Preference Analysis for Enumeration of the Most Influential Attribute of Compute Nodes

R. Arokia Paul Rajan
Research Scholar
Department of Computer Science & Engineering
Pondicherry Engineering College
Pondicherry

S. Ganapathy
Professor
Department of Mathematics
Pope John Paul II College of Education
Pondicherry

F. Sagayaraj Francis
Professor
Department of Computer Science & Engineering
Pondicherry Engineering College,
Pondicherry

ABSTRACT
Request assignment with compute nodes in a large scale distributed computing environment is a challenging research area. To devise a fitting solution, need to identify the impacting parameters and pertinent constraints originating from such an environment. This paper introduces a novel method that helps to ascertain the level of influence of each parameter among the set of parameters of cloud configurations. This work used conjoint analysis, a mathematical statistical method for enumerating the impact level of the parameters. After identifying the most influencing parameter, This work used Z-Score statistical method to quantify the capacity of the compute node into the unit of percentage. Based on this percentage split-off, the users' requests are assigned to the compute nodes. Thus the nodes are assigned to the requests based on their capacity proportion. The focus of this paper is to present the method of conducting conjoint analysis for the virtual machines' configuration in cloud. This work is the first attempt that applies conjoint analysis for identifying the impact level of parameters in the cloud architectures.

General Terms
Distributed computing

Keywords
Cloud computing, Conjoint analysis, Part-worth utility, Z-score

1. INTRODUCTION
Cloud is not a particular product, but a way of delivering IT services that are consumed on demand, elastic to scale up or down as needed, and follow a pay-for-usage model [1]. Cloud applications receive requests from geographically widespread global users. Request scheduling for such a large scale distributed computing environment is always a difficult task. Because, assigning the requests with compute nodes directly impact the performance of the system. Before designing an effective request scheduling algorithm, it is predominantly important to identify impacting parameters and constraints that are prevailing in the environment. Request scheduling is influenced by the load balancing principles because, these strategies actually binds the requests with the computing resources [2].

This work focused on weighted task distribution load balancing scheme that distributes the incoming requests onto the computing resources in the cluster using weights. In this principle, the system designer has to specify the weight of tasks a server should receive relative to other servers. This strategy is effective for the compute nodes in the cluster do not all have the same capacity. For example, if one of three nodes only has 2/3 capacity of the two others, it can be represented as 3, 3, 2 as their weights. It means the first compute node can be assigned with 3 requests, the second node with 3 requests, and the last node with only 2 tasks, for every 8 requests received. That way the server with 2/3 capacity only receives 2/3 tasks compared to the other servers in the cluster. Figure 1 illustrates the weighted task distribution scheme.

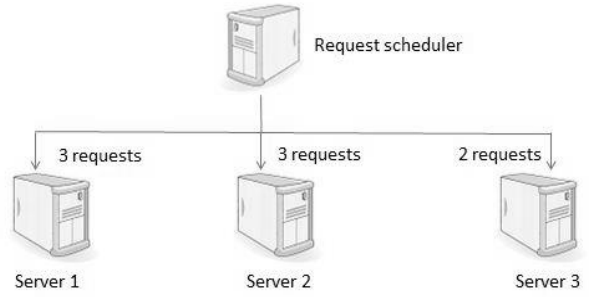

Fig 1: Weighted load balancing principle

Thus, weighted task distribution load balancing principle is effective for request assignment with compute resources of heterogeneous capacity. But the question is, based on what parameter the weight can be determined?

This paper presents a novel approach of applying a statistical method for identifying the most influencing parameter out of a group of parameters. Conjoint Analysis is a mathematical statistical technique used to assess the different contributions aspects of a product or service make to individuals in the purchasing decision [3]. This work used conjoint analysis for enumerating the impact level of the parameters [4]. After identifying the most influencing parameter, this work used Z-Score or standard score statistical method to find the measurement of a value's relationship to the mean in a group of values of that parameter [5]. This work converted this score into a percentage. This percentage represents the capacity proportion of a node. Now the consolidated request at a unit time is split according to this percentage. According to the percentage split-off, requests are assigned to the compute nodes.

The rest of the paper is organized as follows: Section 2 discusses different contributions on request scheduling algorithms that are having major relevance with the proposed strategy. Section 3 presents and elaborates major parameters in the cloud architectures. Section 4 introduces conjoint analysis with an illustration. Section 5 presents the method of implementing conjoint analysis with the sample test data. Section 6 describes the significance of identifying parameter using conjoint analysis and section 7 discusses conclusion with future expansion possibilities.

2. RELATED WORKS

This section discusses some of the relevant contributions on conjoint analysis and load balancing algorithms in large scale computing environments. The Empirical study analyze has been carried out for evaluating whether revenue management models can be applied to Cloud Computing. The conjoint analysis reveals that operating system, price and support level of an IaaS offer have a major impact in the customers' selection process [6]. A qualitative study was carried out on SaaS selection factors that takes into account both the customers' and the vendors' perspectives. According to our findings, selection factors differ across customer segments [7].

There are three load balance policies used for request scheduling in Cloud Analyst. The round robin algorithm assigns the requests to the virtual machines in an orderly manner. In throttled load balancing policy, each virtual machine is assigned with a throttling threshold [8]. The request scheduling algorithm inspired by the honey bees' behavior strives to achieve a load balance among virtual machines. The optimization of the throughput and waiting time of the job queue is achieved by adjusting the priorities of the tasks [9]. The work accommodated the demands of different users by delivering the services at different levels of quality. Therefore, the user is guaranteed of the services that he seeks [10]. Queuing game model for service scheduling schemes was compared and studied for job scheduling [11]. This principle maximizes the Cloud Computing platform's payoff through controlling the service requests, whether to join or balk, based on the value of the CCP's admission fee. A dynamic balancing algorithm was introduced in which the requests are queued to a computing node based on the capacity of the machine [12]. Ant colony optimization principle can be adoptable for workload distribution with the pool of nodes in the cloud [13]. There is an improvised Dynamic Round Robin (DRR) algorithm for energy-aware virtual machine scheduling which yielded better results comparing with greedy, round robin and power save strategies [14].

3. IMPACTING PARAMETERS IN CLOUD

To construct a request scheduler in cloud, it is essential to enumerate various impacting parameters arises from its structural components. Fig. 2 depicts the typical cloud architecture with different components. Computing nodes, users, requests, services and networks are the five constituting domains from which a set of parameters has been derived [15].

The virtual machines are the computing nodes in cloud based applications. In fact, there are many influencing parameters which are having a major impact over the system design [16]. This work designed the proposed requests scheduler principle based on the following parameters [17]:

• Load capacity: Number of parallel sessions a node can sustain. It is limited by the server's hardware configuration as well as the network operating system. This work considered server's load capacity as one of the limiting constraint in job allocation.

• Storage capacity: Maximum Memory capacity of a node which accommodates the service instances measured in terms of bytes. It restricts the number of services instantiated into the computing node.

• Geographical proximity: Groups the users' requests and assign them to the nearest geographically located server. This algorithm adapted DNS based grouping of users.

• Preference of node: Many web service providers are allowing the users to choose their server. This will have an adverse effect with the actual principle of scheduling. This algorithm has accommodated such options by directing those requests to the preferred server. These guest requests are served in the host server along with its own request's pool. But they are appended with the queue of the same priority level.

• Suitability of node: There are services which requires suitable server when get executed. Even if the job scheduler assigns that particular request to some other sever node, that request need to be re-directed to a suitable node. For example, Windows Azure virtual machines with OLTP are suitable for mission-critical transactional services.

• Participation policy of a node: A server may incline or evade some requests due to the request's geographical origin place or type of data encompassed with the service. Those requests are actually additional load of the newly assigned server.

• Processor speed: It is the measure of the number of instructions executed per second by the computer termed in megahertz or gigahertz.

• Demand: It is the consolidated number of requests collected for a time frame between Ti to Ti+1. It directly signifies the value earned by a service.

• User type: Based on the user classification prioritization can be ascertained. Many cloud services, especially SaaS application categories their users, namely free users, subscribers and privileged users. According to their classification, strategy for rendering the services is differentiated.

• Arrival time: Time at which the request received by the scheduler.

• Stay time: Time taken by the server to complete the submitted job. This work assumes that burst time is distinct for each request. This nature makes the proposed model fitting to cloud applications where the same service is invoked by different users but varying in burst time.

• Nature of request: Read and write are the primitive operations performed for the requests in common. Certain servers are specialized to perform those requests with better performance. For example, ESX server family is an apt selection for assigning I/O intense requests.

• Bandwidth: In a network, it is the rate at which the data is transferred from one point to another in a particular unit of time. Response time measurement involves the time delay assessment which is influenced by the bandwidth.

• Connection cost: This is the measure of network costs specifies how long a bit of data takes to travel across the

network from one point to another, measured in microseconds. It includes processing delay, queuing delay, transmission delay and propagation delay.

• Traffic rate: Requests are highly populated at peak hours which will be differing from region to region. Therefore the mutual sharing of workload between peak-on servers with peak-off servers will reduce the traffic congestion. The proposed model not extended this exchange of workload between the servers.

• Queue size: It is the request pool being populated with compute nodes before execution of the requests scheduling principle. The queue is unavoidable in any scheduling strategy, but it is minimized in this model. Because, this model assigned the work load to the servers proportionate to the capacity of each computing node.

4. INTRODUCTION TO CONJOINT ANALYSIS

Conjoint analysis or stated preference analysis is a statistical technique that originated in mathematical psychology. Today it is used in many of the social sciences and applied science, including marketing, product management and operations research [3].

Let's assume a customer goes to shop to buy an MP3 player. The salesperson tells him, he can get the models 32 GB, off the shelf or get a model 64 GB, but then he has to wait one week for the delivery. Now the question is what is his preference? His preference for one of the alternatives will reveal the part-worth utilities of individual attributes. In this example, attribute one is the memory size and attribute two, the delivery time. When he chooses model A, it will show he put higher emphasis on shorter delivery time. Choosing model B will reveal he gave higher emphasis for large memory size. So in a conjoint analysis, the part-worth utilities of individual attributes, in this case, memory size and delivery time are calculated based on the selection or ranking for the defined set of combinations of attribute values.

5. CONJOINT ANALYSIS FOR CLOUD PARAMETERS

Let us consider a simple cluster of virtual machine's configuration panel of the cloud architecture. This work take into consideration of only three attributes as presented in Table 1. They are, operating system - Windows or Linux, load capacity - 8 users or 6 users, and memory size - 4 GB or 8 MB.

Table 1. Chosen attributes

Factor	$Server_1$	$Server_2$
Operating System	Windows	Linux
Load capacity	8 Users	6 Users
Memory (Mem)	4 GB	8 GB

Combine all attributes with their individual values will result in 8 different combinations as presented in Table 2. The combinations are given as,

Table 2. Model derived from the attributes

Model	Conjoint attributes
Model 1	Windows, 6 users, 32 GB
Model 2	Linux, 6 users, 32 GB
Model 3	Windows, 6 users, 64 GB
Model 4	Linux, 6 users, 64 GB
Model 5	Windows, 8 users, 16 GB
Model 6	Linux, 8 users, 16 GB
Model 7	Windows, 8 users, 64 GB
Model 8	Linux, 8 users, 64 GB

In order to solve this problem with a mathematical model, code the levels by -1 and +1 for each. For example, Windows is coded as -1 and Linux is coded as +1. Here the list of combinations with their coding and this is called the design matrix presented in Tab. 2. For k attributes, there are 2k possible combinations. Using all possible combinations is called a full factorial design shown in Table 3. Treat the 3 attributes as variables, each of them with the value of -1 to +1.

Table 3. Design matrix with levels

Model	OS (X1)	LC(X2)	Mem(X3)
Model 1	-1	-1	-1
Model 2	1	-1	-1
Model 3	-1	1	-1
Model 4	1	1	-1
Model 5	-1	-1	1
Model 6	1	-1	1
Model 7	-1	1	1
Model 8	1	1	1

With a graphical notation presented in Fig. 5, each combination is represented as a point in a corner of the cube. One dimension of the cube shows OS, the second shows load capacity and the third memory size.

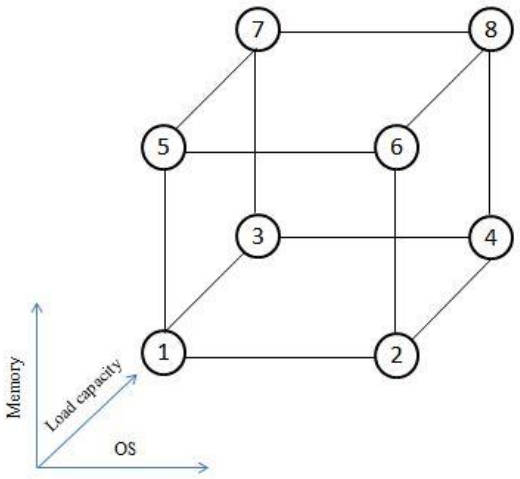

Fig 5: Models represented as a cube

The next step in a conjoint analysis is to ask the system architect for ranking the possible combinations. For example, to give 1 for most preferred combination, going down to 8 for the least preferred combination as presented in Table 4.

Table 4. Ranking the preferences

Model	Conjoint attributes	Preference
Model 1	Windows, 6 users, 32GB	8
Model 2	Linux, 6 users, 32GB	7
Model 3	Windows, 6 users, 64 GB	4
Model 4	Linux, 6 users, 64GB	3
Model 5	Windows, 8 users, 16GB	6
Model 6	Linux, 8 users, 16GB	5
Model 7	Windows, 8 users, 64GB	2
Model 8	Linux, 8 users, 64GB	1

Use linear regression model function to describe the ranking to find the part-worth utilities which is given as,

Ranking = Part-worth of attribute1 * Attribute1 level + Part-worth of attribute2 * Attribute2 level + Part-worth of attribute3 * Attribute3 level + Constant

Using multi-linear regression function, it is given as,

$$Y = \beta_{OS} * X_1 + \beta_{LC} * X_2 + \beta_{Mem} * X_3 + \mu \quad (1)$$

The ranking is expressed as part-worth of attribute1, OS, multiplied by the level for attribute 1, -1 or +1, plus the part-worth of attribute 2, multiplied by the level for attribute 2, plus part-worth of attribute 3 multiplied by level for attribute 3 plus a constant as given (1). As a mathematical equation, it is shown in Table 5.

Table 5. Calculating part-worth utilities

Rank	Part-worth utilities
8	$\beta_{OS} *(-1) + \beta_{LC} *(-1) + \beta_{Mem} *(-1) + \mu$
7	$\beta_{OS} *(1) + \beta_{LC} *(-1) + \beta_{Mem} *(-1) + \mu$
4	$\beta_{OS} *(-1) + \beta_{LC} *(-1) + \beta_{Mem} *(1) + \mu$
3	$\beta_{OS} *(1) + \beta_{LC} *(-1) + \beta_{Mem} *(1) + \mu$
6	$\beta_{OS} *(-1) + \beta_{LC} *(1) + \beta_{Mem} *(-1) + \mu$
5	$\beta_{OS} *(1) + \beta_{LC} *(1) + \beta_{Mem} *(-1) + \mu$
2	$\beta_{OS} *(-1) + \beta_{LC} *(1) + \beta_{Mem} *(1) + \mu$
1	$\beta_{OS} *(1) + \beta_{LC} *(1) + \beta_{Mem} *(1) + \mu$

Now, set up a system of linear equations, using the coded combinations and the ranking for each combination given by the investigator. This system of linear equations can be solved as a multi-variant linear regression. In this illustration, calculate the part-worth utilities in the following way: To find the main effect for attribute 1, OS, take the average ranking for all model combinations vs. X_1 equals *+1*, that means, Linux OS and subtract the average ranking of all combinations vs. X_1 equals *-1*. That represents Windows OS. In the cube, it corresponds to the sum of the ranking values for all points on the right side of the vertical plane, minus the sum of ranking for all points on the left side of the vertical plane as shown in Fig. 6.
Divide by 4 because, this work take the average of 4 points each and set it in relation to total variation of x value of -1 to +1. So divide by 2. As a result, part-worth utility for OS of -0.5.

$$\beta_{OS} = \frac{1}{4}[18 - 20] \div 2 \quad (2)$$

In the same way, proceed for the other two dimensions to calculate part-worth utilities for load capacity and memory size.

$$\beta_{LC} = \frac{1}{4}[12 - 20] \div 2 \quad (3)$$

$$\beta_{Mem} = \frac{1}{4}[10 - 26] \div 2 \quad (4)$$

Part-worth utilities are calculated from (2), (3), and (4) and given in the Table 6.

Table 6. Design matrix with levels

Parameter	Part-worth
Operating System	-0.5
Load Capacity	-1
Memory	-2

The ranking calculated by substituting the values of Table 5 in (1) and is expressed in (8). It is observed that the model function fits exactly the actual ranking presented in Fig. 8.

$$Y = (-0.5) * X_1 + (-1) * X_2 + (-2) * X_3 + 4.5 \quad (5)$$

To calculate the relative preference for each individual attribute given in (8), The total range of variations for level x=-1 and +1 which is 7 in the example. Calculating the variations for the attributes (Xi) as follows,

$$\pm 1.\beta_{OS} = \pm 0.5 = 1 \quad (6)$$

$$\pm 1.\beta_{Mem} = \pm 2 = 4 \quad (7)$$

$$\pm 1.\beta_{LC} = \pm 1 = 2 \quad (8)$$

$$\text{Total variation} = 1 + 4 + 2 = 7 \quad (9)$$

Therefore, relative preference for individual attributes can be calculated as,

$$\text{Relative preference} = \frac{\text{Individual preference}}{\text{Total preference}} \quad (10)$$

Using equation (13), the relative preference ares calculated and the results are given in Table 6.

Table 6. Design matrix with levels

Parameter	Relative preference
Operating System	14%
Load Capacity	57%
Memory	27%

Figure 8 shows the comparison of the actual ranks given by the respondents and the calculated rank by the linear regression function.

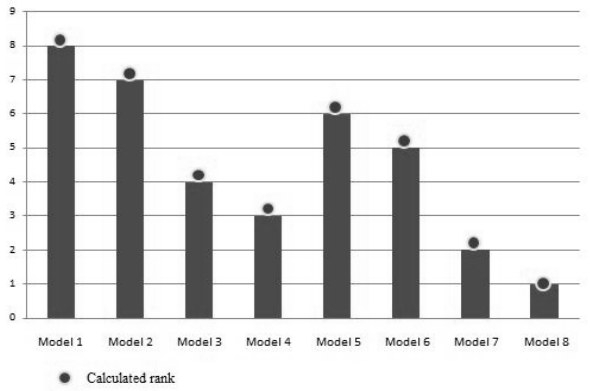

Fig 8: Rank calculated by linear regression correlation

From the Table 3.5, it is concluded that the attribute with highest percentage value reveals that it is the most influencing attribute out of the other attributes. In this illustration, load capacity is the most influential attribute comparing with operating system and memory capacity.

6. USE OF CONJOINT ANALYSIS RESULT

The conjoint analysis brings out the most influencing parameter out of a set of configuration attributes. Based on that parameter, percentage split-off is enumerated using Z-score method. Standard score or Z-Score is a statistical method to find the measurement of a value's relationship to the mean in a group of values of that parameter. This work converted this score into a percentage. This percentage represents the capacity proportion of a node. The total incoming requests computed to split according to this split-off. The request scheduler now assigns the proportioned requests to each compute resource. The percentage split-off represents the capacity proportion of the compute node in a cluster of nodes. This work termed this improvised load balancing principle as capacity proportioned compute nodes load balancing principle. This work developed a simulator namely Request Assignment Simulator (RAS) and conducted experiments. The results are encouraging when compared with round robin and throttling load balancing principles [18].

The consolidated requests at unit time are splitted according to this percentage and then the split-off requests are assigned to the compute nodes. Thus the nodes are assigned to the requests based on their capacity proportion. The focus of this paper is to present the method of conducting conjoint analysis for the virtual machines' configuration in cloud.

7. CONCLUSION

The main focus of this paper is to present a method how conjoint analysis can be carried out to identify the most influencing parameter. There are few approaches that are adopted to enumerate the capacity of a node based on the computing node's attribute, The authors are confident that this paper is the first attempt which incorporated conjoint analysis, a mathematical psychology concept fitting to the cloud deployments. Using the results enumerated by this work, weighted task distribution load balancing scheme using Z-Score can be extended as the future work. The designed solution is more generic and can be extendable to grid, cluster and P2P architectures. Also, careful inclusion of new attributes for cloud architectures has got the potential to extend the existing model in future.

8. REFERENCES

[1] Anthony Velte, Toby Velte & Robert Elsenpeter. 2009 Cloud Computing, A Practical Approach. McGraw-Hill Education.

[2] Armbrust, M., Fox Griffith, A. R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I. & Zaharia, M. 2009. Above the Clouds: A Berkeley View of Cloud Computing. Retrieved November 16, 2014, from the website of University of California, Berkeley: EECS Department:

www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf

[3] Panneerselvam., R. 2014. Research Methodology (2nd ed.). New Delhi: Prentice-Hall of India.

[4] Darius Singpurwalla, A Handbook of Statistics. An overview of statistical methods. Available online at: http://www.e-booksdirectory.com/details.php?ebook=9440.

[5] Gupta., S. C., Kapoor, V. K. 2000. Fundamentals of Mathematical Statistics (10th ed.). New Delhi: Sultan Chand & Sons.

[6] Anandasivam, A., Best, P., & See, S. 2010. Customers' Preferences for Infrastructure Cloud Services, Proceedings of Twelfth Conference on Commerce and Enterprise Computing. pp.144- 149.

[7] Polyviou, A., Pouloudi, N. & Rizou, S. 2014. Which Factors Affect Software-as-a-Service Selection the Most? A Study from the Customer's and the Vendor's Perspective, Proceedings of the 47th Hawaii International Conference on System Sciences. pp.5059-5068.

[8] Wickremasinghe, B., Calheiros, R. N., & Buyya R. 2010. CloudAnalyst: A CloudSim-Based Visual Modeller for Analysing Cloud Computing Environments and Applications, Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications. pp. 446-452.

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

[10] Huankai Chen, F. Wang, Helian, N. & Akanmu, G. 2013. User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing, Proceedings of the National Conference on Parallel Computing Technologies. pp. 1-8.

[11] Fuhong Lin, Xianwei Zhou, Daochao Huang, Wei Song & Dongsheng Han. 2014. Service Scheduling in Cloud Computing based on Queuing Game Model, KSII Transactions on Internet and Information Systems, Vol.8 (5), pp. 1554-1566.

[12] Bo, Z., Ji, G., & Jieqing, A. 2010. Cloud loading balance algorithm, Proceedings of the IEEE 2nd International Conference on Information Science and Engineering. pp. 5001–5004.

[13] Nishant, K., Sharma, P., Krishna, V., Gupta, C., Singh, K. P., Nitin, N., & Rastogi R. 2012. Load balancing of nodes in cloud using ant colony optimization, Proceedings of the 14th International Conference on in Computer Modelling and Simulation. pp. 3–8

[14] Ching-Chi Lin, Pangfeng Liu &Jan-Jan Wu. 2011. Energy-Aware Virtual Machine Dynamic Provision and Scheduling for Cloud Computing, Proceedings of the IEEE International Conference on Cloud Computing. pp. 736-737.

[15] Kashyap, S. R. 2007. Algorithms for Data Placement, Reconfiguration and Monitoring in Storage Networks, Ph.D. dissertation report, University of Maryland.

[16] Ito, R. 2009. Job Scheduler Parameter Analysis for Evaluation of Effectiveness. Proceedings of the 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing. pp.62-69.

[17] Arokia Paul Rajan & R., Sagayaraj Francis, F. 2014. Dynamic Scheduling of Requests Based on Impacting Parameters in Cloud Based Architectures, Proceedings of the 48th Annual Convention of Computer Society of India, Advances in Intelligent Systems and Computing, Springer International Publishing, Vol. I (248). pp. 513-521.

[18] Arokia Paul Rajan, R. and Sagayaraj Francis, F. 2014. Experimenting with Request Assignment Simulator (RAS), International Journal on Computer Science and Engineering, vol. 6, issue 11, 363-373.

[19]