Service Request Scheduling based on Quantification Principle using Conjoint Analysis and Z-score in Cloud

R. Arokia Paul Rajan
Department of Computer Science, Christ University, India

Article Info

Article history:

Received Oct 15, 2017
Revised Dec 28, 2017
Accepted Jan 8, 2018

Keyword:

Cloud computing
Conjoint analysis
Service request scheduling
Scheduling principles
Z-score

ABSTRACT

Service request scheduling has a major impact on the performance of the service processing design in a large-scale distributed computing environment like cloud systems. It is desirable to have a service request scheduling principle that evenly distributes the workload among the servers, according to their capacities. The capacities of the servers are termed high or low relative to one another. Therefore, there is a need to quantify the server capacity to overcome this subjective assessment. Subsequently, a method to split and distribute the service requests based on this quantified server capacity is also needed. The novelty of this research paper is to address these requirements by devising a service request scheduling principle for a heterogeneous distributed system using appropriate statistical methods, namely Conjoint analysis and Z-score. Suitable experiments were conducted and the experimental results show considerable improvement in the performance of the designed service request scheduling principle compared to a few other existing principles. Areas of further improvement have also been identified and presented.

Corresponding Author:

R. Arokia Paul Rajan,
Department of Computer Science,
Christ University,
Bengaluru, Karnataka – 560 029, India.
Email: arokia.rajan@christuniversity.in

1. INTRODUCTION

Distributed system contains clustered and networked heterogeneous hardware and software working together by passing messages. A distributed system is a software system in which there are several autonomous computational entities called servers. The heterogeneous distributed computing environment refers to systems that use more than one type of computer usually incorporating specialized processing capabilities to handle particular service requests [1]. A resource manager is a key component of distributed resource management whose job is to determine the best method to manage the resources of the servers. Service request scheduler of the resource manager assimilates requests as a batch and assigns these requests to the servers by adopting a suitable request scheduling technique. Among the categories of request scheduling principles, weighted nodes scheduling distribute the incoming requests across the servers using pre-assigned or computed weight for each server. In a heterogeneous distributed environment with limited resources, the request scheduling has to be sophisticated, so as to work under strict constraints. This necessitates a request scheduling principle that is flexible to adapt to the constraints [2].

In distributed computing like cloud systems, scheduling is the technique by which a request submitted by the user is assigned to resources that complete the work. A scheduler performs the scheduling activity. A request scheduler is a computer application for the batch processing service requests [3]. Among the category of different service request scheduling principles, weighted nodes scheduling principles distributes the incoming requests across the cluster of servers using a pre-assigned or computed weight for each server [4]. The capacities of the available servers can be defined by weighing the servers based on its attributes. Based on this weight, the number of requests a server should receive relative to other servers is computed [5]. Figure 1 represents the step by step process of how a request is processed in a large scale computing environment like cloud systems. The objective of this paper is to design an efficient request scheduling technique.

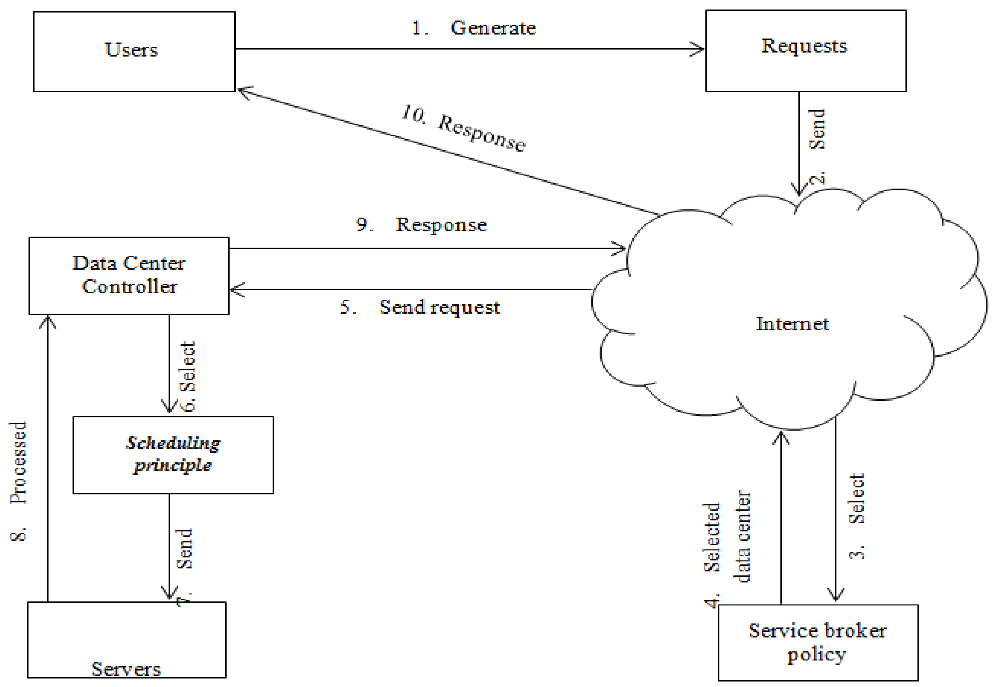


Figure 1. Request scheduling in Cloud architecture

The rest of this paper is organized as follows: Section 2 presents the related works. Section 3 presents the research problem with the assumptions. Section 4 introduces the designed request scheduling principle in detail. Section 5 presents the experimental results and the performance evaluation comparing with few existing scheduling principles. Section 6 presents the conclusion and the future extendable work of this research paper.

2. RELATED WORKS

The following literature survey presents the contributions that are influential in this research work based on design, principles, parameters and metrics:

In round-robin principle, the scheduler assigns the requests to a list of the servers on a circular basis. The first request is allocated to a server picked randomly from the group so that if more than one scheduler arrives simultaneously, not all of these requests go to the same server. Throttled load balancing algorithm is implemented with a Throttled Load Balancer to monitor the loads on each Virtual Machine (VM). It ensures only a pre-defined number of Internet Cloudlets are allocated to a single VM at any given time. Active Monitoring principle manages the load among available VM's in a way to even out the number of active tasks on each VM at any given time. In such cases, the scheduler will assign two requests to the powerful server for each request assigned to the lower one [6]. The ant colony optimization approach is aimed to provide efficient distribution of workload among the nodes. When a request is initialized, the ant starts moving towards the source of food from the head node. The unprocessed request keeps a record of every node, it visited and records their data for future decision making [7].

In the task scheduling principle, two-level task scheduling mechanism is carried out to meet dynamic requirements of users as well as to obtain high resource utilization. It achieves load balancing by first mapping tasks to virtual machines and then the virtual machines to host resources, thereby improving the task response time, resource utilization and overall performance of the cloud computing environment [8]. User-prioritized guided Min-Min scheduling algorithm accommodates the demands of different users by delivering the services at different levels of quality. Therefore, the user gets the guarantee for the service that he sought for [9]. In cloud light weight policy, which not only balances the virtual machine work load in cloud computing datacenters, but it also assures QoS for users. It reduces both the number of VM migration processes and the migration time during applications execution [10].

A new type of federate container, virtual machine (VM), and its dynamic migration algorithm considering both computation and communication cost is designed. Experiments show that the migration scheme effectively improves the running efficiency of the system when the distributed system is not saturated [11]. The servers are polled in the cluster, and the mild load node is selected to response users' requests directly when a user-connection requests arrive. If there are no mild load nodes in the cluster, the weights are adjusted adaptively according to the server connection status, and the nodes are selected which has the minimum ratio of weight to load to provide service. At monitoring stage, it takes both the previous and current system condition into account to avoid unnecessary migrations [12].

Uncertainty-aware evolutionary scheduling method aims at dealing with uncertainties during execution and updating the scheduling so as to meet the deadline and optimize the execution cost of cloud applications [13]. In cloud computing, resources are consumed as services hence utilization of the resources in an effective way is achieved by deploying service scheduling and load balancing principles. The quality of service is an influencing parameter to assess the reliability of the cloud. It is possible to improve the efficiency of the quality of service based on the service scheduling algorithms by considering various factors like arriving time of the request, the time taken by the request to execute on the resource and the cost of use network communication [14].

The objective of this research work is based on the inferences from the literature survey.

3. RESEARCH PROBLEM STATEMENT

The focal point of this research is to design a scheduling principle which is suitable for a heterogeneous pool of computing resources that serve a set of services. The research problem is defined as follows: There is a set of services $S(S_1, S_2,..., S_m)$. Each service S_m is assigned with a value v based on a criterion. These services are served through the dissimilarly configured servers $N(N_1, N_2,..., N_n)$. Each server N_n is characterized by its capacity constraints. There is a set of requests $R(R_1, R_2,...R_k)$ at time t_i. Each request R_k seeks a particular service S_m. The scheduling principle is to assign the requests optimally across the servers proportionate to their serving capacity satisfying the capacity constraints [15].

The following are some of the assumptions based on which the solution is designed: The servers are dissimilarly configured. Each request has its own memory requirement and service time. Queued requests for the resources cannot be balked at, reneged or jumped. A schedule assigned to the resources is unaltered. When a fault occurs, schedule assigned to it is re-assigned to the existing pool of resources with the highest priority. Increase or decrease in the resources is continuously monitored and a change occurring at t_i will affect the schedule at t_{i+1} [16].

4. DESIGNED METHODOLOGY

This research introduces a novel method, namely *Quantification principle* which is incorporated in the service request scheduling process. Quantification principle involves a method to identify a server's processing ability based on the attribute with the highest level of influence on the server and a method to split the total number of requests that is proportionate to each server based on its capacity [15], [17]. Figure 2 shows the different steps involved in the request scheduling based on the quantification principle. The following sections present the phases of the request scheduling technique designed in this research.

4.1. Choosing the Most Influential Attribute using Conjoint Analysis

In a heterogeneous distributed computing environment, the weighted nodes scheduling principle, assign the number of requests to each server based on its serving capacity. The weights are assigned to each server based on one or more of its attributes. To design a weighted node scheduling technique, in this research work, the most preferred attribute of a server is used to measure its serving capacity. In order to identify the most preferred attribute of a server, Conjoint analysis, a statistical method is used.

Conjoint Analysis or Stated preference analysis is a mathematical statistical technique widely used in used in social sciences and applied science, including marketing, product management and operations research. This analysis is used to measure the customers' preferences based on the attributes of a product. It quantifies each attribute's preference value using multi-linear regression. The outcome of the analysis unveils the attributes' part-worth values and relative preference value of these values. The attribute with the highest relative preference value is the most preferred attribute [18]. Algorithm 1 presents the step by step processes for performing the conjoint analysis.

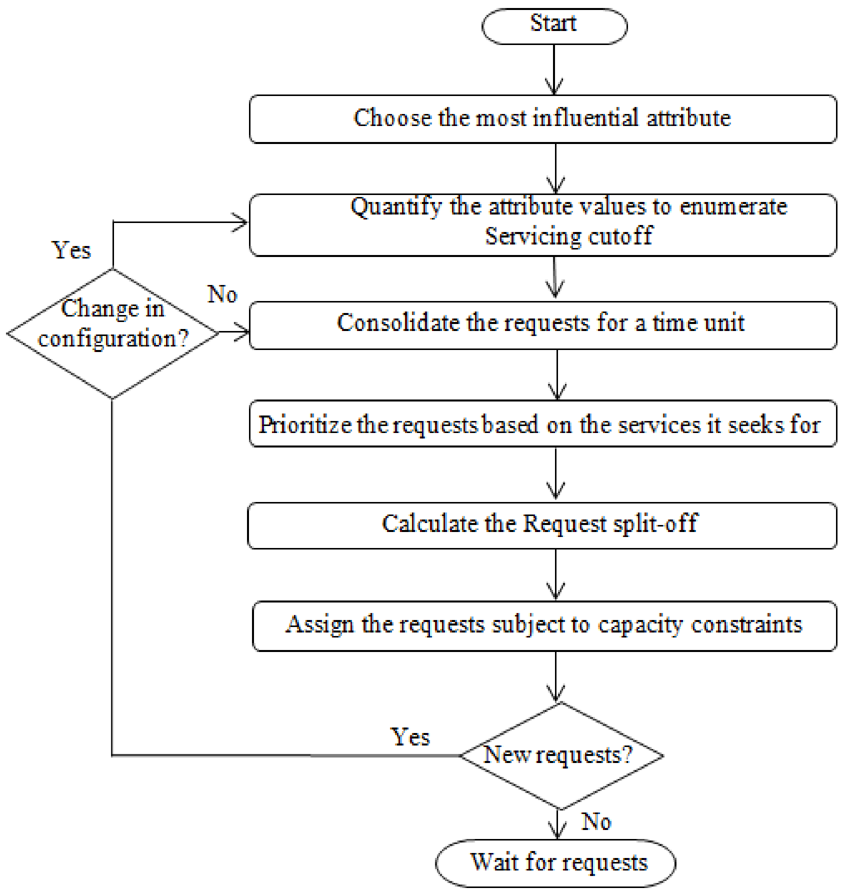

Start
Choose the most influential attribute
Quantify the attribute values to enumerate Servicing cutoff
Consolidate the requests for a time unit
Prioritize the requests based on the services it seeks for
Calculate the Request split-off
Assign the requests subject to capacity constraints
New requests?
Yes
No
Wait for requests
Change in configuration?
Yes
No

Figure 2. Overall processes involved in the Quantification principle

Algorithm 1: Performing Conjoint Analysis

```
Conjoint_Analysis(Attribute_List)
begin
    1.  Let each combination of the Attributes be a product_ profile.
    2.  rank each product_ profile based on user preference.
    3.  represent the ranks in a hypercube.
    4.  calculate the part_worth utility from hypercube.
    5.  calculate the relative_preference of the individual attribute.
    6.  return the attribute with the maximum relative_ preference as the preferred_attribute
End
/* End of Conjoint Analysis */
Output: Attribute with maximum preference
```

For this research, conjoint analysis has been carried out with the following attribute list: Server's throughput, the number of parallel connections it can handle ie, load capacity and memory size. Load capacity was identified as the most preferred attribute of the server.

An elaborate discussion on conjoint analysis and the method of performing conjoint analysis on a set of attributes are presented in [19].

4.2. Quantifying the Attribute Values using Z-score Method

The allocation share for each server is determined using its serving capacity based on the most preferred attribute of the server. A statistical method called Z-score is used to do this. Standard score or Z-score is a measure to quantify the difference between members of a group and the mean value of the group. It is a method of calculating the probability of a score occurring for the number of common distributions, such as the normal distribution. The probability value obtained using Z-score is the quantified value of the score's relative measure. Therefore, the Z - score method is used in this research is to enumerate a threshold value for each server that indicates the quantified measure of requests corresponding to the server's capacity [20].

An attribute's value set is taken as the input for the Z-score method and it measures the standard scores from the standard normal distribution. These values were converted into the unit of the percentage, namely *servicing cutoff*, that signifies the allocation share of each server for the total incoming requests. Based on this allocation share, the request split-off representing the number of requests out of the total number of requests that a server is expected to serve namely *request split off* is calculated. Algorithm 2 presents the steps of computing Z-score for the value set of the preferred attribute.

```
Algorithm 2: Computing Z-score
Algorithm Z_score(Pi)
 /*
  P is the preferred attribute for all the servers;
  Pi is the value of preferred attribute P of server i;
  Zp is the Z-score value of Pi ;
*/
begin
    1. compute Mean = $\frac{\sum_{j=1}^{n} P_j}{n}$;   where $P_j$ is the value of the preferred attribute of server j      (1)
    2. compute SD = $\sqrt{\frac{1}{n}\sum_{j=1}^{n}(P_j - \text{Mean})^2}$; where SD is the Standard Deviation;      (2)
    3. compute Std_Scorei = $\left(\frac{P_i - \text{Mean}}{SD}\right)$;      (3)
    4. compute Zp = map(Std_Scorei);   /* Function of mapping the Std_Score in Z- Table */
    5. return Zp;
end
 /* End of Z_score algorithm */

Output: Zp
```

An elaborate discussion on Z-score and the method of computing the request split off using Z-score with a set of values are presented in [21].

4.3. Consolidation of the Service Requests

The request scheduler collects the incoming requests for a discrete time interval between t_0 and t_1. Each request's arrival is time stamped. A single user can make any number of requests, but each request is considered as a separate job. Requests are queued based on a first-come, first-served principle.

4.4. Assignment of Priorities to the Service Requests

Since the capacity constraints restrict the number of requests a server can process, the scheduler assigns priorities to the requests based on a criterion. The requests are prioritized based on the services it seeks. The prioritization may be based on the business value of the service or its demand history [22].

4.5. Assignment of Service Requests to the Server

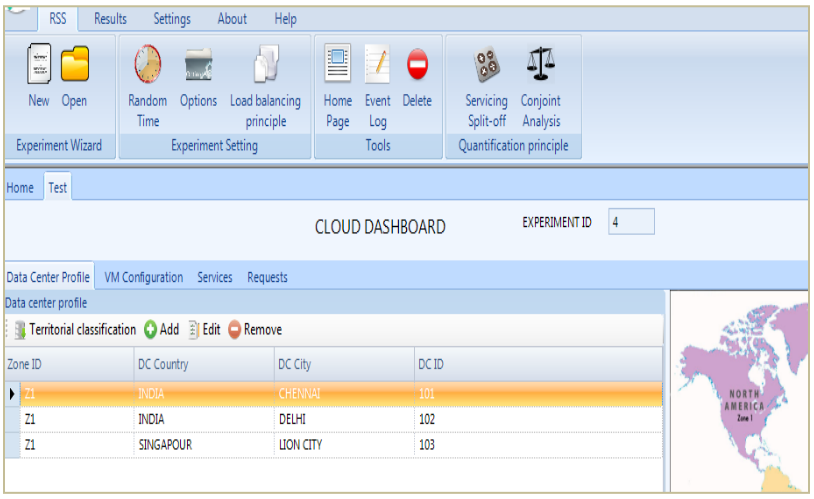
The load capacity of the server and the memory requirement of the requests are the two important constraints that impact the assignment of requests to a server. After computing the request split-off of requests for each server as described in section 4.2, request scheduler assigns the requests to the servers without violating the capacity constraints. Requests are assigned to the servers in the order of their priority. Algorithm 3 presents the request scheduling based on the quantification principle.

```
Algorithm 3: Service request scheduling technique based on Quantification principle
Algorithm Req_Scheduling( n, N, S, R, k, M, v)
/*
  n - number of servers where n is a natural number;
  N - Set of servers; N = {$N_1$, $N_2$, $N_3$,...$N_n$};
  S - Set of services, S = {$S_1$, $S_2$, $S_3$,... $S_m$}; where m is a natural number, m > 0
  K - Number of requests;
  R - Set of requests for the services, R = {$R_1$, $R_2$, $R_3$,....$R_k$}; where k is a natural number;
  v - A natural number assigned for the Item based on the business value;
  k - Number of requests;
*/
begin
    1. P = Conjoint_Analysis(Attribute_List);
            /*Using Conjoint_Analysis algorithm the most preferred attribute P from the attributes list is chosen */
    2. compute $z_i$ = Z_Score($P_i$);   /* Quantification of P by Z_Score algorithm      (4)
```

```
            where $1 \leq i \leq n$
    3.  compute $Z = \sum_{i=1}^{n} z_i$;                                                   (5)
    4.  compute $T_i = \text{abs}\left(\frac{z_i}{Z}\right) * 100$;                          (6)
        /* For each server calculate Servicing cutoff $T_i$ and i denotes $i^{th}$ server */
         where $1 \leq i \leq n$
         /* End of Quantification */
    5.  reorder requests R based on the service value v; /* Prioritize the requests */
    6.  compute $RS_i = \frac{k * T_i}{100}$;                                                (7)
        /* For each server, calculate Request Split-off $RS_i$ and i denotes $i^{th}$ server */
        where $1 \leq i \leq n$
        /* Assign request to servers */
    7.  if ((No. of connections available in $N_i > 0$)
                          && (Memory needed by $R_j \leq$ Unused Memory of $N_i$))
                      assign request $R_j$ to server $N_i$;
        where $1 \leq j \leq k$, $1 \leq i \leq n$
end
/* End of Service Request Scheduling algorithm */
Output: Request assignment with server
```

5. EXPERIMENT RESULTS AND DISCUSSION

The designed request scheduling technique based on quantification principle has been tested with the *Request Scheduler Simulator (RSS)*. It is an open-sourced, customizable visual tool which is the forerunner in the direction of dedicated simulation tools for the request assignment process, which evaluates the performance changes with respect to load balancing principles. The functionality of RSS is much inspired by CloudAnalyst [23]. An elaborate discussion on the design principle as well as the method of conducting experiments with RSS is presented in [24]. The dashboard of RSS simulator is shown in Figure 3.



CLOUD DASHBOARD

EXPERIMENT ID 4

Data center profile

Zone ID	DC Country	DC City	DC ID
Z1	INDIA	CHENNAI	101
Z1	INDIA	DELHI	102
Z1	SINGAPOUR	LION CITY	103

Figure 3. Dashboard of RSS simulator

Using the Performance Evaluator in RSS, parameters like Average Wait Time and Total Earned Value were computed and compared with few existing scheduling principles [25].

5.1. Average Wait Time

The scheduler has to appropriately route the requests to servers for enhancing user experience. Therefore, the scheduling technique should be efficient to assign requests to the servers as quickly as possible that entails in minimizing the requests wait time in the request queue. The Average Wait Time (measured in seconds) for a server is given as,

$$W = \frac{\sum_{i=1}^{k}(S_i - A_i)}{k} \tag{8}$$

where W is the average wait time of the i^{th} request, S_i is the server's processing start time for i^{th} request, A_i is the arrival time of i^{th} request and k is the number of requests.

Experiments were conducted in RSS and the average wait time of the requests for all the services is obtained using different scheduling principles for the sample of the three services is shown in Figure 4.

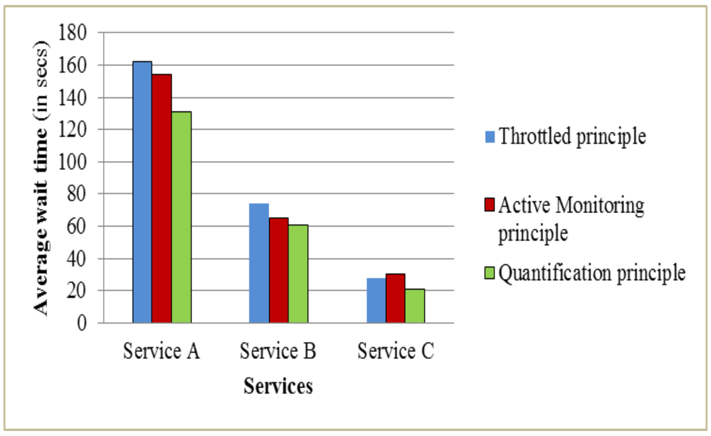


Figure 4. Average wait time of the requests for the services

5.2. Total Earned Value

Since the capacity constraints on the servers restrict the number of requests that can be processed, priorities for the requests have been assigned. Each service is assigned a value based on a criterion. This value may be assigned dynamically based on the service's demand history for a specific time frame or a static business value decided by the system designer. In RSS, the investigator has to choose on these options. The Total Earned Value is computed as follows:

$$T_{VAL} = \sum_{i=1}^{n} NS_i * v_i \tag{9}$$

where T_{VAL} is Total Earned Value measured as a natural number, NS_i is the number of requests served by the serveri where $i = 1, 2, \ldots n$, n is the number of servers and v_i is the weight assigned to i^{th} service. Figure 5 show the total Earned Value of the servers.

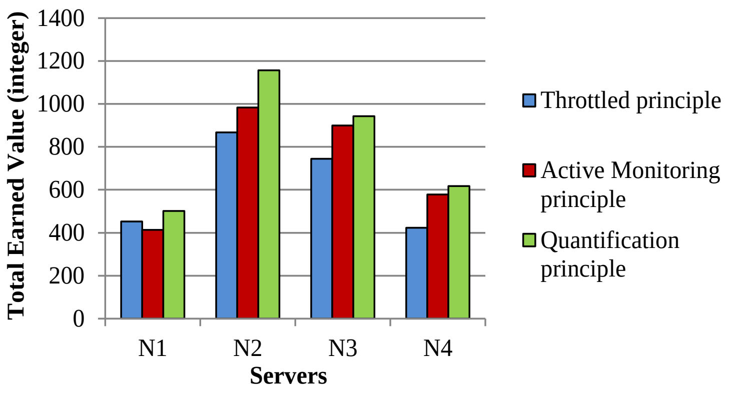


Figure 5. Total Earned Value of the servers

Experiments were conducted using RSS to prove the effectiveness of the quantification principle based request scheduling technique. After a series of experiments, the results were consolidated and ANOVA was carried out [26]. It is proved from the results that the quantification principle based request scheduling technique performs better compared to active monitoring and throttled scheduling principles.

6. CONCLUSION

The overall objective of this research is to design a scheduling principle that assigns the requests based on the values of the preferred attribute of the servers satisfying multiple constraints. The first objective of this research was achieved by introducing a method to identify the most preferred attribute of a server. This research used a mathematical statistical method called Conjoint Analysis to enumerate the level of influence of each attribute among the set of attributes of a server. The second objective of this research was achieved by identifying a method that determines each server's allocation share. A statistical method called Z-score was used to do this.The third objective of this research was accomplished by designing a scheduling principle that prioritizes the requests based on the services they seek and assigns requests to each server based on its allocation share satisfying the capacity constraints.

Experiments were conducted using a cloud simulator to prove the effectiveness of the quantification principle based request scheduling technique. The designed scheduling principle found to be suitable for Infrastructure as a Service cloud model. Extending the designed solution for other cloud models is a desirable extension of this research.

REFERENCES

[1] S. Andrew Tanenbaum and M. Steen, *"Distributed Systems: Principles and Paradigms,"* Prentice-Hall Inc., 2008.

[2] D. Vidyarthi, *et al.*, "*Scheduling in Distributed Computing Systems, Analysis, Design and Models,"* Elsevier, 2009.

[3] H. Yu, et al., *"Job Scheduling Algorithm In Cloud Environment,"* in 5th International Conference on Computational and Information Sciences, pp. 1652-1655, 2011.

[4] A. Thomas Henzinger, et al., *"Static scheduling in Clouds,"* in 3rd USENIX conference on Hot topics in cloud computing, pp. 1-6, 2011.

[5] W. Wang and G. Casale, *"Evaluating Weighted Round Robin Load Balancing for Cloud Web Services,"* in 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, pp. 393-400, 2014.

[6] N. Rodrigo Calheiros, *et al.*, "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms," *Software: Practice and Experience*, Wiley Press, vol. 41, no.1, pp. 23-50, 2011.

[7] L. D. Dhinesh Babu and P. Venkata Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," *Applied Soft Computing*, vol. 13 (5), pp. 2292–2303, 2013.

[8] Yuan, *et al.*, "Temporal task scheduling with constrained service delay for profit maximization in hybrid clouds," *IEEE Transactions on Automation Science and Engineering*, vol. 14, pp. 337-348, 2017.

[9] H. Chen, et al., *"User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing,"* in National Conference on Parallel Computing Technologies, pp. 1-8, 2013.

[10] M. Mesbahi, et al., *"Cloud light weight: A new solution for load balancing in cloud computing,"* in International Conference on Data Science & Engineering, pp.44-50, 2014.

[11] Xiao Song, *et al.*, "A Load Balancing Scheme Using Federate Migration Based on Virtual Machines for Cloud Simulations," *Mathematical Problems in Engineering*, vol. 2015, Article ID 506432, 2015.

[12] W. T. Wen, et al., *"An ACO-based Scheduling Strategy on Load Balancing in Cloud Computing Environment,"* in 9th International Conference on Frontier of Computer Science and Technology, pp. 364-369, 2015.

[13] Meng, et al., *"An Uncertainty-Aware Evolutionary Scheduling Method for Cloud Service Provisioning,"* in 2016 IEEE International Conference on Web Services (ICWS), pp. 506-513, 2016.

[14] Sirisha Potluri and Katta Subba Rao, "Quality of Service based Task Scheduling Algorithms in Cloud Computing," *International Journal of Electrical and Computer Engineering (IJECE)*, Vol 7, No 2, pp. 1088-1094, April 2017.

[15] R. Arokia Paul Rajan, *"Request Scheduling based on Quantification Principle in Cloud,"* in International Conference on Big data and Cloud Computing (ICBDCC'17), pp. 135-142, April 2017.

[16] Jiao Jintao, *et al.*, "Research on Batch Scheduling in Cloud Computing," *TELKOMNIKA (Telecommunication, Computing, Electronics and Control)*, Vol 14, No 4, pp. 1454-1461, December 2016.

[17] R. Panneerselvam, *Research Methodology*, Prentice-Hall of India, 2004.

[18] R. Vithala Rao, *Applied Conjoint Analysis*, Springer, 2014.

[19] R. Arokia Paul Rajan, *et al.*, "Preference Analysis for Enumeration of the Most Influential Attribute of Compute Nodes," *International Journal of Computer Applications*, Volume 121, Issue 22, July 2015, pp. 17-22.

[20] S. C. Gupta and V. K. Kapoor, *Fundamentals of Mathematical Statistics*, Sultan Chand & Sons, 2014.

[21] R. Arokia Paul Rajan and F. Sagayaraj Francis, *"Quantified Weighted Nodes Scheduling Principle For Cloud Architectures,"* in 12th International Conference on Innovative Engineering Technologies (ICIET), pp. 56-60, 2015.

[22] Z. Lee, et al., *"A dynamic priority scheduling algorithm on service request scheduling in cloud computing,"* in International Conference on Electronic and Mechanical Engineering and Information Technology, vol. 9, pp. 4665-4669, 2011.

[23] B. Wickremasinghe, et al., *"CloudAnalyst: A CloudSim-Based Visual Modeller for Analysing Cloud Computing Environments and Applications,"* in 24th IEEE International Conference on Advanced Information Networking and Applications, pp. 446-452, 2010.

[24] R. Arokia Paul Rajan and F. Sagayaraj Francis, "RAS: Request Assignment Simulator for Cloud-Based Applications," *KSII Transactions on Internet and Information Systems*, Vol. 9, No. 6, pp. 2035–2049, 2015.

[25] Tianshu You, *et al*., “Performance Evaluation of Dynamic Load Balancing Algorithms”, *Indonesian Journal of Electrical Engineering and Computer Science*, Vol 12, No 4, pp. 2850-2859, April 2014.
[26] A. Hamdy Taha, *Operations Research: An Introduction*, Prentice Hall of India Private Limited, 1999.

BIOGRAPHY OF AUTHOR

R. Arokia Paul Rajan is currently working as Associate Professor, Department of Computer Science, Christ University, Bengaluru, India. He holds Ph.D in Computer Science & Engineering from Pondicherry University, India. His research area is data management in Cloud architectures. He published 11 research papers in international journals and conferences.