Parallel Simulation of Oil Extraction on Heterogeneous Networks of Computers

A. Lastovetsky

Institute for System Programming, Russian Academy of Sciences
25 Bolshaya Kommunisticheskaya str., Moscow 109004, RUSSIA
Phone: (7 095) 912 0754 E-mail: lastov@ispras.ru

B. Chetverushkin, N. Churbanova and M. Trapeznikova

Institute for Mathematical Modelling, Russian Academy of Sciences
4-A Miusskaya Square, Moscow 125047, RUSSIA
Phone: (7 095) 972 1159 E-mail: sergepol@kiam.ru

Keywords: Multiprocessors, Heterogeneous Networks of Computers, Fluid Filtration Problems, Implicit Numerical Methods, Parallelization Efficiency

Abstract

Progress in network technologies has led to the situation when not only specialized parallel computers, but also local networks of computers and even global ones could be used as parallel computer systems for high performance parallel computing. The main peculiarity of such networks differing them from supercomputers, is their heterogeneity. While a typical distributed memory multiprocessor (MPP) consists of identical processor nodes interconnected via special equipment providing fast communications between the nodes, networks of computers are inherently heterogeneous and comprise diverse workstations, PCs, servers and, sometimes, specialized parallel computers interconnected via mixed network equipment. In particular, it causes the heterogeneity of both performances of computing processor nodes and communication speeds/bandwidths of links having an essential impact on efficiency of parallel applications. The paper addresses the problem and presents experience of development of portable parallel application efficiently simulating oil extraction process on heterogeneous networks of computers.

1 Introduction

As far as ten years ago, parallel computer systems were restricted mostly by so-called supercomputers — distributed memory multiprocessors (MPPs) or shared memory multiprocessors

(SMPs). Parallel computing on common networks of workstations and PCs did not make sense, since it could not speed up solving most of problems because of low performance of commodity network equipment. But in the 1990s, network capacity increases surpassed processor speed increases [1]. This has led to the situation when not only specialized parallel computers, but also local networks of computers and even global ones could be used as parallel computer systems for high performance parallel computing.

So, the problem of development of portable parallel applications efficiently solving problems on networks of computers has become a new challenge in high performance computing. Two polar approaches to this problem are possible. The simplest approach concludes in porting parallel applications written in PVM [2], MPI [3] or HPF [4] from MPPs to networks of computers. The approach is based on the fact that, in their turn, the above programming tools, originally aimed at MPPs, are currently widely ported to SMPs and networks of computers. The second extreme approach concludes in development of brand new applications taking into account all peculiarities of such a parallel platform as networks of computers. This approach needs new programming tools, one of them is mpC [5], just intended for networks of computers.

The main peculiarity of networks of computers differing them from MPPs, is their heterogeneity. A typical MPP consists of identical processor nodes interconnected via special equipment providing fast communications between the nodes. Performance of the network equipment is usually balanced with performance and the number of computing processor nodes in order to ensure an acceptable speedup when adding processors for typical scalable and quasi-scalable message-passing algorithms. At the same time, networks of computers are inherently heterogeneous and comprise diverse workstations, PCs, servers and, sometimes, specialized parallel computers interconnected via mixed network equipment.

The heterogeneity has two aspects having an impact on different properties of applications. On the one hand, the diversity of machine arithmetics in heterogeneous computing environment has an influence on such aspects of parallel numerical algorithms as convergence, accuracy, stability, etc. Practical experience of porting numerical software from MPPs to networks of workstations [6] has shown that the heterogeneity of floating point arithmetics often causes a wide range of failures — from erroneous results without warning to deadlock.

On the other hand, the heterogeneity of both performances of computing processor nodes and communication speeds/bandwidths of links has an essential impact on efficiency of applications. As a rule, performances of the processors and the network equipment are not balanced for high-performance parallel computing. Usually, a network of computers evolves rather spontaneously, and a lot of factors affect on its evolution, among which high-performance parallel computing is very seldom of paramount importance. Practical experience of developing applications in mpC to solve a wide range of scientific problems on heterogeneous networks of workstations and PCs [5], [7], has demonstrated that mpC-like programming tools allow to work out portable modular parallel applications, much faster solving both regular and irregular problems in heterogeneous environments than their counterparts, developed with traditional tools and ported from MPPs. Speedup was obtained because the mpC applications could dynamically adapt to heterogeneity of processor performances and link speeds, distributing data and computations among processors in such a way that ensured their efficient running.

The paper presents an application efficiently solving the practical problem of oil extraction simulation on heterogeneous networks of computers. We started from porting a PVM application, intended for execution on MPPs, to a heterogeneous network of workstations. Then we rewrote the application in mpC, slightly modifying the underlying algorithm. As a result the developed mpC application could distribute computations over executing processors in proportion to their performances.

Section 2 introduces the oil extraction problem and a parallel algorithm for its solution on homogeneous distributed memory multiprocessors. Section 3 discusses modifications of the algorithm aimed at efficient solving the problem on heterogeneous networks of computers and principles of the modified algorithm implementation in the mpC language as well as presents some experimental results. Section 4 concludes the paper.

2 Oil Extraction Problem and Its Solution on MPPs

The oil-extraction process by means of non-piston water displacement is governed by the classical 2D Bucley-Leverett model [8] which describes two-phase (oil/water) fluid filtration through a porous medium at the water-pumping regime and does not take into account capillary and gravity forces. The oil bearing stratum is assumed to be uniform, unbounded and thin so that the flat configuration of the stratum can be considered. Fluids are immiscible and incompressible; the porous medium is undeformable; oil and water flows are independent, unidirectional and determined by the Darcy law of filtration. The oil field is supposed to be covered by oil extracting wells (sinks) and water supplying wells (sources). Disposition schemes of wells may have different arrangements.

For calculations the Bucley-Leverett model is transformed to the next system of equations:

$$m\frac{\partial S_w}{\partial t} + \mathrm{div}\Big(F_w(S_w)K(S_w)\mathrm{grad}P\Big) = \begin{cases} q \times F_w(\bar{S}) & \text{– at sources} \\ q \times F_w(S_w) & \text{– in the whole domain} \end{cases} \tag{1}$$

$$\mathrm{div}\Big(K(S_w)\mathrm{grad}P\Big) = q \tag{2}$$

Where

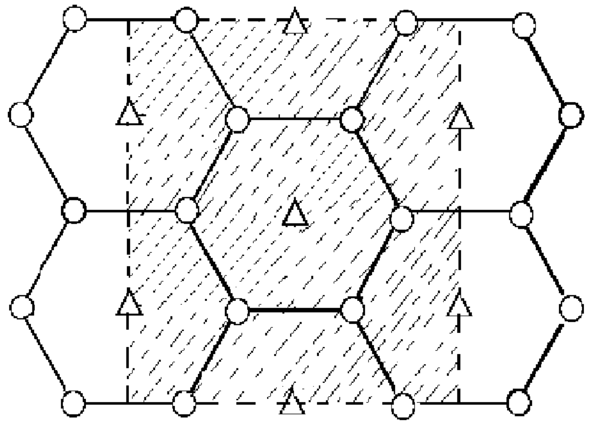
$$K(S_w) = -k\left(\frac{k_w(S_w)}{\mu_w} + \frac{k_o(S_w)}{\mu_o}\right), \tag{3}$$

$$F_w(S_w) = \frac{k_w(S_w)/\mu_w}{k_w(S_w)/\mu_w + k_o(S_w)/\mu_o}. \tag{4}$$

Initial and boundary conditions are

$$S_w|_{t=0} = \underline{S}, \quad P|_{t=0} = P_0, \tag{5}$$

$$\left.\frac{\partial S_w}{\partial n}\right|_{\Gamma} = 0, \quad \left.\frac{\partial P}{\partial n}\right|_{\Gamma} = 0. \tag{6}$$



△ — water supplying wells (sources)
○ — oil extracting wells (sinks)

Figure 1: "Honeycomb" well disposition scheme — the computational domain is shaded

Equation (1) is the water fraction transport equation and equation (2) is the pressure elliptic equation. Solutions of this system are water saturation S_w (fraction of water in the fluid flow) and pressure in the oil field P. These equations include coefficients for medium characteristics: the coefficient of porosity (m), the absolute permeability (k) and nonlinear relative phase permeabilities of oil ($k_o(S_w)$) and water ($k_w(S_w)$), the viscosities of oil (μ_o) and water (μ_w), the function of sources/sinks (q), critical and connected values of water saturation ($\bar{S}$ and $\underline{S}$) and the strongly nonlinear Bucley-Leverett function ($F_w(S_w)$).

Numerical solutions are sought in a domain with conditions of impermeability at the boundary (6). This domain is a subdomain of symmetry singled out from the unbounded oil field which is simulated.

The following parallel numerical algorithm [10] underlies the Fortran/PVM application solving this oil production problem on MPP Parsytec PowerXplorer. It is based on completely implicit methods of solving equations (1)-(2), namely, equation (1) is solved by the iterative secant method, meanwhile the $(\alpha - \beta)$-iterative algorithm [9] is employed to solve equation (2).

The $(\alpha - \beta)$-elliptic solver is an extension of the sweep method for the many-dimensional case. It does not need any a priori information about problem operators and is enough general-purpose. According to this algorithm the solution sought for is obtained via eight auxiliary functions ($\alpha, \beta, \gamma, \delta$ and $\bar{\alpha}, \bar{\beta}, \bar{\gamma}, \bar{\delta}$) calculated iteratively. It is profitable to include a relaxation parameter into equations for $\bar{\beta}$- and $\bar{\delta}$-coefficients in order to reduce the number of $(\alpha - \beta)$-iterations.

The standard seven-point ("honeycomb") scheme of oil/water well disposition depicted in Fig. 1 is simulated. The computational domain is approximated by uniform rectangular grid of 117×143 points. The parallel implementation is based on computational domain partitioning (data parallelization): the domain is divided into equal subdomains in one direction according to the Y-coordinate, with each subdomain being computed by a separate

Table 1: Performance of the Fortran/PVM parallel oil extraction simulator on the Parsytec PowerXplorer System

Number of Processors	ω	Number of Iterations	Time (sec)	Real Speedup	Efficiency
1	1.197	205	120	1	100%
2	1.2009	211	64	1.875	94%
4	1.208	214	38	3.158	79%
8	1.22175	226	26	4.615	58%

processor of an executing MPP. That domain distribution is more profitable for reducing the number of message-passing operations in the data parallel $(\alpha - \beta)$-algorithm than domain distributions according to the X-coordinate and according to the both coordinates. In each subdomain, system (1)-(2) is solved as follows. At every time level, the water saturation is obtained by solving equation (1) using pressure values from the previous time level. Then, employing the just found water saturation, new pressure is calculated at the present time level by solving equation (2). After that, this procedure is repeated at the next time level.

The main difficulty of this parallel algorithm is estimation of the optimal relaxation parameter ω for the $(\alpha - \beta)$-solver because this parameter varies while dividing the computational domain into different quantities of equal subdomains. Employing a wrong parameter leads to slow convergence or in some cases to non-convergence of $(\alpha - \beta)$-iterations. Numerous experiments allowed to find the optimal relaxation parameter for each number of subdomains.

Results of comprehensive study of the oil extraction problem were published in [10]. Parallel simulation of oil extraction processes for different dispositions of wells were performed and a number of technological characteristics of oil production were predicted in order to develop the optimal strategy of oil field exploitation. In Fig. 2 oil distribution (a) and pressure (b) are shown at the end of the first year of the oil field exploitation — for the "honeycomb" wells' configuration (see Fig. 1). At the right side of figure (a) there are the corresponding values of water saturation. For pressure the corresponding values (atmospheres) are shown near figure (b). Distance (meters) 20-times decreased is indicated at axises.

The above parallel algorithm was implemented in Fortran 77 with the PVM message-passing library as a communication platform and demonstrated good scalability, speedups and parallelization efficiency while running on the Parsytec PowerXplorer System — an MPP consisting of PowerPC 601 processors as computational nodes and T800 transputers as communicational nodes (one T800 provides four bi-directional 20 Mbits/sec communication links).

Table 1 presents some results of the experiments for one time level. Parallelization effi-

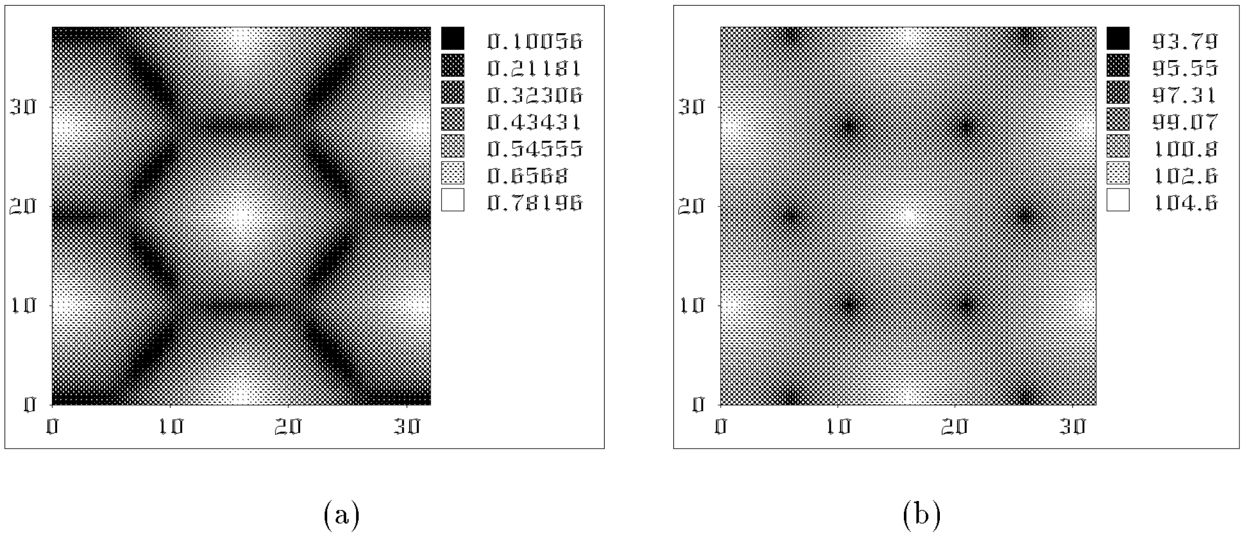


Figure 2: Oil (a) and pressure (b) distributions for "honeycomb" disposition of wells

ciency is defined as S_{real}/S_{ideal}, where S_{real} is the real speedup achieved by the parallel oil extraction simulator on the parallel system, and S_{ideal} is the ideal speedup that could be achieved while parallelizing the problem. The latter is calculated as the sum of performances of processors, constituting the executing parallel system, divided by the performance of a base processor. All speedups are calculated relative to the original sequential oil extraction simulator running on the base processor. Note, that the more powerful are communication links and the less powerful are processors, the higher efficiency is achieved.

3 Solving Oil Extraction Problem in Heterogeneous Environments

Generally speaking, the parallel oil extraction simulator is intended to be a part of a portable software system able to run on local networks of heterogeneous computers and providing a computerized working place of an oil extraction expert. Therefore, a portable application efficiently solving the oil extraction problem on networks of computers is needed.

As the first step toward such an application, we ported the above Fortran/PVM program to a local network of heterogeneous workstations based on 10 Mbits Ethernet. Our weakest workstation (SPARCclassic) executes the serial application a little bit slower than PowerPC 601, while the most powerful one (UltraSPARC-1) executes it more than six times faster. In general, we used nine uniprocessor workstations, and Table 2 shows their relative performances.

Table 3 shows some results of execution of the Fortran/PVM program on different two, four, six and eight-workstation subnetworks of the network. In this table, for every subnetwork the speedup is calculated relative to the running time of the serial program on the

Table 2: Relative performances of workstations

No. of Workstation	1	2	3	4	5	6	7	8	9
Performance	1150	575	460	460	325	325	325	170	170

Table 3: Performance of the Fortran/PVM parallel oil extraction simulator on subnetworks of workstations

Subnetwork (No. of Workstations)	ω	Number of Iterations	Time (sec)	Ideal Speedup	Real Speedup	Efficiency
{2, 5}	1.2009	211	46	1.57	0.88	56%
{5, 6}	1.2009	211	47	2.0	1.52	76%
{2, 5, 6, 7}	1.208	214	36	2.7	1.13	42%
{2, 3, 4, 5, 6, 7}	1.21485	216	32	4.3	1.27	30%
{2, 3, 5, 6, 7, 8}	1.21485	216	47	3.8	0.87	23%
{1, 2, 3, 4, 5, 6, 7, 8}	1.22175	226	46	3.3	0.41	12%

fastest workstation of the subnetwork. (The total run time of the serial oil extraction simulator when executing on different workstations can be found in Table 4.) Visible degradation of parallelization efficiency is explained by the following three reasons — slower communication links, faster processors, and not balanced workload of the workstations.

The first two reasons are unavoidable, while the latter is avoided by means of slight modification of the parallel algorithm implemented by the Fortran/PVM program. Namely, to provide the optimal load balancing, the computational domain is decomposed into subdomains of nonequal sizes, proportional to relative performances of participating processors. To be exact the number of grid columns in each subdomain is the same, while the number of rows differs. Regarding the relaxation parameter, it is reasonable to assume its optimal value to be a function of the number of grid rows and to use for each subdomain its own relaxation parameter $\omega = \omega(N_{row})$. While distributing the domain into different quantities of subdomains with equal numbers of grid rows a sequence of optimal relaxation parameters was found empirically. Now using the experimental data end piecewise linear interpolation, the optimal parameter ω can be calculated for any N_{row}. Note, that this approach gave high convergence rate and good efficiency of the parallel $(\alpha - \beta)$-solver with relaxation (see Numbers of Iterations in Table 5).

Table 4: Performance of the serial oil extraction simulator when running on workstations

No. of Workstation	1	2	3	4	5	6	7	8	9
Number of Iterations	205	205	205	205	205	205	205	205	205
Time (sec)	18.7	40.7	51.2	51.2	71.4	71.4	71.4	133	133

Table 5: Performance of the mpC parallel oil extraction simulator on subnetworks of workstations

Subnetwork (No. of Workstations)	Number of Iterations	Time (sec)	Real Speedup	Effic.	Time of 205 Itr (sec)	Speedup on 205 Itr	Effic. on 205 Itr
{2, 5}	324	41.6	0.98	63%	28.2	1.44	92%
{5, 6}	225	38.8	1.84	92%	36.4	1.96	98%
{2, 5, 6, 7}	279	26	1.57	58%	19.7	2.07	77%
{2, 3, 4, 5, 6, 7}	245	17.9	2.27	54%	15	2.71	63%
{2, 3, 5, 6, 7, 8}	248	20.2	2.01	54%	17	2.39	64%
{2, 3, 5, 6, 7, 8} ***)**	260	32.8	1.24	33%	26.8	1.52	40%
{2, 3, 4, 5, 6, 7, 8, 9}	268	21	1.94	40%	16	2.54	53%

***)** – the computational domain was forcedly divided on equal subdomains

Unfortunately, this modified algorithm can not be easily expressed in PVM in a portable form. The point is that PVM (as well as MPI or HPF) does not support creating a group of processes based on such properties of the created group as related speeds of processes or speeds of data transfer among processes. Therefore, we rewrote the algorithm in the mpC language. The mpC language is an ANSI C superset providing facilities that allow the user to explicitly specify properties of different groups of processes participating in the execution of different parts of the entire distributed program. In particular, it allows to detect in run time the number and relative performances of available processors, to create a group of processes in such a way that each process runs on a separate processor and to distribute data and computations in proportion to relative performances of the processors. The mpC application not only provided new functionalities but allowed to reduce the size of source code more than three times.

Table 5 shows empirical results of the mpC application execution on our network of workstations. In the experiments, the mpC programming environment used LAM MPI 6.0 as a communication platform. One can see that the mpC application demonstrates much higher efficiency in the heterogeneous environment than its PVM counterpart. To estimate pure contribution of the load balancing in the improvement of parallelization efficiency, we run the mpC application on subnetwork {2, 3, 5, 6, 7, 8} with the forcedly even domain decomposition resulting in essential (more than 1.5 times) efficiency degradation (compare rows 5 and 6 in Table 5).

4 Conclusion

The paper has presented an approach to efficient solving problems on heterogeneous networks of computers. The approach consists in the development of such applications that adapt in run time both to performances of processors and to speeds of communication links interconnecting the processors of any particular computing heterogeneous environment. To implement such applications, an appropriate programming tool is necessary. We used the mpC programming language just aimed at efficient portable parallel programming heterogeneous distributed memory machines. In the framework of the paper, we dealt only with processor performances and proposed a data distribution taking into account this aspect of heterogeneity. It is easy to see that often the best data distribution essentially depends on speeds of communication links. Indeed, in case of slow links, communication latencies can exceed parallelization speedup, slowing down solving the problem. Therefore, it can make sense to use not all available processors, but only some part of them. In other words, the corresponding heterogeneous data distribution does not distribute data to some processors, and the application does not use those processors for computations. Currently, we are working on an mpC application, simulating oil extraction, that takes into account not only processor performances, but also speeds of communication links.

References

[1] H. El-Rewini, and T. Lewis, *Introduction To Distributed Computing*, Manning Publications Co., 1997.

[2] A. Geist, A. Beguelin, J. Dongarra, W. Jlang, R. Manchek, V. Sunderam, *PVM: Parallel Virtual Machine, Users' Guide and Tutorial for Networked Parallel Computing* , MIT Press, Cambridge, MA, 1994.

[3] Message Passing Interface Forum, *MPI: A Message-passing Interface Standard, version 1.1* , June 1995.

[4] High Performance Fortran Forum, *High Performance Fortran Language Specification, version 2.0* , Rice University, Houston TX, January 31, 1997.

[5] D. Arapov, A. Kalinov, A. Lastovetsky, I. Ledovskih, and T. Lewis, "A Programming Environment for Heterogeneous Distributed Memory Machines", *Proceedings of the Sixth Heterogeneous Computing Workshop (HCW'97) of the 11th International Parallel Processing Symposium (IPPS'97)* , IEEE CS Press, Geneva, Switzerland, April 1997, pp.32-45.

[6] L. Blackford, A. Cleary, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, A. Petitet, H. Ren, K. Stanley, and R. Whaley, *Practical Experience in the Dangers of Heterogeneous Computing UT* , CS-96-330, July 1996.

[7] D. Arapov, A. Kalinov, A. Lastovetsky, I. Ledovskih, "Experiments with mpC: Efficient Solving Regular Problems on Heterogeneous Networks of Computers via Irregularization", *Proceedings of the Fifth International Symposium On Solving Irregularly Structured Problems in Parallel (IRREGULAR'98)* , to appear.

[8] S.E. Bucley and M.C. Leverett, "Mechanism of fluid displacement in sands", *Trans. AIME*, **146**, 1942.

[9] B.N. Chetverushkin and N.G. Churbanova, "Solution of elliptic equations using the iterative $(\alpha - \beta)$-solver and domain decomposition technique", *Parallel Computing Technologies* (PACT-93) (Ed. V. Malyshkin), pp.97–101, Obninsk, Russia, 1993.

[10] B.N. Chetverushkin, N.G. Churbanova and M.A. Trapeznikova, "Simulation of oil production on parallel computing systems", *HPC'97: Grand Challenges in Computer Simulation. Proc. of Simulation MultiConference*, April 6-10, 1997, Atlanta, USA (Ed. A.Tentner), pp.122–127, SCS, San Diego, CA, 1997.