A DYNAMIC LEVEL SCHEDULING ALGORITHM BASED ON AHP

Jintao JIAO[*1,2] ,Wensen YU[1,3],Lei GUO[1,3]

Batch scheduling strategies mainly focus on computing resources allocation in the existing cloud computing environment. Considering the heterogeneity of computing resources and the risk undertaken by computing resources, this paper proposes a dynamic level scheduling algorithm based on AHP (DLSAHP). From the perspective of cloud computing resources providers, the algorithm uses scale 1-9 to fully consider the risk undertaken by computing resources. The experiment results show that DLSAHP can effectively improve the rate of successful tasks execution within deadline, at the same time the profit in one unit of time is also increased.

Keywords: AHP, Batch Scheduling, Cloud Computing, DLS

1. Introduction

According to the difference of task scheduling time, task scheduling strategy can be divided into two kinds: batch scheduling and online scheduling. In batch scheduling model, the pending tasks will be collected into a set and won't be processed until an appointed time. The collected tasks will be processed together. After the appearance of cloud computing scheduling market model, many researchers start to focus on user's demand of service quality. Cloud computing scheduling market model [1-4] could manage and allocate computing resources more effectively. The model brings four benefits: (1) user's fair use of computing resources, (2) adjusting the balance of supply and demand of computing resources in cloud computing. When demand exceeds supply, the higher price of computing resources could help to reduce the number of users and tasks, on the other hand, when supply exceeds demand, the lower price could help to attract more users, (3)providing quality of service to users, such as task deadline, cost to complete the task, security of computing resources, (4)providing the effective computing resources management and allocation mechanism.

Kavitha compares the performance of five algorithms based on QoS in three aspects: user satisfaction, task completion period and meta-task utility [5].

[1] Prof., School of Mathematics and Computer Science, Wuyi University, China. E-mail: jiaojintao@163.com
[2] School of Information Management, Jiangxi University of Finance and Economics, China.
[3] Prof., The Key Laboratory of Cognitive Computing and Intelligent Information Processing of Fujian Education Institutions, China

The result indicates that the performance of each algorithm is different when different service quality is required, so different task scheduling algorithm should be adopted according to the specific application environment. Vanmechelen proposes an economic management approach to solve the problem of CPU binding task based on the deadline [6]. Sundaram discusses the relationship between task scheduling throughput and fairness under deadline constraints [7]. If the scheduling throughput is increased, it will bring unfair scheduling to some tasks. On the other hand, if fair scheduling is the focus, it will reduce scheduling throughput. So, the literature adjusts the balance between scheduling throughput and fair scheduling by setting some simple parameters.

1.1 Cloud Computing Scheduling Market Model

Cloud computing scheduling market model could manage and evaluate resources allocation more effectively. The model contains users, broker, resources providers and cloud information service, the architecture shows in Fig. 1.

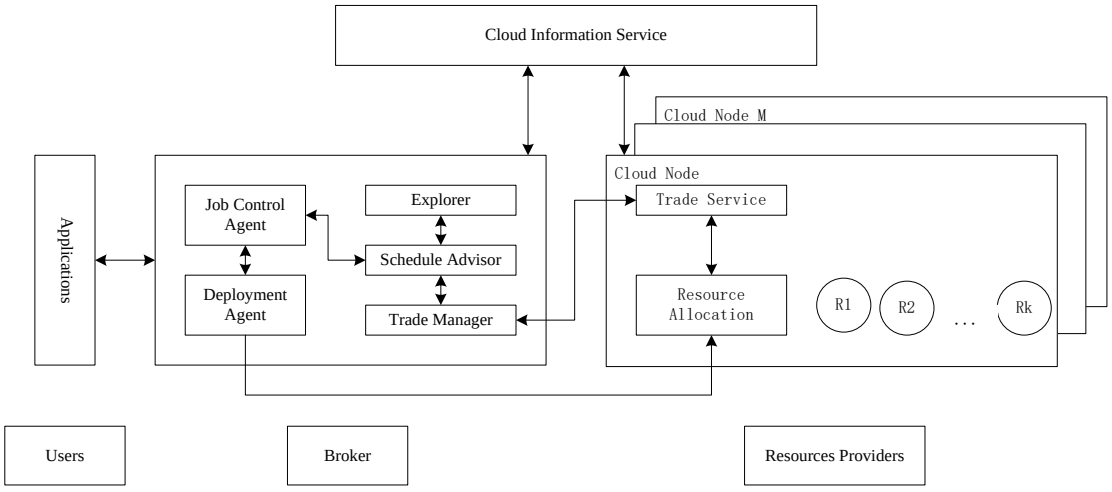

Fig. 1. Cloud Computing Scheduling Model

The model allows user to set the resource requirements and preferences for parameters, and the user pays for the using resources.

Broker is the intermediate interface between user and resource, function of which is to discover resource, select resource, receive tasks, return scheduling results and exchange information. And broker supports different scheduling policies which can find resource and schedule tasks according to user's demands. Broker is mainly composed of job control agent, schedule advisor, explorer, trade manager and deployment agent. Information service mainly records available resources. Broker will query information service when searches for appropriate resources, then interact with resources providers after getting information of

resources that meet user's demand. If resources providers have new resource to lease, they must register the resources information in information service so that broker could find it.

In the process of transaction between users and resources providers, resources providers register the resources information in the information service. After user submits a task to broker, broker will get available resources information from information service, and then schedules the task to the appropriate resources according to the scheduling algorithm. Broker also estimates the completion time and the cost before the task is executed. If the time exceeds the deadline or the cost is higher than user's budget, broker will refuse to accept the task. If the task is executed successfully, broker will return results to the user and obtain the profit, otherwise return the error information.

1.2 Elements in the Scheduling Algorithm

There are many factors in the scheduling algorithm. User's demand and service fee are closely related to elements such as deadline, reparation duty and profit. Deadline is the time baseline of task scheduling. Each task has its own deadline. The purpose of service provider is to complete as many as possible tasks before the deadline and maximize the profit. In order to take full account of the risk undertaken by computing resources, reparation duty and profit are introduced to reflect the compensation and the possible profits.

1.3 AHP and scale 1-9

AHP is also known as the analytic hierarchy process [8]. With reference to the configured relative priority scales, AHP compares elements in pairs to obtain result. We can use AHP to calculate the user's task elements and obtain the weight parameters [9-11]. Since the parameters contain the information of reparation rate and profit, computing resources scheduling could be more reliable and effective. The specific scheduling algorithm is shown in section 4.

For the comparison between task elements, the proposed scale 1-9 can be used to solve the problem. The scale 1-9 is shown in Table 1.

Table1

Scale 1-9

Intensity of Importance	Definition	Explanation
1	Equal Importance	Two activities contribute equally to the objective
2	Weak or slight	
3	Moderate importance	Experience and judgement slightly favor one activity over another
4	Moderate plus	
5	Strong importance	Experience and judgement strongly favor one activity over another
6	Strong plus	
7	Very strong or demonstrated	An activity is favored very strongly over another;

	importance	its dominance demonstrated in practice
8	Very, very strong	
9	Extreme importance	The evidence favoring one activity over another is of the highest possible order of affirmation

2. Calculation of Risk Weight

We could use AHP and scale 1-9 to calculate the risk weight of tasks. The main steps are shown as follows.

Algorithm 1: AHP

Step 1: build the hierarchy diagram. The hierarchy diagram contains three layers: top layer, rule layer with m elements and solution layer with n elements;

Step 2: build an m×m comparison matrix of rule layer elements; get the eigenvector denoted by X corresponding to the maximum eigenvalue of the matrix;

Step 3: build m n×n comparison matrices of solution layer elements associated with m rule layer elements; get the eigenvectors corresponding to the maximum eigenvalue of the m matrices;

Step 4: calculate the weighted sum of m eigenvectors of solution layer elements by using the element values in eigenvector X as weight factor;

Step 5: return the weighted sum as result;

Step 6: end;

In order to get relevant weights of the task, some presetting is made (as shown in Fig. 1): The target layer is Goal, which is the optimal scheduling task. The rule layer contains three elements: deadline, reparation duty and profit. Deadline is used to reflect the user's needs. Reparation duty and profit reflect the risk assumed by computing resources. The solution layer contains all computing resources. In summary, we form a hierarchy diagram as shown in Fig. 2.

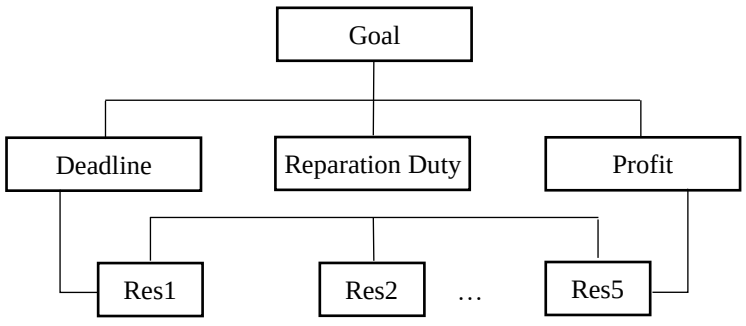

Fig. 2. Hierarchy Diagram

The weight of the task is calculated based on the hierarchy diagram. For example, for a given task, the importance ratio of deadline, reparation duty and profit are set to 3:1:1 by cloud computing user according to scale 1-9. This means

that deadline is the most important element, followed by reparation duty and profit with the same importance. The pairwise comparison matrix of the task is built as shown in Formula 1.

$$T = \begin{bmatrix} 1 & 3 & 3 \\ 1/3 & 1 & 1 \\ 1/3 & 1 & 1 \end{bmatrix} \tag{1}$$

The eigenvector corresponding to the maximum eigenvalue of the matrix is (0.6, 0.2, 0.2), which is the relative weight of deadline, reparation duty and profit to the task.

Table2

Pairwise Comparison Matrix Associated with The Target

	Deadline	Reparation Duty	Profit	Priorities
Deadline	1	3	3	0.6
Reparation Duty	1/3	1	1	0.2
Profit	1/3	1	1	0.2

Suppose there are only 5 resource nodes, the pairwise comparison matrices with deadline, reparation duty and profit are given by the cloud computing broker. The results are derived from the perspective of the user who submit the task. When deadline is used as the reference element, the weight of these resources nodes is shown in Formula 2. When reparation duty is used as the reference element, the weight of these resources nodes is shown in Formula 3. And when profit is used as the reference element, the weight of these resources nodes is shown in Formula 4. The pairwise comparison matrix of these resources is as follows.

$$R_D = \begin{bmatrix} 1 & 1/2 & 2 & 3 & 3 \\ 2 & 1 & 4 & 5 & 5 \\ 1/2 & 1/4 & 1 & 1 & 1 \\ 1/3 & 1/5 & 1 & 1 & 1 \\ 1/3 & 1/5 & 1 & 1 & 1 \end{bmatrix} \tag{2}$$

The eigenvector of matrix R_D is (0.248, 0.460, 0.106, 0.093, 0.093).

Table 3

Pairwise Comparison Matrix Associated With Deadline

	Resource1	Resource2	Resource3	Resource4	Resource5	Priorities
Resource1	1	1/2	2	3	3	0.248
Resource2	2	1	4	5	5	0.460
Resource3	1/2	1/4	1	1	1	0.106
Resource4	1/3	1/5	1	1	1	0.093
Resource5	1/3	1/5	1	1	1	0.093

$$R_R = \begin{bmatrix} 1 & 1/3 & 3 & 2 & 2 \\ 3 & 1 & 8 & 5 & 5 \\ 1/3 & 1/8 & 1 & 1 & 1 \\ 1/2 & 1/5 & 1 & 1 & 1 \\ 1/2 & 1/5 & 1 & 1 & 1 \end{bmatrix} \quad (3)$$

The eigenvector of matrix R_R is (0.196, 0.536, 0.079, 0.094, 0.094).

Table 4

Pairwise Comparison Matrix Associated With Reparation Duty

	Resource1	Resource2	Resource3	Resource4	Resource5	Priorities
Resource1	1	1/3	3	2	2	0.196
Resource2	3	1	8	5	5	0.536
Resource3	1/3	1/8	1	1	1	0.079
Resource4	1/2	1/5	1	1	1	0.094
Resource5	1/2	1/5	1	1	1	0.094

$$R_P = \begin{bmatrix} 1 & 1/3 & 3 & 2 & 3 \\ 3 & 1 & 9 & 5 & 5 \\ 1/3 & 1/9 & 1 & 1 & 1 \\ 1/2 & 1/5 & 1 & 1 & 1 \\ 1/3 & 1/5 & 1 & 1 & 1 \end{bmatrix} \quad (4)$$

The eigenvector of matrix R_P is (0.208, 0.539, 0.075, 0.092, 0.085).

Table 5

Pairwise Comparison Matrix Associated With Profit

	Resource1	Resource2	Resource3	Resource4	Resource5	Priorities
Resource1	1	1/3	3	2	3	0.208
Resource2	3	1	9	5	5	0.539
Resource3	1/3	1/9	1	1	1	0.075
Resource4	1/2	1/5	1	1	1	0.092
Resource5	1/3	1/5	1	1	1	0.085

By combining the above matrix T and R, we can get the result as shown in Table 6. For the given task, Table 6 shows that the risk weights of these resource nodes are: (0.230, 0.491, 0.094, 0.093, 0.092).

Table 6

Result by Combining Matrix T And R

	Deadline	Reparation Duty	Profit	Priorities
Resource1	0.6	0.2	0.2	0.230
Resource2	0.6	0.2	0.2	0.491
Resource3	0.6	0.2	0.2	0.094
Resource4	0.6	0.2	0.2	0.093
Resource5	0.6	0.2	0.2	0.092

3. Consistency Check

If a pairwise comparison matrix A satisfies Formula 5, then A is a consistent matrix.

$$a_{ik} * a_{kj} = a_{ij} \quad i, j, k = 1, 2 \cdots\cdots n \tag{5}$$

Since deadline, reparation duty and profit are the reference element of the pairwise comparison matrix, and the level of each computing resource is evaluated from a subjective view, the pairwise comparison matrix may not be a consistent matrix, such as Formula 2. But we still accept such inconsistency within a certain allowable range.

Theorem: If A is a pairwise comparison matrix, then the maximum eigenvalue $\lambda \geq n$. And when $\lambda = n$, A is a consistent matrix [7]. Proofs are detailed in the reference article written by Saaty.

It can be known from Theorem 1 that the greater the difference is between λ and n, the more inconsistent A will be, and the greater the judgment error will be caused by using the eigenvector as the weight vector. Therefore, the degree of inconsistency of A can be measured by value $\lambda - n$. Saaty defines the consistency indicator as shown in Formula 6. When *CI*=0, A is a consistent matrix. The bigger *CI* is, the more inconsistent A will be.

$$CI = (\lambda - n) / (n - 1) \tag{6}$$

Saaty introduces the random consistency indicator *RI* to find the criteria for the consistency indicator *CI*. For different n (n is an integer between 1 and 11), Value of *RI* calculated from 100,500 samples are shown in Table 7.

Table 7

Value of Random Consistency Indicator RI

n	1	2	3	4	5	6	7	8	9	10	11
RI	0	0	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49	1.51

In Table 7, where n = 1, 2, *RI* is 0 because the 1st or 2nd order pairwise comparison matrixes are always consistent matrix.

For the pairwise comparison matrix A with n≥3, Saaty defines the consistency ratio as shown in Formula 7.

$$CR = CI / RI \tag{7}$$

If *CR* < 0.1, the inconsistency degree of the paired comparison matrix A is considered to be within the allowable range.

For any pairwise comparison matrix A, if A is a consistent matrix, then we accept the weights calculated from A. However, if A is not a consistent matrix, then the consistency check should be performed on A. Only when *CR*<0.1, we accept the weights calculated from A; if *CR*≥0.1, A is inconsistent and should be re-adjusted until it passes the consistency check.

The consistency check values of *T*, R_D, R_R, R_P are shown in Table 8. The result indicates that all the matrices we built pass the consistency check.

Table 8

Value of maximum eigenvalue λ and CR

	T	R_D	R_R	R_P
λ	3	5.01984	5.03415	5.05616
CR	0	0.00443	0.00762	0.01254

4. Dynamic Level Scheduling Algorithm Based on AHP(DLSAHP)

DLS (dynamic level scheduling algorithm) is a fast and efficient algorithm. After calculation of DLS, if the preliminary task N_i and the idle computing resource P_j match a higher dynamic level than any other tasks and computing resources, then task N_i is scheduled onto the computing resource P_j. The dynamic level of tasks and computing resources can be obtained from DLS, and the definition of DLS is shown in Formula 8[12].

$$\begin{aligned} &DLS(N_i, P_j) = \\ &SL(N_i) - max(TDA(N_i, P_j), TRF(P_j)) + \Delta(N_i, P_j) \end{aligned} \tag{8}$$

Algorithm 2: DLS

Step 1: $S \leftarrow$ input the set of all tasks;

Step 2: form the directed graph *G* with *S* according to the constraints of each task;

Step 3: obtain the value $SL(N_i)$ for task N_i;

Step 4: compute and obtain the value of $TDA(N_i, P_j)$ and $TRF(P_j)$ for each computing resource P_j, then get $max(TDA(N_i, P_j), TRF(P_j))$;

Step 5: get the value of $TA(N_i)$ and $T(N_i, P_j)$, then compute $\Delta(N_i, P_j) = TA(N_i) - T(N_i, P_j)$;

Step 6: set $\begin{aligned} &DLS(N_i, P_j) = \\ &SL(N_i) - max(TDA(N_i, P_j), TRF(P_j)) + \Delta(N_i, P_j) \end{aligned}$;

Step 7: if there are unscheduled tasks, goto Step 3;

Step 8: end;

All tasks in the task set *S* form a directed graph *G*={*N*,*A*} due to the execution constraints, *N* is a set of computation nodes (tasks),{ N_i } i=1…n, with known execution times, where each node is executed exactly once in each

invocation of the scheduling program. *A* is the set of directed arcs $\{A_{ij}\}$ between nodes which define a partial order or precedence constraint (<) on N such that arc A_{ij} directed from one node N_i into node N_j implies that N_i must precede N_j ($N_i < N_j$) in execution. Each arc A_{ij} also carries label D_{ij} which specifies the amount of data that N_i passes to N_j on each invocation. $SL(N_i)$ denotes the static level of N_i, which indicates the longest directed path from N_i to an end node of the graph, and it also means the importance of the task N_i under execution constraints. $max(TDA(N_i, P_j), TRF(P_j))$ represents the earliest time when the computing resource P_j starts the execution of task N_i. $TDA(N_i, P_j)$ is the time when input data of task N_i is available after the task N_i is scheduled onto the resource P_j. $TRF(P_j)$ indicates the time when the idle resource P_j is free and can be used to execute the task N_i. $\Delta(N_i, P_j) = TA(N_i) - T(N_i, P_j)$. $TA(N_i)$ is the average time cost for task N_i to be executed on each idle resource. $T(N_i, P_j)$ is the time cost for task N_i to be executed on resource P_j. So $\Delta(N_i, P_j) = TA(N_i) - T(N_i, P_j)$ reflects the execution speed of each computing resource.

In order to improve DLS algorithm, the proposed algorithm DLSAHP combines AHP with DSL algorithm and fully consider the heterogeneity of computing resources and the risk undertaken by computing resources in cloud computing environment. When the task is scheduled onto the target computing resource, the trust degree reflects the risk undertaken by the target computing resource, which also reflects the degree of reparation duty and profit. We define DLSAHP (dynamic level scheduling algorithm based on AHP) as shown in Formula 9.

$$\begin{aligned} &DLSAHP(N_i, P_j) = \\ &SL(N_i) - max(TDA(N_i, P_j), TRF(P_j)) + n * w(N_i, P_j) * \Delta(N_i, P_j) \end{aligned} \tag{9}$$

meaning of $SL(N_i), max(TDA(N_i, P_j), TRF(P_j)), \Delta(N_i, P_j)$ is the same in Formula 8. $w(N_i, P_j)$ is calculated by the broker with AHP method, and it means the weight of the computing resource P_j relative to the task N_i. $w(N_i, P_j)$ reflects the risk (or trust) undertaken by the computing resource P_j. n is the number of computing resources. n is introduced because the weight calculated by

the AHP method may be too small to fully reflect the risk undertaken by the computing resource. Therefore, n is used to reasonably amplify the weight of risk.

5. Simulation Experiment and Result Analysis

5.1 Experimental Methods and Results

The resources parameters are shown in Table 9. The processing capability indicates the task processing capability weight of the resource. The price per unit refers to the unit charging weight when the resource executes the task. The deviation indicates the proportion of the risk that the resource undertakes when processing the task. The risk includes the possibility of speed reduction or failing to complete the task as expected and etc. Percentage represents the proportion of each type of resource.

Table 9

Resource Types and Proportion

Process Capability	Price Per Unit	Deviation	Percentage
0-3	1-2	25%-30%	50%
3-6	4-5	15%-20%	30%
6-9	8-9	5%-10%	20%

Totally there are total seven task types. The deadline, reparation duty, profit and percentage of the tasks are set up as shown in Table 10. Percentage is the proportion of each type of tasks.

Table 10

Task Weights and Proportion

Task Type	Deadline	Reparation Duty	Profit	Percentage
Type 1	0.6	0.2	0.2	15%
Type 2	0.45	0.45	0.1	10%
Type 3	0.45	0.1	0.45	10%
Type 4	0.2	0.6	0.2	15%
Type 5	0.2	0.2	0.6	15%
Type 6	0.1	0.45	0.45	10%
Type 7	0.33	0.33	0.33	25%

In order to complete the DLSAHP experiment, we introduce five properties of the task including deadline, reward, decay, bottom line and penalty. The parameter deadline, reward and penalty are used to compute the AHP weight factors, while the parameter decay and bottom line are used to get the value of penalty.

(1) Deadline: When the task is completed before this time, the computing resource provider does not need to pay reparation.

(2) Reward: The fee paid by the user to the computing resource provider when the task is completed before the deadline. So, reward is the profit of the service provider when the task is completed on schedule.

(3) Decay: When the computing resource provider does not complete the task at the deadline, the computing resource provide shall pay reparation for each one unit of time.

(4) Bottomline: If the computing resource provider does not complete the task at the deadline, the reparation will not increase when bottom line is reached. Bottom line>deadline.

(5) Penalty: The reparation paid by the computing resource provider when bottom line is reached. At this point, the task may have been partially completed, or it may not be executed at all. The decay and bottom line must be considered in order to compute the penalty. So penalty is the reparation duty that the user could get from the service provider when the task is not completed on schedule.

For example, the resource 1 has a small deviation, and the task can be completed within the solid line, while the deviation of the resource 2 is large, and the task can be completed within the illustrated dotted line. According to the basic information of the two resources (as shown in Table 10), the user can get the weight of the deadline, the reparation duty and the profit according to AHP method (See Table 3 to 5). Combined with the task's weight (as shown in Table 2), the final risk undertaken by the two resources can be get (as shown in Table 6).

The computing resource 1 and 2 are exemplified as shown in Fig. 3.

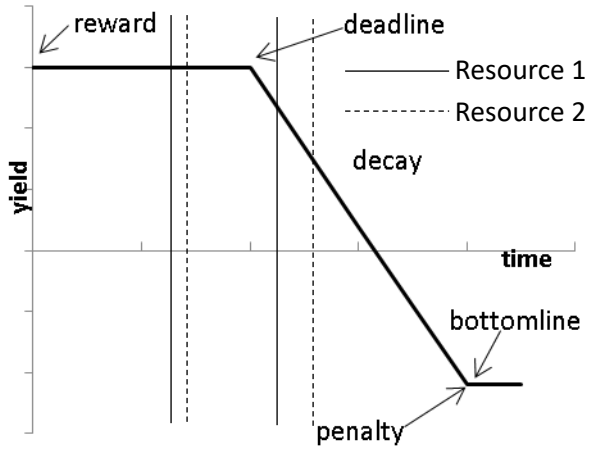

Fig. 3. State of the Task

Two experiments are designed in order to test DSL and the proposed DLSAHP algorithm. The simulation platform is Cloudsim [13-14]. Cloudsim is an event driven cloud computing simulation toolkit based on Java; its main goal is to research in the effective resource allocation method based on the computing market model. The comparison experiment results are shown in Fig. 4 and Fig. 5.

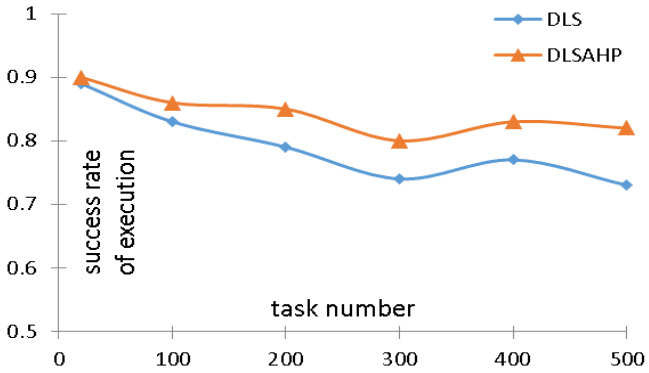

Fig. 4. Average Success Rate of Tasks Execution

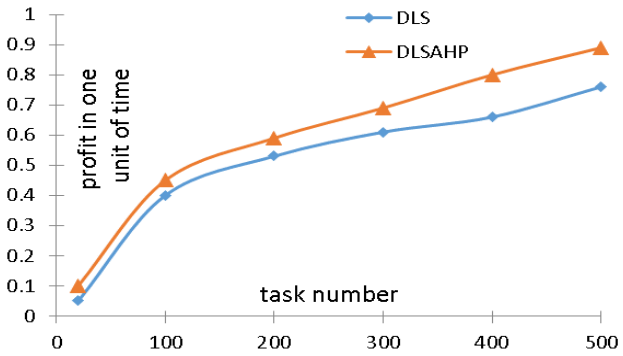

Fig. 5. Profit in One Unit of Time

As the number of tasks increases, the average success rate of tasks execution of DLSAHP is slightly higher than that of DLS. We could see the success rate from Fig. 4. Because scale 1-9 is used to calculate the degree of the risk undertaken by computing resource, the number of tasks completed beyond deadline is effectively reduced. Since DLSAHP calculates the risk undertaken by the computing resource, the risk can be effectively reduced. It can be seen from Fig. 5 that the profit in one unit of time of DLSAHP is higher than that of DLS.

5.2. Algorithm Performance Analysis

Table 11 show that the time and space costs of the algorithm are increased when the number of tasks is increased.

Table 11

Time and Space Cost of Different Number of Tasks

Number of Tasks	100	200	300	400	500
Time Cost(ms)	5	11	17	23	29
Space Cost(KBytes)	202	437	683	895	1106

The time cost of the algorithm is in the millisecond level, which is far less than the execution time of the task, so the time cost of the scheduling algorithm can be ignored. The space cost is increased linearly, so the number of tasks should be limited.

6. Conclusions

For the first time this paper introduces AHP algorithm in order to improve DLS algorithm. DLSAHP is proposed, which fully considers the scheduling factors such as priority, deadline, profit, risk and so on. From the perspective of cloud computing resource provider, DLSAHP effectively improves the number of tasks completed, and increases the profit of the cloud computing resource provider.

Acknowledgements

This work was supported by the Central Committee Guides Local Science and Technology Development Projects (2018L3013) and Fund of Fujian Education Institution (JT180561).

R E F E R E N C E S

[1] *B. Ravi, D. Sanjukta, G. Robert, J. Stallaert*, A market design for grid computing. in INFORMS Journal on Computing, **vol. 20**, no. 1, Feb. 2018, pp. 100-111

[2] *G. Stuer, K. Vanmechelen, J. Broeckhove*, A commodity market algorithm for pricing substitutable grid resources, in Future Generation Computer Systems, **vol. 23**, no. 5, Jun. 2016, pp. 688-701

[3] *R. Buyya, M. Murshe, D. Abramson*, A deadline and budget constrained cost-time optimization algorithm for scheduling task farming applications on global grids, in ICPDP, Las Vegas, Nevada, USA, 2012.

[4] *H. Song, SB. Yang, XQ. Liu*, A quality-driven algorithm for task scheduling in grid market, Journal of the Graduate School of the Chinese Academy of Sciences, **vol.28**, no.1, Jan. 2018, pp.86-93

[5] *S. Kavitha, Golconda, F. Ozguner*, A Comparison of static QoS-based scheduling heuristics for a meta-task with multiple QoS dimensions in heterogeneous computing, in IPDPS, Washington DC, USA, 2014.

[6] *K. Vanmechelen, W. Depoorter, J. Broeckhove*, Economic grid resource management for CPU Bound Applications with hard deadlines, in CCGrid, Lyon, France, 2018.

[7] *V. Sundaram, A. Chandra, J. Weissman*, Exploring the throughput-fairness tradeoff of deadline scheduling in heterogeneous computing environments, In ICMMCS, Annapolis, Maryland, USA, 2017.

[8] *TL. Saaty*, Decision making with the analytic hierarchy process, in International Journal of Services Sciences, **vol. 1**, no. 1, Jan. 2008, pp. 83-98

[9] *Suvendu Chandan Nayak, Chitaranjan Tripathy*, Deadline sensitive lease scheduling in cloud computing environment using AHP. Journal of King Saud University - Computer and Information Sciences, **vol. 301**, no. 2, Apr. 2018, pp. 152-163

[10] *SH. Mian, A. Al-Ahmari*, Comparative analysis of different digitization systems and selection of best alternative, Journal of Intelligent Manufacturing, **vol.30**, no. 5, Jun. 2019, pp. 2039-2067

[11] *SE. Falahat, H. Hadi, SS. Moghaddam, A. Aryanfar*, Exploring cultivation site of saffron (Crocus sativus L.) by utilizing GIS linked to AHP, Spatial Information Research, **vol. 27**, no. 3, Jun. 2019, pp. 285-293

[12] *Gilbert C. Sih, Edward A. Lee*, Dynamic-level scheduling for heterogeneous processor networks, in SPDP, Washington DC, USA,1990.

[13] *N. Rodrigo*, CloudSim: A Novel Framework for the Modeling and Simulation of Cloud Computing Infrastructures and Services, The University of Melbourne Australia, 2009.

[14] *RN. Calheiros, R Ranjan, A Beloglazov, CAF De Rose, R Buyya*, CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software Practice and Experience, **vol. 41**, no.1, Aug. Jan. 2011, pp. 23-50