A Novel Predictive Resource Allocation Framework for Cloud Computing

R.Rengasamy[1] M.Chidambaram[2]

[1] *Research Scholar, A.V.V.M Sri Pushpam College, Poondi,Thanjavur, Tamilnadu, India*
[2] *Department of Computer Science, Rajah Serfoji Government College, Thanjavur,Tamilnadu, India*
e-mail: rrsamy74@gmail.com

***Abstract*:** Cloud computing is a new paradigm in which computing is offered as services rather than physical products. Computing services are provided by third-party service providers which offer consumers affordable and flexible computing services via shared resources. Cloud providers offer different layers/levels of services to consumers over many types of application domains. Server consolidation is a technique for reducing operating costs of computer resources in virtualization. These expenses can lead to increasement of financial costs of servers, power consumption of servers, data-center cooling systems and labour costs. It is inefficient if there are under-utilized servers that have more space and consume more resources. The proposed method meets the requirements for effective resource allocation for cloud computing. The effectiveness of the proposed method is evaluated and reveals higher performance in cloud computing.

***Keywords* ---** *Cloud computing, Resource allocation, Workload prediction*

1. Introduction

Virtualization is the abstraction of computing resources such as CPU, storage, memory, network, from applications and users which make shared resource available and accessible to any authorized users and applications [1]. Specifically, virtualization creates simulated virtual machines for guest software to run on as if physical servers. The guest software can be applications, databases, services, or even operating systems [2].

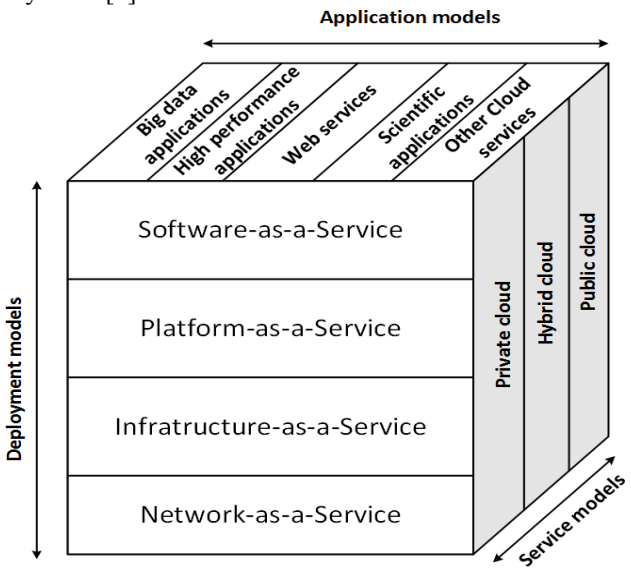

Figure 1: Taxonomies of service models, networks over the past few years

Figure 1 shows the taxonomies of service models. Server consolidation is to combine the workload (e.g., virtual machines) of different servers and assign them to the common set of target servers [3][4]. Server consolidation is enabled by the ability to migrate virtual machines from one server to Hyper-visor technique. It is often employed to create and manage virtual machines such that physical resources are allocated or adjusted dynamically as well as allow many virtual machines to be hosted by on the same physical server [5].

2. Technical Background

Software as a Service is the top layer of cloud computing services where software applications mainly standard software is offered as a cloud service to the users[6]. An outstanding example of a SaaS service is Google Docs. Google docs offer a free fully functional word processor, the spread sheet application, and presentation creator software enabling users to collaborate with each other from different locations[7]. If users need to develop their own application on the cloud, they must use Platform as a Service (PaaS)[8].

This platform provides a cloud service environment in which developers can use appropriate APIs to make an application such as Facebook, which can be run and shared in anywhere in the world with any platforms without the risk of software pirating[9]. Infrastructure as a Service segment of cloud services provide developing tools with limitless storage and computing powers to developers and ordinary users [10]. For example, Google drive and Apple iCloud offer cloud storage service for all people including ordinary users and developers [11].

2.1 Resource Allocation Problem

A simple example of multi-objective resource allocation problems is the assignment problem in the distributed client-server Environment. Assume that there are M users that need to be assigned to N servers. Let X is the optimization variable which is a matrix of M ×N whose elements represents clients to be assigned to servers. The objective function is to simultaneously minimize two quantities: the inter-server

978-1-5386-9533-3/19/$31.00 ©2019 IEEE

communication load I(X) and the imbalance factor ΔL(X).

$$\text{Minimize: } \{I(X), \Delta L(X)\} \quad (1)$$

$$\text{Subject to: } X \in \{0, 1\}^{M \times N} \quad (2)$$

$$XR \quad c \quad (3)$$

$$X1 = 1 \quad (4)$$

The constraint (1) and (2) represents the assignment decision, or the domain of the optimization variable $X = \{0, 1\}^{M \times N}$, the inequality constraint (2), i.e., $g(X) = XR$, represents no server exceeds its capacity, where R and c are the client resource demand vector and the server capacity vector respectively, and the equality constraint (3), i.e., $h(X) = X1$, represents the condition of each client has to be assigned to one and only one server. By applying the scalarization method, and weighting the two objectives equally, the global objective function can be rewritten as $0.5I(X) + 0.5\Delta L(X)$. The resource allocation problems considered in this research work will be investigated further in future work.

3. Proposed Framework

The core contribution of this paper is a new predictive load balancing of running tasks, for the purpose of resource allocation [12][13]. Predictive workload balancing enables cloud service providers to prepare their resource allocation for all different scenarios beforehand of any events[14][15]. We will call the algorithm of allocating resources based on Cicada predictions C-Rule algorithm. Cloud resource provisioning and power consumption of data centers and its efficiency has been proven in previous research papers. C-Rule algorithm first predicts workloads during the early stage by a predictor called Cicada. Then, Cicada uses CloudSim framework to simulate the workload balancing by our rule-based algorithm. C-Rule Algorithm focuses on preventing over-loads in a first place rather than balancing current over-loads. In this new approach, a prediction can be achieved in a less than 20 milliseconds and with a help of a Cloud simulator, an overload can be in a matter of seconds. If C- Rule algorithm detects any over-loads, CloudSim can find the most accurate resource allocation in a matter of seconds which is faster than all previous algorithms. Resource allocation with a CloudSim requires less computational power than using complex statistical and mathematical formulas for resource allocation. C-Rule algorithm can achieve the most efficient cloud resource allocation which includes number of host machines and the required number of virtual machines for each host machine with minimal resources. After finding the most system configuration for a specific workload, C-Rule algorithm will lower number of virtual machines and amount of physical memory for every given task, up until it finds the minimum resource requirement for a specific workload. C-Rule algorithm needs to receive efficient prediction data and if there are no historical prediction data then CloudSim will use the random algorithm for workload balancing until it receives a reliable workload data. Previous researches have proven that Random algorithm is the most efficient algorithm for peak time traffic and when Cicada cannot detect a reliable prediction, random algorithm can distribute the workload evenly between different VMs. The main purpose of a predictive workload balancing with C-Rule is that, cloud service providers can install SFlow-enabled devices on their cloud network and gather workload data from a traffic link of their cloud network and use C-Rule to simulate the workload on a simulated network based on a specific workload and later increase the amount of workload to test the maximum handling of their workload. This method of the provisioning can also prevent any over-provisioning by finding the minimum amount of computing resources for a workload.

Since our proposed algorithm involves reassigning users at every iteration and a main component of the objective function is the total system load, we will consider how the total system load changes when a user u is moved from server i to server j. At time t, if user u is moved from server i to server j, then there are changes in load for servers i and j only; the loads of another server remain the same. Specifically, load at servers at time t + 1 will be:

• The new load in server i is:

$$Si(t + 1) = Si(t) - \sum AulXts(t) \quad (5)$$

• The new load in server j is:

$$Sj(t + 1) = Sj(t) + \sum AulXls(t) \quad (6)$$

• The load in any server s = i, j does not change: $Ss(t + 1) = Ss(t)$ (7)

• Consequently, the new total system load is:

$$S(t + 1) = S(t) + \sum Aul(Xli(t) - Xl\,j(t)) \quad (8)$$

The first assertion in (5) is true because the term with the double sum represents the load that user u incurred on server i due to its communication with other users in other servers. Note that after u is moved from server i to server j, the amount of load incurred by u on server i due its communication with the users assigned to server i remain the same before and after u is moved. Therefore, i is excluded from the index in the outer sum in (6). The minus sign reflects the amount of load taken away from server i. By similar reasoning, the second assertion in (7) is true. The only difference is that the sign in the double sum term is positive, reflecting a load increase for server j. The third assertion in (8) is true because for a server $s \in \{i, j\}$, the number of messages that it sends and receives, remain the same before and after the move. The fourth assertion in (8) is true by summing the changes in (5) and (6). In addition to the total load, we also need to consider the load balance, or more precisely the system load imbalance.

All predictions are transmitted from Cicada to Cloud. CloudSim will run a simulation and detect any possible overloads as shown in figure 2.

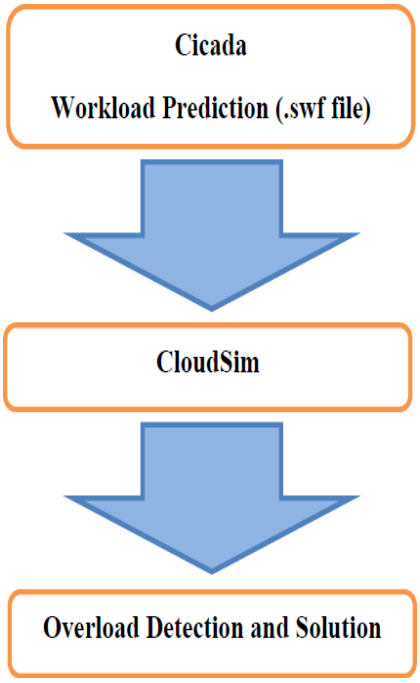

Figure 2: Architecture of overload detection and solution

If the simulation detects any overloads, it will simulate different scenarios by adding more host machines or virtual machines to achieve zero CPU waiting time. Cloud simulation can also generate list of CPUs wait times for each cloudlet each time that CloudSim adds or removes different cloud resources. All waiting times can stored as set and compared by Paired t-test to the previous data set of CPY waiting times. In the following example the number of host machines has been increased from 1 host to 2 hosts. The sum of the waiting times has been decreased from 200020.38 to 40003.75. To make a statistical comparison, all waiting times will be added into a list and will be compared by Paired t-test.

In the following example the final values for t is 6.98 which indicates there has been a significant change as shown in Table.I.

Table I: Table of results from CloudSim. Adding a new host machine decreases the total wait-time.

Total # of Virtual Machines: 19 Total Number of Host Machines 1 Hosts				19 2 Hosts			
CloudletID	STATUS	VmID	WaitTime	CloudletID	STATUS	VmID	WaitTime
3	Success	4	0.00	7	Success	8	0.00
1	Success	2	0.00	3	Success	4	0.00
0	Success	1	0.00	1	Success	2	0.00
2	Success	3	0.00	5	Success	6	0.00
7	Success	4	5000.06	11	Success	12	0.00
5	Success	2	5000.62	2	Success	3	0.00
4	Success	1	5000.75	8	Success	9	0.00
6	Success	3	5000.86	0	Success	1	0.00
9	Success	2	10000.88	9	Success	10	5000.24
11	Success	4	10000.77	6	Success	7	5000.13
10	Success	3	10000.99	4	Success	5	5000.24
8	Success	1	10000.99	10	Success	11	5000.94
13	Success	2	15000.91	19	Success	8	5000.94
14	Success	3	15001.69	15	Success	4	5000.49
15	Success	4	15001.55	17	Success	6	5000.94
12	Success	1	15001.95	13	Success	2	5000.84
17	Success	2	20001.13	18	Success	7	10000.46
18	Success	3	20001.91	14	Success	3	10001.53
19	Success	4	20002.43	12	Success	1	10001.31
16	Success	1	20002.88	16	Success	5	10001.53

3.1. C-Rule Workload balancing diagram

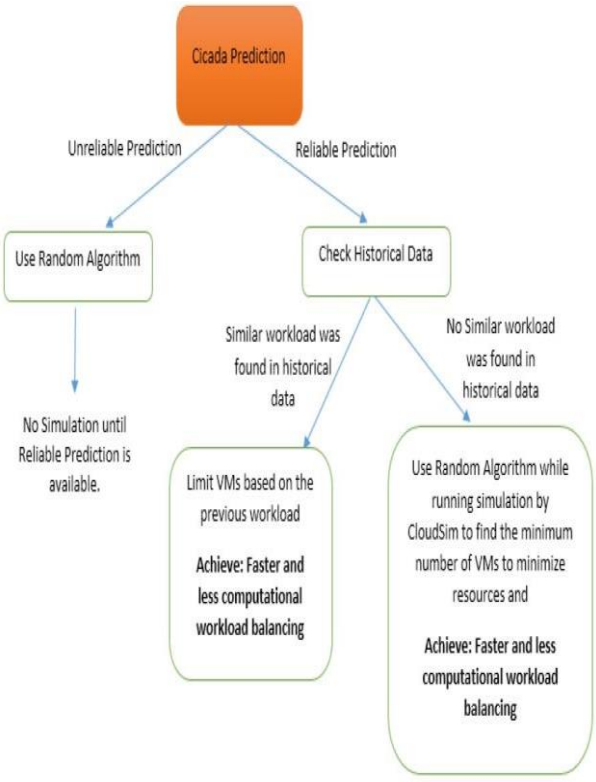

Figure 3: Architecture of the Workload Balancing Algorithm

The above diagram demonstrates the overall concept of C-Rule algorithm as shown in figure 3. Initially, Cicada generates a load prediction and if the prediction is unreliable, then it will use the random algorithm for

load balancing until it receives a reliable prediction. Random Algorithm connects cloudlets and servers randomly by assigning random numbers to each server. Unlike Round Robin algorithm, Random

algorithm can handle many requests and evenly distribute the workload to each node. Like the Round Robin algorithm, another advantage of the Random algorithm is that it is enough for machines with similar Ram and CPU specifications. The Random algorithm is the most efficient algorithm for peak time traffic and when Cicada cannot detect a reliable prediction, The Random algorithm can distribute the workload evenly between different VMs.

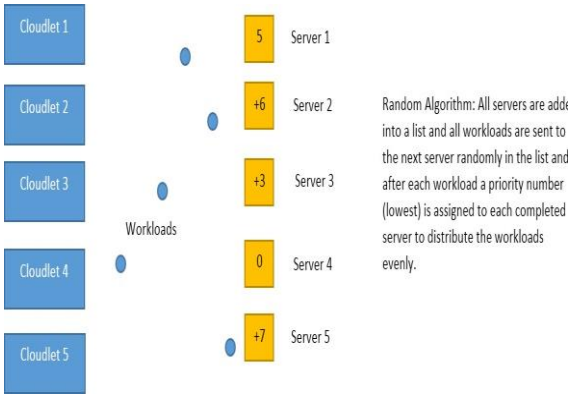

Figure 4: Random Algorithm

In the C-Rule algorithm, any unreliable prediction can be handled with the random algorithm and when a reliable workload arrives, the C-Rule algorithm can compare the incoming workload with historical workloads as shown in figure 4. If a similar historical workload is detected, the C-Rule can detect whether current resource management policy is suitable for that specific workload. If C-Rule algorithm does not find any historical data related to the incoming workload then it will begin simulating the workload in the CloudSim simulator. Initially, C-Rule will find the optimal number for physical machines and virtual machines to achieve minimum CPU waiting time. The ideal CPU waiting time is always zero. Each time C-rule algorithm can also use the paired t-test to compare the new result to the previous one.

4. Result and Evaluation

Before introduced load-balancing algorithms, C-Rule Algorithm focuses on preventing over-loads in a first place rather than balancing current over-loads. C-Rule Algorithm can find a solution for any workloads in a fraction of a second. In most cases, Cicada can make a prediction in less than 25 milliseconds and it needs a minimum of only 1 hour of historical data to make a prediction. In some cases, the speed of predictions is less than 5 milliseconds. The figure below demonstrates the speed of Cicada's prediction based on the size of the Dataset as shown in figure 5.

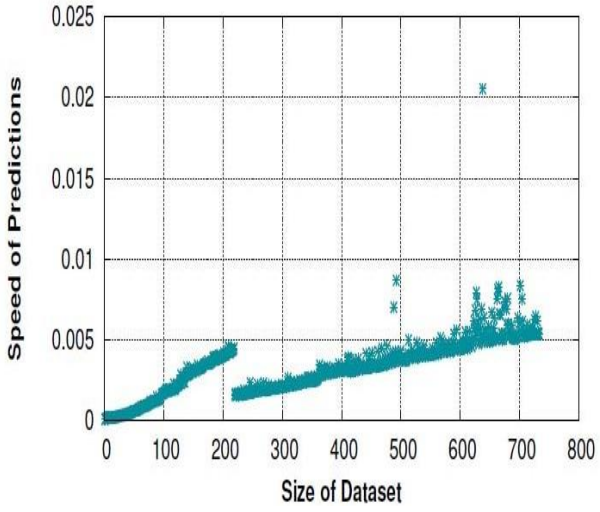

Figure 5: Speed of Predictions based on dataset

CloudSim can also simulate a workload of less than a second depending on the processing power of the centralized server. All previously introduced algorithms need complicated mathematical and statistical computation and demand a very high computational power, where the C-Rule algorithm can require a very small processing power. After achieving the minimum CPU waiting time, C-Rule algorithm will reduce the amount of resources in the simulations to find the minimum number of required resources for that workload to prevent any over-provisioning.

Table II. Final result of resource reduction after achieving a zero processing wait-time.

Total # of Virtual Machines: Total Number of Host Machines			20 7 Hosts	20 5 Hosts			
CloudletID	STATUS	VmID	WaitTime	CloudletID	STATUS	VmID	WaitTime
3	Success	9	0.00	7	Success	8	0.00
1	Success	14	0.00	3	Success	4	0.00
0	Success	15	0.00	1	Success	2	0.00
2	Success	12	0.00	5	Success	6	0.00
7	Success	5	0.00	11	Success	12	0.00
5	Success	1	0.00	2	Success	3	0.00
4	Success	11	0.00	8	Success	9	0.00
6	Success	3	0.00	0	Success	1	0.00
9	Success	8	0.00	9	Success	10	0.00
11	Success	13	0.00	6	Success	7	0.00
10	Success	10	0.00	4	Success	5	0.00
8	Success	17	0.00	10	Success	11	0.00
13	Success	18	0.00	19	Success	8	0.00
14	Success	20	0.00	15	Success	4	0.00
15	Success	6	0.00	17	Success	6	0.00
12	Success	7	0.00	13	Success	2	0.00
17	Success	4	0.00	18	Success	7	0.00
18	Success	16	0.00	14	Success	3	0.00
19	Success	2	0.00	12	Success	1	0.00
16	Success	19	0.00	16	Success	5	0.00

The above chart demonstrates a simulation result for host reduction in a successful load balancing as shown in Table II. CPU waiting time is zero whether service provider uses 7 host machines or 5 host machines.

5. Conclusion and Future Work

Unlike other algorithms Random Algorithm connects cloudlets and servers randomly by assigning random numbers to each server and can handle large number of requests and evenly distribute the workload to each node. In a load balancing dependent on the Random algorithm each client can be given list of available servers which can eliminate the need for a centralized broker. Most of current solutions solve the VM placement and migration separately. However, the later, i.e., the migration problem, might affect the optimality of the former, i.e., the placement problem, if they are optimized in sequence. Additionally, cloud providers also offer on-demand services, i.e., VMs might be deployed or re-deployed frequently The proposed algorithm can balance a workload in a matter of seconds rather than several minutes.

References

[1] L. Hyafil and R.L. Rivest. Graph Partitioning and Constructing Optimal Decision Trees are Polynomial Complete Problems. Rapport de Recherche no. IRIA laboria, Institut de Recherche d'Informatique et d'Automatique, 1973.

[2] Yichuan Jiang, Yifeng Zhou, and Wanyuan Wang. Task allocation for undependable multiagent systems in social networks. Parallel and Distributed Systems, IEEE Transactions on, 24(8):1671–1681, 2013.

[3] Jon Kleinberg. The small-world phenomenon: An algorithmic perspective. In Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing, STOC '00, pages 163–170, New York, NY, USA, 2000. ACM.

[4] Lun Li, David Alderson, John C Doyle, and Walter Willinger. Towards a theory of scale-free graphs: Definition, properties, and implications. Internet Mathematics, 2(4):431–523, 2005.

[5] R.Udendhran. A hybrid approach to enhance data security in cloud storage.Proceeding ICC '17 Proceedings of the Second International Conference on Internet of things, Data and Cloud Computing, Cambridge University, United Kingdom — March 22 - 23, 2017 ACM ISBN: 978-1-4503-4774-7 doi10.1145/3018896.3025138

[6] engasamy, M.Chidambaram, "Challenges and Opportunities of Resource-Aware Allocation Frameworks for Big data tools in Cloud Computing", International Journal of Computer Sciences and Engineering, Vol.6, Issue.12, pp.99-105,2018.

[7] Xiaoqiao Meng, Vasileios Pappas, and Li Zhang. Improving the scalability of data center networks with traffic-aware virtual machine placement. In INFOCOM, 2010 Proceedings IEEE, pages 1–9, 2010

[8] Issawi, S. F., Halees, A. A., & Radi, M. (2015). An efficient adaptive load-balancing algorithm for cloud computing under bursty workloads. Engineering, Technology, & Applied Science Research, 5(3), 795-800.

[9] Jena, S. R., & Ahmad, Z. (2013). Response time minimization of different load balancing algorithms in cloud computing environment. International Journal of Computer Applications, 69(17), 22-27.

[10] LaCurts, K. L. (2014, June). Application workload prediction and placement in cloud computing systems (Unpublished doctoral dissertation). Massachusetts Institute of Technology, Cambridge Massachusetts.

[11] Lee, R., & Jeng, B. (2011). Load-balancing tactics in cloud. In Proceedings of the International Conference on Cyber-Enabled Distributed Computing and Knowledge Cyber Discovery, pp. 447-454.

[12] Mahmood, Z. (2011). Cloud computing: characteristics and deployment approaches. In the 11th IEEE International Conference on Computer and Information Technology, pp. 121-126.

[13] Mathur, S., Larji, A. A., & Goyal, A. (2017). Static load balancing using SA Max-Min algorithm. International Journal for Research in Applied Science & Engineering Technology, 5(4), 1886-1893.

[14] Nae, V., Prodan, R., & Fahringer, T. (2010, October). Cost-efficient hosting and load balancing of massively multiplayer online games. In the 11th IEEE/ACM International Conference on Grid Computing (GRID), Brussels, Belgium, pp. 9-16.

[15] Nema, R., & Edwin, S. T. (2016). A new efficient balancing algorithm for a cloud computing environment. International Journal of Latest Research in Engineering and Technology, 2(2), 69-75.

.