Bull. Nov. Comp. Center, Comp. Science, 10 (1999), 23–30

The domain decomposition parallel algorithm for multi-dimensional parabolic equations*

V.D. Korneev and S.A. Litvinenko

In the present paper, one of the methods of parallel solution to the multidimensional parabolic equations on the multiprocessor computing systems of the MIMD type is proposed. Much attention is given to the domain decomposition method and to distribution of subdomains among computers. As a computing system, on which the parallel solution to the considered problem was carried out and the results of numerical experiments were obtained, the multiprocessor system PowerXplorer was used.

Introduction

In the present paper, one of the methods of parallel solution to the multidimensional parabolic equations on the multiprocessor computing system of the MIMD type is proposed.

The main purpose, when solving a problem on parallel computing systems, is to attain efficiency. The efficiency (with respect to time) depends on the relationship between the time of calculations and the time of communications between nodes (in data exchange). The less time needed for communications, the higher the efficiency. For the MIMD systems with distributed memory, the optimal relationship between calculations and communications is provided by the methods of the coarse-grained parallelization, when parallel algorithms are constructed of large and rarely interacting blocks [1–3]. Problems solved by the grid methods, problems of linear algebra and many other problems, are effectively parallelized by the coarse-grained methods. The MIMD systems have different architectures, therefore those problems will be most effectively solved, whose structure of parallel algorithms are best suited to the architecture of a parallel computer system.

The algorithms considered here are parallelized by the domain decomposition method. Briefly, the essence of this method consists in the following. The basic data of a problem are distributed among nodes (branches of a parallel algorithm), and the algorithm is the same in all the nodes, but operations of this algorithm are distributed according to the data, available in

*Supported by the Russian Foundation for Basic Research under Grant 98–01–00709.

these nodes. The distribution of operations of an algorithm consists, for example, either in assignment of different values to a variable of the same cycle in different branches, or in execution of the different number of the same loops in different nodes, etc. The equal distribution of data among nodes serves a basis for the balance between the time needed for calculation, and the time needed for interactions of nodes.

When parallelizing the proposed algorithm, the following problems are solved: domain decomposition and optimal distribution of subdomains among processors.

As computer system, on which the parallel solution to the considered problem was carried out, and the results of numerical experiments were obtained, the multiprocessor system PowerXplorer [4] was used. In the first section, the statement of the problem is given. In the second section, the method of parallelization of the algorithm of the problem is described, in which much attention is given to the domain decomposition and distribution method of subdomains among computers. Finally, in the third section, the results of numerical experiments are given.

1. Statement of the problem

In the present paper, a parallel algorithm of solution to the multi-dimensional parabolic equation in a connected polygon is considered. For this purpose the solution to the following two problems is considered:

1. Domain decomposition with allowance for the balanced computing load on processors.
2. Optimal distribution of subdomains among processors.

In [4], for a solution to parabolic problems the domain decomposition method with splitting in subdomains was proposed. If the initial domain can be represented as a set of non-intersecting subdomains, the process of calculation can be performed in parallel. Let Ω be a bounded, open, connected polygon in R^2 which is a union of non-intersecting rectangles $\{\Omega_p\}_{p=1}^{s}$, i. e.,

$$\overline{\Omega} = \bigcup_{p=1}^{s} \overline{\Omega}_p, \qquad \Omega_p \cap \Omega_q = 0, \qquad p \neq q.$$

In the space $H^1(\Omega_p) \times H^1(\Omega_p)$, we set m one-parameter families of the bilinear forms of the kind

$$a_p^{(k)}(t; u_p, v_p) = \int_{\Omega_p} \lambda_k(t, \bar{x}) \frac{\partial u_p}{\partial x_k} \frac{\partial v_p}{\partial x_k} \, d\bar{x}, \qquad k = 1, \cdots, m, \qquad (1)$$

where $t \in [t_0, t_*]$ is a real parameter, $\bar{x} = (x_1, \ldots, x_m)$ is a point in R^m. Let us formulate a Neumann parabolic problem with conditions of nonideal contact in the component-wise form. Let $u_0 \in \hat{L}_2(\Omega)$ and $f \in L_2((t_0, t_*); \hat{H}^{-1}(\Omega))$. It is required to find the function $u^\rho \in L_2((t_0, t_*); \hat{H}^1(\Omega))$, such that $\frac{du^\rho}{dt} \in L_2((t_0, t_*); \hat{H}^{-1}(\Omega))$ and $\forall v \in \hat{H}^1(\Omega)$ for almost all $t \in (t_0, t_*)$ the following equalities hold:

$$\left(\frac{du_p^\rho}{dt}(t), v_p\right)_p + a_p(t; u_p^\rho(t), v_p) + \frac{1}{\rho} \sum_{q \in J_p} \int_{\Gamma_{(p,q)}} (u_p^\rho(t) - u_q^\rho(t)) v_p \, d\sigma$$

$$= (f_p(t), v_p)_p, \tag{2}$$

$$(u^\rho(t_0), v) = (u_0, v), \tag{3}$$

where $\rho > 0$, $a_p(tq; u_p, v_p) = \sum_{k=1}^s a_p^{(k)}(t; u_p, v_p)$, $p = 1, \ldots, s$.

For the solution to the third boundary value problem in a subdomain the splitting scheme is used. Let us introduce some notations. Let $\bar{n}_p$ and $\bar{x}^k$ be orthonormals to Γ_p and to the k-th coordinate axis. Let

$$\Gamma_p^{(k)} = \{\bar{x} \in \Gamma_p \mid \cos^2(\bar{n}_p, \bar{x}^k) = 1\}.$$

Let us introduce a set of numbers

$$J_p^{(k)} = \{q \in J_p \mid \Gamma_{(p,q)} \subset \Gamma_p^{(k)}\}.$$

In this case, as Ω_p is a rectangle, $\cup_{k=1}^m J_p^{(k)} = J_p$. And, finally, let $J_{p,\pm}^{(k)} = J_p^{(k)} \cap J_{p,\pm}$.

Let us formulate a grid Neumann problem in the component-wise form. Let N be a certain integer, $\tau = (t_* - t_0)/N$ and $t_n = t_0 + n\tau$, $n = 1, \ldots, N$. It is required to find a sequence $\{u^{n+k/m}, n = 1, \ldots, N-1; k = 1, \ldots, m\}$, such, that $u^{n+k/m} \in \hat{V}_h$ and $\forall v^k \in \hat{V}_h$ the following equalities hold:

$$d_{h,p}\left(\frac{u_p^{n+k/m} - u_p^{n+(k-1)/m}}{\tau}, v_p^k\right) + a_p^{(k)}(t_{n+1}; u_p^{n+k/m}, v_p^k) +$$

$$\frac{1}{\rho} \sum_{q \in J_{p,-}^{(k)}} \int_{\Gamma_{p,q}} (P_{h,p} u_p^{n+k/m} - P_{h,q} u_q^{n+k/m}) P_{h,p} v_p^k d\sigma +$$

$$\frac{1}{\rho} \sum_{q \in J_{p,+}^{(k)}} \int_{\Gamma_{p,q}} (P_{h,p} u_p^{n+k/m} - P_{h,q} u_q^{n+k/m-1}) P_{h,p} v_p^k d\sigma = (f_p^{n,k}, v_p^k)_p, \tag{4}$$

$$u_p^0 = \Pi_{h,p} u_{0,p}, \tag{5}$$

where $f_p^{n,k} = 0$, $k \le m-1$ and $f_p^{n,m} = f_p(t_{n+1})$, $u_q^{k/m-1} = u_q^0$, $d_{h,p}(u_p, v_p) = (P_{h,p} u_p, P_{h,p} v_p)$, $P_{h,p}$ is an operator making the mass matrices diagonal.

2. Method of problem parallelization

From formulas (4), (5) one can see that the solution to the problem in the complicated domain Ω is reduced to the solution to several problems in the rectangles Ω_p, $p = 1, \ldots, s$, and these problems can be solved in parallel, but only in such rectangles that have no common edges. In the present paper, the whole domain Ω is represented as a "red-black" partition to rectangles. The solution to the problem can be carried out in two half-steps. At the first half-step the problem is solved in "red" subdomains, at the second half-step – in the "black" subdomains. The data obtained at the previous iteration are necessary to solve the problem in the "red" subdomains and the data obtained at the same iteration in the "red" subdomains are necessary for solving in the "black" subdomains. The execution the problem in each subdomain is a process, and the solution to the whole problem is a set of interacting processes. At the first half-step the processes in the "red" subdomains are executed in parallel, then the obtained values of the boundaries of subdomains are passed to the adjacent processes performed at the second half-step in the "black" subdomains. Further the subdomains will be referred to as macroelements (m/e).

The program for the solution to the problem consists of the following:

1. Partition of the domain Ω into macroelements Ω_p.
2. Distribution of the macroelements Ω_p among processors.
3. Construction of a grid in macroelements.
4. Realization of the splitting algorithm.

Note that the first two stages are common for all the processors.

Let us consider each step in detail.

2.1. Partition of the domain Ω into macroelements Ω_p

Actually, partition of the domain into m/e means a transformation of the data structure of the initial domain Ω to the data structure of m/e Ω_p. The data structure of the initial domain includes geometric and physical parameters of the problem. The geometric parameters are the number of subdomains (since Ω can be multiply connected), the number of vertices in each subdomain and the coordinates of the vertices. The physical parameters include the right-hand side of the equation, the thermal conductivity, the boundary conditions of the problem. With the help of the function *geometry* (struct *macro* $*me$, int $*numelem$) the "red-black" partition of the initial domain Ω into m/e is done, where *me* is the array of structures m/e, *numelem* [2] is the number of the "black" and "red" m/e.

Let us describe the data structure of one m/e:

gr[*NUMGR*] – the array of coordinates of vertices of m/e;

jeo[*NUMGR*] – the array of a "state" of the boundaries of m/e, its element is equal either to the number of the adjacent m/es, or to -2, if the third kind boundary condition is given, or to -1 if the Dirichlet condition is given;

alfa[*NUMGR*] – the array containing either a penalty parameter, or a value of the factor α from the boundary condition;

f – the right-hand side;

lamda – the thermal conductivity;

nPros – the number of the processor calculating the given m/e;

LogLinId[*NUMGR*] – the array containing either the number of a logic data link, or -1 (if there is no neighbour).

Here *NUMGR* is the number of the m/e boundaries. At this stage the two latter elements of the structure are set equal to -1.

2.2. Distribution of macroelements Ω_p among processors

For an effective solution to the problem, the correspondence of a structure of parallel algorithm of a problem and the topology of the computer system is necessary. In this case, such topology is the torus with the dimension equal to the problem dimension. For this problem, it is *a two-dimensional torus.* The torus of the necessary size is set as a virtual topology. (The optimum map of virtual topology on concrete physical one is made by the software). The number of the processors TP is calculated as a maximum between the number of "red" and "black" m/e. However, if this number is odd and there is an m/e with four neighbours, it is necessary to increase TP by 1.

For each processor we introduce the structure *process* including:

kme[2] – the amount of "red" and "black" m/e in the processor;

nme[*MAXME*] – the number of an m/e in the array of structures *macroelem.*

In this case, the function *fprosess* (int *TP*, struct *macro* $*me$, int $*ne$) realizes the automatic distribution only in the case, when Ω is a rectangle, the amounts of "red" and "black" m/e are the same and even. After the distribution is made, the function *ProsFunc* (int *TP*, int $*dimx$, int $*dimy$, struct $*macro\ me$, int $*ne$, struct *process* $*pros$) determines the logic data links on the boundaries of m/e.

2.3. Construction of a grid in macroelements

The use of the lumping operators on the boundaries m/e allows us to solve problem (4), (5) on the non-matched grids. For the description of a grid, we introduce the structure *setgrid*, including such elements as:.

numps[*DIMEN*] – the number of nodes along each coordinate;

h[*DIMEN*] – a step of the grid on each coordinate.

On each processor, the memory is distributed for an array of structures *setgrid*, the amount of array cells being equal to the number m/e in the given processor. The construction of a grid is realized in the function *grid* (struct *setgrid* $*$*set*, struct *macro* $*$*me*, struct *process* $*$*ProsId*). With the help of the given function, the construction of three types of a grid is possible: a non-matched uniform grid, a matched uniform grid, an arbitrary grid. The parameters of a grid are read out from a disk file. All the processors will call to the same file. At the given stage, the calculation on all the processors is done independently of one another, and the information exchange does not take place.

2.4. Realization of the component-wise splitting algorithm

The algorithm of splitting (1.4), (1.5) reduces the solution to problem (1.2), (1.3) to the solution to a system of algebraic equations. As we deal with the rectangular domains, the partition into triangles is correct, hence the matrix of the system will be three-diagonal. The solution to such a system of equations is sought for with the help of the sweep method. The algorithm is implemented in the function *method* (struct *macro* $*$*me*, struct *process* $*$*ProsId*, struct *process* $*$*pros*, struct *setgrid* $*$*set*, struct *timestep* $*$*ts*, int *TP*, int *TopId*), where *TopId* is the name of topology, *ts* is the structure of iterative parameters such as the number of iterations and a time step. First, the memory for arrays is selected: u is solution to the problem, g is the right-hand side of the boundary conditions, a are factors of the sweep method. Then, with the help of the function *kprog* (int n, int *dim*, struct *macro* $*$*me*, double r, double $*a$, int *ifin*, int *ind*) the sweep factors are calculated. Further, immediately follows an iterative cycle. Note that we first do the calculations for the "red" m/e, and then for the "black" ones. In the iterative cycle the function *prog* (int n, struct *macro* $*$*me*, struct *setgrid* $*$*set*, double $*r$, double $*u$, double $*g$, double $*a$, int *nmax*, struct *process* $*$*ProsId*, int *TopId*) (n is the number of m/e in an array of structures *me*), implementing the sweep method, is called. An important part of the program is the function *ps* (struct *process* $*$*ProsId*, int *dim*, struct *macro* $*$*me*, struct *setgrid* $*$*set*, int n, double $*u$, double $*g$, int *TopId*). It calculates the right-hand side of the system of equations (4), (5) for the sweep method which includes the

integrals on the boundaries m/e of the non-matching grids. The parameter *dim* specifies the part of the boundary on which the integration takes place: 1 – integration over the horizontal boundary, 2 – integration over the vertical one. In this function, communications among the processors take place, as the integrals in it are calculated for the adjacent m/e.

3. Numerical experiments

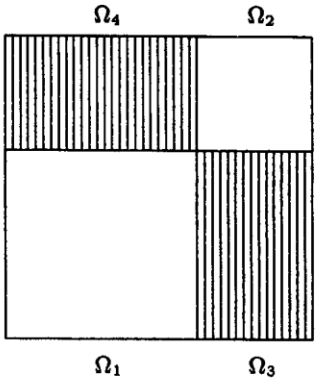
Numerical experiments were carried out on a test example for checking the program. In the square $\Omega = (0,1) \times (0,1)$, we consider a heat equation with the homogeneous Dirichlet conditions:

$$\begin{aligned} &du/dt = \lambda_0 \Delta u, && (t, x_1, x_2) \in (0,1) \times \Omega, \\ &u(t, x_1, x_2) = 0, && (t, x_1, x_2) \in (0,1) \times \Gamma, \\ &u(0, x_1, x_2) = \sin \pi x_1 \sin \pi x_2, && (x_1, x_2) \in \Omega. \end{aligned}$$

The exact solution to the given problem is the function

$$u(t, x_1, x_2) = e^{-2\lambda_0 \pi^2 t} \sin \pi x_1 \sin \pi x_2.$$

In the calculation, we set $\lambda_0 = 0.05$. For such a value of the parameter λ $L_2(\Omega)$-norm of the solution, is decreased, approximately, by $e^{-1}/2$ times at $t = 1$. The partition of the domain is shown in the figure.



In the figure, we used the following notation:

$\Omega_1 = (0, 5/8) \times (0, 5/8)$, $\Omega_2 = (5/8, 1) \times (5/8, 1)$,
$\Omega_3 = (5/8, 1) \times (0, 5/8)$, $\Omega_4 = (0, 5/8) \times (5/8, 1)$.

In the domain Ω, the square grid with a step h is introduced. For the discrete $L_2(\Omega)$-norm of an error, we use the notation:

$$\varepsilon_k = \left\{ \sum_{p=1}^{s} \sum_{i \in I_p} [u^N(x_{1,i}, x_{2,i}) - u(1, x_{1,i}, x_{2,i})]^2 \right\}^{1/2} h,$$

where $k = 1, 2, 3$, $N\tau = 1$. The error ε corresponds to the calculation with the parameter ρ, for which an error is minimal for h and τ. The table shows the results of computations by the method (4), (5) setting $h = \tau$.

$\log_2 h$	−3	−4	−5	−6	−7	−8	−9
ρ	0.25	0.14286	0.09524	0.06452	0.04545	0.03117	0.02222
ε	0.05109	0.03140	0.02058	0.01397	0.00966	0.00674	0.00474
t_1	0.36	0.36	0.44	0.92	4.57	36.96	380.17
t_2	0.64	0.66	0.73	1.15	3.76	26.37	245.86

In the table, the computational time for the solution to the problem with one processor (t_1) and two processors (t_2) is presented. The behaviour of the magnitudes ρ and ε corresponding to evaluations of errors of the given method is considered in detail in work [1]. As it is seen from the table, with magnification of the number of steps the relation of t_1 to t_2 increases. Thus, for the problems with a large number of nodes of a grid, the use of the proposed method is highly effective.

References

[1] Mirenkov N.N. Parallel programming for multimodular computing systems. – Moscow: Radio Svyaz, 1989.

[2] Malyshkin V.E. Linearization of mass calculations // System Computer Science / Ed. V.E. Kotov. – Novosibirsk: Nauka, 1991. – № 1. – P. 229–259.

[3] Korneev V.D. A system and methods of programming of multicomputers on an example of the computer complex PowrXplorer. – Novosibirsk, 1998. – (Preprint / Russ. Acad. Sci. Sib. Branch. Ins. Comp. Math. and Math. Geoph.; 1123).

[4] Laevsky Yu.M., Litvinenko S.A. On a method of the component-wise domain decomposition for solution of the parabolic equations. – Novosibirsk, 1993. – (Preprint / Russ. Acad. Sci. Sib. Branch. Comp. Cent.; 991).