John von Neumann Institute for Computing

Object-Oriented Programming and Parallel Computing in Radiative Magnetohydrodynamics Simulations

Vladimir Gasilov, Sergei D'yachenko, Olga Olkhovskaya, Alexei Boldarev, Elena Kartasheva, Sergei Boldyrev

published in

Parallel Computing: Architectures, Algorithms and Applications ,
C. Bischof, M. Bücker, P. Gibbon, G.R. Joubert, T. Lippert, B. Mohr, F. Peters (Eds.),
John von Neumann Institute for Computing, Jülich,
NIC Series, Vol. **38**, ISBN 978-3-9810843-4-4, pp. 475-482, 2007.
Reprinted in: Advances in Parallel Computing, Volume **15**,
ISSN 0927-5452, ISBN 978-1-58603-796-3 (IOS Press), 2008.

Object-Oriented Programming and Parallel Computing in Radiative Magnetohydrodynamics Simulations

Vladimir Gasilov, Sergei D'yachenko, Olga Olkhovskaya, Alexei Boldarev, Elena Kartasheva, and Sergei Boldyrev

Institute for Mathematical Modelling, Russian Academy of Sciences
Miusskaya Sq. 4-A, 125047 Moscow, Russia
E-mail: $\{$*gasilov, boldar, bsn*$\}$*@imamod.ru*

The subject of this paper is the computer simulation of transient processes in strongly radiative plasma, that refers to solving the problems of radiative magnetohydrodynamics (MHD). The program system MARPLE developed in IMM RAS is described, where the application of OOP and parallel computing is emphasized. The strategy and scheme of code parallelizing are explained, with the outcome being presented in the results of practical computations.

1 Introduction

Modern problems in pulsed-power energetics issue a real challenge to the computer simulation theory and practice. High-performance computing is a promising technology for modelling complex multiscale nonlinear processes such as transient flows of strongly radiative multicharged plasmas. An essential part of such numerical investigations is devoted to computer simulation of pinches resulted from electric explosion of cold matter, e.g. gas-puff jets, foam strings, or metallic wire arrays. These investigations were significantly stimulated by impressive results in the soft X-ray yield produced by wire cascades or double arrays obtained in Sandia National Laboratory, US[1]. The goal of numerical research in pulsed-power is to study the evolution of very intensive transient electric discharges and to perform a multiparametric optimization of future experimental schemes.

2 Mathematical Model and Numerical Methods

Numerical modelling of strongly radiative plasma flows goes along with development and sophistication of research codes. This paper concerns the code MARPLE (Magnetically Accelerated Radiative PLasma Explorer) created in the Institute for Mathematical Modelling, RAS[2]. MARPLE performs calculations in terms of cylindrical (r, z) frame of reference, while (r, φ) or Cartesian (x, y) geometry is also available. A plasma flow is considered in the single-fluid magnetohydrodynamics model given by the well-known theory of Braginsky. The MHD equations are written in a so-called 2.5-dimensional fashion, where the flowfield vectors are presented by all three components: velocity $\vec{w} = (w_r, w_\varphi, w_z)$, magnetic inductance $\vec{B} = (B_r, B_\varphi, B_z)$, and electric intensity $\vec{E} = (E_r, E_\varphi, E_z)$. The anisotropy of dissipative processes in presence of magnetic field is taken into account. The energy balance is described in terms of the electron-ion-radiation relaxation. The governing system of equations includes the radiative transport equation for spectral intensity, and is completed by data bases and/or analytical description of plasma state, coefficients of transport processes, spectral opacities and emissivities.

Depending on the studied problem the governing system can also be supplemented by an equation describing electric current evolution for the whole electric circuit. For a typical pulsed-power problem the circuit includes an electric generator, current supply facilities, and a discharge chamber.

2.1 Governing System of Equations for 2-Temperature MHD

Non-dissipative MHD:

$$\frac{\partial \rho}{\partial t} + \nabla(\rho \vec{w}) = 0, \qquad \frac{\partial \rho w_i}{\partial t} + \sum_k \frac{\partial}{\partial x_k}\Pi_{ik} = 0,$$

$$\Pi_{ik} = \rho w_i w_k + P\delta_{ik} - \frac{1}{4\pi}\left(B_i B_k - \frac{1}{2}B^2\delta_{ik}\right),$$

$$\frac{\partial \vec{B}}{\partial t} - \nabla \times (\vec{w} \times \vec{B}) = 0,$$

$$\frac{\partial}{\partial t}\left(\rho\varepsilon + \frac{1}{2}\rho w^2 + \frac{B^2}{8\pi}\right) + \nabla \vec{q} = 0,$$

$$\vec{q} = \left(\rho\varepsilon + \frac{1}{2}\rho w^2 + P\right)\vec{w} + \frac{1}{4\pi}\vec{B} \times (\vec{w} \times \vec{B}), \qquad P_{e,i} = P_{e,i}(\rho, \varepsilon_{e,i}).$$

Dissipative processes:

$$\vec{E} = \frac{\vec{j}_{\parallel}}{\sigma_{\parallel}} + \frac{\vec{j}_{\perp}}{\sigma_{\perp}},$$

$$\operatorname{rot}\vec{B} = \frac{4\pi}{c}\vec{j} = \frac{4\pi}{c}\hat{\sigma}\vec{E}, \qquad \frac{1}{c}\frac{\partial \vec{B}}{\partial t} = -\operatorname{rot}\vec{E},$$

$$\frac{\partial \rho\varepsilon_e}{\partial t} = -\operatorname{div}(\hat{\kappa}_e \operatorname{grad} T_e) + Q_{ei} + G_J + G_R,$$

$$\frac{\partial \rho\varepsilon_i}{\partial t} = -\operatorname{div}(\hat{\kappa}_i \operatorname{grad} T_i) - Q_{ei}, \qquad \varepsilon = \varepsilon_i + \varepsilon_e, \quad P = P_i + P_e.$$

2.2 The Physical Splitting Scheme and Numerical Algorithms

Non-dissipative MHD	Dissipative processes	
	Magnetic field diffusion, heat transfer, electron-ion exchange, Joule heat	Radiative energy transfer
Local processes	Quasi-local processes	Non-local processes
Explicit scheme	Implicit scheme	Explicit fractional-step scheme
High-res. TVD scheme with flux correction	Integro-interpolation finite volume schemes	Characteristic scheme
Time advance: 2nd order predictor-corrector scheme		

The code provides for the problem solution in a complex geometry domain with use of unstructured triangular and quadrilateral meshes where all physical variables are stored at the mesh vertices (nonstaggered storage). The conservation laws for mass, momentum, and energy are approximated by means of the finite volumes technique. In the current version of code the finite volumes are formed by modified Voronoi diagrams. An example of computational mesh and finite volumes is presented below in Fig. 1.

The MHD system is solved by the generalized TVD Lax-Friedrichs scheme which was developed for the unstructured mesh applications[3]. For the case of regular triangulation this scheme ensures the second order approximation of spatial derivatives (the third order is possible with a special choice of the antidiffusion limiters). For solving parabolic equations, describing magnetic diffusion and conductive heat transfer, we developed new finite-volume schemes constructed by analogy with the mixed finite-element method. The time-advance integration is explicit, the second approximation order being achieved due to the predictor-corrector procedure. The time step is restricted by the Courant criterion.

2.3 Radiation Transport

Radiative energy transfer is described by the equation for spectral radiation intensity. In practice, calculations are performed via multi-range spectral approximation. We solve the equation by means of a semi-analytical characteristic interpolation algorithm constructed on the base of Schwarzschild-Schuster approximation. The analytical solution along a characteristic direction is constructed by means of the forward-backward angular approximation to the photon distribution function[4,5]. The two-group angular splitting gives us the analytical expression for radiation intensity dependent on opacity and emissivity coefficients. The energy exchange between the radiation field and the gas is taken into account via radiative flux divergence, which is incorporated into the energy balance as a source function (G_R in the above system).

The transport equation for quasistationary radiation field in cylindrical geometry:

$$\sin\theta\left(\cos\varphi\frac{\partial I_\omega}{\partial r}+\frac{\sin\varphi}{r}\frac{\partial I_\omega}{\partial\varphi}\right)+\cos\theta\frac{\partial I_\omega}{\partial z}=-\aleph_\omega I_\omega+j_\omega.$$

A set of equations for the forward/backward intensity functions $I^{f/b}$:

$$\frac{\cos\theta_n-\cos\theta_{n+1}}{\Delta\theta_n}\left(\frac{\partial I^f_{n+1/2}}{\partial r}+\frac{I^f_{n+1/2}}{r}\right)+\frac{\sin\theta_{n+1}-\sin\theta_n}{\Delta\theta_n}\frac{\partial I^f_{n+1/2}}{\partial z}=-\aleph I^f_{n+1/2}+j,$$

$$\frac{\cos\theta_{n+1}-\cos\theta_n}{\Delta\theta_n}\left(\frac{\partial I^b_{n+1/2}}{\partial r}+\frac{I^b_{n+1/2}}{r}\right)+\frac{\sin\theta_{n+1}-\sin\theta_n}{\Delta\theta_n}\frac{\partial I^b_{n+1/2}}{\partial z}=-\aleph I^b_{n+1/2}+j.$$

The radiative energy density (radiative contribution in energy balance):

$$G_R=\sum_{s=1}^{k}\frac{\pi}{c}\sum_{n=1}^{N}\left(I^f_{n+1/2}+I^b_{n+1/2}\right)(\cos\theta_n-\cos\theta_{n+1}).$$

The forward/backward intensities along the i-th ray in the direction $\theta_{n+1/2}$:

$$I^f_{j+1}=\left(I^f_j-J_j\right)\exp(-\aleph_j l_j)+J_j,\qquad I^b_j=\left(I^b_{j+1}-J_j\right)\exp(-\aleph_j l_j)+J_j.$$

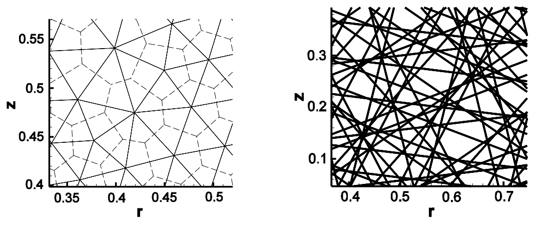

Figure 1. Fragments of the triangular mesh with finite volumes and of the grid of rays.

For the purpose of radiative energy transfer calculation a special grid of characteristics is constructed in the computational area. This grid is comprised by a number of sets (families) of parallel lines and is further referred to as the grid of rays. Each set of rays is characterized by the angle of inclination to coordinate axes and by spatial density of rays. The grid of rays introduces some discretization of computational domain in the plane (r, z) and with respect to the angle θ $(0 \leq \theta < \pi)$, that is required for numerical integration of the radiation transport equation according to the above described model. The grid of rays is superimposed on the initial computational mesh designed for MHD and heat transfer computations. A fragment of the grid (12 angle sectors) is shown in Fig. 1.

3 Parallel Implementation

All in all, the described sort of problems is computationally very hard, due to the big number of involved physical processes and corresponding equations. It consumes substantial amounts of processing time even on meshes which might look rather small for some other commonly known problems. At this, we often need to carry out quite long computations, because processes under investigation are inherently time-dependent. And industrial applications usually require numerous series of similar computations. So it makes code parallelization for us not only challenging, but naturally highly desirable.

3.1 Parallelism & C++

Certain specificity of introducing parallelism into a program complex in our case relates to the object-oriented nature of MARPLE code, which essentially employs C++ language facilities, such as polymorphism, encapsulation, inheritance, and parametric programming.

Our data model takes its origin in the finite-element technique and provides a description of geometrical and topological properties for computational domains as well as for numerical meshes. Special data structures based on the concept of topological complex have been elaborated to provide problem statement in an arbitrary domain, and for handling unstructured meshes, including dynamic mesh changes. From some point, the programming language C++ was chosen for further elaboration of MARPLE codes, because it provides the most convenient way to implement data structures for handling dynamic unstructured meshes. Besides, the program implementation of various types of boundary conditions, equations of state, and some other physical elements of the model, can be realized in a natural way by use of inheritance and virtual functions. Also, specialized classes have been developed for handling data tables, comprising parameters of ionization, electric conductivity, thermodynamic and optical properties. The C++ language and, on the whole, the object-oriented approach, simplifies the creation and further development of large complicated computer codes, especially in case of team development.

When the challenge to parallelize the algorithm arises, the necessity to modify the data structures normally appears too. Some of the above mentioned structures have to be adapted to allow for parallel computations and data exchanges, taking into account also the requirement of keeping interprocessor communication adequately small. Although, as a matter of fact, many of the classes and functions can be used in each branch of parallel code as well as in a usual sequential code.

For the problems like those described above and solved on triangular meshes, the geometrical parallelism is commonly used. It implies that each branch of the code processes a subset of computational mesh, while the data exchanges provide the possibility of computations in the vicinity of "inter-branch" bounds. In case of explicit schemes on unstructured meshes it means that each branch of the code stores two types of the computational mesh elements. The "real" elements require regular computations specified by the used scheme, while the "fictive" elements are the images of some "real" elements processed by another branch, and the data in these "fictive" elements are not computed but received from that branch. The "fictive" elements form "margins" along the inter-branch bounds.

It should be noted that the "fictive" elements and the "margins" concepts may be used as well in a single-branch (sequential) code for problems with some spatial symmetry (problems with translational or rotational periodicity). The "fictive" elements in this case are the images of some "real" elements in the geometrical transformation, with respect to which the problem is symmetrical. But the principal idea remains the same: instead of regular computations in a "fictive" element, the data associated with this element are transferred from the element-preimage.

So, in case of using geometrical parallelism, the data structures describing computational domains and meshes have to be modified, in order to handle the "fictive" mesh elements and encapsulate the methods for proper data exchanges, when needed.

3.2 Radiation Transport

Highly accurate simulation of radiative energy transfer, including detailed reproducing of the radiation spectrum, is among the most important requirements to the developed code. The reason is its great significance to the design of powerful X-ray pulse sources. Thereto, the entire spectrum is divided into a number of frequency ranges (from tens to hundreds). It is necessary to reconstruct the radiation field with respect to its angular distribution for each frequency range. So, the radiative transfer is calculated independently in each waveband, and then the corresponding results are summarized. Besides, as a rule, the grid of rays has the number of points much bigger than in the basic mesh. This is why the radiation transport computation is one of the most laborious steps in radiative MHD simulations. Profilings run at a monoprocessor computer showed that sometimes up to 80-90% of the total processing time may be spent on the radiative transfer computation, depending on a particular problem. For instance, typical figures corresponding to our benchmark problem (see Section 4 below) with 100 frequency ranges are shown in Table 1.

Therefore, the radiation transport was naturally chosen as our primary target for parallelizing. Now it's time to note, that in the above shown scheme of physical processes

Ideal MHD:	3.4%
Magnetic field diffusion:	4.7%
Heat conduction + e-i exchange:	3.0%
Joule heat:	4.0%
Radiative energy transfer:	84.9%

Table 1. Computational costs on different physical processes.

radiative transport stands apart in certain sense, because its most computations are performed on a special grid of rays, different from the basic grid, and have an essentially non-local character. Hence, domain decomposition and geometrical parallelism seem to be not applicable here. We chose another way of distributing computational load, which appears more natural and usable in this case. Indeed, the frequency ranges model requires repeating large volume of uniform computation with different sets of opacity and emissivity values for each range. It seems appropriate to carry out these computations concurrently on several processors, and then to collect the results by simple summation, using the fact that all frequency ranges produce uniform and independent contributions to the total radiative energy fluxes.

Thus, the whole volume of computation within each time step can be divided into two principal parts: the computations performed on the basic triangular mesh, and the computations performed on the grid of rays. Since currently the parallelization of the former part is not yet finished, the same calculations in this part are carried out by all processors simultaneously. In fact, this doesn't lead to any time loss, but helps to avoid any data exchange before the second part begins, as all processors will have exactly the same data by that moment. Then computations along the rays are performed in parallel, each processor calculating energy fluxes in its own set of wavebands. As these computations are well independent, no data related to the rays or ray points need to be exchanged. It makes this parallel computation quite efficient, provided that the number of processors is the divisor of the number of wavebands. However, to finalize this part, the radiative energy density must be summed up over all frequency ranges, so the global data collection is once required, actually in a scale of one variable defined on the triangular mesh (G_R).

3.3 C++ & MPI

Parallel computations on our cluster systems (with distributed memory architecture) were organized by means of the MPI framework, which is well efficient and highly portable as the prevalent today standard for message-passing communication model. However, the original library doesn't make use of OOP capabilities, that could be quite profitable in simplifying parallel programming[6]. On the other hand, the library contains plenty of efficient specialized functions with numerous options giving great flexibility to users, but practically many researchers tend to use some very limited subset of this richness for their particular computational problems. In order to simplify library usage and bring it closer to the C++ character of MARPLE codes, an intermediate interface has been elaborated, that hides the less exploitable details of the library but at the same time conveys some appropriate amount of flexibility provided by MPI to the core computational codes. As a result, the interprocessor communications are organized in the overall manner of C++ paradigm, i.e. by using some set of datatype-secure polymorphic functions, convenient for our actual needs. E.g. asynchronous Send calls will look like the following:

```
aSend( int dest_proc, DataType *data_ptr, int data_count );
aSend( int dest_proc, DataType &variable );
```

However, this is yet our first approach realized for the moment. In fact, the sending/receiving functions are called now directly from modules that implement numerical methods and schemes. The next step will be encapsulating such calls into classes that are responsible for data description and storage, so that a command given by a computing module would be not to send/receive/share some particular data array, but to exchange or refresh, or acquire any specific data structures, like those described above in Section 3.1.

4 Practical Results

The program codes have been proved in practical computations on two cluster systems currently available in IMM RAS. Both systems are Linux-driven self-made clusters based on the Intel platform[a]. The results obtained in a set of test computations are presented in Table 2. Here, the triangular mesh contains 21221 node, and the grid of rays consists of 6160 rays with total of 1054553 points for computation. The number of frequency ranges for this series of computations was taken 100. Times given in seconds correspond to one time step calculations on our single-core Xeon-based cluster.

	Number of processors				
	1	2	4	10	20
Full time (100%)	48.82 sec	29.51 sec	18.76 sec	13.56 sec	11.16 sec
Grid of rays	41.44 (85%)	20.8 (70%)	10.1 (54%)	3.94 (29%)	1.89 (17%)
Data exchange	0 (0%)	0.29 (1.0%)	0.51 (2.7%)	0.55 (4.1%)	0.58 (5.2%)
Speed-up on rays	1.00	1.99	4.10	10.52	21.90
Overall speed-up	1.00	1.65	2.60	3.60	4.37
Overall efficiency	100%	83%	65%	36%	22%

Table 2. Timings of practical computations.

As it can be clearly seen from the table, the overall efficiency rapidly falls down with the growing number of processors, although the part of computation run on the grid of rays exhibits a perfect linear acceleration. This is an evident consequence of the fact that the latter is the only part of computation actually parallelized by now. So even if on a single processor it takes 85% of the total time cost, this proportion tends to change very quickly. Actually, for the small numbers of frequency ranges the parallelization gives here very limited effect, e.g. initially in the above problem with 20 wavebands the contribution

[a]The first cluster consists of 26 modules, each of them containing two Intel Xeon 3 GHz processors and 2 Gb of RAM, all modules being connected by Gigabit Ethernet network. The second cluster is similarly comprised by 14 dual-processor modules with 4 Gb RAM, where each processor is the dual-core Xeon 2.66 GHz.

of calculations along the rays into the whole computation time was only about 50%, so it would be hard to reach even the acceleration of factor 2. Obviously, doing only some distinct part of computations in parallel is not enough, though we must admit that even such limited acceleration is quite useful for us in practice.

Thus, from this point we are going to parallelize the rest of our code. First of all, we think to employ physical splitting scheme further, so that to parallelize computations on different physical processes independently, using different techniques (e.g. for explicit and implicit schemes), and dedicating different groups of processors for these tasks. The nearest stage of code development is to detach hyperbolic processes from parabolic ones, and to organize additional iterations within the latter group, together with radiative transfer. Hence, we suppose that parallelizing radiation transport computation has been only the first though important step towards making the whole MARPLE code parallel.

5 Conclusions

It proved possible to make good use of the proposed technique in numerical simulation of heterogeneous liners implosion with powerful X-ray pulse generation. The corresponding experiments are performed at "ANGARA 5-1" facility (TRINITI, Troitsk, Moscow reg., Russia)[7]. Parallel computing technology applied to the radiative energy transfer calculation helped us to reduce the total processing time by factor of 2 to 4. This is a significant practical achievement, since for the experiment scheme optimization a big series of similar numerical simulations is actually needed. Another concurrent advantage is that the number of ranges in a spectrum can be increased, that gives immediate effect on the accuracy and quality of numerical solutions. For example, in the above mentioned numerical experiments initially only 20 frequency ranges were differentiated, and for parallel computations this number was notably increased to one hundred.

Acknowledgements

The work is carried out with the help of Russian Academy of Sciences, Mathematical Sciences Department (program 10002-251/OMN-03/026-023/240603-806), and Russian Foundation for Basic Research (grant No. 05-01-00510).

References

1. J. P. Chittenden, et al., Plasma Phys. Control Fusion, **46**, B457–B476, (2004).
2. V. A. Gasilov, et al., Mathematical Modelling, **15**, No. 9, (2003).
3. V. A. Gasilov and S. V. D'yachenko, *Quasimonotonous 2D MHD scheme for unstructured meshes*, in: Mathematical Modelling: Modern Methods and Applications, pp. 108–125 (Janus-K, Moscow, 2004).
4. R. Siegel and J. R. Howell, *Thermal Radiation Heat Transfer*, (McGraw-Hill, 1972).
5. B. N. Chetverushkin, *Mathematical modelling in radiative gasdynamic problems*, (Nauka, Moscow, 1985).
6. C. Hughes and T. Hughes, *Parallel and Distributed Programming Using C++*, (Addison-Wesley, 2004).
7. V. V. Alexandrov, et al., Plasma Phys. Reports, **27**, No. 2, (2001).