RESEARCH ARTICLE OPEN ACCESS

An Efficient Priority Based Load Balancing Algorithm for Cloud Environment

Harmandeep Singh Brar[1], Vivek Thapar[2]
Research Scholar[1], Assistant Professor[2],
Department of Computer Science and Engineering,
Guru Nanak Dev Engineering College,
Punjab Technical University, Ludhiana, 141006
Punjab –India

ABSTRACT

Widespread use of computers for almost every task encountered in our day-to-day lives has increased the demand for resources like computing power, storage and bandwidth. The problem was initially addressed with the provisioning of computer networks, which later lead to the development of concepts and paradigms like distributed computing, grid computing and cluster computing. A recent addition to this list has been "cloud computing", aiming to provide users with all their computing requirements through the notion of "services". But such provisioning requires sophisticated techniques to handle the workload and respond to user requests in time. One of these major issues in cloud computing is load balancing, which is the process of assigning tasks to virtual machines in such a way that neither of them either gets overwhelmed by requests or remains idle for a longer duration. Such algorithms must be efficient enough as not to waste host resources and robust enough to withstand increasing number of users. Larger the organization, greater is the cost and hence the risk involved. It is therefore better and safe to analyze the risks involved in implementing the algorithms for providing a particular level of services. This involves an evaluation of the algorithms, applications and policies. The most successful and easy method is to simulate the working environment using a simulation tool. Cloudsim is a simulation toolkit that provides suitable facilities to model and simulate different types of cloud environments. It has a provision of applying any of the three existing cloud broker policies and load balancing algorithms, and also provides facility to extend the toolkit by implementing and testing newer algorithms. This paper proposes a priority based load balancing algorithm, wherein execution length of cloudlets is input as the workload to the system and is used to assign priority to the tasks. A comparison of the execution times of cloudlets is also being made using the proposed algorithm and the existing Round Robin load balancing algorithm.

Keywords:- Cloud computing, Software as a Service, Platform as a Service, Hardware as a Service, Hypervisor, Service proximity, Round Robin, Throttled, Active monitoring, Fairness, Execution length, Real time.

I. INTRODUCTION

Cloud computing provides a way of letting users access resources like computing power, memory, applications, bandwidth and development environments in the form of services using a pay-per-use method. These services are provided by a cloud service provider using a set of protocols and service level agreements with the clients. The hosts are housed in data centres, which are maintained by the service provider. A third part, called a service broker is required in order to act as an intermediary for routing user requests to the most appropriate data centres and provide efficiency and better performance. The process of sending a user request, through processing it and returning the result to the user, involves a number of policies and algorithms at each logical layer. Brokers make use of service broker policies in order to select the most appropriate data centre for servicing a user request. Load balancers, on the other hand, make use of an efficient load balancing algorithm in order to send the request to the most suitable virtual machine. The aim here is to select a virtual machine in such a way that no virtual machine gets overwhelmed by requests, when some others may be completely free. Another task involves Virtual machine management, which is handled by the hypervisor or the Virtual Machine Manager. The job of the hypervisor regards creation, deletion, management and migration of virtual machines within the hosts. Amongst the tasks discussed above, load balancing is an important issue that needs to be addressed for greater efficiency. Selection of the most appropriate virtual machine affects completion time of the task, and hence the number of jobs performed and the cost involved. A number of load balancing algorithms have been discussed in literature, and a few have efficiently been implemented to address the issue of load balancing in cloud computing. But the best amongst these algorithms are not suitable for all of the situations encountered in our day-to-day lives. One such major area is that of real-time applications that are governed by real-time completion of the tasks. In many cases, delayed completion may result cause catastrophic effects. Therefore, an efficient load balancing algorithm is required to serve such applications. In order to test the efficiency of algorithms, using a real cloud set-up is often not feasible. The most effective alternative is to simulate the environment and model the scenario using the required algorithms. CloudSim is an extensible cloud simulation toolkit that provides facilities to model and simulate various types of cloud scenarios and also to implement newer algorithms and embed them into the toolkit. This paper presents an efficient execution length based load balancing algorithm for real time applications. Section II introduces the concept of load balancing and describes some of the best load balancing algorithms that are generally used. Section III analyses the problems with the existing algorithm and forms a basis for the new algorithm. Section IV discusses the proposed algorithm. Section V provides a comparison between the results of the existing Round Robin algorithm and the proposed algorithm. Section VI concluded the discussion while also mentioning the future perspective.

II. BACKGROUND

The process of providing computation facility to clients over the cloud begins with a user request. This request is taken as input by the cloud system in the form of workload that is calculated in Millions of Instructions per Second (MIPS). A logical model of cloud computing representing the underlying concept, beginning with the request is shown below:

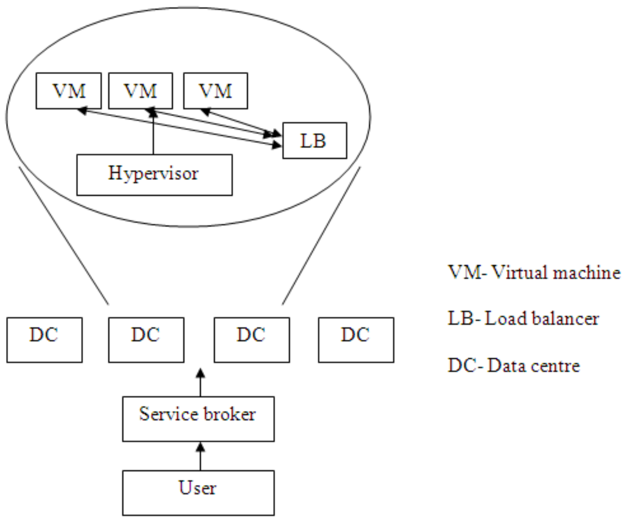


Figure 1: Logical model of cloud computing

The request from a user is directed to a suitable data centre by a service broker. Besides researching an appropriate data centre for addressing user requests, the broker may optionally provide a user interface to the client in order to hide the internal functioning of the cloud. Brokers make use of certain service broker policies (1) to perform the function of routing user requests to the selected data centres. The most popular policies include the following:

A. ***Closest data centre:*** A region proximity list is maintained in order of increasing network latency. This policy selects the data centre that is encountered earlier in the list, and hence is the closest one.

B. ***Optimal response time:*** A similar method is used to first find out the closest data centre. In addition, the current response time is found out for each data centre. If the response time calculated is one for the closest data centre, it is selected. Else, the closest data centre or the one with least response time are selected with equal probability.

C. ***Dynamically reconfigurable routing with load balancing:*** This broker policy (2) involves the use of current load for scaling the application deployment. The number of virtual machines can be increased and decreased accordingly.

After selection of a data centre, the workload needs to be assigned to a suitable virtual machine. Load balancing algorithms must select virtual machines in such a way that neither of the virtual machine remains idle for long. Also, none must be overwhelmed by a greater number of requests than it is actually capable of handling. Some popular load balancing strategies include the following:

A. ***Round Robin:*** Round Robin strategy employs definite time slices for allocating nodes in order to address user requests, in a round robin fashion, according to a locally maintained process allocation order. This algorithm serves a very fast response in case the workload has equally been distributed amongst processes (3).

B. ***Throttled:*** Throttled load balancing policy (4) maintains an index of virtual machines and the state of each virtual machine, either BUSY or AVAILABLE. The allocation table is parsed until the first AVAILABLE virtual machine is found. The task is thus assigned to that virtual machine and the table is updated. The process is then continued for all other tasks.

C. ***Active monitoring/Equally spread technique:*** This policy is based on maintaining equal workload. An index that contains the number of requests that are currently allocated to a virtual machine is created, along with the index of virtual machines. The virtual machine that is the least loaded is identified. If more than one such virtual machine is identified, the first one is selected. The allocation table is then updated accordingly.

The hypervisor is responsible for managing virtual machines in the cloud environment. Its job involves creation, deletion, migration and management of virtual machines among the hosts. Therefore, it is also called Virtual Machine Manager (VMM) (5).

III. ANALYSING THE PROBLEM

The existing literature on cloud computing shows that there are a number of algorithms that can be used for load balancing in different scenarios, depending upon the conditions or criteria of usage or the nature of service being provided. But none of them so far considers real time applications. The proposed algorithm performs efficient load balancing in case of time-critical applications.

Round Robin algorithm, the best known load balancing algorithm maintains a list of tasks and serves the tasks in that particular order by making use of definite time slices (6). This solves the problem of starvation by processing each task for that stipulated time period. But the use of this algorithm is restricted to those situations that do not follow a strict time period of completion, i.e., where hard bound completion of tasks is not the major governing principle. So, it can be used only for the general basic-purpose applications. But it cannot be applied to situations that do not afford to miss their completion deadline even by fractions of a second. Round Robin algorithm has the following disadvantages (7):

A. ***No priority:*** Priority cannot be assigned to tasks using Round Robin algorithm, which is a problem when dealing with time critical applications.
B. ***Unequal task lengths:*** Allocation of equal time slices is never suitable for tasks with a medium length, which may have to wait much longer than they would actually have to wait using any other algorithm.
C. ***Perceived fairness:*** Users often wish to have they job completed once their task gets in front in the processing queue, which is not possible using Round Robin load balancing.

Considering all the above problems and execution scenarios, we find out that Round Robin cannot be used in all the situations and hence some efficient load balancing algorithm is needed to solve these problems in case of real time applications.

IV. PROPOSED ALGORITHM

The algorithm being proposed mainly focuses on assigning tasks to virtual machines in a way so that the most important tasks can be completed earlier, without having to wait for some other tasks of lesser importance, which is done based on execution length. This is the most suitable case for time-critical applications wherein a little delay in time can have disastrous effects. The aim here is to balance the workloads among virtual machines in such a way that they are utilized efficiently. The algorithm takes as input workload in the form of cloudlets. Cloudlets are characterized by a definite execution length, which is used in the algorithm to perform a length wise sort and then to perform the execution based on this sorted list. The algorithm involves the following series of steps:

```
Input:   Cloudlet(s)
Output: Result

    function STF

        for each cloudlet do
            $\sum_N$ [ t(x) := getLengthBasedPriority(cloudlet(x)) ]
        end for

    sort_func (t)

$\sum_N$ Result (k) := execute (t)

    end function
```

The function STF initially fetches the execution length of each of the cloudlets and then calculates the priority of the task based on increasing length of cloudlets. It then carries out a sorting function based on these priority values. Later, the list that has been sorted according to increasing execution lengths of cloudlets is executed using the defined procedures and the results are stored and displayed to the user. Any type of priority resolution is carried out using the default provision of the First Come First Serve algorithm.

The sorting function works out by taking as input the cloudlet execution length. The sort is performed length wise by comparing the execution lengths and performing a swap of lengths in the list in case the greater one lies earlier in the list. The sorting function has been mentioned below:

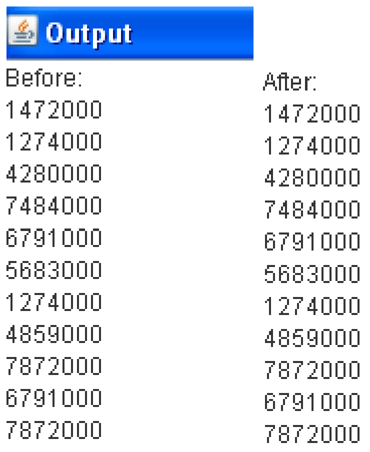
```
Input:   cloudlet_length
Output: sorted_length

    function sort_func(t)

        for i = (n-1) to 1
            for j = 0 to (i-1)
                if t[j] < t[j+1]
                    swap ( t[j] , t[j+1] )
                end if
            end for
        end for

    end function
```

The CloudSim simulation toolkit has been used to implement the proposed algorithm and improved results have been obtained, which are being discussed and compared with the existing Round Robin algorithm in the next section.

V. RESULTS AND DISCUSSION

CloudSim, as an effective simulation toolkit, provides the entire necessary execution environment for modelling and simulating the cloud infrastructure and services, and for the evaluation of various resource provisioning and load balancing algorithms. It provides a virtualization engine for creating and managing virtualized services (8). This gives a path to switch flexibly from time and space shared core allocation to virtualized services. All the available powerful features of CloudSim can be extended to model other cloud environments that involve custom resource and load distribution and allocation. Here, CloudSim has been extended to implement the proposed algorithm. A graphical interface is initially provided to select from the both implementations of load balancing algorithms, i.e., the existing Round Robin algorithm and the proposed temporal load balancing algorithm. Number of cloudlets to be executed is input along with the selection of the algorithm. Workload in the form of cloudlets' execution length is chosen as selected from the Planet Lab real time data. In order to resolve execution of cloudlets of the same execution length, the in-built First Come First Serve resolution policy is used. The simulation results as configured for a set of 11 cloudlets, for both of the algorithms have been shown below:

Simulation A: Execution of 11 Cloudlets using default Round Robin algorithm.

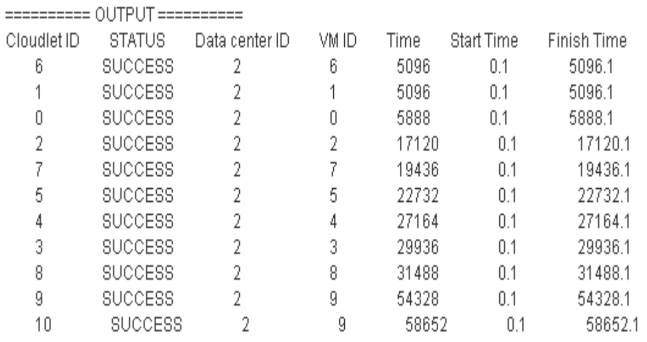


```
Output
Before:          After:
1472000          1472000
1274000          1274000
4280000          4280000
7484000          7484000
6791000          6791000
5683000          5683000
1274000          1274000
4859000          4859000
7872000          7872000
6791000          6791000
7872000          7872000
```

Figure 2: Order of occurrence of cloudlets using Round Robin algorithm

The output in figure 2 shows that the order of occurrence of the cloudlets remains unaffected. This is because Round Robin algorithm provides time slices for execution in the same order as decided in its list, here, in the First Come First Serve basis. No strict length-wise ordering of cloudlets persists in this case.

Figure 3 below shows the output after execution of these 11 cloudlets:



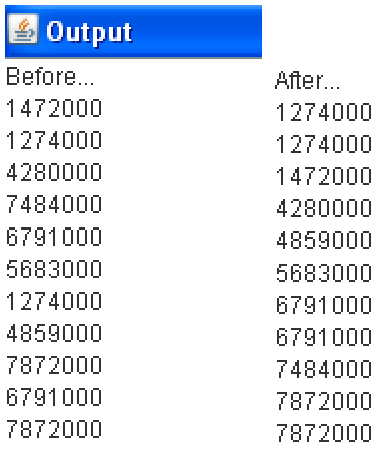
```
========== OUTPUT ==========
Cloudlet ID   STATUS    Data center ID   VM ID   Time    Start Time   Finish Time
    6         SUCCESS        2             6     5096       0.1        5096.1
    1         SUCCESS        2             1     5096       0.1        5096.1
    0         SUCCESS        2             0     5888       0.1        5888.1
    2         SUCCESS        2             2     17120      0.1        17120.1
    7         SUCCESS        2             7     19436      0.1        19436.1
    5         SUCCESS        2             5     22732      0.1        22732.1
    4         SUCCESS        2             4     27164      0.1        27164.1
    3         SUCCESS        2             3     29936      0.1        29936.1
    8         SUCCESS        2             8     31488      0.1        31488.1
    9         SUCCESS        2             9     54328      0.1        54328.1
    10        SUCCESS        2             9     58652      0.1        58652.1
```

Figure 3: Simulation output using Round Robin algorithm

As clear from the figure above, 11 cloudlets are run on data centre 2, providing a distinct virtual machine to each cloudlet. The cloudlet ids and the virtual machine ids, to which the cloudlets have been bound, are displayed. The start, finish and total execution time are displayed in milliseconds.

The simulation performed above is based on the default Round Robin algorithm that is most frequently used by the cloud service provider in order to perform load balancing at the data centres. As all the virtual machines are allocated initially whenever a burst of workloads arrives, the start time of each of the cloudlets under execution is the same.

Simulation B: Execution of 11 Cloudlets using proposed algorithm.

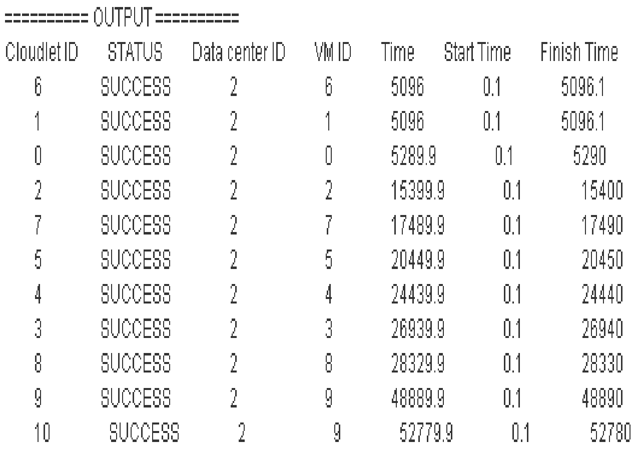


```
Output
Before...        After...
1472000          1274000
1274000          1274000
4280000          1472000
7484000          4280000
6791000          4859000
5683000          5683000
1274000          6791000
4859000          6791000
7872000          7484000
6791000          7872000
7872000          7872000
```

Figure 4: Order of occurrence of cloudlets using proposed algorithm

The output in figure 4 shows that the cloudlets have been arranged according to increasing execution lengths. Two cloudlets of the same length that were initially at their position as encountered during selection later occur at consecutive positions.

Figure 5 below shows the output after execution of these 11 cloudlets. As visible from the results, 11 cloudlets are being run on data centre 2, with each of them being provided a distinct virtual machine. The start, finish and total execution time are displayed in milliseconds.



```
========== OUTPUT ==========
Cloudlet ID   STATUS    Data center ID   VM ID   Time      Start Time   Finish Time
    6         SUCCESS        2             6     5096         0.1        5096.1
    1         SUCCESS        2             1     5096         0.1        5096.1
    0         SUCCESS        2             0     5289.9       0.1        5290
    2         SUCCESS        2             2     15399.9      0.1        15400
    7         SUCCESS        2             7     17489.9      0.1        17490
    5         SUCCESS        2             5     20449.9      0.1        20450
    4         SUCCESS        2             4     24439.9      0.1        24440
    3         SUCCESS        2             3     26939.9      0.1        26940
    8         SUCCESS        2             8     28329.9      0.1        28330
    9         SUCCESS        2             9     48889.9      0.1        48890
    10        SUCCESS        2             9     52779.9      0.1        52780
```

Figure 5: Simulation output using proposed algorithm

All the results displayed above provide a clear understanding and distinction of the ordering of cloudlets as well as the total execution time in both the algorithms. The results inferred from these simulations are mentioned below:

A. ***Sorted cloudlets:*** As Round Robin algorithm does not maintain any sorted list of cloudlets, the cloudlets that are of equal length do not necessarily get executed together. While the proposed algorithm serves such workloads consecutively in a First Come First Serve manner.

B. ***Priority:*** As there is a provision of priority according to execution lengths, it is most suitable for real time applications, where execution of the task is strictly governed by its being completed at the earliest.

C. ***Unequal task lengths:*** Workloads that are normally encountered are of different sizes. So, whereas Round Robin is unable to provide fairness to medium-sized tasks, the proposed algorithm has an edge due to a strict ordering according to lengths.

D. ***Execution time:*** The execution time for the proposed algorithm is far better than the existing Round Robin algorithm because longer time slices allotted to shorter length tasks that were wasted in Round Robin algorithm are not an issue in the proposed algorithm.

E. ***Perceived fairness:*** As the entire task is completely executed once its turn in the queue arrives, the users of the service are satisfied with the level of fairness provided by the proposed algorithm, which was not possible in the existing algorithm.

F. ***Real time applicability:*** All the above benefits provided by the proposed algorithm make it suitable for real time applications, which cannot afford to wait for such longer time periods as in Round Robin algorithm; due to time slices.

VI. CONCLUSION AND FUTURE SCOPE

Real time applications form an important part of the daily activities, be it general ones like video conferencing or the specific purpose ones like the air traffic control. Efficiently addressing all such applications decides the success of solving all the major human problems. As most of the computing tasks are being shifted to the cloud, real time applications over the cloud form a major part of the migration issue. As all such applications rely on strict time bounds of completion, efficient algorithms are required that are able to address such tasks in a distributed environment. Considering load balancing as the most important issue, a temporal load balancing algorithm for real time cloud applications has been proposed, which executes the tasks in an order of increasing execution lengths of the workloads. Results show that it is quite advantageous to use this algorithm for real time applications as it shows a great improvement in execution time of the cloudlets.

The future scope of this study involves the development of some dynamic priority based algorithm, wherein priority is decided dynamically, rather than deciding it in the beginning according to a pre-specified criterion.

REFERENCES

[1] A Proposed Service Broker Strategy in CloudAnalyst for Cost-Effective Data Center Selection. Limbani, Dhaval and Oza, Bhavesh. 1, s.l. : International Journal of Engineering Research and Applications, Jan-Feb 2012, International Journal of Engineering Research and Applications, Vol. 2. ISSN: 2248-9622.

[2] Modeling Local Broker Policy Based on Workload Profile in Network Cloud. Sandhu, Amandeep and Kaur, Maninder. 8, Banur : s.n., August 2013, International Journal of Science and Research, Vol. 2, p. 5. ISSN: 2319-7064.

[3] A Survey of Load Balancing Algorithms in Cloud Computing Al. Brar, Harmandeep Singh, Thapar, Vivek and Kishor, Kunal. 3, Ludhiana : s.n., June 2014, International Journal of Computer Science Trends and Technology, Vol. 2, p. 4. ISSN: 2347-8578.

[4] Wickremasinghe, Bhathiya. CloudAnalyst: A CloudSim-based Tool for Modelling and Analysis of Large Scale Cloud Computing Environments. CSSE department, University of Melbourne. Melbourne : s.n., 2009. p. 44, MEDC Project Report.

[5] VMWare. Virtualization overview. White paper. Palo Alto : VMWare. p. 11.

[6] Load Balancing On Cloud Data Centres. Mahalle, Hemant S, Kaveri, Parag R and Chavan, Vinay. 1, 2011, International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3. ISSN: 2277 128X.

[7] Wayne, Jake. Azcentral. [Online] [Cited: 26 July 2014.] http://yourbusiness.azcentral.com.

[8] Calheiros, Rodrigo N, et al. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Melbourne, Australia : Wiley Online Library, Wiley Online Library, 2010. DOI: 10.1002/spe.995.