Simulation of parallelization options for the computational image formation algorithm in the synthetic aperture radar for oil spill detection

V A Shpenst
Saint-Petersburg Mining University, Saint-Petersburg, 21, line VO, 2, 199106, Russia
E-mail: shpenst@spmi.ru

Abstract. The article discusses the methods of parallelization of the computational image formation algorithm in radar stations designed to detect oil spills. The methods are based on parallel calculations. The methods for modeling the image formation process, the method for extracting information dependencies and the method for scaling and distributing subtasks were developed. The technique for analyzing the efficiency of the parallelization process and evaluating the algorithm stability was suggested, and the computational experiment was conducted. Its results were evaluated.

1. Introduction
Today, the environmental problem is one of the most urgent for the world community. Environmental pollution is an important problem, since anthropogenic activities affect all terrestrial spheres: atmosphere, hydrosphere and lithosphere. The studies on ocean pollution [10-11] identified that the total amount of oil entering the waters is 2-8 million tons per year. The largest number of spills occurs during loading and unloading operations in ports (about 90%). The amount of oil entering the waters as a result of spills does not exceed 7 tons [11]. Oil enters the waters during well drilling (up to 100 thousand tons per year). When it enters seas, it forms “oil spills” in the upper layer of the water mass [11]. A large number of living organisms live in the oil spills. In addition to the environmental damage, oil spills can cause enormous economic damage. These problems could have been avoided if oil spills were prevented. In addition, the volume of oil transportation is increasing annually. For example, the volumes of sea transportation through the Barents Sea along the coast of Norway have doubled since 2005. According to the reports [10], about 35% of oil spills are due to the production and transportation.

Emergency oil spills should be detected as soon as possible. Rapid detection of oil spots is one of the main requirements for monitoring anthropogenic oil pollution of the sea surface. To fulfill this requirement, the most promising method is the use of radar tools [12]. In recent years, much attention has been paid to the development of methods for detecting oil spots on the water surface using synthetic aperture radars installed on spacecraft [10, 11, 12, 14, 15]. Space tools make it possible to obtain information 1-2 times a day, which does not provide the necessary speed of oil spill detection. One of the promising methods for obtaining information on oil spills is the use of radars installed on unmanned aerial vehicles or offshore oil platforms and transport vessels [6, 10].
2. Information processing methods
Modern radar systems installed on airplanes and spacecraft are one of the most rapidly developing segments of electronic equipment. The main differences between space and aviation radar stations (RLS) are the principles of processing radar signals associated with different aperture sizes, characteristics of the propagation of radar signals in different layers of the atmosphere, the need to take into account the curvature of the earth’s surface, etc. [8].

At present, airborne RLS with synthetic aperture radars (SAR) can solve problems of detecting pollution, selecting mobile and stationary objects, analyzing changes in the surface situation, and detecting small-sized marine objects. Due to the artificial increase in the aperture of the onboard antenna, the main principle of which is the coherent accumulation of reflected radar signals in the synthesis interval, it is possible to obtain a high resolution. In modern systems, the resolution can reach tens of centimeters when working in the centimeter wavelength range. Processing the trajectory signal to obtain ultra-high resolution in range and azimuth in real time requires very high speed ($\approx 10^{12}$-$10^{15}$ op/s) and a large amount of processor memory ($\approx 10^{10}$-$10^{12}$ bits), which is an extremely difficult task for onboard computers. In terrestrial conditions, this task is successfully solved by an optical processor, which records the trajectory signal onto the photographic film and the analog signal is processed using a coherent optical system [1].

Synthesis of the aperture is time-consuming, which leads to a delay in information in the SAR. The minimum delay of information is determined by the synthesis time, i.e. time of SA formation. The maximum delay is determined taking into account the execution time of the synthesis algorithm for processing path signals. Terrestrial optical processors have the greatest delay periods. It consists of the flight time of the aircraft in the operating area of the SAR, the time of return to the base, the time of delivery of the film with the recorded trajectory signals to the laboratory, the time of photochemical film processing, optical image processing and recording and photochemical processing of the secondary films. This time can be several hours. The above considerations and areas of application of SARs determine the high relevance of research and development of methods to solve this problem. Separate publications and technical solutions allow us to conclude about the technical feasibility of the solutions. The main purpose of the article is to assess possibilities of a new direction in information processing when forming images in SAR, which involves the development of methods and algorithms for distributing calculations performed during their operation.

3. Distributed Computing Modeling
The sequence of steps for determining effective ways for parallel computing can be as follows:
- analysis of existing computational schemes and their decomposition into parts (subtasks), which can be implemented independently of each other;
- allocation of subtasks of information interactions that should be carried out in solving the original task;
- determination of the computing power required to solve the problem and distribution of subtasks.

It is clear that the amount of computation for each processor should be the same - this will ensure uniform computational loading (balancing) of the processors. In addition, it is also clear that the distribution of subtasks should be performed in such a way that the presence of communication interactions is minimal [3]. After completing all the design stages, it is possible to evaluate the effectiveness of the developed parallel methods - for this, quality indicators of the generated parallel computing (acceleration, efficiency, scalability) are determined. Based on the results, it may be necessary to repeat the individual development stages - a return to the previous steps can occur at any stage. In this regard, a frequently performed additional action is to adjust the composition of the generated set of tasks after determining the available number of processors - sub-tasks can be enlarged (aggregated) in the presence of a small number of processors. In general, these actions can be defined as scaling the algorithm being developed.

To apply the parallel method, it is necessary to develop programs for solving the subtasks and launch the developed programs on the processors in accordance with the selected distribution scheme.
of subtasks. To carry out calculations, programs are launched for execution (programs at the stage of execution are usually referred to as processes), implementation of information interactions; programs should have means of data exchange (message channels). It should be noted that each processor usually solves one single subtask; however, if there are a large number of subtasks or a limited number of processors used. This rule may not be observed and, as a result, several programs (processes) can be executed on processors simultaneously. In particular, during the development and initial verification of a parallel program, a single processor can be used to execute all the processes (when located on the same processor, the processes are executed in the time distribution mode) [4].

This approach is focused on computing systems with distributed memory, when the information interactions are realized by transmitting messages over communication channels. Nevertheless, this scheme can be used without losing any efficiency of parallel computing and for developing parallel methods for systems with shared memory - in this case, message transfer mechanisms to ensure information interactions should be replaced by operations to access the shared variable [5]. The scheme of designing and implementing the parallel computing provides a way to understand parallel algorithms and programs. At the design stage, the parallel method can be represented in the form of a “message subtask” column, which is nothing more than an enlarged (aggregated) representation of the information dependency graph. Similarly, at the execution stage, to describe the parallel program, a model can be used in the form of a “processes channel” graph, in which processes are used instead of subtasks, and information dependencies are replaced by message transfer channels. On this model, the distribution of processes among processors of a computing system can be shown if the number of subtasks exceeds the number of processors. The use of two parallel computing models makes it possible to better separate the problems that appear when developing parallel methods. The first model, the “subtasks-messages” graph, allows you to focus on the sub-tasks of the same computational complexity, while ensuring a low level of information dependence between the subtasks. The second model, the “processes - channels” graph, focuses on the distribution of subtasks, providing another opportunity to reduce the complexity of information interactions by locating intensively interacting processes on the same processors. In addition, this model allows you to better analyze the effectiveness of the developed parallel method and provides the possibility of a more adequate description of parallel calculations. The process is a program running on the processor that uses part of the local memory of the processor and contains a number of data reception / transmission operations for organizing informational interactions between the processes of the parallel program. Logically, a data transmission channel can be considered as a message queue to which one or several processes can send forwarded data and from which the destination process can retrieve messages sent by other processes [5].

An important issue in the allocation of subtasks is the choice of the desired level of decomposition of the calculations. The formation of the maximum possible number of subtasks ensures the use of the maximum achievable level of parallelism of the problem being solved; however, it complicates an analysis of parallel calculations. The use of “large” subtasks when decomposing computations creates a clear parallel computation scheme; however, it may complicate the efficient use of a large number of processors. A possible reasonable combination of these two approaches may involve the use of those subtasks for which decomposition methods are known as structural elements of decomposition. For example, when analyzing the matrix multiplication problem, the methods of a scalar product of vectors or algorithms of matrix-vector product can be used as subtasks. This intermediate method of decomposing computations can simplify the representation of computational schemes and ensure the efficiency of parallel calculations.
For the problem under consideration, a sufficient level of decomposition can involve the division of the matrix into many separate rows and determination of maximum values in separate rows; the structure of information links corresponds to a linear graph. To assess these provisions, we consider a typical parallel algorithm for processing information in PCA, which is based on a tape data separation scheme. Multiplication of matrix $A$ of size $m \times n$ and matrix $B$ of size $n \times l$ creates matrix $C$ of size $m \times l$, whose each element is determined in accordance with formula:

$$c_{i,j} = \sum_{k=0}^{n-1} a_{i,k} \cdot b_{k,j}, 0 \leq i < m, 0 \leq j < l. \quad (1)$$

As follows from (1), each element of matrix $C$ is a scalar product of the corresponding rows of matrix $A$ and the column of matrix $B$:

$$c_{i,j} = (a_i, b^T_j); a_i = (a_{i,0}, a_{i,1}, \ldots, a_{i,n-1}); b^T_j = (b_{0,j}, b_{1,j}, \ldots, b_{n-1,j}). \quad (2)$$

This algorithm involves performing $m \cdot n \cdot l$ multiplication operations and the same number of addition operations. When multiplying square matrices of size $n \times n$, the number of operations performed is $O(n^3)$. Sequential matrix multiplication algorithms are less complex (e.g., the Strassen’s algorithm), but these algorithms require some efforts to learn them. When developing parallel methods, a sequential algorithm was used. This algorithm is iterative and focuses on the sequential calculation of rows of matrix $C$. Indeed, when performing one iteration of the external loop (the cycle by variable $i$), one row of the resulting matrix is calculated [6]. Since each element of the resulting matrix is a scalar product of the row and the column of the source matrices, to calculate all the elements of matrix $C$ of size $n$, it is necessary to perform $n^2 \times (2n -1)$ scalar operations. It requires time [6]:

$$T_i = n^2 \cdot (2n -1) \cdot \tau, \quad (3)$$

where $\tau$ is the execution time of one elementary scalar operation.

Then, we studied the applicability of two parallel matrix multiplication algorithms in which matrices $A$ and $B$ were divided into continuous sequences of rows or columns.

**Identification of subtasks.** All elements of matrix $C$ can be calculated independently of each other. As a result, a possible approach for organizing parallel computations involves the use of the procedure for determining one element of matrix $C$. Each subtask must contain one row of matrix $A$ and one column of matrix $B$. The total number of the subtasks is $n^2$ (by the number of elements of matrix $C$). The achieved level of parallelism is redundant. During practical calculations, the number of generated subtasks exceeds the number of available processors and, as a result, the stage of enlargement of basic tasks is inevitable. In this regard, it may be useful to aggregate computations at the step of identifying basic subtasks. A possible solution is to combine all the calculations associated with several elements of matrix $C$. It is necessary to identify the basic problem as a procedure for calculating all the elements of one of the rows of matrix $C$. This approach decreases the total number of subtasks to $n$. To perform all the calculations for the base subtask, one of the rows of matrix $A$ and columns of matrix $B$ must be available. A simple solution to this problem is duplication of matrix $B$ in all subtasks. It is usually unacceptable due to the large memory costs for storing data. As a result, organization of the calculations should be structured so that at each current point, the subtasks contain only part of the data required for the calculations. Access to other data should be ensured through the transmission of messages. Then, we studied the applicability of radar data synthesis in SAR, two possible ways to perform parallel calculations of this type.

**Identification of information dependencies.** To calculate one row of matrix $C$, each subtask should contain a row of matrix $A$ and access to all columns of matrix $B$. Possible ways to organize parallel calculations are as follows. The algorithm is an iterative procedure whose number of iterations coincides with the number of subtasks. At each iteration of the algorithm, each subtask contains one row of matrix $A$ and one column of matrix $B$. When performing the iteration, scalar multiplication of the rows
and columns is performed, which allows the identification of elements of matrix C. Upon completion of the calculations, the columns of matrix B should be transferred between the subtasks so that in each subtask there are new columns of matrix B and new elements of matrix C. Moreover, this transfer should be organized so that after the iteration of the algorithm, all the columns of matrix B are in each subtask. A possible simple scheme for organizing the sequence of transfers of the columns of matrix B between the subtasks involves the representation of the topology of the information links of the subtasks in the form of a ring structure. In this case, at each iteration, subtask $i$, $0 \leq i < n$ will transmit its column of matrix B to the subtask $i + 1$ (in accordance with the ring structure, subtask $n-1$ transfers its data to the subtask with number 0). After all iterations of the algorithm have been completed, the necessary condition will be provided - in each subtask, all columns of matrix B will appear.

Scaling and distribution of subtasks across processors. Dedicated base subtasks are characterized by the same computational complexity and equal amount of transmitted data. In the case when the size of the matrices $n$ is larger than the number of processors $p$, the basic subproblems can be enlarged by combining several adjacent rows and columns of the multiplied matrices within one subproblem. In this case, the original matrix A is divided into a series of horizontal bands, and the matrix B is represented as a set of vertical (for the first algorithm) or horizontal (for the second algorithm) bands. The size of the bands in this case should be chosen equal to $k = n / p$ (under the assumption that $n$ is a multiple of $p$), which will still ensure uniform distribution of the computational load across the processors that make up the multiprocessor computing system. To distribute subtasks, any method can be used that provides an effective representation of the ring structure of information interaction of subtasks. For this, it is enough, for example, that the subtasks adjacent to each other in the ring topology are located on processors between which there are direct data lines.

Scaling and distribution of subtasks to the processors. The basic subtasks are characterized by the same computational complexity and equal amount of transmitted data. When the size of the matrices $n$ is larger than the number of processors $p$, the basic subtasks can be enlarged by combining several adjacent rows and columns of the multiplied matrices within one subtask. Matrix A is divided into a series of horizontal bands, and matrix B is a set of vertical (for the first algorithm) or horizontal (for the second algorithm) bands. The size of the bands should be equal to $k=n/p$ ($n$ is a multiple of $p$), which will ensure uniform distribution of the computational load across the processors that make up the multiprocessor computing system. To distribute subtasks, any method that provides an effective representation of the ring structure of information interaction of subtasks can be used. It is enough that the subtasks adjacent to each other in the ring topology are located on the processors between which there are direct data lines.

4. Algorithm Performance Analysis.
The complexity of the sequential algorithm is proportional to $n^3$. For a parallel algorithm, each processor multiplies the bands of matrix A and matrix B (the size of the bands is $n/p$ and the total number of operations performed during this multiplication is $n^3/p^2$). Since the number of iterations of the algorithm coincides with the number of processors, the complexity of the parallel algorithm can be determined by formula:

$$T_p = (n^3 / p^2) \times p = n^3 / p.$$  \hfill (4)

With this assessment, the acceleration and efficiency indicators of this parallel matrix multiplication algorithm are as follows:

$$S_p = \frac{n^3}{(n^3 / p)} = p; \quad E_p = \frac{n^3}{(n^3 / p)} / p = 1.$$  \hfill (5)

Thus, a general complexity analysis provides ideal indicators of the effectiveness of parallel computing. To specify the obtained relations, the number of computational operations of the algorithm should be calculated, and costs of performing data transfer operations should be taken into account.
With the number and duration of the operations performed, the computational time of the parallel algorithm can be estimated as follows:

\[ T_p(\text{calc}) = \left( \frac{n^2}{p} \right) \times (2n-1) \times \tau. \]  

(6)

To assess the communication complexity of parallel computing, we will assume that all data transfer operations can be performed in parallel. The amount of data transferred is determined by the size of the bands. It is \( \frac{n}{p} \) rows or columns of length \( n \). The total number of parallel message transfer operations is less than the number of iterations of the algorithm (data transfer is optional at the last iteration). Thus, assessment of the complexity of the performed data transfer operations can be determined as [4]:

\[ T_p(\text{comm}) = (p-1) \times (\alpha + w \cdot n \cdot (n/p)/\beta), \]  

(7)

where \( \alpha \) is latency, \( \beta \) is bandwidth of the data network, and \( w \) is the size of the matrix element in bytes. Taking into account the obtained relations, the total execution time is determined by the following expression:

\[ T_p = \left( \frac{n^2}{p} \right) \times (2n-1) \times \tau + (p-1) \times (\alpha + w \cdot n \cdot (n/p)/\beta). \]  

(8)

5. Results of computational experiments.

The experiments were conducted on a computing cluster based on Intel i7 - 3100 MHz processors and a Gigabit Ethernet network running the Microsoft Windows Server operating system. The results of the computational experiments are shown in Table 1. The experiments were performed using two, four, and eight processors. A comparison of experimental time \( T_p^* \) and theoretical time \( T_p \) from formula (8) is presented in Table 1 and in Figures 1 and 2.

| Size of matrices | Sequential algorithm | 4 processors | Acceleration | 8 processors | Acceleration | 16 processors | Acceleration |
|------------------|----------------------|--------------|--------------|--------------|--------------|--------------|--------------|
| 1000             | 0,8752               | 0,1535       | 5,6982       | 0,0968       | 9,0371       | 0,04668      | 18,0234      |
| 2000             | 12,8787              | 2,2628       | 5,6912       | 0,6998       | 18,4014      | 0,6998       | 34,0564      |
| 3000             | 43,4731              | 11,0804      | 3,9234       | 5,1766       | 8,3978       | 2,6578       | 17,5671      |
| 4000             | 103,0561             | 21,6001      | 4,7710       | 9,4127       | 10,9485      | 4,7435       | 19,3965      |
| 5000             | 201,2915             | 56,9203      | 3,5363       | 18,3303      | 10,9813      | 9,07564      | 20,8713      |
| 6000             | 347,8434             | 111,9642     | 3,1067       | 45,5482      | 7,6368       | 22,1254      | 13,5621      |

Figure 1. The dependence of acceleration on the number of processors when implementing the first parallel matrix multiplication algorithm with a tape data distribution scheme
Table 2. Comparison of experimental and theoretical runtimes of the first parallel matrix multiplication algorithm with a tape data distribution scheme

| Differences in matrices | 4 processors $T_p$ | 4 processors $T_p^*$ | 8 processors $T_p$ | 8 processors $T_p^*$ | 16 processors $T_p$ | 16 processors $T_p^*$ |
|-------------------------|-------------------|---------------------|-------------------|---------------------|-------------------|---------------------|
| 1000                    | 0.8243            | 0.3758              | 0.4313            | 0.1535              | 0.2353            | 0.0968              |
| 2000                    | 6.51822           | 5.4427              | 3.3349            | 2.2628              | 1.7436            | 0.6998              |
| 3000                    | 21.9137           | 20.9503             | 11.1270           | 11.0804             | 5.7340            | 5.1766              |
| 4000                    | 51.8429           | 45.7436             | 26.2236           | 21.6001             | 13.4144           | 9.4127              |
| 5000                    | 101.1377          | 99.5097             | 51.0408           | 56.9203             | 25.9928           | 18.3303             |
| 6000                    | 174.6301          | 171.9232            | 87.9946           | 111.9642            | 44.6772           | 45.5482             |

**Figure 2.** The dependence on the volume of initial data of theoretical and experimental execution time of the parallel algorithm for four processors

6. Conclusion

To estimate the duration $\tau$ of the basic scalar operation, the matrix multiplication problem was solved using a sequential algorithm and the calculation time was divided by the total number of operations performed. As a result of such experiments, it was determined that $\tau$ was 6.4 nsec. The experiments conducted to determine the parameters of the data transmission network showed latency $\beta$ and throughput capacity $\beta$ which were equal to 130 $\mu$s and 53.29 MB/s, respectively. All calculations were performed on numerical values of the double type, i.e., the value of $w$ was 8 bytes.

References

[1] Kondratenkov G S et al 2005 *Radiovision* (M.: Radio and communications)
[2] Sablin V N 2002 *Reconnaissance-strike systems and radar systems for observing the earth's surface* (M.: Radio engineering)
[3] Gergel V P and Strongin R G 2001 *Basics of parallel computing for multiprocessor computing systems* (N. Novgorod: UNN)
[4] Bogachev K Yu 2003 *Basics of parallel programming* (M.: BINOM. Laboratory of Knowledge)
[5] Voevodin V V and Voevodin VI V 2002 *Parallel computing* (SPb: BHV-Petersburg)
[6] Andrews G R 2000 *Foundations of Multithreaded, Parallel, and Distributed Programming* (Reading MA: Addison-Wesley)
[7] Roosta S H 2000 *Parallel Processing and Parallel Algorithms: Theory and Computation* (NY: Springer-Verlag)
[8] Ushakov I E 2016 *Radar monitoring of sea surface pollution by oil products from drilling plat-
forms and transport vessels Notes of the Mining Institute 219 421-427 DOI 10.18454
[9] Ed Bondura V G 2012 Aerospace monitoring of oil and gas facilities (M.: Scientific World)
[10] Bass F G and Fuks I M 1972 Wave scattering on a statistically uneven surface (M.: Nauka)
[11] Sources of pollution of the hydrosphere by oil, available at: http://biofile.ru/geo/23616.html
[12] Nichiporenko N T, Marenich I E, Petrov A V, Misyuchenko I, Trofimov B S, Ushakov I E 2010 Detection of oil spills using a navigation radar Su-completion 2 39-41
[13] Satellite radar monitoring of the seas, available at: http: // live map.rf / ru / environment / monitoring / radar /
[14] The technology of multi-level environmental monitoring in order to information security of off-shore oil and gas production, available at: http://neftegas.ru/scince/view/181
[15] Ushakov I E and Shishkin I F 1997 Radar sounding of the sea surface (M.: RIC "Tatyana's Day")
[16] Salberg A-B, Rudjord O and Solberg A H S 2014 Oil spill detection in hybrid-polarimetric SAR images IEEE Transactions on Geoscience and Remote Sensing 52 (10) 6521-6533
[17] Ed Halpern D 2000 Satellites oceanography and society (Elsvier)
[18] Ushakov I E, Vinogradova A A 2019 Non-contact methods and means of measuring the oil film thickness on the water surface Journal of Physics: Conference Series 1384(1) DOI:10.1088/1742-6596/1384/1/012066
[19] Iakovlev P V 2019 Technogenic risks of the construction and operation of oil and gas complexes in the Caspian Sea Bulletin of the Astrakhan State Technical University. Series: Marine Technology 4 pp. 46-59 DOI:10.24143/2073-1574-2019-4-46-59
[20] Pashkevich M A, Petrova T A 2019 Development of an operational environmental monitoring system for hazardous industrial facilities of Gazprom Dobycha Urengoy Journal of Physics: Conference Series 1384(1) DOI:10.1088/1742-6596/1384/1/012040
[21] Vyboldin Y K, Borisov S V 2019 Noise immunity of signal demodulation procedure in binary coded communication channels with a frequency-position multiplexing Journal of Physics: Conference Series 1384(1) DOI: 10.1088/1742-6596/1384/1/012069