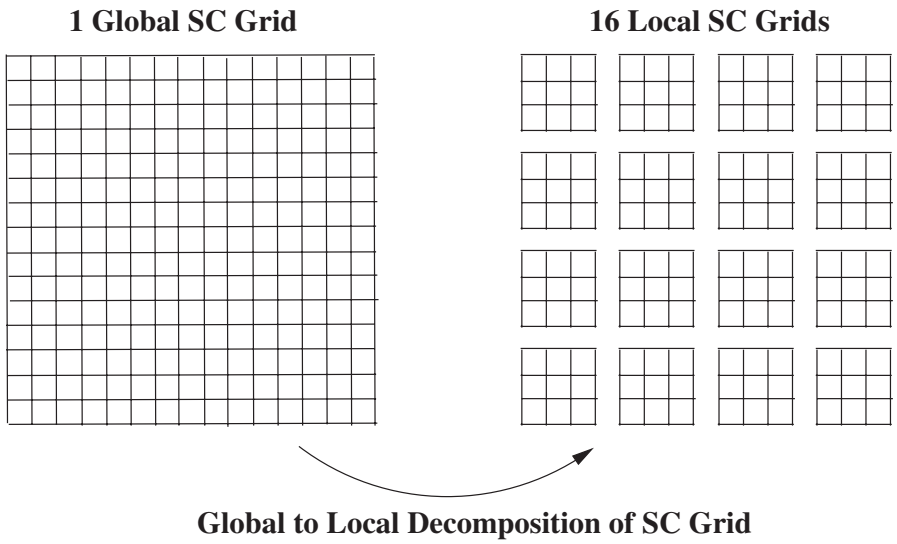

FIG. 7. Example: 2D decomposition of a $16 \times 16 \times N_z$ global SC grid into 16 $4 \times 4 \times N_z$ local SC grids. The Z dimension is not shown because it is not affected by this decomposition.

that each processor deposits the charges of particles which are located on it. At the end of this step, every processor will have a partial SC distribution including only the charge of its particles. To have the SC distribution of the whole beam, we sum up the partial SC distributions on the SC grid using the "All_Reduce" routine of the MPI Library [23], so every processor will have the full SC distribution of the whole beam. In order to use 2D domain decomposition of the Poisson solver presented in III A 2, we subdivide the full SC grid into smaller scale SC grids using 2D space decomposition. Each processor will have a local SC grid containing only part of the SC data as illustrated in Fig. 7. After solving the Poisson equation, we have the solution in the form of potential data shared among the local grids of all processors but not on the global SC grid. A second global communication by calling All_Reduce will bring the potential data from the local grids to the global one to be ready for the tracking part of the calculation. The complete procedure is summarized in Fig. 8.

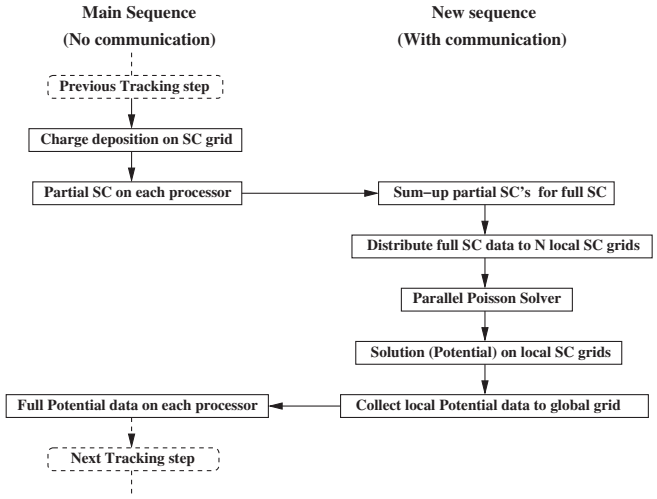

FIG. 8. The parallel model implemented into TRACK showing the main sequence similar to the serial version which does not involve any communication to which a new parallel sequence involving interprocessor communication has been added to calculate space charge forces.

A. Benchmarking against the serial TRACK

In order to benchmark the newly developed parallel version of TRACK, we have performed detailed comparisons with the original serial version of the code. Both codes were used to simulate a 325 MHz RFQ designed to bunch and accelerate a ~45 mA H^- beam from 50 keV to 2.5 MeV. Comparisons of the results for 10^6 particles are shown in Figs. 9–11. Figure 9 compares the evolution of important beam parameters along the RFQ, and Fig. 10 compares phase space plots in the horizontal, vertical, and longitudinal planes. A perfect agreement is obtained. Figure 11 compares the emittance levels (fractions of the beam outside a given emittance) in the three phase planes. These plots show more details in the structure of the beam such as tail or halo formation that may not clearly show up in other plots.

B. Performance tests of the parallel TRACK

A typical 10^6 particle simulation in the 325 MHz RFQ discussed above takes about 6 days on a single processor (700 MHz) of the BG/L machine. This time reduces to 2.2 h on 64 processors and to only 40 min on 256 processors, corresponding to an almost linear speedup with the number of processors. This excellent performance was obtained using a $32 \times 32 \times 64$ SC grid. We noticed, how-

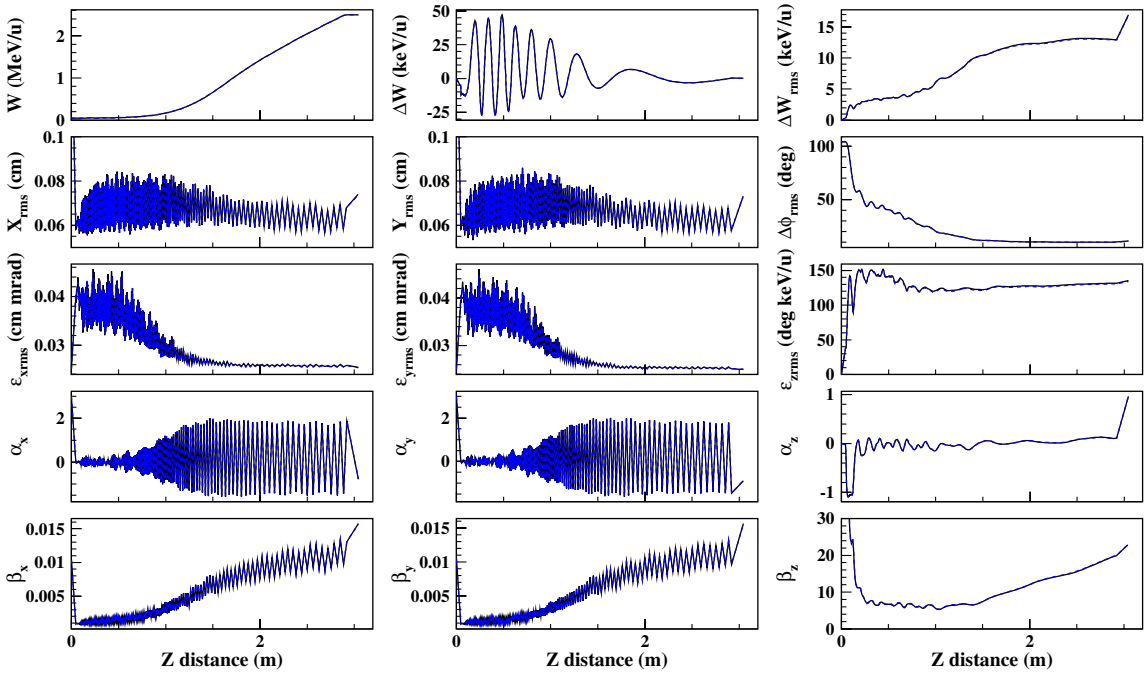

FIG. 9. (Color) Comparisons of simulation results between the serial and parallel versions of TRACK for a 325 MHz RFQ. 10^6 particles of H^- are accelerated from 50 keV to 2.5 MeV. The plots compare the evolution of most important beam parameters along the RFQ, where solid-black curves are from the serial TRACK and the dashed-blue are obtained from the parallel TRACK.

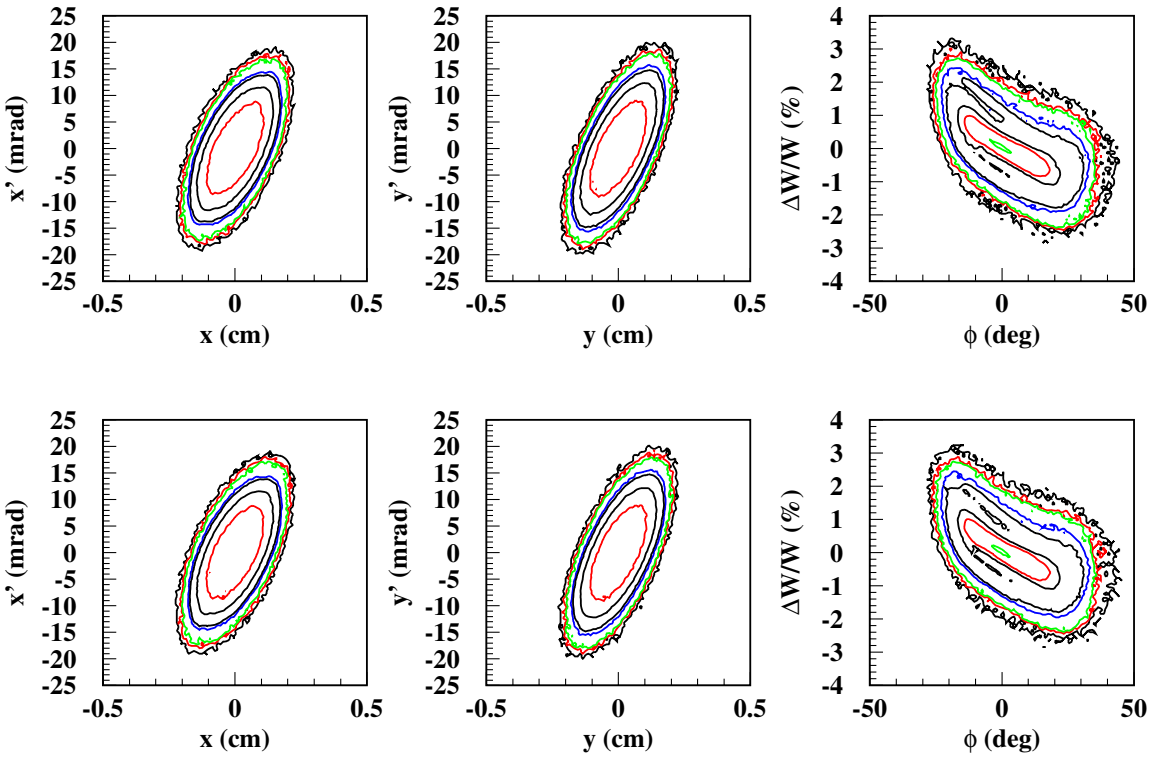

FIG. 10. (Color) Comparisons of simulation results between the serial and parallel versions of TRACK for a 325 MHz RFQ. 10^6 particles of H^- are accelerated from 50 keV to 2.5 MeV. The upper phase space plots are from the parallel code, and the lower ones are from the serial code.

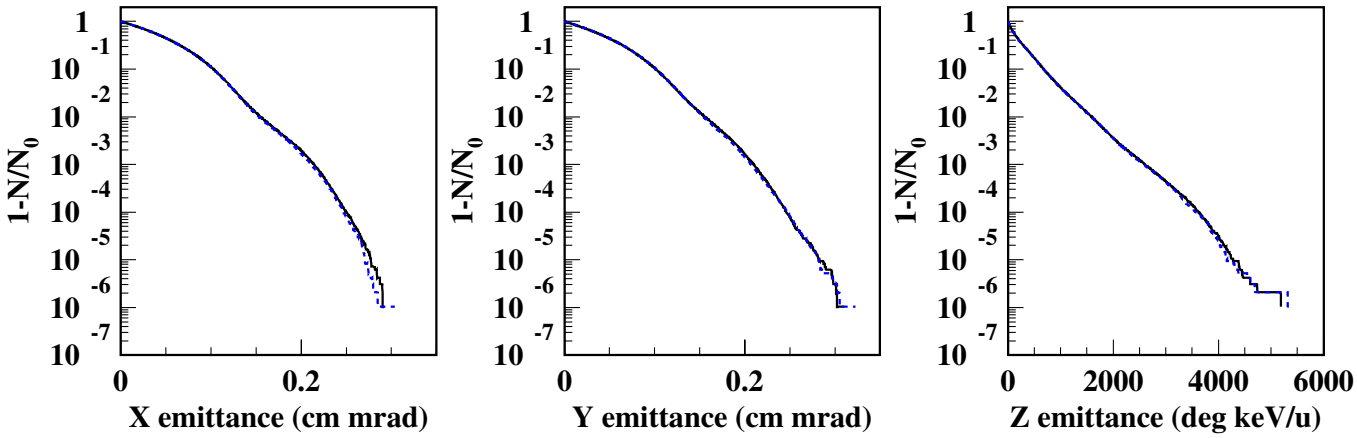

FIG. 11. (Color) Emittance levels, or fractions of the beam outside a given emittance, in the three phase space planes. The solid-black curves are from the serial version and the dashed-blue are from the parallel version.

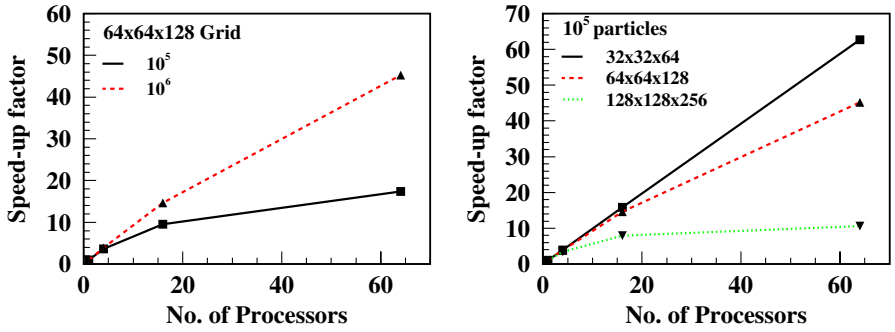

FIG. 12. (Color) Performance test results of the parallel version of TRACK. The left plot compares the speedup factor for the same SC grid $64 \times 64 \times 128$ and different number of particles: 10^5 and 10^6. The right plot compares the speedup factor for the same number of particles (10^6) but different SC grids.

ever, that the performance depends on the size of the SC grid. While we expect a 100% scaling for the tracking part of the calculation, the scaling of the SC component, involving global communications of the SC and Potential data on the SC grid, should depend on the size of the SC grid. This is confirmed by the results presented in Fig. 12. The left plot compares the speedup factor for the same SC grid ($64 \times 64 \times 128$) and different number of particles. Increasing the number of particles from 10^5 to 10^6 reduces the contribution of the SC calculation from 60% to 8% and hence increasing the scaling of the global calculation. The right plot of Fig. 12 compares the speedup factor for different grid sizes with the same number of particles (10^6). The contribution of the SC calculation is about 1% for a $32 \times 32 \times 64$ SC grid, it increases to 8% for a $64 \times 64 \times 128$ grid, and to 60% for a $128 \times 128 \times 256$ grid, which explains the observed reduction in scaling. For beam dynamics simulations in usual accelerator devices, the medium size SC grid ($64 \times 64 \times 128$) is sufficient in most applications. Using this medium size grid solves also the eventual memory limitation discussed in Sec. II. For example, on BG/L, with $256MB$ live memory per processor, every processor could simulate up to $500k$ particles. This should allow us an actual beam bunch simulation of a 50 mA H^- corresponding to 10^9 particles when using the whole machine (2048 processors).

V. LARGE-SCALE SIMULATION OF THE INITIAL SECTION OF A H^- LINAC

Recently we have designed a 325 MHz RFQ for the proton driver (PD) being developed at Fermi National Accelerator Laboratory (FNAL) [16]. The PD RFQ will operate at 325 MHz and accelerate H^- beam with ~45 mA pulsed current up to 2.5 MeV. The parallel TRACK code has been applied for the simulation of 10^8 particles in the RFQ, which consists of 269 accelerating cells and radial matchers at the entrance and exit of the RFQ. Previous simulations have been restricted to 10^6 particles using the TRACK code on a single PC.

The simulation of 100 million particles on $32 \times 32 \times 64$ mesh for the Poisson solver takes about 18 h on BG/L using 1024 processors. To study the beam behavior in space charge fields using a $32 \times 32 \times 64$ SC grid, the simulation

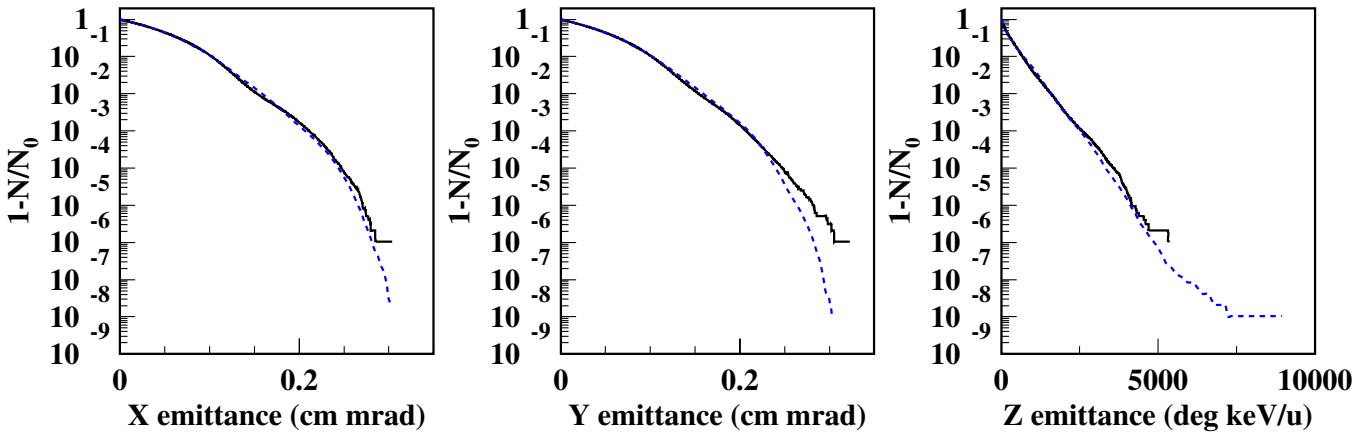

FIG. 13. (Color) Comparison of normalized emittance levels, or fractions of the beam outside a given emittance, in the phase space planes for 10^6 and 10^8 particles. The solid-black curves are for 10^6 particles, and the dashed-blue curves are from 10^8 particles.

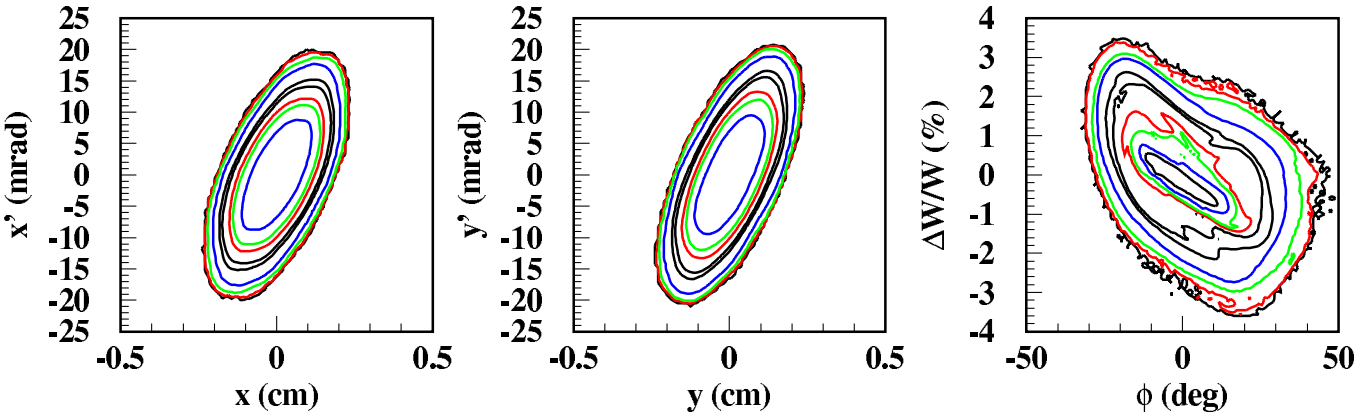

FIG. 14. (Color) Phase space contours for 10^8 particles in the three phase space planes.

of $\sim 10^6$ particles is considered adequate. However, for the detailed beam dynamics in the RFQ where DC beam is bunched in a strong nonlinear external field in the presence of the beam space charge field it is more reasonable to track 10^8 particles. Figure 13 plotted at 15 cm downstream of the accelerating section of the RFQ vanes compares the emittance levels in the three phase space planes for 10^6 and 10^8 particles. The largest possible emittance in the transverse phase space is defined by the RFQ acceptance while the largest emittance in the longitudinal phase space does not

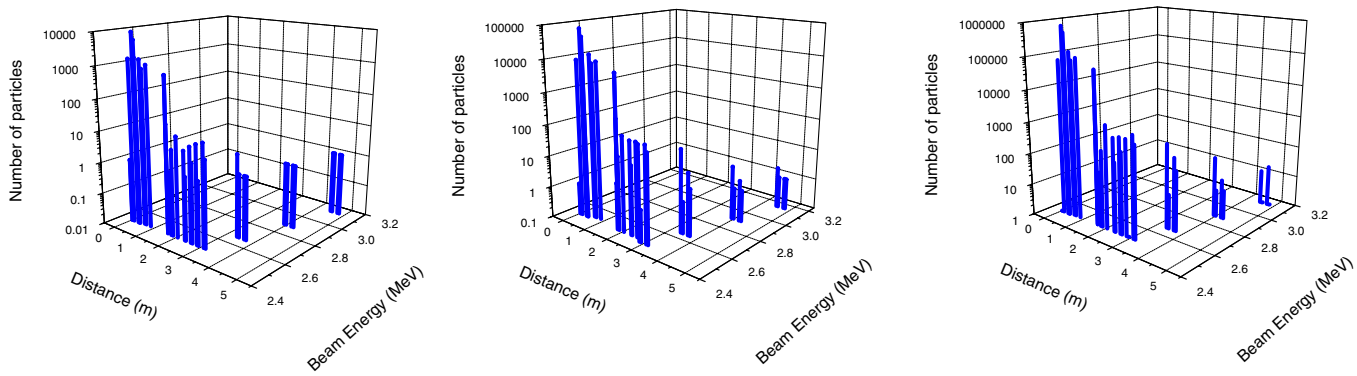

FIG. 15. (Color) Number of lost particles as a function of energy and distance. 10^6 particles (left), 10^7 particles (middle), and 10^8 particles (right).

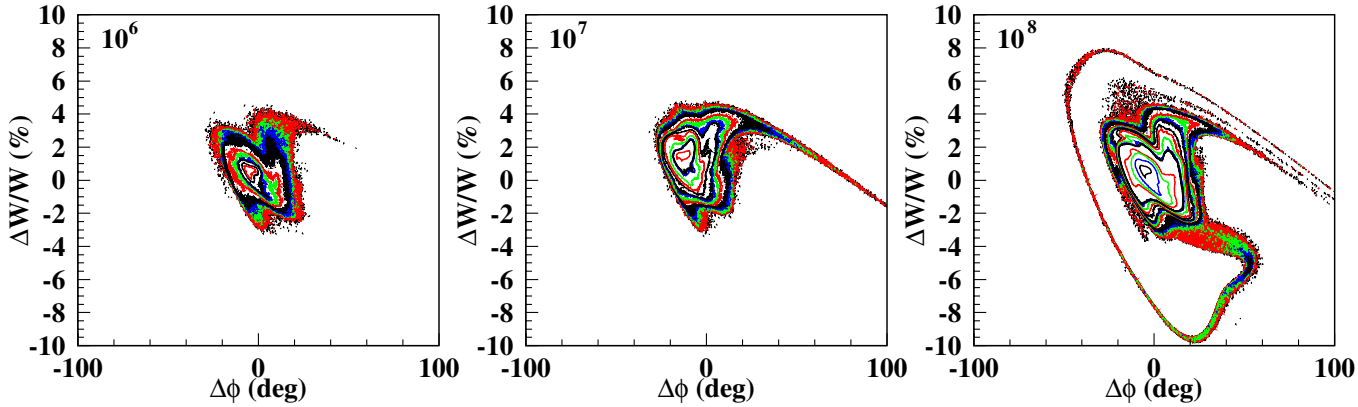

FIG. 16. (Color) Longitudinal phase space plots in the form of $(\phi, \Delta W/W)$ contours. 10^6 particles (left), 10^7 particles (middle), and 10^8 particles (right).

have any distinct boundary. The larger number of particles reveals a larger beam halo in the longitudinal phase space. The phase space 2D contours in (x, x'), (y, y'), and $(\phi, \Delta W/W)$ planes are shown in Fig. 14.

The dynamics of 10^6, 10^7, and 10^8 particles have been simulated through the RFQ, MEBT, and the first five focusing periods of the linac front end [24]. The results are presented in Figs. 15 and 16. As we can see from Fig. 15, the relative number of lost particles does not change as we simulate a different number of particles. But the simulation of the large number of particles reveals a significant beam halo in the $(\phi, \Delta W/W)$ phase plane and increase of the total beam emittance. The latter cannot be seen with a smaller number of simulated particles. Beam halo in the longitudinal phase plane can result in beam losses at higher energies, especially if machine errors are included in the simulations.

VI. SUMMARY

Several parallel models for solving the Poisson equation have been developed and benchmarked. Numerical solutions have been validated by comparing to cases with known analytical solutions. The fastest Poisson solver has been incorporated into the parallel TRACK code. Benchmarking results have been obtained on several different platforms, such as BG/L, Iceberg, Jazz, Lemieux, and Seaborg. The first parallel version of TRACK has been validated and successfully used for beam dynamics simulation of a large number of particles through the initial section of the FNAL-PD linac including the RFQ and MEBT. Various statistical results have been shown, and results from simulating different numbers of particles have been analyzed. The advantage of large-scale parallel simulation of 10^8 particles has been clearly proven. The capabilities of the TRACK code have been greatly extended and can be effectively applied to run on up to 131 072 processors for high-statistics beam dynamics studies including space charge and machine errors.

ACKNOWLEDGMENTS

We would like to thank J. Nolen and K. J. Kim at Argonne National Laboratory for their support on this project. We would also like to thank J. B. Gallagher, J. Valdes, and S. Coghlan from Mathematics and Computer Science Division at Argonne National Laboratory for their help on visualization and computer time on BG/L machine. This work is supported by the U.S. Department of Energy, Office of Nuclear Science, under Contract No. W-31-109-Eng-38.

[1] P. N. Ostroumov and K. W. Shepard, Phys. Rev. ST Accel. Beams **4**, 110101 (2001).
[2] P. N. Ostroumov, J. A. Nolen, and B. Mustapha, Nucl. Instrum. Methods Phys. Res., Sect. A **558**, 25 (2006).
[3] V. A. Moiseev and P. N. Ostroumov, Proceedings of PAC-1989, Chicago, IL, p. 1406.
[4] D. V. Gorelov, and P. N. Ostroumov, Proceedings of European Particle Accelerator Conference 1996, Barcelona, Spain, Vol. 2, 1996, p. 1271.
[5] D. V. Gorelov, P. N. Ostroumov, and R. E. Laxdal, Proceedings of 1997 Particle Accelerator Conference, Vancouver, Canada, 1998, p. 2621.
[6] V. A. Moiseev and P. N. Ostroumov, Proceedings of EPAC-1998, p. 1216.
[7] M. Pasini, R. E. Laxdal, and P. N. Ostroumov, Proceedings of EPAC-2002, p. 933.
[8] A. Lehrach, Proceedings of the PAC-2003, p. 2811.
[9] J. Rodnizki (unpublished).
[10] B. Mustapha, J. Xu, V. N. Aseev, P. N. Ostroumov, D. Jeon, and S. D. Henderson, Proceedings of the LINAC-2006, Knoxville TN, 2006.
[11] V. N. Aseev, P. N. Ostroumov, E. S. Lessner, and B. Mustapha, Proceedings of PAC-05 Conference, Knoxville, Tennessee, 2005.
[12] P. N. Ostroumov, V. N. Aseev, and B. Mustapha, Phys. Rev. ST Accel. Beams **7**, 090101 (2004).
[13] B. Mustapha and P. N. Ostroumov, Phys. Rev. ST Accel. Beams **8**, 090101 (2005).

[14] K. R. Crandall, T. P. Wangler, L. M. Young, J. H. Billen, G. H. Neuschaefer, and D. L. Schrage, Report No. LA-UR-96-1836, 1996 (revised 2005).
[15] J. Qiang, R. D. Ryne, S. Habib, and V. Decyk, J. Comput. Phys. **163**, 434 (2000).
[16] G. W. Foster and J. A. MacLachlan, Proceedings of the LINAC-2002, p. 826.
[17] P. N. Ostroumov, V. N. Aseev, and A. A. Kolomiets, J. Instr. **1**, P04002 (2006).
[18] S. Kowalsky and H. A. Enge, *RAYTRACE* (MIT, Cambridge, Massachusetts, 1987).
[19] J. Qiang, R. D. Ryne, and R. W. Garnett, Phys. Rev. ST Accel. Beams **5**, 064201 (2002).
[20] J.-P. Carneiro, Proceedings of the LINAC-2006, Knoxville, TN, 2006.
[21] R. Hockney and J. Eastwood, *Computer Simulation Using Particles* (Institute of Physics Publishing, London, 1988).
[22] BlueGene: http://www.research.ibm.com/journal/rd/492/adiga.html
[23] MPI: http://www.mpi-forum.org/
[24] P. N. Ostroumov, New J. Phys. **8**, 281 (2006).