社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2009－ARC－182 (1)
2009－HPC－119 (1)
2009／2／26

Performance Evaluation of OhHelp'ed 3D Particle-in-Cell Simulation

Hiroshi Nakashima,[†1] Yohei Miyake,[†2] Hideyuki Usui[†2]
and Yoshiharu Omura[†2]

We proposed an efficient and scalable load balancing method named *OhHelp* for Particle-in-Cell (PIC) simulations. This method simply and equally partitions the space domain, in which charged particles are distributed nonuniformly in general, so that each computation node works on each partitioned *primary* subdomain. Load imbalance problem caused by the nonuniformity of the particle distribution is solved by making every but one node also work on another subdomain where particles densely populate as its *secondary* subdomain together with a part of particles in it. We applied the OhHelp method to a production level full-3D PIC simulator for space plasma and evaluated its performance on our T2K Open Supercomputer. As a result, we confirmed our simulator is not only efficient showing 150–190 speedup with 256 CPU cores compared to the sequential execution of a reference simulator, but also scalable in terms of both the space domain size and the number of particles as the break down of execution times evidences.

1. Introduction

Particle-in-Cell (PIC) simulations are indispensable for theoretical and practical research of high-energy physics, space plasma physics, cloud modeling, combustion engineering, and so on[5]. Since a PIC simulation works on a huge number of particles residing in a space domain represented by a large number of grid points, its parallelization is essentially required.

However, a simple *particle decomposition* (e.g., 4)), which gives a statically partitioned subset of particles to a computation node while makes the space domain shared by all nodes, is not scalable because the domain size cannot be enlarged proportionally to the number of nodes. On the other hand, a simple *domain decomposition* (e.g., 1)), which assigns a statically partitioned subdomain to a node which also works on particles incidentally visiting the subdomain, is also unscalable because a subdomain may have all particles when they are concentrated in a small region. More sophisticated domain decomposition, which dynamically shift subdomain boundaries to keep the uniformity of particle populations among subdomains by, for example, ORB[6], is not scalable again in the concnetrated situation because a subdomain with sparsely populated particles can be almost as large as the whole domain.

Therefore, we proposed a new domain-decomposed PIC simulation method named *OhHelp*[3] aiming at the scalability in terms of both space domain size and particle population. This method simply and equally partitions the space domain so that each partitioned subdomain is assigned to each computation node for scalable simulation with respect to the domain size. Load balancing and thus scalability in terms of the particle populations are accomplished by making each node *help* another node having a densely populated subdomain, a part of particles in which is deputed to the helper node together with replicated field data associated to grid points in the subdomain.

In our first report[3], we exhibited the efficiency of a prototype simulator with OhHelp but the good evaluation result was not very confident due to the followings; the simulator was two-dimensional rather than three-dimensional for production level simulation; the implementation and evaluation were done with a large scale shared memory supercomputer Fujitsu PrimePower HPC2500 instead of distributed memory systems being our real target; and the performance was compared with a particle-decomposed simulator which is automatically parallelized possibly resulting in a certain underestimation. Therefore in this paper, we show our evaluation result of a full-3D production level space plasma PIC simulator implemented on our T2K Open Supercomputer[2]. We also show the performance of a particle-decomposed and manually parallelized simulator as the ref-

†1 ACCMS, Kyoto University
†2 RISH, Kyoto University

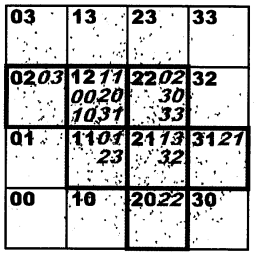


Fig. 1 Space Domain Partitioning

erence, execution time breakdown of our and reference simulators, and fundamental performance numbers to predict the performance according to future extension of our simulator in terms of functionality and/or scale.

2. OhHelp Overview

As shown in **Fig. 1**, OhHelp simply partitions the simulated space domain into equal-size subdomains and assigns each subdomain to each computation node as its *primary subdomain*. In the figure, non-italic numbers are the identifiers of nodes and also those of primary subdomains assigned to them. Each node is responsible for its primary subdomain, and also all the particles in it if the numbers of those *primary particles* in subdomains are balanced well, or more specifically, if the number of particles P_d in a subdomain d satisfies the following inequality for all d,

$$P_d \leq (P/N)(1+\alpha) \equiv P_{lim} \qquad (1)$$

where P is the total number of particles, N is the number of nodes, and α is the tolerance factor greater than 0. We refer to the simulation phases in this fortunate situation as those in *primary mode*.

Otherwise, i.e., if the inequality (1) is not satisfied for some subdomain d as shown in Fig. 1, the simulation is performed in *secondary mode*. In this mode, every node, except for one node (12 in the figure), is responsible for a *secondary subdomain* having particles more than the average, in addition to its primary one. For example, the subdomain 22 has *helper* nodes 02, 30 and 33 shown in italic letters in Fig. 1. The particles in a heavily loaded subdomain are also distributed to its helper nodes as their *secondary particles* so that each node n has Q_n particles in total, which reside in the primary or secondary subdomain of n, satisfying the following inequality for balancing similar to (1) for all n.

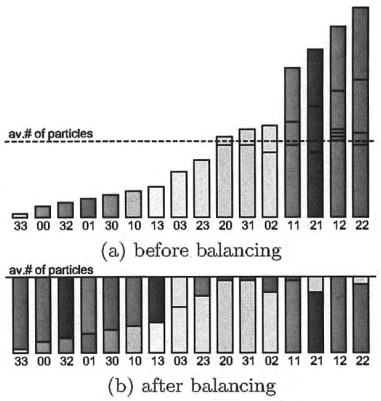


Fig. 2 Subdomain Assignment with Perfect Balancing of Number of Particles

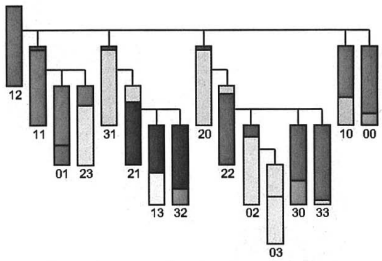


Fig. 3 Helper-Helpand Tree for Balancing Result in Fig. 2(b)

$$Q_n \leq (P/N)(1+\alpha) = P_{lim} \qquad (2)$$

Note that since all but one nodes have secondary subdomains, a node whose primary subdomain is heavily loaded, e.g., node 22, is not only helped by other nodes but also helps another node 20, as the balancing algorithm orders as shown in Fig. 2. That is, when it is found that the inequality (1)/(2) is not satisfied in primary/secondary mode, the balancer establishes the *helpand-helper family* for the helpand node 22 with helper nodes 02, 30 and 33 giving them secondary particles of $P/N - P_m$ to make $Q_m = P/N$ for each $m \in H(22) = \{02, 30, 33\}$. Then, since this results in that the number of primary particles remaining in the node 22, i.e., $P_{22} - \sum_{m \in H(22)}(P/N - P_m)$ becomes less than P/N, the balancer makes the node 22 help the other node 20 to result in $Q_{22} = P/N$, and then makes the node 20 help 12 so that $Q_{20} = P/N$, while the node 12 does not help any other node as the *root* of the

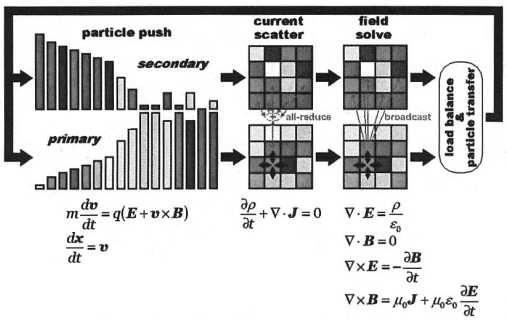


Fig. 4 3D PIC Simulator with OhHelp

helpand-helper tree shown in **Fig. 3**.

The tree is traversed in bottom-up (leaf-to-root) manner to examine whether the inequality (2) is satisfiable in secondary mode. For example, if P_{01} or P_{23} becomes larger than P_{lim}, we cannot keep good balancing without reestablishing the tree to give a helper to one of them. Otherwise, but if $P_{11} > 3P_{lim} - (P_{01} + P23) \equiv P^0_{11}$, the family cannot sustain particles in the subdomain 11, 01 and 23 and thus we fail the examination. Otherwise, after similar examinations for the families rooted by 31 and 20 and the leaf nodes 10 and 00, we examine whether;

$$P_{12} \leq 6P_{lim} - (P^0_{11}+P^0_{31}+P^0_{20}+P_{10}+P_{00})$$

holds and if so the helpand-helper configuration is still capable to keep good balancing.

Then a top-down traverse of the tree takes place to redistribute particles in the subdomain primary to each helpand among the helpand-helper family. We need this redistribution for the particles just coming to the subdomain crossing its boundaries, and for those which cannot be sustained by helpers because primary particles of them and/or their descendants become too many.

For detailed algorithm of balancing, sustainability examination of helpand-helper configuration and particle redistribution in helpand-helper families, see 3).

3. 3D PIC Simulator with OhHelp

We applied the OhHelp method to our production level full-3D PIC simulator for space plasma[4]. As the implementation outline in **Fig. 4** shows, the main loop of the simulator consists of the following phases.

particle pushing: Each node accelerates its primary and secondary particles by Lorentz force law referring electromagnetic field data $\boldsymbol{E}$ and $\boldsymbol{B}$ associated to the grid points in its primary and secondary subdomains, and then moves particles according to their updated velocities. Particle movements crossing subdomain boundaries will be taken care of by the last phase but the number of crossing is counted to build the histogram of the particle amount in each subdomain. This phase is executed locally.

current scattering: Each node locally calculates the contributions of the movement of its primary and secondary particles to the current density $\boldsymbol{J}$ at the grid points in its primary and secondary subdomains. Then an all-reduce communication is performed in each helpand-helper family to sum up the current density in the subdomain of the family. Finally, the boundary values of $\boldsymbol{J}$ are exchanged between primary subdomains and then broadcasted from the helpand to its helpers.

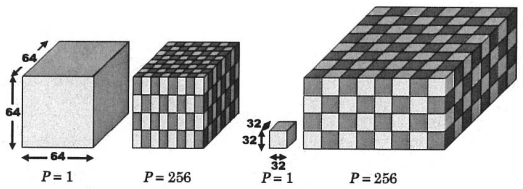
field solving: Each node locally updates the value of $\boldsymbol{E}$ and $\boldsymbol{B}$ at the grid points in its primary and secondary subdomains using leapfrog method to solve Maxwell's equations. Then the boundary values of $\boldsymbol{E}$ and $\boldsymbol{B}$ are exchanged between primary subdomains and then broadcasted from the helpand to its helpers.

load balancing and particle transferring: The histograms of particle residence are exchanged by an all-to-all communication and an all-gather one. The former acquaints each node with the number and the sources of boundary crossing particles into its primary region. The latter is to make all nodes share the number of primary and secondary particles in each node, with which the balancing algorithm discussed in Section 2 is executed in all nodes. If it is necessary to change the secondary subdomain assignments, each node broadcasts its electromagnetic field values to its helpers after new helpand-helper families are established. Then, after each node makes the schedule of particle transfer for its primary region and notifies involved nodes of it, particles are transferred among nodes.

4. Performance Evaluation

4.1 Evaluation Setup

We coded the PIC simulator discussed in the previous section using Fortran 90 for the main part of the simulation and C for OhHelp load



(a) strong scaling (b) weak scaling

Fig. 5 Strong and Weak Scaling

balancer, with MPI 2.0 communication library. Then the program was executed on our T2K Open Supercomputer[2], which has 416 Fujitsu's HX600 server nodes each of which is equipped with four quad-core Opteron 8356 processors, 32 GB shared memory and four-way Infiniband 4x DDR links for full-bisection and 8 GB/s per-node high-throughput interconnect. We used up to 16 nodes or 256 cores by mapping each MPI process onto a core.

We evaluated our simulator with two types of scaling, namely *strong scaling* and *weak scaling* (**Fig. 5**). For the former, we fixed the space domain size to 64^3 grid points and the number of particles to 2^{27}. Then the domain is decomposed for $2^x \times 2^y \times 2^z$ process arrays where $2^x : 2^y : 2^z$ is either of $1:1:1$, $2:1:1$ or $2:2:1$. For weak scaling, on the other hand, we fixed the subdomain size to 32^3 and the average number of particles in a subdomain to 2^{23} with the shapes of process arrays same as the strong case. Thus the total domain size of 256-process execution is $256 \times 256 \times 128$ and the number of particles is 2^{31}.

With both scaling types, we examined the performance with two extreme initial particle distributions, balanced and unbalanced. In the balanced case, particles are uniformly distributed in the space domain and have a constant initial velocity toward x-axis so that particles steadily travel to keep simulated in primary mode with perfect load balancing. In the unbalanced case, particles are also uniformly distributed and have the constant initial velocity but reside in a small cubic region of 32^3 at the bottom-south-west corner of the cuboid domain. Therefore, we have an extreme imbalance of particle amounts especially in the weak scaling experiment in which particles reside in at most two subdomains.

We performed simulations for 6400 time steps, in which a particle travels 32 grid points,

Table 1 Performance of Strong Scaling

#proc	part.decomp.	balanced	unbalanced
1	2.88/ 1.00	2.55/ 0.89	2.55/ 0.89
2	5.61/ 1.95	4.74/ 1.65	4.31/ 1.50
4	11.27/ 3.92	9.98/ 3.47	8.57/ 2.98
8	20.45/ 7.11	18.31/ 6.36	15.70/ 5.46
16	36.21/ 12.58	34.30/ 11.92	30.54/ 10.61
32	59.32/ 20.62	68.94/ 23.96	60.42/ 21.00
64	83.63/ 29.06	137.98/ 47.95	121.42/ 42.19
128	98.88/ 34.36	269.20/ 93.55	238.67/ 82.94
256	101.42/ 35.25	519.04/180.38	457.49/158.98

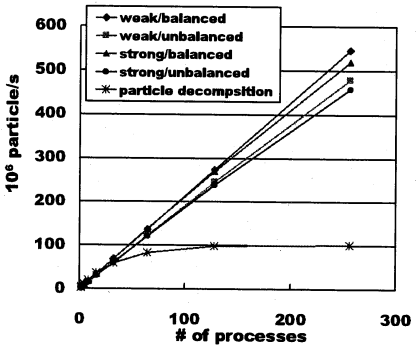


Fig. 6 Performance of Simulations

and with periodic boundary condition commonly for the four combinations of strong/weak scaling and balanced/unbalanced cases. The tolerance factor α in inequalities (1) and (2) was also commonly set to 0.2.

4.2 Strong Scaling Performance

Measured performance numbers of strong scaling executions with up to 256 processes (cores) are shown in **Table 1**, while **Fig. 6** shows them graphically together with those of weak scaling which is discussed latter. The performance numbers in the table are displayed in the form of "*speed*/*speedup*". The simulation *speed* in the unit of particle-per-second (PPS) is defined by the number of particle pushes performed in on second. The *speedup* is the simulation speed normalized by that of the sequential execution of the reference implementation only using particle decomposition method whose performance is also shown in Table 1. Note that this reference sequential execution has no overhead for parallelization, while one-process execution of our simulator incurs self communications of particle movements crossing periodic boundaries and for exchanging boundary field/current values by itself. The absolute execution time of the reference sequential ex-

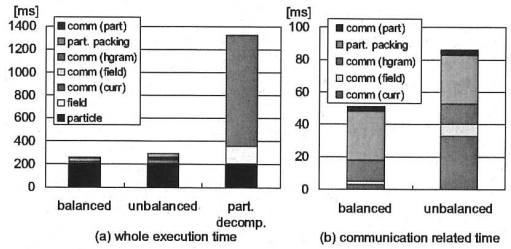


Fig. 7 Breakdown of Strong Scaling Execution Time

ecution is about 83 hours or three days and a half, while 256-process strong scaling executions take 28 minutes and 31 minutes with balanced and unbalanced setting respectively.

Table 1 and Fig. 6 clearly shows that that our simulator exerts good scalability and much more scalable than the reference particle-decomposed type simulator as expected. The performance saturation of the reference simulator is caused by the cost of all-reduce communication in current-scatter phase for the $64^3 \times 3 \times 8 = 6\,\mathrm{MB}$ array of current density, which becomes the dominant factor as the number of processes increases. In fact, from the observation of **Fig. 7**(a) which shows the breakdown of the 256-process execution time for one time-step, we can find that execution time proportional to the number of particles ("particle") takes only 15.5 %, while 73 % is for the all-reduce ("comm. (current)") and the remaining and non-negligible 11.5 % is for the field solving ("field").

On the other hand, our simulator looks exerting unsaturated performance. A closer look of the numbers, however, reveals that the parallel efficiency descends from almost stable 74 % in balanced and 65 % in unbalanced case up to 128-process execution down to 70 % and 62 % in 256-process one. This is an Amdahl's law effect caused by the collective communications for particle amount histogram which takes about 5 % of the total execution time (see "comm. (hgram)" in Fig. 7(a) and (b)). This proves, on the other hand, that our 256-process simulation is only 5 % slower than the best case of a simple domain-decomposed type one because the histogram manipulation is the sole thing added to it in the balanced case.

The dominant factor to decrease the performance of the unbalanced case from the balanced

Table 2 Performance of Weak Scaling

#proc	balanced	unbalanced
1	2.55/ 0.89	2.55/ 0.89
2	4.95/ 1.72	4.63/ 1.61
4	9.11/ 3.16	8.06/ 2.80
8	18.31/ 6.36	15.70/ 5.46
16	34.44/ 11.97	30.58/ 10.63
32	68.57/ 23.83	59.88/ 20.81
64	137.13/ 47.66	122.29/ 42.50
128	274.00/ 95.22	245.64/ 85.36
256	545.86/189.69	478.93/166.44

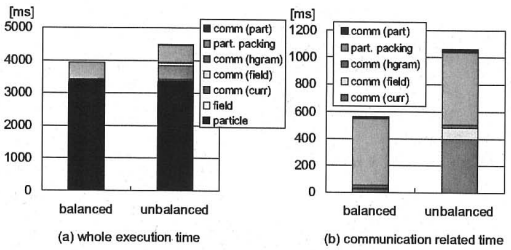


Fig. 8 Breakdown of Weak Scaling Execution Time

one is again the all-reduce of the current density in each helpand-helper family, which takes about 11 %, but this is much smaller than that in the reference simulator because the reduced array size is shrunk to $8 \times 8 \times 16$ or 1/256 of the reference simulator's. The other factors are so small that 1.5 % degradation by the broadcast of electromagnetic field data on boundaries ("comm. (field)") is most significant.

4.3 Weak Scaling Performance

The weak scaling simulation took 6–8 hours approximately irrespective of the number of processes because the problem size is enlarged according to the number of processes. Its performance shown in **Table 2** and Fig. 6 is better than the strong scaling as expected. More importantly, the parallel efficiencies above 16 processes of balanced and unbalanced cases are both stable even in 256-process executions at 75 % and 66 % respectively. This is led by the fact that the Amdahl factor of histogram communication is almost invisible in the breakdown shown in **Fig. 8** and merely occupies about 0.5 % of the whole. Thus, as far as this factor concerns, the simulation will be weakly scalable at least up to 4096 processes at which it will incur about 10 % histogram communication overhead even if we assume collective communication costs proportional to the number of processes.

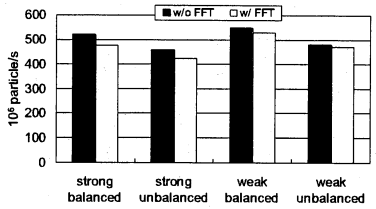


Fig. 9 256-Process Perforamance with and without Poisson's Equation Solving

As in the strong scaling, the major degradation factor from balanced to unbalanced setting is the all-reduce of current density which explains about two-third of 14 % degradation, and is followed by the boundary communication of electromagnetic field which causes about 2 % degradation. The other factors, such as solving field twice, checking good load balancing and relatively irregular particle transfer communications, are quite small and less than 1 % for each.

4.4 Poisson's Equation Solving

Since our simulator is for space plasma and adopts so-called charge conservation technique, it is unnecessary or at most very infrequently required to solve Poisson's equation explicitly to have electrostatic potential from charge density. However, in other types of problems such as those involving conductors in the space domain, solving Poisson's equation in every time step is essentially required[4].

Thus we experimentally attached a Poisson's equation solver based on a simple but parallelized three-dimensional FFT and evaluated the performance of this version. As the 256-process performance in **Fig. 9** shows, the global operations in FFT, such as transposition of three-dimensional grid point array, is not very harmful to scalability. That is, the performance degradation due to the Poisson solver is small especially in weak scaling executions, only 3 % and 1.6 % in balanced and unbalanced settings respectively. Although this cannot conclude that FFT-based Poisson solver is scalable with more processes and larger space domain, at least it is confirmed that PIC simulations requiring frequent solving of Poisson's equation are fairly efficient with thousand-scale parallelization.

5. Conclusion

In this paper, we proposed a new method for PIC simulations, named OhHelp, to achieve both good load balancing and scalability. The OhHelp method simply and equally partitions the space domain for scalability with respect to domain size. It also copes with the imbalance of the number of particles in subdomains by making each node help another node to which a densely populated subdomain is assigned. Our implementation of a three-dimensional space plasma simulation with OhHelp exhibits a good scalability showing 150–190 speedup with 256 processes compared to the sequential execution of a reference particle-decomposed simulator whose speedup is saturated at about 35-fold.

Acknowledgments A part of this research work is pursued as a Grant-in-Aid Scientific Research #20300011 supported by the MEXT Japan.

References

1) Candel, A.E., Dehler, M.M. and Troyer, M.: A Massively Parallel Particle-in-Cell Code for the Simulation of Field-Emitter Based Electron Sources, *Nuclear Instruments and Methods in Physics Research*, Vol.A-558, pp.154–158 (2006).
2) Nakashima, H.: T2K Open Supercomputer: Inter-University and Inter-Disciplinary Collaboration on the New Generation Supercomputer, *Intl. Conf. Informatics Education and Research for Knowledge-Circulating Society*, pp.137–142 (2008).
3) Nakashima, H., Usui, H. and Omura, Y.: OhHelp: A Simple but Efficient Space-Partitioned Dynamic Load Balancing for Particle Simulations, *IPSJ SIG Notes*, 2007-HPC-113 (2007).
4) Usui, H. et al.: Development and Application of Geospace Environment Simulator for the Analysis of Spacecraft-Plasma Interactions, *IEEE Trans. Plasma Science*, Vol.34, No.5, pp. 2094–2102 (2006).
5) Usui, H. and Omura, Y.(eds.): *Advanced Methods for Space Simulations*, TERRAPUB (2007).
6) Wolfheimer, F., Gjonaj, E. and Weiland, T.: A Parallel 3D Particle-in-Cell Code with Dynamic Load Balancing, *Nuclear Instruments and Methods in Physics Research*, Vol. A-558, pp.202–204 (2006).