An Efficient Task Scheduling Method in Cloud Computing based on Particle Swarm Optimization

1Cao Zheng, *2Sun Songmei
1 Shanghai Tobacco Group Co., Ltd., aberster@gmail.com
*2 Shanghai Institute of Ceramics, Chinese Academy of Sciences, smsun@mail.sic.ac.cn

Abstract

In cloud computing environment, task scheduling is one of key issues. The tasks are always needed to be allocated with different priorities according to users’ demands. To improve task execution efficiency, a novel scheduling strategy based on particle swarm optimization is proposed. The information of load state and available route is employed to achieve automatic load balance. The wandering position of task delivery is calculated by particle swarm optimization. The experimental results indicate this method is efficient with the comparison of random selection and ant colony optimization.

Keywords: task scheduling, load balance, cloud computing, particle swarm optimization

1. Introduction

The magnificent development of cloud computing, smart devices and internet of things bring us to the era of big data. The huge amount of data needs various distributed cloud computing systems such as IDC, public and hybrid cloud services with high performance of data processing and task scheduling. The goal of job scheduling is to increase the cloud resource utilization and not affect its service. The quality of service of cloud computing depends on the effect resource management and job scheduling by reasonable task allocation and load balancing mechanism in different servers clusters. The cloud scheduler can identify the right service node and dispatch appropriate task to finish the job. There are two major issues to be considered: the strategy of task assignment handled by the scheduler and the node selection method among the available servers. A task-level scheduling algorithms[1] were proposed with respect to budget and deadline constraints for a batch of map reduce jobs on a set of provisioned heterogeneous virtual machines in cloud platforms. Both local greedy and global gradual refinement were adopted for the optimization problems. A new Hadoop scheduling algorithm[2] was proposed based on the knowledge of workload patterns to reduce average job response times by dynamically tuning the resource shares among users. The problem of multiple users’ computations partition with the scheduling of offloaded computations on the cloud resources[3] was studied to achieve minimum average completion time for all the users. A priority-based method [4] was employed to consolidate parallel workloads in cloud by partitioning computing capacity of each node into two tiers. A a step-by-step slot allocation technique[5] was adopted to improve the performance of resource allocation of cloud workloads. An iterative ordinal optimization method[6] was proposed to achieve sub-optimal schedules from a global perspective over a long period. A system for scheduling job sequences was presented according to the optimization based on map reduce framework either with respect to execution time or monetary cost[7]. In order to improve resource utilization efficiency, the load balancing mechanism is designed to assign the workload to each computing node equally according to mission requirement, increasing the resource utilization ratio and decreasing the total task execution time. In this way, the load balancing mechanism and job scheduling method should be fair enough to explore the max computing power of each server.

In cloud computing environment, the state of node load and complicated network captivity change very fast. It is difficult to obtain the accurate information of all servers simultaneously. A load balancing and scheduling algorithm[8] was presented without known job sizes or upper-bounded. A trust service-oriented workflow scheduling algorithm[9] was presented combing direct trust and recommendation trust to balance different requirements of time, cost, and trust. In fact, the trust is usually hard to measure to be taken into count of cloud service.
Load balancing is one of hot topics in network research area. There are always two kinds of load balancing techniques: centralized and distributed. Lots of research has been done in the area of centralized load balancing. A centralized optimization framework LEISURE[10] was proposed for load-balancing network measurement workloads across distributed monitors. The flexible monitor deployment case was formulated as mixed integer linear programming problem and several heuristic algorithms were proposed to approximate the optimal solution and reduce the computation complexity. An integer linear programming model [11] was employed to handle multimedia task scheduling. The performance model of multiple servers worked together as a pooled resource was evaluated to achieve scalable mean delays in file downloads under stochastic loads[12]. An approach was proposed by creating several replicas of each job and sending each replica to a different servers. Based on the arrival signals, the latter sequences of replicas were removed them from the queues[13]. However, numerous information updates and complicated computation are needed to be provided by the centralized scheduler. The risk of single point of failure is increased in the centralized load balancing system.

There are various research have been carried out in the area of distributed load balancing. Konstantinou et.al [14] proposed a cost-effective load balance algorithm in distributed range-queriable systems. A hybrid mechanism based on iterative key redistribution between neighbors and node migration was utilized to achieve load balance. Manfredi et al. [15] presented network queues equilibrium method. The distributed algorithm is adopted in both time-continuous and time-discrete version for load balancing. The joint spatial-temporal load balancing algorithm[16] is proposed for energy cost optimization in data centers. The decentralized distributed gradient-ascent algorithm[17] is given to guarantee perfectly balance load around the network. The load balancing effect of distributed scheduling approach depends on the system model and optimization target. A lot of factors such as computation complexion and communication cost in different circumstance should be considered. Meanwhile, numerous of challenges of random node failure [18] and slow converge still exist in distributed load balancing.

A novel task scheduling approach based on particle swarm optimization is proposed in this paper. With the information of nodes type and its neighbor route, tasks can be delivered to different service nodes smartly. Moreover, the wandering position of each task can be found based on PSO to minimize the total task execution time.

2. System Model

The proposed task scheduling algorithm in cloud computing environment is based on following job scheduling framework as shown in Fig.1. The cloud model consists of a set of virtual machines (VM) made up of a series of entity servers (ES). The network facilities (NF) with high communication performance link all these computing and storage nodes together. When users submitted their jobs, the job scheduler receive them in sequence and deliver these tasks to the cloud control center. The control center then dispatch each job to appropriate VM in either parallel or distributed ways to accomplish the task. There are usually several kinds of clusters including numerous heterogeneous nodes in the distributed computing environment. The tasks are initialized randomly on different nodes. The optimization of scheduling is aimed to minimize the total tasks execution time.
2.1. Problem Formulation

In the proposed model, a typical cloud application contains several user tasks operated by job schedulers. There are usually two kinds of tasks: dependent and independent jobs. The former tasks should be carried out in several computing nodes sequentially while the latter can be implemented in distributed ways. In this way, the job scheduling problem can be represented in Directed Acyclic Graph (DAG), denoted as $G(T, E)$ (Fig. 2). The set of graph nodes $T = \{T_1, T_2, \cdots, T_n\}$ stands for the users’ job tasks. The set of graph arcs $E = \{E_{ij}\}$ represents the different sequential constraints of data or control flow. $E_{ij}$ denotes the edge between task $T_i$ and $T_j$, where $T_i$ is the parent task of $T_j$. In this way, the output of task $T_i$ carried out in computing node $i$ is the input of task $T_j$ running in node $j$. Besides, the $T_{start}$ is called start task without parent node task and $T_{end}$ denotes the end task node without child node task. Generally we add both $T_{start}$ and $T_{end}$ of no running time to formulate the task execution process, as shown in Fig. 2. The independent task can be actually executed in any node with both virtual task $T_{start}$ and $T_{end}$. The problem is how to schedule users’ job to appropriate computing node in cloud computing environment made up of various kinds of data centers. In the proposed model, the computing node is homogeneous and independent. There is no special scheduling policy with any precedence related to users’ jobs and the scheduling process cannot be interrupted.
In proposed cloud computing model, the users’ tasks can be scheduled to different computing nodes. The proposed parameters and definitions are listed in Table 1.

**Table 1. Parameters and Definitions**

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_i$</td>
<td>Task $1 \leq i \leq K$</td>
</tr>
<tr>
<td>$E_{ij}$</td>
<td>Task sequence transition duration between task $i$ and task $j$ if $i \neq j$</td>
</tr>
<tr>
<td>$R_j$</td>
<td>Computing node $R$ $1 \leq j \leq N$</td>
</tr>
<tr>
<td>$C_i$</td>
<td>Running time of task $i$</td>
</tr>
<tr>
<td>$D_i$</td>
<td>Deadline of Task $i$</td>
</tr>
<tr>
<td>$B_i$</td>
<td>Budget of Task $i$</td>
</tr>
<tr>
<td>$S$</td>
<td>Makespan of task</td>
</tr>
</tbody>
</table>

The aim of task scheduling is to dispatch each job to appropriate computing node within time deadline and budget constraint. These assumption is based on the different users’ requirement of various jobs and the real operation situation such as charge of services in cloud computing. Therefore, the task scheduling in cloud computing environment can be formulated into a multi-objective optimization problem.
\[ \text{Min} \sum_{x} F(x) = S(x), B(x) \quad (1) \]

s.t. \[ S(x) = \sum_{i,j=1}^{K} C_{ij}(x) + E_{ij}(x) \]

\[ C(x) \leq \sum_{i=1}^{K} D_{i}(x) \]

\[ B(x) = \sum_{i=1}^{K} B_{i}(x) \]

In the proposed optimization formulation, \( x \) is a feasible solution of this multi-objective function. \( D(x) \) is a function of performance objective related to makespan. \( B(x) \) is the financial constraint refer to the users’ cost supplied by cloud computing service. According to the given function, there is no global optimal solution to such multi-objective optimization problem because the constraints are conflict. Therefore, the solution often presented with compromise either in efficiency or in economy. In this way, the optimization method based on PSO algorithm is proposed.

3. Task Scheduling Based on PSO

3.1. PSO Algorithm

The multi-objective optimization problem brings great challenge in cloud computing research area because it is often difficult to find global optimal solution. In recent years, the evolution method attract a lot of interests. Particle swarm optimization[19] is a population based optimization technique inspired by social behavior. The optimization is implemented by iterative improvement of candidate solution with a given quality measurement. Every member in the swarm can benefit from the others’ experience. The particle in swarm can move in the space with a velocity affected by its own best position and the best position found by its neighbors at the time. The fitness of each particle is evaluated by the quality of its position. By the iterative optimization, the optimal position can be achieved. Recently, several scheduling algorithms based PSO have been presented, such as combining AIS[20], hybrid meta-heuristic[21][22] and self-adaptive learning[23] method etc. These derived PSO techniques may have good performance in general job-shop scheduling. There is still a big challenge for dynamic miscellaneous job scheduling in complex cloud computing environment.

The status of a particle \( i \) in time \( k \) is denoted by two parameters: its position \( x_i^k \) and velocity \( v_i^k \). The particles are initialized with random position and velocity. \( a_j \) is called acceleration coefficient of each component. \( r_j \) is random factor which is usually of uniform distribution. \( p^* \) is the best position found by the particle and \( g^* \) represents the best position found by its neighbors. The fitness is evaluated by the quality of the particles’ position and the best position \( p^* \) and \( g^* \) are updated simultaneously.

\[ v_{i+1}^k = w v_i^k + a_1 r_1 (p^* - x_i^k) + a_2 r_2 (g^* - x_i^k) \quad (2) \]

\[ x_i^{k+1} = x_i^k + v_i^{k+1} \quad (3) \]

The impact of historical velocity on current of the particle is regulated by the inertia weight \( w \). The portion effect of the best local and global position is adjusted by the acceleration coefficients \( a_j \). Equation 2 indicates that particle's velocity is updated comprehensively by its previous velocity and the distances to its own best historical position and its neighbors' best position. The new position of the particle is calculated in Equation 3.
3.2. General Set of Scheduling

Since the communication cost in distributed computing environment should be considered, the edges in the proposed direct acyclic graph are assigned with different weights to represent the communication cost. To implement efficient task scheduling, every task was assigned with ID, type, current node, target node and execution time. Meanwhile, each node also includes its node ID, type, capacity, current task and service time etc. If there is no task in current task list, the node state is set to be idle and denoted as serviceable (SA). Every task is delivered to the SA node to execute. During scheduling, every node’s load state (LS) is monitored and updated simultaneously to guide the tasks’ delivery. If the task was served by a node, it was removed from the global task list (GTL). Though it is called global task list, it can be used for task scheduling in local and distributed processing scenario. In others words, it can be seemed as an container of all tasks.

3.3. Task Execution Condition

First, the GTL contains all tasks randomly assigned on different nodes. Then, the type of current node is examined by LS monitor. There are three cases of initial task as it is shown in Figure 3. The arrow denotes the task and the circle represent the node. The different lengths of edge indicate the communication cost. The white circle means it is a SA node while the black one shows it is a non SA node.

![Figure 3. Different cases of initial tasks and servers](image-url)
For case 1, the task is luckily on the SA node. For case 2, the task is on the non SA node but its neighbor is SA node. For case 3, the task is on the non SA node and its neighbor is non SA node either. If the node is SA and its type is the same as task’s type like case 1, the task will be executed by this node. If the task is accidentally on its target node and the node is SA, the task can also be executed. That means, if the computing power of the nodes in cloud is strong enough, the users’ requirements can be always fulfilled.

If the task cannot be served by current node, it will be delivered to other nodes. In the proposed method, each node contains its route information (RI). One of nearest SA neighbor nodes is selected randomly. If the type of node is the same as the task’s type, the task will be delivered to this node as it is shown in case 2.

If the task cannot be accomplished by current node and its neighbor as it is shown in case 3, the new position of this task is decided by PSO. In proposed algorithm, every task is denoted by a particle and the particle’s position is the task’s current node. The nearest SA node is particle’s best position and task’s target position is the particle’s global best position. Each task is initialized with random velocity. The particle’s new position is computed according to Equation 1 until the task is executed by a node.

![Figure.4. The flow diagram of TPSO](image-url)
3.4. Scheduling Algorithm

Generally, the scheduling process can be described as following:
1. A DAG is created to denote clusters.
2. A number of tasks are initialized on the nodes randomly.
3. GTL full of tasks is generated. If GTL is not empty, the following loop begins.
4. For each task, the LS of current node is checked. If it is SA and the type is the same as the task’s type, the task will be executed.
5. If current node is not SA, the nearest neighbor SA node will be selected by RI. If its type is the same as task’s type, the task will be executed.
6. If current node and its neighbors are not SA, the task will be delivered to a new position decided by PSO.
7. The whole process will be repeated until GTL is empty.

Fig. 4 shows the flow diagram of proposed algorithm. In the beginning, the GTL is full of tasks. All the task are randomly assigned with the clustering nodes. Then, the LS of current node will be checked. If the current node is SA and its type is the same as the task’s type, the task will be executed by the current node. Otherwise, the nearest SA neighbor node will be selected according to RI. If its type is the same as the task’s, the task will also be served by the node. If the current node and its neighbors are all non SA, the new wandering position of the task will be computed by PSO.

Algorithm TSPSO

While GTL is NOT empty
{
    if (current OR neighbor node is SA) AND (task.type == node.type)
        Remove_from_GTL(Task)
    else
        new_position = PSO(Task)
}

The goal of proposed approach is to maximize the utilization of all SA nodes and minimize total task delivery and execution time. During the scheduling, the LS and RI were adopted to guide the tasks’ delivery and no more cost of communication and computing is needed. Considering the communication cost, it is not necessary to find all SA nodes in the system because some node will become idle after the wandering time given by PSO algorithm. The procedure of approach can be generalized as algorithm Task scheduling based on Particle Swarm Optimization (TSPSO).

![Figure 5. The distributed scheduler architecture](image-url)
Fig. 5 shows the distributed architecture of proposed task scheduling method. The approach can be used in distributed computing systems. The GTL is only employed for recording the total task lists. Each scheduler receives the LS and RI from network facilities of local cluster to exchange the information with each other. Meanwhile, the global nodes’ LS and RI can also be obtained from the all the schedulers. According to the proposed algorithm, each scheduler with comprehensive information processing ability can decide the target node of tasks. Besides this, there are no frequent LS updates in this method. Though the LS changes very quickly, it does not need the LS of all nodes in PSO computation. The intrinsic mechanism contains both the local and iterative global optimization. The task will be executed by local SA nodes at first to improve the efficiency. Therefore, the schedulers construct a robust scheduling system which is not only fast to process the tasks delivery but also strong to resist the single point failure.

4. Experimental Results

4.1. General Set of Experiments

In the experiments, random selection and ant colony optimization (ACO) [20] were implemented for comparison. The test system containing a number of clusters with heterogeneous computing nodes. The simulation environment is set up according to the cloud computing framework in Fig. 1. All of tasks were generated randomly from any place based on real user demand. To evaluate the performance of different algorithms, the power consumption and cloud service charge are not considered in the comparison experiments. CloudSim-3.0 were used to verify the algorithms performance.

<table>
<thead>
<tr>
<th>Type</th>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Center</td>
<td>Number</td>
<td>10-30</td>
</tr>
<tr>
<td>Processing Elements</td>
<td>Number</td>
<td>10-30</td>
</tr>
<tr>
<td>Task</td>
<td>Number</td>
<td>50-200</td>
</tr>
</tbody>
</table>

4.2. Scheduling Fairness

The experiments were carried out to evaluate two sides of load balance (LB) methods: fairness and efficiency. To evaluate the fairness of comparative approaches, the difference and variance of number of tasks of the heaviest and lightest node were calculated. The heaviest node means the number of task to serve is the largest while the lightest has the least. In the experiment, the number of nodes was set to twenty and the number of tasks is one hundred. Table 2 shows the result.

<table>
<thead>
<tr>
<th>Method</th>
<th>Difference</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>16</td>
<td>9.8</td>
</tr>
<tr>
<td>ACO</td>
<td>11</td>
<td>5.3</td>
</tr>
<tr>
<td>PSO</td>
<td>6</td>
<td>1.7</td>
</tr>
</tbody>
</table>

As it is shown in table 3, the random selection method has the largest difference and variance. It is because that no information on LS and RI is employed for task scheduling. In ACO algorithm, the global and local pheromone can be updated with the change of LS. However, that update process cannot keep up with the change of LS quite well due to the mechanism of uncertainty of route selection. The proposed approach based on PSO performs best in the fairness comparison of LB. The reason is mainly because the utilization of LS, RI, GTL and the efficient update process. Therefore, the max resource utilization can be achieved.
4.3. Scheduling Efficiency

The efficiency is another important aspect of task scheduling. To evaluate the efficiency, the number of servers was fixed to twenty and the number of task increased.

![Figure 6. Scheduling Efficiency of Different Methods](image)

Fig. 6 shows the comparison result of task scheduling of different approaches. It illustrates that the proposed algorithm based on PSO performs better than the other methods. Because of the random node and route selection, the random method has the worst result of fairness and efficiency of task scheduling. ACO is a communication based group decision algorithm affected by a number of factors. Due to its mechanism, the route and target node selection process is always too slow to keep up with the change of servers’ LS in time. In proposed method, both local and global solutions given by LS, RI and GTL are adopted to select proper destination position of task delivery. Most of tasks can be finished on proper node through shortest paths. Therefore, the efficiency of task scheduling is improved.

5. Conclusion

A novel task scheduling method is proposed based on particle swarm optimization. The load state and route information of each node is utilized to select proper position of task delivery. Experimental results indicate the proposed approach performs better than random and ant colony optimization both in the fairness and efficiency of load balancing.

6. References