УДК 811.111

ABOUT CALCULATION OF TWO-DIMENSIONAL FAST FOURIER TRANSFORM

Тутатчиков В.С., Курбатова Е.А.
научный руководитель – д-р физ.-мат. наук Носков М.В, Курбатова Е.А.
Сибирский федеральный университет

Two-dimensional fast Fourier transform (2D FFT) is a task difficult enough for use of parallel calculation tools. As a rule, in the literature there is a standard method of 2D FFT calculation – so-called «rowwise, columnwise» method. Research in this direction is usually done in search of the algorithms relating to the architecture of a computer: a cluster, a video card, GPU. In the given article a possibility of paralleling of another FFT calculation algorithm, which is a two-dimensional similarity to Cooley-Tukey method, is investigated. Cooley-Tukey method research is catching due to the fact that a number of operations for 2D FFT calculation is much less than in a conventional algorithm: $2N^2 \log_2 N$ operations of additions and $\frac{3}{4} N^2 \log_2 N$ operations of multiplication against: $N^3 \log_2 N$ operations of additions and $\frac{1}{2} N^3 \log_2 N$ multiplication operations in «rowwise, columnwise» algorithm.

The recurrent scheme of two-dimensional similarity to Cooley-Tukey method can be developed as follows. At the fixed N we will name a two-dimensional periodic signal $x(j_1, j_2)$ periodic, with the period $N = 2^s$ with respect to each variable, complex-valued function of integer argument. With addition operations of two signals x_1, x_2:

$$y(j) = x_1(j) + x_2(j) \tag{1}$$

and signal x multiplication by complex number c:

$$y(j) = cx(j), \tag{2}$$

where $x(j)$ – counting of signal x in the point $j \in Z^2$, a set of signals C_N^2 becomes linear complex space. A zero element in C_N^2 is the signal O such that $O(j) = 0$ for all $j \in Z^2$. We will enter scalar product and norm in C_N^2:

$$\langle x, y \rangle = \sum_{j \in B_2(N)} x(j)\overline{y(j)}, \|x\| = \langle x, x \rangle^{1/2}, \tag{3}$$

where $B_2(N)$ - a set of integer vectors from $[0, N-1]$. Designate $w_N = e^{\frac{2\pi i}{N}}$. Mapping $F_N : C_N^2 \to C_N^2$ comparing a signal X with values

$$X(j) = \sum_{t \in B_2(N)} x(t) w_N^{-(j,t)}, \; j \in B_2(N) \tag{4}$$

with a signal x is called reiterated (two dimensional) discrete Fourier transform (DFT).

Let $N = 2^s, N_v = 2^{s-v}, \Delta_v = 2^{v-1}$. We construct a recurrence sequence of bases $f_0, f_1, ..., f_s$, where f_t - t-th basis consisting of N^2 bases $f_t(k), k \in B_2(N)$. The value of the signal $f_t(k)$ in counting $j = (j_1, j_2), j \in B_2(N)$ is denoted as $f_t(k, j)$. We denote $B_2^1(N)$ the set of integer vectors in $[0, N_v - 1]$, and a $B_2^2(N)$ the set of integer vectors in $[0, \Delta_v - 1]$. Let j - an integer from the set $J = \{0,1,...,2^v - 1\}$ can be represented in a binary system in the form $j_{v-1}2^{v-1} + ... + j_1 2 + j_0$, where $j_i = 0,1$ for all $i = 0,...,v-1$. We call the vector $(j_{v-1}, ..., j_1, j_0)$ a binary code of the number j. Compare a number $j_1 \in J$, which is

binary-encoded $(j_0, j_1, ..., j_{v-1})_2$, with the number j. Permutation $rev_v j = j_1$ of the set J is called reverse. Recurrence scheme for the calculation of the signal $x \in C_N^2$:

$$x_0(k) = x(rev_s k_1, rev_s k 2);$$

$$x_v(l_1 + \sigma_1\Delta_v + p_1\Delta_{v+1}, l_2 + \sigma_2\Delta_v + p_2\Delta_{v+1}) = \sum_{\tau_1}^{1}\sum_{\tau_2}^{1} w_{\Delta_{v+1}}^{\sum_{i=0}^{2}\tau_i(l_i+\sigma_i\Delta_v)} \times \quad (5)$$

$$\times x_{v-1}(l_1 + 2\Delta_v p_1 + \tau_1\Delta_v, l_2 + 2\Delta_v p_2 + \tau_2\Delta_v),$$

where $p = (p_1, p_2), p \in B_2^1(N)$, $l = (l_1, l_2), l \in B_2^2(N)$, $v = 1,..., s$ and σ_1, σ_2 equal 0 or 1.
Parallel algorithm for 2D FFT similar to Cooley-Tukey method.

1) in each of the s $(s = \log_2 N)$ recursions the original signal is represented as: $x = x_{11} + x_{12} + x_{21} + x_{22}$, where subsignal x_{ij} contains components of x, the parity of the coordinates coincides with i and j, respectively;
2) in each step of the recursion $v = 0,1,..., s-1$ choose all the minors of the second order

$$\begin{pmatrix} a_{i,j} & a_{i,j+2^v} \\ a_{i+2^v,j} & a_{i+2^v,j+2^v} \end{pmatrix}, \quad (6)$$

where $a_{i,j} \in x_{11}$, $a_{i,j+2^v} \in x_{12}$, $a_{i+2^v,j} \in x_{21}$, $a_{i+2^v,j+2^v} \in x_{22}$, moreover, the elements $a_{i,j}$, $a_{i,j+2^v}$, $a_{i+2^v,j}$, $a_{i+2^v,j+2^v}$ are not repeated for different minors;
3) in each recursion step the main process sends each computational process different row pairs i and $i+2^v$, containing minors (6);
4) in each recursion step computational processes calculate DFT over minors (6) contained in the derived rows, write the result in the rows at the place of the calculated elements and send the calculated result to the main process;
5) in each step of the recursion the main process takes the rows and writes them in the initial positions of the matrix;
6) after computation the matrix is normalized by the main process.

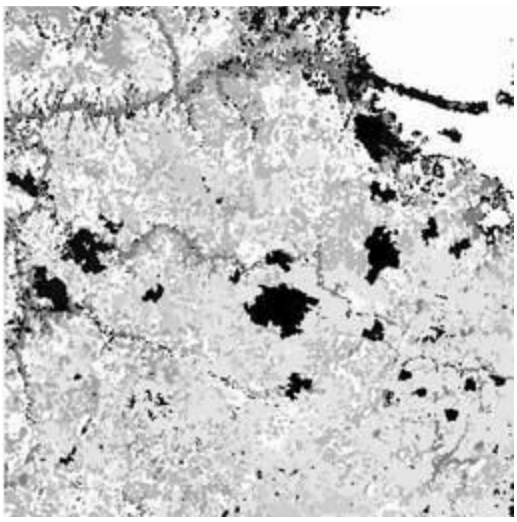

Figure 1. An example of the original signal

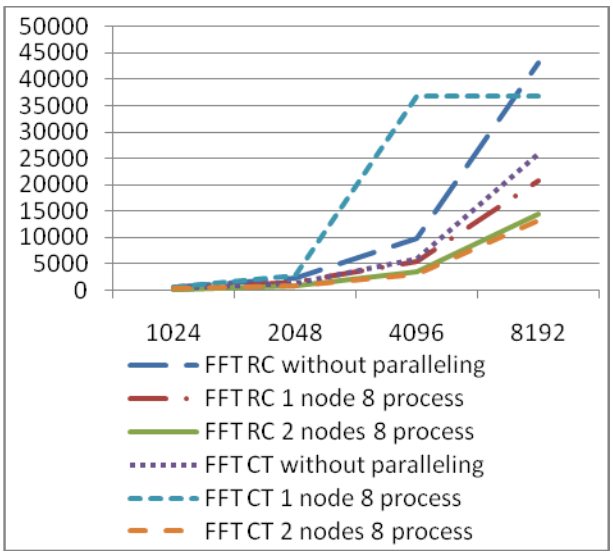

Figure 2. Comparison of various algorithms run time

To test the algorithm the program in language C ++ with the use of MPI library was written. The program implements two algorithms: 2D FFT «rowwise, columnwise» (2) and

2D FFT similar to Cooley-Tukey method (7). The two-dimensional FFT «rowwise, columnwise» is used in such mathematical packages as MathLab, MatCad, fftw open library. Only «rowwise, columnwise» (fftw) algorithm is performed as an algorithm for parallel FFT as it is the easiest to implement.

Testing was performed using the supercomputer of the Institute of Space and Information Technology at Siberian Federal University. The supercomputer consists of 224 IBM Blade HS21 compute nodes. Each node includes a 16 Gb of common RAM and two Xeon quad core E5345@2.33 GHz quad-processors. While testing, the program run time under different conditions was measured: on a single node without paralleling, on a single node with paralleling for 8 processes, on two nodes with paralleling for 8 processes. The table 1 shows the results of measuring the average time in milliseconds when run on a single node with paralleling for 8 processes. Satellite images were used as an original signal, an example is shown in Figure 1. Figure 2 shows comparison of various algorithms run time when run on one and two nodes.

Table 1

The test result on a single node

N	P	FFT RC	FFT CT
1024	1	490	310
	2	310	310
	4	220	270
	8	180	270
	16	180	350
2048	1	2300	1330
	2	1550	1260
	4	1060	950
	8	840	850
	16	850	1000
4096	1	9880	5850
	2	6240	4590
	4	4370	3590
	8	3430	3060
	16	3740	3390
8192	1	43210	25190
	2	26990	19160
	4	18550	14870
	8	14520	13130
	16	14090	11700

Explanation:

$N = 2^s$, $N \times N$ – number of counting of a signal $x(j_1, j_2)$;

P – number of started MPI-processes;

FFT RC – 2D FFT «rowwise, columnwise» method;

FFT CT – 2D FFT similarity to Cooley-Tukey method.

The result of the calculation is given in milliseconds.

The research results in the implementation of 2D FFT algorithm similar to Cooley-Tukey; it works faster than 2D FFT «rowwise, columnwise» when run on a single node. However, when run on multiple nodes at the same time, it works slower because of a large number of data exchanges between processes.