Advances in Computer Science and Information Technology (ACSIT)
Print ISSN: 2393-9907; Online ISSN: 2393-9915; Volume 2, Number 10; April-June, 2015 pp. 17-20

http://www.krishisanskriti.org/acsit.html

Priority based Scheduling Algorithm with Fast Task Completion Rate in Cloud

Pankajdeep Kaur[1] and Parampreet Singh[2]

[1]Guru Nanak Dev University Guru Nanak Dev University India
[2]Regional Campus, Jalandhar Regional Campus, Jalandhar India
E-mail: [1]pankajdeepKaur@gmail.com, [2]param.pasricha@gmail.com

Abstract—*Task Scheduling is one of those critical issues that can tremendously increase or hamper the Quality of Service in a cloud environment. This Paper presents to you a new Algorithm whose sole purpose is to provide efficient scheduling based mechanism so thatQoS is achieved while handling different Tasks. The Algorithm presents a Two-Fold mechanism. In the First phase, the presented algorithm calculates the priority of the given tasks according to certain specialized attributes in the tasks. Then, tasks are arranged in a sorted order by considering the calculated priority. The second phase computes the time that each task require to execute under different services and thereby scheduling each task given in the sorted queue to a service that will cater the task with minimum amount of time. The Experimental results under CloudSim undoubtedly shows that the proposed algorithm achieves good performance not in scheduling but also in users' tasks completion rate.*

Keywords*: cloud computing; Quality of Service; Task Scheduling; Priority based task scheduling Fast task completion rate.*

1. INTRODUCTION

Cloud Computing is one of the technology that has completely revolutionized the use of computer in the world. It is a new way of modern business computing and provides services virtually through the use of "cloud centers" [1]. Apart from just computing power, users can also opt for cloud storage, infrastructure and numerous other services using different applications and paying on the basis of their usage. Scheduling different tasks on the given number of resources is one of the inevitable issue that needs some serious attention in order to achieve best service for both cloud service providers and to users as well.

Quality of Service (QoS) is something that is determined mostly at a user level as the satisfaction level of the user for the particular service [6]. Various factors determines the QoS such as completion time, execution price, packet loss rate, reliability etc. For providing QoS in cloud computing, paper [7] provides a scheduling algorithm completely based on Berger Model. This algorithm uses a two-fold fairness constraint In the First part tasks are classified according to QoS preferences, then a general expectation function is established corresponding to the classified tasks. In the second part fairness justice function is computed to achieve fairness in resource allocation. In paper [8], a reliability based scheduling algorithm was catered to consider reliability of the tasks. For this a Reliability priority rank(RPR) was used to keep track of the reliability requirements of different tasks.

This Paper presents an algorithm that tries to schedule the task in such an order that apart from considering only the priority of tasks, it tries to calculate expected executable time of different tasks on different services. The priority is calculated by considering different attributes of task such as User Level, Expected Priority, Length, Waiting Time etc.

Further part of the paper is organized into the following sections: Section 2 contains the nomenclature that is going to be used in the entire algorithm and method of implementation of the algorithm. Section 3 describes the complete detailed algorithm. Section 4 contains the simulation results. Section 5 provides the conclusion and references.

2. SCHEDULING MODEL

In this paper, we use the following two criteria's: (1) Higher priority tasks should be scheduled prior to the one's having lower priority. (2) Complete the task as early as possible which eventually will lower the cost of using resources of the service provider and also achieves QoS at user level.

2.1 Nomenclature of Services

As we are pretty much aware about the fact that in cloud computing, we have resources enclosed into cloud services. We are donating service set as P(q),so P(q)={P1,P2,P3… Pm} where $m \in I^+$, I^+ is positive set of Integers. Service P(*z*) is donated by a multi-tuple as follows:

P(*z*) = {P_Code, PSP, PName, PAtt}, Where each one is defined as under:

1) P_Code is the ID of P(*z*);

2) PSP is the service provider of P(*z*);

3) PName represents the name as well as method of service.

4) PAtt represents attributes of the service. We define PAtt={Storing Capacity, Processing Elements,and Bandwidth}.

2.2 Task Nomenclature

Task is something i.e. more often regarded as that container which contains the needs of the users that are required to be fulfilled. In general, Tasks are organized as:

Dependent tasks.

Independent tasks.

Here in this paper we will be concentrating only on independent tasks. Assume the task in denoted as W(*k*), then, W(*k*)={W_1, W_2, W_3…..W_k}, Where $k \in I^+$, I^+ being the set of positive Integers. Now suppose W(*i*)∈ W(*k*). Then We define Task W(*i*) as:

W(*i*) = {W_Code, WUserLevel, WTypeService, WPriorityExpectation, WServiceAbilToExe, WLen, WSubTime, WCurrState, WIOData} Where:

(1) W_Code is the ID of W(*i*).

(2) WUserLevel denotes the privilege class or level of the user, we set them as WUserLevel={X, Y, Z};

(3) WTypeService represents the expected Services of task W(*i*).

(4) WPriorityExpectation is the Expected Scheduling priority of task W(*i*). In other words, it represents the urgency of the task. We define it as:

WPriorityExpectation∈{VI, I, M, L} Where:

VI= Very Important.

I= Important.

M= Medium.

L= Low.

(5) WServiceAbilToExe shows the expected quantitative need of service to complete task W(*i*). We may define it as:

WServiceAbilToExe = {Pro,Band,Store} Where:

Pro= processor.

Band= Bandwidth.

Store= Storage.

(6) WLen is the length of task W(*i*). Here length corresponds to the unrolled loop instructions in taskW(*i*).

(7) WSubTime is the submission time of W(*i*).

(8) WCurrState is the current State of the task W(*i*).

(9) WIOData is the data(Input/Output) that needs to be stored in files.

Now in the next section we will be concentrating on calculating the priority of task.

2.3 Priority of Tasks

Priority is quite a vast concept that can inherit quite a number of attributes. Here, while evaluating priority we will mainly focus our attention on the following:

User Level, Task urgency, Task Load and Time queuing up.

According to the attributes of task specified above, we will corresponds the User Level with *WUserLevel.* Task Urgency is shown via *WPriorityExpectation.* Task Load will correspond to *WLen.* And the Time Queuing up (TQU) can be easily calculated by current time and *WSubTime.*With the introduction of TQU, priority of the task can be dynamically changed. Hence, the major issue like "Starving" in "FCFS Scheduling" can be easily handled.

Calculating priority with just single attribute is quite much easier then calculating priority with many different number of attributes. In order to calculate priority of task that includes different number of attributes together, there must be way a combine the different attributes of the task together to get a single attribute. So here, In order to calculate priority that includes the attributes specified above, we will use the concept of "Standardization" that will merge these different attributes together to form a single attribute. Standardization is really beneficial when we want to converge wide range data within a certain range. In order to standardize the priority, we will apply the following formula:

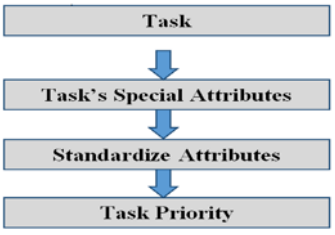
$$\boldsymbol{SDZ(i)} = \frac{\boldsymbol{SA(i) - LowestVal}}{\boldsymbol{HighestVal - LowestVal}}$$

Where SDZ(i) is the Standardization resultfor task W_i. SA_i is the special attribute value of task W(i).HighestVal and LowestVal are the highest and lowest value of special attributes respectively from all tasks.

Now we can use this standardization formula to calculate priority as:

$$\boldsymbol{PRIOR\ (i) = \alpha * SUL(i) + \beta * SPE(i) + \gamma * SLen(i) + \delta * STQU(i)}$$

Where PRIOR (i) is the calculated priority of task W(i). $\alpha, \beta, \gamma, \delta$Represents load factor of priority, and $\alpha + \beta + \gamma + \delta = 1$. Perhaps, changing load will make an impact on the priority. SUL(i), SPE(i), SLen(i), STQU(i) shows corresponding standardized values of WUserLevel, WPriorityExpectation, WLen, WSubTime respectively.



2.4 Selecting Appropriate Service

The Cloud Service Provider Basic Aim is to maximize its profit without affecting the SLA with the user. So Cloud provider always seek solution to assign tasks in such a manner that their overall execution cost of tasks gets lower. This can be achieved only if execution time of the tasks is minimized. Suppose EET be the expected Execution time for a given task W_i having length WLen on service P(z) (provided P(z) has no load when W(i) was submitted). Then:

$$\boldsymbol{EET(Wi)} = \frac{\boldsymbol{WLen}}{\boldsymbol{P(z)MIPS}}$$

Where MIPS is the speed of computer's processor elaborated as Million Instructions per Second.

Now, for cloud provider, to earn maximum profit, ETT (Wi) should be Minimum. Hence If CET is Cost of executing task on a service then,

$$\boldsymbol{CET(Wi)} = \mathbf{min}[\boldsymbol{EET(Wi)}]$$

In order to find min[EET(Wi)], The following algorithmic approach can be used which tries to find the minimum completion time of each task on different services. Moreover, certain definitions that are required to be known in advance are illustrated as follows:

*Definition 1.*ETM: stands for "EXPECTED TIME MATRIX" is the matrix that is stored on a central mapping system or scheduling system. The basic function of this matrix is that it includes expected time for execution of all tasks on all the services. The values along the row indicates the expected time for executing(ETE) a task on different services, Whereas values along the column represents the ETE of different task on same service. Value(i,z) denotes a particular ETE value for Task W(i) on service P(z).

Definition 2. STTS: stands for "SHORTEST TIME TO START" is a matrix that contains the shortest time after which the service can start again after completing the assigned tasks. Value(z) denotes the shortest time for service P(z). This matrix is also placed on the scheduling system.

*Definition 3.*STTC: stands for "SHORTEST TIME TO COMPLETE" is basically a matrix that contains the minimum level of time required by all tasks on all services. This includes ETM of all tasks and STTS of all services. The values along the row indicates the STTC of a task on different services, whereas those along the column indicates the STTC of different task on a single given service.

$$\boldsymbol{STTC(i,z)} = \boldsymbol{ETM(i,z)} + \boldsymbol{STTS(z)}$$

From the above defined definitions, it can be clearly stated that the STTC(i,z) contains the minimum time that is required by a task to complete on a particular service. Based on values in this matrix, we can choose the service which is proving to provide the service in a minimum possible time. Thus, this way we can reduce the net completion time of tasks which eventually will lower the overall cost of services and helps in achieving a well-balanced system in terms of load.

3. THE OVERALL ALGORITHM

Based on all definitions and method stated above in this paper, the complete algorithmic approach to achieve scheduling is as follows:

Algorithm 1:

Input: Task(W), Service(P);

Output: The complete scheduled list.

Start.

For given set of task Task(W)

{

Evaluate special attributes of task W_i.

Standardize the value of special attributes as described in section 2.3

Evaluate the priority of task Wi using the formula stated in section 2.3

}

Sort Task set Task(W) according to priority.

Create and initialize ETM and STTC matrix.

Evaluate STTS for each service.

Initialize mapping list. //on scheduling system to map a task onto a particular service.

For sorted set of tasks Task(W)

{

For service set Service(P)

{ Find least value for task W_i.

Save service corresponding to this value.

}

Update mapping list with task W_i onto the selected minimum time service.

Remove completed/mapped task $W_i$from task set.

}

For mapping list

{

Schedule task W_i according to the mapping list.

}

End.

4. EXPERIMENTAL RESULTS

To verify the algorithm working and its results we tested our algorithm on a simulation based platform namely CloudSim 3.0.3. Different cloud data centers as well as various users were created. Different virtual machines were made that provide services to the users. Different tasks were also created to simulate the tasks of users. The machine with CloudSim was particularly configured with Intel i3 Processors with 2.4GHz and 2 GB Memory.

Different Test has been conducted on the proposed algorithm to verify its validity. Number of Tasks were increased considerably from 20 to 100 to 400 and so on. The algorithm showed following results when used with 20 number of tasks. The Following shows the results of our scheduling algorithm.

```
size is 20
Priority calculated for 0=3.3000000000000003
Priority calculated for 1=2.8
Priority calculated for 2=3.5
Priority calculated for 3=3.5
Priority calculated for 4=2.8
Priority calculated for 5=2.7
Priority calculated for 6=2.9
Priority calculated for 7=3.0
Priority calculated for 8=2.4000000000000004
Priority calculated for 9=3.1
Priority calculated for 10=2.9
Priority calculated for 11=2.5
Priority calculated for 12=3.5
Priority calculated for 13=2.4000000000000004
Priority calculated for 14=2.5
Priority calculated for 15=2.9
Priority calculated for 16=2.9
Priority calculated for 17=2.9
Priority calculated for 18=3.1
Priority calculated for 19=2.9
```

```
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: VM #1 has been created in Datacenter #2, Host #1
0.1: Broker: Sending cloudlet 8 to VM #1
submitted one is 8
0.1: Broker: Sending cloudlet 13 to VM #1
submitted one is 13
0.1: Broker: Sending cloudlet 11 to VM #1
submitted one is 11
0.1: Broker: Sending cloudlet 14 to VM #1
submitted one is 14
0.1: Broker: Sending cloudlet 5 to VM #1
submitted one is 5
0.1: Broker: Sending cloudlet 1 to VM #1
submitted one is 1
0.1: Broker: Sending cloudlet 4 to VM #1
submitted one is 4
0.1: Broker: Sending cloudlet 6 to VM #1
submitted one is 6
0.1: Broker: Sending cloudlet 10 to VM #1
submitted one is 10
```

```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time    Priority
    8          SUCCESS        2             1       1602.46      0.1        1602.56    2.4000000000000004
   13          SUCCESS        2             1       1603.36      0.1        1603.46    2.4000000000000004
   11          SUCCESS        2             1       1603.06      0.1        1603.16    2.5
   14          SUCCESS        2             1       1603.48      0.1        1603.58    2.5
    5          SUCCESS        2             1       1601.68      0.1        1601.78    2.7
    1          SUCCESS        2             1       1600.36      0.1        1600.46    2.8
    4          SUCCESS        2             1       1601.38      0.1        1601.48    2.8
    6          SUCCESS        2             1       1601.96      0.1        1602.06    2.9
   10          SUCCESS        2             1       1602.88      0.1        1602.98    2.9
   15          SUCCESS        2             1       1603.59      0.1        1603.69    2.9
   16          SUCCESS        2             1       1603.7       0.1        1603.8     2.9
   17          SUCCESS        2             1       1603.81      0.1        1603.91    2.9
   19          SUCCESS        2             1       1603.92      0.1        1604.02    2.9
    7          SUCCESS        2             1       1602.22      0.1        1602.32    3.0
    9          SUCCESS        2             1       1602.68      0.1        1602.78    3.1
   18          SUCCESS        2             1       1603.81      0.1        1603.91    3.1
    0          SUCCESS        2             1       1599.98      0.1        1600.08    3.3000000000000003
    2          SUCCESS        2             1       1600.72      0.1        1600.82    3.5
    3          SUCCESS        2             1       1601.06      0.1        1601.16    3.5
   12          SUCCESS        2             1       1603.22      0.1        1603.32    3.5
```

5. CONCLUSION

The presented algorithm provides quite a unique way of achieving a scheduling based on priority parameter along with a decent level of load balancing and cost minimization by distributing the tasks to services that are assuring to complete those tasks in a minimum possible time. The algorithm also drives a better QoS by minimizing the net required cost as well time for the user task. Overall the algorithm plays an important part for both user as well as for the provider.

REFERENCES

[1] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandie. Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems, 2009, 25(6), p. 599-616.

[2] Syed Muhammad Ahsan. A framework for QoS computation in web service and technology selection. Computer Standards & Interfaces, 2006, 28(6), p. 714-720.

[3] BaorninXu, Chunyan Zhao, Enzhao Hua, Bin Hu. Job scheduling algorithm based on Berger model in cloud environment. Advances in Engineering Software, 2011, 42(7), p. 419-425.

[4] Xiaoyong Tang, Kenn Li, Renfa Li, BharadwajVeeravalli. Reliability-aware scheduling strategy for heterogeneous distributed computing systems. Journal of Parallel and Distributed Computing, 2010, 70(9), p. 941-952.