https://doi.org/10.33864/2309-4494.2026.n1.50-56

METHODS FOR OPTIMIZING DATA PROCESSING PERFORMANCE IN PARALLEL COMPUTING ENVIRONMENTS

Sabuhi Ahmadov Novruz oglu

Odlar Yurdu University, Baku city, Azerbaijan

ehmedovsebuhi22@gmail.com

Abstract. Despite the widespread use of modern multi-core architectures and cluster systems, the problems of maximizing performance in big data processing remain relevant. This article is devoted to a comprehensive analysis of Methods for Optimizing Data Processing Performance in Parallel Computing Environments. The main goal of the research is to justify the need to transition from the sequential computing paradigm to multi-core systems and to develop algorithmic and programming methods that increase efficiency. The article proves that optimal performance is achieved not only by minimizing the computing speed (total time, Tp), but also through the complex optimization of Resource Costs (C) and Communication Costs (Tcomm).

Keywords: Parallel Computing, Data Processing, Performance Optimization, Multi-core Systems, Computing Efficiency

1. Introduction.

The rapid development of information technologies and the large-scale data flow generated in various fields (scientific research, finance, industry, artificial intelligence) significantly exceed the capabilities of traditional sequential computing methods. This reality necessitates the transition to parallel computing environments and distributed systems for effective, fast and reliable data processing. The application of parallel computing allows large tasks to be divided into many small parts and executed simultaneously by several processors (cores), which significantly increases computing productivity.

The relevance of the topic is due to the fact that, despite the existence of modern multi-core and cluster architectures, the problems of improving performance in data processing in parallel environments have not yet been fully resolved. In particular, the development of methods for minimizing communication costs during parallel programming, optimal load balancing and efficient management of computing resources requires fundamental research.

The main purpose of the presented article is to analyze and develop effective methods and algorithms aimed at optimizing the performance of large-scale data processing in parallel computing environments.

To achieve this goal, the following main tasks are defined in the article:

1. Analysis of the architectural features of modern parallel computing systems and the theoretical foundations of data processing.
2. Identification of key performance indicators that assess the effectiveness of parallel algorithms.
3. Proposing optimization methods that increase the speed and efficiency of data processing, especially in terms of programming sequence and resource management.

The results of this research will provide a theoretical and practical basis for the design and programming of high-performance computing systems in any field working with big data (Machine Learning, Big Data Analysis and Scientific Computing).

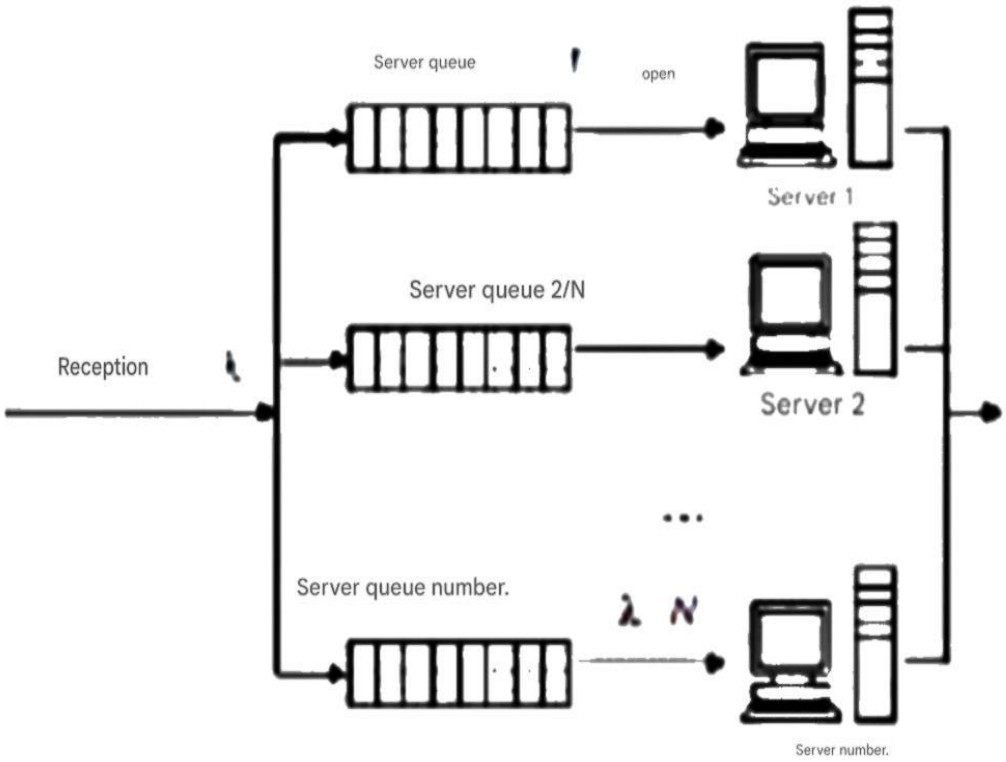

Figure 1. Multichannel data processing system

2. Experimental Section

Optimization of data processing performance is based on the complex relationship between the architecture of parallel computing systems, the algorithms used, and the programming models applied. In this context, the use of theoretical mechanisms and mathematical models for measuring and improving performance is of fundamental importance.

2.1. Performance Evaluation Criteria

The efficiency of data processing in parallel computing environments is mainly measured by time and resource efficiency criteria. Studies show that optimal performance is achieved as a result of the synthesis of two main parameters:

$$S(N) = \frac{1}{(1-p) + \frac{p}{N}}$$

• Computational Speed: Minimizing the total time (Tp) spent on processing the entire data volume. This criterion directly reflects the system productivity (P). In order to achieve maximum productivity, the intensity of the input data flow and the power of the processing devices must be taken into account.

• Resource Cost: Minimizing the total cost (C) of computing (processor, memory) and communication (network channels) resources used in parallel processing. This includes not only financial but also energy efficiency.

An effective optimization method should balance these two opposing criteria (time and cost).

2.2. Basic Principles of Optimization in Parallel Processing

Three main directions are identified for optimizing performance:

• Load Balancing: Distributing parallel tasks among all processors in such a way that idle time is minimized. This guarantees optimal use of resources.

• Communication Minimization: Interprocessor information exchange (synchronization, data transfer) creates additional costs called overhead. Optimization methods should reduce the communication time (Tcomm) by adjusting the relationship between data parallelism and task parallelism.

• Adaptation of Architectures: Depending on the type of data processing (e.g. matrix operations, database queries), the implementation of SIMD, MIMD or Hybrid architectures gives different results. Optimization should ensure that the chosen programming model (e.g. MPI, OpenMP, CUDA) works efficiently with these architectures [1, pp. 25-28].

In-depth Analysis of the Efficiency of Parallel Algorithms

Although the implementation of parallel computing reduces the execution time of tasks, this does not always happen linearly (as much as the number of cores). In order to measure the real efficiency of parallel algorithms, two main laws and metrics should be analyzed: 1. Amdal's Law; 2. Gustafson's Law

Amdal's Law shows that even when the percentage of parallel part (p) in data processing tasks is very high (e.g. 95%), the sequential part (5%) severely limits the speedup, even with an infinite increase in the number of processors. This proves the critical importance of optimizing the constraints arising from communication, synchronization, and data access sequencing.

$$S_s(N) = N - (N-1)f$$

Gustafson's Law shows that in the case of processing large-scale data (Big Data), when the task size is increased adequately to the number of processors, it is possible to achieve almost linear speedup. The optimization of performance in processing large-scale data, which is the main focus of your article, is best associated with this law [7]

3. Result and Discussion

Parallel processıng programmıng and optımızatıon

The application of theoretical models to improve performance should be directly integrated into the parallel programming sequence. The task of optimizing data processing belongs to the category of discrete programming from a mathematical point of view and is usually as complex as NP-complete tasks. Such complexity requires the development of efficient algorithms.[6]

3.1. Ensuring Performance in Parallel Programming

To achieve optimized performance, the following basic sequences and methods should be applied during programming:

• Module Parallelization: The data processing task should be divided into small modules that can be executed independently. During division, data dependencies between tasks should be minimized.

• Dynamic Resource Management: Instead of static distribution, dynamic load balancing algorithms that redistribute the workload among resources in real time should be applied.

• Selection of an Improved Programming Model: According to the requirements of the environment, a rational choice of programming approaches should be made, such as either LIP (Lazy-Interactive Parallelism), which operates with a higher communication cost, or HIP (High-Interactive Parallelism), which requires more frequent synchronization [3]

Analysis of the Impact of Modern Technologies on the New Approach

The application of effective methods for optimizing data processing performance in parallel computing environments requires moving away from the sequential programming paradigm and using the innovations offered by modern programming languages (C++, FORTRAN).

Older computing implementations were based on the use of only one processor, which led to the emergence of a circle of programmers developing sequential codes. However, the "supercomputer boom" that began in 2004 and the transition to multicore systems showed that the "one CPU" ideology was losing its relevance. High-performance systems, due to the slowdown in the growth rate of productivity (Graph 1), were forced to go the way of increasing the number of processor cores.[4]

This transformation directly affected the standards in programming languages:

• FORTRAN, as the main tool for high-performance mathematical calculations, became the main consumer of these systems [2]

• In C++, parallel cycles were introduced with the C++17 standard, and module technology was introduced in the C++20 standard. Modules greatly simplified the management of parallel code and programming consistency by eliminating the struggle with redefinition problems encountered in header files.

In this context, a significant difference arises between the concepts of "parallel code" and "parallel computing". Although directive-based technologies such as OpenMP and MPI are used to "force" old sequential codes to run on modern architectures, this does not fully replace true parallel computing. Analyses show that due to programmer conservatism, existing sequential codes lag far behind their potential speed. For example, if the growth rate up to 2004 had been maintained, sequential codes could run 12 times faster today [5].

Consequently, optimizing data processing performance requires abandoning the ideology of sequential code and using pure parallel methods provided by modern programming standards.

4. Conclusion

The conducted research and analysis have confirmed that optimizing data processing performance in parallel computing environments is one of the most pressing scientific and technical problems of the modern era.

Main Results and Scientific Innovations:

• The Necessity of Transition to Parallel Computing: It has been proven that the sharp slowdown in the growth rate of sequential code productivity after 2004 (as shown in Figure 1) indicates the ineffectiveness of the sequential code ideology. This necessitates the transition to increasing the number of cores (parallelism) as the only way to increase computational productivity.

• Complex Optimization of Criteria: It has been established that optimal performance is achieved not only by minimizing the Calculation Speed (total time, Tp), but also by maximizing the balancing of Resource Costs (C) (financial, energy, communication costs). An effective optimization method should take these two opposing criteria into account together.

• Performance Management Methods: Three key principles for optimizing performance — Load Balancing, Minimizing Communication (Tcomm), and Adapting Architectures — should be systematically applied.

• Role of Programming Standards: It has been analyzed that the parallel cycles and modular technologies introduced by modern programming standards such as C++17 and C++20 create higher and cleaner parallel computing capabilities than the adapted sequential codes based on the outdated OpenMP and MPI directives. Overcoming programmer conservatism and adopting these new standards is a key condition for fully realizing the potential of parallel processing.

Based on the results obtained, future studies can investigate the testing and real-time performance of these optimization methods in specific application areas such as Smart-Grid, Cloud Computing, or Big Data Analysis.

References

1.Шевченко С. В. ОПТИМИЗАЦИЯ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ В СИСТЕМАХ ПАРАЛЛЕЛЬНОЙ ОБРАБОТКИ ДАННЫХ. Интеллектуальные ИТ в управлении, ИТНОУ. 2018. № 3.

2.Reinders J. Vtune Performance Analyzer Essentials — New York: Intel Press, 2005.

3.Бобрышев Д. Н. (Bobrışev D. N.). Организация управления разработками новой техники. М.: Экономика, 1971.

4.Краснощеков П. С., Петров А. А. Принципы построения моделей. М.:МГУ, 1983.

5.https://www.google.com/amp/s/habr.com/ru/amp/publications/881488/

6.М. А. Максютов, Оптимизация параллельных вычислений в гетерогенных средах,Междунар. Науч.-исслед. Журн., 2023, выпуск 8

7.Hill, Mark D.; Marty, Michael R. (2008). “Amdahl’s Law in the Multicore Era”. Computer. 41 (7)