2015 Seventh International Conference on Measuring Technology and Mechatronics Automation

Embedded database query optimization algorithm based on particle swarm optimization

Xiao mingyao, Li xiongfei
Key Laboratory of Symbolic Computation and Knowledge Engineering for Ministry of Education (Jilin University), Changchun 130012,China
xiaomingyao2014_cn@126.com

***Abstract*—This paper concentrates on the problem of embedded database query optimization, which is a crucial problem in embedded system design. Firstly, we describe the structure of the embedded database system, in which the database engine is a key module in the database system, and it can ensure the database system correctly and efficiently work. Secondly, the embedded database query optimization algorithm based on an improved particle swarm optimization is given. The main innovations of this paper lie in the following aspects: 1) a high inertia weight is used to find new searching space, 2) inertia weight decreases in terms of paths of different values of particle number, 3) final inertia weight is obtained after executing the max number of iterations. Thirdly, to test the effectiveness of our algorithm, we construct an experimental embedded system platform. Compared with the B+Tree, our proposed algorithm can achieve better performance in both space utilization and time cost.**

Keywords- Embedded system, Database query optimization, Particle swarm optimization, Fitness value.

I. INTRODUCTION

Embedded database management system refers to a new technology of data management rising in recent years. Exploiting the existing database technology and special embedded device, the data's storage, organization and management on mobile devices and embedded devices have been developed well[1][2]. In the traditional model, database systems are set on the high performance computer equipments, which contain host computer system and all types of servers[3].

With the rapid development of computer technology, embedded system has become an important field of the computer research field. Meanwhile, embedded database has increasingly shown its advantages, more and more researches. In particular, in the existing embedded operating system, Windows CE and Linux have attracted the most attentions[4]. Database application development based on Windows CE application development platform is an important part of developing handheld computer applications. Moreover, the SQLite technology has obtained more interests of those developers in embedded database field[5][6]. On the other hand, as the computer devices become smaller and smaller, the database's miniaturization is more and more important. Embedded system's database requires high reliability and real-time ability, because it usually works on the condition that without humans operation[7].

In this paper, we concentrate on how to promote the accuracy of embedded database query, and we creatively introduce the particle swarm optimization technology in our research. The PSO technology is firstly proposed by Kennedy and Eberhart. PSO refers to an effective solution to tackle many types of complex optimization problems without evolution operators[8][9]. Because there are a lot of parameters for PSO to be chosen, it achieves better performance than the traditional algorithm. The main innovations of PSO lie in that it contains local search process and global search process, and two processes are utilized to give a balance between exploration and exploitation[10]. Additionally, each particle in PSO has a fitness value which is evaluated by a given fitness function[11].

The rest of the paper is organized as follows. Section 2 analyzes the structure of embedded database system. In section 3, we introduce the PSO based embedded database query optimization algorithm. To test the effectiveness of the proposed algorithm, in section 4, experiments are conducted. Finally, the conclusions are provided in section 5.

II. EMBEDDED DATABASE SYSTEM

As is shown in Fig.1, the structure of the embedded database system is provided.

The database engine is the master control module of the database system, and it aims to achieve global control and ensure that the database system can correctly and efficiently coordinated work. It monitors the database during operation of all operations, control the allocation and management of resources, such as databases. Database engine interface layer is not required for processing the API call, and it is directly processed to determine the database operations.

Data access module implements all the basic operations on the base table, which is the core of the operating system to achieve database module. Its main functions lie in the following aspects: 1) finding a record based on property values, 2) finding records by the relative position, 3) adding a record to a base table, 4) deleting a record from the base table, 5) modifying a record and writing a result back to the base table.

Database maintenance module is used to backup and restore the database if it is necessary to complete the debris management. When frequent operating the database files (such as delete, modify, and indexing operations) may cause fragmentations. Fragmentations management can optimize the system effectively associated processing and storage.

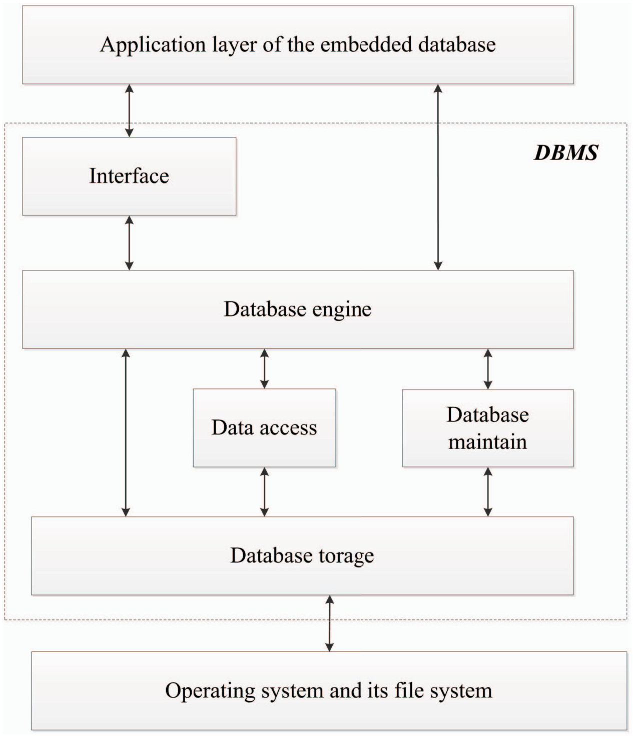
Application layer of the embedded database
DBMS
Interface
Database engine
Data access
Database maintain
Database torage
Operating system and its file system

Fig. 1 Structure of the embedded database system

III. THE PROPOSED ALGORITHM

For the original PSO, each solution is namded as a particle that can search space to achieve the optimal position. In each iteration, the task of PSO is to find 2 best solutions, that is, pbest and gbest. Pbest refers to the fitness which has been achieved till current iteration, and gbest is the global best solution.

$$v_i^d(N+1) = v_i^d(N) + c_1 \cdot rand1_i^d \cdot (pbest_i^d(N) - x_i^d(N)) + c_2 \cdot rand2_i^d \cdot (gbest^d(N) - x_i^d(N)) \quad (1)$$

$$x_i^d(N+1) = x_i^d(N) + v_i^d(N+1) \quad (2)$$

where $v_i^d(N)$ means the current velocity of a design variable. Parameter d denotes the d^{th} particle, and i is the i^{th} variable, and k refers to the iteration number. $rand1_i^d$ and $rand2_i^d$ represent two random numbers which are ranged from [0,1] respectively. Moreover, parameter c_1 and c_1 represent acceleration degrees, and the value of c_1 and c_1 are set to two. The weighting factors for prevent premature convergence is described as follows.

$$v_i^d(N+1) = \omega \cdot v_i^d(N) + c_1 \cdot rand1_i^d \cdot (pbest_i^d(N) - x_i^d(N)) + c_2 \cdot rand2_i^d \cdot (gbest^d(N) - x_i^d(N)) \quad (3)$$

where ω is utilized as a weighting factor.

Assuming that the searching space of dynamic PSO population includes D-dimension and m particles. For a given particle, it is represented as $X_i = (x_{i1}, x_{i2}, \cdots, x_{iD}), 1 \le i \le m$. The velocity of Particle i is represented as $V_i = (v_{i1}, v_{i2}, \cdots, v_{iD}), 1 \le i \le m$, and the best position of particle i is represented as $P_i = (p_{i1}, p_{i2}, \cdots, p_{iD}), 1 \le i \le m$. The destination of dynamic PSO is to find the best values of P_i, P_l, and P_g, then, the velocity and position of each particle are obtained as follows.

$$v_{id}(N+1) = w \cdot v_{id}(N) + r_1 \cdot (\alpha + \frac{1}{endgen - N + 1}) \cdot (p_{id}(N) - x_{id}(N)) + (p_{ld}(N) - x_{id}(N)) \cdot (b - \frac{1}{endgen - k + 1}) + cr_2(p_g(N) - x_{id}(N)) \quad (4)$$

$$x_{id}(N+1) = x_{id}(N) + v_{id}(N+1) \quad (5)$$

$$p_l(N+1) = p_l(N-1, N-2, \cdots, 1) \cup p_l(N) \quad (6)$$

where w is an inertia weight, and the initial value of w is set between [0,1].

Based on the above analysis, to enhance the PSO, we design a new version of PSO, which is named as dynamic PSO algorithm. The main innovations of our algorithm lie in the following aspects: 1) a high inertia weight is exploited to find new searching space, 2) inertia weight decreases according to different paths of different values of particle number, 3) final inertia weight can be obtained when reaching at the max number of iterations. Our proposed embedded database query optimization algorithm based on the improved PSO algorithm is given as follows.

```
Algorithm 1: Embedded database query optimization algorithm based on the improved PSO algorithm
Input: Parameter of the embedded database setting
Output: Embedded database query optimization scheme
(1) Defining a function to calculate the initial dynamic inertia weight:
    $\bar{\gamma} = \left(\frac{i_{max} - i_{cur}}{i_{max}}\right)^n \cdot (\gamma_{ini} - \gamma_{fin}) + \gamma_{fin}$
(2) Obtaining inertia weight as follows.
    $\gamma^* = (d)^r \cdot \gamma_{ini}, d \in [0.1, 1]$
(3) If $f(P_{gd-new}) < f(P_{gd-old})$
(4) Then $r = r - 1$
(5) Else $r = r + 1$
```

(6) Calculating stochastically initialization population and velocity and evaluating the fitness value of each particle.
(7) Initializing the $gbest$ position and the $pbest$ position.

(8) Initializing the $lbest$ position with the best particle of initializing population
(9) **End IF**
(10) **While** the max endgen of generation is not reached do
(11) $k++$
(12) Generating the next swarm and evaluating the swarm
(13) **End while**
(14) **Return** the optimization results as the embedded database query optimization scheme.

IV. EXPERIMENT

In this section, we test the proposed algorithm in the embedded system platform. Particularly, the server utilized in this experiment is IBM xSeries236 with 2GB RAM, Windows Server 2003 Enterprise Edition, and Oracle 9i. We choose a PDA provided by Invengo Company to construct the embedded system platform, and this PDA is built by ARM920T dxA27x with 64MB RAM. The embedded database is SQLite3.3.4. To make performance comparison, the B+tree algorithm[12][13] is used in embedded database query.

Firstly, to test the space utilization, each search key is set to 8 bytes, we choose different number to test the data query performance. space utilization is calculated as follows.

$$space\ utilization = \frac{key\ space}{actual\ space} \times 100\% \quad (7)$$

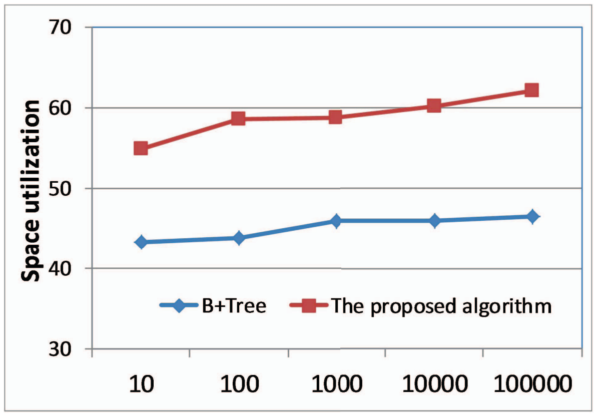

Fig. 2 Performance evaluation for space utilization

Fig. 2 demonstrates that the proposed algorithm achieve higher space utilization than B+Tree.

Secondly, we test the time efficiency for random query, to make the experimental results more objective, we average the experimental results for 15 times. Using the SQlite 3.3.4 software system, the experimental results are shown in Fig.3.

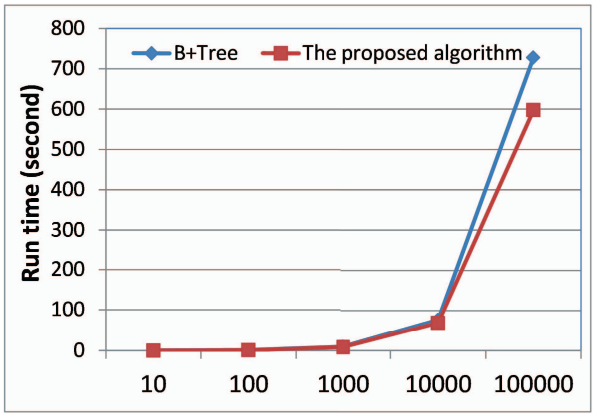

Fig.3 Performance evaluation for random query efficiency

Fig. 3 shows the runtime used for our proposed algorithm is lower than B+Tree, because the time complexity of B+Tree is $O(\log_m n)$, and our proposed algorithm has the ability to promote the query efficiency by fully exploiting the node space.

Thirdly, we will test the time cost in sequence query, and the experimental result is illustrated in Fig. 4 as follows.

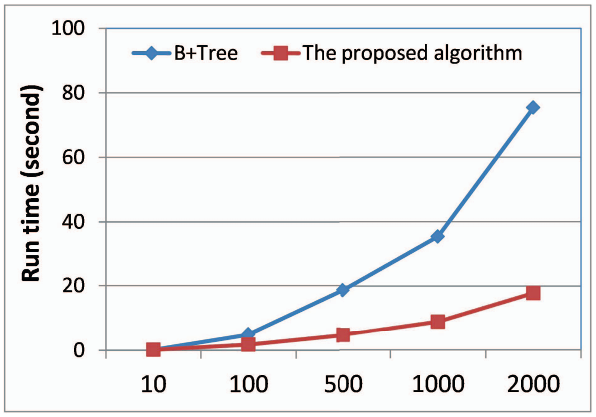

Fig. 4 Performance evaluation for sequence query efficiency

Fig. 4 demonstrates that our proposed algorithm can effectively reduce time cost than B+Tree, the reason lies in that the array is memorized sequentially in the memory, and the leaf nodes of B+Tree utilize the link list structure.

Fourthly, we test the insertion performance of our proposed algorithm and B+Tree in Fig. 5.

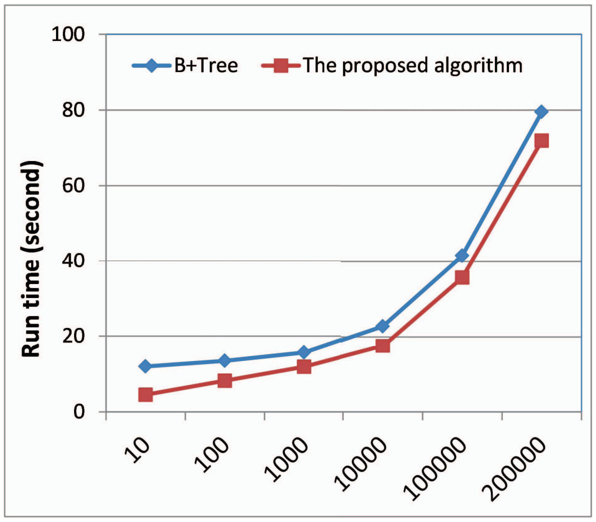

Fig. 5 Performance evaluation for insertion efficiency

Fig. 5 shows that the insertion time of B+Tree is close to our proposed algorithm when a large number of data records are inserted into the embedded database.

Integrating all the experimental results together, the conclusions can be drawn that our proposed achieve better performance than B+Tree in the problem of embedded database query optimization.

V. Conclusions

This paper focuses on the problem of embedded database query optimization, which is of great importance in embedded system design. The main innovations of our proposed algorithm lie in 1) a high inertia weight is used to find new searching space, 2) inertia weight decreases in terms of paths of different values of particle number, 3) final inertia weight is obtained after executing the max number of iterations.

In the future, we will extend our works in the following aspects:

(1) We will try to extend the scale of embedded system to test if our algorithm is suitable to be used in large scale database.

(2) We will use some other types of embedded system platform in the experiment.

References

[1] Huang Po-Chun, Chang Yuan-Hao, Lam Kam-Yiu, Wang Jian-Tao, Huang Chien-Chin, Garbage Collection for Multiversion Index in Flash-Based Embedded Databases, ACM Transactions on Design Automation of Electronic Systems, 2014, 19(3), Article No. 25.

[2] Park Sangwon, Flash-Aware Cost Model for Embedded Database Query Optimizer, Journal of Information Science and Engineering, 2013, 29(5): 947-967.

[3] Kaart M., Klaver C., van Baar R. B., Forensic access to Windows Mobile pim.vol and other Embedded Database (EDB) volumes, Digital Investigation, 2013, 9(3-4): 170-192.

[4] Park Sangwon, Energy-aware Cost Model for Flash-aware Embedded Databases, Information-an International Interdisciplinary Journal, 1343-4500, 2012, 15(11B): 5099-5116.

[5] Kang Woochul, Son Sang H., Power- and time-aware buffer cache management for real-time embedded databases, Journal of Systems Architecture, 2012, 58(6-7): 233-246.

[6] Hjertstrom Andreas, Nystrom Dag, Sjodin Mikael, Data management for component-based embedded real-time systems: The database proxy approach, Journal of Systems and Software, 2012, 85(4): 821-834.

[7] Kang Woochul, Son Sang Hyuk, Stankovic John A., Design, Implementation, and Evaluation of a QoS-Aware Real-Time Embedded Database, IEEE Transactions on Computers, 2012, 61(1): 45-59.

[8] Xu Gang, Yang Yu-qun, Liu Bin-Bin, Xu Yi-hong, Wu Ai-jun, An efficient hybrid multi-objective particle swarm optimization with a multi-objective dichotomy line search, Journal of Computational and Applied Mathematics, 2015, 280: 310-326.

[9] Upadhyay Subho, Sharma M. P., Development of hybrid energy system with cycle charging strategy using particle swarm optimization for a remote area in India, Renewable Energy, 2015, 77: 586-598.

[10] Lu Wei, Du Xiaoyong, Hadjieleftheriou Marios, Ooi Beng Chin, Efficiently Supporting Edit Distance Based String Similarity Search Using B+-Trees, IEEE Transactions on Knowledge and Data Engineering, 2014, 26(12): 2983-2996.

[11] Fang Hua-Wei, Yeh Mi-Yen, Suei Pei-Lun, Kuo Tei-Wei, An Adaptive Endurance-Aware B+-Tree for Flash Memory Storage Systems, IEEE Transactions on Computers, 2014, 63(11): 2661-2673.

[12] Rasoulzadeh-akhijahani A., Mohammadi-ivatloo B., Short-term hydrothermal generation scheduling by a modified dynamic neighborhood learning based particle swarm optimization, International Journal of Electrical Power & Energy Systems, 2015, 67: 350-367.

[13] Jin Cong, Jin Shu-Wei, Automatic image annotation using feature selection based on improving quantum particle swarm optimization, Signal Processing, 2015, 109: 172-181.