LONG TIME ELECTRON CLOUD INSTABILITY SIMULATION USING QUICKPIC WITH PIPELINING ALGORITHM*

B. Feng#, P. Muggli, T. Katsouleas, USC, Los Angeles, CA, U.S.A.
V. Decyk, C. Huang, W. Mori, UCLA, Los Angeles, U.S.A.

Abstract

We proposed a novel algorithm, which uses pipelining to reduce the simulation time for beam-electron cloud interaction. In the pipelining algorithm the processors are divided into subgroups, and during the simulation different groups will be on consecutive time steps. The pipelining algorithm is applied to the fully parallelized Particle-In-Cell (PIC) code QuickPIC to overcome the limit of the number of processors that can be used at each time step. With the new algorithm, the accuracy of the simulation is preserved; and the speed of the simulation is improved by a factor proportional to the number of processors available. The long term beam evolution results for the CERN-LHC using the QuickPIC with pipelining algorithm are presented.

PIPELINING ALGORITHM

QuickPIC is a 3D PIC (particle-in-cell) code using a quasi-static approximation [1]. QuickPIC has been adapted to model the beam dynamics in circular particle accelerators including the effects of persistent electron cloud [2]. Both the 3D outer layer and the 2D inner layer of the code have been fully parallelized. However, in order to model the beam propagation of the timescale that is of the same order as the real beam lifetime with high resolution, a faster and more efficient simulation code is needed. Pipelining algorithm is used to improve the efficiency of the parallel computing of QuickPIC.

The pipelining algorithm begins with the initialization of the parallel computing environment. All the processors to be used are divided into subgroups instead of keeping them as one big group in the original QuickPIC; each processor is assigned with a rank in the subgroup and also a group id representing which group this processor belongs to. Then the calculation starts with the beam initialization on all the processors of all the subgroups simultaneously. Each processor is in charge of the calculation of a particular part of the beam. The first group then starts the 2D field calculation routine. The 2D field calculation uses a quasi-static approximation. It is done by solving Poisson equation on slabs in the x-y plane, and advancing the 2D slab through the beam in the longitudinal (z) direction. For the first group, the 2D slab will be advanced up to the end of the domain of the group. The information of the last slab from the first group is sent to the second group (see Fig. 1a). Upon receiving the information, the second group starts the 2D field calculation and continues advancing the slab through its domain. In the mean time, the first group moves on to push the beam particles with the forces on the beam obtained from the 2D field calculation. After the beam particles are pushed the beam charge density is deposited, and the first groups communicate with the other groups to put the beam particles that are pushed out of the original domain into the new domain. After this, the first group can start the computation for the next time step (see Fig. 1b). At this time, the second group finishes the 2D field calculation, sends the last slab to the third group, and starts to push the beam. The third group receives the 2D slab and begins the field calculation (see Fig. 1c). In this way, all the groups repeat the same procedure until the predetermined number of time steps is reached. One of the characteristic of the pipelining algorithm is that during the computation, the Nth group is always approximately half a time step ahead of the (N+1)th group (see Fig. 1d).

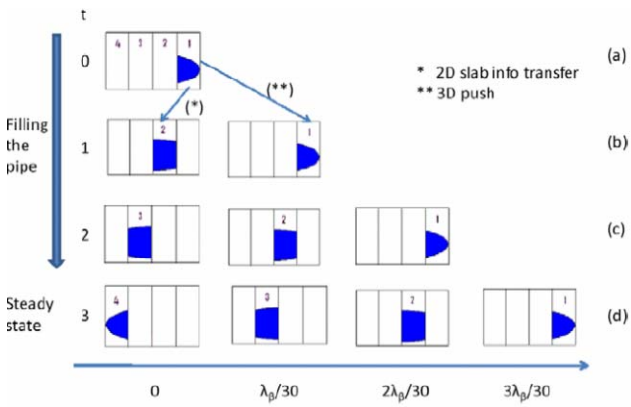

Figure 1: Algorithm of pipelining with QuickPIC.

RESULTS

Computational Time

For the original QuickPIC code, the number of processors that can be used is limited by the number of cell decomposition in the y direction. This puts a limitation on the simulation speed. With pipelining, the processors work in subgroups. Although the number of processors in each subgroup is under the same limitation as before, the total number of processors is increased because multiple subgroups are used. With more processors used, the computational time is further reduced.

In order to compare the computational time, the number of cells for x, y and z directions are set to be 64:64:1024 respectively. As shown in Figure 2, with the original QuickPIC code, the computing time first decreases as the number of processors is increased. But this trend does not continue once the number of processors used is increased to 32. In this case, the computing time dramatically increases because of the increasing cost of

*Work supported by NSF, DoE and SciDAC
#bfeng@usc.edu

communication among the 32 processors during the 2D calculations. The communication cost increase exceeds the time saving from using more processors. In the case of pipelining, the processors used for the calculation are divided into subgroups with 4 processors in each group. The continuous computational time saving trend is observed up to 128 processors. Since the communication among the processors during the 2D calculation only occurs within the subgroups, which contains 4 processors independent of the total number of processors used, the communication cost does not increase as fast as the total number of processors increases, so that the time saving trend continues. Due to the same time saving of the communication in the 2D calculation, when using the same number of processors, for example, as shown in Figure2, 32 processors, pipelining algorithm saves about 30% time when compared to non-pipelining code.

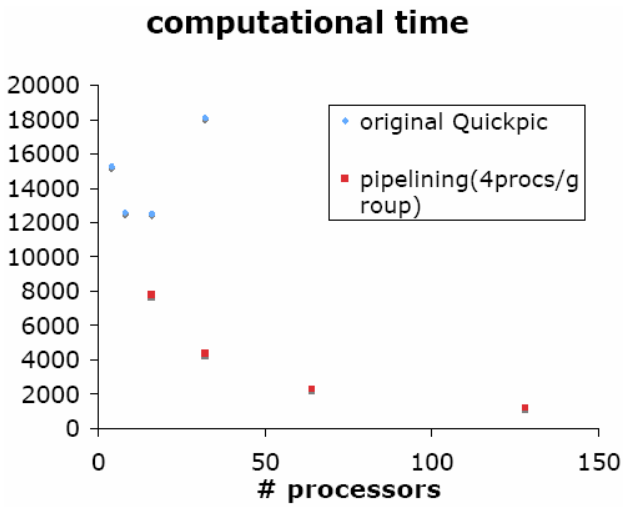

Figure 2: Computational time comparison between pipelining and original QuickPIC.

Benchmark against the Original QuickPIC Code

The results of the pipelining algorithm are benchmarked against the original QuickPIC code to make sure the accuracy of the simulation is not sacrificed. This is demonstrated by the comparison of the long term simulation results of the LHC ring using both version of the QuikPIC code with and without pipelining. The simulation parameters are listed in Table 1.

Table 1: Simulation Parameter for LHC

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
Proton Bunch Population	1.1×10^{11}
Momentum Spread	4.68×10^{-4}
Beam Momentum (GeV/c)	4.796×10^{8}
Ring 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})	6×10^{5}

Long term simulation of the beam evolution over about 290 turns is performed. The results of the beam spot size from QuickPIC with and without pipelining are compared. (See Fig. 3).

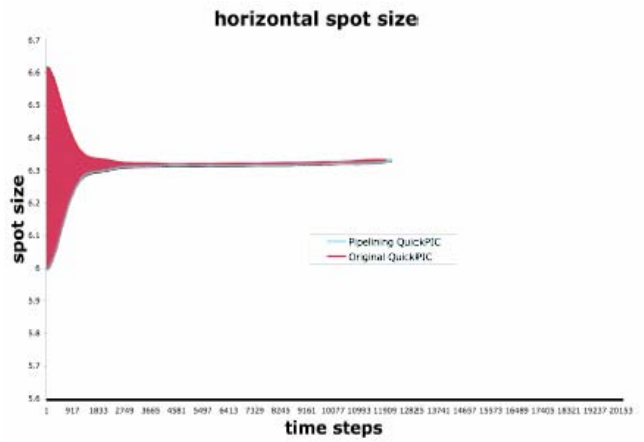

Figure 3 (a): Horizontal beam size obtained with pipelining algorithm (blue curve) and original QuickPIC code (red curve).

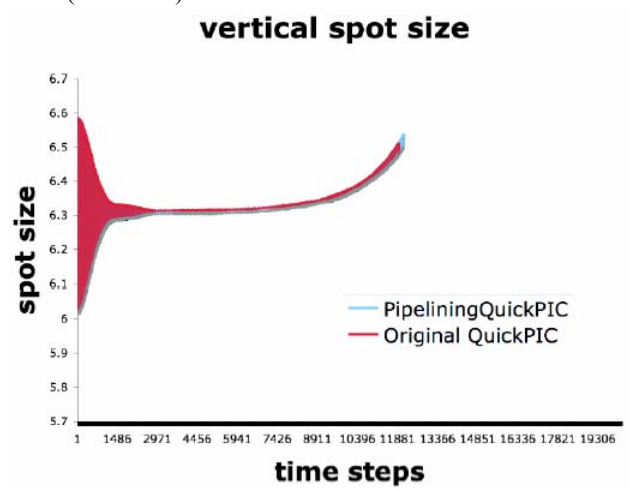

Figure 3 (b): Vertical beam size obtained with pipelining algorithm (blue curve) and original QuickPIC code (red curve).

Figure 3 shows that the results from pipelining algorithm exactly overlap the results of the original QuickPIC for both horizontal and vertical beam sizes. The pipelining algorithm faithfully reproduces the results of the original QuickPIC with the 30% time saving shown in Fig. 2.

SUMMARY

In this paper, the implementation of pipelining algorithm into the QuickPIC simulation model is presented, and the computational efficiency and the fidelity of the pipelining algorithm are tested. The pipelining algorithm is able to overcome the limit of the number of the total processors that can be used for a given simulation. With more processors being used and more efficient 2D field calculations due to reduced communication on each simulation slab, pipelining has demonstrated a great potential for time saving. In addition, there is no accuracy of the results is sacrificed compared to the original QuickPIC. QuickPIC with the pipelining algorithm will be used to simulate with high fidelity the influence of electron cloud on existing and future damping rings and circular accelerators.

REFERENCES

[1] C. Huang, et, al. 'QUICKPIC: A highly efficient partical-in-cell code for modelling wakefield acceleration in plasma', Journal of Computational Physics, 658-679, 2006.

[2] A. Z. Ghalam, et, al. 'Simulation of Electron-Cloud Instability in Circular Accelerators using Plasma Models', Proceedings of the 2003 Particle Acceleration Conference