Cloud Computing Real-time Task Scheduling Optimization Based on Genetic Algorithm and the Perception of Resources

Jian Dong, Su-Juan Qin

State Key Laboratory of Network and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, 100876, China

Keyword: Genetic algorithm, Resource scheduling, Mult-fitness, Resources perception

Abstract: Current real-time task scheduling algorithm only focus on the user tasks' real-time demand, and these algorithms are not flexible enough to adapt for real-time change in heterogeneous systems. In this paper, by means of the characteristics of global optimization searching of genetic algorithm, from the point of user's real-time demand and the overall throughput of system, we design the fitness function based on real-time and overall throughput. Aimed at solving the existing problems of slow convergence speed of genetic algorithm, based on the strategy of resources perception, according to the size of the workload, the virtual machine parameters and load conditions (I/O intensive or CPU intensive), we guide the process of convergence of genetic algorithm, to make it to the larger probability of the corresponding type task variation or evolve to the adapted virtual machine, therefore we can accelerate the convergence process.

1. Introduction

Cloud computing is a new kind of calculation mode that can obtain application, data and IT service through the network [1]. How to schedule tasks timely and effectively, improve the efficiency of resource utilization and task execution, and maximize meet the needs of users on the service quality has been one of the core problems in cloud computing research. Meanwhile, the users have a variety of demand on cloud service, how to meet these demands (including the task completion rates, overall task completion time, cost and etc.) has bee a big problem.

There has been traditional real-time task scheduling algorithms like EDF (Earliest Deadline First)[2], MCT (Minimum Completion Time)[3],PRS (Positive And Reactive Scheduling)[4]. These algorithms are non-intelligent algorithms. The main advantage of these algorithms is simple, fast, but for the resource differences under cloud computing environment and a variety of customers' needs, and independent adjustment of extensibility and dynamic adaptability in the process of operation, these algorithms can't meet. Genetic algorithm is a classical optimization algorithm, it has implicit parallelism and intelligence which make it has advantages that traditional scheduling algorithm does not have. When the genetic algorithm is applied in scheduling, the main consideration is the task of the overall running time, ignoring QoS such as real-time task, cost and etc. And because the traditional genetic algorithm is easy to fall into local optimal solution, slow search speed, how to get rid of disadvantages of traditional genetic algorithm and improve the quality of its QoS has became the focus of research.

In this paper, we use dual fitness function including the real-time and overall throughput[5], and use genetic algorithm to search the approximate optimal solution instead of the global optimal solution to meet the demand of the user's real-time and overall efficiency. By the method based on resource awareness[6], we dynamically adjust the selection and evolution parameters of genetic algorithm and improve the convergence speed of the algorithm through interference in its evolution direction.

2. Algorithm Design

In this paper, on the basis of the original model [7], by introducing the overall running time and the completion of the task fitness function into the genetic algorithm, we meet the requirements of the traditional genetic algorithm in real time and quality of service. At the same time, we introduce the concept of resource awareness, control the evolution process of genetic algorithm through the

type of the task and virtual machine, thus we can make its evolution to where the task is appropriate to the virtual machine. In this method, we greatly improve the convergence speed of genetic algorithm.

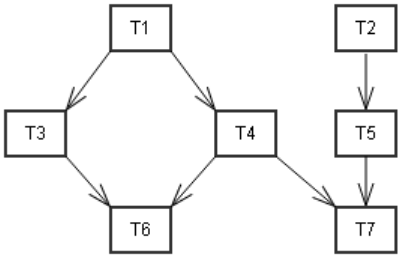
①model definition：

A collection of m processors P={P1，P2，..., Pm}，Pi is the ith processor. A collection of p user submitted job J={J1，J2，..., Jp}, Ji is the ith job. p jobs are divided into n tasks, with the corresponding collection T={T1，T2, ..., Tn}，Ti is the ith task.

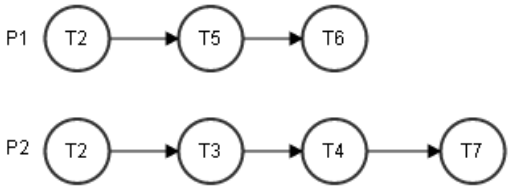
The task scheduling problems under cloud environment is to solve the problem of efficiently performing n parallel tasks depend on each other on the limited m resources, and to fully meet the user's real-time assignments and the overall throughput requirements.

We can use DAG(Directed Acyclic Graph) to represent each subtask scheduling constraints between the relationship.

For a DAG graph G<T,E>(like graph 1), where T is a sub-task collection. A sub-task Ti is a node of the graph G, E is to the edge of task dependency graph set. <Ti,Tj> Tj can said that perform before Ti is completed, then called Tj is a Ti successor. Now the target is to find a distribution scheduling strategy(like graph 2) that can assign n tasks to m processing machine and reasonably schedule execution order of each sub-task to make the individual tasks under the constraint of the dependency graph, improve the task of real-time response rate and reduce the total completion time.



graph 1



graph 2

②The generation of initial solution group:

We divide all the nodes in a DAG to $h'+1$ subsets DAG(i) according to the height where $0 \le i \le h'$ and h' is the maximum height among nodes in DAG. For subset DAG(i), randomly assign all the tasks in DAG(i) to m processing machine. then sort all the tasks assigned to the same processor in ascending order, thus we can get a legal scheduling. Repeat the above process, we can get a certain scale of the initial solution group.

③Fitness Function

Fitness function of genetic algorithm is used to determine the direction of evolution and evaluate the superiority of the individuals in the group. In this paper, the genetic algorithm is applied to the real-time task scheduling environment, compared with traditional real-time scheduling algorithm, we make the genetic algorithm can not only meet the real-time requirement, but also provide maximum throughput.

Suppose we have N*P-dimensional ETC(Expected Time to Compute) matrix, $t_{ETC}(i,j)$ indicate the expectations of the execution time when the ith task runs on the jth processor, then the J_m task execution time can be expressed as:

$$t(J_m) = \max_{j=1}^{P} \sum_{i=T_{Total}(m-1)}^{T_{Total}(m)} t_{ETC}(i,j)$$

The total time required to complete all p homework is: $t_{Total} = \sum_{m=1}^{p} t(Jm)$

We set $t_{\exp t}$ is the user's expected completion time for the job J_m, then the user satisfaction function of the completion time of job J_m is $W_{Time}(J_m) = \theta \ln[t(J_m)/t_{\exp t}]$, user satisfaction

function for the total task completion time is $W_{Totle} = \theta \ln[\max_{i=1}^{p} t(J_m)/t_{Total}]$.

Therefore, the fitness function of the job scheduling is: $f = -\omega_1 W_{Time} - \omega_2 W_{Totle}$, ω_1、ω_2 represent the weight coefficient of real-time and overall throughput.

④Genetic operator：

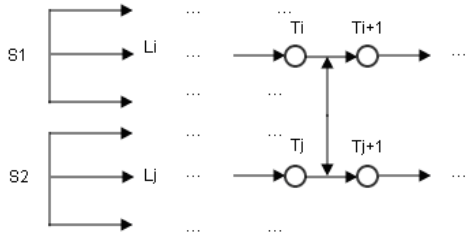
According to the disk I/O and CPU usage, we classify task workloads. Because we divide the task into smaller、process-similar sub-tasks, for each sub-task we suppose sub-task's DI(Data In), DO(Data Out) have a fixed proportion β. We define the task completion time is TCT(Task Completed Time), IOR(I/O Rate) represents disk I/O rate and n represents number of tasks asynchronously executing in a virtual machine. We define $n \times (1+\beta) DI \div TCT \div IOR$ as task load type parameter, if the parameter is bigger than 1, that means the virtual machine disk can't satisfy the I/O requests, it's I/O intensive; if it's smaller than 1, that means it's CPU intensive.

For the types of load for the virtual machine, we define its load type parameter as quotient of ratio between CPU utilization with base CPU utilization and ratio between I/O rate with base I/O rate. If it's bigger than 1, it means CPU intensive; else it means I/O intensive.

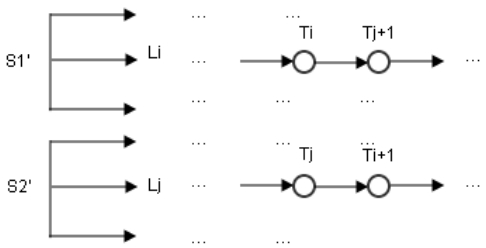
For any scheduling, We define the load fitness as: if both task and virtual machine load type parameter is bigger than 1 or smaller than 1, it means their work load fit, then the load fitness is their product of themselves (or their reciprocals); if their load type parameter is different, then the load fitness is product of the reciprocal of the one bigger than 1 with the other. By the load fitness we adjust the probability of crossover and migration in genetic algorithm to dynamically adjust the convergence of genetic algorithm process.

Crossover process mainly includes the MCX (improved hybrid operator) and NCX (internal cross operator).

MCX is to generate a new solution from two known solutions, the process is: for given two scheduling, first is to generate a random number n' , $0 \le n' \le h'$. Then we choose a cross point that can meet two points: the height of tasks behind the cross point is not equal with n'; the height of tasks preceding the cross point is less than or equal to n' (This process is like from graph 3 to graph 4). Thus the scheduling after hybrid of two scheduling is also legal. The probability of MCX is the production of the MCX probability that the user sets and the ratio of load fitness after the cross and the load fitness before cross.



graph 3



graph 4

NCX is the interchange based on the same scheduling within two tasks, the process is: first is to generate a random number n', $0 \le n' \le h'$. Then randomly select two task lists Li and Lj from a scheduling S1, determine the hybrid between Li and Lj according to n'. Exchange right half of Li and Lj from their crossing point, then we can generate a new solution $S'1$ (the crossing point should meet two conditions like in the MCX). The probability of NCX is like the process of MCX. The migration operator corresponds to mutation operator of genetic algorithm.

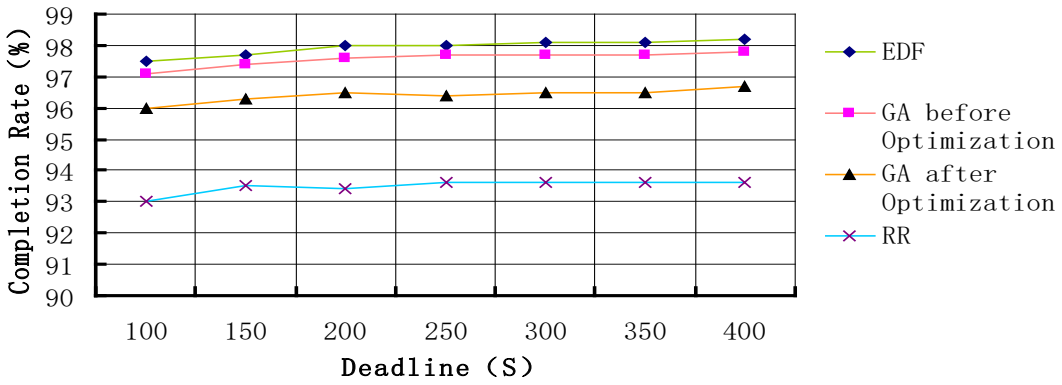
When the solution group is local convergence, we keep the diversity of solution group through the role of the mutation operator. In scheduling the task allocation, migration operator is randomly exchanging two sub-tasks at the same height in one scheduling.

The process of migration operator is that: first is to generate a random number n', $0 \le n' \le h'$. Then calculate the number of sub-tasks q'_i of each task list in scheduling S whose height is n',

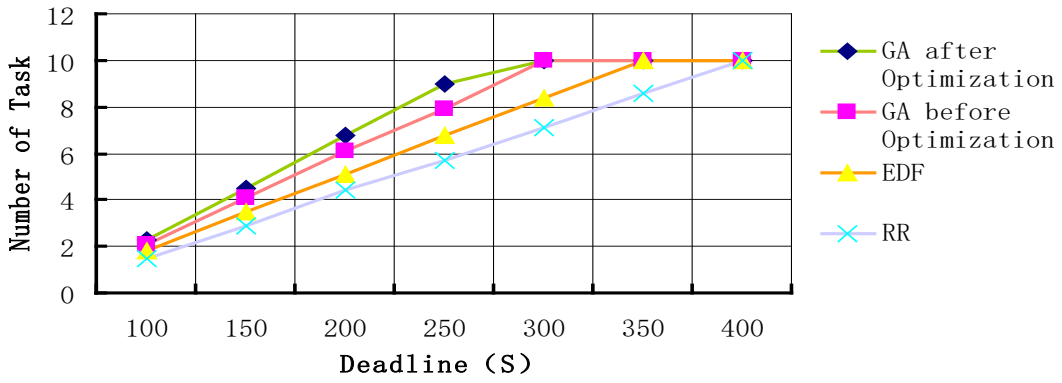
$1 \le i \le m$. At last we randomly select a task $T_k(h(T_k) = n')$ from task list with the maximum q'_i, then migrate T_k in to the task list with the minimum q'_i. Thus we can ensure that the new solution after the migration operation is also legal. The probability of migration is like the process of MCX.

3. Simulation test and the result analysis

In this paper, we use CloudSim to simulate. We import CloudSim jar to our experiment, and develop with Eclipse. We compare our algorithm with algorithms like RR(Round-Robin Scheduling), EDF(Earliest Deadline First) and genetic algorithm before optimized.



graph5



graph6

From graph 5, we can see that the algorithm in this paper can better meet the needs of users in real-time. As there is a process of convergence in genetic algorithm, it might be weaker in real-time compared to EDF. Because of the improved algorithm in the optimization in the process of convergence, it's better than genetic algorithm. From graph 6, we can see that compared with the traditional real-time algorithm EDF, optimized genetic algorithm have a obvious improvement in overall run time.

4. Conclusion

Current real-time task scheduling algorithm like EDF, MCT and etc, they only consider the efficiency of the completion of the task without other user's QoS requirements. In this paper, we import genetic algorithm into real-time task scheduling under cloud computing, and optimize this algorithm based on resources perception to change its evolution parameters, thus improve the convergence speed. Through comparing with traditional real-time task scheduling algorithm and genetic algorithm before optimized, we validate the improvement of efficiency of this algorithm. This improved algorithm can better meet the needs of the user's real-time task scheduling.

In addition, the work need to be solved is how to divide a real-time task into several sub-tasks,

and how we can evaluate the size of sub-task and I/O size. Thus this task scheduling algorithm is applied to the production environment.

Acknowledgment

This work is supported by NSFC (Grant Nos. 61300181, 61502044), the Fundamental Research Funds for the Central Universities (Grant No. 2015RC23).

References

[1]Travis J,King J, Qiao Rui-ping.Lab VIEW University practical tutorial[M]. Beijing electronic industry publishing house，2008.

[2]Mills, A.F.,Anderson, J.H.,2010, A Stochastic Framework for Multiprocessor Soft Real-Time Scheduling. In: IEEE 16th International Conference on Real-Time and Embedded Technology and Application Symposium(RTAS). IEEE, pp.311-320.

[3]Li,J.,Ming,Z.,Qiu,M.,Quan,G.,Qin,X.,Chen,T.,2011.Resource allocation robustness in multi-core embedded systems with inaccurate information(J).J.Syst.Archit.57(9),840-849.

[4]Chen Huang-ke, Zhu Xiao-min, Guo Hui, Zhu Jiang-han, Towards energy-efficient scheduling for real-time tasks under uncertain cloud computing environment(J).The Journal of System and Software 99(2015)20-35

[5]Li Jian-feng,Peng Jian. Cloud environment task scheduling algorithm based on improved genetic algorithm[J].computer application,2011,31(1):184-186.

[6]Xu Peng. Job scheduling algorithm research cloud computing platform[D].2011

[7]Zhong Qiu-xi，Xie Tao，Chen Huo-wang. Task allocating and scheduling based on genetic algorithm[J].Research and development of the computer，2000,10