An Ant Colony Optimization for Grid Task Scheduling with Multiple QoS Dimensions

Jing Hu, Mingchu Li, Weifeng Sun*, Yuanfang Chen
School of Software
Dalian University of Technology
Dalian Liaoning, China
cicihj_0795@gmail.com, mingchuli@dlut.edu.cn, wfsun@dlut.edu.cn*, 27596731@qq.com

***Abstract*—Task scheduling and quality of service (QoS) are two curial problems in grid computing. Focusing on the meta-task with QoS requirements, this work presents an ant colony optimization for grid task scheduling with multiple QoS dimensions (QACO). The proposed algorithm considers five kinds of QoS dimensions: time, reliability, version, security and priority which are transformed to utility as the heuristic information of the algorithm. The objective of the algorithm is maximizing the total utility. Simulation studies compare the performance of QACO, QoS-Min-Min and the improved Min-Min. Simulation results shown that QACO finds the best results.**

Keywords- multiple QoS dimensions; ant colony optimization; task scheduling; utility

I. INTRODUCTION

Grid computing enables users to assemble large-scale geographically distributed computational resources to create a secure virtual supercomputer cooperatively to accomplish a specific computational goal [1]. The research about task scheduling is the core in the design and implementation of the grid resource management which is the important component of the core middleware in the grid computing. The high-efficient scheduling strategy could optimize process capability of grid and sequentially improve the performance of applications; therefore, it is crucial to study the task scheduling in grid computing. From the definition of the grid, we can see "to deliver nontrivial qualities of service (QoS)" is one of the guide line of estimating grid performance, thus grid task scheduling strategy should consider the QoS requirements of users. It is required to have the best task scheduling strategy and guarantee the QoS as well.

A volume of mature task scheduling algorithms have been proposed, such as Min-Min， Max-Min, XSufferage[2-3], which are the simple efficient heuristics. But they do not consider the QoS requirements of users. After that, many QoS guided task scheduling algorithms have been proposed. He etc al. [4] have proposed QoS-Min-Min by improving traditional Min-Min using bandwidth as the heuristic information. The work only considers bandwidth which would affect task complete time，so it can obtain good performance. Weng etc al [5] have proposed QoS-Sufferage by improving Sufferage using average response time as the heuristic information. The two algorithms only consider the situation of one QoS dimension, but it can not satisfy the users' demands of multiple QoS dimensions. Chen etc al. [6] have proposed the grid resource scheduling algorithm senior integrating the thought of forecast mechanism with dual constraints of deadline and bandwidth. The algorithm can obtain a better accomplishment ratio according to forecasting executive time of grid tasks.In the heterogeneous computing environments, many QoS-based scheduling of task with multiple QoS demands have been proposed. Tracy D.Braun etc al. [7] have presented static resource allocation algorithms for heterogeneous computing environments with tasks having dependencies, priorities, deadlines, and multiple versions. To aim at task scheduling model with multiple QoS dimensions, the work adapts the static techniques from some previous studies and applies to the model: GA, GENITOR-style algorithm, and a two phase greedy technique based on the concept of Min-Min heuristics. The experimental showed that the GENITOR technique finds the best results, and the faster two phase greedy approach also performs very well. Braun etc al. [8] have carried on a comparative study of five heuristics, QSMTS-IP, Min-Min, Genetic Algorithm, Least Slack First and Sufferage. The heuristics have been modified from their original implementations to incorporate additional QoS attributes. The performances of the five heuristics are compared in terms of number of satisfied users (tasks), Makespan and sum of utilities of tasks. It has shown that GA and Min-Min produced better results.

Task scheduling in grid computing is proved to be a NP-hard problem, and intelligence optimization algorithms (such as GA, PSO) suit to solve it. Ant colony optimization (ACO) [9] is also one of intelligence optimization algorithms. ACO has the advantages of robustness, positive and negative feedback mechanism, avoiding premature and so on. Moreover, a good task scheduling algorithm would adjust its scheduling strategy according to the changing status of the entire environment and the types of tasks. Therefore a dynamic algorithm in task scheduling such as ACO is appropriate for grid [10]. A number of grid task scheduling algorithms based on ACO have been

Supported by Nature Science Foundation of China under grant No.: 60673046，90715037，University Doctor Subject Fund of Education ministry of China under grant No.: 200801410028，National 973 Plan of China under grant No.: 2007CB714205 and Natural Science Foundation Project of Chongqing, CSTC under grant No.: 2007BA2024.

*Corresponding author.

978-0-7695-3766-5/09 $25.00 © 2009 IEEE
DOI 10.1109/GCC.2009.57

proposed, and they obtained good results [10-11]. But the algorithms do not take QoS into account to satisfy the requirements of users.

On the basis of the studies, we apply ACO further to solve the problem of task scheduling with multiple QoS dimensions in grid computing. We can improve the performance of task scheduling at the same time guaranteeing the quality of service. QoS can be divided into metrics and policies in the large-scale distributed computing system [12]. QoS of Metrics contains time-related parameters (such as deadline) and veracity-related parameters (such as precision) which are used to define performance, security needs, and the relativity of tasks and so on. We define the QoS model of metrics considering the five kinds of QoS attributes: timeliness, reliability, security, version, and priority. We modify the QoS model in heterogeneous computing environment proposed in [7-8] to use it in grid computing. The proposed algorithm aims to maximize the users' utilities. From papers [7] and [8], we can see the modified Min-Min could produce good results. We also modify the Min-Min with our QoS model, namely improved Min-Min. Then we compare the proposed QACO (Ant Colony Optimization for Grid Task Scheduling of multiple QoS dimensions) algorithm with QoS-Min-Min and improved Min-Min. According to the simulation results, it can be seen that QACO is capable of achieving the objective of scheduling better than the other two algorithms.

The rest of the paper is organized as follows. Section □describes the details of the task scheduling model with multiple QoS dimensions. We introduce the proposed algorithm detailed in section □. In Section □ the results from the simulations are examined. Section □ concludes this paper.

II. THE TASK SCHEDULING MODEL WITH MULTIPLE QOS DIMENSIONS

Task scheduling mode is divided into on-line mode and batch mode. In the batch mode, tasks are not mapped onto the machines as they arrive; instead they are collected into a set that is examined for mapping at prescheduled times called mapping events. The independent set of tasks that is considered for mapping at the mapping events is called a meta-task [2]. Our work is based on batch mode and the meta-task with multiple QoS dimensions. The QoS requirements are transformed to utility as users' degree of satisfaction. From the user perspective, each user expects his utility maximum, but scheduler should maximize all users' utilities instead of single user. Therefore, maximizing all users' utilities (total utility) becomes our objective.

A. Prolem Formulation

The tasks in grid computing may have amount of QoS requirements that need to be satisfied. We list five considered types.

1) *Timeliness:* Timeliness includes the start time, complete time, deadline, transmit delay time and so on. In this paper, only the deadline is considered. Each task may be assigned a receivable deadline.

2) *Reliability:* It's possible that the machines in grid would disable. The reliability of a machine is defined to be the failure rate of tasks executing on it.

3) *Versions:* The resources in the grid are not dedicated machines; hence their attributes are changing continuously. A task may exist in different versions. We consider various versions impact the task execution time and user-defined preference.

4) *Security:* Each user may require different levels of security services including authenticity, confidentiality, integrity, etc for their task and data. Each machine is assigned a level of security.

5) *Priority:* When several tasks compete for the exiguous resources, scheduler should satisfy the QoS requirements of higher priority. Each task may be assigned the priority which presents the importance of the task. Priority can be specified by user and scheduler.

Although we only consider these QoS requirements, our definition and solution could be applied to more types of QoS. Assume the execution time of tasks have been forecasted using the method proposed by Shoukat Ali [13]. Let R= $\{r_1, r_2, \ldots r_m\}$ denotes the heterogeneous machines in the grid, and T= $\{t_1, t_2, \ldots, t_n\}$ denotes the set of *n* independent tasks.

$ET=\begin{Bmatrix} et_{11} & \cdots & et_{1n} \\ \vdots & & \vdots \\ et_{m1} & \cdots & et_{mn} \end{Bmatrix}$, defines the expect execution time for tasks, where et_{ij} denotes the execution time of task t_i on machine r_j. Let $M(i)$ denotes the machine assigned to task t_i, where $1 \le i \le n$. And S_j is a function of scheduling which denotes the order of task t_i on machine r_j, where $S_j = \{S_j(i) \mid 1 \le i \le n, 1 \le j \le m\}$. $V(i)$ is defined as the version of task t_i, where $1 \le i \le n$. Let each task t_i be associated with d_i number of QoS dimensions. In addition, let Q_i^j be either a definite or infinite set of QoS choices for the *j*th QoS dimension of task t_i, where $1 \le j \le d_i$ and $q_i^j \in Q_i^j$ denote a QoS choice for the *j*th QoS dimension of task t_i. Thus, $Q_i = \{Q_i^1, Q_i^2, \ldots, Q_i^{d_i}\}$ defines a d_i dimensional space of the QoS choices for task t_i and a point in the space is given by $q_i = \{q_i^1, q_i^2, \ldots, q_i^{d_i}\}$.

B. Evaluation

The evaluation standard of the original task scheduling is Makespan, that is, the time is given by the start time subtracted from the task complete time. Taking into account QoS requirements we concern users' QoS requirements more than the time of tasks executing. Therefore, we evaluate the objective of scheduling by total utility. The utility is

transformed by QoS requirements through utility functions. The utility function of task t_i is defined as,

$$\mathrm{U_i}(q_i) = \sum_{j=1}^{d_i} w_i^j U_i^j(q_i^j) \times p_i \qquad (1)$$

where, $U_i^j(q_i^j)$ is the utility for the *j*th QoS dimension q_i^j of task t_i, w_i^j is weight assigned to the *j*th QoS dimension, p_i is the priority f task t_i. Maximize the total utility $\sum_{i=1}^{n} U_i(q_i)$ is the objective of our algorithm, where, *n* is the number of tasks.

III. THE PROPOSED QACO ALGORITHM

We find the best mapping of tasks and machines applying ant colony optimization (ACO). We assume the scheduler is the ant, and the process of scheduling is the process of ants searching for food. We modify the global pheromone update function in ant colony with total utility, and calculate heuristic function with complete time to make the complete time less. When ants select the next task, they will incline to choose the one with high pheromone density (maximum total utility and little complete time). Total utility is firstly considered, and then complete time is considered.

The details of QACO are shown in Fig.1, where η_j denotes heuristic function, *Tabu* denotes taboo list， U_i denotes the utility of task t_i, *ANT* denotes the set of ants, T denotes the set of task, *NC-max* is the maximum number of cycles, τ_{ij} is the pheromone of task t_i and task t_j.

In step 7, ant ant_k will move from task t_i to task t_j with probability

$$\begin{cases} p_{ij}^k(t) = \dfrac{[\tau_{ij}(t)]^\alpha [\eta_j(t)]^\beta}{\sum_{s \in A} [\tau_{is}(t)]^\alpha [\eta_s(t)]^\beta}, j \in A \\ p_{ij}^k(t) = 0 \qquad\qquad otherwize \end{cases} \qquad (2)$$

In this work, maximizing the total utility is our objective, thus we update the pheromone with the utility $\sum_{i=1}^{n} U_i(q_i)$. And heuristic function is defined as $\eta_j = \dfrac{1}{et_j}$, where et_j denotes the execution time of task t_j. α is a parameter to control the influence of pheromone, and β is a parameter to control the influence of the heuristic function. A is the set of tasks which have not been scheduled.

In step 13, the global pheromone update function is defined as

$$\tau_{ij}(t+n) = (1-\rho)\tau_{ij}(t) + \rho \Delta \tau_{ij}(t) \qquad (3)$$

1: Initialize α, β, et, ρ, Q, *NC-max*
2: while NC < *NC-max* do
3: initialize *Tabu*
4: for all tasks in meta-task *T*, $t_i \in T$
5: for all ants in *ANT*, $ant_k \in ANT$
6: Compute U_i, η_j
7: Compute P_{ij}^k
8: select the next task t_i; assign the machine $M(i)$ giving the task maximum utility to the task t_i
9: update *Tabu*, add the task t_i into *Tabu*
10: end for
11: end for
12: set the best utility *U_best* of the cycle as the jumping-off of the next cycle
13: update the pheromone τ_{ij}
14: end while
15: compute *U*, *T*
16: end

Fig.1 the proposed algorithm QACO

where, ρ is the rate of pheromone evaporation. $\rho \Delta \tau_{ij}(t)$ is the amount of pheromone deposited, given by

$$\Delta \tau_{ij} = \sum_{k=1}^{m} \Delta \tau_{ij}^k \qquad (4)$$

$$\Delta \tau_{ij}^k = Q U_k \qquad (5)$$

where *Q* is a constant, and U_k is the total utility.

IV. SIMULATION AND PERFORMANCE EVALUATION

Braun etc al. [8] have been carried on a comparative study of five heuristics, QSMTS-IP, Min-Min, Genetic Algorithm, Least Slack First and Sufferage and found the Min-Min produces better results. Hence, through comparing with the improved Min-Min, we can evaluate the performance of the proposed algorithm adequately. We improved the original Min-Min which is applied to our task scheduling model. The improved Min-Min is modified from original Min-Min to replace the minimum complete time by maximum utility. The details of the improved Min-Min are shown in Fig.2, where *T* is the set of tasks, *M* is the set of machines.

For performance evaluation of our work, a series of experiments are performed, and the other two algorithms are evaluated together with the proposed algorithm QACO: QoS-Min-Min[6], the improved Min-Min.

A. Simulation Enviroment

In the simulation studies, a grid with 10 machines was considered and the number of tasks ranged from 20 to 120.

1: for each task $t_i \in T$ (in an arbitrary order)
2: for each machines $m_j \in M$ (in a fixed arbitrary order)
3: compute each utility

$$U_{ij} = \sum_{k=1}^{d_i} w_i^k U_{ij}^k(q_i^k) \times p_i$$

4: endfor
5: endfor
6: repeat
7: for each task in T find the maximum utility and the machine that obtains it
8: find the task t_k with the maximum utility
9: assign task t_k to the machine m_l that gives the maximum utility
10: remove task t_k from T
11: update U_{ij} for all tasks
12: until all tasks in T are mapped

Fig.2 the improves Min-Min algorithm

Each task t_i is associated with five QoS dimensions:

1) *Timeliness:* Each task t_i is assigned the value of deadline $d_i = (\delta_i \times et_i) + a_i$, where $\delta_i \in \{1, 2, 3, 4\}$ (each value is equally to be assigned), and a_i is the task arrival time. A deadline achievement function D_i is also defined. According to the mapping, D_i =1 if task t_i completes before d_i, otherwise D_i =0.

2) *Security:* Each machine m_j is assigned a security level from (poor, low, medium, or high) in random.

3) *Reliability:* The failure rate of a machine is assumed to be uniformly distributed between 0.0005 and 0.0015 failures per unit time.

4) *Version:* We assume the simulation environment has *M*=10 machines and *V*=3 versions. Execution times of version V_{k+1}, $ET(V_{k+1}) = random * ET(V_k)$, where *random* is randomly selected between 50% and 90%. Let r_{ik} be the user-defined preference for task t_i of version V_k. And r_{i0} =1(most preferred), $r_{i1} = r_{i0} \times UR(0,1)$, where $UR(0,1)$ is a function which generates a uniformly distributed random number between 0 and 1. Users maybe consider the task version of low preference, because it has less execution time which can be the only version which can complete before the deadline.

5) *Priority:* Each task is randomly assigned a priority chosen from（1，2，3）, and 3 represents the highest priority level.

These parameters are based on previous research [7-8], experience in the field. And the expected execution time of version 0 of tasks on machines *ET* is generated using the method proposed by Shoukat Ali etc al. [13], where $\mu_{task} = \mu_{mach}$ =100, and $V_{tash} = V_{mach}$ =0.5.And the other parameters about ACO are defined as popular values shown in TABLE Ⅰ.

TABLE I. THE VALUE OF PARAMETERS IN QACO

Parameter	values
ant	1.5t (t is the number of tasks)
NC_max	200
α	1.5
β	1
ρ	0.1
Q	10

B. *Performance Evaluation*

The performances of the three algorithms are compared in terms of Makespan and total utility $\sum_{i=1}^{n} U_i(q_i)$. And the total utility is the chief performance parameter. The one who has the best utility could offer the best service to the system. We have done a large number of experiments, and select the representative results obtained on the scale total utility, Makespan and number of tasks. The results are shown in Fig.3 and Fig.4.

We can see from Fig.3 that the total utility of QACO finds the best results. QoS-Min-Min only considers the QoS requirement related with time instead of utility; hence it produces the minimum utility. QoS-Min-Min is unfit for the situation of multiple QoS dimensions. The improved Min-Min guided by utility is better than QoS-Min-Min but not as good as QACO. QACO improves increasingly with the positive and negative feedback information of utility until the total utility is the maximum.

Fig.4 shows that QoS-Min-Min provides the minimum Makespan in the most case, since it optimizes the complete time specially, however, it is not obvious. QoS-Min-Min is

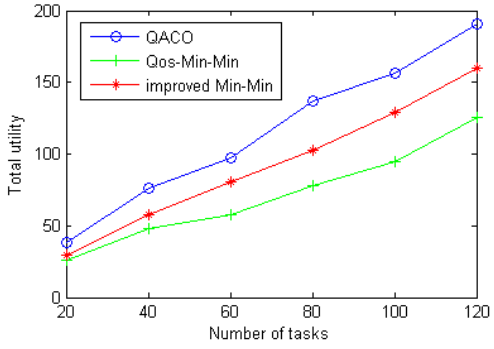

Fig. 3 the comparison of total utility

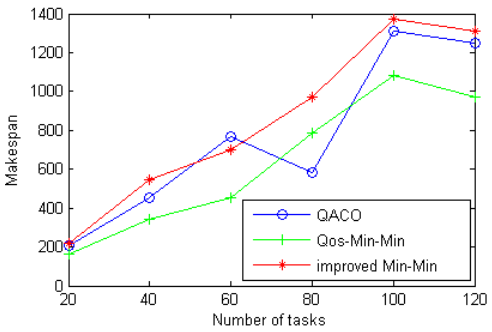

Fig.4 the comparison of Makespan

worse than QACO once in a while, because QACO takes the probability into account as the time factor, and it finds the less complete time guaranteeing utility maximum. The Makespan of QACO fluctuates slightly between that of the other two algorithms; because it depends the complete time of the case with maximum utility. While the Makespan of QACO is also acceptable.

In a word, although the Makespan of QACO is not always the best, it reaches the aim of scheduling in term of the total utility. Therefore, QACO can solve the problem of task scheduling with multiple QoS dimensions effectively.

V. CONCLUTION

Our contribution is applying ACO in grid task scheduling with QoS guaranteed. Because of the complexity of grid task scheduling and the importance of QoS, an ant colony optimization for grid task scheduling of multiple QoS dimensions (QACO) is presented. QACO use the utility as the heuristic information to finish task scheduling effectively, and it can maximize user's utilities at the same time. Simulations are done to compare the QACO with QoS-Min-Min and improved Min-Min in terms of two performance parameters, Makespan and total utility. The simulation rerults show that QACO performs higher utility than the other two algorithms all the time. Although QACO is not as good as QoS-Min-Min in Makespan, the gap couldn't affect the performance of grid task scheduling.

In future work, we will study more situations where we don't consider in our task scheduling model, and will improve the performance of QACO.

REFERENCES

[1] Foster. What is the grid? A three point checklist, Grid Today, vol. 1, p. 6, 2002.

[2] Muthucumaru M, Shoukat A, Howard JS et al., "Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems", In: Proc. of the 8th Heterogeneous Computing Workshop(HCW'99), 1999.

[3] Henri C, Arnaud L, Dmitrii Z, et al., "Heuristics for Scheduling Paramete Sweep Applications in Grid Environments", In: Proc. of the 9th Heterogeneous Computing Workshop (HCW'2000), 2000.

[4] He XS, Sun XH, von Laszewski G, "QoS guided min-min heuristic for grid task scheduling", Journal of Computer Science and Technology, vol. 18, pp. 442-451, 2003.

[5] C Weng , X Lu, "Heuristic scheduling for bag-of-tasks applications in combination with QoS in the computational grid", Future Generation Computer Systems, vol. 21, pp. 271-280, 2005.

[6] Chen J, Kong L, Pan X, "Research on Grid Resource Scheduling Algorithm Integrating Forecast Mechanism with QoS Constraint", Journal of Computer Research and Development, vol. 45, pp. 11-16, 2008.

[7] TD. Braun, H. J. Siegel, and A. Maciejewski, "Static resource allocation for heterogeneous computing environments with tasks having dependencies, priorities, deadlines, and multiple versions", Parallel Distributed Computing, pp. 1504–1516, 2008.

[8] K Golconda, F Ozguner, A Dogan1, "A comparison of static QoS-based scheduling heuristics for a meta-task with multiple QoS dimensions in Heterogeneous Computing", In : Proc of the Int' l Parallel and Distributed Processing Symposium. Los Alamedas, CA: IEEE Computer Society Press, 2004.

[9] M.Dorigo, V.Maniezzo, and A.Colorni, "Positive feedback as a search strategy", Technical Report, Dipartimento di Elettronica, Politecnico di Milano, IT, pp. 91-106, 1991.

[10] Ruay-Shiung C, Jih-Sheng Ch, Po-Sheng L, "An ant algorithm for balanced job scheduling in grids", Future Generation Computer Systems, vol. 25, pp. 20-27, 2009.

[11] Zhihong X, Xiangdan H, Jizhou S, "Ant Algorithm-based task scheduling in grid computing", Proc of 2003-Canadian Conf on Electrical and Computer Engineering, vol. 2, pp. 1107-1110, May 2003.

[12] Chatterjee BSS, Sydir MDJJ, Lawrence TF, Taxonomy for QoS specifications, In: Proc. of the 3rd Int'l Workshop on Object-Oriented Real-Time Dependable Systems (WORDS'97), Newreport Beach, pp. 100-107.

[13] S. Ali, H. J. Siegel, M. Maheswaran, and D. Hengsen, "Task Execution Time Modeling for Heterogeneous Computing" , In IPDPS Workshop on Heterogeneous Computing, , pp. 185–199, May 2000.