Designing a Parallel Programs on the Base of the Conception of Q-determinant

Valentina Aleeva[0000−0002−4717−0045]

South Ural State University (National Research University),
Chelyabinsk, Russia
aleevavn@susu.ru

Abstract. The paper describes a design method of parallel programs for numerical algorithms based on their representation in the form of Q-determinant. The result of the method is Q-effective program. It uses the parallelism resource of the algorithm completely. The results of this research can be applied to increase the implementation efficiency of algorithms on parallel computing systems. This should help to improve the performance of parallel computing systems.

Keywords: Q-determinant of algorithm. Algorithm representation as Q-determinant. Q-effective implementation of algorithm. Parallelism resource of algorithm. Parallel computing system. Parallel program. Q-effective program.

1 Introduction

The development of parallel computing systems has the history for decades, but the implementation effectiveness of algorithms remains low on those. That problem can be solved by using parallelism resource of algorithms completely. The conception of Q-determinant [1] allows to detect the parallelism resources of numerical algorithms. The basis of the conception is the universal description of algorithms. This is the algorithms representation in the Q-determinant form. All of the algorithm implementation are described by Q-determinant, including the Q-effective one. The Q-effective implementation uses the parallelism resource of the algorithm completely. From a formal point of view that is the most rapid implementation. The description of the method of designing a parallel program for the Q-effective implementation of numerical algorithm is the aim of this paper. The concept of the Q-determinant has been theoretical development and used to study the resource of algorithm parallelism in the papers [2,3,4,5,6]. This conception for the design of parallel program is offered for the first time.

The investigation of parallel structure of algorithms and programs is very important and highly developed for their implementation on parallel computer systems. The basis of research is described in [7,8]. Representations by graphs are used for a description of parallel algorithms. The Internet encyclopedia AlgoWiki [9] is created nowadays. The encyclopedia describes the features, peculiar properties, static and dynamic characteristics of the algorithms. This help to implement algorithms effectively. In the report [10] there is the current state of researches of parallelization algorithms and their implementations on parallel computing systems. The report contains the formulations of some problems. Here are some of these problems. How to represent a potentially infinite graph? How to represent a potentially multi-dimensional graph? How to show dependence graph structure on the size of the tasks? How to express the available parallelism and show affordable way of the parallel execution? The conception of Q-determinant gives answers for these questions. This research contains one of the answers to the last question.

There are many studies, in which the specific nature of algorithms and the architecture of parallel computing systems take into account for the development of parallel programs. Examples of such studies are [11,12,13]. The efficiency of implementing specific algorithms or implementing algorithms on parallel computing systems of a particular architecture is increased in cases of those studies. However, they do not provide general universal approach. The parallel program synthesis is other approach to creating parallel programs also. The synthesizing parallel programs method is to construct new parallel algorithms using the knowledge base of parallel algorithms for solving more complex problems. The technology of fragmented programming and its implementing language and programming system LuNA are developed on the base of synthesizing parallel programming method. This direction of research is developing [14,15] at present time. The approach is universal, but it does not solve the problem of research and use of the algorithm parallelism resource. The investigation of the parallelism resource of algorithms is provided using their software implementation [16]. If you want to solve the problem of determining the parallelism resource of algorithm, then use of any program implementing of algorithm may be wrong because that program can not contain all implementations of the algorithm. In particular, the Q-effective implementation can be lost under program creation. We can notice that the analysis of the existing approaches of problem solution of studying the parallelism resource of algorithm and its implementation on parallel computing systems shows they are inapplicable, or ineffective, or non-generic. The approach is perspective if it is based on the universal description of the algorithm showing the parallelism resource in full. For example, such approach is an approach on the base of the Q-determinant concept.

2 Q-determinant of Algorithm

Let $\mathcal{A}$ be an algorithm for solving an algorithmic problems $\bar{y} = F(N, B)$, where N is a set of dimension parameters of the problem, B is a set of input data, $\bar{y} = (y_1, \ldots, y_m)$ is a set of output data, $y_i \notin B$ $(i = 1, \ldots, m)$, m is a computable function parameters N on condition $N \neq \varnothing$ or constant.

The set N satisfies the conditions: $N = \varnothing$, or $N = \{n_1, \ldots, n_k\}$, where $k \geqslant 1$, n_i $(1 \leqslant i \leqslant k)$ are every positive integers. If $N = \{n_1, \ldots, n_k\}$ then we denote vector $(\bar{n}_1, \ldots, \bar{n}_k)$, where $\bar{n}_i$ is some assigned value of parameter n_i $(1 \leqslant i \leqslant k)$ by $\bar{N}$. We denote by $\{\bar{N}\}$ the set $\bar{N}$ of possible vectors.

Let is Q a set of operations those are used by algorithm $\mathcal{A}$. Assume that the operations of Q are 0-ary (constant), unary or binary. An example of a set Q is a set of arithmetic, logic operations and comparison operations. The expressions can be formed by sets B and Q. We call chain an expression that obtained from the n expressions with the help of use of $n-1$ times one of associative operations of Q.

One of the basic notions of Q-determinant conception is Q-term.

Definition of Q-term:

1. If $N = \varnothing$ then unconditional Q-term is called every expression w over B and Q (term of signature Q). Let V be a set of all expressions over B and Q. If $N \neq \varnothing$ then every map $w : \left\{\bar{N}\right\} \to V$ is called unconditional Q-term also.
2. Let $N = \varnothing$ and be given an unconditional Q-term w. Let under each interpretation of B the expression w over B and Q have a logical type value. Then unconditional Q-term w is called unconditional logical Q-term. Let $N \neq \varnothing$ and be given an unconditional Q-term $w : \left\{\bar{N}\right\} \to V$. Let be an expression $w(\bar{N})$ for every $\bar{N} \in \{\bar{N}\}$ have logical type value under each interpretation of B. Then unconditional Q-term w is called unconditional logical Q-term.
3. Let $u_1, \ldots, u_l$ be an unconditional logical Q-terms. $w_1, \ldots, w_l$ are an unconditional Q-terms. We denote the set of pairs (u_i, w_i) $(i = 1, \ldots, l)$ as $(\bar{u}, \bar{w}) = \{(u_i, w_i)\}_{i=1,\ldots,l}$ and call conditional Q-term of length l.
4. Assume we have a countable set of pairs unconditional Q-terms $(\bar{u}, \bar{w}) = \{(u_i, w_i)\}_{i=1,2,\ldots}$ such that $\{(u_i, w_i)\}_{i=1,\ldots,l}$ is conditional Q-term for each $l < \infty$ then we call it conditional infinite Q-term.
5. If it does not matter whether the Q-term unconditional, conditional or conditional infinite then we call it Q-term.

Q-terms can be calculated.

We mean by the calculation of unconditional Q-term w under each interpretation of B as the calculation of the expression of w if $N = \varnothing$ and the calculation of the expression of $w(\bar{N})$ for some $\bar{N} \in \{\bar{N}\}$ if $N \neq \varnothing$.

We describe the calculation of conditional Q-term $(\bar{u}, \bar{w}) = \{(u_i, w_i)\}_{i=1,\ldots,l}$ under any interpretation of B. If $N = \varnothing$ is necessary to calculate the expressions u_i, w_i $(i = 1, \ldots, l)$. If there are expressions of u_{i_0}, w_{i_0} $(i_0 \leqslant l)$ such that u_{i_0} takes the value `true` and the value of w_{i_0} is determined we will set $(\bar{u}, \bar{w})$ taking value is equal to w_{i_0}. Also we assume the value of $(\bar{u}, \bar{w})$ for this interpretation B is not determined otherwise. If $N \neq \varnothing$ then we set value $\bar{N} \in \{\bar{N}\}$. We obtain the expressions $u_i(\bar{N}), w_i(\bar{N})$ $(i = 1, \ldots, l)$ and calculate them. If there are expressions of $u_{i_0}(\bar{N}), w_{i_0}(\bar{N})$ $(i_0 \leqslant l)$ such that $u_{i_0}(\bar{N})$ takes the `true` and the value of $w_{i_0}(\bar{N})$ is determined we will set $(\bar{u}, \bar{w})$ taking value is equal to $w_{i_0}(\bar{N})$. Also we assume the value of $(\bar{u}, \bar{w})$ is not determined for given $\bar{N}$ and interpretation of B otherwise.

We describe for given interpretation of B the calculation of conditional infinite Q-term $(\bar{u}, \bar{w}) = \{(u_i, w_i)\}_{i=1,2,\ldots}$. If $N = \varnothing$ it is necessary to find u_{i_0}, w_{i_0} such that u_{i_0} is set to `true` and the value of w_{i_0} is determined. Then w_{i_0} is the value of $(\bar{u}, \bar{w})$. If we have no such expressions u_{i_0}, w_{i_0} the value of $(\bar{u}, \bar{w})$ is not determined for this interpretation B. Similarly, we can define calculation of conditional infinite Q-term in the case of $N \neq \varnothing$.

Suppose that I_1, I_2, I_3 are subsets of the set $I = (1, \dots, m)$, satisfying the following conditions:

1. $I_1 \cup I_2 \cup I_3 = I$;
2. $I_i \cap I_j = \varnothing$ $(i \neq j; i, j = 1, 2, 3)$;
3. One or two subsets I_i $(i = 1, 2, 3)$ may be empty.

We consider the set of Q-terms $\{f_i\}_{i \in I}$ such that:

1. f_{i_1} $(i_1 \in I_1)$ is an unconditional Q-term, $f_{i_1} = w^{i_1}$;
2. f_{i_2} $(i_2 \in I_2)$ is conditional Q-term, $f_{i_2} = \left\{(u_j^{i_2}, w_j^{i_2})\right\}_{j=1,\dots,l_{i_2}}$, l_{i_2} is either constant or computable function of parameters N for $N \neq \varnothing$;
3. f_{i_3} $(i_3 \in I_3)$ is a conditional infinite Q-term, $f_{i_3} = \left\{(u_j^{i_3}, w_j^{i_3})\right\}_{j=1,2,\dots}$.

Suppose that the algorithm $\mathcal{A}$ is that Q-term f_i should be computed in order that y_i $(i \in I)$ evaluates. Then the set of Q-terms f_i $(i \in I)$ is called a Q-determinant of algorithm $\mathcal{A}$ and presentation of algorithm in the form $y_i = f_i$ $(i \in I)$ is called a presentation of the algorithm in the form of Q-determinant. Every numerical algorithm can be represented in the form Q-determinant.

3 Q-effective Implementation of Algorithm

Let $\mathcal{A}$ be an algorithm in the form of Q-determinant $y_i = f_i$ $(i \in I)$. The process of calculating the Q-terms f_i $(i \in I)$ is called an implementation of the algorithm $\mathcal{A}$. If the implementation of the algorithm is such that two or more operations are performed simultaneously, it will be called a parallel implementation. We describe a realization algorithm $\mathcal{A}$ represented in the form of Q-determinant.

Let $N = \varnothing$. We specify the variable interpretation of B. Expressions

$$W = \big\{w^{i_1}(i_1 \in I_1); u_j^{i_2}, w_j^{i_2}(i_2 \in I_2, j = 1, \dots, l_{i_2}); \\ u_j^{i_3}, w_j^{i_3}(i_3 \in I_3, j = 1, 2, \dots)\big\} \quad (1)$$

will be calculated at the same time (in parallel). We say that the operation is ready to perform if you have calculated the value of its operands already. In calculating each of the expressions W we perform the operations as soon as they are ready to perform.

If you are ready to perform several operations chain, they are calculated by doubling scheme. For example, the doubling scheme of calculating the chain $a_1 + a_2 + a_3 + a_4$ is the following. First, we calculate $b_1 = a_1 + a_2$ and $b_2 = a_3 + a_4$ simultaneously, after that we calculate $c = b_1 + b_2$.

If we obtain `false` value for some expression u_j^i $(i \in I_2 \cup I_3, j = 1, 2, \dots)$ the calculation of the corresponding expression of w_j^i comes to an end. If the calculation of some pair of expressions (u_j^i, w_j^i) $(i \in I_2 \cup I_3, j = 1, 2, \dots)$ implies that the value of one of the expressions is not defined then the calculation of other expression is terminated. If the calculation of some pair of expressions $(u^i_{j_0}, w^i_{j_0})$ $(i \in I_2 \cup I_3)$ finds that their values define and $u^i_{j_0}$ is set to `true`, the calculation of expressions u^i_j, w^i_j $(i \in I_2 \cup I_3, j = 1, 2, \ldots; j \neq j_0)$ stops. Calculation the identical expressions of u^i_j, w^i_j $(i \in I_3, j = 1, 2, \ldots)$ and their identical subexpressions may not duplicate.

Let N be nonempty. We define $\bar{N} \in \{\bar{N}\}$ and the interpretation of the variables of B . We get the set of expressions

$$W(\bar{N}) = \bigl\{w^{i_1}(\bar{N})(i_1 \in I_1); u^{i_2}_j(\bar{N}), w^{i_2}_j(\bar{N})(i_2 \in I_2, j = 1, \ldots, l_{i_2}); \\ u^{i_3}_j(\bar{N}), w^{i_3}_j(\bar{N})(i_3 \in I_3, j = 1, 2, \ldots)\bigr\}. \quad (2)$$

Expressions $W(\bar{N})$ can be calculated by analogy with the expressions W.

Described implementation of the algorithm $\mathcal{A}$ will be called Q-effective. We say that the implementation of the algorithm $\mathcal{A}$ is realizable, if finite number of operations needed to be performed simultaneously.

4 The Method of Parallel Program Design for the Q-effective Implementation of Algorithm

Q-determinants can be constructed for any numerical algorithm. Q-determinant allows us to describe Q-effective implementation of the algorithm. If Q-effective implementation of the algorithm is realizable it can be programmed directly. You can develop a sequential program using a flow chart of a numerical algorithm. Also you can develop a parallel program using the Q-determinant of a numerical algorithm. This idea is the basis of the proposed method. The model of the Q-determinant concept allows us to investigate machine-independent properties of algorithms just only. The basic model of the Q-determinant concept is expanded to take into account the features of implementing algorithms on real parallel computing systems. The extended model of the Q-determinant concept obtained by adding model of parallel computing PRAM [17] for shared memory and BSP [18] for distributed memory. We proposed the method of parallel program design for the Q-effective implementation of the algorithm that is based on the extended model of the Q-determinant concept.

The method consists of the following stages.

1. Construction of Q-determinant of algorithm.
2. Description of Q-effective implementation of algorithm.
3. If Q-effective implementation is realizable then a parallel program is developed for it.

The program will be called Q-effective if it is designed with the help of this method. Q-effective program uses the parallelism resource of algorithm completely because it performs a Q-effective implementation of the algorithm. So, it has most high parallelism among the programs implement the algorithm. For this reason the Q-effective program uses the resources of the computing system more efficient than the programs perform other implementations of the algorithm. A Q-effective program can be used parallel computing systems with shared and distributed memory. We show the use of this method for modeling algorithms, which have Q-determinants that consist of Q-terms of various types. First, consider the first two stages of the method.

The algorithm of matrix multiplication

Stage 1. Consider the algorithm of multiplication of matrices

$$A = [a_{ij}]_{i=1,\ldots,n;j=1,\ldots,k} \text{ and } B = [b_{ij}]_{i=1,\ldots,k;j=1,\ldots,m}. \tag{3}$$

The result is matrix $C = [c_{ij}]_{i=1,\ldots,n;j=1,\ldots,m}$, where $c_{ij} = \sum_{s=1}^{k} a_{is}b_{sj}$.
The algorithm of matrix multiplication can be presented in the form of Q-determinant. Q-determinant is composed of nm unconditional Q-terms.

Stage 2. Q-effective implementation of the algorithm for multiplication of the matrices is that all of Q-terms $\sum_{s=1}^{k} a_{is}b_{sj} (i = 1, \ldots, n; j = 1, \ldots, m)$ are calculated simultaneously. First, all multiplication operations should are ready to perform, so they need to be performed simultaneously. The result will nm chains formed by the operation addition. Each chain is calculated by doubling scheme. So, Q-effective implementation of the algorithm for multiplication of matrices is realizable.

Gauss–Jordan method of solving of linear equation systems

Stage 1. Gauss-Jordan method of solving linear equation systems $A\bar{x} = \bar{b}$ can be applied to each dimension. For simplicity, we assume that $A = [a_{ij}]$ is a matrix of dimension $n \times n$ with a nonzero determinant. $\bar{x} = (x_1, \ldots, x_n)^T$, $\bar{b} = (a_{1,n+1}, \ldots, a_{n,n+1})^T$ has a column-vectors, $\bar{A}$ is augmented matrix of the system. We construct Q-determinant method of Gauss-Jordan.

Gauss-Jordan method consists of n steps.

Step 1.

We select element a_{1j_1} with properties a_{11} if $a_{11} \neq 0$ otherwise $a_{1j} = 0$ for $j < j_1 \leqslant n$ and $a_{1j_1} \neq 0$ as leading element. We get augmented matrix $\bar{A}^{j_1} = [a_{ij}^{j_1}]$, whose elements are calculated by rules

$$a_{1j}^{j_1} = \frac{a_{1j}}{a_{1j_1}},\ a_{ij}^{j_1} = a_{ij} - \frac{a_{1j}}{a_{1j_1}} a_{ij_1} (i = 2, \ldots, n; j = 1, \ldots, n+1). \tag{4}$$

Step k $(2 \leqslant k \leqslant n)$.

We get augmented matrix $\bar{A}^{j_1 \cdots j_{k-1}}$ after we made the step $(k-1)$. We select element $a_{kj_k}^{j_1 \cdots j_{k-1}}$ with properties $a_{k1}^{j_1 \cdots j_{k-1}}$ if $a_{k1}^{j_1 \cdots j_{k-1}} \neq 0$ otherwise $a_{kj}^{j_1 \cdots j_{k-1}} = 0$ for $j < j_k \leqslant n$ and $a_{kj_k}^{j_1 \cdots j_{k-1}} \neq 0$ as leading element. We get augmented matrix $\bar{A}^{j_1 \cdots j_k} = [a_{ij}^{j_1 \cdots j_k}]_{i=1,\ldots,n;j=1,\ldots,n+1}$, whose elements are calculated by the rules

$$a_{kj}^{j_1 \cdots j_k} = \frac{a_{kj}^{j_1 \cdots j_{k-1}}}{a_{kj_k}^{j_1 \cdots j_{k-1}}},\ a_{ij}^{j_1 \cdots j_k} = a_{ij}^{j_1 \cdots j_{k-1}} - \frac{a_{kj}^{j_1 \cdots j_{k-1}}}{a_{kj_k}^{j_1 \cdots j_{k-1}}} a_{ij_k}^{j_1 \cdots j_{k-1}}$$
$$(i = 1, \ldots, n; i \neq k; j = 1, \ldots, n+1). \tag{5}$$

We get system of equations $A^{j_1\cdots j_n}\bar{x} = \bar{b}^{j_1\cdots j_n}$ after step n, where

$$A^{j_1\cdots j_n} = [a_{ij}^{j_1\cdots j_n}]_{i=1,\ldots,n;j=1,\ldots,n}, \bar{b}^{j_1\cdots j_n} = (a_{1,n+1}^{j_1\cdots j_n}, \ldots, a_{n,n+1}^{j_1\cdots j_n})^T. \tag{6}$$

We denote by

$$L_{j_1} = \bigwedge_{j=1}^{j_1-1} (a_{1j} = 0) \text{ if } j_1 \neq 1,\ L_{j_1} = \texttt{true} \text{ if } j_1 = 1, \tag{7}$$

$$L_{j_l} = \bigwedge_{j=1}^{j_l-1} (a_{lj}^{j_1\cdots j_{l-1}} = 0) \text{ if } j_l \neq 1, L_{j_l} = \texttt{true} \text{ if } j_l = 1\ (l = 2, \ldots, n). \tag{8}$$

Permutations of elements $(1, \ldots, n)$ may be numbered. Let i be a number of permutation $(j_1, \ldots, j_n)$. Then the terms

$$w_i^{j_l} = a_{l,n+1}^{j_1\cdots j_n} (l = 1, \ldots, n), \tag{9}$$

$$u_i = L_{j_1} \wedge (a_{1j_1} \neq 0) \wedge \left(\bigwedge_{l=2}^{n} \left(L_{j_l} \wedge \left(a_{lj_l}^{j_1\cdots j_{l-1}} \neq 0 \right) \right) \right) \tag{10}$$

are unconditional Q-terms.

Q-determinant of Gauss–Jordan method consists of n conditional Q-terms and

$$x_j = \{(u_1, w_1^j), \ldots, (u_{n!}, w_{n!}^j)\} (j = 1, \ldots, n) \tag{11}$$

is the representation in the form of Q-determinant.

Stage 2. By the definition of Q-effective implementation all unconditional Q-terms $\{u_i, w_i^j\}(i{=}1, \ldots, n!; j{=}1, \ldots, n)$ should be calculated simultaneously. Therefore, two computational process should be carried out at the same time: parallel calculation of matrices $\bar{A}^{j_1}, \bar{A}^{j_1 j_2}, \ldots, \bar{A}^{j_1 j_2 \cdots j_n}$ for all possible values of the numbers $j_1, j_2, \ldots, j_n$, as well as a parallel calculation of the Q-terms $u_i(i{=}1, \ldots, n!)$. The leading elements of the matrix for each step of the algorithm are determined in the calculation of Q-terms $u_i(i{=}1, \ldots, n!)$ successively. Calculation of matrices $\bar{A}^{j_1}, \bar{A}^{j_1 j_2}, \ldots, \bar{A}^{j_1 j_2 \cdots j_n}$ stops if they do not correspond to the leading elements. The first cycle of calculations is as follows. We begin to calculate the matrices $\bar{A}^{j_1}$ and Q-terms $u_i(i = 1, \ldots, n!)$ simultaneously. The calculations $u_i(i{=}1, \ldots, n!)$ start with the subexpressions $L_{j_1} \wedge (a_{1j_1} \neq 0)(j_1{=}1, \ldots, n)$, because only their operations are ready for execution. Only one of the subexpressions $L_{j_1} \wedge (a_{1j_1} \neq 0)$ will be set to `true`. Let r_1 be a value of j_1. Further we end calculating $\bar{A}^{j_1}(j_1 = 1, \ldots, n)$ and $u_i(i = 1, \ldots, n!)$ under the condition that j_1 is not equal r_1. We compute the matrices $\bar{A}^{r_1 j_2}(j_2 = 1, \ldots, n; j_2 \neq r_1)$ and Q-terms $u_i(i = 1, \ldots, n!)$ to $j_1 = r_1$ in the second cycle of calculations simultaneously. In the calculation of $u_i(i = 1, \ldots, n!)$ one should be calculated only subexpression $L_{j_2} \wedge (a_{2j_2}^{r_1} \neq 0)(j_2 = 1, \ldots, n; j_2 \neq r_1)$ so as soon as their operations are ready for execution. Only one of the subexpressions $L_{j_2} \wedge (a_{2j_2}^{r_1} \neq 0)(j_2 = 1, \ldots, n; j_2 \neq r_1)$ will be set to `true`. Let r_2 be a value of j_2. Further, the calculating $\bar{A}^{r_1 j_2}(j_2 =$

$1, \ldots, n; j_2 \neq r_1$) and $u_i (i = 1, \ldots, n!)$ to $j_2 \neq r_2$ is stop. The following $n - 3$ cycles of calculations are executed similarly. In conclusion, you need to calculate a single matrix $\bar{A}^{r_1 \cdots r_{n-1} j_n} (j_n \neq r_1, r_2, \ldots, r_{n-1})$ as the parameter j_n has single value. Let r_n be a value of j_n. The result is $x_{r_j} = a^{r_1 \ldots r_n}_{j,n+1} (j = 1, \ldots, n)$ that is the solution of original system of linear equations. Q-effective implementation of the method of Gauss-Jordan is realizable.

Jacobi method of solving a system of linear equations

Stage 1. We construct Q-determinant of the Jacobi method of solving a system of linear equations $A\bar{x} = \bar{b}$, where $A = [a_{ij}]_{i,j=1,\ldots,n}$, $a_{ii} \neq 0 \quad (i = 1, \ldots, n)$, $\bar{x} = (x_1, \ldots, x_n)^T$, $\bar{b} = (a_{1,n+1}, \ldots, a_{n,n+1})^T$. We denote as $c_{ij} = -\frac{a_{ij}}{a_{ii}}$ and $d_i = \frac{b_i}{a_{ii}}$. Let $\bar{x}^0$ be an initial approximation. Then the iteration process can be written as $x_i^{k+1} = \sum_{j=1,\ldots,n; j \neq i} c_{ij} x_j^k + d_i (i = 1, \ldots, n; k = 0, 1, \ldots)$. The criterion of the iterative process ending is the condition $||\bar{x}^{k+1} - \bar{x}^k|| < \varepsilon$. There ε is the calculation precision.
Q-determinant of the Jacobi method consists of n conditional infinite Q-terms. Presentation of the Jacobi method in the form of Q-determinant is written as

$$x_i = \{(||\bar{x}^1 - \bar{x}^0|| < \varepsilon, x_i^1), \ldots, (||\bar{x}^k - \bar{x}^{k-1}|| < \varepsilon, x_i^k), \ldots\} (i = 1, \ldots, n). \quad (12)$$

Stage 2. We denote $u^l = ||\bar{x}^l - \bar{x}^{l-1}|| < \varepsilon (l = 1, 2, \ldots)$ that simplifies the description of Q-effective implementation. Then the Q-determinant has the form $x_i = \{(u^1, x_i^1), (u^2, x_i^2), \ldots, (u^k, x_i^k), \ldots\} (i = 1, \ldots, n)$. All unconditional Q-terms $\{u^l, x_i^l\} (i = 1, \ldots, n; l = 1, 2, \ldots)$ by the definition of Q-effective implementation should be calculated simultaneously. At first Q-terms $x_i^1 (i = 1, \ldots, n)$ should be calculated simultaneously. Then Q-terms $u^1, x_i^2 (i = 1, \ldots, n)$ are calculated simultaneously. If the value of u^1 is `true` then the calculation should be finished. In this case $x_i = x_i^1 (i = 1, \ldots, n)$ is the solution of system of linear equations. If the computation will continue the Q-terms $u^k, x_i^{k+1} (i = 1, \ldots, n)$ are calculated at the same time for any value of $k \geqslant 2$. If the value of u^k is `t`rue then the calculation should be finished. In this case $x_i = x_i^k (i = 1, \ldots, n)$ is the solution of system of linear equations. So, Q-effective implementation of the method of Jacobi is realizable.

Stage 3 of our proposed method is a parallel program development for the Q-effective implementation of the algorithm. To make this we should be used for parallel programming tools. We use a description of the Q-effective implementation of the algorithm for developing a Q-effective program for shared memory. The description of the Q-effective implementation of the algorithm should be supplemented with a description of the distribution of computation by computing nodes for development of Q-effective program for distributed memory. If we use distributed memory computing research is limited by a principle of a “master-slaves”. That principle is used on cluster computing systems often. The principle can be described as follows. To compute we use node M (Master) and several computing nodes S (Slave). The computational process is divided into several steps.
Step 0. Initialization.
Step 1. Make a task from the node M to all nodes of S.

Step 2. Make the calculating on the each node of S without exchanges with other nodes.
Step 3. Sending the results of all the nodes S to M node.
Step 4. Merge the results on the node M.
We describe the development features of Q-effective programs using distributed memory.

The algorithm of matrix multiplication

Each Q-term $\sum_{s=1}^{k} a_{is}b_{sj}(i = 1, \dots, n; j = 1, \dots, m)$ is calculated on your compute node of S. If number of nodes of S is less than the number Q-terms then some node of S can be calculated several Q-terms. The result of the calculation of each Q-term is transmitted to the node M.

Gauss–Jordan method of solving of linear equation systems

Each matrix $\bar{A}^{j_1}(j_1 = 1, \dots, n)$ and the corresponding Q-term $u_i(i{=}1, \dots, n!)$ are to be calculated at its node of S. If the number of nodes S is less than n then nodes of S should perform calculations for several values of j_1. Nodes of S receive information from a node M to compute matrices $\bar{A}^{j_1}(j_1 = 1, \dots, n)$ and corresponding Q-terms $u_i(i = 1, \dots, n!)$. Results of calculation r_1 and $\bar{A}^{r_1}$ are transmitted to node M. Nodes of S receive information from a node M to compute matrices $\bar{A}^{r_1 j_2}(j_2 = 1, \dots, n; j_2 \neq r_1)$ and corresponding Q-terms $u_i(i = 1, \dots, n!)$. Each matrix $A^{r_1 j_2}(j_2 = 1, \dots, n; j_2 \neq r_1)$ and the corresponding Q-term $u_i(i = 1, \dots, n!)$ are to be calculated at its node of S. Results of calculation r_2 and $\bar{A}^{r_1 r_2}$ are transmitted to node M. The following $n - 2$ cycles of calculations are executed similarly.

Jacobi method of solving a system of linear equations

Each component of the vector $\bar{x}^k(k = 1, 2, \dots)$ is calculated on different compute node S. If the number of nodes of S is less than n then nodes S should perform the calculations for several components of vector $\bar{x}^k(k = 1, 2, \dots)$. At first node M sends to the nodes of S the necessary information to calculate the Q-terms $x_i^1(i = 1, \dots, n)$. Results of calculating are transmitted to node M. Node M sends values $x_i^1(i = 1, \dots, n)$ to the nodes S. Node M calculates the value of $||\bar{x}^1 - \bar{x}^0|| < \varepsilon$. At the same time $x_i^2(i = 1, \dots, n)$ are computed on the nodes of S simultaneously. Next iterations are executed similarly.

At the present time Q-effective programs are designed for the considered algorithms. To develop programs we use the programming language C. OpenMP technology is used for shared memory, MPI and OpenMP are used for distributed memory. The research was performed on the supercomputer “Tornado” of South Ural State University. The programs for shared memory were executed on one compute node. Programs for distributed memory used several compute nodes. Dynamic characteristics of the programs were evaluated. The students of South Ural State University N. Val’kevich [19], D. Tarasov [20] and Yu. Lapteva [21] developed the programs and found the evaluations of their characteristics.
Figure 1 shows the efficiency of Q-effective programs of the matrix multiplication algorithm for shared (top graph) and distributed memory (lower graph) [19].

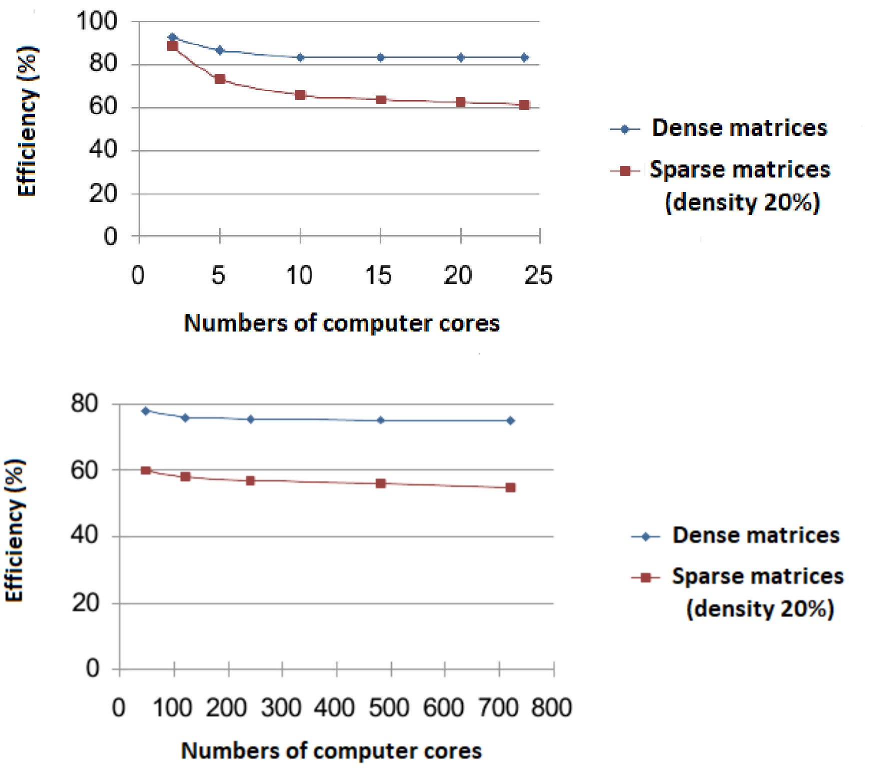

Fig.1. The efficiency of Q-efficient programs of the matrix multiplication algorithm for shared (top) and distributed memory (bottom)

The study of the Q-effective implementation of the algorithm shows that implementation of principle “master-slaves” for some algorithms impractical because it increases the amount of transfers between computing nodes. An example of such algorithm might be Jacobi method for solution of five-point difference equations. The study also reveals that usage of distributed memory is impractical for some algorithms because it increases execution time compare to usage of shared memory. An example is sweep method which is used for solution of three-point difference equations.

5 Conclusion

This method of designing a parallel program gives the possibility of the development of Q-effective program for numerical algorithms. Q-effective program uses the parallelism resource of the algorithm completely. The method can be used for the development of libraries of parallel programs for different classes of numerical algorithms.

Q-determinant makes the numerical algorithm clearer in terms of structure and implementation. In particular, it makes an opportunity to show the existing parallelism of the algorithm and the possible way of its parallel execution. So we can develop efficient implementations of algorithms for real parallel computing systems.

Acknowledgements

The reported study was funded by RFBR according to the research project № 17-07-00865 a. The work was supported by Act 211 Government of the Russian Federation, contract № 02.A03.21.0011.

References

1. Aleeva, V.N.: Analysis of parallel numerical algorithms: Preprint No. 590. Novosibirsk, Computing Center of the Siberian Branch of the Academy of Sciences of the USSR (1985)
2. Ignatyev, S.V.: Definition of parallelism resource of algorithms on the base of the concept of Q-determinant. In: Scientific Service on the Internet: Supercomputer Centers and Tasks: Proceedings of the International Supercomputer Conference (Novorossijsk, Russia, September, 20-25, 2010), pp. 590–595. Publishing of the Moscow State University, Moscow (2010)
3. Svirihin, D.I.: Definition of parallelism resource of algorithm and its effective use for of a finite number of processors. In: Scientific Service on the Internet: the Search for New Solutions: Proceedings of the International Supercomputer Conference (Novorossijsk, Russia, September, 17-22, 2012), pp. 257–260. Publishing of the Moscow State University, Moscow (2012)
4. Svirihin, D.I., Aleeva, V.N.: Definition the maximum effective realization of algorithm on the base of the conception of Q-determinant. In: Parallel Computational Technologies (PCT'2013): Proceedings of the International Scientific Conference (Chelyabinsk, Russia, April, 1-5, 2013), p. 617. Publishing of the South Ural State University, Chelyabinsk (2013)
5. Aleeva, V.N.: The Parallelization of Algorithms on The Base of The Conception of Q-determinant. In: Groups and Graphs, Algorithms and Automata, 2015: Abstracts of the International Conference and PhD Summer School of the 80th Birthday of Professor Vyacheslav A. Belonogov and of the 70th Birthday of Professor Vitaly A. Baransky (Yekaterinburg, Russia, August, 9-15, 2015), p. 33. UrFU Publishing house, Yekaterinburg (2015)
6. Aleeva V.N., Sharabura I.S., Suleymanov D.E. (2015) Software System for Maximal Parallelization of Algorithms on the Base of the Conception of Q-determinant. In: Malyshkin V. (eds) Parallel Computing Technologies. PaCT 2015. Lecture Notes in Computer Science, vol 9251. Springer, Cham https://doi.org/10.1007/978-3-319-21909-7_1
7. Voevodin, V.V., Voevodin, V.V.: The V-Ray technology of optimizing programs to parallel computers. In: Vulkov, L.G., Yalamov, P., Waśniewski, J. (eds.) WNAA 1996. LNCS, vol. 1196, pp. 546–556. Springer, Heidelberg (1997)
8. Voevodin, V.V., Voevodin, V.V.: Parallel computing. BHV-Petersburg, St.Petersburg (2002). (in Pussian)

9. Open Encyclopedia of Parallel Algorithmic Features. http://algowiki-project.org/en/Open_Encyclopedia_of_Parallel_Algorithmic_Features
10. Voevodin, V.V.: Parallel algorithms under the microscope. In: Parallel Computational Technologies (PCT'2016): Report on the International Scientific Conference (Arkhangelsk, Russia, March, 28 - April, 1, 2016). http://omega.sp.susu.ru/books/conference/PaVT2016/talks/Voevodin.pdf
11. Gurieva Y.L., Il'in V.P. (2015) On Parallel Computational Technologies of Augmented Domain Decomposition Methods. In: Malyshkin V. (eds) Parallel Computing Technologies. PaCT 2015. Lecture Notes in Computer Science, vol 9251. Springer, Cham https://doi.org/10.1007/978-3-319-21909-7_4
12. Suplatov, D.A., Voevodin, V.V., Svedas, V.K. Robust enzyme design: Bioinformatic tools for improved protein stability. In: Biotechnology journal, v. 10, № 3, pp. 344–355. Publishing Wiley - VCH Verlag GmbH & CO. KGaA (Germany) (2015) https://doi.org/10.1002/biot.201400150
13. Venkata, M.G., Shamis, P., Sampath, R., Graham, R.L., Ladd, J.S. Optimizing blocking and nonblocking reduction operations for multicore systems: hierarchical design and implementation. In: Proceedings of IEEE Cluster, pp. 1–8 (2013)
14. Malyshkin V.E., Perepelkin V.A., Schukin G.A. (2015) Distributed Algorithm of Data Allocation in the Fragmented Programming System LuNA. In: Malyshkin V. (eds) Parallel Computing Technologies. PaCT 2015. Lecture Notes in Computer Science, vol 9251. Springer, Cham https://doi.org/10.1007/978-3-319-21909-7_8
15. Malyshkin V.E., Perepelkin V.A., Tkacheva A.A. (2015) Control Flow Usage to Improve Performance of Fragmented Programs Execution. In: Malyshkin V. (eds) Parallel Computing Technologies. PaCT 2015. Lecture Notes in Computer Science, vol 9251. Springer, Cham https://doi.org/10.1007/978-3-319-21909-7_9
16. Legalov, A.I. Functional language for creating architecturally independent parallel programs. In: Computational technologies, № 1 (10), pp. 71–89 (2005) (in Pussian)
17. McColl, W.F.: General Purpose Parallel Computing. In: Lectures on Parallel Computation. In: Cambridge International Series on Parallel Computation, pp. 337–391. USA: Cambridge University Press (1993)
18. Valiant, L.G.: A bridging model for parallel computation. In: Communications of the ACM, vol. 33, № 8, pp. 103–111 (1990)
19. Val'kevich, N.V. Q-effective implementation of the algorithm for matrix multiplication on a supercomputer "Tornado SUSU": Graduate qualification work of bachelor in direction "Fundamental informatics and information technology": 02.03.02. 32 s. South Ural State University, Chelyabinsk (2017) http://omega.sp.susu.ru/publications/bachelorthesis/17-Valkevich.pdf
20. Tarasov, D.E. Q-effective co-design of realization of the Gauss-Jordan method on the supercomputer "Tornado SUSU": Graduate qualification work of master in direction "Fundamental informatics and information technology": 02.04.02. 41 s. South Ural State University, Chelyabinsk (2017) http://omega.sp.susu.ru/publications/masterthesis/17-Tarasov.pdf
21. Lapteva, Yu.S. Q-effective implementation of the Jacobi method for solving SLAE on the supercomputer "Tornado SUSU": Graduate qualification work of bachelor in direction "Fundamental informatics and information technology": 02.03.02. 30 s. South Ural State University, Chelyabinsk (2017) http://omega.sp.susu.ru/publications/bachelorthesis/17-Lapteva.pdf