Cloud Computing for Nanophotonic Simulations

Nikolay L. Kazanskiy[1] and Pavel G. Serafimovich[2]

[1] Image Processing Systems Institute
of the Russian Academy of Sciences
Samara, Russia
[2] S.P. Korolyov Samara State Aerospace University
Samara, Russia
paulserch@gmail.com

Abstract. Design and analysis of complex nanophotonic and nanoelectronic structures require significant computing resources. Cloud computing infrastructure allows distributed parallel applications to achieve greater scalability and fault tolerance. The problems of effective use of high-performance computing systems for modeling and simulation of subwavelength diffraction gratings are considered. Rigorous Coupled-Wave Analysis (RCWA) is adapted to cloud computing environment. In order to accomplish this, data flow of the RCWA is analyzed and CPU-intensive operations are converted to data-intensive operations. The generated data sets are structured in accordance with the requirements of MapReduce technology.

Keywords: cloud computing, subwavelength diffraction grating, optimization, scatterometry, MapReduce.

1 Introduction

In recent years there has been a steady increase in the research and development activity in the nanoscale fabrication area. Optical methods are widely used to characterize nanoelectronic and optimize nanophotonic structures [1-4]. When the characteristic size of the nanostructure is comparable to the wavelength of the used light, the conventional scalar diffraction methods become inadequate. Therefore, rigorous Maxwell's electromagnetic theory should be adopted to analyze such structures. With the advances of computer technology, many numerical techniques have been developed to rigorously solve the diffraction problem.

Rigorous Coupled Wave Analysis (RCWA) [5-10] has been frequently used to solve the problem of electromagnetic wave diffraction by periodic structures. Solving optimization and scatterometry problems may require millions of diffraction pattern evaluations for each possible combination of geometric structure and incident light parameters. Computational cost estimate shows that these problems would clearly call for supercomputing capacity.

Thus, current computational tasks of nanostructure analysis generate ever-higher demands on the methods used for parallel computing and data storage [11, 12]. The

S. Dolev and M. Oltean (Eds.): OSC 2012, LNCS 7715, pp. 54–67, 2013.

developed applications should work efficiently on multicore and multiprocessor systems. An implementation of MPI (Message Passing Interface) is the traditional means of creating such applications [13]. MPI provides a highly flexible ability to develop applications that address specific requirements of the algorithm structure and the used computing infrastructure. However, this ability makes it necessary for developers to implement some of the low-level services.

Compared with MPI, MapReduce programming paradigm for cloud computing [14] provides higher level of abstraction for the developer of parallel applications. On the one hand, MapReduce imposes certain restrictions on data format and the flexibility of the algorithm used. On the other hand, MapReduce offers a simple programming model, automatic parallelization and distribution, fault-tolerance, I/O scheduling, monitoring tools.

In the following we consider how RCWA can be implemented to utilize MapReduce technology. The RCWA relies heavily on the computation of eigenvalues of an intermediate matrix and the solution of a corresponding linear system. To reduce the compute time and enhance fault tolerance, we build a distributed, dynamically growing look-up table containing the precomputed eigenvalues and eigenvectors for the set of lamellar layers that approximate the analyzed nanostructure. The look-up table resides in a specialized distributed file system.

The paper is organized as follows. Section II describes the main features of MapReduce framework implementation for cloud computing. Section III outlines the algorithm and data flow of RCWA. Section IV reports the adaptation of RCWA for MapReduce technology. Section V describes some performance tests for the suggested algorithm on the Hadoop-cluster [14, 16]. Conclusions are given in the final section.

2 Main Features of MapReduce Technology

MapReduce is based on the old approach of splitting a task into subtasks, executing them in parallel and merging the intermediate results to obtain the final one [17-19]. However, the MapReduce implementations of this scheme provide effective means for controlling resource utilization in large scale distributed systems.

The processing of large data sets (more than hundreds of gigabytes) is a challenge for the traditional High Performance Computing (HPC) attitude. In comparison to CPU-intensive tasks, the data-intensive tasks require demanding rate of access to the used storage system. The bottleneck of HPC for these problems is the shared files system network bandwidth. Hence, a large number of computing nodes would remain idle and wait for the data. Because of this, the load balancing and data locality become important to efficiently handle.

MapReduce framework provides load balancing of computing nodes by organizing the “master-workers” architecture. The master node is responsible for the assignment of computing tasks to worker nodes every time a worker node becomes idle.

For a large data set it is crucial to ensure the closeness of data and computation to reduce data transfer between nodes. Thus, the input data should already be stored locally on the corresponding computing nodes. MapReduce achieves this by using a

specialized distributed file system. This file system divides each file into blocks with size of about several tenths of megabytes and stores several copies of each block on different nodes. This data replication in the most cases allows the master node to assign computing tasks to a worker that stores corresponding input data.

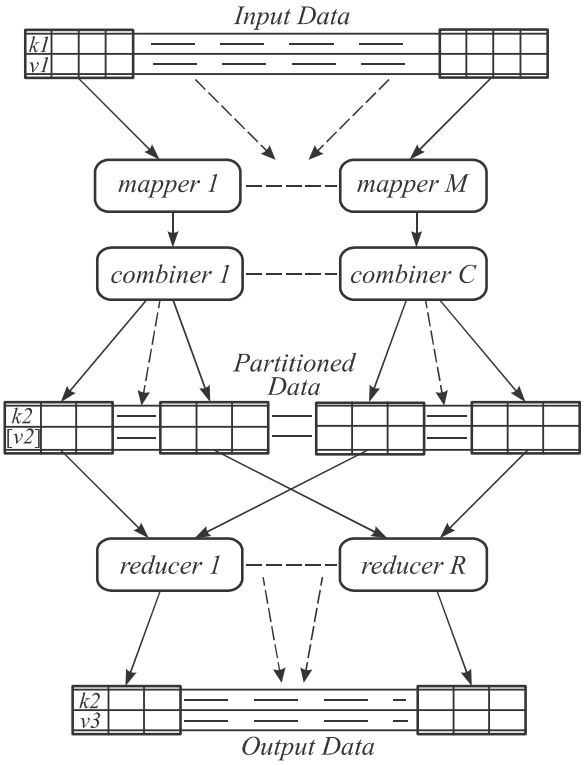

Fig. 1. MapReduce scheme

The next problem of traditional HPC is the difficulty in handling partial failure of computing nodes. In large clusters of thousands of computing nodes it is necessary to permanently deal with crashes. In MapReduce, the master node pings every worker periodically to recognize possible failures of individual nodes. Thus, master detects a failed computing task and reschedules it to another worker which owns a replica of the input data block.

The two-stage processing scheme of MapReduce is depicted in Fig. 1. The first and second stages correspond, respectively, to Map() and Reduce() functions. Both functions need to be implemented by the developer of distributed application. Key/value pairs form the required processing data structure. Every input key/value pair is processed by the Map() to produce a set of intermediate key/value pairs. The Reduce() takes all values associated with the same intermediate key as input and generates a set of final key/value pairs. Thus, it is sufficient to implement the two functions to obtain a usable distributed application.

To reduce the network traffic between nodes MapReduce supports the use of auxiliary Combine() function which is similar to Reduce(). Combine() processes the output of the Map() on each node of the cluster separately to prepare the intermediate results to subsequent shuffling over a network to the Reduce() nodes.

The shuffling of the intermediate results is based on the dividing up the set of intermediate keys and assigning intermediate key/value pairs to Reduce() nodes. The goal is to assign approximately the same number of keys to each Reduce() node.

3 Overview of RCWA

The general grating diffraction problem is illustrated in Fig. 2. It is convenient to separate the space into three regions: two homogeneous semi-infinite regions above and below the grating, and an inhomogeneous region which includes the relative permittivity periodic modulation of the analyzed nanostructure. A linearly polarized plane electromagnetic wave of wavelength λ is obliquely incident at an arbitrary polar angle θ and at an azimuthal angle ϕ upon a two-dimensional, possibly multi-level or surface-relief, dielectric or metal grating. The normalized electric-field vector that corresponds to this plane wave is a solution of Maxwell's equations in an infinite homogeneous region [20]. The variable ψ is the angle between the polarization vector $\boldsymbol{E}$ and the plane of incidence.

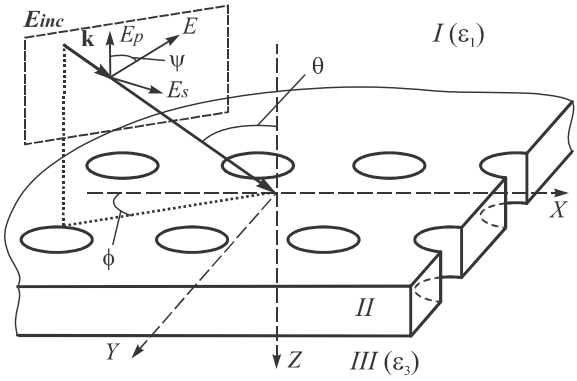

Fig. 2. Analyzed nanostructure

RCWA was formulated for both principal plane and conical incidence [6]. The problem of the slow convergence of the solution when a conducting grating is illuminated with a transverse magnetic (TM) polarized wave has been subsequently resolved [7].

RCWA involves several steps. As shown in Figure 3(a), the grating is sliced into a set of two-dimensional layers and each layer is approximated by a rectangular slab. The periodic permittivity distribution in each slab of the grating region is expanded into the Fourier series.

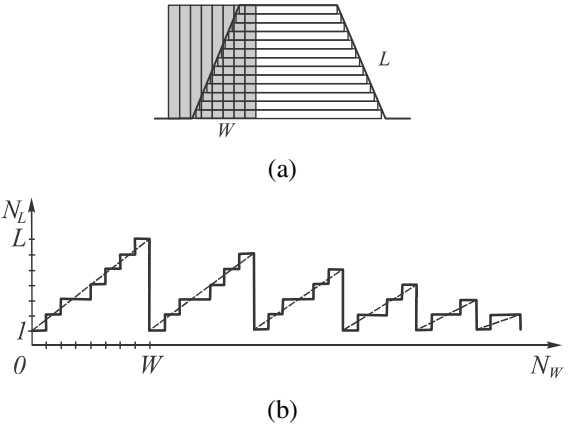

Fig. 3. Example of two-dimensional diffraction grating analysis; (a) the grating period with the admissible value ranges of the upper and lower radius of the truncated cone; (b) the number of eigenvalues and eigenvectors calculations for each set of the truncated cone upper and lower radius without storing the intermediate information

The field in the grating region can be also expressed as Fourier expansion through application of the Floquet condition. Substituting these expansions into the Maxwell equations, we can derive a set of first-order coupled wave equations. Its homogeneous solution can be obtained by solving for the eigenvalue and eigenvector of the matrix that corresponds to this set of wave equations. The size of the matrix is equal to $(2\boldsymbol{N}+1)^4$ for conical incidence, where $\boldsymbol{N}$ is the number of positive observed diffraction orders for each of the coordinates $\boldsymbol{X}$ and $\boldsymbol{Y}$. The electromagnetic field above and below the grating region can be expressed using linear superposition of plane waves in the direction of various orders of grating diffraction. In the final step, this representation is used as the boundary conditions to determine the specific solution of the diffraction efficiency.

Accuracy of RCWA is related to the number of diffraction orders that are included in the superposition. In general, the greater number of observed diffraction orders the higher the accuracy. However, it is necessary to control the systematic computational error when a large number of observed diffraction orders is selected.

4 Mapping RCWA to the MapReduce Scheme

We consider the technique of using the RCWA for optical simulation or optimization of the nanostructures. As an example of such structure, we take a two-dimensional diffraction grating. The grating period consists of a truncated cone, and its central cross section is shown in Fig. 3 (a). The gray rectangle in Fig.3 (a) shows the set of the possible (evaluated) geometric parameters. The slope of the truncated cone lies inside this rectangle. The parameter $\boldsymbol{W}$ determines the number of vertically homogenous

layers that approximate the slope in RCWA. We have the number of these layers is fixed and equal to $\boldsymbol{L}$.

As mentioned in the preceding section, eigenvalues and eigenvectors of the corresponding matrix are calculated for each vertically homogeneous layer. The calculated eigenvalues and eigenvectors do not depend on the thickness of a homogeneous layer. Without saving the intermediate results, this time-consuming task would be repeated many times. The sawtooth step function in Fig. 3 (b) shows the dependence of number $\boldsymbol{N_L}$ of eigenvalues and eigenvectors calculations without storing the intermediate information on $\boldsymbol{N_W}$ sets of radius variation. The parameter $\boldsymbol{L}$ in the Fig. 3 (b) is assumed to be equal to 7 and parameter $\boldsymbol{W}$ is equal to 8. Note that the number of eigenvalues and eigenvectors calculations is equal to W when the intermediate information is stored.

Such storage of the intermediate information converts the CPU-intensive problem into a data-intensive one. We estimate the amount of memory needed to store the calculated eigenproblem data.

The structure and number of input parameters for the task of optical modeling of nanostructures depends in particular on the simulation purposes and the geometry of the structure. Thus, we can distinguish two main groups of input parameters. The first group determines the illumination conditions of the simulation. The second group consists of the geometric parameters of the nanostructure and its variations.

We estimate the possible number of input parameters for the task of modeling the nanostructure as shown in Figure 3 (a). We start with the group of geometrical parameters. Let the number of possible values of the truncated cone radius be 100 ($\boldsymbol{W}$ = 100). Suppose that a variation of the structure period is also allowed. The number of possible values of the structure period is equal to 10. Let us now consider the group of the illumination conditions parameters. Suppose that in our case this group contains two parameters. First, it is the meridional angle of incidence. Second, it is the azimuth angle of incidence. For each of these two parameters, we set the number of possible variations to 30. Thus, the number of eigenproblems to solve is equal to the product of the number of variations of the aforesaid four parameters, i.e. ~10^6.

Having computed the parameters number, we now evaluate the size of the stored intermediate data. This value depends on the number of observed diffraction orders. Next we assume that the calculations are done with long double precision (16 bytes).

Fig. 4 (a) shows the dependence of the size of the eigenvectors matrix on the number of observed diffraction orders. Fig. 4 (b) shows the total amount of the stored intermediate data for the different number of the observed diffraction orders and the parameters numbers. The figure shows that the amount of data may exceed 1 TB. Thus, our task can be classified as a data-intensive problem. To reduce the size of the stored intermediate data some compression procedure can be used. However, this in turn implies there is a corresponding computational overhead.

We will now consider how RCWA can be mapped into MapReduce scheme. The structure of stored data for MapReduce technology is defined by a pair of “key /

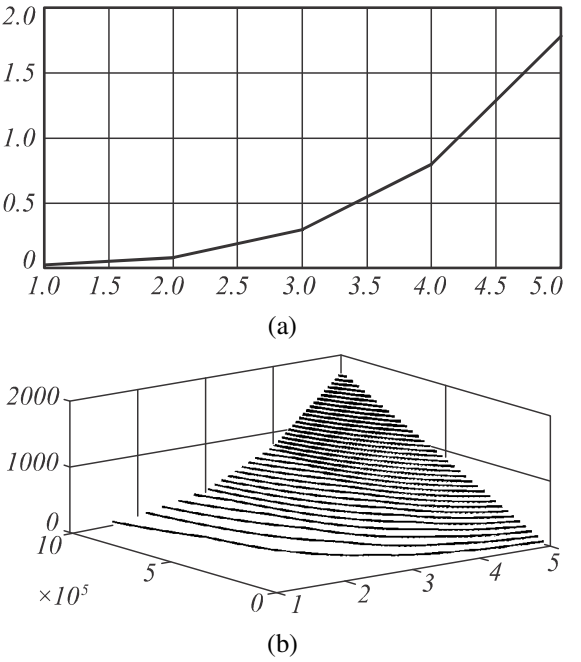

Fig. 4. The size of the stored intermediate data; (a) the dependence of the size of the eigenvectors matrix on the number of observed diffraction orders; (b) the total amount of the stored intermediate data for the different number of the observed diffraction orders and the parameters numbers.

value". Here the "key" $\mathbf{K}_1$ consists of two sets. First, it is the set of geometric parameters that define one of the layers which is homogeneous in the vertical direction. Second, it is the set of parameters that define the illumination conditions of the simulation. The following relation is an example of such "key".

$$\mathbf{K}_1 = \left\{ d_x, m_x, \mathbf{b_x}, \mathbf{n_x}, d_y, m_y, \mathbf{b_y}, \mathbf{n_y}, \lambda, \theta, \phi \right\}$$

where d_x, d_y are the periods of two-dimensional structure, m_x, m_y are the number of material sections on the period, $\mathbf{b_x}, \mathbf{b_y}$ are the size of material sections on the period, $\mathbf{n_x}, \mathbf{n_y}$ are the index of refraction of these material sections. The last three parameters in the "key" - λ, θ, ϕ - describe the lighting conditions. Notice that the thickness of the layer is absent in the above list.

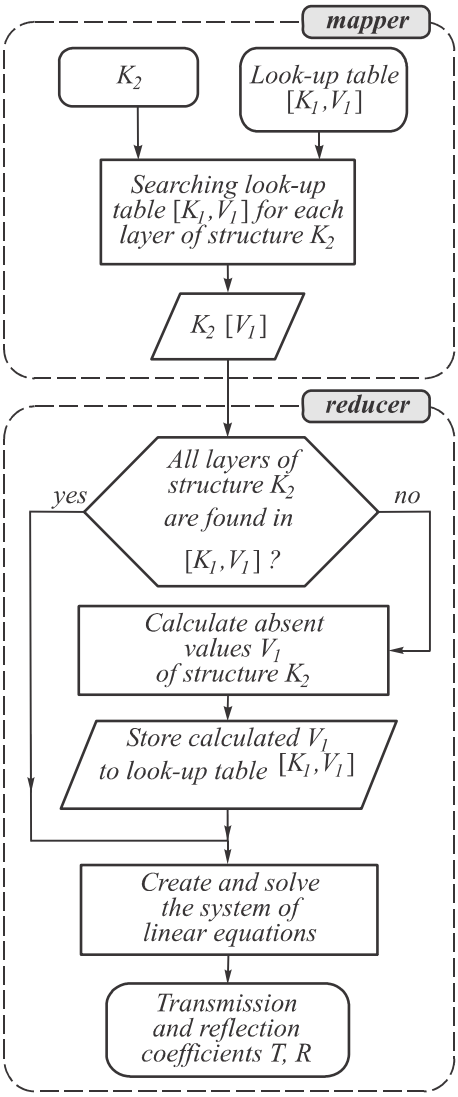

Fig. 5. Flowchart of RCWA implemented with MapReduce technology

The "value" $\mathbf{V}_1$ contains the calculated eigenvalues and eigenvectors for one layer:

$$\mathbf{V}_1 = \mathbf{V}_2 = \{\mathbf{Eig}\}$$

The look-up table $\mathbf{K}_1$ / [$\mathbf{V}_1$] resides in the distributed file system mentioned in Section II. By combining with MapReduce, the distributed file system allows providing efficient load balancing, data locality and faulting tolerance to the whole computing system.

Fig. 5 shows the flowchart of RCWA method in terms of MapReduce technology. The Map() function receives as input the key $\mathbf{K}_2$ that describes the analyzed nanostructure. This key consists of a set of keys $\mathbf{K}_1$ that define the layers of the structure, as well as the array $\mathbf{t}$ that describes the thicknesses of these layers.

$$\mathbf{K}_2 = \mathbf{K}_3 = \left\{ \vec{\mathbf{K}}_1, \mathbf{t} \right\}$$

The Map() looks for the intermediate data in the look-up table with the given key $\mathbf{K}_2$. Thus, the output of the Map() function are pairs of $\mathbf{K}_2$ / [$\mathbf{V}_2$]. Each structure is associated with the list of the eigenproblem solutions [$\mathbf{V}_2$] for the array of $\vec{\mathbf{K}}_1$ layers. This list of the eigenproblem solutions may contain null values, in the case where the necessary information is not yet in the look-up table.

The pairs of $\mathbf{K}_2$ / [$\mathbf{V}_2$] are the input parameters of the Reduce(). This function calculates the missing intermediate data, stores them in the look-up table, then generates the system of linear equations and solves it.

In this paper, we assume that the output parameters of the Reduce() are the diffraction efficiency of observed diffraction orders. However, the same approach is valid for other output parameters, for example, electromagnetic field distribution in the so-called near-field [10]. Thus, in our case

$$\mathbf{V}_3 = \left\{ \mathbf{R}, \mathbf{T} \right\}.$$

where $\mathbf{R}, \mathbf{T}$ are the diffraction efficiency of observed diffraction orders respectively for reflection and transmission.

5 The Results of Computational Experiments on Hadoop-Cluster

Computational experiments are performed on a relatively small Hadoop-cluster consisting of four nodes. Each node in the cluster contains a CPU Intel Xeon 2.13 GHz and 4 GB RAM.

Namenode-server is run on one node, and jobtracker-server is run on another node. The remaining two cluster nodes are used as working nodes. Two mappers and two reducers can be run simultaneously on each of the working nodes, which correspond to the parameters that are set by default. Other parameters of Hadoop-cluster are also set by default.

The experiment measures the parallel read/write average throughput for the HDFS file system. The parallel read average throughput is equal to r=460 MB/s and the parallel write average throughput is equal to w=60 MB/s. It is not our goal in this work

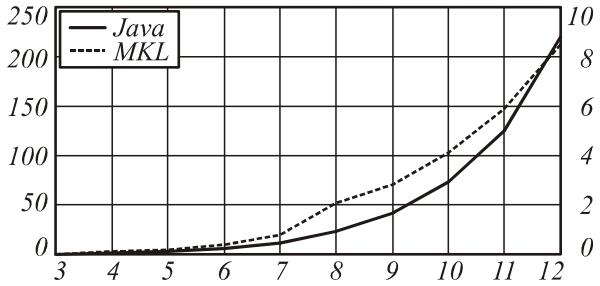

Fig. 6. The dependence of the time (in seconds) of eigenvectors calculation on N value for Java-implementation (the left Y-axis) and Intel MKL implementation (the right Y-axis)

to enhance *r* and *w* values by optimizing the default parameters of Hadoop-cluster. Therefore, the above values *r* and *w* are assumed to be fixed.

We will now assess the feasibility of the proposed scheme of RCWA implementation. The standard and the proposed algorithms are different only at the eigenvectors calculation step. We compare the time of eigenvectors calculation for the standard RCWA algorithm (without saving the intermediate results) with the time of eigenvectors calculation for the proposed algorithm (with saving the intermediate results). The main share of the RCWA computational time is contributed by the eigenvectors calculation. The eigenvectors are calculated for each layer of the modeled nanostructure.

Java programming language is the most appropriate for the implementation of a flexible application within the Hadoop package. Therefore, the algorithm for finding the eigenvectors of a complex general matrix was implemented with Java.

Figure 6 shows the dependence of the time (in seconds) of eigenvectors calculation on ***N*** value for Java implementation and Intel MKL implementation. ***N*** value denotes the number of the positive diffraction orders for one coordinate. Here we assume that ***N*** value is the same for each coordinate.

The left ***Y***-axis on the graph shown in the Figure 6, corresponds to the results of Java-application. The right ***Y***-axis of this graph shows the same results for Intel MKL implementation.

We define the following parameters:

s - Upload file size (MB), which is processed by one mapper;

f - The "useful" share of the uploaded file, which is used by mapper to find the eigenvectors (normalized value);

m - The average number of uses of one matrix from the uploaded file;

a - The size of the matrix (MB), which contains the computed eigenvectors for one layer of the modeled nanostructure;

e - The time for computing the eigenvectors of the matrix for one layer of the structure (s);

L - The total number of layers in structures that are processed by one mapper.

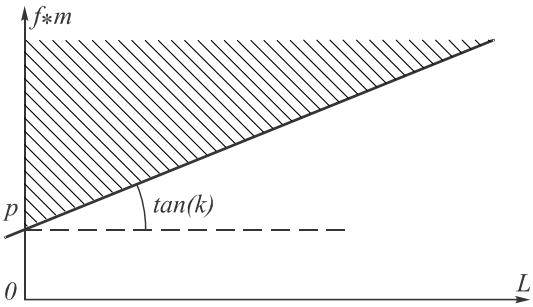

Fig. 7. The dependence of L on the overall usage ratio of the uploaded file; the cross hatched area shows where the relation (1) is satisfied

Then the condition for the feasibility of the proposed RCWA algorithm can be written as follows: the total computational time that one mapper spends *(L e)* should be greater than the time spent on the following steps: (1) the reading from the HDFS file with size of *s*, (2) the calculation of the unresolved eigenvectors, (3) the writing of the calculated eigenvectors in HDFS.

The above described condition can be represented as a linear dependence of the *L* value on the overall usage ratio of the uploaded file *f m*:

$$(f \cdot m) > k \cdot L + p \tag{1}$$

where

$$k(N) = \frac{a^2(N)}{s(e(N) \cdot w - a(N))}, \quad p(N) = \frac{a(N) \cdot w}{r(e(N) \cdot w - a(N))}.$$

In Fig. 7 the cross hatched area shows where the relation (1) is satisfied. The slope of the curve in the Figure 7 (*k* value) defines the rate at which the overall usage ratio *f m* should increase when *L* value increased.

Fig. 8 shows the dependence of the curve slope value *k* on ***N*** value for (a) Java-implementation and (b) Intel MKL implementation.

Note that the curve slope value *k* is very small on the Figure 8(a). Thus, to stay within the feasibility requirements, there is no much importance struggling for the increasing the overall usage ratio *f m* when using the relatively slow Java-implementation. Such need appears for the more effective Intel MKL implementation (Fig. 8(b)). In this case, the different approaches of meta-data usage can be considered.

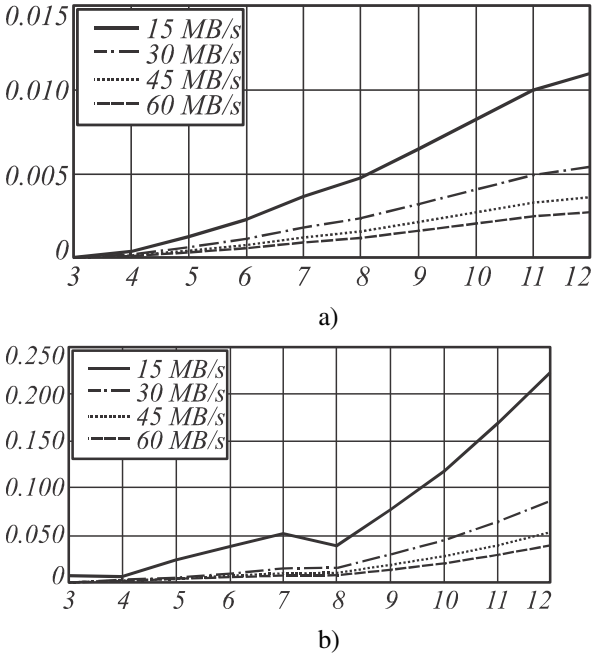

Fig. 8. The dependence of k value on the number of diffraction orders value N for (a) Java-implementation and (b) Intel MKL implementation

6 Conclusions

The paper discusses the computational problems arising in modeling and optimization of complex nanophotonic structures by Fourier modes method (RCWA). The cloud computing infrastructure usage is suggested to solve these problems. This approach allows to effectively exploit the potential of modern computational tools by improving the scalability of the computational problem and by enhancing the fault tolerance of the computer system. This opens up new possibilities in solving problems of diffraction nanophotonic [21-23], magneto-optics [24,25] and plasmonics [26,27].

Acknowledgments. This work is supported by the program of the RAS Presidium "Problems of a national environment for distribution of scientific information and computing based on the development of GRID technology and modern telecommunications networks", grant of Russian Federation President for Support of Leading Scientific Schools NSh-4128.2012.9 and RFBR grants No 11-07-00153, 13-07-97002, 13-07-97004, program No 5 of basic research for Nanotechnology and Information Technology Department of RAS “Basic problems of physics and technology of epitaxial nanostructures and related devices”.

References

1. Golub, M.A., Kazanskii, N.L., Sisakyan, I.N., Soifer, V.A.: Computational experiment with plane optical elements. Optoelectronics, Instrumentation and Data Processing (1), 78–89 (1988) (in Russian)
2. Kazanskiy, N.L.: Mathematical simulation of optical systems. SSAU, Samara (2005) (in Russian)
3. Kazanskiy, N.L., Serafimovich, P.G., Khonina, S.N.: Harnessing the guided-mode resonance to design nanooptical transmission spectral filters. Optical Memory & Neural Networks (Information Optics) 19(4), 318–324 (2010)
4. Golovashkin, D.L., Kazanskiy, N.L.: Solving Diffractive Optics Problem using Graphics Processing Units. Optical Memory and Neural Networks (Information Optics) 20(2), 85–89 (2011)
5. Moharam, M.G., Pommet, D.A., Grann, E.B.: Stable implementation of the rigorous coupled-wave analysis for surface-relief gratings: Enhanced transmittance matrix approach. J. Opt. Soc. Am. A 12(5), 1077–1086 (1995)
6. Gystis, E., Gaylord, T.: Three-dimensional (vector) rigorous coupled wave analysis of anisotropic grating diffraction. J. Opt. Soc. Am. A 7, 1399–1419 (1990)
7. Lalanne, P., Morris, G.M.: Highly improved convergence of the coupled-wave method for TM polarization. J. Opt. Soc. Am. A 13(4), 779–784 (1996)
8. Li, L.: Use of Fourier series in the analysis of discontinuous periodic structures. J. Opt. Soc. Am. A 13(9), 1870–1876 (1996)
9. Bezus, E.A., Doskolovich, L.L., Kazanskiy, N.L.: Evanescent-wave interferometric nanoscale photolithography using guided-mode resonant gratings. Microelectronic Engineering 88(2), 170–174 (2011)
10. Bezus, E.A., Doskolovich, L.L., Kazanskiy, N.L.: Scattering suppression in plasmonic optics using a simple two-layer dielectric structure. Applied Physics Letters 98(22), 221108 (3 p.) (2011)
11. Armbrust, M., et al.: A view of cloud computing. Communications of the ACM 53(4), 50–58 (2010)
12. Volotovskiy, S.G., Kazanskiy, N.L., Seraphimovich, P.G., Kharitonov, S.I.: Distributed software for parallel calculation of diffractive optical elements on web-server and cluster. In: Proc. IASTED, pp. 69–73. ACTA Press (2002)
13. Snir, M., Otto, S., Huss-Lederman, S., Walker, D., Dongarra, J.: MPI-The Complete Reference. The MPI Core, vol. 1. MIT Press, Cambridge (1998)
14. Dean, J., Ghemawat, S.: MapReduce: a flexible data processing tool. Communications of the ACM 53(1), 72–77 (2010)
15. `http://Hadoop.apache.org/` (Tested June 15, 2011)
16. Venner, J.: Pro Hadoop. Springer (2009)
17. Voevodin, V.V.: Mapping computational problems in computer architecture. Computational Methods and Programming: New Information Technologies 1(2), 37–44 (2000) (in Russian)
18. Popov, S.B.: Modeling the task information structure in parallel image processing. Computer Optics 34(2), 231–242 (2010) (in Russian)
19. Soifer, V.A. (ed.): Computer Image Processing, Part I: Basic concepts and theory, 283 p. VDM Verlag Dr. Muller e.K. (2009)
20. Born, M., Wolf, E.: Principles of Optics. Pergamon, Oxford (1980)
21. Soifer, V.A.: Nanophotonics and diffractive optics. Computer Optics 32(2), 110–118 (2008) (in Russian)

22. Soifer, V.A., Kotlyar, V.V., Doskolovich, L.L.: Diffractive optical elements in nanophotonic devices. Computer Optics 33(4), 352–368 (2009) (in Russian)
23. Kazanskiy, N.L., Serafimovich, P.G., Popov, S.B., Khonina, S.N.: Using guided-mode resonance to design nano-optical spectral transmission filters. Computer Optics 34(2), 162–168 (2010) (in Russian)
24. Belotelov, V.I., Doskolovich, L.L., Zvezdin, A.K.: Extraordinary magneto-optical effects and transmission through metal-dielectric plasmonic systems. Physical Review Letters 98(7), 5 p. (2007)
25. Bykov, D.A., Doskolovich, L.L., Soifer, V.A., Kazanskiy, N.L.: Extraordinary Magneto-Optical Effect of a Change in the Phase of Diffraction Orders in Dielectric Diffraction Gratings. Journal of Experimental and Theoretical Physics 111(6), 967–974 (2010) (in Russian)
26. Bezus, E.A., Doskolovich, L.L., Kazanskiy, N.L., Soifer, V.A., Kharitonov, S.I., Pizzi, M., Perlo, P.: The design of the diffractive optical elements to focus surface plasmons. Computer Optics 33(2), 185–192 (2009) (in Russian)
27. Bezus, E.A., Doskolovich, L.L., Kazanskiy, N.L., Soifer, V.A., Kharitonov, S.I.: Design of diffractive lenses for focusing surface plasmons. Journal of Optics 12(1), 015001 (7 p.) (2010)