The Parallelization of Algorithms on The Base of The Conception of $Q$-determinant

Valentina N. Aleeva

South Ural State University, Chelyabinsk, Russia

2015
Introduction

We describe the approach to parallelization algorithms based on their representation as $Q$-determinant. The proposed approach gives the possibility of the maximal parallelization of every algorithm if it enables the parallelization. The obtained results are oriented to ideal model of parallel computer system. However they can be a basis for automated execution of the most rapid algorithm implementations for real parallel computing systems.
Conception of $Q$-determinant

Let $\alpha$ be an algorithm for solving an algorithmic problem

$$\bar{y} = F(N, B)$$

where $N$ is a set of dimension parameters of the problem, $B$ is a set of input data, $\bar{y}$ is a set of output data. $N$ may be empty.

Let $Q$ be a set of operations those are used by algorithm $\alpha$.

One of the basic notions of $Q$-determinant conception is $Q$-term. Here I give a brief definition of $Q$-term.

If $N$ is empty then $Q$-term is a structured set of expressions over $B$ and $Q$ (terms of signature $Q$).

If $N$ is nonempty then $Q$-term is a map such that every set of parameters’ values of $N$ is corresponded to the structured set of expressions.

A $Q$-term can be unconditional, conditional or conditional infinite according to the structure of the expression set.
Conception of $Q$-determinant

$Q$-terms can be calculated. $Q$-determinant of the algorithm $\alpha$ is the set of $Q$-terms that we need to calculate each of the problem output data.

Let an algorithm $\alpha$ be in the form of

$$y_i = f_i(i = 1, \ldots, m)$$

where $f_i$ is $Q$-term to calculate $y_i$, $m$ is the number of output data.

Then we say that the algorithm $\alpha$ is represented in the form of $Q$-determinant.

If the algorithm has some representation as flowchart then it can be represented in the form of $Q$-determinant (S. Ignatyev).
Conception of $Q$-determinant

Realization of the algorithm $\alpha$ represented in the form of $Q$-determinant is called the process of calculating the $Q$-terms $f_i(i = 1, \ldots, m)$ that are included in the $Q$-determinant. If the calculation of all $Q$-terms $f_i(i = 1, \ldots, m)$ is produced at the same time and as rapid as possible, i.e. the operations from the set $Q$ are executed as soon as they are ready to perform, in this case we have the most rapid implementation of the algorithm. We can estimate difficulty of the most rapid implementation of the algorithm: the number of processors $P_\alpha$ and cycles of work $T_\alpha$ of the computer system.
Conception of $Q$-determinant

In examples 1–3 we consider representations of the algorithm in the form of $Q$-determinant.

Example 1. Algorithm of calculation of the scalar product of the vectors (algorithm Scalar)

$$\bar{a}^1 = (a^1_1, \ldots, a^1_n), \bar{a}^2 = (a^2_1, \ldots, a^2_n).$$

The $Q$-determinant consists of one unconditional $Q$-term and the algorithm representation in the form of $Q$-determinant has the shape

$$(\bar{a}^1, \bar{a}^2) = \sum_{i=1}^{n} a^1_i a^2_i.$$
Conception of $Q$-determinant


Let $A\bar{x} = \bar{b}$ be a system of linear equations, where $A = [a_{ij}]_{ij=1,...,n}$ is a $n \times n$ invertible matrix, $\bar{x} = (x_1, \ldots, x_n)^T$, $\bar{b} = (a_{1,n+1}, \ldots, a_{n,n+1})^T$.

At the first step we suppose that the leading element is the first nonzero element of the first row of the original matrix, and at $k$-th step ($2 \leq k \leq n$) we select the first nonzero element of the $k$-th row of the matrix $A^{j_1 \ldots j_{k-1}}$ that obtained at $(k-1)$-th step.

$Q$-determinant of Gauss–Jordan method consists of $n$ conditional $Q$-terms and the representation in the form of $Q$-determinant has the shape

$$x_j = \left\{ (u_1, w_1^j), \ldots, (u_n!, w_n^j) \right\} \ (j = 1, \ldots, n).$$
Conception of $Q$-determinant

Permutations of elements $(1, \ldots, n)$ may be numbered. Let $j_k$ is a column number of the leading element at $k$-th step, $i$ is a permutation number $(j_1, \ldots, j_n)$,

$$L_{j_1} = \bigwedge_{j=1}^{j_1-1} (a_{1j} = 0), \quad L_{j_l} = \bigwedge_{j=1}^{j_l-1} (a_{lj}^{j_1 \ldots j_{l-1}} = 0).$$

Then

$$u_i = L_{j_1} \land (a_{1j_1} \neq 0) \land \left( \bigwedge_{l=2}^{n} \left( L_{j_l} \land \left( a_{lj_l}^{j_1 \ldots j_{l-1}} \neq 0 \right) \right) \right), \quad w_{li}^{j_l} = a_{l,n+1}^{j_1 \ldots j_n} (l=1, \ldots, n).$$
Conception of $Q$-determinant

Example 3. Algorithm for calculating the sum $S$ of series

$$\sum_{m=1}^{\infty} (-1)^m \frac{1}{m}.$$ 

$Q$-determinant consists of one conditional infinite $Q$-term and the algorithm representation in the form of $Q$-determinant has the shape

$$S = \left\{ \left( \frac{1}{2} < \varepsilon, -1 \right), \left( \frac{1}{3} < \varepsilon, -1 + \frac{1}{2} \right), \ldots \right\}$$

$$\ldots, \left( \frac{1}{m} < \varepsilon, -1 + \frac{1}{2} - \cdots + (-1)^{m-1} \frac{1}{m-1} \right), \ldots \right\}.$$
Software System QStudio

The software system QStudio (D.E. Suleymanov, I.S. Sharabura) makes possible to calculate $Q$-determinant of any algorithm (if the algorithm has some representation as flowchart), to find the most rapid possible implementation and to build its execution plan for the parallel system.

The figure below displays the graph of the execution plan of the most rapid implementation of the algorithm Scalar. That algorithm calculates the scalar product of the vectors. In the given example the vector dimension is equal to 9. The figure is obtained with the help of the system Qstudio.
Figure: The execution plan of the most rapid implementation of the algorithm Scalar
Conclusion

The next step of the investigation is to generate executable code on the basis of the execution plan of the most rapid algorithm implementation plan. Also we develop the idea of optimal superposition of algorithm structure that described with the help $Q$-determinant on the architecture of concrete computer systems.