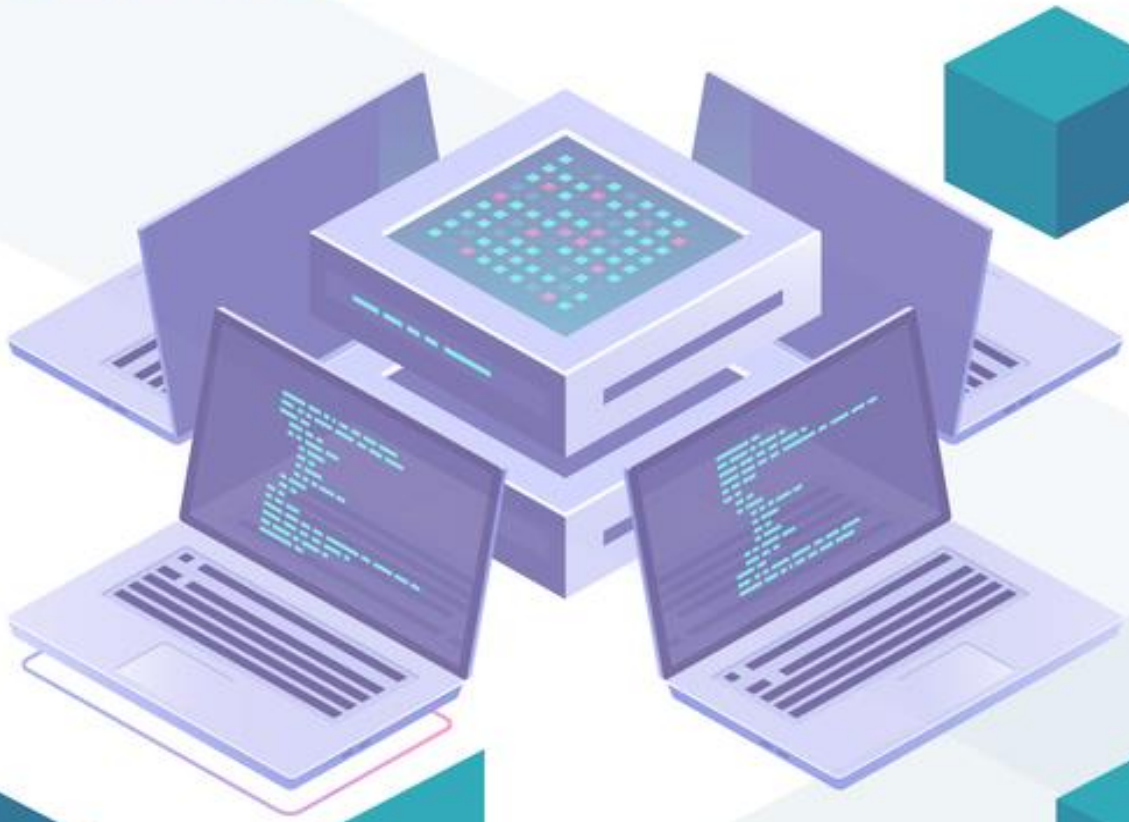
ISSN : 2348-5205

INTERNATIONAL JOURNAL OF

TRENDS COMPUTER SCIENCE



Indexed by:

ROAD Crossref Universal Impact Factor IMPACT FACTOR SEARCH

International journal of trends computer science ISSN: 2348-5205
https://academicjournalonline.org/index.php/ijtcs/
2022: 4.622
http://sjifactor.com/passport.php?id=22112

Editorial Team

10/25 Thamotharan Street, Arisipalayam, Salem, India

- Principal Contact
- Academic Journal Online
- info@academicjournalonline.org

International journal of trends computer science ISSN: 2348-5205
https://academicjournalonline.org/index.php/ijtcs/
2022: 4.622
http://sjifactor.com/passport.php?id=22112

EXISTING APPROACHES TO PARALLELIZATION OF DATA ANALYSIS ALGORITHMS

Dusmatova M.A. – Assistant of the Department of Information technologies of Samarkand Institute of Economics and Service. Uzbekistan. Samarkand.

Abstract- The article focuses on the promising directions of introducing special directives that support parallel processing tools in standard programming languages. Examples of such development are given. The advantages of the parallel processing approach are suggested.

Keywords - Realizing The Algorithm, Communication Tools, Data Mining, Narrow Range, Communication Information, Computational Experiment, Operating System.

I. INTRODUCTION

Various definitions of the algorithm, explicitly or implicitly, contain the following set of general requirements:

• Discreteness - the algorithm should represent the process of solving the problem as an ordered execution of some simple steps. At the same time, each step of the algorithm requires a finite period of time, that is, the transformation of the initial data into the result is carried out discretely in time.

• Determinism (certainty). At each moment of time, the next step of work is uniquely determined by the state of the system. Thus, the algorithm produces the same result (answer) for the same input data. In the modern interpretation, different implementations of the same algorithm must have an isomorphic graph. On the other hand, there are probabilistic algorithms in which the next step of work depends on the current state of the system and the generated random number. However, when the method of generating random numbers is included in the list of "input data", the probabilistic algorithm becomes a subspecies of the usual one.

• Clarity - the algorithm should include only those commands that are available to the performer and are included in his command system.

• Termination (finiteness) — in a narrower understanding of the algorithm as a mathematical function, with correctly specified initial data, the algorithm must complete the work and give the result in a certain number of steps. Donald Knuth calls a procedure that satisfies all the properties of an algorithm, except possibly finiteness, a computational method [4]. Quite often, however, the definition of an algorithm does not include finite-time termination[5]. In this case, the algorithm (calculation method) defines a partial function[en]. For probabilistic algorithms, terminability generally means that the algorithm produces a result with probability 1 for any given initial data (that is, it may not terminate in some cases, but the probability of this should be equal to 0).

• Mass character (universality). The algorithm must be applicable to different sets of initial data.

• Efficiency - completion of the algorithm with certain results.

II. LITERATURE REVIEW

The following scholars have considered the existing approaches to parallelization of data analysis algorithms in their research: Barseghyan A.A. [1], Kupriyanov M.S. [2], Amol G., Prabhanjan K., Edwin P., Ramakrishnan K. [3].

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

When solving a problem on a sequential computing system (one processor and one core), it is considered that five stages are performed:

1. Formulation of the problem;

2. Creation of a mathematical model;

3. Development of a solution algorithm within the framework of the created mathematical model;

4. Writing a program that implements the algorithm in one of the selected programming models and in the selected algorithmic language;

5. Execution of a program as a set of processes and/or threads of execution on a computer system and obtaining results.

At the fourth step, the programming model determines the main ideas and style of software implementation, abstracting from the algorithmic language and, in part, from the hardware component. For example, functional programming model, object-oriented programming model, production programming model, etc.

When solving a problem on a parallel computing system (several processors and/or several cores), additional stages appear in this set [1]. There are eight of them:

1. Formulation of the problem;

2. Creation of a mathematical model;

3. Algorithm development;

4. Algorithm decomposition (decomposition);

5. Assignment of work (assignment);

6. Choosing a software model and determining the required synchronization of the work of executors, which will largely depend on the software model;

7. Writing a program that implements the algorithm in the selected programming model and in the selected algorithmic language;

8. Mapping (mapping).

At the fourth step, when the algorithm is implemented in parallel, we assume that it will be executed by several executors. To do this, it is necessary to select in the algorithm sets of actions that can be carried out simultaneously, independently of each other - to decompose the algorithm. Decomposition is not always possible. There are algorithms that fundamentally do not allow the participation of several performers in their implementation.

After the successful completion of the decomposition stage, the entire

algorithm is a set of sets of action sets aimed at solving subtasks by individual performers.

At the last step, when running the program on a parallel computer system, it is necessary to match the virtual executors that appeared at the previous stages of the programming paradigm with real physical devices. Depending on the chosen programming model, this can be carried out both by the person conducting the computational experiment and by the operating system.

Parallelization methods can be conditionally divided into the following [2] :

1. Optimizing compiler directives;

2. Special directives that extend the language's capabilities for parallel execution;

3. Parallel programming languages;

4. Communication means, or interprocessor interface means.

Of course, the list of possible means is not limited to the listed positions, however, the above methods are the most widely used.

Parallelization by compiler directives is the simplest, "automatic" tool that can be effectively used for a selective class of tasks, but it has the following disadvantages:

a) intolerance from one system to another, or even when changing the compiler from one to another;

b) since the computational efficiency directly depends on the organization of the algorithm, for a goodresult, the programmer's "participation" is still required when writing parallel code;

c) a rather narrow range of possibilities provided by the compiler for parallelization.

The introduction of special directives that support parallel processing tools in standard programming languages is a promising direction. An example of such a development-highPerformanceFortran , which is a parallel extension of Fortran with parallel processing directives formatted as comments . The following advantages of

this approach can be noted [3] :

a) portability from one system to another source code, only the presence of High is requiredPerformanceFortran - compiler;

b) compatibility with the serial version of the program, since the High directivesPerformanceFortran , are perceived by the usual Fortran compiler as comments (thus, debugging of the sequential algorithm can be carried out on a machine without HighPerformanceFortran compiler);

c) clarity of the source code of the program, since it is not overloaded with "redundant" communication information, as, for example, in MPI;

d) ease of development by the user, who should only indicate to the compiler the blocks for parallel execution, and not specify what data and where to send it.

Widespread means of interprocessor data exchange include ParallelVirtualMachine - parallel virtual machine and MessagePassingInterface - programming using message passing.

It should be noted that a standard was published on MPI, which has a positive effect on the development of this tool, since computer manufacturers include support for MPI in their software products, and also, some massively parallel systems use only MPI as a means of interprocessor exchange.

Let's list the advantages of using MessagePassingInterface for program parallelization :

a) portability from one platform to another at the source code level. This interface can function both on systems with shared or shared memory, and on computers with distributed memory.

b) Message standardizationPassingInterface allows you to implement item a)and ensure that once written a parallel program will function on any platform that supports this standard.

c) greater flexibility in the development of parallel code, provided by a wide range of functions.

To the disadvantages, rather inherent not so much MessagePassingInterface ,

and parallelization tools in general, can be attributed to the difficulties of debugging for fairly complex programs.

Parallel and distributed data mining

DATA MINING is most effective when applied to a large amount of data, in which there is a lot of new knowledge (patterns), but which cannot be extracted by simple methods. However, the processing of large amounts of data is a very laborious task and takes a lot of time. In this regard, quite a lot of attention is paid to the issues of parallel and distributed execution of DATA INTELLIGENCE algorithms . In the field of INTELLIGENT DATA ANALYSIS, separate areas are even highlighted:

- parallel data mining (paralleldatamining ,) is designed to work with strongly coupled systems with shared (shared) or distributed memory and clusters (groups) of workstations with shared memory and fast interconnection;
- distributed INTELLIGENT DATA ANALYSIS (distributeddatamining , DDM) is designed to work with loosely coupled systems consisting of nodes connected to a local Ethernet network or geographically distributed nodes of the global Internet.

To better understand the main differences between paralleldata and distributeddatamining , you can consider PARALLELDATAMINING as a gradual transition from tightly coupled fine-grained parallel machines to loosely coupled medium-grained LAN workstations, and finally to very coarse-grained WANs.

On the other hand, PARALLEL DATA MINING can be presented as an essential component of the DDM architecture. Distributed data mining nodes can be workstations, supercomputers, or clusters of workstations with shared memory. In other words, each node supports PARALLEL DATA MINING locally. Several PARALLEL DATA MINING nodes constitute DISTRIBUTED DATA MINING, similar to the current trends of metacomputers and supercomputers.

Thus, the main differences between PARALLEL DATA MINING and DISTRIBUTED DATA MINING are scalability, communication costs, and data distribution. In PARALLEL DATA MINING, shared memory machines can share an

entire database and generate a global model, while shared memory machines typically share and allocate a database and then generate a global model. In DISTRIBUTED DATA MINING , you cannot share a database, and processors cannot exchange data with each other. First, local models are generated at each node, and then they are combined in different ways to get a global model.

PARALLEL DATA MINING is the ideal choice in organizations with centralized data stores, while DISTRIBUTED DATA MINING is effective in cases where there are multiple distributed databases. In fact, successful large-scale DATA MINING requires a hybrid PARALLEL DATA MINING/DISTRIBUTED DATA MINING approach , in which parallel methods are used to optimize the process of discovering hidden knowledge in local nodes and distributed methods, which are then used to build a global model.

V. CONCLUSION/RECOMMENDATIONS

We use algorithms in various fields, sometimes knowingly and sometimes unknowingly in our life. Algorithms are not only mathematical in nature, but we also use them a lot in our everyday life. For example, preparing food, making tea, doing a task, etc. We will follow specific instructions in order to do these things. If these instructions are not followed in a specific sequence, we will not get the desired result.

REFERENCES

[1] Barseghyan A.A. and others. Analysis of data and processes: Proc. allowance for universities. 3rd ed. / Petersburg, 2009. P. 512;

[2] Kupriyanov M.S. and others. Data mining in distributed systems / - St. Petersburg .: Publishing house of St. Petersburg Electrotechnical University, 2012. P. 110;

[3] Amol G., Prabhanjan K., Edwin P., Ramakrishnan K. NIMBLE: A Toolkit for the Implementation of Parallel Data Mining and Machine Learning Algorithms on Map Reduce . 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.