Offering a New Approach to Optimal Scheduling of Tasks in the Cloud Using Chromosome Portioning

Ali Heydarzadegan*, Yaser Nemati, Mohammad Iman Jamnezhad, Mohsen Moradi

Department of Computer Engineering, Beyza Branch, Islamic Azad University, Beyza, Iran

*Corresponding author email: A.heidarzadegan@yahoo.com

ABSTRACT: It is for a long time that the cloud computing shows off in the range IT. Scheduling of the Cloud services have influenced by providers cost benefit of the computing paradigm. In this Scenario the whole of tasks must be Scheuled in the best way so that the cost and runtime reduce. In this thesis we have tried to discuss scheduling tasks based on genetic algorithms and parallelism will be investigated in detail. In the proposed method, independent tasks on homogeneous cloud environment are considered. In this algorithm, each task in addition to runtime, has survival time (That is, each task has a limited time to run if not running in a particular period its implementation value is reduced) Just like real-time systems and the factors of survival time for each task show its priority task to run. The main advantage of the proposed model is that it is both reducing the total execution time and trying to complete each task in the fastest possible time, before the deadline finishes. This issue is particularly important in real-time systems and compared to a simple real-time scheduling algorithm its execution time is less. Simulation of the algorithm in Cloudism environment shows that by increasing the resources for a fixed number of tasks, how much run time can be reduced to the system.

Keywords: cloud computing, task scheduling, genetic algorithms, fitness function, composition and mutation

INTRODUCTION

Cloud computing is a new developed computing model, based on distributed computing, cluster computing and grid computing. In addition to all the features of the case, the above calculation also has the following benefits such as: paid based on the desired application, reliable and low cost. Depending on the type of services that can be divided them into three categories: IaaS that infrastructure as a Service, PaaS considers Platform as a Service, SaaS considers software as a service. Timing is a problem of assigning tasks to machines to complete the job. In this problem, we must have a number of tasks and resources. Generally, the scheduler assigns tasks to workers. Achieving to the most optimal scheduling is the main motivation of timing. Genetic algorithm, which is fairly a complete theory was presented by Professor Holland and his colleagues is growing rapidly, and with a set of solutions (which are represented by chromosomes) begins. When users send their requests to the cloud to complete the request in a queue are collected and the most important issue in the timing is timely response to the demands of a queue. Whatever the answer may be faster, users will be more satisfied.

For an appropriate response we should have an appropriate scheduling and for an appropriate scheduling should be applied a suitable policy.

Genetic Algorithm

Genetic algorithm is a search algorithm that is based on the laws of evolution and works based on natural genetics. Numerous of the population solutions through reproduction and mutation, new generations are produced. According to the theory of evolution, only the best survival elements that generate, their biological inheritance is passed down to future generations. Genetic formulation of a problem, begins by defining an appropriate chromosome coding. To achieve a good performance, chromosomes must be simple because it makes operators are simple and quick to use. To schedule a task, a chromosome represents a solution to the scheduling problem, in other words, a schedule including allocation of processors and start time to each task. To create a schedule, an allocation of processors and the execution order of the tasks is enough so that it is not longer than any other schedule with the same allocation of processor and the execution order of the tasks implementation of the tasks.
Cloud computing

The basic idea behind cloud computing is that softwares must be installed on the user’s computer rather than placed on a server to be online. Programs typically run in a web browser and thus, you can run the program from any computer that has access to the Internet. In today's society, information technology and the Internet has become an integral part of people's lives population and by changing lifestyle needs such as security of information, rapid processing, instant access to information, and the most importantly save costs has also changed. Thus, by developing of these needs, organizations and individuals, have the requirements are quite different in the field of electronic services.

A Background Scheduling In Cloud Computing

Many researchers in the past, several scheduling algorithms like static, dynamic and scheduling strategy combination has proposed. Some static algorithms, such as MCP (Gajsky et al, 1990), ISH (Runnin et al, 1994), ETF (Gueve et al, 2009) for high-speed Internet, overlook the communication latency and they are suitable for the small environment. While some other algorithms such as DSL and MH (Lewis and Runnin, 1990) are algorithms based on the ANP, these methods are suitable for large distributed environments. Some algorithms in the dynamic scheduling and in sharing efficient load and load balancing that is in the distribution of tasks have performed by smart distributions and consistent distributions. The main purpose of the complex scheduling is that to provide equal distribution of dedicating computational tasks and to reduce communication costs of all nodes in distributed computing, also balance of the scheduling is performed by the capacity calculation of each node. Other studied scheduling algorithms by researchers include: central scheduling, timing intelligent scheduling and scheduling based on the agent negotiation. In cloud computing environment, VM resources are scheduling purposes, in the system aggregation is large, while in the traditional computing environment environment, the purpose of scheduling is a process or a procedure so that aggregation is tiny and data transmission is so small.

In the following three strategies for resource allocation are described.

Scheduling Background of the genetic algorithm in cloud computing

Genetic algorithm is fairly a complete theory, and it is the way that they were presented by Professor Holland and his colleagues. In this algorithm, trying to explain the natural systems from the complexity of biological and adaptive processes and, as well as Mechanisms of Biological Evolution to stimulate an artificial model. Its nature is a kind of parallel efficient in the form of global search method that can automatically collect knowledge of the search space and control the process of adaptive search in order to achieve an optimal solution. In recent years, methods of artificial intelligence, such as evolutionary computation, particularly in the field of genetic algorithms, due to its implicit intelligence and parallelism has been considered by different people. GA, has extensively been used to solve the problem of resource scheduling in large-scale, non-linear cluster systems, and to achieve ideal effects. When using GA Classic for removing the problems of aggregation of large, high-dimensional and large data, setting optimization problems such as incomplete convergence and slow convergence or lack of that is inevitable. Thus, researchers have proposed a variety of improved genetic algorithm.

The Proposed Algorithm

In the proposed algorithm at first we determine the priority tasks in a separate queue. In the proposed method, the first priority tasks is determined in which for all of the tasks the remaining time is calculated through its survival time. Thus, if we assume that the execution time of each task Runtime(Ti) and survival time each task is TTL(Ti) also to consider the viability of the deadline for each task, the following equation is obtained.

$$\text{Deadline(Ti)} = \text{TTL(Ti)} - \text{Runtime(Ti)}$$  \hspace{1cm} Equation 1-5

After timeline achieved for each task, tasks are ordered according to the respite thus for each of them search operations on the number of tasks is done whether for a task with its own deadline, there is another task that its execution time is less than the deadline time of that task?

$$\text{Runtime Tj} < \text{TTL-Runtime Ti}$$

If exist (the smaller task that its runtime is smaller than the deadline of the first job) is placed before it. This means that smaller tasks that have more respite is not necessary to wait long for all the great tasks that have little respite.

Search operation is performed on the number of task to determine which tasks can be started for each task before the task. When it was determined for each task in order to set priorities, tasks that no other task can not be placed before them put in a priority queue, after those the tasks put that only one task can be run before them and so on to the end, tasks, according to the number of tasks that can be performed ascending and take priority. For example:
if the task is considered 3
static
final int taskN = 3;
And the number of sub-tasks for each task, it is defined as:
public static int[] subtaskNA = new int[taskN]; //set maximum value
Determining the two main parameters in the matrix below is to take place at the beginning of storage.
The required time each task to perform. 1) 2) The survival time of each task.
Runtime and survival time tasks and sub-tasks of (regarding the simulator will generate the data) will be
determined randomly in the subtask matrix.
static int[][] subtask = new int[taskN][subtaskMax]; //set subtask number in for loop
For each task the deadline is calculated. Search operations for each task to determine the number of tasks can be
executed before the task is performed.
The next step is to sort tasks based on the number of tasks that each task can run before it and places at
the first of list and so on are arranged. May several task have the same priority, then regarding to the less deadline
any of Min(Deadline Ti) be taken priority.
Obtained list is the best sequential that the our algorithm has fined to run tasks, such tasks that have
deadline less than the rest are listed at the beginning of list tasks have deadlines, and they have a little time to
wait for ending the short-term tasks are placed after them. (This process is illustrated with a numerical example in
Chapter VI).
After obtaining the best sequence of tasks, taking turns is an appropriate arrange of the resources so that
are written on these tasks. Obtain the appropriate resources is made by using genetic algorithms.
The algorithm should give the appropriate order of resources to us in output so when we apply the order on the
prioritized tasks, both the runtime of the whole tasks reduces and all tasks in the fastest time (before the end of
their respite) are performed.
1. The first stage of the algorithm, the initial population is generated. In this method, the initial population and the
length of the chromosome (based on population is considered) is calculated according to the formula (For example,
if we want the length of chromosome 15, the initial population should be 2.5 to 2 times the length of the
chromosome) and then we coded (Various permutations of the numbers of each source) of course we apply one of
theses sections in the fitness function to prevent dispersion of the population in terms of the chromosomes to be
much better.
The initial population: static final int populationN = 30;
In the first stage, the chromosome packaging operations with different permutations of the resources starts.
Of course the initial gene in the permutations should not be duplication but the other genes are randomly can be
repeated. This condition is due to the resources that they want to run tasks, for tasks with high priority time to start
from scratch. This means that if we have 10 sources and in the queue tasks we have seven high-priority tasks (no
task can be performed before them), the tasks at zero of any source are begun.
ABC chromosomes Construction or in fact the values that each gene can take is the following permutation of the
sources. For example, if the number of sources considered to be 3, meaning that each chromosome contains a
permutation of the numbers 0, 1, or 2 will be.
private static final char[] ALPHABET = new
char[taskN];for(int i = 0; i < taskN; i++)
{ // k is number of vms
subtask[i][k] = i;
}
for(int i = 0; i < taskN; i++)
for(int j = 0; j < taskNA[i]; j++)
for(int k = 0; k < k; k++)
{subtask[i][j][k] = new Random().nextInt(10);}
The output of this stage of the algorithm, 30 numbers are from the initial population of chromosomes and each
chromosome contains a permutation of number of sources that is 0, 1, or 2 will be. However not duplicate the
condition of the early genes. Below is an example of a chromosome in the initial population can be seen.
01211021011011 01201102021201 01220012011111 01211000200100 01222101020101
2. The second stage of the algorithm, the fitness function is applied on the chromosome that it is an important part
of the genetic algorithm, such a policye the fitting function is applied in order to achieve a better resource in
chromosomes. In this function, the order of more appropriate resources is obtained in two conditions. First, the repeated absence of a particular resource to a certain extent, which means that the length of a chromosome is filled from 3 sources, the more closer the iterations sources together in the chromosome, the greater value gives to the chromosome second, the early genes are not repeated on the number of references for example, if you have 10 sources in the system, not be duplicated 10 early genes, this is because of the genes want to run high-priority tasks and it is better the primary tasks with high priority begin from resources in which the start time for the task at the source be zero.

This condition in the production of the initial population of chromosomes was for. The following formula gives us the appropriate number of this repetitions in the chromosome.

\[
N(i) = \left[ \frac{L\text{(length of chromosome)}}{N\text{(number of VM)}} \right]
\]

It is estimated that, how much replication of each resources in a chromosome can maintain the load balancing as well?

If we divide the length of chromosome on the number of sources, the target number will be the best number of replication for each source in the chromosome.

And no matter how many the number of replication of any source in chromosomes is closer to this number the chromosome will be better and has better load balancing and thus receives higher value of the function. On the other side the more distance number of replication of any source be far from the obtained number the more unbalanced is the chromosome in load distribution thus it will have the less value. In addition no matter how many the number of primary genes on the number of sources not to be replication in the chromosome it will be better.

The output of the stage from the algorithm is the valuable chromosome. So that to all of existing chromosomes in the initial population are given a value from 0 through 1. In the following is seen several chromosomes various values.

<table>
<thead>
<tr>
<th>Chromosome</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>0121102101110</td>
<td>0.21671</td>
</tr>
<tr>
<td>012011020210111</td>
<td>0.92962</td>
</tr>
<tr>
<td>012110002001002</td>
<td>0.31221</td>
</tr>
<tr>
<td>012012020212011</td>
<td>0.99899</td>
</tr>
</tbody>
</table>

3. The next stage is the step of selection that here is used the way of elitism in the way that the best chromosomes of each generation is transferred automatically to the next one. In the step chromosomes are ordered descending and after being arranged the more high-value tasks (that are in the range of first, second, third, forth and etc) for the combination step are selected together.

The output stage of the algorithm is the best choice of the initial population has earned a higher value from the fitting function. In the following you can observe some selected chromosomes.

<table>
<thead>
<tr>
<th>Chromosome</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>012012020212011</td>
<td>0.99899</td>
</tr>
<tr>
<td>012011020210111</td>
<td>0.92962</td>
</tr>
<tr>
<td>012001122020110</td>
<td>0.67860</td>
</tr>
</tbody>
</table>

4. After selecting the appropriate chromosomes, two chromosomes and to produce optimal offspring arrive. Here for composition operator, we have used single-point method. The output of this stage is the children produced from superior chromosomes that were selected in the previous step. An example of a combination of two chromosomes with a single point (the seventh point) you can see below.

<table>
<thead>
<tr>
<th>Chromosome</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>012012012201201</td>
<td>0.99899</td>
</tr>
<tr>
<td>012012020212011</td>
<td>0.92962</td>
</tr>
<tr>
<td>012001122020110</td>
<td>0.67860</td>
</tr>
</tbody>
</table>

5. And finally the time of applying mutation operator comes up. In this algorithm, a method to modify a gene is randomly selected, we have taken the mutation rate 20% of the children. On the below chromosome fourteenth gene has changed.

<table>
<thead>
<tr>
<th>Chromosome</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>012012020212011</td>
<td>0.99890</td>
</tr>
<tr>
<td>012011020210111</td>
<td>0.77860</td>
</tr>
</tbody>
</table>

6. Next, applying again of the fitness function is on the new offspring resulting from the combination. If the high value of the fitness function is gained to the initial population are added to find the re-election chances and if they have little value (less than half) will be removed.

<table>
<thead>
<tr>
<th>Chromosome</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>012112020212001</td>
<td>0.99890</td>
</tr>
<tr>
<td>012001122020110</td>
<td>0.77860</td>
</tr>
</tbody>
</table>
The termination condition of our algorithm, is that to repeat a certain extent (5 times replication of the algorithm steps).

Finally, the output algorithm in each generation to optimize goes up, and the final chromosome means that is the best distribution of resources our algorithm has. The distribution of resources, have initially been mapped on the tasks prioritized.

By the order of sources produced by the output algorithm, in addition to we get to better load balancing when we map this order of sources on the tasks that initially classified both the runtime tasks will be less and they will be as fast as they can till ending their time.

**CONCLUSION**

Since this system is a real time and in real time system, and the main problem of real-time systems is the implementation tasks before the deadline is completed in addition, these principles are respected in our proposed algorithm, any task is performed at the earliest possible time and by placing the condition of uniform distribution of sources in the fitness function, in addition the result of scheduling quickly responds to the tasks with deadlines less, it also causes to reduce the runtime. Our algorithm is considered static type and two special advantages of the method is static rather than dynamic method where first scheduling done based on all of the waited tasks. Secondly, there is less of a burden. Sources with the best of the situation are assigned to the tasks. So better to share the load on the CPU, the better the efficiency of the sources we will have. Because the sent requests to the cloud is very high, and the cloud is full of resources. This type of scheduling in the cloud with enormous scale of tasks and resources, will have a great runtime reduction. The most manner for the optimal algorithm is the one that the ratio of number resources to the number tasks, not be too low. Otherwise, inevitably a number of tasks with less deadlines delay fall. For example, if the number of tasks is 100 times and the number of sources is 10 times it is suitable. But if resources is 3 times resulting in a loss of tasks with high priority that is not suitable for real-time systems. This kind of scheduling tasks on cloud systems that are rich in resources, is very convenient.

**REFERENCES**


