Journal of Electronics and Information Science (2023)
Clausius Scientific Press, Canada

DOI: 10.23977/jeis.2023.080408
ISSN 2371-9524 Vol. 8 Num. 4

High performance data processing of distributed database and multi-core processor based on particle swarm optimization

Lixia Liu

College of Information Engineering, Engineering University of PAP, Xi'an, Shaanxi, 710086, China

Keywords: Particle swarm optimization; distributed database; multi-core processor

Abstract: As a product of the combination of computer network technology and database technology, distributed database system has the characteristics of independence and transparency, centralized node combination, replication transparency and easy expansion. However, due to its complex access structure, distributed database system naturally has a high demand for query optimization. This paper proposes a high-performance data processing method between distributed database and multi-core processors based on PSO (Particle Swarm Optimization) to solve the task scheduling problem between multi-core processors. Inertia weight is introduced, which is added to the speed of particle flight to adjust the global and local search ability of stationary particles. The research results show that this method reduces the error rate of database query, and the overall performance of database query method is better. The improved PSO algorithm improves the searching ability of particles by dynamically adjusting the inertia weight. Therefore, the improved PSO is a high-performance algorithm to solve the real-time task scheduling problem of multi-core processors.

1. Introduction

With the rapid development of the times, more and more users hold mobile data terminals, and their delivery of information with the Internet is increasing, which leads to an increasing amount of information brought by users. The emergence of distributed database is to meet people's demand for convenient access to massive data and the urgent need for super-large storage capacity [1]. As a product of the combination of computer network technology and database technology, distributed database system has the characteristics of independence and transparency, centralized node combination, replication transparency and easy expansion. However, due to its complex access structure, distributed database system naturally has a high demand for query optimization.

Task scheduling is always a NP-hard problem in multi-processing systems [2-3]. Therefore, how to assign different tasks to processors with different computing capabilities has become the primary problem whether the performance of multi-core processors can be fully exerted, which is also the reason why the research on task scheduling strategy has become a hot topic in multi-core processor technology research [4]. At present, heuristic algorithm and cuckoo algorithm are mostly used in the

research direction of task scheduling among multi-core processors. Literature [5] proposes a task scheduling algorithm for multi-core processors based on cuckoo search; Literature[6] proposes an improved clonal selection algorithm for solving TSP problem, which is used to solve the multi-task optimization combination problem in the processor.

Distributed analysis database mainly meets the needs of massive data storage and query analysis, and mainly meets the challenges of scalability and high availability. The distributed transaction database mainly solves the problem of distributed transactions [7-8]. PSO (Particle Swarm Optimization) is a commonly used random search algorithm, which has better search ability than traditional algorithms. In this paper, a high-performance data processing method of distributed database and multi-core processor based on PSO is proposed to solve the task scheduling problem between multi-core processors.

2. Data query optimization of distributed database

Compared with the traditional database, the distributed database has more advanced technology as its own support, which enables it to better meet the needs of big data processing. Synchronization technology mainly refers to the synchronization in the process of storing and transmitting data information, and also refers to the synchronization of all stations and nodes when receiving instructions. This is because many nodes and sites will work at the same time during the actual operation of distributed database. When users need to pick up items, they only need to look at the corresponding signs and tell the technicians what they need, and then they can get the items through the corresponding instructions. Therefore, the emergence of synchronization technology not only greatly improves the actual rate of users obtaining relevant data information, but also makes the information exchange between users more convenient.

The ultimate goal of database query optimization is to improve the performance of database system, but different query algorithms often make great differences in the execution efficiency of query operations. Database query optimizer is an integral part of relational database management system server. If it is cost-based optimization, the task of the database query optimizer is to optimize an SQL statement by generating alternative execution plans and finding the execution plan with the lowest estimated cost [9]. Even in the process of single table query, whether to select or project first will produce great efficiency difference. If we add the idea of distribution, this difference will become more and more obvious.

PSO is group-based, which moves individuals in the group to a good area according to their adaptability to the environment. However, unlike other evolutionary algorithms, PSO does not use evolutionary operators on individuals, but regards each individual as a particle without volume. The standard PSO affects the performance of the algorithm for speed and position. However, in practical application, there are many other factors that will affect the performance of the algorithm, which makes the stability of the standard PSO poor. Therefore, this paper improves the standard PSO from many aspects to improve its working stability.

Because the working environment of distributed database management system is very complex, there are many kinds of query schemes, and the best scheme should be searched as the final query result of distributed database management system, so the principle of solving query problems of distributed database management system is shown in Figure 1.

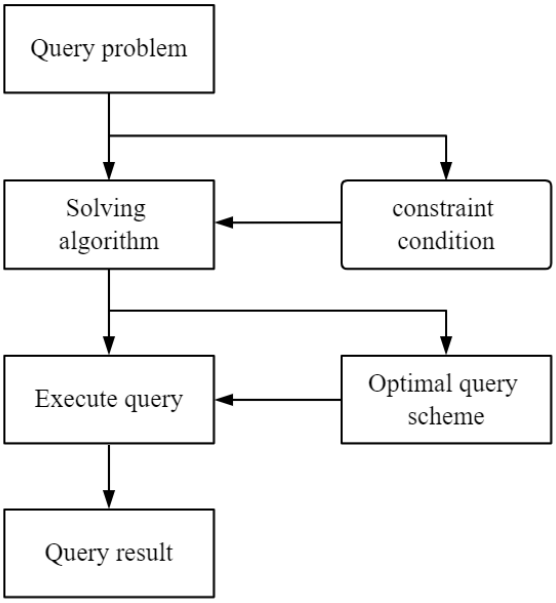

Figure 1: The principle of solving database query problem

The query optimization goal of distributed database management system is to minimize the total query cost and ensure the shortest query response time. The query cost calculation formula in a distributed database system can be expressed as formula (1).

$$T = I/O_{cost} + CPU_{cost} + COM_{cost} \tag{1}$$

Where COM_{cost} is the communication cost.

In the research of database query optimization based on PSO, this research chooses the left deep tree, and its process can be described as: "Numbering the relations contained in the query with n → coding the left deep tree", in which the coding order of the left deep tree needs to be combined with the sequence composed of leaf nodes, and the length of the sequence is n.

Inertia weight is introduced, which is added to the speed of particle flight to adjust the global and local search ability of stationary particles.

$$V_{id}^{k+1} = WV_{id}^{k} + c_1 r_1 \left(p_{id} - X_{id}^{k}\right) + c_2 r_2 \left(p_{gd} - X_{id}^{k}\right) \tag{2}$$

Where W stands for inertia weight.

In addition, the speed V_i of particles has a maximum speed limit. If the velocity V_{id} of a dimension exceeds the maximum velocity $V_{\max,d}$ of that dimension during the acceleration process, the velocity of that dimension is limited to the maximum velocity $V_{\max,d}$ of that dimension.

$$if\left(V_{id} > V_{\max,d}\right) V_{id} = V_{\max,d} \tag{3}$$

The first term is the previous velocity of the particle; The latter two terms are usually understood as the cognitive process of particles themselves and the interaction process between particles. The cognitive part represents the thinking of the particle itself, which strengthens the acquired correct knowledge and encourages the particle to reduce the error. The social part represents the process of information sharing and cooperation between particles, and the correct cognition of particles is imitated by other particles.

3. Thread scheduling of multi-core processor

The enhancement of processor performance has long relied predominantly on boosting its clock frequency. However, this approach is fast approaching its limits due to concerns related to power consumption and heat generation. As a result, the future of processor development increasingly points toward multi-core processors. In multi-core processor systems, the key to improving the efficiency of thread scheduling lies in effectively grouping threads and optimizing their scheduling. The Particle Swarm Optimization (PSO) algorithm has emerged as a viable solution for addressing these challenges. In comparison to Genetic Algorithms (GA), PSO operates without complex genetic operations such as reproduction, crossover, and mutation. Instead, it evolves through simple arithmetic operations, making it a practical choice for accelerating the search for optimal solutions.

Task scheduling in a multi-core processor environment involves the allocation of a set of tasks to appropriate processors while adhering to specific constraints[10]. The objective is to minimize the overall application completion time. This task scheduling problem is a combinatorial optimization challenge that has been proven to be NP-complete. Consequently, finding an optimal solution within polynomial time complexity remains a daunting task.

In practical PSO algorithm implementations, the consideration of information from neighboring particles during the particle update process introduces some challenges. As the number of iterations increases, a phenomenon known as "particle piling" occurs, leading to a gradual loss of diversity among particles. This phenomenon in the problem-solving process can result in premature convergence and a susceptibility to local optima. To mitigate these issues, the PSO algorithm often incorporates meta-heuristic algorithms for local search, thereby enhancing its convergence performance.

The quest for improved processor performance has been a driving force behind advancements in computer technology. Historically, processor performance improvements have been primarily achieved through increases in clock frequency. This approach, known as frequency scaling, has allowed processors to execute instructions at higher speeds, resulting in faster computational performance. However, as processors have continued to evolve and their clock frequencies have increased, they have encountered significant challenges related to power consumption and heat generation.

The limitations associated with increasing clock frequencies have led to the exploration of alternative approaches to enhancing processor performance. One such approach is the use of multi-core processors, which involve integrating multiple processing cores onto a single chip. These cores can execute tasks independently and in parallel, offering the potential for substantial performance gains without a significant increase in clock frequency.

The shift toward multi-core processors represents a fundamental change in processor design and architecture. Instead of relying solely on increasing clock frequencies to achieve performance improvements, the focus has shifted to optimizing the execution of parallel tasks across multiple cores. This shift has profound implications for various computing domains, including task scheduling algorithms.

Efficient thread scheduling is crucial for realizing the full potential of multi-core processors. In a multi-core environment, multiple threads can run concurrently, allowing for greater parallelism in task execution. However, effective thread scheduling involves assigning tasks to specific processor cores in a manner that optimizes overall performance. This task scheduling problem is known as the thread scheduling problem.

The thread scheduling problem is inherently challenging and falls into the category of combinatorial optimization problems. It involves selecting an assignment of threads to processor cores while considering various constraints, such as task dependencies and processor core

capabilities. The objective is to minimize the makespan, which is the total time required to complete all tasks.

Finding an optimal solution to the thread scheduling problem is known to be NP-complete, indicating that it is computationally intractable to solve within polynomial time complexity. As a result, researchers have turned to heuristic and meta-heuristic algorithms to address this complex problem efficiently.

One such meta-heuristic algorithm that has gained prominence in the context of multi-core processor task scheduling is the Particle Swarm Optimization (PSO) algorithm. PSO is a population-based optimization technique that draws inspiration from the social behavior of birds flocking or fish schooling. It models potential solutions to a problem as particles in a multi-dimensional search space and iteratively updates their positions based on their own experiences and the experiences of neighboring particles.

One of the notable advantages of PSO is its simplicity. Unlike Genetic Algorithms (GA), which involve complex genetic operations such as reproduction, crossover, and mutation, PSO relies on straightforward arithmetic operations for updating particles. This simplicity makes PSO relatively easy to implement and computationally efficient.

In the context of multi-core processor task scheduling, PSO operates by representing potential thread scheduling solutions as particles in a search space. Each particle's position corresponds to a candidate solution, and the quality of the solution is evaluated based on an objective function that reflects the makespan or another performance metric. PSO's key innovation lies in its ability to iteratively adjust the positions of particles to explore the solution space efficiently.

However, the PSO algorithm is not without its challenges. One common issue encountered during PSO optimization is the phenomenon of "particle piling." As particles update their positions, they tend to converge toward a limited region of the solution space. This convergence can result in premature convergence, where the algorithm settles on a suboptimal solution.

To mitigate the problem of premature convergence, researchers often incorporate various strategies, including the use of meta-heuristic algorithms for local search. These meta-heuristic algorithms complement PSO by providing a means of exploring diverse regions of the solution space, thereby enhancing the algorithm's ability to find optimal or near-optimal solutions.

In summary, the quest for improved processor performance has led to the exploration of multi-core processors as a viable alternative to frequency scaling. Thread scheduling in multi-core environments presents a challenging combinatorial optimization problem, with the objective of minimizing the makespan. The Particle Swarm Optimization (PSO) algorithm has emerged as a promising approach for addressing this problem. Its simplicity and effectiveness make it a valuable tool for optimizing thread scheduling in multi-core processors. However, researchers continue to explore ways to enhance the algorithm's performance and overcome challenges such as premature convergence through the integration of complementary techniques.

PSO algorithm is a continuous algorithm, and its updating formula and process are designed for continuous space, while the scheduling problem is a discrete problem. According to the characteristics of multi-core processor scheduling problem, this paper reconstructs particle expression, codes the position and speed of particles, and maps PSO algorithm to discrete space, making it suitable for solving task scheduling problem.

Each particle in PSO algorithm represents a potential solution of a task scheduling problem. Particle position vector is defined as a $n*m$ matrix X, with each column representing a task assignment and each row representing a processor execution. Formula (4) is the particle position coding scheme, and the constraint condition is formula (5).

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \tag{4}$$

$$s.\ t.\ \ x_{ij} \in \{0,1\},\ \sum_{i=1}^{m} x_{ij} = 1 \tag{5}$$

According to the constraint condition, the value of the element x_{ij} of the position matrix X is 0 or 1, each row can have multiple 1s, any element in each column can be 1, and only one element in each column can be 1.

4. Experimental analysis

In order to test the performance of the improved PSO in this paper, the test platform is AMD Ryzen 5 2600 XCPU, Weigang XPG DDR 43200 8G RAM, Kingston A1000 NVME M.2240 G hard disk and Linux operating system, and the programming environment is VC++6.0.

The success rate of all database query optimization methods is counted, and GA and standard PSO are selected for comparison test to analyze the advantages and disadvantages of database query optimization results. It carries out five test experiments, with 100 queries each time, and the results are shown in Figure 2.

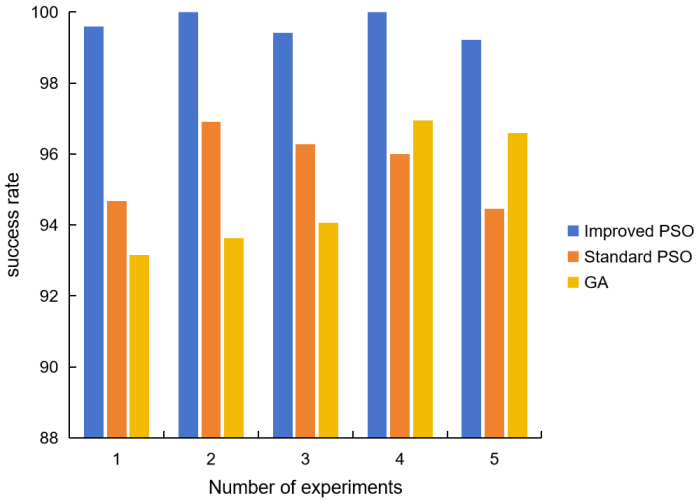

Figure 2: Comparison of database query success rate

It can be seen that the database query success rate of this method is much higher than that of GA and standard PSO, and the database query error rate is reduced, and the overall performance of the database query method is better.

With the increase of the number of processors and tasks, the convergence speed of the improved PSO algorithm is still very fast, and the system takes less scheduling time than the standard PSO and GA, and can achieve better scheduling results under time constraints, as shown in Figure 3.

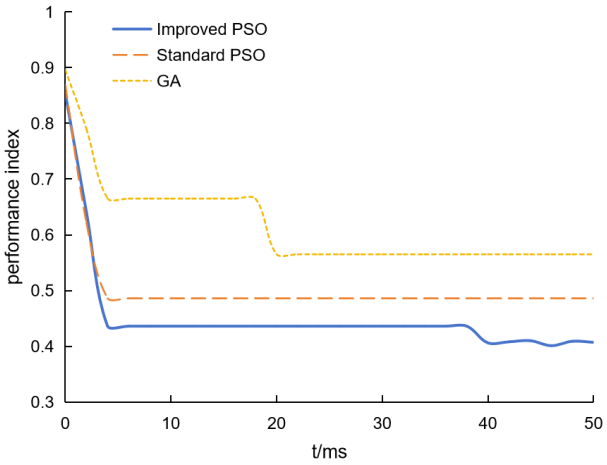

Figure 3: Scheduling results of 30 tasks assigned to 20 processors

The improved PSO algorithm improves the search ability of particles by dynamically adjusting the inertia weight. The simulation results show that the improved PSO algorithm can get better scheduling results in a very short time than other swarm algorithms.

5. Conclusions

PSO is a commonly used random search algorithm, which has better search ability than traditional algorithms. In this paper, a high-performance data processing method of distributed database and multi-core processor based on PSO is proposed to solve the task scheduling problem between multi-core processors. The database query success rate of this method is much higher than that of GA and standard PSO, and the database query error rate is reduced, and the overall performance of the database query method is better. The simulation results show that the proposed improved algorithm can get better scheduling results in a very short time than other group algorithms.

References

[1] Mohsin, S. A., Darwish, S. M., & Younes, A. (2021). Qiaco: a quantum ant system for query optimization in relational database. IEEE Access, (99), 1-1.

[2] Kim, H. J., & Kang, S. (2011). Communication-aware task scheduling and voltage selection for total energy minimization in a multiprocessor system using ant colony optimization. Information Sciences, 181(18), 3995-4008.

[3] Mahmood, A., Khan, S., Albalooshi, F., & Awwad, N. (2017). Energy-aware real-time task scheduling in multiprocessor systems using a hybrid genetic algorithm. Electronics, 6(2), 40.

[4] Guan, F., Qiao, J., & Wang, H. (2021). A multiprocessor real-time scheduling embedded testbed based on linux. International Journal of Embedded Systems, 14(5), 451.

[5] Davis, R. I., & Burns, A. (2011). Priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems. Real-Time Systems, 47(1), 1-40.

[6] Malik, A., & Gregg, D. (2015). Heuristics on reachability trees for bicriteria scheduling of stream graphs on heterogeneous multiprocessor architectures. Acm Transactions on Embedded Computing Systems, 14(2), 1-26.

[7] Davis, R. I., & Burns, A. (2011). Improved priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems. Real-time systems(1), 47.

[8] Rincon C., C. A., Zou, X., & Cheng, A. M. K. (2017). Real-time multiprocessor scheduling algorithm based on information theory principles. IEEE Embedded Systems Letters, (4), 1-1.

[9] Chon, H., & Kim, T. (2010). Resource sharing problem of timing variation-aware task scheduling and binding in mpsoc. Computer Journal, 53(7), 883-894.

[10] Shujuan, H., & Yian, Z. (2012). Improving multi-core scheduling method through using pptt (parallel priority task tree) model. Journal of Northwestern Polytechnical University, 30(5), 652-656.