Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 12, No. 2, 2025

Algorithm Optimization and Performance Enhancement in Computer Science

Jinyi Jiang *

Faculty of Science and Engineering, University of Liverpool, Liverpool, L69 3BX, United Kingdom
*** Corresponding author Email:** Jinyi0208@outlook.com

Abstract: In the broad field of computer science, algorithm optimization is regarded as the cornerstone of improving system performance. With the rapid development of science and technology, the requirements for data processing and computing efficiency are increasing, and the optimization design of algorithms becomes critical. This paper will deeply discuss the method of algorithm optimization and how it can help significantly improve the computer performance in practical applications, Process data more efficiently, which prompts all work to improve efficiency accordingly.

Keywords: Computer Science; Algorithm Optimization; Performance Enhancement.

1. The Importance of Algorithmic Optimization

In today's digital world, algorithms are like the engine driving the development of modern science and technology, and they are ubiquitous, from the search engine's search result sorting, to the personalized recommendation of social media, to the real-time decision-making of financial transactions, all of them cannot be separated from the precise calculation and optimization of algorithms. Algorithm optimization, as one of the core technologies of computer science and software engineering, plays a crucial role in improving system performance, guaranteeing service quality and promoting technological innovation. [1]

Algorithm optimization is the key to big data processing. With the rapid development of the Internet, Internet of Things and other technologies, the size and complexity of data is growing exponentially. To deal with these massive data, algorithms need to be able to give accurate results within a limited time, otherwise, even the most powerful hardware can not cope with the challenges of massive data. Optimized algorithms can process data more efficiently, reduce computation time, reduce dependence on hardware resources, and make data analysis and utilization more feasible.

Optimization algorithms can improve the stability and reliability of the system. A well-performing algorithm is not only able to respond quickly to user requests, but also able to cope with a variety of abnormal situations to ensure the continuity of service. In highly concurrent and real-time demanding scenarios, such as financial trading systems or online game servers, the performance of the algorithm directly affects the user experience and system security. Optimizing algorithms can help reduce errors and exceptions, improve the robustness of the system, and thus ensure the stable operation of the service [2].

2. Algorithmic Optimization Strategies

In the practice of algorithm optimization, developers often use a series of strategies to improve the performance of algorithms. These strategies can be classified into different categories according to the problem characteristics and optimization goals, including the use of efficient algorithm design, optimization of data structures, parallel and distributed computing, and algorithmic improvement using specific techniques, etc. In the following, we will discuss each of these strategies and illustrate their application and effect through examples. In the following, we will discuss these strategies one by one, and illustrate their applications and effects through examples.

2.1. Choosing an Appropriate Algorithm Design That

Choosing the correct algorithm is the first step in the optimization process. Different algorithms have different time complexity and space complexity, which are suitable for different problem scenarios. For example, greedy algorithms are suitable for solving problems with optimal substructure and greedy selection properties. For example, Huffman Coding is used to construct the minimum weight path tree, so that the generated coding dictionary has the optimal compression effect. Dynamic programming is often used to solve problems with overlapping subproblems and optimization properties, such as finding the shortest path in the traveling salesman problem (TSP). The divide and conquer strategy performs well in dealing with large-scale data problems, such as quick sorting and merge sorting. They decompose large problems into small problems and then solve them recursively.

2.2. Optimizing the Data Structure

The selection and optimization of data structures have an obvious impact on the performance of the algorithm. Appropriate data structure can greatly reduce the time complexity of the algorithm. For example, when implementing the dictionary function, the hash table directly maps the key to the storage location through the hash function, and the time complexity constant level of access, insert and delete operations is significantly better than the linear search array. Balanced binary search trees (such as AVL trees or red black trees) ensure that the time complexity of search, insert and delete operations is close to O (log n), which is very effective for processing large amounts of ordered data.

2.3. Parallel and Distributed Computing

For large-scale data and computing intensive tasks, single

machine or single thread computing often cannot meet the performance requirements. At this time, parallel computing and distributed computing can significantly improve the efficiency of the algorithm. For example, when processing large-scale image data, the image can be divided into small pieces, and then processed in parallel on multiple processors. Distributed computing can distribute tasks to multiple computers for execution, such as Google's MapReduce framework, which is commonly used for big data processing. Through the two stages of "Map" and "Reduce", the data can be distributed for processing, and then the results can be summarized.

2.4. Genetic Algorithms and Evolutionary Computation

In some cases, the optimization problem may not have a clear optimal solution, or it is difficult to obtain the optimal solution directly. In this case, genetic algorithm simulating natural selection and genetic mechanism can be used for optimization. For example, the genetic algorithm used to solve the traveling salesman problem generates multiple possible solutions, and then iteratively optimizes through crossover, mutation and selection operations to finally find the near optimal solution [3].

2.5. Utilizing Tools and Libraries

Using performance evaluation tools, such as MATLAB Profiler or Python cProfile, can help developers identify the bottleneck of the algorithm, so as to carry out targeted optimization. In addition, many high-performance computing libraries (such as NumPy, Pandas, and TensorFlow) provide functions and operations optimized for specific problems, such as matrix operations, data cleaning, and deep learning, greatly simplifying the optimization process.

2.6. Caching and Preprocessing

When processing data intensive tasks, caching technology can improve the efficiency of data access and reduce the overhead of disk or network IO. Preprocessing data, such as data standardization, feature selection or dimension reduction, can help reduce the amount of computation and improve the efficiency of the algorithm.

2.7. Code Optimization

Streamlining and refactoring code can reduce unnecessary computation and memory consumption. The use of efficient data structures and algorithms, such as binary lookups instead of linear lookups, avoiding recursion and using loops, are common strategies for code optimization.

Algorithm optimization strategy is not a quick fix, but requires the combination of the characteristics of the problem, hardware resources, as well as the needs of the actual application, to choose the appropriate means for continuous iteration and optimization. Through continuous practice and learning, developers can master more optimization techniques and continuously improve the performance of algorithms, so as to maintain an advantage in the competition at the forefront of science and technology.

3. Performance Enhancement Practices

In real projects, algorithm optimization is a key step to improve system performance. The following two concrete cases show how to implement algorithm optimization in real applications, and discuss the use of performance testing and tuning tools.

Case 1: Search Sorting on E-Commerce Platforms

In e-commerce platforms, the ranking algorithm of search results is crucial to user experience. Suppose our platform processes millions of search requests every day. In order to provide faster response speed, we need to optimize the search sorting algorithm. First, we use the data structure to optimize, and store the search results as an ordered balanced binary search tree, so that the sorting can be completed in O (log n) time. Secondly, the cache technology is introduced to store the results of the most popular products recently queried in memory. For repeated query requests, the results are returned directly from the cache, avoiding frequent database access. Finally, through code refactoring, unnecessary calculations are removed to improve code efficiency. To evaluate the optimization effect, we use performance testing tools such as Apache JMeter to simulate a large number of concurrent requests, compare the response time before and after optimization, and ensure the effectiveness of the optimization strategy [4].

Case II: Image Processing and Recognition

In the field of image processing, such as face recognition applications, it is often necessary to process a large number of image data. In order to improve the recognition speed, we can adopt the parallel computing strategy. For example, using OpenCV and CUDA library, the image is divided into small blocks, and then feature extraction and matching are performed in parallel on GPU. At the same time, optimize data pre-processing, for example, use PCA to reduce the dimension of features to reduce the amount of calculation. Through Matlab's Profile or Python's cProfile tool, we can locate the performance bottlenecks in the code, such as CPU intensive operations, and then optimize these parts. Through these optimizations, the recognition speed can be significantly improved, making real-time facial recognition applications more smooth.

In practice, performance testing and tuning tools are important helpers in the optimization process. Apache JMeter, LoadRunner and other tools can help developers simulate real user loads and test the concurrent processing capability of the system. Code analysis tools such as MATLAB Profiler, Python cProfile, and Java VisualVM can show the resource consumption of code in detail, help us identify hot functions, and guide the optimization direction. For parallel and distributed computing, we can use distributed computing frameworks such as Mesos and Spark to schedule tasks and manage resources. At the same time, we can use the monitoring system provided by these frameworks to view the execution efficiency of tasks.

In the practice of performance enhancement, combining practical problems, choosing appropriate optimization strategies, and tuning with performance testing tools are effective ways to improve system performance. At the same time, continuous learning of new algorithms and techniques, such as the application of machine learning in algorithm optimization, can also provide a broader space for algorithm optimization. Remember, optimization is an iterative process, constantly try, test and improve, in order to truly play the potential of the algorithm, to provide users with a smooth experience, to create value for the enterprise.

4. Conclusion

In conclusion, algorithm optimization is an indispensable

part of computer science, which enables computer systems to handle complex data and computational tasks more efficiently through subtle design and implementation. With the continuous progress of science and technology, we expect to see more innovative algorithm optimization strategies, which will open up new possibilities for computer performance improvement. Let's witness the continuous development of this field and its far-reaching impact on the future world of science and technology.

References

[1] Xia Shang Jin The transformation and development of computer science and technology in the context of big data and artificial intelligence [J] Digital Communication World, 2024, (09): 206-208.

[2] Wang Yingjie, Zhang Chengye, Bai Fengbo, Wang Zumin, Zhu Jiuqi Exploration of AI platform course and micro specialty construction for computer science and technology [J] Research and Practice of Innovation and Entrepreneurship Theory, 2024, 7 (15): 10-13.

[3] Introduction to School of Computer and Information Engineering (School of Artificial Intelligence) of Nanjing University of Technology [J] School Party Building and Ideological Education, 2024, (15): 2.

[4] Huang Xianwu, Wang Jingyu, Zhang Jikai, Zhao Yuhong, Liu Lixin Exploration and practice of computer science and technology talents training mode based on integration of production and education [J] University Education, 2024, (15): 135-138.

[5] Dong Shunkai Application and development of computer science in the context of multimodal communication [J] Information and Computer (Theoretical Edition), 2024, 36 (14): 41-43.