Characteristics of parallel execution of multiple date algorithms.

DUSMATOVA M.A.

Samarkand Institute of Economics and Service.

Abstract - *This article focuses on issues related to the increase in efficiency of computing technologies over the last few years, both the development of multi-core processors and the wide spread of cluster systems, including "cloud" systems. Parallelization of individual analysis algorithms and their further optimization are extremely important for today's scientists. and suggestions and recommendations for the development of general approaches to the parallelization of algorithms are given.*

Key words - *Databases, Statistics, Artificial Intelligence, Data Volumes, Block, Cycle, Vector, Communication, Structure, Algorithm, Cycle By Vectors, Information Exchange.*

I. INTRODUCTION

Data mining (DM) is the process of discovering previously unknown non-trivial, practically useful and accessible knowledge in “raw” data, which is necessary for decision-making in various areas of human activity [1]. DM lies at the intersection of several sciences, the main of which are database systems, statistics and artificial intelligence.

Mining algorithms extract knowledge from large datasets. At the same time, the greatest value and non-triviality of the knowledge gained is possible when analyzing significant amounts of data. The following main analysis problems arise here :

- performance - analysis of large volumes (measured in terabytes) requires large computing resources and can be performed in an unacceptable time for an analyst;

• distribution - due to the large amount of data, information can be stored in a distributed storage, in addition, due to the nature of the data, they can be stored in different sources.

Both problems can be solved by parallel and/or distributed execution of data mining (DM).

II. LITERATURE REVIEW

The following scholars have considered the peculiarities of parallel execution of data mining algorithms in their research: Barseghyan A.A. [1], Kupriyanov M.S. [2], Amol G., Prabhanjan K., Edwin P., Ramakrishnan K. [3], Talia D. [4], Stolyarov L. N., AbramovB. M. [5], Karpov V. E. [6], Kholod I.I. [8], [10], [11], Kupriyanov M.S. [9], Noughton P., Shildt G. [13].

III. RESEARCH METHODOLOGY

The methodological basis of the research was formed as a result of the study of theoretical and practical information, legislation and other legal documents, literary sources and publications. The research is based on the connections between theory and practice, but also made extensive use of methods such as analysis, comparison.

IV. ANALYSIS AND RESULTS

The increase in the performance of computing technology is associated both with the development of multi-core processors and with the increasing spread of cluster systems, including "cloud" systems in the last few years. However, modern software lags far behind the hardware and often uses the available computing resources inefficiently. This problem is primarily related to the complexity of the task of parallelizing computational algorithms.

Unfortunately, IAD algorithms are no exception. A large number of studies are currently being carried out in this area. Separate areas in the field of IAD are identified (in foreign literature, this area is called Data Mining): parallel DM (Parallel Data Mining) and distributed DM (Distributed Datamining). Most of the efforts of researchers in the field of parallel DM algorithms are spent on parallelizing individual analysis algorithms and their further optimization. The situation is aggravated by the fact that these efforts are applied on the basis of a certain computing environment, and therefore, when transferring such a solution to other conditions, it becomes ineffective. In this regard, research in the field of general approaches to parallelization of existing mining algorithms is a rather urgent task.

To develop a general approach to the construction of parallel DM algorithms, we consider their features, the general concept of execution and the presence of common typical blocks.

The analysis of DM algorithms made it possible to identify a number of features of such algorithms:

1) iterative processing of large amounts of data;

2) the presence in the algorithms of typical blocks (cycles on vectors and attributes, etc.);

3) formation on the basis of the input data of a clear result in the form of a knowledge model;

The input to the DM algorithm is a data set. As a rule, such data is represented by a set of separate vectors of the same type, characterized by a set of attributes. The same type of vectors means that they all have the same set of attributes. The DM algorithm, in the course of its work, processes each vector (possibly several times). The processing process, the attributes to be parsed, the methods applied to the data, and other parameters that affect the operation of the algorithm are specified as algorithm settings. Depending on the settings in the process of processing the input data, a knowledge model is built. Such a knowledge model reflects the patterns extracted from the data set.

Thus, the execution of the DM algorithm can be represented, as it is shown in the figureone.

It should be taken into account that different DM algorithms can build knowledge models of the same structure. For example, various algorithms for building decision trees build a knowledge model in the form of a decision tree. Therefore, the creation of an empty knowledge model of a certain structure can be taken out of the scope of the algorithm and it can be assumed that, in addition to data and settings, an empty structure of the knowledge model is transmitted to the input of the algorithm.

DM algorithms, like any algorithms, include blocks such as actions, decisions, and cycles. However, they also have other common blocks specific only to DM algorithms. Consider the following well-known algorithms that solve various functions of the DM (the flowchart of these algorithms is given in Appendix1):

- Naive algorithmBayes ;
- algorithm for constructing ID3 decision trees;
- clustering algorithms K - Means and EM-algorithm;
- association rule search algorithms Apriori and D HP .

Analyzing the presented algorithms, it can be noted that all of them have a common block - a cycle by vectors (“Cycle by vectors”).

Considering that all data analysis algorithms consistently work with vectors, it can be argued that this block is typical not only for the considered algorithms, but also for all DM algorithms. Within the loop over vectors, one or more operations can be performed on individual vectors.

Another characteristic block in the analysis algorithms is the loop over attributes. Within this loop, a single attribute value is parsed. In this case, as a rule, this cycle is performed inside the cycle over vectors, which provides an iterative analysis of each attribute of each vector. A similar combination of blocks can be shown using the Naive algorithm as an example.Bayes and the ID 3 algorithm . In them, the Attribute Loop is executed inside the Vector Loop.

In addition to cycles over vectors and attributes, which are inherent in almost all DM algorithms, it is possible to single out blocks inherent in algorithms of one type of DM functions. So the algorithms for solving the classification function are inherent in cycles over classes (values of the dependent attribute). This cycle is also present in the Naive algorithmBayes - cycle "Cycle by classes".

Clustering algorithms are characterized by cycles over clusters, as well as a cycle over vectors of an individual cluster. Such cycles are present, for example, in the K - Means algorithm and the EM -algorithm - the cycles "Cluster Cycle" and "Cluster Elements Cycle".

In addition to cyclic blocks in the same type of algorithms, there are identical blocks. For example, for algorithms for searching for association rules, the block "Formation of output rules" is common. In addition, often the algorithm produces a different result when replacing one block with another. For example, in clustering algorithms, such a block is a distance calculation block. In algorithms for constructing decision trees, such a block can be a function for calculating the information content of an attribute.

From the analysis carried out, it can be concluded that the DM algorithms contain the same blocks. The decomposition of algorithms into separate blocks and the selection of identical ones among them will reduce the effort for the development and debugging of DM algorithms. In addition, such block division allows the algorithm to be reformulated for its parallel execution.

The work of the mining algorithm when parallelized by data and by tasks

In any parallel algorithm, parallel executing branches must be explicitly distinguished. Before executing such branches, preparations must be made to allow multiple branches to run in parallel:

- create branches;
- initialize them with data for processing, algorithm execution parameters and the current state of the model;
- distribute the branches of the algorithm according to the means of their implementation (threads, actors , agents, etc.).

In this regard, an operation is added to the structure of the algorithm that provides such parallelization - split . After the execution of parallel branches, an operation must follow that ensures a single result.

- synchronization of executed branches of the algorithm;
- combining results.

join operation is added to the algorithm structure .

There are two types of parallelization of algorithms [Ошибка! Источник ссылки не найден.]: by data and by tasks. In data parallelization , each branch includes the same sequence of steps that is applied to different data. When parallelizing by tasks, the branches will contain different sequences of steps.

FIGURE 2. PIPELINE MODEL OF THE ANALYSIS ALGORITHM

For DM algorithms, the second method of parallelization is the most appropriate, given that the same operations are applied to different data. However, parallelization by tasks cannot be ruled out. It is possible to use the pipeline model of parallelism, when each stage of the algorithm (including data preparation) is executed on a separate processor (computer) (Figure 2). In this case, data is extracted from the source in portions, and the result of their processing is transferred to the next processor.

In both cases, the split and join operations must be added to the structure of the algorithm (Figure 3.). When parallelizing the DM algorithm on data, the split operation will perform the following actions:

- creation of identical branches of the algorithm (cloning of the main branch);
- distribution of the data set, execution settings and the knowledge model under construction for each branch;
- assignment of a branch to a specific execution handler;
- launching parallel branches for execution.

FIGURE 3. SPLIT AND JOIN OPERATIONS IN THE STRUCTURE OF ALGORITHMS (A - WITH DATA PARALLELIZATION

, b - with task parallelization)

The join operation during data parallelization must synchronize the threads and combine the knowledge models built by them.

When parallelizing the DM algorithm across tasks, the split operation must:

- create parallel branches;
- determine the order of their execution (indicate links to each subsequent branch);

• pass to the first branch a link to the data set, execution settings and the knowledge model being built;

• run the algorithm.

The join operation must get all the results from the last branch in the chain and merge them (essentially, performing the final operation in the pipeline).

Parallel operation of the mining algorithm with and without a dispatcher

Parallel algorithms can also be classified depending on the subordination of parallel branches. We can distinguish parallelization with independent threads and parallelization with a dispatcher thread.

When parallelizing with independent threads , all handlers of parallel execution of algorithm branches (threads) are equal and independent. For data and task parallelism, the structure of such algorithms is shown in Figure 4. Each independent branch of the parallel DM algorithm must receive everything necessary before execution : a data set, execution settings, and a knowledge model. Next, they start and wait for the completion of work. This is done with the split operation as described above.

Once started, each thread runs independently. After the work of all threads is completed, the results will be combined with the join block .

The distinguishing feature of dispatcher thread parallelization is that a single master thread is allocated to manage the other threads. Control may consist, for example, in the distribution of data between dependent threads.

The structure of such an algorithm is shown in Figure 4. Between the split and join operations , the main branch is allocated - the dispatcher thread.

Split operation may have some differences, depending on what actions the dispatcher thread takes on. For example, it will not perform the distribution of data, execution settings, and the knowledge model being built for each branch if this function was taken over by the dispatcher thread. In either case, the split operation initializes and starts the dispatcher thread.

a) b)

FIGURE 4. STRUCTURE OF A PARALLEL ALGORITHM WITH A DISPATCHER THREAD (A - WITH DATA PARALLELIZATION , B - WITH TASK PARALLELIZATION)

The join operation in such an algorithm structure, depending on the functions of the dispatcher thread, can also change the composition of the actions performed. So, if the dispatcher thread performs the union of intermediate and final results, then the join operation performs only the synchronization of branches and the launch of the next block.

Another characteristic that distinguishes parallel algorithms from each other is the presence or absence of information exchange between parallel branches. Any exchange is divided into two phases:

 preparing and sending a message;

 receiving and processing messages.

Thus, two types of operations responsible for message forwarding must be performed in a parallel algorithm:

 send - send, which prepares the message and sends it through the available means of transmission;

 receive - receive, waiting for a message and performing its receipt using the available message delivery means.

send operation is executed in the general flow of the algorithm, i.e. it will be executed immediately after the completion of the previous block, and after its execution (sending a message), control will pass to the block following it (Figure 5). Unlike the sender, the receive operation will wait for a message until it arrives, i.e., after the execution of the previous block, the execution of the sequence will be suspended until the message is received. After processing the received message, the sequence will continue.

As you know, information exchange can be asynchronous and synchronous. With asynchronous interaction, after sending a message, the execution process continues, and a response can be received at any time or not received at all (Figure

5, a). In a synchronous exchange, the sender, having sent a message, goes into a waiting state until a response is received (Figure 5, b).

The behavior of individual functional blocks of IAD algorithms depends on: initial data, knowledge model, and settings transferred to the block [Ошибка! Источник ссылки не найден.]. In this case, only the knowledge model is subject to change during the execution of the block. The settings and data remain unchanged while running the analysis algorithms. Thus, only the knowledge model is subject to exchange between parallel executing functional blocks.

FIGURE 5. SENDER AND RECEIVER BLOCKS IN THE STRUCTURE OF THE PARALLEL ALGORITHM:

a – asynchronous exchange, b – synchronous exchange

Consider the following typical situations in which the problem of message exchange arises in DM algorithms:

1. distribution of initial data between parallel executing branches of the algorithm;
2. collection of the obtained knowledge models for their integration and presentation of a single result;
3. exchange of intermediate results (knowledge models) between branches to correct further analysis.

In the first situation (Figure 6), the sender block distributes the vectors of the dataset among parallel threads in accordance with the internal algorithm. This block must be executed on the dispatcher thread. In this case, the remaining threads are executed when the next vector is received. Thus, in the described situation, asynchronous messaging is used.

An example of the described situation is the parallelization of the Naive algorithmBa yes [Ошибка! Источник ссылки не найден.] where each parallel branch calculates the Bayesian probability for a particular class. The sender block, which is executed in the dispatcher thread, determines for the next vector the class to which it belongs and, depending on this, passes it to the appropriate branch of the algorithm.

FIGURE 6. DISTRIBUTION OF INITIAL DATA BETWEEN PARALLEL EXECUTING BRANCHES OF THE ALGORITHM

When collecting the results (Figure 7), each thread sends the results to the dispatcher thread, which summarizes them and presents a single result. At the same time, the dispatcher thread is in the state of waiting for results from all other threads, and only when it receives results from all threads does it process them. In this case, asynchronous messaging also occurs.

An example of the above scenario can also be the Naive Bayes algorithm, in which parallel executing branches build knowledge models for individual classes, while processing the vectors passed to them. After processing all the vectors, all branches pass the knowledge models they have built (for each class) to the dispatcher thread. It combines knowledge models into a single analysis result.

FIGURE 7. COLLECTION OF THE RESULTING MODELS FOR THEIR INTEGRATION AND PRESENTATION OF A SINGLE RESULT

In the situation with the exchange of intermediate results (Figure 8), each thread after the formation of the intermediate result sends it to the other threads. When intermediate results are received from all threads, each thread processes them and continues execution. It should be noted that despite the presence of a bunch of sender-receiver blocks in each flow, this exchange is not synchronous, because in this case, the recipient does not expect a response to the sent message, but information that can be sent by another thread at any time (and before receiving a message from the first thread). In this case, the execution of the thread will continue only after receiving intermediate results from all parallel threads (including after receiving its own result).

FIGURE 8. EXCHANGE OF INTERMEDIATE RESULTS (KNOWLEDGE MODELS) BETWEEN BRANCHES TO CORRECT FURTHER ANALYSIS

An example of this situation is the algorithm for constructing classification rules C4.5[Ошибка! Источник ссылки не найден.],in which each parallel branch calculates the informativeness of the attributes for the data set defined by it. In this case, it is possible to exchange between the branches of the algorithm the calculated values of the information content of the attributes. After receiving this information, each branch determines the most informative attribute for the entire data set and, taking this into account, builds a decision tree.

V.CONCLUSION/RECOMMENDATIONS

The following results are obtained and presented in this chapter:

1. The analysis of DM algorithms was carried out and this made it possible to highlight a number of features of such algorithms.

2. From the analysis carried out, it can be concluded that the IAD algorithms contain the same blocks. The decomposition of algorithms into separate blocks and the selection of identical ones among them will reduce the effort for the development and debugging of DM algorithms. In addition, such block division allows the algorithm to be reformulated for its parallel execution.

REFERENCES

Analysis of data and processes: Proc. allowance for universities. 3rd ed. /
A. A. Barseghyan [and others] - St. Petersburg : BHV-Petersburg, 2009. - 512 p.;
Data mining in distributed systems / M. S. Kupriyanov [and others] - St. Petersburg : Publishing House of St. Petersburg Electrotechnical University "LETI", 2012. - 110 p.;

Amol G., Prabhanjan K., Edwin P., Ramakrishnan K. NIMBLE: A Toolkit for the Implementation of Parallel Data Mining and Machine Learning Algorithms on MapReduce . Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'11), San Diego, California, USA, August 21–24, 2011.P. 334-342;
Talia D. Parallelism in Knowledge Discovery Techniques. In Proceedings of the 6th International Conference on Applied Parallel Computing Advanced Scientific Computing . London , UK. Springer Verlag . 2002. P. 127–138;

Stolyarov L . N ., AbramovB . M . Beginningsinformatics . From task to program / L.N. Stolyarov, V.M. Abramov - M.: MAKET Publishing House, 2007. - 120 p.;

Karpov V. E. Introduction to the parallelization of algorithms and programs /
V. E. Karpov // Computer Research and Modeling. - 2010, - V. 2, No. 3, - P. 231-272;

Website of the Center for Advanced Studies of SPbSPU [Electronic resource] - Access mode: http://www.spbcas.ru/cfd/techn/Parallel.htm#ch2, free.- Zagl . from the screen;

Kholod I.I. Parallelization of the Naïve algorithmBayes based on block structure / I.I. Kholod, Z.A. Karshiev // Proceedings of the XV International Conference on Soft Computing and Measurement SCM`2012, St. Petersburg, June 25-27, 2012 г. - St. Petersburg, 2012. - Volume 1, -
P. 182-185;

Intelligent analysis of distributed data based on cloud computing / M.S. Kupriyanov [and others] - St. Petersburg : Publishing House of St. Petersburg Electrotechnical University "LETI", 2011. - 148 p.;

Kholod, I.I. Technique of parallelization of data mining algorithms / I.I. Kholod, Z.A. Karshiev// Izvestiya SPbGETU "LETI". - From St. Petersburg - 2013. - No. 3, - P. 38-45;

Kholod I.I., Karshiev Z.A., Shkolny R.E. Types of data distribution and the possibility of their intellectual analysis / I.I. Kholod, Z.A. Karshiev , R.E. School // Collection of reports of the XIV International Conference on Soft Computing and Measurement SCM`2011, St. Petersburg, June 23-25, 2011 - St. Petersburg 2011.- Volume 1, - P. 255-258;

PMML _Specification .: Data Mining group . – Access mode: http :// www . dmg . org / pmml - v4-0-1 . html , free .- Zagl . from the screen;

Noughton P., Shildt G. JAVA 2. - St. Petersburg. : B HV-Petersburg, 2007.–1072 p