Available online at www.sciencedirect.com

ScienceDirect

Computers and Mathematics with Applications 55 (2008) 1094–1101

An International Journal
computers & mathematics with applications

www.elsevier.com/locate/camwa

Parallel implementation of 3D global MHD simulations for Earth's magnetosphere[☆]

Zhaohui Huang[a,*], Chi Wang[a], Youqiu Hu[b], Xiaocheng Guo[b]

[a] State Key Laboratory of Space Weather, Center for Space Science and Applied Research, Chinese Academy of Sciences, Beijing 100080, People's Republic of China

[b] School of Earth and Space Sciences, University of Science and Technology of China, Hefei 230026, People's Republic of China

Received 20 October 2005; received in revised form 22 March 2007; accepted 21 June 2007

Abstract

This paper presents a dynamic domain decomposition (D^3) technique for implementing the parallelization of the piecewise parabolic method (PPM) for solving the ideal magnetohydrodynamics (MHD) equations. The key point of D^3 is distributing the work dynamically among processes during the execution of the PPM algorithm. This parallel code utilizes D^3 with a message passing interface (MPI) in order to permit efficient implementation on clusters of distributed memory machines and may also simultaneously exploit threading for multiprocessing shared address space architectures. 3D global MHD simulation results for the Earth's magnetosphere on the massively parallel supercomputers Deepcomp 1800 and 6800 demonstrate the scalability and efficiency of our parallelization strategy.
© 2008 Published by Elsevier Ltd

Keywords: Parallel; Dynamic domain decomposition (D^3); PPM; MHD; MPI

1. Introduction

Numerical simulation becomes more and more popular in experimenting with physical models, and computing plays an important role in investigating the interaction with experiments and analytical theory. Advances in high performance computing make detailed and accurate calculations of hydrodynamics and magnetohydrodynamics (MHD) feasible [1–8]. Specifically, 3D global MHD simulations can obtain the magnetospheric configuration and examine the response of the magnetosphere–ionosphere system to changing solar wind conditions. The MHD model combines fluid equations and Maxwell's equations, and the intrinsic complexity of the MHD equations requires high resolution numerical methods [9–11]. Finite volume methods are among several different techniques available for solving the MHD equations since they are simple to implement, easily adaptable to complex geometries, and well

[☆] This work was supported by the National Natural Science Foundation of China (Nos. 40574069, 40674082, 40325010, 40390150, 40474053, 40621003) and INF105-SCE-02-15.

[*] Corresponding address: State Key Laboratory of Space Weather, Center for Space Science and Applied Research, Chinese Academy of Sciences, PO Box 8701, Beijing 100080, People's Republic of China.
E-mail address: hzh@cssar.ac.cn (Z. Huang).

0898-1221/$ - see front matter © 2008 Published by Elsevier Ltd
doi:10.1016/j.camwa.2007.06.025

suited to handling non-linear terms. The Godunov approach is a finite volume method in which non-linearity is introduced into the difference scheme via solution of the Riemann problem, which leads to an accurate and very well-behaved treatment of shock discontinuities [12–14]. An important Godunov-type approach is the piecewise parabolic method (PPM) given by Colella and Woodward in 1984 [15], which introduces a number of changes to achieve high order resolution in a Godunov method to ensure at least second-order up to fourth-order accuracy in space and second-order accuracy in time. An essential ingredient of PPM is a spatial reconstruction step to compute time-advanced estimates of the conserved variables at the grid face. It was originally developed to study the dynamics of supernova explosions and as such includes a Riemann solver which is capable of treating non-gamma law gases [16–18]. For relatively recent works refer to [19–22].

In recent years a variety of numerical algorithms and codes for multidimensional MHD based on the PPM employing the Lagrangian remap implementation have been developed and used extensively in astrophysics simulations. Massively parallel computers have a large effect on the quality of solution obtained with relatively short calculation times. Many PPM parallelization versions, e.g. EVH-1 based on VH-1 due to Blondin and colleagues [23], etc. are presented, and they have their own advantages when applied to the corresponding physical problems.

Here our objective is to propose a dynamic domain decomposition (D^3) technique after analyzing the concrete characteristics of the PPM algorithm and its computational steps applied to simulate the 3D global Earth's magnetosphere and develop a new PPM parallelization software bag including the MPI version and the OpenMP version based on two main parallel architectures.

This paper is divided into six parts. First we give the MHD equations and PPM scheme in Section 2; then the parallel strategy for PPM is presented in Section 3. Section 4 introduces two architectures of supercomputers applied, for which the simulation results are analyzed and discussed in Section 5. Finally, we draw some conclusions in Section 6.

2. The MHD equations and PPM scheme

The ideal MHD equations can be written in differential conservative Eulerian form as

$$\frac{\partial}{\partial t}\begin{bmatrix}\rho\\ \rho\mathbf{v}\\ \mathbf{B}\\ E\end{bmatrix}+\nabla\cdot\begin{bmatrix}\rho\mathbf{v}\\ \rho\mathbf{vv}+I\left(p+\dfrac{1}{2\mu}B^2\right)-\dfrac{1}{\mu}\mathbf{BB}\\ \mathbf{vB}-\mathbf{Bv}\\ \left(E+p+\dfrac{1}{2\mu}B^2\right)\mathbf{v}-\dfrac{1}{\mu}(\mathbf{v}\cdot\mathbf{B})\mathbf{B}\end{bmatrix}=\begin{bmatrix}0\\ \rho\mathbf{g}\\ 0\\ \rho\mathbf{v}\cdot\mathbf{g}\end{bmatrix}. \tag{2.1}$$

Here, ρ is the plasma density, p is the pressure, v is the velocity. $E=\frac{\rho}{\gamma-1}+\frac{1}{2}\rho v^2+\frac{1}{2\mu}B^2$ is the total energy density of the plasma. I and $\mathbf{g}$ are the unit matrix and the acceleration due to gravity, respectively. We take $\gamma=5/3$ for the ratio of specific heats.

The MHD equations represent coupling of the fluid dynamic equations with Maxwell's equations of electrodynamics, and describe the conservation of mass, momentum, magnetic flux, and energy.

We uses the finite difference method with the piecewise parabolic method (PPM) and a Riemann solver to solve the aforementioned equations [15–18]. PPM consists of three steps:

1. In Lagrangian coordinates, we implement the piecewise parabolic interpolations, compute the effective left and right states and solve the Riemann problems at each zone interface in order to determine the new zone averages on the Lagrangian grid.

2. Calculate the updated zone averages by differencing the Lagrangian fluxes, get new piecewise parabolic interpolations and integrate those of each variable over regions of overlap between the Lagrangian and Eulerian grids.

3. Map quantities from the Lagrangian grid back to the Eulerian grid, and difference them to obtain the final zone averages on the Eulerian grid.

In the Lagrangian evolution step, by introducing

$$V=\frac{1}{\rho},\qquad \frac{\mathrm{d}}{\mathrm{d}t}\equiv\frac{\partial}{\partial t}+\mathbf{v}\cdot\nabla, \tag{2.2}$$

we will obtain the following Lagrangian conservative form of MHD equations from the Eulerian conservative form:

$$\begin{aligned}
&\frac{\mathrm{d}V}{\mathrm{d}t} - \frac{1}{\rho}\nabla\cdot\mathbf{v} = 0,\\
&\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} + \frac{1}{\rho}\nabla\cdot\left(p^{*}I - \frac{1}{\mu}\mathbf{BB}\right) - \mathbf{g} = 0,\\
&\frac{\mathrm{d}(V\mathbf{B})}{\mathrm{d}t} - \frac{1}{\rho}\nabla\cdot(\mathbf{Bv}) = 0,\\
&\frac{\mathrm{d}e}{\mathrm{d}t} + \frac{1}{\rho}\nabla\cdot\left[p^{*}v - \frac{1}{\mu}(\mathbf{v}\cdot\mathbf{B})\mathbf{B}\right] = \mathbf{v}\cdot\mathbf{g},
\end{aligned} \tag{2.3}$$

where

$$e \equiv \frac{E}{\rho} = \frac{p}{(\gamma-1)\rho} + \frac{1}{2}(v^2 + v_A^2), \qquad v_A \equiv \frac{B}{\sqrt{\mu\rho}}, \qquad p^{*} = p + \frac{1}{2\mu}B^2. \tag{2.4}$$

3. Dynamic domain decomposition (D^3) technique

Multilevel Schwarz methods include multigrid methods [24–28], overlapping domain decomposition and iterative substructuring algorithms, which base the subspace splitting on the decomposition of the computational domain as the union of smaller subregions. Dynamic domain decomposition (D^3) combines domain decomposition with co-adaptation [29] of parallelization and solvers for the subproblems.

3.1. D^3 algorithm

Domain decomposition (D^2) methods decompose the given domain into subdomains and treat the subproblems — the given problem restricted to the subdomains, and the interface problems which represent the connection between the subproblems. In this respect, D^2 seems a natural concept for the parallelization of large scale computation for the solution of MHD equations. Usually, the partitioning, even recursive divide-and-conquer methods employed for decomposing the computational domain are not flexible and scalable since their principles guiding the design of decomposition for subdomains are less concerned with the joint adaptation of partitioning, communication, mapping and the efficient solution of subproblems and interface problems, which will have a serious effect on the implementation of parallelization and thus the whole efficiency. D^3 is characterized by its dynamic hierarchical structure which typically leads to parallelization. PPM possesses such an important characteristic that it is intrinsically one-dimensional, the extension to more directions being obtained by a direction splitting procedure [30,31]. If we indicate with L_k the operator in the kth direction, the update value for the MHD variable u is

$$u^{n+1} = L_z L_y L_x u^n, \tag{3.1}$$

and another alternate form is

$$u^{n+2} = L_x L_y L_z u^{n+1}, \tag{3.2}$$

where the index n represents the time $t^{n+1} = t^n + \Delta t$.

Let Ω_d be a shape-regular and quasiuniform decomposition of the domain Ω, with mesh width d, consisting of $\Omega_{i=1}^{m}$, where m is the number of elements. $\bar{\Omega}_{i=1}^{M}$ with M the number of subregions is an open covering Ω_D of Ω such that each subregion $\bar{\Omega}_i$ is the union of elements of Ω_d.

The D^3 algorithm has the following basic principles. First, choose a suitable open covering $\bar{\Omega}_x$ for the operator L_x, in order that the subproblem can be solved efficiently. Then, we have to transfer the data in $\bar{\Omega}_x$ so that they can be directly used by L_y, and another open covering $\bar{\Omega}_y$ can be obtained by the treatment of data communication between the subproblems is crucial to D^3, which is satisfied by substructuring the interface problem, that is, creating an auxiliary problem for the unknowns associated with the interface. Finally, compute L_y in $\bar{\Omega}_y$, do the same in the z-direction, and map the three direction data back to the whole region.

3.2. D^3 implementation

In MHD computations, the solution domain is subdivided into some subdomains which are then partitioned into grids. Grid points contain inner points, boundary points and imaginary points—"ghost" points, which are required since the PPM method is a finite volume technique with each grid point using the information at four nearest grid points along each spatial dimension to update the values of its variables. The boundary conditions are handled using the ghost zones. As expressed in (3.1) and (3.2), we always employ a one-dimensional sweep in one spatial direction, that is to say PPM applied a one-dimensional operator multiple times to compute three-dimensional MHD problems, so the kernel operates on one-dimensional arrays. It also tells us that the computational processes applied to all dimensions have some similar property; therefore the algorithm can be given as follows.

Step 1. *Choose the kth direction* ($k = x, y,$ *and* z) *as the basic partitioning direction and decompose the domain into subdomains, on which all variables and arrays are defined.*

After determining the basic direction (e.g. z), we can apply the operator L_x and L_y smoothly. But for L_z, we convert into Step 2.

Step 2. *From the remainder we select* x *or* y *as the auxiliary partitioning direction, along which* L_z *is employed.*

Here we need several interim arrays to transfer the data from the basic to auxiliary partitioning direction. How do we do this? See Step 3.

Step 3. *If any two in the number of subdomain or grid points for three directions are equal, we only perform 1D striped or 2D checkerboard partitioning for matrix transposition* [23]; *otherwise another two global data transfers have to be applied in the main program in order to make three sweep subroutines in three directions retain their structures.*

Obviously, the latter need communications twice as much as the former, but the extra load can usually be avoided since it is very easy to adjust the subdomain partition in a direction and increase less computations.

4. Architectures of parallel computers

Here we introduce architectures of two popular parallel computers used in our investigation:

Lenovo Deepcomp 1800 cluster at LSEC, CAS, consists of 256 computing nodes, with dual Intel Xeon 2 GHz processors and 1 GB main memory. The combined theoretical peak of this system is about 2 Tflops. Both fast ethernet (Intel pro 100, MPICH-1.2.4-p4mpd) and Myrinet 2000 (LANai 9, gm-1.5.2.1, MPICH-GM-1.2.1.7b) are equipped.

Lenovo Cluster Server Deepcomp 6800 at Supercomputing Center, CAS, is a four-node system, each node with four Intel Itanium 2 1.3 GHz processors running Oracle Database 10g with real application clusters and the Linux operating system. These 1024 processors are connected by QsNet and can reach a peak computing speed of 5 Tflops, an actual computing speed of 4 Tflops, and an overall efficiency of 78.5%. Deepcomp 6800 ranked 14th among the world's top 500 supercomputers in the ranking announced in mid-November, 2003. That was the highest ranking that a Chinese supercomputer had ever attained.

5. Performance results and discussions

We solve the MHD equations in the 3D computational domain $x \in (-300, 30)Re$, $y, z \in (0, 150)Re$, where Re is the radius of Earth, with the regular grid 160×80^2, using our parallel code on Deepcomp 1800 and 6800 supercomputers.

Before giving the computational results, we first introduce two indices which will be used in the following tables and text [32–35].

We introduce the speedup factor — a measure of relative performance of a multiprocessor system and a single-processor system, defined as

$$s(n) = \frac{t_s}{t_p} = \frac{\text{Execution time using one processor}}{\text{Exection time using a multiprocessor with } n \text{ processors}}$$

where t_s is the execution time on a single processor and t_p is the execution time on a multiprocessor. $s(n)$ gives the increase in speed in using a multiprocessor.

Table 1
Computational results on Deepcomp 1800 and 6800

Num. of processors	Time steps/min of 1800	Speedup factor of 1800	Efficiency of 1800 (%)	Time steps/min of 6800	Speedup factor of 6800	Efficiency of 6800 (%)
1	5	1.0	100.0	9	1.0	100.0
2	10	2.0	100.0	16	1.8	90.0
4	18	3.6	90.0	31	3.4	85.0
8	35	7.0	87.5	57	6.4	80.0
10	40	8.0	80.0	68	7.5	75.0
16	60	12.0	75.0	108	12.0	75.0
20	70	14.0	70.0	114	12.7	63.5
40	110	22.0	55.0	180	20.0	50.0

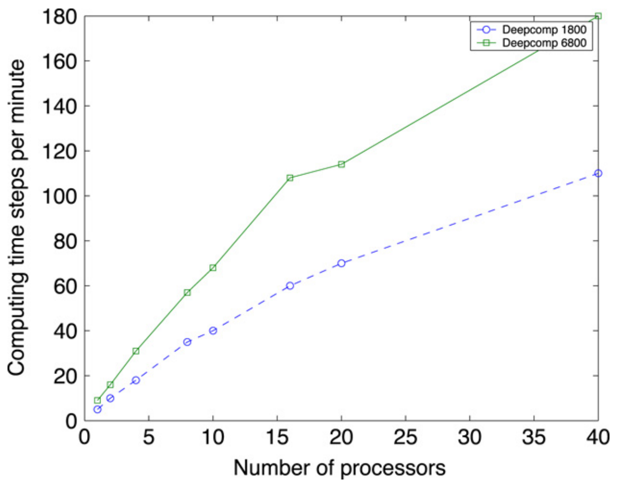

Fig. 1. Computing time steps per minute on Deepcomp 1800 and 6800.

Then the system efficiency E is defined as

$$E = \frac{t_s}{t_p \times n} = \frac{\text{Execution time using one processor}}{\text{Execution time using a multiprocessor } \times \text{ number of processors}}$$

which leads to

$$E = \frac{s(n)}{n} \times 100\%,$$

when E is given to a percentage. Efficiency gives the fraction of the time that the processors are being used on the computation.

Application performance is then studied in terms of the number of time step per minute, speedup and efficiency for our parallel code as Table 1 shows.

First we compare the computational speed of parallel code on the different parallel systems. From Fig. 1, we know that Deepcomp 6800 is approximately twice as fast as Deepcomp 1800 as fewer processors are used. When the number of processors is over 16, the efficiency of Deepcomp 6800 degrades faster than that of Deepcomp 1800.

For when using fewer than 20 processors, Fig. 2 tells us that the speedup factors do not differ substantially between Deepcomp 1800 and 6800; Deepcomp 6800's speedup factor for up to 20 processors becomes smaller than that for Deepcomp 1800 because Deepcomp 1800 has run normally and the whole performance of the system is relatively stable, but Deepcomp 6800 is still at the stage of testing and its performance will further be improved.

When this problem is run on up to 40 processors, the total performance degradation is instead observed as indicated by Fig. 3, since the granularity of the problem decomposition in the main partitioning direction is not large enough

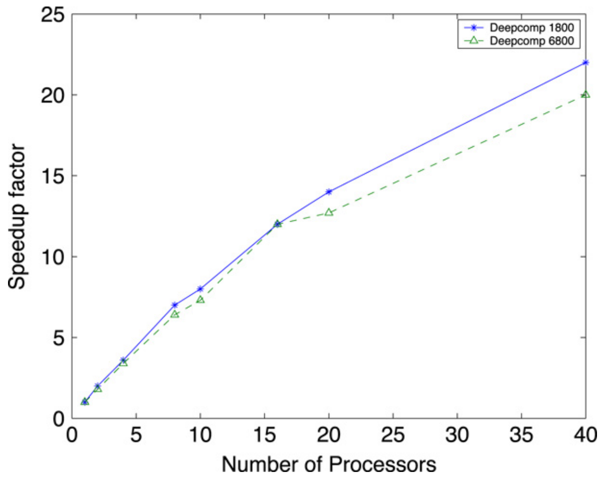

Fig. 2. Speedup factors on Deepcomp 1800 and 6800.

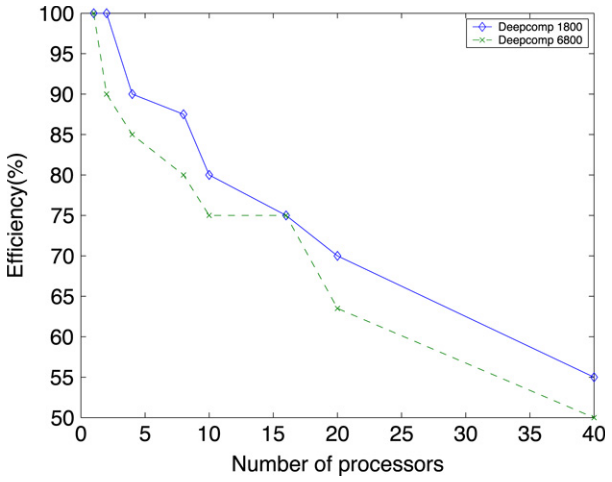

Fig. 3. Efficiency on Deepcomp 1800 and 6800.

and then the communication among blocks and processors increases; on the other hand, if we use fewer processors to enlarge the granularity, our main aim of massively parallel computations will not be realizable.

Therefore, due to the stable efficiency obtained when using 16 processors we think that 16 or 20 processors will be suitable for the problem we would like to solve, which can not only take the efficiency into account, but also save supercomputing resources.

In general, the performance of our code tested on the different parallel systems tells us that an efficiency of 70% can easily be attained for test examples, which shows that the D^3 algorithm is a computationally intensive technique that benefits from the use of massively parallel computers.

Performance has been evaluated on SGI Oringin 3800, and results similar to those obtained with Deepcomp 1800 are obtained.

6. Conclusions

In this paper we have analyzed and compared the performance of parallel code on two modern parallel architectures. Simulation results show that D^3 is a promising technique for parallelism since computation and then communication in each subdomain are local, and a relatively good load balance can be verified by the fact that performance does not degrade sharply with increasing number of processors. Although we propose D^3 based on the specific MHD simulations, the algorithm kernels can be applied to more extensive application areas. The dynamic load balance and better scalability should be investigated to exploit further the D^3 technique.

Acknowledgements

The authors are grateful to Prof. N. Cercone for his important comments and the Supercomputing Center and LSEC, CAS, for supporting us with the supercomputers.

References

[1] S. Dong, G. Karniadakis, Dual-level parallelism for high-order CFD methods, Parallel Comput. 30 (2004) 1–20.
[2] Th. Hauser, T. Mattox, R. Lebeau, H. Diety, P. Huang, Code optimizations for complex microprocessors applied to CFD software, SIAM J. Sci. Comput. 25 (4) (2004) 1461–1477.
[3] K. Itakura, A. Uno, M. Yokokawa, T. Ishihara, Y. Kaneda, Scalability of hybrid programming for a CFD code on the Earth simulator, Parallel Comput. 30 (2004) 1329–1343.
[4] M. Kim, H. Kim, O. Kwan, A parallel cell-based DSMC method on unstructured adaptive meshes, Internat. J. Numer. Methods Fluids 44 (2004) 1317–1335.
[5] R. Loon, P. Anderson, J. Hart, F. Baaijens, A combined fictitious domain/adaptive meshing method for fluid–structure interaction in heat values, Internat. J. Numer. Methods Fluids 46 (2004) 533–544.
[6] M. Prieto, R. Montero, I. Llorente, F. Tirado, A parallel multigrid solver for viscous flows on anisotropic structured grids, Parallel Comput. 29 (2003) 907–923.
[7] J. Trindade, J. Pereira, Parallel-in-time simulation of the unsteady NS equations for incompressible flow, Internat. J. Numer. Methods Fluids 45 (2004) 1123–1136.
[8] J. Walty, Parallel adaptive refinement for unsteady flow calculation on 3D unstructured grids, Internat. J. Numer. Methods Fluids 46 (2004) 37–57.
[9] A. Barmin, A. Kulikovskiy, N. Pogorelov, Shock-capturing approach and nonevolutionary solutions in MHD, J. Comput. Phys. 126 (1996) 77.
[10] J. Qiu, C. Shu, Finite difference WENO schemes with Lax–Wendroff-type time discretizations, SIAM J. Sci. Comput. 24 (6) (2003) 2185–2198.
[11] P. Roe, D. Balsara, Notes on the eigensystem of MHD, SIAM J. Appl. Math. 56 (1996) 57.
[12] W. Dai, P. Woodward, A high-order Godunov-type scheme for shock interactions in ideal MHD, SIAM J. Sci. Comput. 18 (4) (1997) 957.
[13] S. Godunov, In nonlinear hyperbolic problems, in: Proceedings of an Advanced Research Workshop, in: C. Carasso, P. Raviart, D. Serre (Eds.), Lecture Notes in Math., vol. 1270, Springer-Verlag, Berlin, 1987.
[14] A. Zachary, A. Malagoli, P. Colella, A high-order Godunov method for multidimensional ideal MHD, SIAM J. Sci. Comput. 15 (1994) 263.
[15] P. Colella, P. Woodward, The piecewise parabolic method (PPM) for gas-dynamic simulations, J. Comput. Phys. 54 (1984) 174–201.
[16] W. Dai, P. Woodward, An approximate Riemann solver for ideal MHD, J. Comput. Phys. 126 (1996) 77.
[17] W. Dai, P. Woodward, Extension of the piecewise parabolic method (PPM) to multidimensional ideal MHD, J. Comput. Phys. 115 (1994) 485.
[18] W. Dai, P. Woodward, On the divergence-free condition and conservation laws in numerical simulations for supersonic MHD flows, Astrophys. J. 494 (1998) 317.
[19] C. Wang, Z.H. Huang, Y.Q. Hu, X.C. Guo, Interaction of the interplanetary shocks with the magnetosphere, AIP Conf. Proc. 781 (2005) 320–324.
[20] C. Wang, C. Li, Z.H. Huang, J. Richardson, Effect of interplanetary shock strengths and orientations on storm sudden commencement rise times, Geophys. Res. Lett. 33 (14) (2006) L14104, doi:10.1029/2006GL025966.
[21] N.L. Borodkova, J.B. Liu, Z.H. Huang, G.N. Zastenker, C. Wang, P.E. Eiges, Effect of change in large and fast solar wind dynamic pressure on the geosynchronous magnetic field, Chin. Phys. 15 (10) (2006) 2458–2464.
[22] Y. Hu, X. Guo, G. Li, C. Wang, Z.H. Huang, Intrinsic oscillations of the earth's magnetosphere, Chin. Phys. Lett. 22 (10) (2005) 2723–2726.
[23] J. Blondin, G. Mahinthakumar, M. Sayeed, P. Worley, VH-1 User's Guide and EVH-1, North Carolina State University and Oak Ridge National Lab.
[24] Z.H. Huang, P.L. Shi, Notes on convergence of an algebraic multigrid method, Appl. Math. Lett. 20 (3) (2007) 335–340.
[25] Z.H. Huang, Q.S. Chang, Gauss–Seidel-type multigrid methods, J. Comput. Math. 21 (4) (2003) 421–434.
[26] Z.H. Huang, Q.S. Chang, An improved algorithm for the MG interpolation operator, Acta Math. Appl. Sin. 26 (3) (2003) 443–450.
[27] Z.H. Huang, Q.S. Chang, Multigrid solver based on the defect equation, Chin. J. Comput. Phys. 18 (5) (2001) 423–428.
[28] Q.S. Chang, Z.H. Huang, Efficient algebraic multigrid algorithms and their convergence, SIAM J. Sci. Comput. 24 (2) (2002) 597–618.
[29] Q. Du, Z.H. Huang, D.S. Wang, Mesh and solver coadaptation in finite element methods for anisotropic problems, Numer. Methods Partial Differential Equations 21 (4) (2005) 859–874.
[30] Z.H. Huang, D.S. Wang, Parallel numerical simulations for quantized vortices in Bose–Einstein condensates, Chin. Phys. 16 (1) (2007) 32–37.
[31] G. Strang, On the construction and comparison of difference schemes, SINUM 5 (1968) 506–517.
[32] R. Barrio, J. Sabadell, A parallel algorithm to evaluate Chebyshev series on a message passing environment, SIAM J. Sci. Comput. 20 (3) (1998) 964–969.

[33] V. Dolean, S. Lanteri, Parallel multigrid methods for the calculation of unsteady flows on unstructured grids: Algorithmic aspects and parallel performances on clusters of PCs, Parallel Comput. 30 (2004) 503–525.

[34] M. Parashar, S. Hariri, Interpretive performance prediction for parallel application development, J. Parallel and Distrib. Comput. 60 (1) (2000) 17–47.

[35] B. Wilkinson, M. Allen, Parallel Programming Techniques and Applications Using Networked Workstations and Parallel Computers, Addison-Wesley, 1999.