Journal of Computational Physics 228 (2009) 5340–5348

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Enhancing parallel quasi-static particle-in-cell simulations with a pipelining algorithm

B. Feng [a,*], C. Huang [b], V. Decyk [b], W.B. Mori [b,c], P. Muggli [a], T. Katsouleas [d]

[a] Departments of Electrical Engineering and Physics and Astronomy, University of Southern California, Los Angeles, CA 90089, USA
[b] Department of Physics and Astronomy, University of California, Los Angeles, CA 90095, USA
[c] Department of Electrical Engineering, University of California, Los Angeles, CA 90095, USA
[d] Pratt School of Engineering, Duke University, Durham, NC 27708, USA

ARTICLE INFO

Article history:
Received 25 July 2008
Received in revised form 18 February 2009
Accepted 10 April 2009
Available online 23 April 2009

MSC:
65Z05
78M50

Keywords:
Quasi-static
Particle-in-cell simulation
QuickPIC
Pipelining
e-Cloud
Plasma wakefield acceleration
Plasma accelerator

ABSTRACT

A pipelining algorithm to overcome the limitation on scaling quasi-static particle-in-cell models of relativistic beams in plasmas to a very large number of processors is described. The pipelining algorithm uses multiple groups of processors and optimizes the job allocation on the processors in parallel computing. The algorithm is implemented on the quasi-static code QuickPIC and is shown to scale to over 10^3 processors and increased the scale and speed by two orders of magnitude over the non-pipelined model. The new approach opens the door to performing full scale 3D simulations of future plasma wakefield accelerators or full lifetime models of beam interaction with electron clouds in circular accelerators such as the Large Hadron Collider (LHC) at CERN.

1. Introduction

Electron clouds in large circular accelerators and particle acceleration in plasma wakefield are examples of nonlinear beam–plasma interactions where a large range of timescales are involved. Electron cloud (e-cloud) has been observed in many circular proton and positron machines, where spurious electrons are generated and accumulated inside the pipe and lead to beam instabilities. The built-up electron cloud interacts with the beam particles during 10^3–10^4 turns and causes beam degradation [1–3], such as emittance growth, beam size blowup, beam loss and tune shift. In addition, plasma wakefield acceleration for both particle beam and laser drivers has attracted a great deal of interest recently [4]. Much of this interest is in a regime in which plasma electrons are evacuated by the intense electron or laser drive beam and then pulled back by the stationary ions, forming a spherical accelerating structure. Self-trapped or externally injected electron beam needs to propagate 10^3–10^6 plasma wavelengths to reach the desired energy.

* Corresponding author. Tel.: +1 213 880 1352.
E-mail address: bfeng@usc.edu (B. Feng).

0021-9991/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.04.019

In both of these applications a short "beam" (electrons, positrons, protons, or photons) propagate through a long region of plasma or electron cloud. The "beam" evolves very slowly compared to the time it takes the "beam" to pass by a plasma particle, which enables the frozen field or quasi-static approximation. QuickPIC is a fully parallelized particle-in-cell (PIC) code based on the quasi-static approximation. With this approximation, the three-dimensional (3D) full Maxwell's equations are not solved in the full 3D PIC code such as OSIRIS, but instead the electromagnetic fields are obtained by solving two-dimensional (2D) quasi-static equations [5] shown below,

$$-\nabla_{\perp}^2\phi(x,y,s,\xi) = 4\pi\rho(x,y,s,\xi) \tag{1}$$

$$-\nabla_{\perp}^2\vec{A}(x,y,s,\xi) = 4\pi\vec{J}(x,y,s,\xi)/c \tag{2}$$

where ϕ and $\vec{A}$ are scalar potential and vector potential in the Lorentz gauge, and ρ and $\vec{J}$ are the charge density and current density, respectively. With the quasi-static approximation, the coordinates are transformed from (x, y, z, t) to $(x, y, s = z, \xi = ct - z)$, where z is the coordinate for the propagation direction and x, y are the transverse coordinates, c is the speed of light. s measures the propagation distance and corresponds to the time scale over which the beam evolves. ξ measures the position relative to the beam and corresponds to the time scale over which the plasma or the electron cloud evolves as they interact with the beam. Eqs. (1) and (2) are solved by treating s as a parameter, and using ξ as a time-like variable.

The 2D quasi-static equations are solved with spectral solvers using Fast Fourier Transform (FFT). The 2D routine starts with initialization of an unperturbed plasma or electron cloud slab in x–y plane, and advances it through the simulation box slab by slab by increasing ξ. The 2D routine updates the positions and the velocities of the plasma or e-cloud particles as well as solves the electromagnetic field. The 2D calculation cannot be performed on all slabs simultaneously but sequentially since each slab is updated from its previous one, except the first slab. There are two different modes of the quasi-static approximation in QuickPIC: full quasi-static and basic quasi-static, and they are used to model the plasma wakefield acceleration experiments [4] and the electron cloud beam interaction in circular machines [6], respectively. The full quasi-static approximation includes the axial plasma current. This current is important for the plasma wakefield modeling because the plasma electrons move at relativistic speeds. Detailed description of the QuickPIC algorithm with the full quasi-static approximation is described in Ref. [5]. In the electron cloud case, the cloud electrons move at non-relativistic speeds, and they are modeled by the basic quasi-static approximation where the axial current is ignored. Eqs. (1) and (2) are further reduced to 2D scalar Poisson equations for $\phi$and A_z(A_z is the z component of the vector potential), in the basic quasi-static approximation [7]. For the simulation of the e-cloud problem, the electron cloud is distributed all over the ring and interacts with the beam continuously [8]. Betatron and synchrotron oscillations of the beam particles are also included as an external force term ($\vec{F}_{ext}$ in Fig. 1) added to the equation of motion of the beam particles. This captures the effect of the external quadrupole magnets and RF fields of the accelerators [6].

Compared to full PIC codes such as OSIRIS, QuickPIC speeds up the computation by 2–5 orders of magnitudes with the frozen field or quasi-static approximation. The CPU time saving mainly comes from two multiplicative factors: (i) the reduced total number of particle pushes of the wake calculation and (ii) the usage of large 3D time steps [5]. But in order to simulate a multi-TeV Plasma Wakefield Accelerator (PWFA) stage or the beam–electron cloud interaction over a real beam life span in a circular accelerator, a much faster and more efficient program is needed. For example, using 16 processors, it takes QuickPIC about five days to simulate LHC beam–electron cloud interaction during 500 turns, which corresponds only to about 44 ms in real time. However, this is still much shorter than the beam lifetime of 30 min [9]. In QuickPIC much of the computation is spent in the 2D field solver. Simply increasing the number of processors used in the simulation does not further reduce the computational cost, since the transposes in the FFT and the communication of field data among processors become more and more time-consuming for a fixed problem size. In order to overcome this limit, a novel algorithm, the pipelining algorithm, is described here and implemented into QuickPIC. The pipelining algorithm makes it possible to use up to 10^3 processors, and to accelerate the computational speed of the simulations by a corresponding factor.

2. The QuickPIC algorithm

Despite the differences in the 2D field solvers between the full and the basic quasi-static modes of QuickPIC, the program structure of both modes prior to the implementation of the pipelining algorithm remains the same. Because it is much

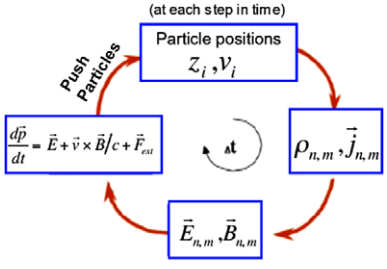

Fig. 1. Outer loop of the QuickPIC code, the electric (E) and magnetic (B) fields are calculated in the quasi-static field solver.

simpler, we use the basic quasi-static mode for the e-cloud problem to illustrate the original algorithm and the addition of the pipelining algorithm in QuickPIC. We will point out the differences between the pipelining algorithms for these two modes when they arise. In addition, while QuickPIC can model both laser and particle drivers for wakefield acceleration, only the particle beam driver case has been pipelined. Pipelining the laser driver case requires additional work on the laser advance and this will be done at a future date. However, the pipelining of the plasma solver does not change.

The QuickPIC simulation of the interaction between the beam and the electron cloud starts from the initialization of the beam and the electron cloud particles in a simulation box that travels at the initial speed of the beam (in the PWFA simulation, the simulation box travels at the speed of light). The positions and velocities of the beam particles are initialized with a Gaussian random number generator in order to account for beam size and emittance. The electron cloud particles are initialized with a uniform distribution. The beam–electron cloud interaction is calculated through a loop. For each step of the loop, the computational cycle is broken up into four parts as shown in Fig. 1 [10]. First, from the information of the beam particle positions and velocities, the current and charge densities on the grids are derived. Second, the electromagnetic field on the grids is determined by a series of 2D field solvers based on the quasi-static approximation, starting from the head of the beam and proceeding to the tail. In carrying out these field solves, the fields from the first 2D solve are used to push the cloud (or plasma) particles with a standard leap frog pusher and the new cloud density is passed to and used in the next 2D solve. Third, the field is then used to calculate the force on the beam particles. Fourth, the beam particles are pushed using a standard leap frog technique and the updated positions and velocities are deposited. The cycle repeats until the desired number of time steps is reached.

The domain decompositions of the beam and the electron cloud in QuickPIC are illustrated in Fig. 2 [11]. The beam is divided into equal space domain in the longitudinal or axial (z) direction. The 2D electromagnetic field calculation is carried out on a series of slabs in the transverse plane (x–y plane) with a fully parallelized spectral solvers using FFT. The electron cloud is decomposed evenly in the y direction on each slab.

3. Pipelining algorithm

In the original QuickPIC, the 2D field calculation is performed on successive slabs of the beam beginning at the head. For those beam particles located in the 2D slabs at the front for which the field calculation has been performed, all the information to continue to the 3D pusher has been obtained. Those beam particles can be pushed immediately while the 2D field calculation continues on the rest of the slabs, so there is no waiting for the 2D calculation to be completed on all slabs. This introduces the basic idea of the pipelining algorithm. We now describe the algorithm in detail.

3.1. Domain decomposition of pipelining algorithm

In the implementation of the pipelining algorithm, the processors are divided into N subgroups, instead of working as one big group, as in the original QuickPIC. The decompositions also need to be changed accordingly. Fig. 3 shows the domain decomposition used for the pipelining algorithm. The beam is evenly divided into N slabs in the longitudinal direction, and each slab is allocated to one subgroup. Within each subgroup, the beam is further decomposed evenly in slabs in the longitudinal direction, which are assigned to each processor in the subgroup. For the electromagnetic field calculation, the 2D slab is decomposed in the transverse y direction based on the number of processors in each subgroup. Each subgroup performs 2D field calculation on the 2D slabs only within its own 3D domain, instead of on the whole simulation box.

3.2. Advantage of pipelining algorithm

During the decompositions, the domain of each processor must contain at least one cell in the decomposed direction. The current implementation of QuickPIC uses 1D domain decompositions in the y direction for the 2D field solver, and in the z-

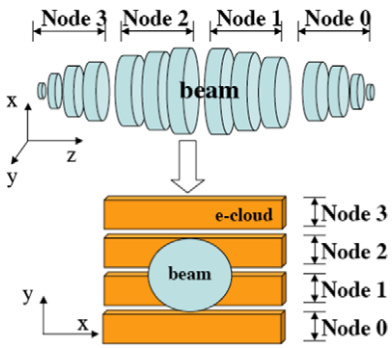

Fig. 2. Domain decomposition for the QuickPIC for the 2D and 3D calculations.

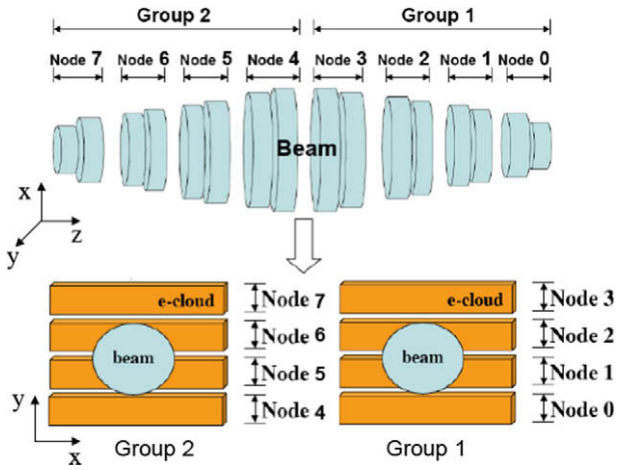

Fig. 3. Domain decomposition used in pipelining (two subgroups of processors and 4 processors per subgroup).

direction for the beam pusher. For the non-pipelined QuickPIC, the number of cells N_y in the y-direction puts an upper limit on the total number of processors that can be used. Moreover, if too many processors are used, the communication among all the processors, particularly during the 2D field calculation becomes so intensive that it detracts from the time saving of parallel computing with multiple processors. This effect reduces the upper limit to $f \times N_y$, where f is platform dependent and $0 < f < 1$.

The pipelining algorithm utilizes multiple subgroups of processors. Within each subgroup, the number of processors used is still limited by $f \times N_y$. However, since one can use multiple subgroups of processors, the total number of processors can be scaled up to $f \times N_y \times N$, where N is the number of subgroups, speeding up the simulation and/or improving the resolution of the simulation by a corresponding factor. Currently the 1D (in z) decomposition strategy in the 3D beam pusher in the pipelined version of QuickPIC practically limits the total number of processors to be N_z, i.e. the number of cells in the z-direction. However, this limit can be increased to $f \times N_y \times N_z$ with a 2D (in both y and z) decomposition strategy in the beam pusher (or $f \times N_x \times N_y \times N_z$ if 3D decomposition is used, and usually $0 < f < f < 1$) in conjunction with the pipelining algorithm. This upper limit is comparable to the theoretical upper bound ($N_y \times N_z$ for 2D decomposition or $N_x \times N_y \times N_z$ for 3D decomposition) on the number of processors that can be efficiently used for a fixed problem.

3.3. Pipelining algorithm

The pipelining algorithm starts with the parallel computing environment set up. All the processors to be used are divided into N subgroups. Each processor is labeled with the group ID representing the subgroup it belongs to, and a rank within its subgroup.

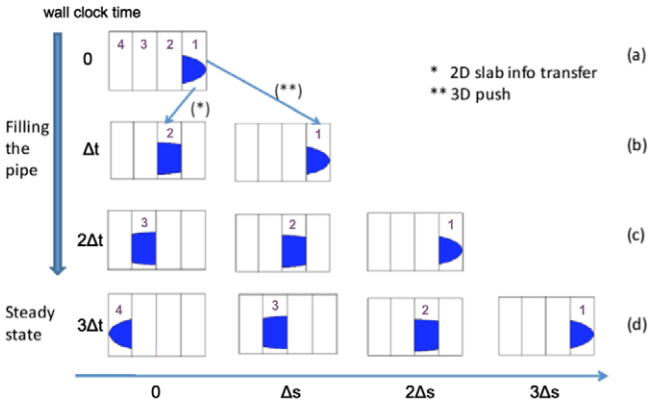

Fig. 4. Pipelining algorithm for QuickPIC. The beam moves from left to right. Four subgroups are used in this case. The boxes represent the domains for each subgroup, and are labeled with the group ID. The blue blocks are the parts of the Gaussian beam in the domain of the subgroup. The vertical labels are the wall clock times in units of execution time of a step in a subgroup; the horizontal labels are the beam propagation distances. The head and the tail of the beam at the same wall clock time are at different locations along the propagation trajectory. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The calculation starts with the beam initialization on all the processors of all the subgroups. After that, the first subgroup starts the 2D field calculation routine with the initialization of uniform electron cloud. The 2D slab is advanced through the part of the beam in the domain of the first subgroup. At this stage, the other subgroups are idling. After the 2D calculation is done, the first subgroup sends the results of the last slab to the second subgroup (Fig. 4(a)), which is necessary for the second subgroup to start its 2D calculation. Now that the second subgroup is performing the 2D field calculation, the first subgroup moves on to push the part of the beam within its domain and then deposits the beam charge density. Since some of the beam particles may be pushed out of the domain of a subgroup, each subgroup communicates with the adjacent subgroup(s) to transfer the beam particles into the new domain if necessary. The first subgroup is then ready for the computation of the next time step. Meanwhile, the second subgroup finishes the 2D field calculation and starts the 3D beam push. Then the third subgroup begins the calculation of 2D field calculation of the first time step (Fig. 4(b and c)). All the subgroups repeat the same procedure until the desired number of time steps are achieved.

In summary, if N subgroups are used, it takes about $(N-1)$ time steps to fill up the pipe. In the full pipe, the K^{th} subgroup is always approximately a half time step ahead of the $(K+1)^{th}$ subgroup (Fig. 4(d)). The number of particles of e-cloud (or plasma) is the same for all the slabs, so e-cloud (or plasma) update for each slab takes roughly the same time (there is indeed a load balance issue concerning the number of particles in each processor in a particular subgroup, but this would not affect the pipeline operation). The number of beam particles does vary significantly among subgroups, and the beam update only consists of a very small fraction of the total computation time, and pipeline stall has never occurred.

3.4. Inter-subgroup communication

There are several situations that involve communication among the subgroups: (a) in the 2D field calculation, the last electron slab needs to be passed to the next subgroup, so that the next subgroup can start the 2D field calculation from the received slab; (b) after the 3D beam push, beam particles that move out of the original domain, due to synchrotron oscillation or acceleration/deceleration from the wake, need to be sent to the other subgroups.

During the pipelining process, subgroup K is always approximately a half step ahead of subgroup $(K+1)$. If the communication occurs backward in the direction from head to tail, i.e. from subgroup K to subgroup $(K+1)$, after subgroup K sends out the information, the information can be stored in a buffer and subgroup K can move on to the next part of computation; subgroup $(K+1)$ picks up the information whenever it is ready to receive it. Obviously, backward communication does not interrupt the computational flow and preserves the time saving benefit of the pipelining algorithm.

The other communication direction is forward, i.e. from tail to head. The information is sent from subgroup $(K+1)$ to subgroup K. Since subgroup K is ahead of subgroup $(K+1)$, when subgroup K is ready to receive, it has to stop and wait for subgroup $(K+1)$ to be ready to send. This basically synchronizes the computational progress on subgroups K and $K+1$. The waiting time involved could diminish the time saving benefit of the pipelining algorithm.

Situation (a) mentioned above only involves backward communication. For situation (b), and in the electron cloud case, the simulation box is chosen to move at the initial speed of the beam along the center trajectory. Because of the presence of accelerating RF fields in accelerators beam particles gain or lose energy depending on their phase with respect to the RF wave. Particles with higher momentum tend to travel along a larger orbit than particles with the design momentum and hence slip backward relative to the particles moving with the design momentum. Conversely, particles with lower momentum travel along a smaller orbit and may move faster than the simulation box. It is thus possible for beam particles to move out of their original domain during the particle push, through either boundary in the longitudinal direction. Both backward and forward inter-subgroup communications are required to reallocate those particles to the new domain. Note that in the case of modeling the beam propagation in a PWFA or the laser propagation in a LWFA, the speed of the moving simulation box is chosen as the speed of light, therefore one only needs to consider the backward communication, which can be easily handled in the pipelining algorithm.

In order to solve the problem induced by the forward communication, the timing of the communication is changed. For beam–electron cloud instability simulations with QuickPIC, the 3D time step is chosen to resolve the betatron oscillation wavelength (λ_β), and typically corresponds to $\lambda_\beta/30$. In most cases, the synchrotron oscillation frequency is much smaller than the betatron frequency, so that

$$(v_{th} + \omega_s L_B)\Delta s < \frac{\Delta\xi}{2} \tag{3}$$

is satisfied. Here v_{th} is the thermal velocity of the beam, ω_s is the frequency of the synchrotron oscillation, L_B is the length of the simulation box in z, Δs is the 3D time step, and $\Delta\xi$ is the cell size in longitudinal (z) direction. Eq. (3) guarantees that only the particles within the $\Delta\ \xi$ distance from the boundary of the group domain can possibly travel to the domain of another subgroup, and the particles cannot move to a subgroup that is not adjacent to its current subgroup since for each 3D time step, the particles displacement in z-direction is less than $\Delta\xi/2$. So, if there are particles from subgroup $(K+1)$ that need to be sent to subgroup K, there is no need for subgroup K to wait until the 3D beam pusher is completely done on all the particles in the domain of subgroup $K+1$. Instead, the beam particles are sorted, and all the particles to be sent can be gathered after the 3D push of the particles within $\Delta\xi$ distance from the boundary between subgroup K and subgroup $(K+1)$. After the communication, subgroup K moves on to the next step 2D field calculation and subgroup $K+1$ continues the rest of the beam push. This keeps the computational flow of the pipelining algorithm and effectively limits the waiting time.

4. Performance of the pipelining algorithm

4.1. Fidelity of the pipelining algorithm

The goal of pipelining algorithm is to speed up the simulation without losing the accuracy of the original QuickPIC code. To check that the pipelined version of QuickPIC reproduced the results of the original version, long-term simulation of 12,000, 3D time steps were performed. These simulations represent 300 turns of LHC beam propagation with the parameters of Table 1.

As shown in Fig. 5, the spot size obtained with and without the pipelining algorithm overlap perfectly in both transverse directions, and the difference is less than 0.5%. Both show the onset of electron cloud instability in the vertical plane at approximately 280 turns. This indicates that the pipelining algorithm preserves the physics modeled in the original QuickPIC code.

Table 1
Simulation parameters for LHC at CERN.

Horizontal spot size (mm)	0.884
Vertical spot size (mm)	0.884
Bunch length (m)	0.115
Horizontal box size (mm)	18
Vertical box size (mm)	18
Bunch population	1.1 × 1011
Momentum spread	4.68 × 10-4
Beam momentum (GeV/C)	4.796 × 108
Circumference (km)	26.659
Horizontal betatron tune	64.28
Vertical betatron tune	59.31
Synchrotron tune	0.0059
Horizontal vertical chromaticity	2, 2
Electron cloud density (cm^{-3})	6 × 105
Total cell number	128 × 128 × 128
Total beam particles	128 × 128 × 256
e-Cloud particles per cell	2 × 2

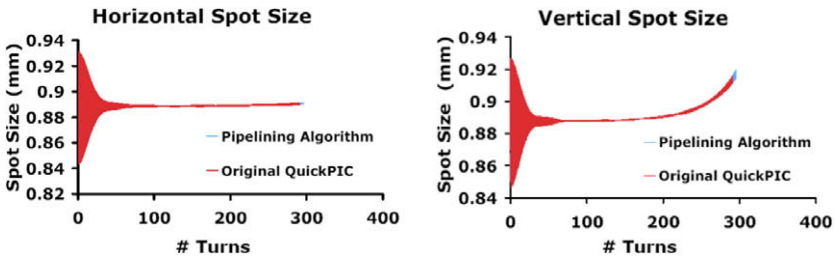

Fig. 5. Horizontal (left graph) and vertical (right graph) beam spot sizes obtained with and without pipelining using QuickPIC in the basic quasi-static mode. Perfect overlap is observed.

Table 2
Simulation parameters for the computational efficiency test of full quasi-static approximation (plasma wakefield calculation for an electron drive beam).

Horizontal spot size (μm)	7
Vertical spot size (μm)	7
Bunch length (μm)	45
Bunch population	1.8×10^{10}
Relativistic factor	55800
Plasma length (cm)	15.04
Plasma density (cm^{-3})	2.0×10^{16}
Horizontal box size (μm)	600
Vertical box size (μm)	600
Box length (μm)	500
Total cell number	256 × 256 × 1024
Total beam particles	128 × 128 × 512
Plasma particles per cell	2 × 2
3D time step ($1/\omega_p$)	22.5

Test of the pipelining algorithm in the full quasi-static mode of QuickPIC also confirms the fidelity of the algorithm. Again, a typical PWFA simulation was carried out with and without the pipelining algorithm. This simulation models the propagation of an intense electron beam in an underdense plasma with the parameters specified in Table 2. Eight processors were used in the baseline simulation in which pipelining was turned off. Since the simulation box contains 1024 grid cells in the longitudinal direction, for the pipelining algorithm we were able to use 128 subgroups, each with 8 processors. The results from both the baseline and pipelined simulations show excellent agreement, as seen in Fig. 6, with the difference in beam density less than 0.3%.

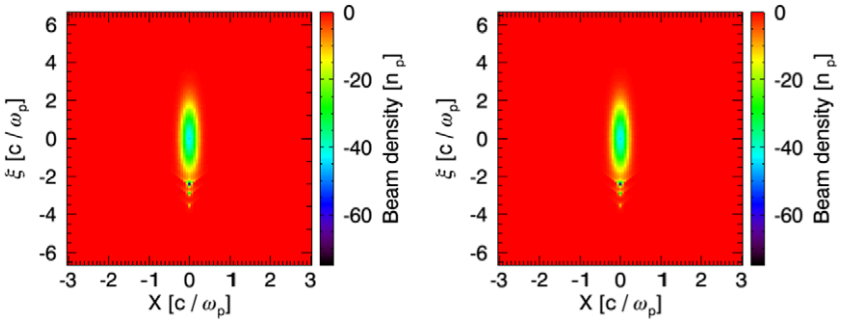

Fig. 6. Electron beam densities in PWFA simulations at the 150th time step, with the original QuickPIC (left figure) and with pipelining algorithm (right figure). The beam moves downward in these figures.

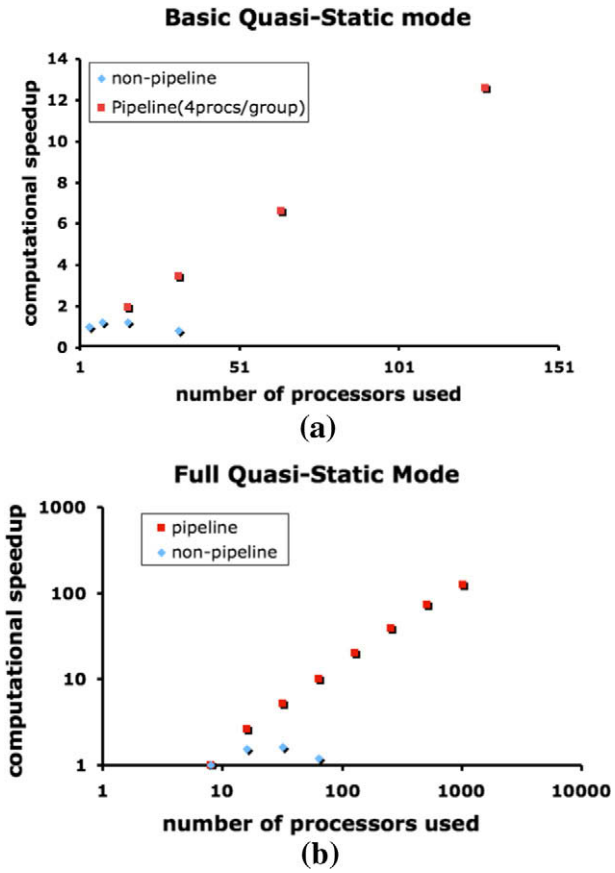

Fig. 7. Comparison of the computation speedup between pipelining and non-pipelining versions of the QuickPIC code. (a) Speedup for QuickPIC with basic quasi-static approximation (the vertical axis shows the speedup over the original QuickPIC using 4 processors). (b) Speedup for QuickPIC with full quasi-static approximation, normalized to the simulation time with 1 subgroup with 8 processors.

Table 3
Simulation parameters for the computational efficiency test of basic quasi-static approximation (modeling of beam and electron cloud interaction).

Horizontal and vertical spot size (mm)	0.884, 0.884
Bunch length (mm)	115
Bunch population	11×10^{10}
Horizontal emittance	21.70
Vertical emittance	20.02
Momentum spread	4.68×10^{-4}
Relativistic factor	480
Radius (km)	4.24291
Horizontal and vertical betatron tune	64.28, 59.31
Synchrotron tune	0.0059
Electron cloud density (cm^{-3})	6×10^5
Chromaticity x and y	2, 2
Phase slip factor	3.47×10^{-4}
Horizontal and vertical box size (mm)	18, 18
Box length (mm)	2000
Total cell number	$64 \times 64 \times 1024$
Total beam particles	$64 \times 64 \times 2048$
e-Cloud particles per cell	2×2
3D time step ($1/\omega_p$)	1.90059894561
Total simulated 3D time steps	2000

4.2. Efficiency of the pipelining algorithm

Fig. 7(a) and (b) shows the comparison of the computational speedup between the pipelined and the original version of QuickPIC for both the basic and full quasi-static modes respectively.

Simulations with the basic quasi-static mode version of QuickPIC for an electron cloud problem (see Fig. 7(a)) is performed on computer clusters with a dual Intel P4 3.0 GHz processor on each node and interconnected with Ethernet and Myrinet [12]. Both the pipelined and the original versions simulate 2000, 3D time steps using the same simulation parameters, as presented in Table 3. The number of grids in three-dimensions is set to 64:64:1024 in the x, y and z-direction, respectively. The speedup of 4 processors running the original QuickPIC is normalized to 1, and is used as the baseline in Fig. 7(a). As seen in Fig. 7(a), with the original QuickPIC, the computational speedup increases as the number of processors is first increased, but this trend reverses when the number of processors reaches 16.

With the pipelining algorithm, on the other hand, when 16 processors are used, there is a 60% increase in speed compared to the case of the original QuickPIC. Here subgroups contain 4 processors. For 2D calculation, instead of using 16 processors for each slab, only 4 processors are used. The communication among 4 processors is much more time efficient compared to that among 16 processors. Furthermore, as we keep increasing the number of subgroups, which proportionally increases the total number of processors, the computation speedup continues to increase, and is proportional to the total number of processors used for the pipelining code, up to 128 processors for this particular problem.

A similar trend of efficiency improvement is observed in the QuickPIC code in the full quasi-static mode for PWFA simulations (see Fig. 7(b)). A series of simulations are conducted with 8 processors in each subgroup and the number of subgroups varies from 1 to 128. The simulations are carried out on computer nodes with 2.6 GHz dual-core AMD Opteron processor. All the computer nodes are connected to a high performance, low-latency SeaStar2 network [13]. The simulation parameters are presented in Table 2 and the speedup of each simulation is plotted in Fig. 7(b) with the performance of the baseline simulation (1 subgroup with 8 processors) normalized to 1. The overall speedup relative to the non-pipelined code is not only a function of number of subgroups (with the number of processors in each subgroup held fixed) but also a function of the number of time steps; because there is a pipeline filling time and emptying time. When the number of time steps is much greater than the number of subgroups, the speedup approaches the number of subgroups. In Fig. 7(b), the speedup is calculated from the computational time of one 3D step for a particular subgroup. This would be the asymptotic value for the speedup in a long-term simulation, for which the pipelining filling time and the emptying time can be ignored. More than 10^3 processors are used in the 128 subgroups simulation and the speedup of the simulation is increased by a proportional factor. As a comparison, the baseline simulation is run with all processors in a single subgroup and the performance (blue dots) rolls over quickly at fewer than 100 processors, as seen in Fig. 7(b).

5. Conclusion

We have described and implemented a novel parallel pipelining algorithm that allows the quasi-static code QuickPIC to scale to well over 10^3 processors. The pipelining algorithm therefore can increase the effective speed of the QuickPIC code by two to three orders of magnitude. The parallel scalability of the traditional QuickPIC code is limited by the number of processors that can be used for 2D Poisson solvers, which is currently limited by the cost of the communication in the FFT. The pipelining algorithm uses multiple subgroups of processors and optimizes the computing job allocation on them. The

computing speed is shown to be proportionally increased as more processors are used for the simulation. The pipelining algorithm reproduces the results obtain with the original version of the QuickPIC code, and therefore preserves the physics included in the modeling while greatly reducing the simulation time.

Acknowledgments

Computational time and support from the USC High-Performance-Computing and communications Center and Franklin at NERSC are gratefully acknowledged. This work was supported by the US Department of Energy through Grants DE-FG02-92ER40745, DE-FG02-06ER54886.

References

[1] G. Arduini et al., Beam Observations with Electron Cloud in the CERN PS and SPS Complex, ECLOUD'04, Napa, California, 2004.
[2] H. Fukuma et al., Observation of Vertical Beam Blow-Up in KEKB Low Energy Ring, EPAC'00, Vienna, June 2000.
[3] K. Harkay et al., Electron Cloud Observations: A Retrospective, ECLOUD'04, Napa, California, 2004.
[4] I. Blumenfeld et al, Energy doubling of 42 GeV electrons in a metre-scale plasma wakefield accelerator, Nature 445 (2007) 741–744.
[5] C. Huang et al, QUICKPIC: a highly efficient particle-in-cell code for modeling wakefield acceleration in plasma, Journal of Computational Physics (2006) 658–679.
[6] G. Rumolo et al, Electron cloud effects on beam evolution in a circular accelerator, Physical Review Special Topics-Accelerators and Beams 6 (2003) 081002.
[7] C. Huang et. al, A parallel particle-in-cell code for efficiently modeling plasma wakefield acceleration: QuickPIC,' in: Proceeding of the Applied Computational Electromagnetics Society Conference, Monterey, CA, March 2002.
[8] A. Ghalam et al, Three-dimensional continuous modeling of beam–electron cloud interaction: continuous modeling of beam–electron cloud interaction: comparison with analytic models and predictions for the present and future circular machines, Physics of Plasmas 13 (2006) 056710.
[9] B. Feng et al., Enhancing plasma wakefield and e-cloud simulation performance using a pipelining algorithm, in: Proceedings of the 12th Workshop Advanced Accelerator Concepts.
[10] A. Ghalam et al., 3D parallel simulation of continuous beam-cloud interactions, in: Proceedings of ECLOUD04 Workshop, Napa, California, 2004.
[11] A. Ghalam, High Fidelity Three-Dimensional Models of Beam–Electron Cloud Interactions in Circular Accelerators, PhD Thesis, University of Southern California, 2006.
[12] Linux Computing Resource (uschpc), <http://www.usc.edu/hpcc/systems/use-l-0>.
[13] Franklin-Cray XT4, <http://www.nersc.gov/nusers/systems/franklin/>.