Intelligent Method for Cloud Task Scheduling Based on Particle Swarm Optimization Algorithm

Ashraf B. El-Sisi, Medhat A. Tawfeek, Arabi E. keshk, Fawzy A. Torkey
Department of Computer Science, Faculty of Computers and Information, Menoufia University, Egypt

Abstract: The core idea of cloud computing is managing and scheduling uniformly computing resources that are connected by a network to provide user services according to the needs. One of the important issues in this environment is related to task scheduling. The scheduler should do the scheduling process efficiently in order to utilize the available resources. In this paper, a particle swarm optimization algorithm for cloud task scheduling has been proposed. Particle swarm optimization algorithm is a computational method that optimizes a solution to problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. The main goal of the proposed algorithm is minimizing the makespan of a given tasks set. The proposed algorithm has been compared with ant colony optimization and random algorithms using CloudSim toolkit package. Experimental results showed that the proposed algorithm outperformed ant colony optimization and random algorithms.

Keywords: Cloud computing, task scheduling Makespan, Particle swarm optimization, CloudSim.

1. Introduction

Recently, there has been a dramatic increase in the popularity of cloud computing systems that rent computing resources on-demand, bill on a pay-as-use basis, and multiplex many users on the same physical infrastructure [1]. Cloud computing services are becoming ubiquitous, and are becoming the primary source of computing power for both enterprises and personal computing applications. It provides an illusion of infinite computing resources to cloud users so that they can increase or decrease their resource consumption rate according to the demands [2]. Efficient algorithms are needed for task scheduling in the cloud environment to utilize the available resources. The scheduler should adapt its scheduling strategy to the changing environment and the types of tasks [3]. There are two players in cloud computing environments, cloud providers and cloud users. Each one has different goals. Providers want to maximize revenue by achieving high resource utilization, while users want to minimize expenses while meeting their performance requirements [4]. It is difficult to allocate resources in a mutually optimal way. Moreover, increasing heterogeneity and variability of the environment poses even harder challenges [5]. Therefore, dynamic task scheduling algorithms that are based on meta-heuristics, such as Artificial Bee Colony (ABC), Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO) are preferred for clouds that can be used to find solutions for difficult or impossible combinatorial problems [6]. In this paper, we address “the cloud task scheduling problem”, which is to allocate and schedule the submitted tasks in a way that providers achieve high resource utilization and users meet their application’s performance requirements with minimum expenditure. From provider’s perspective, PSO algorithm is proposed to optimize the task scheduling process in cloud computing. PSO belongs to the field of swarm intelligence inspired by the social foraging behaviour of some animals such as flocking behaviour of birds and the schooling behaviour of fish [7]. Over a number of iterations, a group of particles have their values adjusted closer to the member whose value is closest to the target at any given moment. Imagine a flock of birds circling over an area where they can smell a hidden source of food. The one who is closest to the food chirps and the other birds swing around in his direction. If any of the other circling birds comes closer to the target than the first, it chirps louder and the others veer over toward him. This pattern continues until one of the birds happens upon the food [6, 8, 9]. This scheduling strategy was simulated using the CloudSim toolkit package. Experimental results compared to Ant Colony Optimization (ACO) in [10] and random in [11] showed that PSO algorithm satisfies expectation. The remainder of this paper is organized as follows. Section 2 introduces background and scans some of the important related work. Section 3 covers the basics of PSO and the details of cloud scheduling based PSO algorithm. The implementation and simulation results are seen in section 4. Finally, Section 5 concludes this paper.

2. Background and Related Work

The goal of cloud computing service providers is to use the resources efficiently and gain maximum profit [5]. This leads the task scheduling to be a core issue in cloud computing. Scheduling of cloud tasks is a combinatorial optimization problem [10]. In combinatorial optimization problem, we are looking for an object from a finite set. This object is typically an integer number, a subset, a permutation, or a graph structure [12]. A new kind of approximate algorithm
has emerged which basically tries to combine basic heuristic methods in higher level frameworks aiming at efficiently and effectively exploring a search space. This class is known as metaheuristic and includes ABC, PSO, ACO, Simulated Annealing (SA) and Tabu Search (TS) algorithms [13]. A metaheuristic is formally defined as “An iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions” [12].

In cloud, millions of user share cloud resources by submitting their computing task to the cloud system. Scheduling these millions of task is a challenge to cloud computing environment [10]. Scheduling based genetic algorithms is proposed in [14, 15]. These algorithms optimizes the energy consumption, carbon dioxide emissions and the generated profit of a geographically distributed cloud computing infrastructure. Scheduling in cloud environment based ACO algorithms are proposed in [1, 3, 10, 16]. In these methods, the requests are collected. The scheduler considers the approximate execution time for each task and use heuristic approach to possibly make better decision. This enables to know about the actual execution times of a larger number of tasks.

An optimized algorithm for virtual machine placement in cloud computing scheduling based on A multi-objective ant colony system algorithm in cloud computing is proposed in [2]. Scheduling algorithms based ABC are proposed in [16, 17, 18]. These algorithms were proposed for solving job scheduling problems and handling the load balancing of tasks in cloud computing environments. A heuristic approach proposed in [7] relying on the PSO technique was built for dealing with job scheduling in Grid environments. This approach modifies the classical PSO algorithm by varying the inertia weight used in the inertia term of the velocity equation. The inertia weight is varied by applying a scheme where the weight decreases over the whole run. The research work in [8] proposed a PSO-based scheduler to assign jobs in heterogeneous computing systems and Grids. The algorithm updates particles in a discrete domain and proposes a position update mechanism that takes into account the characteristics of discrete variables. A heuristic based on PSO to schedule jobs to resources in a cloud that considers both job computation costs and job data transfer costs has been proposed in [9]. It dynamically optimizes the monetary cost of a job-resource mapping combination by basing on the solution obtained via the classical PSO algorithm. The existing scheduling techniques in clouds, consider a parameter or various parameters like performance, makespan, cost, scalability, throughput, resource utilization, fault tolerance, migration time or associated overhead.

In this paper, cloud task scheduling based PSO approach has been proposed for allocation of incoming jobs to virtual machines (VMs) considering into account the makespan to help in utilizing the available resources optimally, minimize the resource consumption and achieve a high user satisfaction.

3. Particle Swarm Optimization for Cloud Task Scheduling

3.1. Particle swarm optimization (PSO)

PSO was invented in 1995 [9]. Particles in the swarm fly through an environment following the fitter members of the swarm and generally biasing their movement toward historically good areas of their environment. The goal of the algorithm is exchanging information to share experiences about the carried out search and to find a place with enough food [7]. This is achieved by assigning initially random positions to all particles in the space and initial velocities. The algorithm is executed like a simulation, advancing the position of each particle in turn based on its velocity using the best known global position in the problem space and the best position known to a particle. The objective function is sampled after each position update. Over time, the particles converge together around good solution [8]. The algorithm keeps track of two global variables:

- Global best (gBest) value indicating which particle’s data is currently closest to the Target.
- Stopping value indicating when the algorithm should stop if the Target isn’t found.

Each particle consists of:

- Data representing a possible solution
- A velocity value indicating how much the data can be changed
- A particle best (pBest) value indicating the closest the particle’s data that has ever come to the target

The velocity value is calculated according to how far an individual’s data is from the target (the further it is, the larger the velocity value). In the bird’s example, the individuals furthest from the food would make an effort to keep up with the others by flying faster toward the best bird. The velocity value is computed by Eq. (1).

$$V_i(t+1) = V_i(t) + U_1 C_1 \times (pb_i - X_i(t)) + U_2 C_2 \times (gb - X_i(t))$$

Where, $V_i(t+1)$ represents the new velocity of an particle and $V_i(t)$ represents its current velocity. $U_1$ and $U_2$ are two random variables in the range [0, 1]. The constants $C_1$ and $C_2$ represent the learning factors. The x-vector records the current position of the particle in the search space. Each particle keeps track of its coordinates in the solution space which are associated with the best solution (fitness) that has achieved so far by that particle. This value is called personal or particle best (pb). Another best value that is tracked by the PSO is the best value obtained so far.
by any particle in the neighbourhood of that particle. This value is called global best (gb). There are a few common population topologies (neighbourhoods) that are used in computing velocity value [7].

- Single-sighted, where individuals only compare themselves to the next best.
- Ring topology, where each individual compares only to those to the left and right.
- Fully connected topology, where everyone is compared together.
- Isolated, where individuals only compare to those within specified groups [13].

After updating the velocity of each particle, each particle will move to the new position in the decision space by Eq. (2).

\[
\begin{align*}
X_i(t+1) & = X_i(t) + V_i(t + 1) \\
\end{align*}
\]

Algorithm 1 shows the basic template for the PSO algorithm [13].

Algorithm 1: Template of the PSO
Random initialization of the whole swarm

Repeat

Evaluate \( f(x_i) \)

For all particles \( i \)

Update velocities by Eq. (1)

Move to the new position by Eq. (2)

If \( f(x_i) < f(pbi) \) Then \( pbi = x_i \)

If \( f(x_i) < f(gb) \) Then \( gb = x_i \)

EndFor

Until stopping criteria

Output: Best solution found.

3.2. Task scheduling based on PSO

Cloud task scheduling is one of the most widely studied problems in computer science research that can be defined by the following question: Given a set of jobs, a set of VMs, a set of constraints, and an objective function, how should jobs be allocated to resources?

Task scheduling based PSO algorithm will be proposed to handle this question. Decreasing the makespan of tasks is the basic goal (objective function) from the proposed method. To model task scheduling, PSO is instantiated with particles, each maintaining one potential solution to the entire scheduling problem. The position of a particle is being placed in a search space. Furthermore, the global best position will indicate the best possible scheduling solution. The PSO algorithm tracks the overall best solution found by any of the particles in the PSO and the best solution here is associated with the makespan length.

The pseudo code of the proposed PSO procedure is shown in Algorithm 2.

Algorithm 2: Pseudo code of the proposed PSO procedure

Input: List of Cloudlet (Tasks) and List of VMs

Output: the best solution for tasks allocation on VMs

Steps:

1. Initialize:

   Set value of parameters Number_of_particles, \( V_{Max} \), \( t_{max} \).

   \( t=1 \).

   \( gBest=null \).

   \( gData=null \).

   Generate random solution for each particle

2. For each particle

   \{ Calculate solution fitness value

   If the fitness value is better than pBest

   Set pBest = current fitness value

   Set pData = current solution

   End If

   If pBest is better than gBest

   Set gBest = pBest

   Set gData = pData

   End If

\}

3. Sort particles by their pBest scores, best to worst

4. For each particle

   \{ Calculate particle Velocity

   Use Velocity to update particle Data

   \}

5. Increment \( t \) by one.

6. If \( (t < t_{max}) \)

   Goto step 2

Else

   Print gBest and gData.

End If

7. Return

In an initialization phase of the proposed algorithm, the numbers of particles are determined and the parameters are initialized. More particles are better but slow the program’s execution. \( V_{Max} \) variable is used to determine the allowed maximum velocity. \( gBest \) variable is null and \( t \) will be 1 to indicate that the algorithm will be in the first iteration. The solution
can be represented as an array of IDs of VMs representing the order of VMs. The first task will be mapped to the first VM in the array of VM’s IDs and so on (this solution is not the final solution but it just an initial solution). The data fitness value of solution here is related to the expected makespan of the solution. The expected makespan of the solution is computed by Eq. (3).

\[ EM = \arg \max_{j \in J} \{ \sum_{i \in I} (d_{ij}) \} \]  (3)

Where, EM is the expected Makespan, \( J \) is the set of tasks that assigned to the VM\(_i\) and \( d_{ij} \) which expresses the expected execution time and transfer time of the task, on VM\(_i\), can be computed with Eq. (4).

\[ d_{ij} = \frac{TL_{Task_i}}{Pe_{num_j}*Pe_{mips_j}} + \frac{InputFileSize_i}{VM_{bw_j}} \]  (4)

Where, \( TL_{Task_i} \) is the total length of the task that has been submitted to VM\(_j\), \( Pe_{num_j} \) is the number of VM\(_j\) processors, \( Pe_{mips_j} \) is the MIPS of each processor of VM\(_j\), \( InputFileSize_i \) is the length of the task before execution and \( VM_{bw_j} \) is the communication bandwidth ability of the VM\(_j\).

Each particle contains pBest and pData variables. pBest variable represents the data fitness value of particle’s solution. pData variable represents the solution itself.

The iterative phase simulates the behavior of particles to solve a problem. On each iteration in the PSO processing loop, each particle’s pBest value is changed when the found fitness value of the reached solution is better (also the pData is modified). The gBest and gData values only will be changed when any particle’s pBest value is better than gBest. gBest and gData gradually move closer and closer to the target until the algorithm reaches the stopping criteria. Different shape of how to compute velocity is used in the proposed algorithm. The velocity score is calculated using the global worst, defining velocity as the measure of how bad each particle is doing (as opposed to how good). The velocity of each particle is first computed by Eq. (5).

\[ V = \frac{V_{Max} \times pBest}{WorstpBest} \]  (5)

Where, \( V \) is the computed velocity of particle. The pBest is particle’s pBest value and WorstpBest is the pBest of worst particle and can be obtained from the step 4 in Pseudo code of proposed PSO procedure in Algorithm 2. It will be equal the last particle’s pBest in the sorted list. After computing velocity for the current Particle, the velocity is checked with \( V_{Max} \). If it is out of range, it will be back in range. Once the velocity of the current Particle has been determined, modifying data is done by swapping VMs within each particle own data set (updating the current Particle’s position). The amount of swapping depends on how bad it’s doing (velocity). Particles are pushed towards the global best by copying pieces of the next best particle’s data (single-sighted topology). Swapping and copying are done to all particles except the global best.

### 4. Implementation and Experimental Results

#### 4.1. Parameters Setting of CloudSim

Simulation is a technique where a program models the behaviour of the system by calculating the interaction between its different entities using mathematical formulas, or actually capturing and playing back observations from a production system [19]. CloudSim is a framework developed by the GRIDS laboratory of university of Melbourne which enables seamless modelling, simulation and experimenting on designing Cloud computing infrastructures [20].

The experiment is implemented and simulated with 10 Datacenters, 50 VMs and 50-1000 tasks under the CloudSim simulation platform. The length of the task is from 1000 MI (Million Instructions) to 20000 MI. The parameters setting of cloud simulator are shown in Table 1.

<table>
<thead>
<tr>
<th>Entity Type</th>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task (cloudlet)</td>
<td>Length of task</td>
<td>1000-20000</td>
</tr>
<tr>
<td></td>
<td>Total number of task</td>
<td>100-1000</td>
</tr>
<tr>
<td>Virtual Machine</td>
<td>Total number of VMs</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>MIPS</td>
<td>500-2000</td>
</tr>
<tr>
<td></td>
<td>VM memory (RAM)</td>
<td>128-2048</td>
</tr>
<tr>
<td></td>
<td>Bandwidth</td>
<td>500-1000</td>
</tr>
<tr>
<td></td>
<td>clouplet Scheduler</td>
<td>Space_shared and Time_shared</td>
</tr>
<tr>
<td>Datacenter</td>
<td>Number of PEs required</td>
<td>1-4</td>
</tr>
<tr>
<td></td>
<td>Number of Datacenter</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Number of Host</td>
<td>2-6</td>
</tr>
<tr>
<td></td>
<td>VmScheduler</td>
<td>Time_shared</td>
</tr>
</tbody>
</table>

#### 4.2. PSO Parameters Evaluation and Setting

Sensitive parameters that must be fine-tuned for PSO include the number of particles, \( V_{Max} \) and tmax. Table 2 shows the selected best parameters of PSO.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number_of_particles</td>
<td>100</td>
</tr>
<tr>
<td>( V_{Max} )</td>
<td>8</td>
</tr>
<tr>
<td>tmax</td>
<td>100</td>
</tr>
</tbody>
</table>

The cloud task scheduling algorithms to be compared in the experiments include ACO in [10], random algorithm in [11] and the proposed algorithm. Table 3 shows the selected best parameters of ACO algorithm as in [10].

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a )</td>
<td>3</td>
</tr>
<tr>
<td>( \beta )</td>
<td>1</td>
</tr>
<tr>
<td>( \rho )</td>
<td>0.4</td>
</tr>
<tr>
<td>Q</td>
<td>100</td>
</tr>
</tbody>
</table>

---

**Table 1. Parameters setting of CloudSim**

**Table 2. Selected best parameters of PSO**

**Table 3. Selected parameters of ACO**
4.3. Experimental Results of PSO, ACO and Random Algorithms

In the following experiments, the average makespan with different tasks set is compared. The average makespan of the PSO, ACO and random algorithms is shown in Figure 1. It can be seen that, with the increase of the quantity task, PSO takes less time than random algorithm. Although ACO performance is very near from PSO, PSO outperformed ACO in all instances of tasks. This indicates that the proposed algorithm take less time to execute jobs in each size of jobs than other methods because it combines intelligently different concepts for exploring the search space using parallelism technique and cooperative strategies that are used to structure information in order to find efficiently near-optimal solutions. That is the reason why proposed algorithm takes less time to execute jobs.

The random has the true random performance. It does not consider about the status of resources and the size of jobs and assign jobs to resources randomly.

ACO algorithm uses a positive feedback mechanism and imitates the behaviour of real ant colonies in nature to search for food and to connect to each other by pheromone laid on paths travelled. The ACO employs artificial pheromone trails that play the role of information that is dynamically updated by ants to reflect their accumulated experience in contributing to solve a scheduling problem. This is the reason that ACO takes less time to execute jobs but not less than PSO.

The degree of imbalance (DI) measures the imbalance among VMs. DI can be measured by different methods. The first method measures degree of imbalance as in Eq. (6).

$$DI = \frac{T_{max} - T_{min}}{T_{avg}}$$  \hspace{1cm} (6)

Where, $T_{max}$, $T_{min}$ and $T_{avg}$ are the maximum, minimum and average $T_i$ respectively [3, 10]. $T_i$ that is computed by Eq. (7) is the expected finishing time of VMi.

$$T_i = \frac{TL_{Tasks}}{Pe_{num} \times Pe_{mips}}$$  \hspace{1cm} (7)

Where, $TL_{Tasks}$ is the total length of tasks which are submitted to the VMi.

The small value of DI tells that the load of the system is more balanced. The average degree of imbalance of each algorithm with the number of tasks varying from 50 to 1000 is shown in Figure 2. It can be seen from Figure 2 that the PSO and ACO can achieve better system load balance than random algorithm.

The second method that measures the degree of imbalance using standard deviation is given by Eq. (8).

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2} \quad \text{for all } i \in \text{VM list}$$  \hspace{1cm} (8)

Where $\sigma$ is the standard deviation, $N$ is the number of virtual machines, $x_i$ is the finishing time of VMi and $\bar{x}$ is the average finishing time of all virtual machines. If the standard deviation value of a method is small, it means that the differences in load are small and the load of the system is more balanced.

The standard deviations for each algorithm are shown in Figure 3. It can be seen from Figure 3 that the cloud task scheduling based on PSO algorithm can achieve good system load balance than ACO and random algorithm.

5. Conclusions and Future Work

This paper experimentally proves that the proposed algorithm for cloud task scheduling based on particle swarm optimization algorithm problem is more...
appropriate than ant colony optimization and random algorithms. The main goal of the proposed algorithm is minimizing the total execution time of given tasks. The algorithms in applications with the number of tasks varying from 50 to 1000 evaluated. In future work the effect of load balancing factor will be considered.

References


