Viktor Malyshkin
Marina Kraeva
Supercomputer Software Department,
Computing Centre RAS
Novosibirsk, RUSSIA
e-mail: malysh@ssd.comcen-1.nsk.su
kraeva@ssd.comcen-1.nsk.su

Galina Dudnikiva
Vitaly Vshivkov
Institute of Computing
Technologies RAS
Novosibirsk, RUSSIA
e-mail:

MAIN PUBLICATIONS:

V.Malyshkin. Functionality in ASSY System and Language of Functional Programming. Proc. "The First Aizu International Symposium on Parallel Algorithms/Architecture Synthesis", 1995, Aizu-Wakamatsu, Fukushima, Japan. IEEE Comp. Soc. Press, Los Alamitos, California. P. 92-97.

V.Vshivkov, M.Kraeva, V.Malyshkin. Parallel Implementations of PIC Method. "Programmirovanie", 1996

THE KEY WORDS:

Parallel programming, Assembly technology, Particle-in-cell method

ANNOUNCEMENT

The subject of our research is assembly method of parallel program construction for supporting the numerical large-size problem solving on MIMD computers. In particularly we are creating the parallel programming language and system including automatic parallel programs synthesis subsystem. As example of large-size problem we consider plasma physic problems. Computer simulation of these problems is based on using of kinetic Vlasov's equations. Particle-in-cell (PIC) method is more acceptable for Vlasov's equations solving and apparently this method is the only in 3D-case. The parallel program realising this problem solving algorithm essentially depend on entrance data structure and size. Thus it's necessary construct new parallel program for every new data.

PIC METHOD

Consider PIC method in its application to solving the collisionless interaction of plasma flows problem. The model for this problem description is equations system than consists of kinetic Vlasov's equations for ions and electrons and Maxwell's equations:

$$\frac{\partial f_\alpha}{\partial t} + \vec{v}\frac{\partial f_\alpha}{\partial \vec{r}} + \frac{q_\alpha}{m_\alpha}\left(\vec{E} + \frac{1}{c}\vec{v}\times H\right)\frac{\partial f_\alpha}{\partial \vec{v}} = 0$$

$$rot\vec{H} = \frac{4\pi}{c}\vec{j} + \frac{1}{c}\frac{\partial E}{\partial t} \qquad rot\vec{E} = -\frac{1}{c}\frac{\partial \vec{H}}{\partial t}$$

$$div\vec{H} = 0 \qquad div\vec{E} = 4\pi\rho$$

Where α is a kind of particle, $f_\alpha(\vec{r},\vec{v},t)$ is a particles distribution function, q_α is charge, m_α is mass, $\vec{j}$ is current density, ρ is partial charge density, $\vec{E},\vec{H}$ are electric and magnetic intensities.

Current density and charge density are determined from distribution functions:

$$\vec{j} = \sum_\alpha q_\alpha \int f_\alpha(\vec{r},\vec{v},t)\vec{v}d\vec{v} \qquad \rho = \sum_\alpha q_\alpha \int f_\alpha(\vec{r},\vec{v},t)d\vec{v}$$

For discretization of this problem PIC method is used. In PIC simulations plasma is represented as a large number of "superparticles" each of which models the action of a large number of physical particles. The motion trajectories of these "superparticles" (further we will drop the "super" prefix) are characteristics of Vlasov's equations. The particles are described by set of some attributes (charge, co-ordinates, velocity). All modelling space is broken up by rectangular grid into cells upon which the electric and magnetic fields, current density and charge density are discretized. Each particle lies in a cell of the grid. The dynamics of the system is followed by integrating the motion equation of each of the particles in a series of discrete time steps (fig.1)

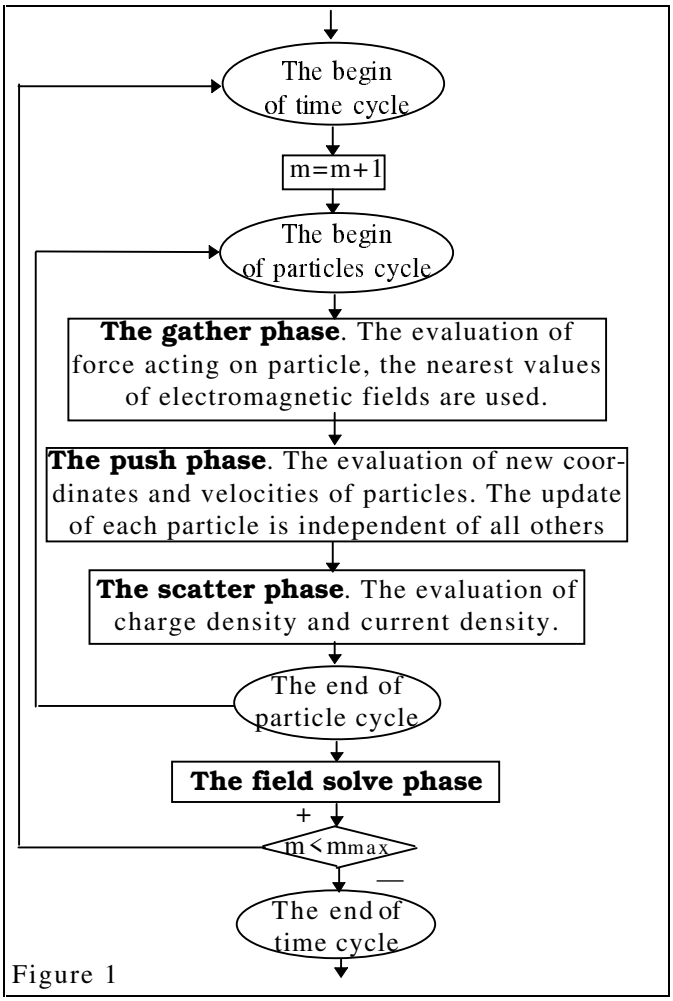


Figure 1

PARALLEL IMPLEMENTATIONS OF PIC METHOD

On MIMD distributed memory computers the performance characteristics of a PIC code depend crucially on how the grid and particles are distributed among the processors. The amount of computation in the scatter, gather and push phases of the PIC algorithm is proportional to the number of particles. The computational cost of the field solve phase is proportional to the number of grid points. Generally this cost much less than for other three phases. The push phase is easy to parallelize (fig. 1). However large amounts of grid data need to be exchanged between processors or load imbalance arises when scatter and gather phases are parallelized.

A parallel PIC code depend on following factors too:

- particles distribution in modelling space, character of particles motion;
- the number of particles and grid points;
- computer system architecture.

There are a great number of parallel implementations of various problems that was solved by PIC method on specific architectures. We realised this method in its application to solving the collisionless interaction of plasma flows problem on some parallel computer systems (multiprocessor computer ES-1068.17, multitransputer system that consist of 5 transputers T800-20). And gain methods will be used in subsystem for automatic synthesis of parallel programs realising PIC method. On the whole, development of ASSY (ASsembly SYstems) language and system is based first on the analysis of results of large-size problems solving including their parallelization, design

of parallel algorithm, data transferring and communications, programming and debugging.

ASSEMBLY TECHNOLOGY

Contrary to widespread parallelization method in which a problem, defined as a whole, is divided into the suitable fragments to be executed on the different processor elements, the assembly technology supports explicit assembling of a whole program out of ready made fragments of computations. These fragments are the "bricks" to build the whole program. An algorithm of problem assembling defines the "seams" of a whole computation, the way of the fragments connection. This algorithm is kept by ASSY and used later for program parallelization - the "seams" give to compiler an information for automatic dividing of entire computation along the "seams" in suitable fragments (as data as program code). Thus user describes the problem by using friendly interactive system and nonparallel program language for determination computations in minimal object.

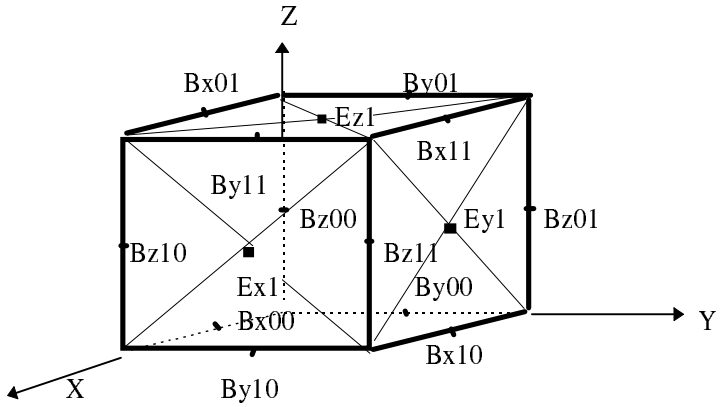
Let us consider assembly technology by the example of description of particle-in-cell method.

Definition of particle, of particle array

```
particle:array[2](coordinates x,y,z:real; u,v,w,q:real;);
```

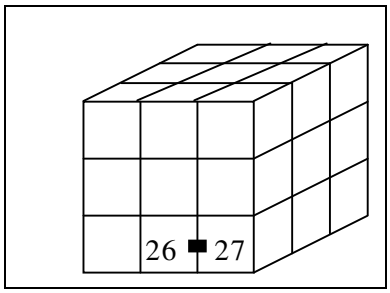
Definition of cell

```
cell:(  i Є N[0,K+1];
        j Є N[0,K+1];
        k Є N[0,M+1];
        Bx00,...,Uz1:array[3]double, Ex0,...,Pz1:array[2]double;
        cpart:array of particle);
```



Definition of minimal fragment of computations

```
box_3x3x3:(  ci,cj,ck:int;
             cell1=cell: {cell.i==ci,cell.j==cj,cell.k==ck};
             ...
             cell27=cell: {cell.i==ci+1,cell.j==cj+1,cell.k==ck-1,);
```



Variables identification

cell26.Bz11 **ident** cell27.Bz10;

...

Description of operations

...

forall particle ∈ cell1.cpart

{oper new_param (in particle[0].x,...,particle[0].q,tau,h1,h2,h3,cell1.Bx00[1],...
out particle[1].x,...,particle[1].w,cell1.Px0[0],..., cell5.Uz1[0])
};

...);

Assembling of modelling space out of fragments (with covering)

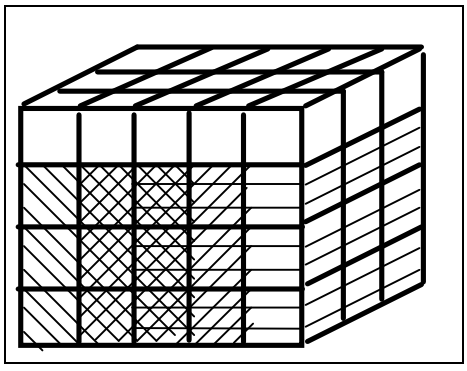
MS:array[](**forall** ci ∈ N[1,K]

forall cj ∈ N[1,L]

forall ck ∈ N[1,M]

{ MS:**U**(box_3x3x3: {box_3x3x3.ci==ci,

box_3x3x3.cj==cj,box_3x3x3.ck==ck}); };



The identification of covered cells

forall ci ∈ N[1,K-1]

forall cj ∈ N[1,L-1]

forall ck ∈ N[1,M-1]

{ box_3x3x3.cell1: {box_3x3x3.ci==ci, box_3x3x3.cj==cj, box_3x3x3.ck==ck}
ident box_3x3x3.cell8: {box_3x3x3.ci==ci+1, box_3x3x3.cj==cj, box_3x3x3.ck==ck};

...

};);

CONCLUSION

Assembly programming technology is destined for numerical solving of various large-size problem. Thus we are interesting in studying and solving such complicated applied problems that can't be solved consistently with success.