Research on IFOA-GA Task Scheduling Algorithm in Cloud Computing MapReduce Model (Postprint)

Authors: Chen Xuan, Pan Chunping, Long Dan

Date: 2018-08-13T00:00:00+00:00

Abstract

To address the problems of low efficiency and low utilization in traditional cloud computing task scheduling algorithms, a hybrid algorithm combining the improved fruit fly optimization algorithm (IFOA) and genetic algorithm (GA) is proposed for task scheduling. First, task scheduling is transformed into a DAG (directed acyclic graph) and the task scheduling order is simplified through Kruskal' s algorithm; second, the population of the fruit fly algorithm is initialized using orthogonal arrays and quantization techniques, boundary handling is applied to the fruit fly algorithm, the exploration step size is dynamically adjusted, and individual selection is performed using the GA; finally, the fused IFOA-GA algorithm is applied to cloud computing task scheduling in a simulation platform, where it demonstrates certain advantages over IGA, IFOA, and IPSO algorithms in the comparison of four QoS metrics, demonstrating that the IFOA-GA algorithm can effectively improve cloud computing scheduling efficiency.

Full Text

Preamble

Title: Research on IFOA-GA Task Scheduling Algorithm in Cloud Computing MapReduce Model

Authors: Chen Xuan[1], Pan Chunping[1], Long Dan[2]
(1. Zhejiang Industry Polytechnic College, Shaoxing, Zhejiang 312000, China; 2. Zhejiang University, Hangzhou 310058, China)

Abstract: Traditional cloud computing task scheduling algorithms suffer from low efficiency and poor resource utilization. To address these issues, this paper proposes a hybrid algorithm that combines an improved fruit fly optimization

algorithm (IFOA) with a genetic algorithm (GA) for task scheduling. First, the task scheduling problem is converted into a directed acyclic graph (DAG) and simplified using Kruskal' s algorithm to determine task execution order. Second, the fruit fly algorithm' s population is initialized using orthogonal arrays and quantization techniques, with boundary handling for the search space and dynamic adjustment of exploration step sizes. The GA is employed for individual selection. Finally, the hybrid IFOA-GA algorithm is applied to cloud computing task scheduling in a simulation platform. Compared with IGA, IFOA, and IPSO algorithms, IFOA-GA demonstrates advantages across four QoS metrics, indicating that it can effectively improve cloud computing scheduling efficiency.

Keywords: cloud computing; task scheduling; fruit fly algorithm; population initialization; boundary processing

0 Introduction

Cloud computing task scheduling has always been a critical research component in cloud computing, representing a multi-objective optimization problem and an NP-hard problem. Traditional task scheduling methods are clearly inadequate for cloud computing environments, making intelligent algorithms the primary research direction for solutions [1]. Scholars have investigated various approaches, including improvements to bat algorithms [2], genetic algorithms [3], ant colony algorithms [4-5], and particle swarm algorithms [6], achieving promising results.

Reference [7] proposed a multi-objective optimization scheduling model based on fireworks algorithm, with experiments demonstrating good scheduling efficiency. While this approach leverages the high performance of fireworks algorithm for rapid scheduling, its limitation lies in comparison only with basic PSO and GA algorithms, lacking evaluation against more recent algorithms of the same type. Reference [8] introduced a cloud computing task scheduling method based on bat algorithm, improving scheduling performance under cloud computing through population initialization and Powell' s local search. Although this enhances virtual machine processing efficiency, it only compares with clustering algorithms and lacks broader comparative validation, limiting its persuasiveness. Reference [9] proposed an improved genetic algorithm for cloud computing that uses bidirectional convergence ant colony operators to obtain precise solutions, improving accuracy and convergence speed but increasing algorithm complexity and solution time. Reference [10] presented a hybrid of improved ant colony and particle swarm algorithms for cloud computing resource scheduling, reducing time consumption and cost but decreasing algorithm precision.

Building upon these research findings, this paper proposes applying an improved fruit fly algorithm to cloud computing task scheduling. Simulation experiments demonstrate superior performance compared with other intelligent algorithms.

1.1 Task Definition Based on MapReduce Model

Currently, most cloud computing task models adopt the MapReduce framework, which is an efficient task scheduling model whose workflow has been extensively documented in MapReduce literature and will not be repeated here. However, the MapReduce model has certain limitations [11]: (1) user-submitted tasks are not always simple; (2) tasks cannot always be decomposed into parallel subtasks, as precedence relationships exist between subtasks; and (3) the Reduce phase requires inner iterations to complete before outer iterations can proceed.

To address these issues, this paper introduces new definitions for the MapReduce model.

Definition 1 (Complex Task): A complex task is a system composed of a set of interdependent subtasks, defined as $T = \{t_1, t_2, ..., t_n\}$ where each t_i is an executable task. The precedence relation $\prec$ is a partial order on T indicating execution priority, where $t_i \prec t_j$ means t_j must start after t_i begins. D_0 is the communication matrix where $d_{ij} > 0$ represents the data volume from t_i to t_j. A is an n-dimensional vector where $a_i > 0$ represents the computational load of task t_i.

Clearly, subtasks comprising complex tasks in cloud computing have precedence constraints, which can be represented using a DAG $G = (T, E)$. Here T represents the set of nodes (node weights indicate task processing time), and E represents the set of edges (edge weights indicate data dependencies and communication time). In a DAG precedence constraint relationship, a node can only execute after all its predecessor nodes have completed. When two nodes are assigned to the same processor, their communication cost becomes zero.

Definition 2: For a directed graph G, let $ID(v)$ denote the number of edges ending at vertex v, and $OD(v)$ denote the number of edges starting from vertex v.

Definition 3: A spanning tree of a DAG must satisfy: (1) it has the same vertices as G; (2) it has sufficient edges to connect all vertices in G; and (3) it contains no cycles. When each edge of the DAG is assigned a weight, the spanning tree with the minimum total edge weight is the minimum spanning tree. This paper uses Kruskal' s algorithm to convert the DAG into a minimum spanning tree, effectively eliminating redundant dependencies while preserving the most essential task relationships. Figure 1(a) shows task relationships in the DAG, while Figure 1(b) depicts the resulting minimum spanning tree. Both graphs contain identical nodes, successfully simplifying redundant dependencies in the DAG.

[Figure 1: see original paper]

1.2 Building QoS-Based Task Scheduling Evaluation Model

This paper adopts Quality of Service (QoS) as a critical mechanism for evaluating task scheduling effectiveness in the MapReduce model. Based on general QoS principles, four evaluation metrics are established: completion time, cost, reliability, and energy consumption.

1. **Completion Time:** The total time a task spends in cloud resource scheduling. This is typically the primary concern for users and a crucial standard for measuring scheduling effectiveness.
2. **Cost:** Task scheduling consumes cloud resources such as CPU, memory, and storage, each with associated usage costs. Therefore, tasks should be completed with minimal cost.
3. **Reliability:** Since cloud computing transforms physically distributed computers into a logically unified virtual system, overall system reliability is fundamental to task scheduling. As reliability cannot be measured directly with formulas, user satisfaction is used to reflect system reliability.
4. **Energy Consumption:** Task scheduling inevitably involves energy consumption due to network bandwidth and hardware usage when different tasks are assigned to different resources. Energy consumption cannot be ignored in task scheduling.

To achieve efficient QoS, all four evaluation criteria must be satisfied simultaneously. Let T_{C_i}, C_{S_i}, r_i, and E_i represent completion time, cost, reliability, and energy consumption for the i-th task scheduling solution, with w_1, w_2, w_3, w_4 as their respective weights. The evaluation function for solution i is:

$$F(i) = w_1 \cdot T_{C_i} + w_2 \cdot C_{S_i} + w_3 \cdot r_i + w_4 \cdot E_i \quad (1)$$

The optimal task scheduling solution based on QoS is found by minimizing this function:

$$BestF = \min\{F(i) \mid i = 1, 2, ..., n\} \quad (2)$$

---

2. Fruit Fly Optimization Algorithm (FOA)

In 2011, Pan proposed a novel swarm intelligence optimization algorithm called Fruit Fly Optimization Algorithm (FOA), which simulates the foraging behavior of fruit fly populations using a cooperative group mechanism. The FOA algorithm consists of the following steps:

Step 1: Set population size $PopSize$, maximum iterations $MaxNum$, group range L, and individual flight range Fr. The position of each fruit fly is given

by 2D coordinates. The initial position is:

$$\begin{cases} X_{axis} = rand(L, r) \\ Y_{axis} = rand(L, r) \end{cases}$$

Step 2: Olfactory Search Process

Step 2.1: When a fruit fly searches via olfaction, it is assigned a random flight direction and distance:

$$\begin{cases} X_i = X_{axis} + rand(Fr) \\ Y_i = Y_{axis} + rand(Fr) \end{cases}$$

Step 2.2: The smell concentration value is inversely proportional to the distance from the food source. First calculate the distance $Dist_i$ from the individual to the origin using equation (5), then compute the smell concentration value S_i using equation (6).

Step 2.3: Calculate the smell concentration $Smell_i$ for each fruit fly using equation (7), where $fitness(S_i)$ is the concentration judgment function (i.e., the objective function to be optimized).

Step 3: Visual Search Process
The fruit fly population flies toward the position with best concentration found in Step 2.

Step 4: Repeat Steps 2 and 3 until reaching maximum iterations. The position with best concentration is the optimal solution.

The FOA algorithm is simple, comprising only olfactory and visual search components, with key parameters being population size and maximum iterations. Literature [12-13] demonstrates that FOA has good global search capabilities compared to other intelligent algorithms. However, its limitations include insufficient local search ability, unstable optimization results, and premature convergence.

---

3. IFOA-GA Based Cloud Computing Task Scheduling Strategy

To address FOA' s limitations, this paper improves four aspects: population initialization, boundary handling, step size transformation, and individual selection, while integrating GA to create the new IFOA-GA algorithm for cloud computing task scheduling.

3.1 Population Initialization

FOA lacks population initialization, which may cause many invalid solutions in early stages. To ensure effective solutions are uniformly distributed in the solution space, this paper employs orthogonal experimental design based on orthogonal arrays and quantization techniques for FOA population initialization. This method enables initial solutions to be evenly distributed in the feasible region, improving optimal solution search speed.

Step 1: Divide the FOA population problem feasible domain into S subsets.

Step 2: After quantizing subset S Q times, obtain $S_1, S_2, ..., S_Q$.

Step 3: Select M individuals with highest fitness from N individuals as the initial population.

The orthogonal array $L_{N\times M}(Q^N)$ is constructed as follows: when Q is odd, construct the basic column $a_{i1} = i - 1$ for $i = 1, 2, ..., Q$. For $j = 2, 3, ..., N$, when $s = 1, 2, ..., Q - 1$ and $t = 1, 2, ..., Q^{j-1}$, compute:

$$a_{i,j} = (a_{i,j-1} + a_{t,1}) \mod Q$$

3.2 Boundary Handling

FOA individuals may exceed feasible domain boundaries. When dimension k of individual x_i exceeds boundaries, it is mapped to a new position using equation (14):

$$\hat{x}_{ik} = \begin{cases} x_{LB,k} + \cos(\alpha) \cdot (x_{UB,k} - x_{LB,k}), & x_{ik} < x_{LB,k} \\ x_{UB,k} + \cos(\beta) \cdot (x_{UB,k} - x_{LB,k}), & x_{ik} > x_{UB,k} \end{cases}$$

where $x_{UB,k}$ and $x_{LB,k}$ are upper and lower bounds for dimension k, and $x_{o,k}$ is the origin.

Figure 2 illustrates the boundary handling process. For upper boundary violation, a circle centered at origin x_o with radius m intersects the x-axis at the new position $\hat{x}_{ik}$. Lower boundary violation is handled similarly. This approach maintains relative positions while keeping data characteristics intact. Figure 3 shows boundary violation scenarios before and after handling.

[Figure 2: see original paper]
[Figure 3: see original paper]

3.3 Search Step Size Improvement

FOA sets a fixed step size when generating new positions, which balances global and local search capabilities. Large step sizes enhance global search and convergence speed but reduce local search precision, while small step sizes have the opposite effect. This paper proposes an adaptive step size improvement:

$$\begin{cases} X_i = X_{axis} + rand(Fr) \cdot H_i \\ Y_i = Y_{axis} + rand(Fr) \cdot H_i \end{cases}$$

where $H_i = L \cdot \alpha_i$, and α_i is defined as:

$$\alpha_i = \begin{cases} (1 + \sin(\frac{\pi \cdot i}{T})), & i \leq T \\ (1 + \sin(\frac{\pi \cdot (i \mod T)}{T})), & i > T \end{cases}$$

Here L is the search space length, T is the iteration count, and i is the current iteration. The step size variation is shown in Figure 4, which demonstrates that this approach divides the search process into multiple cycles, enhancing diversity and reducing local convergence likelihood. The periodic step size variation effectively controls the impact of step size changes on the algorithm.

[Figure 4: see original paper]

3.4 Individual Selection Improvement

In standard FOA, individuals are selected based solely on their own concentration values, ignoring their impact on future generations and the overall population. This paper improves selection using GA concepts.

(1) Individual Selection:
Fitness values are determined from concentration values. However, simply using individual optimal concentration as fitness has drawbacks, as it ignores the relationship between individual and population concentrations, potentially causing local optima. This paper constructs a fitness function based on forward sequencing, sorting individuals by objective function values in descending order, then mapping them to fitness values:

$$f_i = \sin(x_i), \quad i = 1, 2, ..., N$$

where $x_i = \frac{N+1-i}{N}$. The difference between adjacent individuals' fitness values is:

$$f_i - f_{i+1} = \frac{\sin(1)}{N}$$

The selection probability for individual i using roulette wheel selection is:

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$$

The probability difference between adjacent individuals is small, ensuring selection diversity, especially as population size increases.

(2) Mutation Operation:
The i-th fruit fly individual x_i^g in generation g is mutated using:

$$V_i^{g+1} = x_i^g + r_1 \cdot (F - x_i^g)$$

where r_1 is a random factor in $[0, 1]$ controlling scaling.

(3) Crossover Operation:
With certain probability, crossover occurs between mutated individual V_i^{g+1} and parent x_i^g to produce new individual y_i:

$$y_i = \begin{cases} v_{i,j}, & \kappa_{i,j} \in [0, 1] \\ x_{i,j}, & \text{otherwise} \end{cases}$$

This ensures at least one component of y_i comes from V_i^{g+1}.

3.5 Algorithm Flow

Step 1: Represent cloud computing tasks as a DAG, simplify to a minimum spanning tree using Kruskal' s algorithm, remove redundant information, and map tasks to fruit fly individuals. Set iteration parameters.

Step 2: Initialize the fruit fly population using equations (11-13).

Step 3: Update positions during olfactory search using equation (19), execute equations (5-8). During visual search, apply boundary handling from equation (14) if boundary violation occurs.

Step 4: Update fruit fly individuals using equations (20-26).

Step 5: Check if maximum iterations reached. If yes, proceed to Step 6; otherwise return to Step 3.

Step 6: Algorithm terminates. The best fruit fly individual corresponds to the optimal task scheduling solution.

The algorithm flowchart is shown in Figure 5.

[Figure 5: see original paper]

3.6 Algorithm Simulation

To validate IFOA-GA's effectiveness, experiments were conducted on a hardware platform with Intel Core i5-7500 CPU, 4GB DDR3 memory, and 1TB hard drive, running 64-bit Windows with CloudSim simulation platform. IFOA-GA was compared with IFOA [15], IPSO [16], and IGA [17] algorithms. Maximum iterations were set to 100, with task quantities ranging from 10,000 to 100,000. Algorithm parameters followed their respective literature.

Comparisons were made across four dimensions: completion time, cost, reliability, and energy consumption.

Completion Time: Figure 6 shows that as task quantity increases, all algorithms require more time. While IPSO and IGA are improved intelligent algorithms, IFOA-GA consistently requires the least time, demonstrating its advantage.

[Figure 6: see original paper]

Cost: Figure 7 shows cost increases with task quantity for all algorithms. IFOA-GA shows larger fluctuations initially due to population initialization overhead, but stabilizes later and maintains overall cost advantage.

[Figure 7: see original paper]

Reliability: Figure 8 shows IFOA-GA outperforms others as task quantity increases, benefiting from FOA improvements and GA integration.

[Figure 8: see original paper]

Energy Consumption: Figure 9 shows all algorithms consume more energy with increasing tasks. IFOA-GA' s curve is stable with low fluctuation, reducing energy consumption by approximately 7.27%, 9.33%, and 15.49% compared to the other three algorithms.

[Figure 9: see original paper]

These results demonstrate IFOA-GA' s superiority through orthogonal array initialization, boundary handling, adaptive step size, and GA-based selection, improving overall performance and cloud computing task scheduling efficiency.

---

4 Conclusion

This paper addresses cloud computing task scheduling challenges by fusing FOA and GA to create the IFOA-GA algorithm, which achieves good results in cloud computing task scheduling. However, this study does not consider the impact of virtual machine load in task scheduling, which will be investigated in future work.

References

[1] Zhang Jianxun, Gu Zhimin, Zheng Chao. Survey of research progress on cloud computing [J]. Application Research of Computers, 2010, 27(2): 429-433.

[2] Santwana S, Saurabh B. Workflow Scheduling in Cloud Computing Environment Using Bat Algorithm [J]. Proceedings of First International Conference

on Smart System, Innovations and Computing, 2018. DOI: 10.1007/978-981-10-5828-8_{15}.

[3] Kaur K. A Novel Context and Load-Aware Family Genetic Algorithm Based Task Scheduling in Cloud Computing [J]. Data Engineering and Intelligent Computing, 2017. DOI: 10.1007/978-981-10-3223-3_{51}.

[4] Vinothina V. An Approach for Workflow Scheduling in Cloud Using ACO [J]. Big Data Analytics, 2017. DOI: 10.1007/978-981-10-6620-7_{50}.

[5] Shabeera T P. Optimizing VM allocation and data placement for data-intensive applications in cloud using ACO metaheuristic algorithm [J]. Engineering Science and Technology, an International Journal, 2017, 20(2): 712-724.

[6] Sadhasivam N. Cancer Diagnosis Epigenomics Scientific Workflow Scheduling in the Cloud Computing Environment Using an Improved PSO Algorithm [J]. Asian Pacific Journal of Cancer Prevention, 2018, 19(1): 243-246.

[7] Huang Weijian, Guo Fang. Multi-objective task scheduling based on fireworks algorithm in cloud computing [J]. Application Research of Computers, 2017, 34(6): 1718-1720.

[8] Qiao Liang, Lin Weiwei. Cloud Computing Task Scheduling Based on Improved Bat Algorithm [J]. Microelectronics & Computer, 2017, 34(7): 27-32.

[9] Zhao Junpu, Yin Jinyong, Jin Tongbiao. Application of genetic ant colony algorithm in cloud computing resource scheduling [J]. Computer Engineering and Design, 2017, 38(3): 693-697.

[10] Sa Rina. Cloud Computing Resource Scheduling Scheme Based on Ant Colony Particle Swarm Optimization Algorithm [J]. Journal of Jilin University: Science Edition, 2017, 55(6): 1518-1522.

[11] Fang Jinming. An improved Mapreduce model oriented to cloud computing [J]. Computer Measurement & Control, 2012, 20(5): 1417-1419.

[12] Wu Xiaowen, Li Qing. Research of optimizing performance of fruit fly optimization algorithm and five kinds of intelligent algorithm [J]. Fire Control & Command Control, 2013, 38(4): 17-20.

[13] Liu Liqun, Han Junying, Dai Yongqiang. Comparative study on optimization performance of fruit fly optimization algorithm [J]. Computer Technology and Development, 2015, 25(8): 94-98.

[14] Chang Tianqing, Bai Fang, Li Yong. Research on population initialization for swarm intelligence optimization solving WTA [J]. Application Research of Computers, 2013, 30(5): 1377-1380.

[15] Duan Yanming, Xiao Huihui. Research of the self-adaptive step fruit fly optimization algorithm [J]. Journal of Henan Normal University (Natural Science), 2016, 44(1): 161-168.

[16] Wang Xiaotian, Han Xiao. Cloud computing service deployment optimization based on improved particle swarm algorithm [J]. Journal of Jilin University: Science Edition, 2017, 55(2): 393-397.

[17] Li Chao, Dai Bingrong, Kuang Zhiguang. Research on task scheduling with multiple constraints based on genetic algorithm in cloud computing environment [J]. Mini-micro Systems, 2017, 38(9): 1945-1949.

Note: Figure translations are in progress. See original paper for figures.