ABSTRACT: Task scheduling problem in cloud computing has become an active research topic due to the tremendous growth in the use of cloud computing. Cloud computing is a heterogeneous system and it holds and processes large amount of complex data. Scheduling tasks efficiently can lead to better performance and more throughputs in the system. In this paper in order to minimize the cost of processing time, a firefly scheduling algorithm is proposed to schedule tasks in cloud computing environments. Firefly algorithm is a metaheuristic algorithm, inspired by the flashing behavior of fireflies. We considered two costs for each task: communication and computation. A comparison is made between proposed algorithm and particle swarm optimization algorithm in task scheduling. Experimental results show that firefly task scheduling algorithm is more suitable to cloud computing.

Keywords: Firefly algorithm, task scheduling, cloud computing, attractiveness function

1. INTRODUCTION

Cloud computing is the latest distributed computing paradigm and it attracts increasing interests of researchers in the area of Distributed and Parallel Computing [1], Service Oriented Computing [2] and Software Engineering [3]. Cloud computing is a new paradigm for distributed computing. The new emergence of cloud computing technologies provides method to deal with complex applications which are the applications of great deal of data and needing high performance applications. Clouds have been define to be a type of parallel and distributed system consisting of inter-connected and virtualized computers. These computers can be dynamically provisioned as per user's requirements [4]. Berkeley [5] defined cloud as following: "Cloud Computing refers to both the applications delivered as services over the Internet and systems software in the datacenters that provide those services. The services themselves have long been referred to as Software as a Service (SaaS). The datacenter hardware and software is what we will call a Cloud. When a Cloud is made available in a pay-as-you-go manner to the general public, we call it a Public Cloud".

How to efficiently assign the cloud resources to the user, which we called task scheduling, is a key technical issue. Different methods have been proposed to schedule tasks efficiently. Scholars have proposed FCFS algorithm, greedy algorithm, genetic algorithm and other algorithms [6-11] to solve this problem. T. Kosar et al [12] proposed a scheduler in the Grid that guarantees that task scheduling activities can be queued, scheduled, monitored and managed in a fault tolerant manner. [13] Proposed a task scheduling strategy for urgent computing environments to guarantee the data's robustness. T. Xie [14] proposed an energy-aware strategy for task scheduling in RAID-structured storage systems.

Genetic Algorithm is another suitable approach which have been used in task scheduling problem in cloud computing. Genetic Algorithms (GAs) can give several satisfying solutions for choice by iterative evolutions over generations of scheduling solutions. Although GA is more time consuming than list heuristics, it is acceptable for applications with long runtime. In addition, the speed of GA can be accelerated by using parallel GA technology [15].

A genetic algorithm is designed as the optimization method for a new scheduler who provides better make span and better balanced load across all nodes than FIFO and delay scheduling [16]. In 2010, an optimal scheduling policy based on linear programming, to outsource deadline constraint workloads in a hybrid cloud scenario is proposed in [17]. In 2011, Sandeep Tayal proposed an algorithm based on Fuzzy-GA optimization which evaluates the entire group of tasks in a job queue on basis of prediction of execution time of tasks assigned to certain processors and makes the scheduling decision [18]. Authors in [19] extend previous work of [20, 21] to propose the Look-Ahead scheduling algorithm with Reliability-Driven (RD) reputation. To evaluate the reliability of resources, RD reputation considers the runtime of tasks by using the task failure rate (task failures per unit time) of resources to define the reputation. It also provides a real-time reputation that can be used to evaluate the reliability of each task directly using the exponential failure model. Based on the RD reputation, we propose Look-Ahead Genetic Algorithm (LAGA) to intelligently optimize both makespan and reliability for a workflow application.

Another approach is Particle swarm optimization algorithm. Particle Swarm Optimization (PSO) is a self-adaptive global search based optimization technique introduced by Kennedy and Eberhart [22]. The algorithm is similar to other population-based algorithms like Genetic algorithms but, there is no direct re-combination of individuals of the population.

Zhan, S. B et al. [23] proposed an improved PSO task scheduling algorithm. PSO algorithm was utilized by simulated annealing algorithm for faster global convergence. Guo et al. [24] formulates a model for task scheduling and also presented a particle swarm optimization algorithm based on small position value rule.

Salman [25] has shown that the performance of PSO algorithms is faster than GA in solving static task assignment problem for homogeneous distributed computing systems based on their test cases. Lei et al. [26] have shown that the PSO algorithm is able to get better schedule than GA based on their simulated experiments for Grid computing. In addition, the results presented by Tasgutireen et al. [27] have provided evidence that PSO algorithm was able to improve 57 out of 90 best known solutions provided by other well known algorithms to solve the sequencing problems.

Ant colony optimization (ACO) is another algorithm which has been used widely in cloud computing. [28] Studied resource scheduling problem and also presented an ant
2. METHOD

In this paper, we proposed a firefly algorithm to schedule tasks in cloud computing. The following section introduces firefly algorithm, section three is about proposed algorithm, experimental results and conclusion will be the other sections as well.

2.1. Firefly algorithm

Firefly algorithm is a nature-inspired metaheuristic optimization algorithm. This algorithm has been implemented in chemistry [30], clustering [31], image processing [32] and etc. Each firefly is considered as a solution. Each one has a flash. The primary purpose for a firefly’s flash is to act as a signal system to attract other fireflies. The brightness should be associated with the objective function. The objective function is the main function for evaluating solutions (fireflies). More brightness shows the solution is more appropriate. Xin-She Yang formulated this firefly algorithm by assuming [33]:

- All fireflies are unisexual, so that one firefly will be attracted to all other fireflies
- Attractiveness is proportional to their brightness, and for any two fireflies, the less bright one will be attracted by (and thus move to) the brighter one; however, the brightness can decrease as their distance increases
- If there are no fireflies brighter than a given firefly, it will move randomly

Firefly algorithm starts with random solutions. Depending on the problem, each firefly is considered as an answer and contains an array with size of L. These solutions are called population. Afterwards fireflies will be evaluated by objective function. Fireflies which have better solution, they are brighter. Fireflies with less light intensity will be moved towards fireflies with more light intensities. This procedure will be continued till expected answer is reported or max iteration reaches.

The pseudo code of firefly algorithm is described as below:

1. Generate an initial population of fireflies
2. Formulate light intensity (I) so that it is associated with objective function (f)
3. Define absorption coefficient γ
4. While (t < MaxGeneration)
   a. For all fireflies
      i. If light intensity of firefly i is more than firefly j then,
         1. Move firefly j to firefly i
         2. ary attractiveness with distance (r)
   b. For all fireflies
      i. Evaluate new solutions and update light intensity
     c. Rank fireflies and find the current best

In pseudo code t shows iteration in learning cycle. MaxGeneration is maximum number which algorithm is allowed to continue learning phase.

2.2. Proposed Method

In this section, we will introduce firefly algorithm for task scheduling in cloud computing. For simplicity in describing our new Firefly algorithm, we considered following three idealized rules:

1. All fireflies are unisex so that one firefly will be attracted to other fireflies regardless of their sex
2. Attractiveness is proportional to their brightness, thus for any two flashing fireflies, the less bright one will move towards the brighter one. The attractiveness is proportional to the brightness and they both decrease as their distance increases. If there is no brighter one than a particular firefly, it will move randomly.
3. The brightness of a firefly is affected or determined by the landscape of the objective function. For a maximization problem, the brightness can simply be proportional to the value of the objective function.

We modeled the task scheduling problem as a graph where V= {1, 2, …, n} represents the tasks of a application and E=|C| indicates the information exchange between these tasks. The edge weight ei,j between node i and j denotes the information exchange between these pair of tasks. The node weight w corresponds to the work capacity of the node. Figure (1) shows this graph. A task's processing cost will be variety according to the task being assignment to different processors. In this paper, our goal is how to minimize the communication execution time.

![Figure 1. Task scheduling graph. In this graph Cij is cost of information exchange between node i and j](image-url)
request. The attraction is controlled by the decision parameters defined by the proposed approach. The proposed approach defines decision parameters for each node and the parameters are considered as the attributes.

The nodes Attractiveness is measured by following formula:

\[
attr(n_i)_j = \frac{p_j}{cpu_i + mem_i + dlink_i}
\]

- \(attr(n_i)\) represents the attraction of task \( j \) to resource \( i \)
- \(cpu_i\) is CPU rate of node \( i \)
- \(mem_i\) is memory rate of node \( i \)
- \(dlink_i\) is transition rate of node \( i \)
- \(p_j\) represents processing time of task \( j \)

We stored feature of each resource in table like Table.1 which is shown as below.

<table>
<thead>
<tr>
<th>Resource</th>
<th>CPU rate</th>
<th>Memory Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>(r_1)</td>
<td>(C_1)</td>
<td>(M_1)</td>
</tr>
<tr>
<td>(r_2)</td>
<td>(C_2)</td>
<td>(M_2)</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>(r_n)</td>
<td>(C_n)</td>
<td>(M_n)</td>
</tr>
</tbody>
</table>

The process of selecting the node with least cost is inspired from the firefly algorithm in such way that, the Least distinct firefly will possess similar characteristics. Inspired from the theory, we subject a distance calculation Between the nodes in the scheduling queues. Before proceeding to the calculation, we find the node with least \(attr(n_i)\) values. The node with least \(attr(n_i)\) values is considered as the pivot point for the queue to calculates Least distinct nodes. The distance values of the node is calculated based on the Cartesian distance, which is Given by:

\[
Dist = \sqrt{\sum_{i=1}^{n}(n_i - n_j)^2}
\]

The distance is presented using the expression distance the \(n_i\) is the selected node and \(n_j\) is the comparing node. Once all the distance values have been calculated between the node values, the nodes are rearranged according to the least distinct node to the pivot node.

3. RESULTS

The proposed scheduling algorithm is based on firefly algorithm and its application is in optimizing task scheduling in cloud computing. The processing of the proposed approach is explained section three, now, in this section, we show the experimental analysis of the proposed scheduling algorithm by considering a simulated cloud network through Cloudsim [22] tool and java programming. Cloudsim is a Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services. In implementation of firefly task scheduling algorithm, following classes in Cloudsim were assisted:

- Task class: this class was developed for simulating behavior of tasks in cloud computing
- Datacenter Broker class: Datacentre Broker represents a broker acting on behalf of a user. It hides VM management, as vm creation.
- VM class: Vm represents a Virtual Machine. It runs inside a Host, sharing host List with other VMs.
- Datacenter class: This class deals with handling of VMs

Figure (2) shows classes which were developed to implement the proposed algorithm.

Figure 2. Classes which were developed for implementation of proposed algorithm

FireflyScheduling class consists of functions and parameters which are related to task scheduling part. In implementation of the algorithm, we considered resources with different processing and memory abilities. So, we developed Processor class which contains of different processing resources like:

- Intel Core_i7_Extreme_Edition_3960X
- Intel Core_i7_Extreme_Edition_980X
- Intel Core_2_Extreme_QX9770
- Intel Core_2_Extreme_X6800
- Intel Pentium_4_Extreme_Edition
- AMD FX_8150_Eight_core
- AMD Phenom_II_X6_1100T
- AMD Athlon_FX_60_Dual_core
- AMD Athlon_FX_57
- AMD E_350_Dual_core

Datacenter Broker class was implemented for the purpose of encapsulation. This class is responsible for managing resources and assigning them to each task. Also this class controls scheduling algorithm as well.

Firefly class was dedicated for implementing behavior of fireflies in the proposed algorithm.

For evaluation of firefly algorithm, we used NASA Ames Research Center dataset. This dataset has following features:

- Record number: 42264
- Preemption: No
- Start Time: Fri Oct 01 00:00:03 PDT 1993
- End Time: Fri Dec 31 23:03:45 PST 1993
There are two groups of users in dataset:
- Normal users
- System personnel

Following figure demonstrates the format of dataset.

In experimental tests, we evaluated our proposed algorithm with different number of resources. We compared result of firefly algorithm with particle swarm optimization (PSO) task scheduler. Table (2) represents the results:

<table>
<thead>
<tr>
<th>Number of cloudlets</th>
<th>PSO (second)</th>
<th>Proposed algorithm (second)</th>
<th>Number of resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>44.235</td>
<td>40.430</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>25.115</td>
<td>22.320</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>19.889</td>
<td>17.650</td>
<td>10</td>
</tr>
</tbody>
</table>

**REFERENCES**


