# Optimized Replica Placement Strategies in Cloud Environments: Postprint

**Authors:** Wang Xin, Meng Yu, Qin Qin, Jiang Hua

**Date:** 2019-01-28T00:00:00+00:00

## Abstract

To improve the efficiency of data scheduling and replica access in cloud computing, this paper investigates the replica placement problem within replica strategies and proposes a replica placement strategy based on the ant colony algorithm. Drawing upon the principle of ant foraging in nature, the ant colony algorithm is applied to the entire replica placement process; by utilizing an improved ant colony algorithm featuring dynamic pheromone updates and Laplace probability distribution, a set of optimal solutions is derived for replica placement. Simulation experiments were conducted on the CloudSim platform, and experimental results demonstrate that the proposed scheme outperforms the original ant colony algorithm in terms of average job completion time, network utilization, and load balancing degree, while also reducing replica placement time consumption and network load to a certain extent.

## Full Text

### Research on Replica Optimal Placement Strategy in Cloud Environment

**Wang Xin[1],[2], Meng Yu[1], Qin Qin[2], Jiang Hua[1]** [1]College of Computer Science & Engineering, Guilin University of Electronic Technology, Guilin Guangxi 541004, China [2]School of Marine Information Engineering, Guilin University of Electronic Technology, Beihai Guangxi 536000, China

**Abstract:** To improve the efficiency of data scheduling and replica access in cloud computing, this paper investigates the replica placement problem within replica strategies and proposes a replica placement strategy based on ant colony optimization. Drawing from the principles of ant colony foraging in nature, the ant colony algorithm is applied throughout the entire replica placement process. By leveraging dynamic pheromone updating and an improved ant colony algorithm based on Laplace probability distribution, a set of optimal

solutions is obtained for replica placement. Simulation experiments conducted on the CloudSim platform demonstrate that the proposed scheme outperforms the original ant colony algorithm in terms of average job completion time, network utilization, and load balancing, while also reducing replica placement time consumption and network load to a certain extent.

**Key words:** cloud computing; ant colony algorithm; replica placement; load balancing

## 0 Introduction

With the development of the Internet, complex and massive amounts of data require processing. According to research reports from Internet data centers, future data growth will be geometric, particularly from mobile terminals. Consequently, the timely and effective processing of massive data has become an urgent problem to solve. The rise of cloud computing technology, especially the emergence of cloud storage technology, has enabled various types of storage devices to connect for information exchange and achieve collaborative cooperation. However, due to large data volumes and differences in storage and network devices, cloud computing environments can lead to data loss and errors. Replica technology effectively addresses these issues. Typically, multiple replicas are employed in storage systems to ensure high data availability and reliability.

Since system performance is closely related to node load, cloud storage exhibits locality characteristics for recently accessed data. If the system load is unbalanced, the data storage load on individual nodes will increase, significantly degrading user access performance to those nodes. Similarly, uneven distribution of hot data also affects user read and write performance. Therefore, how to reasonably place replicas in cloud storage systems is a problem worthy of investigation.

To effectively improve the performance of replica technology in cloud computing, scholars have conducted research on problems encountered in cloud replica strategies. Reference [1] addresses node heterogeneity in cloud storage systems by combining an improved particle swarm algorithm to enhance replica selection strategy, thereby improving replica selection efficiency and meeting user demands. Reference [2] investigates replica problems in cloud computing, optimizing replicas from the perspective of replica selection and integrating them with ant colony algorithms. It explores virtual machine utilization conditions and ultimately achieves optimized load balancing. Reference [3] considers data overhead and storage costs, proposing a replica strategy that balances cost and storage space. By utilizing genetic algorithms and Dijkstra' s algorithm for optimization, excellent results were achieved. Reference [4] establishes a relational model to maintain the minimum number of replicas for rational allocation on storage nodes, thereby improving data replica availability. Reference [5] addresses the problem of high replica access overhead in cloud storage systems by proposing the application of the firefly algorithm to the replica placement

process, establishing relevant models based on the natural attributes of fireflies. Simulation experiments demonstrate that this algorithm has good effects on replica placement. Reference [6] proposes a dynamic management model for cloud computing replicas in heterogeneous environments, which can effectively reduce data transmission between nodes. Reference [7] proposes a dynamic replica distribution method that effectively balances the relationship between replica overhead and access performance, optimizing replica distribution. Reference [8] proposes an effective greedy heuristic algorithm to solve the replica placement problem for improving QoS and operational overhead, optimizing computation time through layout and refinement, achieving good results. Reference [9] comprehensively considers factors such as statistical cycles, file sizes, and working environments, proposing a replica creation method based on heat analysis. According to file access heat, replicas are dynamically adjusted to improve service performance. Reference [10] proposes a dynamic data replication method in the cloud that uses directory indexes for replica location information, ultimately storing replicas in systems with sufficient space to increase resource availability and achieve minimum access cost.

Although the aforementioned literature has conducted relevant research on replica technology, there are still deficiencies in research on data scheduling and access efficiency in cloud computing. Moreover, in replica management technology, research on replica placement as a key issue is relatively scarce and requires further investigation. Based on the characteristics of replica placement and existing ant colony algorithms, this paper proposes a replica placement research strategy based on an improved ant colony algorithm. This algorithm utilizes dynamic ant colony pheromone updating and Laplace probability distribution to improve ant colony probability selection, ultimately forming the improved ant colony algorithm in this paper. By fully leveraging the distributed, stochastic, and feedback characteristics of ant colony algorithms and combining them with replica placement, this paper' s solution is formed. Through simulation using the cloud computing simulator CloudSim, the algorithm demonstrates good performance in terms of system average job completion time, network utilization, and load balancing.

## 1 Replica Placement Problem Description

This research focuses on the replica placement problem, while other issues in replica management are not fully encompassed herein. In cloud storage systems, after replicas are created, users need to reasonably place replicas to ensure data access performance. The system will select the optimal replica from a series of candidates for placement. Different systems have different criteria for replica placement. When multiple replicas meet system requirements and need placement, placement priority is given to replicas with higher file access frequency contributions by statistically analyzing access overheads for different files.

The file access frequency is defined as:

$$f_{file} = \frac{1}{T}\sum_{r=1}^{R} C_{f_r} \times f(t), \quad C_{f_r} \in R$$

where $R$ represents the total number of replicas in the entire system, $C_{f_r}$ represents the frequency magnitude of high-frequency files in replica $r$, and $f(t)$ represents the number of times the file is requested for access during the $t$-th time period.

The number of replicas that should be placed for a file is:

$$N_r = \left\lceil N \times \frac{f_{file}}{\sum_{r=1}^{R} f_r} \right\rceil, \quad N_r \in R$$

which is finally responded to the requesting user.

Replica placement involves two key steps: a) According to the retrieval file name given in the user request, the cloud storage system' s replica manager finds the corresponding location information of several replicas. b) On the basis of problem a), after finding the set of several replica information pieces, relevant replica placement strategies are utilized for further replica placement.

To improve storage resource utilization and data file reliability, aspects such as network access performance, load balancing, response delay, and storage overhead should be comprehensively considered to meet users' dynamic demands. For example, when placing replicas, consideration must be given to whether to place them locally or remotely. Therefore, the replica placement problem can be considered a process of comprehensive optimization based on multiple factors.

## 2 Ant Colony Optimization-Based Replica Placement Algorithm

The ant colony algorithm is derived from simulating the paths taken by ants foraging in nature. During the foraging process, ants leave pheromones for group communication, and the amount of pheromone determines how many ants will follow that path. Through this positive feedback mechanism, the ant colony gradually converges, ultimately finding the shortest path between the nest and the food source.

### 2.1 Core Description of Ant Colony Optimization Algorithm

This section introduces the basic 思路 (approach) of applying the ant colony algorithm to solve the replica placement problem:

a) The path taken by ants during foraging is abstracted as the selection process before target placement, and the path set is abstracted as a solution space.

b) During ant movement, pheromones are left behind. Pheromones become increasingly concentrated on shorter paths over time, and more ants will choose these paths.
c) The positive feedback effect of ant colony pheromones makes the ants' walking paths gradually unify, ultimately reaching the destination and completing target placement. This path becomes the optimal solution for target placement.

Based on the above 思路 (approach), the ant colony is abstracted as retrieval files (RF) that select and place replicas, while simultaneously creating and finding corresponding replicas for these files. The set of ant walking paths is abstracted as a replica object set $\{r_1, r_2, ..., r_n\} = R$, where $n$ is the number of ants. The pheromones generated during the foraging process are represented as the set $\{p_1, p_2, ..., p_n\} = P$. The entire ant colony is set as a feasible solution space, represented as $A$.

Before the entire process begins, this paper needs to initialize the pheromone values for replicas:

$$\tau_i(0) = \frac{\text{readspeed}}{\text{replicasize}}$$

where `replicasize` refers to replica size and `readspeed` refers to reading speed. As can be seen from the formula, the larger the reading speed and the smaller the replica, the larger the initial pheromone value.

After a replica is accessed multiple times, the pheromone will change accordingly, and the corresponding attribute values need adjustment. The expression is:

$$\tau_i(t+1) = \rho \cdot \tau_i(t) + \Delta\tau_i(t)$$

The setting of the pheromone evaporation coefficient $\rho$ 是否合理 (reasonableness) affects the ant colony' s search capability and computational efficiency. Building upon traditional pheromone updating methods, this paper introduces a method for dynamically changing the $\rho$ value, enabling $\rho$ to adaptively change its magnitude to ensure comprehensive algorithm performance. The adjustment strategy is given as follows:

$$\rho(t) = 1 - \frac{\ln(t)}{\ln(t+c)}$$

where $c$ is a constant. The value of $\rho$ is key to dynamic pheromone updating. Too large a $\rho$ will reduce global search capability, while too small a $\rho$ will deteriorate local search capability and slow down convergence speed. To make the evaporation coefficient $\rho$ adaptive within a certain range, $\rho$ is controlled within $[\rho_{\min}, \rho_{\max}]$ and takes values in $[0.2, 0.8]$. This can widen the gap in pheromones between replicas and accelerate convergence speed in later stages.

The concentration of pheromones left by the ant colony on paths is closely related to whether replicas can be placed. The higher the pheromone concentration, the greater the probability that the replica will be placed. The replica placement probability formula is defined as:

$$P_j(t) = \begin{cases} \frac{[\tau_j(t)]^\alpha [\eta_j]^\beta}{\sum_{u \in R} [\tau_u(t)]^\alpha [\eta_u]^\beta}, & j \in R \\ 0, & \text{otherwise} \end{cases}$$

where $\tau_j(t)$ refers to the pheromone concentration of replica $j$ at time $t$, and $\eta_j$ represents the inherent attribute of the replica itself. $\alpha$ and $\beta$ respectively represent the pheromone heuristic factor and expectation heuristic factor. If $\alpha$ is larger than $\beta$, it indicates that in node selection, the role of pheromone concentration is greater than the influence of replica inherent attributes. To avoid the ant colony algorithm falling into local optima prematurely and to ensure better comprehensive solution performance, through experimental verification, this paper sets $\alpha = 3$ and $\beta = 1.5$.

In traditional ant colony algorithms, replica placement probability selection uses random probability `rand` combined with Equation (7) for probability matching. Random probability selection strategies can avoid stagnation phenomena but converge slowly. To enable the algorithm to converge to the optimal solution more quickly, this paper introduces the concept of Laplace probability distribution based on experimental foundations, adopting a maximum probability selection method to improve the original ant colony algorithm. During the replica placement process, the object with the maximum transfer probability MaxP is first calculated, followed by cyclic traversal to calculate the distance between the replica placement object' s probability and the maximum probability MaxP. The replica closest to MaxP is selected for placement, yielding the formula:

$$P_i = [\text{MaxP} - \text{rand}, \text{MaxP} + \text{rand}], \quad i \in \{1, 2, ...\}$$

From Equation (7), it can be derived that the higher the pheromone concentration of a replica, the greater the probability of being selected for placement. However, when considering replica placement, node load and network smoothness are also important factors. This paper' s method of introducing a Laplace-like probability distribution to select the maximum probability replica for placement can effectively prevent network congestion and, to a certain extent, save storage resources.

### 2.2 Ant Colony Optimization Replica Placement Process

After receiving a replica file RF request, the cloud storage system utilizes the replica placement method proposed in this paper to sequentially place replicas with pheromones. The following is the process of using the ant colony algorithm for replica file placement:

a) Initialize replica pheromones.

b) Calculate the access frequency of replica file RF and the storage node location using Equation (1).
c) If the replica file is stored on a remote node, perform replica placement according to the improved ant colony algorithm.
d) Conduct probability matching for replica objects, and select appropriate replicas based on the selection probability according to Equations (7) and (8).
e) After selecting a replica, data is transmitted to the terminal. Before obtaining the replica, the terminal uses Equation (5) to reduce the replica information value, decreasing the probability of repeated access and balancing node load.
f) If the replica is stored locally, directly read the local replica without executing ant colony replica placement.
g) If the relevant data of the replica is not read successfully, jump to Step 2 to read the next replica object.
h) When the replica is read successfully, determine whether it is the last replica object. If yes, end replica placement; if not, read the next replica object again and perform improved ant colony replica placement.

The algorithm flowchart is shown in Figure 1 [Figure 1: see original paper].

## 3 Experimental Design and Results Analysis

This experiment uses the CloudSim cloud computing simulator for experimental simulation. CloudSim supports resource simulation in cloud computing environments and can simulate relevant environments in replica management. Utilizing CloudSim' s extensibility, the Host and DataCenter classes are modified, and new replica placement components are added to complete replica placement work, comparing replica access time overhead, network utilization, and load balancing degree. This experiment simulates the improved ant colony algorithm, the original ant colony algorithm, and the Min-Min algorithm.

The experimental environment is: software using Windows 10 operating system with CloudSim 3.0.3, hardware using i5-7300HQ@2.5GHz with 8GB memory.

CloudSim simulation experiment steps: a) Initialize the CloudSim package. b) Create a data center. c) Create a data broker center. d) Create cloud tasks. e) Read replicas and perform replica object placement. f) Start simulation. g) Analyze results after simulation ends.

The parameters designed for the algorithm in this experiment are shown in Table 1 .

**Table 1 Basic Data of Ant Colony Algorithms** | Parameter | Value Range | |——-|———-| | Expectation Heuristic Factor | 1.5 | | Pheromone Volatility Factor | 0.2~0.8 | | Number of Ants m | 100~800 | | Maximum Iterations n | - |

This experiment is divided into three groups, evaluating the performance of the improved ant colony algorithm, the original ant colony algorithm, and the

Min-Min algorithm from three aspects: average job completion time, network utilization ratio, and load balancing degree. The experimental results are compared to form this simulation experiment.

### 1) Average Job Completion Time

To verify the advantage of the improved ant colony algorithm in job completion time, this paper selects comparison with the original ant colony algorithm and the Min-Min algorithm. To accurately compare the three different algorithms, average job completion time is introduced:

$$T = \frac{1}{n}\sum_{i=1}^{n} T_i$$

where $T_i$ is the running time of the $i$-th job, and $n$ is the total number of completed jobs. Average job time refers to the total time for executing all jobs divided by the total number of completed jobs.

From the experimental data analysis in Figures 2 and 3 [Figure 2: see original paper][Figure 3: see original paper], the improved ant colony algorithm has shorter job completion time than both the original ant colony algorithm and the Min-Min algorithm. Because the Min-Min algorithm always assigns jobs to resources with the shortest processing time, it causes some resources to remain continuously working while others remain idle, ensuring jobs can be completed in the shortest time but resulting in unreasonable job allocation. The ant colony algorithm's global search capability gives it an advantage in job completion time. After optimizing pheromone and probability selection, the improved ant colony algorithm shows significant improvement in job completion efficiency compared to the original ant colony algorithm. As shown in the figures, the improved ant colony algorithm reduces average job completion time by 14%~25% compared to the original ant colony algorithm and by 27%~40% compared to the Min-Min algorithm.

### 2) Network Utilization Ratio

To describe the time consumption and network bandwidth occupation required for replica updates during replica placement, network utilization ratio is defined to represent the degree of network utilization by different algorithms:

$$E = \frac{N_{\text{remote\_file}} - N_{\text{file\_replicas}}}{N_{\text{remote\_file}} - N_{\text{local\_file}}} + N_R$$

where $N_{\text{remote\_file}}$ refers to the number of replicas related to job operation placement, $N_{\text{file\_replicas}}$ refers to the number of times computing units access replicas from remote nodes, and $N_{\text{local\_file}}$ refers to the number of times replicas are accessed from local nodes.

The smaller the $E$ value, the smaller the network utilization ratio of the algorithm, the less network resources occupied by the algorithm, the smaller the load on the network, and correspondingly, the better the algorithm performance.

As seen in Figure 5 [Figure 5: see original paper], although the network utilization ratio values of the improved ant colony algorithm fluctuate significantly under different job numbers, overall, the improved ant colony algorithm performs better in network utilization compared to the original ant colony algorithm, improving by about 0.2 percentage points.

From Figure 4 [Figure 4: see original paper], it can be seen that the improved ant colony algorithm differs greatly from the Min-Min algorithm. According to the previous definition and Equation (10), the improved ant colony algorithm performs more excellently in network load, improving by about 8.5 percentage points compared to the Min-Min algorithm.

### 3) Load Balancing Degree

During the experiment, load balancing degrees were statistics under different job numbers. Through calculation and analysis, the load balancing situation of different algorithms as job numbers increase can be obtained. The load balancing degree $LB$ formula is:

$$LB = \frac{1}{m}\sum_{j=1}^{m}\left(c_j - \frac{1}{m}\sum_{j=1}^{m}c_j\right)^2$$

As shown in Figure 6 [Figure 6: see original paper], the improved ant colony algorithm has significant advantages over the Min-Min algorithm due to real-time adjustments made by system resource nodes. Compared to the original ant colony algorithm, the improved ant colony algorithm performs slightly better.

## 4 Conclusion

Aiming at the replica placement problem in multi-replica management in cloud environments, this paper proposes an improved ant colony optimization-based replica placement algorithm. Through dynamic pheromone updating and improved probability selection methods in the ant colony algorithm, ants can better select paths, ultimately placing replicas in optimal positions. The impact of replica placement performance on scheduling efficiency and access performance is comprehensively considered, and simulation experiments are conducted in the cloud computing simulator CloudSim. Three groups of experiments verify the improved algorithm. Experimental results show that compared with the original ant colony algorithm and Min-Min algorithm, the improved ant colony algorithm demonstrates excellent performance in average job completion time, can effectively reduce time consumption in the replica placement process, and can better balance network load. Future work will focus on other areas of replica strategies such as replica creation and replica location.

## References

[1] Zhang Cuiping, Guo Zhenzhou, Gong Changqing. Study on strategy of replica selection in cloud storage environment [J]. Computer Science, 2015, 42 (S2): 408-412.

[2] Zuo Fang, He Xin. Pheromone-based ant colony replica selection mechanism in cloud storage [J]. Application Research of Computers, 2015, 32 (11): 3368-3370.

[3] Wu Xiuguo. Minimum-cost based data replication strategy in cloud computing environment [J]. Computer Science, 2014, 41 (10): 154-159.

[4] Zhu Jiayu, Xiao Dan. Dynamic replication management scheme for cloud computing [J]. Computer Engineering & Design, 2012, 33 (9): 3362-3365.

[5] Li Jun, Hou Mengshu. Replica placement strategy based on glowworm swarm optimization [J]. Application Research of Computer, 2018, 36 (3): 314-316.

[6] Tao Yongcai, Zhang Ningning, Shi Lei, et al. Research on dynamic management of data replicas of cloud computing in hetergeneous environments [J]. Journal of Chinese Computer Systems, 2013, 34 (7): 1487-1492.

[7] Sun Xin, Li Qingzhou, Zhao Pu, et al. An optimize replica distribution method for peer-to-peer network [J]. Chinese Journal of Computers, 2014, 37 (6): 1424-1434.

[8] Sahoo J, Glitho R. Greedy heuristic for replica server placement in cloud based content delivery networks [C]//Proc of IEEE Symposium on Computers and Communication. Washington DC: IEEE Computer Society, 2016: 302-309.

[9] Rao Lei, Yang Fande, Li XinMing, Liu Dong, et al. Dynamic replica creation algorithm based on tempera-ture' s analysis [J]. Journal of Computer Applications, 2014, 34 (S2): 130-134.

[10] Rajalakshmi A, Vijayakumar D, Srinivasagan K G. An improved dynamic data replica selection and placement in cloud [C]//Proc of International Conference on Recent Trends in Information Technology.

[11] Liu Tiantian, Li Chao, Hu Qing, et al. Multiple replicas management in the cloud environment [J]. Journal of Computer Research and Development, 2011, 48 (S3): 254-260.

[12] Huang Dongmei, Du Yanling, et al. Marine monitering data replica layout strategy based on multiple attribute optimization [J]. Computer Science, 2018, 45 (6): 72-75,104.

[13] Singh A, Thapar S, Bhatia A, et al. Disk scheduling using a customized discrete firefly algorithm [J]. Cogent Engineering, 2015, 2 (1).

[14] Zhao Qiuyun. Replica Selection strategy based on similar scene recommendation in data grid [J]. Microelectronics & Computer, 2012, 29 (9): 23-26+30.

[15] Long S Q, Zhao Y L, Chen W. MORM: a multi-objective optimized replication management strategy for cloud storage cluster [J]. Journal of Systems Architecture, 2014, 60 (2): 234-244.

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