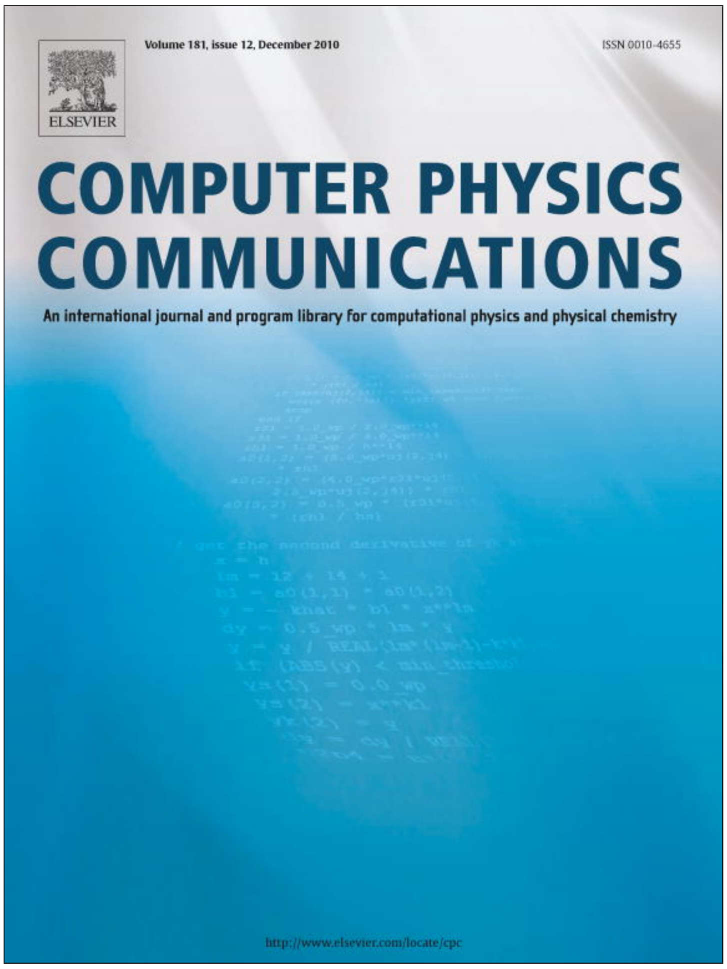
Volume 181, issue 12, December 2010
ISSN 0010-4655
ELSEVIER
COMPUTER PHYSICS COMMUNICATIONS
An international journal and program library for computational physics and physical chemistry
http://www.elsevier.com/locate/cpc

Computer Physics Communications 181 (2010) 2024–2034

Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

Particle-field decomposition and domain decomposition in parallel particle-in-cell beam dynamics simulation

Ji Qiang [*], Xiaoye Li

Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA

ARTICLE INFO

Article history:
Received 11 March 2010
Received in revised form 22 July 2010
Accepted 19 August 2010
Available online 26 August 2010

Keywords:
Parallel particle-in-cell simulation
Particle-field decomposition
Domain decomposition

ABSTRACT

Particle-in-cell (PIC) simulation is widely used in many branches of physics and engineering. In this paper, we give an analysis of the particle-field decomposition method and the domain decomposition method in parallel particle-in-cell beam dynamics simulation. The parallel performance of the two decomposition methods was studied on the Cray XT4 and the IBM Blue Gene/P Computers. The domain decomposition method shows better scalability but is slower than the particle-field decomposition in most cases (up to a few thousand processors) for macroparticle dominant applications. The particle-field decomposition method also shows less memory usage than the domain decomposition method due to its use of perfect static load balance. For applications with a smaller ratio of macroparticles to grid points, the domain decomposition method exhibits better scalability and faster speed. Application of the particle-field decomposition scheme to high-resolution macroparticle-dominant parallel beam dynamics simulation for a future light source linear accelerator is presented as an example.

1. Introduction

The particle-in-cell (PIC) method is widely used for self-consistent simulation in plasma physics, cosmology, and accelerator physics [1–3]. With the advent of high performance large-scale parallel computers, this method was extended to run on those powerful computers to reduce computing time and to explore high resolution physical simulation. In accelerator physics, parallel PIC beam dynamics simulation has been used to study collective effects in high intensity, high brightness beams, to study colliding beams, and to explore advanced methods of particle acceleration [4–12]. Most of those simulations used a standard domain decomposition method to implement the PIC approach on multiple parallel processors. In recent years, we have also explored another parallel implementation method that is based on the particle-field decomposition method [13]. In this paper, we will present an analysis of these two parallel implementation methods on two current leading parallel computers, the Cray XT4, and the IBM Blue Gene/P, with a goal of high-performance, large-scale, high-resolution beam dynamics simulation in a linear accelerator (linac) for a future light source. These light sources provide coherent short wavelength (0.1 nm to 100 nm) radiation and present great opportunities for scientific discoveries in biology, chemistry, material science and physics. In our target simulation, a high brightness electron beam from the photoinjector will pass through a linear accelerator to gain enough energy (multi-GeV or more) for generating short wavelength coherent radiation in an undulator. The quality and efficiency of such a radiation heavily depends on the quality of the electron beam at the exit of the linear accelerator. Large-scale high-resolution beam dynamics simulation is needed in order to accurately predict and to optimize the beam quality through the accelerator.

The organization of the paper is as follows: after the introduction, we will present a brief description of the particle-in-cell method in beam dynamics simulation in Section 2; the parallel domain decomposition implementation method is discussed in Section 3; the particle-field decomposition method is discussed in Section 4; parallel performance studies are presented in Section 5; a target application example is given in Section 6; and a summary is given in Section 7.

2. Particle-in-cell method in beam dynamics simulation

In particle-in-cell (PIC) simulation of beam dynamics in accelerators, a number of macroparticles (i.e. simulation particles with the same charge-to-mass ratio as the real charged particle) are generated from sampling a given initial distribution in phase space. The trajectories

[*] Corresponding author.
E-mail address: JQiang@lbl.gov (J. Qiang).

0010-4655/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.cpc.2010.08.021

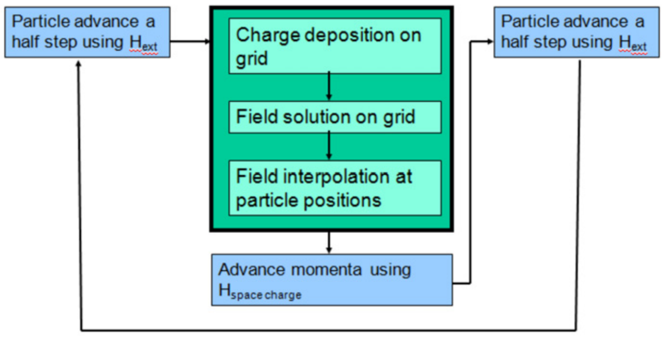
Particle advance a half step using H_{ext}
Charge deposition on grid
Field solution on grid
Field interpolation at particle positions
Particle advance a half step using H_{ext}
Advance momenta using $H_{space\ charge}$

Fig. 1. A schematic plot of a single step loop in the particle-in-cell beam dynamics simulation.

of those charged macroparticles are tracked subject to both the external fields from accelerating and focusing elements and from self-consistent electromagnetic (or electrostatic) space-charge fields from Coulomb interactions among the charged particles. Fig. 1 shows a schematic plot of a single step loop in the PIC beam dynamics simulation. It consists of two major parts: particle advancement and self-consistent field calculation. Using a split-operator method with quasi-static field approximation [14], the macroparticles are advanced in position and momenta half a step using a Hamiltonian corresponding to the external fields. Next, those macroparticles are deposited onto a mesh grid in the moving beam frame to find the charge density on the grid. Then, the Poisson equation is solved for the potential on the grid using the deposited charge density. The electric fields are calculated using the solution of the electric potential on the grid. Those fields are transformed back to the laboratory frame following a Lorentz transformation. The self-consistent Coulomb fields are interpolated back onto individual macroparticles from the mesh grid. The macroparticles are advanced for a full step using the Hamiltonian of the Coulomb fields. Finally, the macroparticles are advanced in position and momenta for another half step using the Hamiltonian from the external fields. This procedure gives second-order accuracy in the step size and can be extended to higher-order accuracy using more substeps of particle advancement and field calculation [15]. Such a single step loop is repeated for a number of times until the beam passes through the accelerator structure.

In accelerator beam dynamics simulation, the transverse size of the beam is often much smaller than the aperture of the conducting vacuum chamber. In this case, the beam is treated as an isolated system. For such a well-isolated charged-particle beam, a 3D rectangular computational box domain is used to contain only the beam itself, and the problem is simulated subject to open boundary conditions. The sizes of the computational box are adjusted at each time step according to the maximum and minimum range of the particles inside the beam. The electric potential inside the beam can be computed using a Green's function method to solve the 3D Poisson equation. The numerical solution in the beam frame on a uniform mesh can be written as

$$\phi(x_i, y_j, z_k) = \frac{h_x h_y h_z}{4\pi\epsilon_0} \sum_{i'=1}^{N_x} \sum_{j'=1}^{N_y} \sum_{k'=1}^{N_z} G(x_i - x_{i'}, y_j - y_{j'}, z_k - z_{k'}) \rho(x_{i'}, y_{j'}, z_{k'}) \tag{1}$$

where h_x, h_y, h_z are the mesh size in each dimension, N_x, N_y, N_z are the mesh grid number in each dimension, $x_i = (i-1)h_x$, $y_j = (j-1)h_y$, $z_k = (k-1)h_z$, and the Green function G is given by

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

The direct summation of the above discrete convolution is computationally expensive and scales as $O(N_x^2 N_y^2 N_z^2)$. Fortunately, this summation can be replaced by a cyclic summation on a doubled computational domain after redefining a new Green function and charge density function [1,8,16]:

$$\phi_c(x_i, y_j, z_k) = \frac{h_x h_y h_z}{4\pi\epsilon_0} \sum_{i'=1}^{2N_x} \sum_{j'=1}^{2N_y} \sum_{k'=1}^{2N_z} G_c(x_i - x_{i'}, y_j - y_{j'}, z_k - z_{k'}) \rho_c(x_{i'}, y_{j'}, z_{k'}) \tag{3}$$

where $i = 1, \dots, 2N_x$, $j = 1, \dots, 2N_y$, $k = 1, \dots, 2N_z$ and

$$\rho_c(x_i, y_j, z_k) = \begin{cases} \rho(x_i, y_j, z_k): & 1 \leqslant i \leqslant N_x;\ 1 \leqslant j \leqslant N_y;\ 1 \leqslant k \leqslant N_z, \\ 0: & N_x < i \leqslant 2N_x \text{ or } N_y < j \leqslant 2N_y \text{ or } N_z < k \leqslant 2N_z, \end{cases} \tag{4}$$

$$G_c(x_i, y_j, z_k) := \begin{cases} G(x_i, y_j, z_k): & 1 \leqslant i \leqslant N_x+1;\ 1 \leqslant j \leqslant N_y+1;\ 1 \leqslant k \leqslant N_z+1, \\ G(x_{2N_x-i+2}, y_j, z_k): & N_x+1 < i \leqslant 2N_x;\ 1 \leqslant j \leqslant N_y+1;\ 1 \leqslant k \leqslant N_z+1, \\ G(x_i, y_{2N_y-j+2}, z_k): & 1 \leqslant i \leqslant N_x+1;\ N_y+1 < j \leqslant 2N_y;\ 1 \leqslant k \leqslant N_z+1, \\ G(x_{2N_x-i+2}, y_{2N_y-j+2}, z_k): & N_x+1 < i \leqslant 2N_x;\ N_y+1 < j \leqslant 2N_y;\ 1 \leqslant k \leqslant N_z+1, \\ G(x_i, y_j, z_{2N_z-k+2}): & 1 \leqslant i \leqslant N_x+1;\ 1 \leqslant j \leqslant N_y+1;\ N_z+1 < k \leqslant 2N_z, \\ G(x_{2N_x-i+2}, y_j, z_{2N_z-k+2}): & N_x+1 < i \leqslant 2N_x;\ 1 \leqslant j \leqslant N_y+1;\ N_z+1 < k \leqslant 2N_z, \\ G(x_i, y_{2N_y-j+2}, z_{2N_z-k+2}): & 1 \leqslant i \leqslant N_x+1;\ N_y+1 < j \leqslant 2N_y;\ N_z+1 < k \leqslant 2N_z, \\ G(x_{2N_x-i+2}, y_{2N_y-j+2}, z_{2N_z-k+2}): & N_x+1 < i \leqslant 2N_x;\ N_y+1 < j \leqslant 2N_y;\ N_z+1 < k \leqslant 2N_z, \end{cases} \tag{5}$$

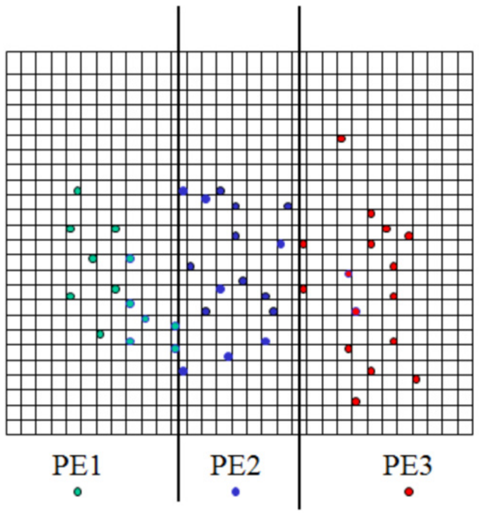

Fig. 2. A schematic plot of the domain decomposition method in particle-in-cell simulation.

$$\rho_c(x_i, y_j, z_k) = \rho_c\big(x_i + 2(L_x + h_x), y_j + 2(L_y + h_y), z_k + 2(L_z + h_z)\big), \tag{6}$$

$$G_c(x_i, y_j, z_k) = G_c\big(x_i + 2(L_x + h_x), y_j + 2(L_y + h_y), z_k + 2(L_z + h_z)\big). \tag{7}$$

The above cyclic summation can be computed efficiently using the FFT method. This makes the computational cost of solving the Poisson equation scale as $O(N_xN_yN_z\log(N_xN_yN_z))$ instead of $O(N_x^2N_y^2N_z^2)$.

3. Parallel domain decomposition implementation method

In the domain decomposition method, the global computational domain defined on the mesh grid is decomposed into a number of small blocks along one direction, two directions, or three directions resulting in the so-called one-dimensional (1D), two-dimensional (2D), or three-dimensional (3D) domain-decomposition method. These blocks are mapped onto a logical Cartesian processor grid. Each processor contains one block domain with local computational boundary. The particles with spatial positions inside the local computational boundary are assigned to the processor containing that part of the computational domain. Since the particles will move around during the process of simulation, after each time step, the spatial coordinates of each particle are checked against the local computational boundary. If they are outside the local boundary range, this particle is sent to the neighboring processor through explicit communication for a new boundary check. This process is repeated a number of times until all particles are inside the boundary of their local computational domain. A schematic plot of a one-dimensional decomposition of the computational domain is shown in Fig. 2. The solid grid lines define the computational domain grids. The thick solid lines define the local computational domain boundary on each processor. In this example, the whole computational domain is decomposed into three blocks, each mapping onto a single processor element (PE) along the z dimension. The local range of the computational domain on each processor is given as $z_{lcmin} \leqslant z \leqslant z_{lcmax}$. Here, the subscripts *lcmin* and *lcmax* specify local minimum and local maximum in that dimension. The number of grids along this dimension on a single processor is defined as

$$Nz_{local} = int\big[(z_{lcmax} - z_{min})/hz\big] + N_g \tag{8}$$

where hz is the mesh size along the z direction, and the quantity N_g refers to the number of guard grids in the local computational domain. The guard grids are the grids added outside the boundary grids for the purpose of parallel simulation. Here, the boundary grids are the outermost grids inside the physical boundary. These guard grids are used as temporary storage of grid quantities from the neighboring processors during the local neighboring communication. In practical applications, a 2D or a 3D decomposition is used in order to maximize the ratio of the computation to the communication.

After particles are moved to their local processor, the charge deposition can be done locally in parallel for all processors. During this stage, the particles located between the boundary grid and the local computational boundary will contribute not only to the local boundary grid in a linear charge density deposition scheme but also contribute to the boundary grid of the neighboring processor along the z dimension. Neighboring communication is used among processors to exchange the data on the guard grids. These data are added to the charge density on the boundary grid to obtain the local density on the grid.

Using the local charge density on the grid, the Poisson equation is solved using the FFT-based Green's function convolution method, with a computational cost that scales as $O(N\log(N))$, where N is the total number of grid points. In our beam dynamics simulation, since we use the FFT method to solve the Poisson equation, we have used a 2D domain-decomposition among processors to keep one global dimension local for the FFT in that dimension followed by transpose and FFT for the other dimensions. Global communication is involved in the transpose that moves the local dimension into the global dimension.

After solution of the Poisson equation, the fields on the grid points are calculated using a finite difference method. To calculate the fields on the boundary grid points, neighboring communication is used to obtain the potentials on the neighboring processor. Then a finite difference method is applied on all processors to obtain the electric fields. With the self-consistent electromagnetic fields and particles on the local computational grid, interpolation is used to obtain the field on individual particles. After obtaining the self-consistent fields on each individual particle, the particle advancement can be done on all processors in parallel for one time step. This process is repeated for many steps until the completion of the simulation.

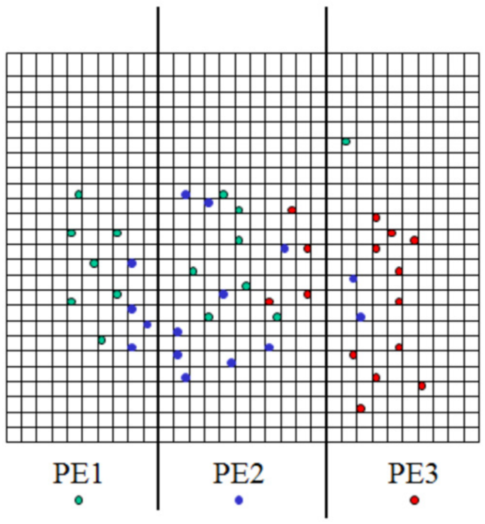

Fig. 3. A schematic plot of the particle-field decomposition method in particle-in-cell simulation.

In the above domain decomposition method, the communication for particle movement is directly related to the number of local macroparticles. The communication for charge density and field exchange is related to the number of local grid points used in the simulation. Those communications are among the local neighboring processors and can be done simultaneously on most parallel computers. Global communication is used only for the matrix transpose in the 3D FFT for solving the Poisson equation.

A uniform computational mesh is used to solve the Poisson's equation using the FFT-based method. For a nonuniform particle density distribution, the number of particles on each processor will not be evenly distributed among the processors if the global computational domain and the mesh grid are uniformly decomposed onto each processor. Dynamic load balance is employed with adjustable frequency to keep the number of macroparticles on each processor approximately equal. This is done by adjusting the local boundary range on each processor using two one-dimensional particle density distribution functions (one for y and one for z) in the two-dimensional domain decomposition method [4]. However, it will work precisely only for a density distribution function which can be separated as a product of two one-dimensional functions along each direction. Meanwhile, this will also result in a different number of local grid points on each processor and load imbalance in the solution of the Poisson equation. Global communication is needed to find the global one-dimensional density function on all processors from the local two-dimensional density function. Here an all-reduce and an all-gather communication operation are used to find the global density function for each dimension in the two-dimensional domain decomposition method.

4. Parallel particle-field decomposition implementation method

The domain decomposition method works well when the particles do not move too far from their positions during each time step. This means that only neighboring processor communication is required. However, in the simulation of beam dynamics in particle accelerators, during each step the particles may move a significant amount inside the problem domain due to the action of external maps associated with the beamline elements. In this case, a lot of communication is required to move these particles to their local processors. Meanwhile, even though the domain decomposition method can achieve a load balance among the particles, the solution of the Poisson equation is not balanced since each processor has a different number of computational grid points, i.e. a different size of subdomain.

Perfect load balance can be achieved, and particle movement avoided, by using a particle-field decomposition method. In this method, the particles and computational domain are uniformly distributed among processors. Each processor possesses the same number of particles and the same number of computational grid points, i.e., the same size of spatial subdomain. Fig. 3 shows a schematic plot of the particle-field decomposition among three processors. The global computational mesh grid is uniformly distributed among three processors. Each processor also has the same number of particles. The spatial coordinates of a particle on a processor might not stay inside the spatial mesh subdomain of that processor. During the stage of charge deposition, the particles are deposited onto a computational grid to obtain the charge density distribution. For the particles with spatial positions outside the local subdomain, an auxiliary computational grid is used to store the charge density. After the deposition, the charge density stored on the auxiliary grid will be sent to the processor containing that subdomain by using a global communication operation. With the charge density distribution local to each processor, the Poisson equation is solved in parallel using the FFT-based Green function method. Since each processor contains the same number of computational subdomain grid points, the work load is well balanced among all processors. After the solution of the Poisson equation, the electric potential on the local subdomain is sent to all processors through another global communication. With the electric potential on each processor, the electric fields are calculated on the grid and interpolated onto individual particles. The particles are advanced using the self-consistent electromagnetic field and the external maps. Since each processor contains the same number of particles and grid points, this operation is also well balanced among processors. Using the particle-field decomposition results in a load balance in both particle advancement and self-consistent field calculation.

There are three major communication steps associated with the particle-field decomposition method. One is the all-reduce operation in the process of collecting charge density onto each processor during charge deposition. One is the all-gather operation for the matrix transpose in the solution of the Poisson equation using the FFT-based method. One is the all-gather operation in the process of gathering the electric potential from the subdomain of each processor to the auxiliary grid of each processor after the solution of the Poisson equation. The volume of communication in the particle-field decomposition approach is proportional to the number of computational grid points instead of the number of moving particles in the domain decomposition approach. Since, in the beam dynamics simulations for our

Table 1
Characteristics of the two parallel machines.

Systems	Cray XT4	IBM Blue Gene/P
Core type	AMD Opteron	IBM PowerPC 450
Clock frequency	2.3 GHz	850 MHz
L1 cache	64 KB (D), 64 KB (I)	32 KB
L2 cache	1 MB	14 stream prefetching
DP Gflops	9.2	3.4
Node		
# cores/node	4	4
L3 cache shared		8 MB
Memory	8 GB	2 GB
Memory bandwidth	42.4–51.2 GB/s	13.6 GB/s
Byte/flop ratio	1.4	1.0
Network		
Topology	3D torus	3D torus
MPI latency	6.5–8.5 μs	3 μs one hop 10 μs to farthest
MPI bandwidth	1.6 GB/s	425 MB/s

target application, the number of particles is much larger than the number of computational grid points, e.g. 10^9 vs. 10^6, the particle-field decomposition approach can significantly reduce the communication cost in this type of simulation.

The particle-field decomposition method also has consequences for memory usage during particle-in-cell beam dynamics simulation. Since there is no particle movement across processors there is no need for a particle manager and no need for dynamic memory management, which simplifies the programming. Since each processor has the same number of particles and computational grid points in the particle-field decomposition method, this results in uniform memory usage among all processors. This avoids the potential problem of insufficient memory on a single processor due to the load imbalance in the domain-decomposition method.

5. Parallel performance studies

5.1. Experiments setup

The performance of the particle-field decomposition implementation method (PFD) and the domain decomposition implementation method (DD) was measured on two mainstream parallel computers: the Cray XT4 at NERSC [19] and the IBM Blue Gene/P at ALCF [20]. We ran the beam dynamics simulation using a section of the accelerator designed for a free electron laser (FEL) based future light source at Lawrence Berkeley National Laboratory [17].

Table 1 summarizes the characteristics of the two parallel machines used in our study. The byte-to-flop ratio shows the balance of the memory bandwidth versus floating-point speed – with larger ratio indicating higher bandwidth relative to the processor speed. The MPI latency and bandwidth are the measured point-to-point communication speed. As can be seen, the XT4 is about 2–3 times more powerful than the Blue Gene/P in most aspects except for the MPI message latency, which is about the same on average for both machines. The Blue Gene/P's lower latency relative to the processor speed gives this machine a scaling advantage for a code requiring many messages. This is apparently the case with the beam dynamics code, and we will show that it scales better on the Blue Gene/P.

On both machines, we studied the following three scenarios of the input datasets:

- *Weak scaling*. The initial problem size is $32 \times 32 \times 32$ grid points with 8 million macroparticles on 64 processors. The problem size is doubled with the doubled number of processors. This is done by doubling the number of grid points and the number of macroparticles, so the average number of macroparticles per grid point is constant, about 244.
- *Strong scaling A*. We fix the problem size with $32 \times 32 \times 2048$ grid points and 1 billion macroparticles. The average number of macroparticles per grid point is about 477.
- *Strong scaling B*. We fix the problem size with $128 \times 128 \times 128$ grid points and 10 million macroparticles. The average number of macroparticles per grid point is about 5.

In our beam dynamics simulation of the light source linear accelerator, the problem size is usually determined by the physical requirements. Previous study suggested that a sufficient number of macroparticles should be used in order to accurately model the shot noise inside the beam [18]. In the other applications such as the simulation of space-charge effects in a ring accelerator, the number of macroparticles can be much smaller. In both cases, the strong scaling will provide more useful information.

In addition to the total runtime, we also studied the computing time spent in the following phases of the simulation:

- *drift* – particle advancement subject to external fields,
- *charge* – particle charge deposition on the grid,
- *Poisson* – solving the Poisson equation,
- *force* – field calculation from the potential, interpolation of fields at grid points to particles, and particle advancement with self-consistent fields, and
- *comm* – communication.

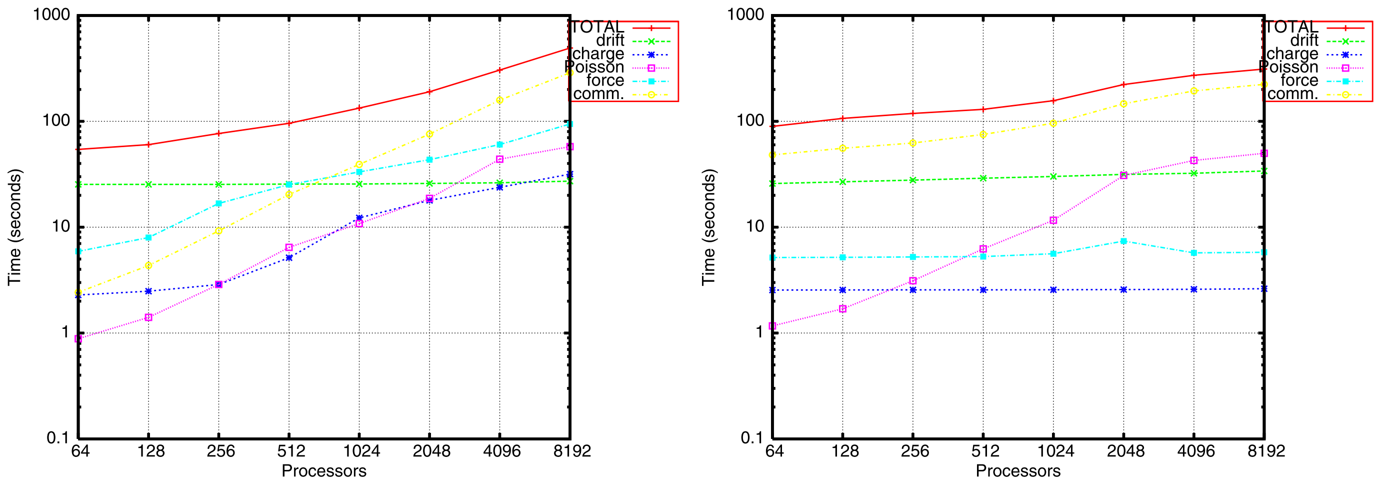

Fig. 4. Computing time as a function of number of processors on Cray XT4 using the particle-field decomposition scheme (left) and the domain decomposition scheme (right) in the weak scaling test.

For the PFD implementation, the *comm* portion includes global communication time in the matrix transpose for solving the Poisson equation, in computing the charge density from particles distributed among processors, in obtaining the electric potential on all mesh points from the local mesh of individual processors. For the DD implementation, the *comm* portion includes communication time spent in moving particles from the physical domain on one processor to the physical domain of another processor, in the global matrix transpose for solving the Poisson equation, in summing up the particle charge density from the particles on the neighboring processors, in exchanging the electric potential between the neighboring processors for field calculation, and in exchanging the electric field between the neighboring processors for field interpolation.

5.2. Performance study on the Cray XT4

We first ran the weak scaling test for both parallel implementation schemes on the Cray XT4. Fig. 4 shows the measured computing time as a function of processor count for both the PFD and the DD implementations.

It is seen that the total computing time with the PFD method increases faster than the DD method. The growth of the computing time spent in communication, charge deposition, and self-consistent force calculation is also faster in the PFD method than that in the DD method. In the PFD method, during the self-consistent electric field calculation, the whole mesh grid is duplicated among all processors for the field computation from the electric potential. When the number of grid points is doubled with the doubled number of processors, this produces twice the computational work on each processor, which is reflected in the increase of the *force* time in Fig. 4. The reason for the *charge* time to increase is because the entire mesh grid is used on each processor to find the charge density on the grid. After the global all-reduce communication, only the charge density on the mesh grid belonging to the local processor is copied into that local mesh grid. As the entire mesh size increases, this causes the growth of computing time in the charge deposition. In contrast, the charge deposition, the self-consistent field calculation, and the particle advancement remain fairly constant in the domain decomposition method because of the roughly constant ratio between problem size and the number of processors in the weak scaling test. The communication in the PFD method involves primarily global communication such as all-gather and all-reduce operations while the communication in the DD method involves mostly nearest-neighbor communication. This results in better scalability of the DD method than that of the PFD method. However, for the total computing time, the PFD method is still faster than the DD method up to 2000 processors until the communication cost becomes dominant. This is probably due to the fact that in those tests, the number of macroparticles is dominant over the number of grid points – about 244 particles per grid point. The fraction of time spent in communication and field calculation is relatively small when the number of processors is below 1000. The time spent on solving the Poisson equation is similar between both methods. The modest difference could be due to load imbalance in the local computational domain grid in the DD method to solve the Poisson equation versus perfect grid balance in the PFD method.

Next we examine the strong scaling case. Fig. 5 shows the computing time as a function of number of processors for both schemes in the strong scaling test. Again, the domain decomposition method shows somewhat better scalability than the particle-field decomposition. Since the problem size is fixed, all the computing time in the particle advancement under external fields, charge deposition, self-consistent field calculation and particle advancement with the field, decreases with the number of processors. The increase in the time to solve the Poisson equation is due to the use of global communication for the matrix transpose. The dominant communication cost in the DD method goes down with increasing number of processors. The communication cost in the PFD method constitutes a small fraction of the total cost at the beginning with 256 processors, increases gradually, and becomes dominant after 2000 processors. However, the total computing time in the PFD method is still less than that of the DD method.

In the above application example, the ratio of the number of macroparticles to that of the grid points is relatively high, about 477. This is needed for the beam dynamics simulation of future light source linear accelerators. In some other applications such as the simulation of space-charge effects in ring accelerators, this ratio can be much smaller. Fig. 6 shows the computing time as a function of number of processors for both schemes in the strong scaling test with $128 \times 128 \times 128$ grid points and 10 million macroparticles. This results in a ratio of macroparticles to grid points that is equal to about 5. It can be seen that the DD method is faster and also scales better than the PFD method for all values of the number of processors.

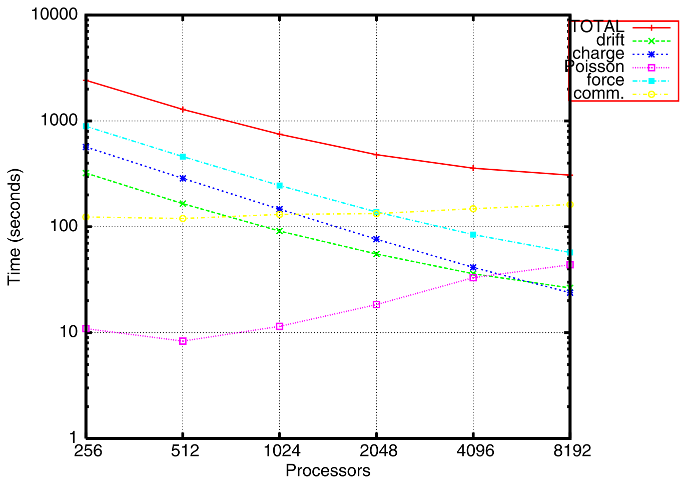

Fig. 5. Computing time as a function of number of processors on Cray XT4 using the particle-field decomposition scheme (left) and the domain decomposition scheme (right) in the strong scaling test A with 32 × 32 × 2048 grid points and 1 billion particles.

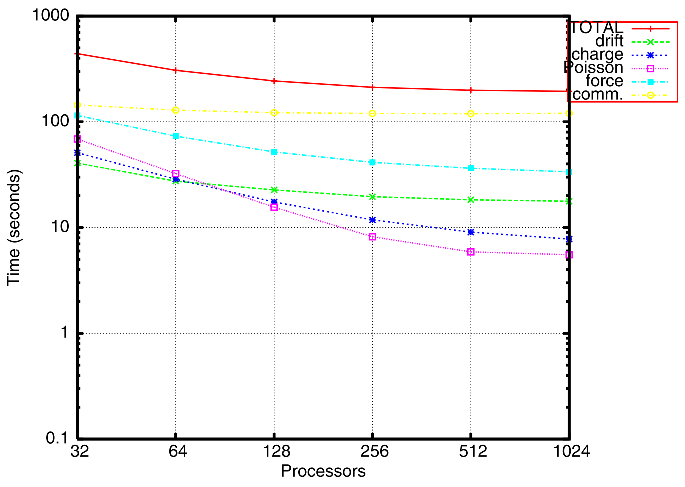

Fig. 6. Computing time as a function of number of processors on Cray XT4 using the particle-field decomposition scheme (left) and the domain decomposition scheme (right) in the strong scaling test B with 128 × 128 × 128 grid points and 10 million particles.

5.3. *Performance study on the IBM Blue Gene/P*

We first consider the weak scaling case, and compare the code performance between the Blue Gene/P and the XT4. Fig. 7 shows the measured runtime as a function of the number of processors for both the PFD and the DD schemes. Compared to Fig. 4, we see that the code ran faster on the XT4 than on the Blue Gene/P with both parallelization schemes, and with different numbers of processors. This is because in absolute terms, the XT4 has overall faster processor, memory and interconnect speed. On the other hand, it is clear that both parallelization schemes scale better on the Blue Gene/P. For example, using 64 processors, on the Blue Gene/P, PFD is over six times as slow and DD is over four times as slow as on the XT4. While using 8192 processors, PFD becomes less than twice as slow and DD is less than three times as slow as on the XT4. The better scaling on the Blue Gene/P is attributed to its faster interconnect speed relative to its processor speed (e.g., lower latency). This is readily confirmed by examining the *comm* phase of the code. In the PFD method, communication takes the smallest amount of time compared to the other four major phases, even with over 8,000 processors. This represents the most dramatic difference compared with the Cray XT4, where communication dominates all the other phases with larger processor count (cf. Fig. 4, left plot). In the DD method, communication is the second dominating phase, only after the *drift* phase. The *drift* phase dominates the total time in both the PFD and DD methods, and is much slower relative to the Cray XT4. This is the result of the much slower processor speed of the Blue Gene/P, because this phase involves a large amount of local computation and no communication. Comparing between the PDF and the DD schemes, we see that in the DD scheme, the common non-scalable phase is the *Poisson* solution, mainly because that involves a nontrivial amount of communication during the matrix transpose. Whereas in the PDF scheme, two other phases are also not very scalable, which are *force* and *charge*. This trend is similar to what was observed on the Cray XT4 (cf. Fig. 4, left plot), and the same reason holds, that is, replicating the mesh points among all the processors introduces an extra amount of work per processor when increasing the mesh size (and the processor count).

Next we consider the strong scaling case. Fig. 8 shows the measured runtime as a function of the number of processors with fixed problem size for both the PFD and DD schemes. In the DD scheme, we have observed severe load imbalance of the macroparticle distribution because the particles move between different domains (and hence processors). This causes some processors to have many more

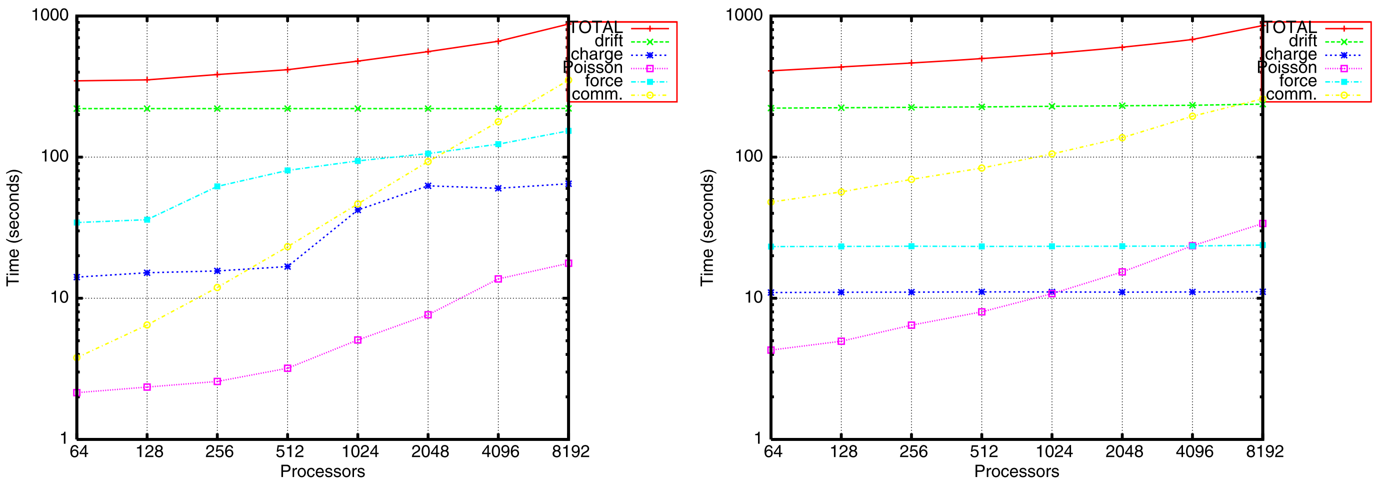

Fig. 7. Computing time as a function of number of processors on the IBM Blue Gene/P using the particle-field decomposition scheme (left) and the domain decomposition scheme (right) in the weak scaling test.

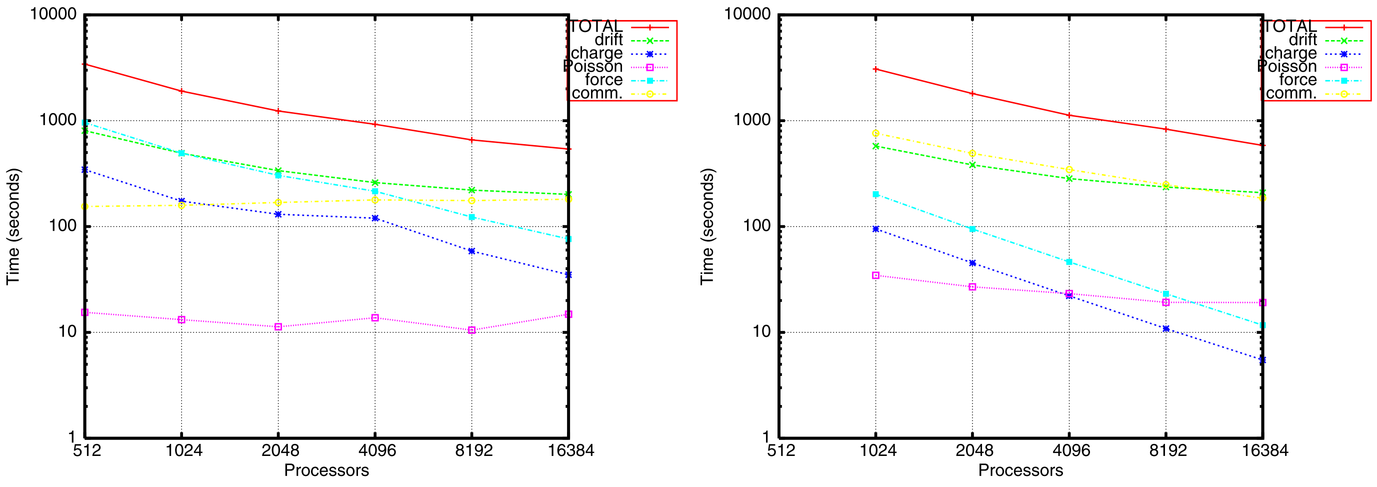

Fig. 8. Computing time as a function of number of processors on the IBM Blue Gene/P using the particle-field decomposition scheme (left) and the domain decomposition scheme (right) in the strong scaling test A with $32 \times 32 \times 2048$ grid points and 1 billion particles.

particles than others at certain simulation steps, and those processors need much more memory; that is why the DD code ran out of memory when using less than 1024 processors.

In terms of the total time, the PFD scheme is much faster in the beginning (1902.4 seconds for PFD vs. 3076.2 seconds for DD at 1024 processor count), but gradually loses the speed relative to the DD method – at the 16K processor count, the DD scheme is almost as fast as the PFD scheme, 584.0 and 540.3 seconds, respectively. Similar to the weak scaling result, the *comm* time is the smallest in the PFD method. But in the DD method, the *comm* time is almost the same as the *drift* time, and these two phases take the majority of the total time. As more and more processors are employed, the *drift* time becomes more and more dominant, which may be due to load imbalance in this phase as reflected in the memory usage shown in the following section.

We also carried out a strong scaling performance study on the IBM Blue Gene/P for the two parallel implementation schemes using a smaller ratio of macroparticles to grid points as we did on the Cray XT computer. Fig. 9 shows the computing time as a function of number of processors for both schemes in the strong scaling test B with $128 \times 128 \times 128$ grid points and 10 million macroparticles. It is seen that in this case, the DD method scales better than the PFD method and also has a faster computing speed.

5.4. *Memory usage*

Besides computing time, we also measured the memory usage in the strong scaling test for both the PDF and the DD methods. This was done on the Cray XT4 at NERSC, using the IPM performance tool [21]. Fig. 10 shows the average, the minimum, and the maximum memory usage among all the processors.

It is seen that for processor counts up to 2000, the memory usage by the particle-field decomposition is smaller than that by the domain decomposition. This difference is especially large on a relatively small number of processors such as the 512 processor case. This is due to severe load imbalance in the domain decomposition method, which precludes the use of such a method on 256 processors because the maximum memory usage on a single processor exceeds the 2 Gbytes memory limit. With increasing number of processors, the average usage of the memory becomes smaller. Even though there still exists a noticeable load imbalance, the maximum memory usage is far below the limit. While the memory usage by the particle-field decomposition method is well balanced among all processors,

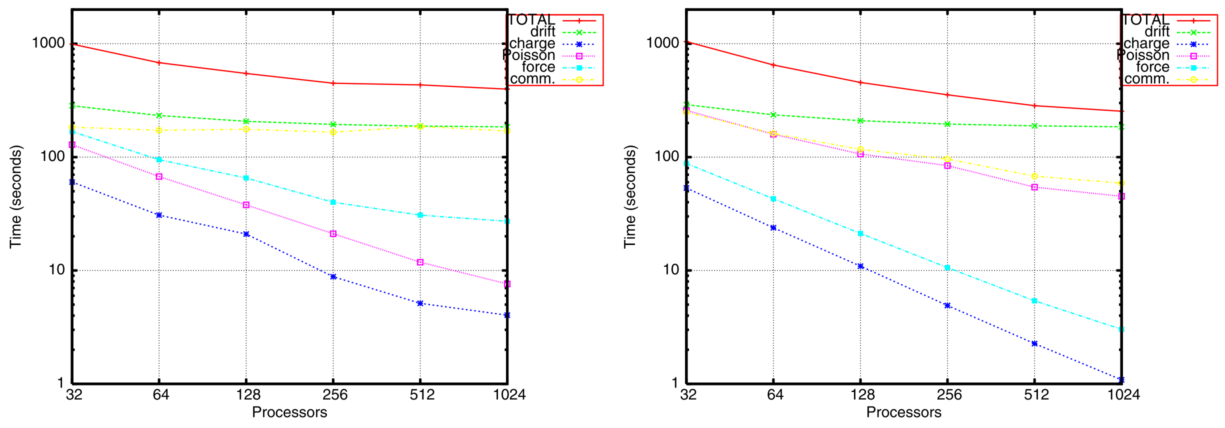

Fig. 9. Computing time as a function of number of processors on the IBM Blue Gene/P using the particle-field decomposition scheme (left) and the domain decomposition scheme (right) in the strong scaling test B with 128 × 128 × 128 grid points and 10 million particles.

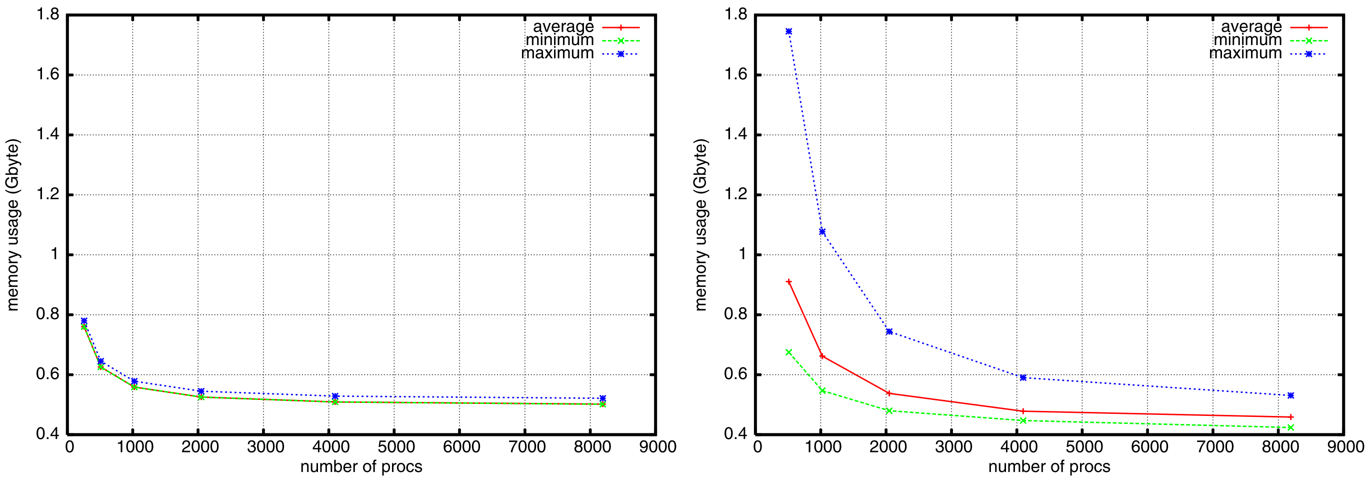

Fig. 10. The average memory usage, the minimum memory usage on a processor, the maximum memory usage on a processor using the particle-field decomposition (left) and the domain decomposition method (right).

the average memory usage does not decrease as fast as the domain decomposition method. This is due to the duplication of the whole mesh on all processors, regardless of the number of processors used.

6. High resolution beam dynamics simulation for future light source linac

From the above performance analysis we see that the particle-field decomposition works more effectively on a medium number of processors when the number of macroparticles is dominant in the simulation. In the beam dynamics simulations for future light sources, the number of macroparticles in a linac simulation needs to be close to the real-world number of electrons in the beam bunch. For a typical application, this number is on the order of one billion. In this section, we show results of high-resolution three-dimensional simulations based on the particle-field decomposition method for a linear accelerator that was designed as a beam delivery system for a next generation free electron laser (FEL) based light source. Fig. 11 shows a schematic plot of this proposed FEL linac at Lawrence Berkeley National laboratory [17]. This accelerator consists of a laser heater, two superconducting RF accelerating linac sections, a longitudinal phase space linearizer, a bunch compressor (BC), and an electron beam switch yard (spreader). The energy of the electron beam at the entrance to the linac is about 40 MeV. The initial peak current is about 70 A with a total charge of 0.8 nC. The initial transverse rms beam size is 0.42 mm with a normalized rms emittance of 0.75 mm-mrad. The electron beam is accelerated to 260 MeV at the end of the first linac (L1) with an energy gain of 13.5 MeV/m within the accelerator cavities. It is followed by the harmonic linearizer (HL) which is the second superconducting linac producing a maximum accelerating gradient of 5 MeV/m at 3.9 GHz. The HL decelerates the electron beam slightly while (together with L1) linearizing the energy chirp of the bunch in front of the bunch compressor (BC). The BC compresses the electron bunches and increases the electron peak current up to about 1 kA. After the bunch compressor, the electron beam is accelerated from about 250 MeV to a final energy of 2.4 GeV before entering a beam switch yard, i.e. a spreader. The spreader includes 10 branch beamline lattices. Each lattice has two parts, the beam take-off section and the FEL fan distribution section. Each part is built as a triplet bend achromat. Here, a kicker, a septum and off-set quadrupoles are used together in place of one bending magnet in the beam take-off section. A more detailed description of this accelerator design can be found in [17].

Fig. 12 shows the transverse root-mean-square (rms) beam size and rms emittance evolving through the accelerator. This simulation was done using 1024 cores on the Cray XT4 computer at NERSC. It took about one and a half hours to complete the simulation. In an accelerator, the transverse beam size has to be controlled within the aperture of vacuum chamber to avoid any potential damage to the accelerator beamline elements from particle losses. The transverse emittance, which is related to the electron beam quality, needs to be

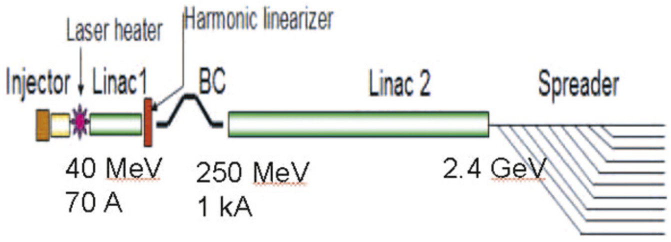

Fig. 11. A schematic plot of a proposed FEL linac at Berkeley Laboratory.

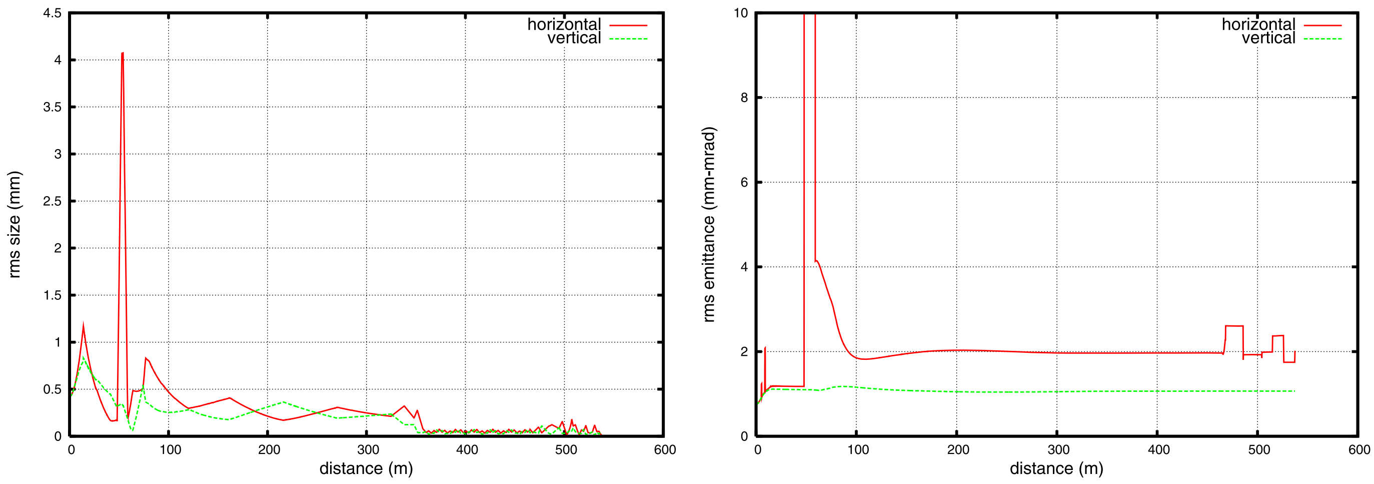

Fig. 12. Transverse rms beam size (left) and transverse rms beam emittance (right) evolve through the accelerator.

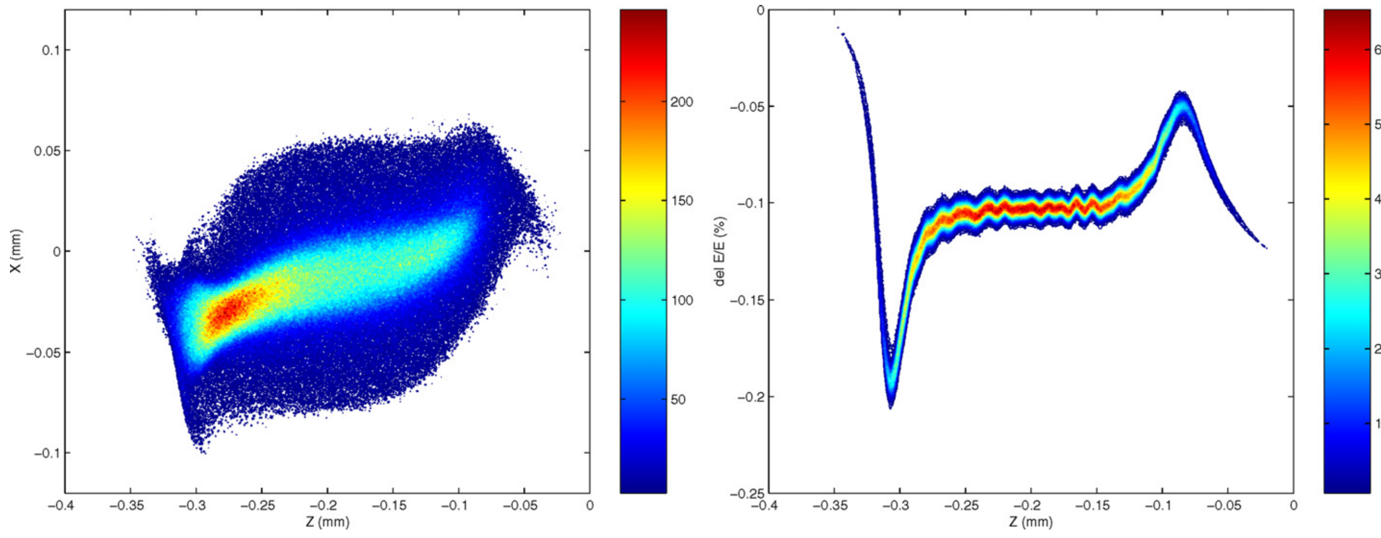

Fig. 13. Final beam phase space distribution in z–x plane (left) and in z–$del\, E/E$ plane (right).

kept as small as possible in order to improve the efficiency of producing light-source radiation. It is seen from Fig. 12 that there is some noticeable growth of the emittance through the accelerator; this is known to be driven by self-consistent space-charge effects. Large-scale simulation will be essential to developing new approaches to minimize emittance growth in future designs. Fig. 13 shows the beam phase space distribution in the z–x plane and in the z–$del\, E/E$ plane at the end of the accelerator. There exists a longitudinal-transverse correlation in the z–x plane due to coherent radiation effects when the beam passes through the bunch compressor in the accelerator. In the longitudinal z–$del\, E/E$ plane, small ripples can be observed in the distribution. This ripple comes from the microbunching instability driven by space-charge effects, structure wake fields and coherent synchrotron radiation inside the accelerator. A large number of macroparticles is needed in order to accurately model the mircobunching instability.

7. Summary

In this paper, we presented a detailed analysis of the domain decomposition and the particle-field decomposition implementation schemes for parallel particle-in-cell simulation of beam dynamics in particle accelerators. The parallel performance of the two methods was studied on two mainstream parallel architectures, the Cray XT and the IBM Blue-Gene. The domain decomposition method shows better scalability on a large number of processors on both computers due to the fact that local neighboring communication is used in this method. However, for a macroparticle dominant application, it is still slower than the particle-field decomposition method up to a few thousand processors. Moreover, the strong load imbalance in the domain decomposition method requires more memory, which, in some cases, precludes the running of beam dynamics simulations on a small number of processors. For our target application of beam dynamics simulation of a linear accelerator for a future light source, a large number of macroparticles is needed in order to accurately model the shot noise of the electron beam. In this case, the particle-field decomposition is more appropriate. For other applications such as the study of space-charge effects in ring accelerators, when the ratio of macroparticles to grid points is smaller, the domain-decomposition method becomes more appropriate.

Acknowledgements

We would like to thank Dr. A.A. Zholents for the light source linear accelerator lattice parameters and Dr. R.D. Ryne for reading the manuscript and useful discussions. This research was supported by the Office of Science of the US Department of Energy under Contract No. DE-AC02-05CH11231. This research used resources of the National Energy Research Scientific Computing Center at Lawrence Berkeley National Laboratory, and the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the US Department of Energy under contract DE-AC02-06CH11357.

References

[1] R.W. Hockney, J.W. Eastwood, Computer Simulation Using Particles, Adam Hilger, New York, 1988.
[2] C.K. Birdsall, A.B. Langdon, Plasma Physics Via Computer Simulation, Taylor & Francis Group, New York, NY 10016.
[3] J.M. Dawson, Rev. Mod. Phys. 55 (1983) 403.
[4] J. Qiang, R. Ryne, S. Habib, V. Decyk, J. Comp. Phys. 163 (2000) 434.
[5] J. Qiang, R. Ryne, B. Blind, J. Billen, T. Bhatia, R. Garnett, G. Neuschaefer, H. Takeda, Nucl. Instrum. Methods 457 (2001) 1.
[6] J. Qiang, P.L. Colestock, D. Gilpatrick, H.V. Smith, T.P. Wangler, M.E. Schulze, Phys. Rev. ST Accel. Beams 5 (2002) 124201.
[7] J. Qiang, D. Todd, D. Leitner, Comp. Phys. Commun. 175 (2006) 416.
[8] J. Qiang, S. Lidia, R.D. Ryne, C. Limborg-Deprey, Phys. Rev. ST Accel. Beams 9 (2006) 044204.
[9] J. Amundson, P. Spentzouris, J. Qiang, R. Ryne, J. Comp. Phys. 211 (2006) 229.
[10] J. Qiang, M. Furman, R. Ryne, Phys. Rev. ST Accel. Beams 5 (October 2002) 104402.
[11] C. Nieter, J.R. Cary, J. Comp. Phys. 196 (2004) 538.
[12] R.A. Fonseca, et al., in: Lecture Notes in Comput. Sci., vol. 2329, Springer, Heidelberg, 2002, p. 342.
[13] J. Qiang, M. Furman, R. Ryne, J. Comp. Phys. 198 (2004) 278.
[14] R.D. Ryne, J. Qiang, S. Habib, in: J. Rosenzweig, L. Serafini (Eds.), The Physics of High Brightness Beams, World Scientific, 2000, p. 91.
[15] H. Yoshida, Phys. Lett. A 150 (1990) 262.
[16] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, Numerical Recipes in FORTRAN: The Art of Scientific Computing, 2nd ed., Cambridge University Press, Cambridge, England, 1992.
[17] E. Kur, G. Penn, J. Qiang, M. Venturini, R.P. Wells, A. Zholents, Accelerator Design Study for a Soft X-Ray Free Electron Laser at the Lawrence Berkeley National Laboratory, LBNL-2670E, Sep. 1, 2009.
[18] J. Qiang, R.D. Ryne, M. Venturini, A.A. Zholents, I.V. Pogorelov, Phys. Rev. ST Accel. Beams 12 (2009) 100702.
[19] http://www.nersc.gov/nusers/systems/franklin/.
[20] http://www.alcf.anl.gov/support/gettingstarted/.
[21] IPM, http://www.nersc.gov/nusers/resources/software/tools/ipm.php.