A Dynamic and Efficient Grid Task Scheduling Strategy

Deng Jianguang, Zhao Yuelong
School of Computer Science & Engineering
South China University of Technology
Guangzhou,510006, China

Deng Jianguang, Yuan Huaqiang
Engineering & Technology Institute
Dongguan University of Technology
Dongguan,523808,China

***Abstract*-A dynamic and efficient grid task scheduling strategy was proposed in this paper by combining the genetic algorithm and the ant algorithm. The proposed method integrated the global search capability of the genetic algorithm and the solution precision of the ant algorithm; moreover, it avoided the imprecise local solution, prematurity and degradation phenomena of genetic scheduler, and overcame the inefficiency of the ant algorithm at its initial search stage. The simulation results show that the proposed scheduling strategy has an obvious superiority of scheduling efficiency in the large-scale grid task scheduling environment, and is better than the genetic algorithm and the ant algorithm as a whole.**

Keywords-Grid task, Scheduling strategy, Chromosome, Pheromone

I INTRODUCTION

Nowadays, with the development of IT technology and the increase of the internet bandwidth, the grid computing technology is emerged [1], which broke through the restriction of a certain individual resource's computing capability, and the geographical constraint of computing node.

As a novel computing mode, the grid had many particular features, such as the dynamic distribution, the heterogeneity and the diversity of its computing nodes. Moreover, every grid node obeyed the independent management of its owner and the unified deployment of the whole grid system synchronously. The above characteristics made the grid task scheduling strategy more complicated than the traditional high-performance systems. In fact, the grid scheduling was a NP-hard issue, and it attracted many researchers into the study of this topic.

Based on the assigning time of grid task, grid scheduling algorithms can be classified into two categories, namely, the static scheduling [2] and the dynamic scheduling [3]. The static scheduling algorithms that are applied widely include the OLB[4], the MCT, the MET, the Min-min, the Max-min, the Duplex [5], the GA(genetic algorithm) [6], the AA(ant algorithm) [7], the SA[8], the GSA [9], the Tabu algorithm, and the heuristic A* algorithm, and so on.

It is noted that, the general static algorithm can also be used as the dynamic method after some adjustments. Every one of the above static methods had its shortcomings, and there existed the similar problems respectively in the above static methods when using as the dynamic method. The studies on the GA and the AA showed that, both of them had the advantages of parallelism and robustness. Nevertheless, the GA did not make best of its feedback information, which made its final local optimization poor and easily leaded to the phenomena of prematurity and degradation. On the other hand, the AA had the property of positive feedback, and the high solution precision; however, it was inefficient at the initial searching stage due to lack of pheromone. Based on the above considerations, we proposed a novel grid task scheduling strategy by combining the GA and the AA, in which, a fast global search was executed at first based on the GA, then we transformed the global search information obtained at the end of the GA into the initialization of the pheromone of the AA, and finally, we achieved the suitable task scheduling optimization results based on the AA.

II THE CORRELATIVE INTERPRETATION

In order to describe the proposed scheduling method conveniently, some correlative symbols and interpretations were given in this paragraph.

Let the sets P={p1, p2, …, pn} and T={t1, t2, …, tm} denote the computing resource node set and the task set of grid system, respectively, and the symbols n and m denoted the number of computing nodes and that of tasks respectively. In general, m was much greater than n. The grid task scheduling problem was described as a five-tuple, ∑: (T, ＜, Q, C, X), where T was the task set; "＜" represented the task priority, and ta＜tb meant that the task ta must be completed before the scheduling of tb; Q was a m×n-dimensional computing cost matrix, the element qi,k of which denoted the computing cost of task ti on computing node pk; C was a m×m-dimensional communication cost matrix, the element ci,k, of which denoted the communication cost from task ti to tk; X was a m×n-dimensional task assignation matrix, and if the task ti was assigned to the computing node pk, the element xi,k =1, else xi,k =0.

III THE FAST GLOBAL SEARCH BASED ON THE GA

A. The chromosome encoding

In order to execute the global search in the GA, the grid task scheduling scheme should be encoded into chromosome. We adapted the binary method to encode chromosome, in

which, a chromosome was corresponding to a binary string, which represented a task scheduling scheme. On the assumption that m=8, n=3, and given the task assignation matrix X as below, the corresponding chromosome encoding scheme was as follows.

$$X=\begin{bmatrix}0&1&0\\1&0&0\\0&1&0\\0&0&1\\0&0&1\\1&0&0\\0&0&1\\0&0&1\end{bmatrix}\xrightarrow{the\ \ binary}\begin{bmatrix}0&1\\0&0\\0&1\\1&0\\1&0\\0&0\\1&0\\1&0\end{bmatrix}\xrightarrow{chromosome\ \ encoding}(0100011010\quad 001010)$$

In the above task assignation matrix X, the task sets {t2, t6}, {t1, t3}, {t4, t5, t7, t8} were assigned to the node p1, p2, and p3, respectively. We applied the binary strings 00, 01 and 10 to denote that these corresponding grid tasks were assigned to the first, second and the third computing node respectively. Finally, we listed the binary task assignation matrix in a row and got a chromosome coding.

B. The objective function

The grid scheduling was a complicated issue, in which the computing cost of task, the communication cost among tasks, and the difference between the computing cost and the communication cost must be taken into consideration. In this paper, the scheduling length (SL) of grid tasks was defined as the objective function COST of the proposed scheduling method, which was measured according to the time-consumption, and so it was not necessary to consider the difference between the computing cost and the communication cost.

C. The fitness function

The fitness function was used to evaluate the quality of chromosome. The greater its value was, the stronger the viability of chromosome was, the better its corresponding solution was, and the more opportunity to be maintained in the iterative evolution the chromosome was. The fitness function adopted in this paper was Fitness=(ETotal-SL)/ETotal, where SL was the scheduling length and ETotal was the total scheduling cost of all tasks. It was obvious that ETotal was a constant, and so the smaller the SL was, the greater the corresponding value of the fitness function was, and the better the corresponding scheduling scheme.

D. The genetic evolution

Encoding all kind of task scheduling schemes in the above chromosome encoding method, we obtained a random chromosome colony, on which then the genetic evolution operation was implemented using the genetic operators including crossover, mutation, and coping.

Crossover was a dominant approach to generate the new chromosome. Selected a certain proportion of chromosomes randomly from the current colony, exchanged some specific loca of two chromosomes in a certain way, and we could complete the crossover reproduction of the current chromosome colony.

Selected some chromosomes from the current colony, changed some loca of these chosen chromosomes, and we could achieve the mutation operation and generate some new chromosomes. Mutation corresponded to the local solution capability of the GA, and so it was needed for the GA to implement mutation operation on a very small number of chromosomes.

The optimal individuals of the current chromosome colony were copied directly to the next generation without any change in the copy operation, in which, the chromosome individual with a fitness value greater than a given threshold was maintained in the next generation.

After the above operation, we obtained a new generation of chromosome. Then an evaluation about the new chromosome colony was implemented based on the objective function. If the new chromosome colony was obviously evolutionary comparing to the last generation of chromosomes, repeat the above steps of genetic operation; otherwise, end the iterative evolution.

IV THE PRECISE SOLUTION BASED ON THE AA

At the global search stage based on the GA, the global search terminated when the evolution rate of the current chromosome colony was less than a given threshold. Then we initialized the pheromone of the AA and started the AA.

A. The initialization of the AA

Let the number of tasks m denote the number of all ants, n denote the number of the computing nodes. bi(t) and τi(t) represented the number of ants and the density of pheromone at the time t on the node pi respectively. Initially, τi(t)=ri-loadi(0), where ri denoted the inherent computing capability of the node pi, loadi(0) denoted the computing load of the node pi, which was assigned based on the current optimal scheduling scheme obtained at the end of the GA.

B. The detailed description of the AA

The detailed description of the adjusted AA was given as follows.

Step1: Assign the m ants to the corresponding computing nodes based on the current optimal scheduling scheme obtained at the end of the GA, and initialize the pheromone of every node, τi(0)=ri-loadi(0), i=1,2, …, n;

Step2: Record the node on which every ant was placed in their respective solution set tabu; /* the solution set tabu is used to recorded the nodes passed through by the corresponding ant */

Step3: N=0;

Step4: Check the current task implementation status and evaluate a corresponding pheromone increment for every computing node. If a task was completed successfully on the node pi, the pheromone increment Δτi=Ce×K is evaluated for this node; otherwise, if the task was implemented unsuccessfully on this node, the corresponding pheromone increment is set as Δτi=Cp×K;

/* K denotes the implementation cost of the corresponding task, and Ce, Cp denote the corresponding incentive factors. */

Step5: Update the pheromone of all nodes: $\tau i(t)=\tau i(t)+\Delta\tau i$, i=1, 2, …, n;

Step6: Check whether there were other computing nodes joining or leaving the grid system. For the newly joined nodes, initialize the pheromone of them based on their computing capability. On the other hand, set the pheromone value of the leaving nodes as zero;

Step7: Based on the pheromone distribution of every computing node, calculate the probability:

$$p_j^k(t)=\begin{cases}\dfrac{[\tau_j(t)]^\alpha\times[\eta_j]^\beta}{\sum_i[\tau_i(t)]^\alpha\times[\eta_i]^\beta} & Task\ \ i,j\ \ is\ \ available\\ 0 & else\end{cases}$$

, and choose the next node pj for every ant based on the calculated maximum probability;

/* $\tau i(t)$ denotes the pheromone of the node pi at the time t; ηi denotes the inherent computing capability of the node pi; α and β denote the importance of pheromone and the inherent computing capability of the node, respectively. */

Step8: Record the node pi in the solution set tabu of the corresponding ant;

Step9: Based on the chosen nodes of all ants, calculate the scheduling length SL of the corresponding scheduling scheme, namely, the objective function COST, and record the current optimal task scheduling scheme;

Step10: If the iteration counter N is equal to the given maximum threshold or the current evolution rate is less than the given minimum threshold, it means that the current record is the calculated optimal scheduling scheme, then:

{ Output the current optimal task scheduling scheme; break; }

Else

{N=N+1; Revise the pheromone of all nodes, $\tau inew=(1-\rho)\times\tau iold-K$, i=1,2, …, n;

/* ρ denotes the volatilization rate of pheromone; K denotes the implementation cost of the corresponding task. */

Clear the solution set tabu of every ant and return Step4; }

V THE EXPERIMENT TEST AND PERFORMANCE ANALYSIS

Based on the principle of objectivity, some simulation experiments were made in this paragraph to test the proposed task scheduling strategy by adopting the toolbar GridSim, which was applied widely in the simulation of grid system, and compared its testing results with that of the GA and the AA, respectively. The three scheduling methods were all tested based on the system of window 7 (CPU: Intel(R) Pentium (R) P6000*2, 1.86GHz; Memory: 2.0GB; HD: 250GB).

A. The parameter evaluation

Base on the respective characteristic of the GA and the AA, we set all the parameters for the proposed task scheduling strategy as follows. At the initial search stage of the GA, we assigned the size of the chromosome colony, the crossover probability, the mutation probability, the maximum iteration number and the minimum evolution rate as 60, 0.75, 0.10, 40 and 3.00%, respectively. At the latter optimization stage of the AA, the importance α of pheromone, and the importance β of the inherent computing capability of node were both weighted as 1, the volatilization rate ρ was set as 0.3, the incentive factors Ce and Cp were evaluated as 1.1 and 0.8 respectively, the maximum iteration number was set as 120 and the minimum evolution rate was set as 0.30%.

B. Experimental results and analysis

We tested the GA, the AA, and the proposed method respectively in the simulation environment of grid system which included 72 computing nodes. The numbers of users of every experiment were 10, 20… 100, respectively. That the evolution rate was less than 0.30% was defined as the termination condition of these scheduling methods. The comparison of the running time of these three methods was shown in table 1.

From the experimental results in table 1, we knew that the implementation efficiencies of the proposed method and the GA were similar when the number of their users was small, which was due to that the GA could search the global solution rapidly when the size of the task scheduling problem was small, and it was not obvious for the combination of the AA to improve the scheduling performance of the GA. With the increase of the size of the task scheduling problem, the efficiency of the proposed scheduling method improved gradually, and when the number of uses was greater than 40, the scheduling performance of the proposed method outperformed the GA and the AA obviously, which was because that the proposed method combined the strong global search capability of the GA and the high solution precision of the AA. In summary, the proposed method was superior to the other two scheduling methods, and has the obvious efficiency advantage in the large-scale grid task scheduling environment.

VI CONCLUSIONS

Focusing on the task scheduling problem of grid system, we analyzed the advantages and the shortcomings of the existing grid task scheduling algorithms. According to comparing the GA with the AA, we found that the properties of the two algorithms complemented each other. On this basis, we proposed a dynamic and efficient grid task scheduling method based on the genetic-ant algorithm, which took full advantage of the strong global search capability of the GA and the high solution precision of the AA, and overcame the imprecise local solution of the GA and the inefficiency of the AA at its initial searching stage. The simulation results showed that the scheduling efficiency of the proposed method was superior to the GA and the AA obviously in the large-scale task scheduling environment.

ACKNOWLEDGMENT

Support by the National Natural Science Foundation of China (Grant NOs. 65073145 and U0935003)

REFERENCES

[1] Ian Foster, Carl Kesselman. The grid 2: blueprint for a new computing infrastructure (2nd edition). Morgan Kaufmann Publisher, USA, San Fransisco, 2004.

[2] Tracy D. Braun, Howard Jay Siegel, Noah Bech, et al. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. Journal of parallel and distributed computing, 2001, 61(6): 810-837.

[3] Harmel Singh, A. Youssef. Mapping and scheduling heterogeneous task graphs using genetic algorithms. In proceedings of the 5th IEEE heterogeneous computing workshop, 1996: 86-97.

[4] Noriyuki Fujimoto, Kenichi Hagihara. A comparison among grid scheduling algorithms for independent coarse-grained tasks. In proceedings of the 2004 symposium on applications and the internet-workshops, IEEE computer society Press, Washington, 2004: 674-680.

[5] Guan-Joe Lai. A novel task scheduling algorithm for distributed heterogeneous computing systems. Applied parallel computing. 2006, (3732): 1115-1122.

[6] Vincenzo Di Martino, Marco Mililotti. Scheduling in a grid computing environment using genetic algorithms. In proceedings of the 16th international parallel and distributed processing symposium, USA, Florida, 2002: 235-239.

[7] Z. Xu, X. Hou, J. Sun. Ant algorithm-based task scheduling in grid computing. IEEE Canadian conference on electrical and computer engineering, 2003, (2): 1107-1110.

[8] Ajith Abraham, Rajkumar Buyya, Baikunth Nath. Nature's heuristics for scheduling jobs on computational grids. In proceedings of the 8th international conference on advanced computing and communications, Cochin, India, 2000, 45-52.

[9] P. Shroff, D. Watson, N. Flann, et al. Genetic simulated annealing for scheduling data-dependent tasks in heterogeneous environments. In 5th IEEE Heterogeneous Computing Workshop, 1996: 98-104.

TABLE I. COMPARISON OF THE RUNNING TIME OF THREE SCHEDULING METHODS (MS)

The number of users	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
The GA	24	54	228	517	1228	2551	5160	9701	17903	30858
The AA	41	89	284	549	919	1578	2638	5176	8340	14931
The proposed method	29	58	194	495	718	1118	1795	2608	4197	6829