Self-execute Model for Cloud Service Providers

Vaibhav Bahuguna
Department of Computer Science and Engineering,
Lingaya's Vidyapeeth, Faridabad, India

Manoj Yadav
Department of Computer Science and Engineering,
Lingaya's Vidyapeeth, Faridabad, India

Md. Aftab Alam
Department of Electronics and Communication Engineering,
Lingaya's Vidyapeeth, Faridabad, India

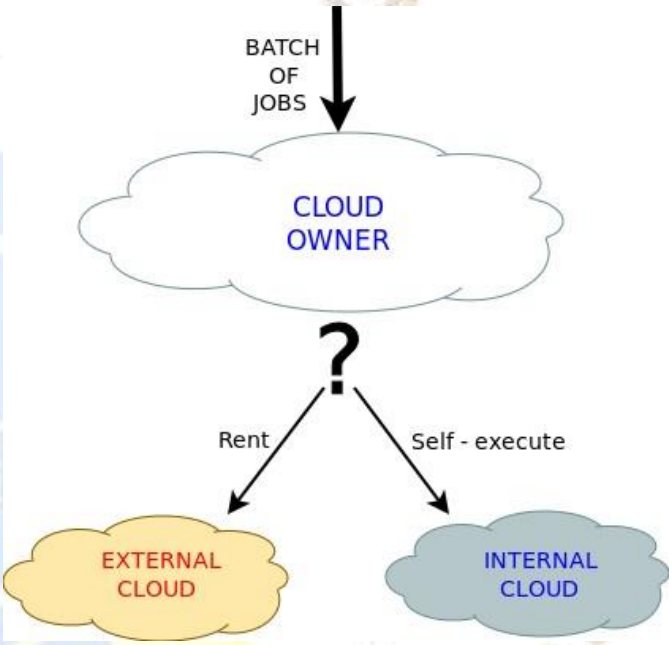
***Abstract*—As Cloud Computing is an emerging field, many improvements are being proposed to provide users with better services and facilities. This paper deals with the illusion of infinite resource availability on demand, one of the new aspect in Cloud Computing. A combination of forecasting models and game theoretic approaches have been proposed so as to continue providing this illusion without any glitches. This implementation is a new way of looking at the problem and with coordination from different providers it becomes possible for each provider to decide his best strategy. This work provides an efficient way for the cloud provider to decide on his strategies to execute a job i.e., whether to use his own services to execute (self-execute) or to pay rent to other cloud providers. Forecasting has been done to get an idea of previous demands. Game theory is a concept that is generally applied for economical undertakings, generally for the current period, and is a good way to engage it in deciding the strategies adopted by an organization. Hence the design considers both the previous as well as current demand to decide on the provider's strategy making the results more accurate. This paper combines two different approaches and based on their results decides upon a strategy that has minimum deviation. The results obtained from this utility function show an almost equal distribution of Rent and Self-execute strategies. This work can be enhanced by including more factors, especially of financial importance, in the utility and providing methods for scalability.**

***Keywords*—Cloud Computing, Forecasting, Game Theory, Cloud Provider, Rent/Self-execute Strategy, Utility**

I. Introduction

Cloud Computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [1]. Load Balancing is the process of distributing the load among various nodes of a distributed system when it becomes difficult to predict the number of requests that will be issued to a server. It considers factors like execution time, resource availability and requirement among others to improve job response time, throughput, etc. In order to provide better service-level agreements, the cloud provider has to provide such improvements to the user.

Quality of Service (QoS) is the resource reservation control mechanism in place to guarantee a certain level of performance and availability of a service. It provides a level of assurance that the resource requirements of an application are strictly supported [2]. It is possible that the resource requirement of a user may not be supported by the cloud provider. In such scenarios, the cloud provider has to provide a means of executing the user's load. The provider has to decide the appropriate strategy such that the user's needs are met. One of the most interesting aspects in Cloud Computing is the feeling of availability of 'infinite' computing resources that the cloud provider tries to distribute to the user in an elastic way [3]. The user does not fully realize the internal allocations while demanding for more resources.



For this infinite demand of the user to be met, the cloud provider has to find ways to do it without incurring any loss. One of the approaches to do it is to calculate the number of times the provider does not have the capacity to execute the load and based on that draw up an agreement with another provider to execute the load. This calculation can be predicted from past data through forecasting and by game theory (to include the current data also). The method of standard deviation can help in deciding the strategy of the cloud provider.

II. Literature Survey

One of the aspects of Cloud Computing is the illusion of infinite computing resources available on demand, thereby eliminating the need for Cloud Computing users to plan far ahead for provisioning. For this, realizing the economies of scale afforded by statistical multiplexing and bulk purchasing requires the construction of extremely large data-centers. Building, provisioning, and launching such a facility is a hundred-million-dollar undertaking [4].

A system composed of a virtual network of virtual machines capable of live migration across multi-domain physical infrastructure have been constructed. By using dynamic availability of infrastructure resources and dynamic application demand, a virtual computation environment is able to automatically relocate itself across the infrastructure and scale its resources [5]. Thus the QoS improvements can be met using this virtual machine setup. Depending on the type of application, the generated workload can be a highly varying process that turns difficult to find an acceptable trade-off between an expensive over-provisioning able to anticipate peak loads and a sub performing resource allocation that does not mobilize enough resources. These properties can be leveraged to derive a probabilistic assumption on the mean workload of the system at different time resolutions [6].

There are many proposals that dynamically manage Virtual Machines by optimizing some objective function such as minimizing cost function, cost performance function and meeting QoS objectives. The objective function is defined as Utility property which is selected based on measures of response time, number of QoS, targets met and profit etc. [5]. This Utility property can then be adjusted to include factors of individual importance. They have been modified to meet needs according to current demand. But they may prove complex and may not work in a practical virtualization cloud system with real workload. These approaches work best for stationary demands and may not give optimal solution for dynamic resource requirements. The utility property can be modified to include dynamic demands and allocation. This requires formulating the appropriate utility property that captures these demands.

III. DESIGN

The process of load balancing considers factors like execution time, the number of resources required and the probability of having to wait for the resource at the server among others. Using these factors, a utility function is formulated:

$$Utility = \frac{\frac{e}{t_{avg}} + n^{1-p}}{s}$$

where e is the execution time of the current job, t_{avg} is the moving average of the execution time of all the previous jobs in the batch, n is the number of resources required to execute the batch, p is the waiting probability and s is the batch size. Taking into account resource utilization, execution time and throughput, this function gives each of these factors a non-trivial weightage.

The term $\frac{e}{t_{avg}}$ deals with execution time and average throughput. n^{1-p} deals with resource allocation. If the number of resources required is less, then the waiting probability will be less and therefore they should never become trivial, hence n is raised to a factor of $1 - p$.

This function is additive because all the factors are given a non-trivial weightage whereby all of them are included in computing the utility value. In order to maintain uniformity between the batches, the batch size has been divided.

For each batch, a forecasting method is applied with the existing utility values to compute the utility value for the next batch. The forecasting methods can depend on the trend present in these input utility data. If no trend is present, exponential smoothing can be applied and the presence of trend will shift the forecasting to Holt's Method.

```
/*Choosing Forecasting Method*/

if trend == true then
   method ← Holt's Method
else
   method ← Exponential Smoothing
end if
```

The computed forecasted value is compared with the actual utility value of the batch (an average of the utility values of all the jobs in the batch) to understand the computing facilities required to execute the batch.

```
/*Choosing Batch Strategy by Forecasting*/

if forecast value < average utility then
   strategy ← Rent
else
   strategy ← Self _ execute
end if
```

When the value of forecast is obtained, the system prepares itself to provide computing facilities atmost equal to that of the utility value forecasted. So if the forecasted value is less than the actual utility value, then the system was not prepared to handle that load as it forecasted only lesser facilities . So the cloud owner pays rent to an external cloud who can provide the necessary computing facilities for executing the batch. If found otherwise, the cloud owner will execute the batch with the facility he owns.

This approach only uses past information to decide the strategy even when the present data are made available. A sudden deviation in the batch load may not be captured in such a case and may result in incorrect forecasting and strategy results. So a game-theoretic approach, where the current data is also considered while deciding the strategy, is adopted.

In this approach, two utility values are calculated for two waiting probabilities, p_1 and p_2 where $0 <= p_1 <= 0.5$ and $0.5 < p_2 <= 1$. The strategies of Rent and Self-execute then randomly take the u_1 and u_2 values:

```
/*On Random Coin − toss*/

if Coin − toss == 1 then
   Rent ← u1
   Self _execute← u2
else
   Self _execute← u1
   Rent ← u2
end if
```

The above random strategy can be interchanged without much effect. So, for each batch of jobs a matrix $M_{BatchSize\,X\,2}$ is

created with utility values for Rent and Self-execute strategies. The maximum utility values for each strategy is then selected alongwith its corresponding rows resulting in a matrix $M'_{2\times 2}$. A method of mixed strategy calculation is applied to M' which chooses the strategy with the higher probability p.

The strategy given by forecasting and game theory may be different. In such a case, further decision has to be taken as to which strategy should be adopted. The method of standard deviation in each conflicting batch is applied about the points:

$\mu_1 = forecasted\ value$ /*Forecast approach*/

$\mu_2 = equilibrium\ value$ /*Game Theory approach*/

Where equilibrium value is obtained as:

$$p * utility_p + (1 - p) * utility_{1-p}$$

The final strategy is given by the approach that has minimum standard deviation.

/*In case of Conflict of Strategies*/

$\sigma_1 \leftarrow$ Standard Deviation about $\mu_1 \sigma_2 \leftarrow$ Standard Deviation about μ_2

if $\sigma_1 < \sigma_2$ **then**
  $strategy \leftarrow strategy_{Forecasting}$
else
  $strategy \leftarrow strategy_{GameTheory}$
end if

IV. Implementation

A batch of jobs (maximum 25 jobs per batch) having randomly generated values for execution time, number of resources required and waiting probability is initially created and 1000 such batches are initialized. The first seven batches are made to randomly adopt a strategy because of the window size taken in the approach.

The utility value for each job is calculated using the utility function and the average of these values are obtained for a batch. While applying a forecasting model, if a linear trend is observed in the average utility values for a window size of 7 consecutive batches, Holt's method is used. Otherwise exponential smoothing is used to calculate the forecast value for the next batch of jobs. A relaxation of 3 values has been provided in the forecasting model so that an almost equal distribution of Holt's and exponential model is observed. This deviation was experimentally chosen. The smoothing constants for the models were taken as $\alpha = 0.2$ and $\beta = 0.1$ to maintain stability of the forecast. The forecast value obtained from this approach is then used to decide the strategy using forecast by the algorithm for choosing batch strategy by forecasting.

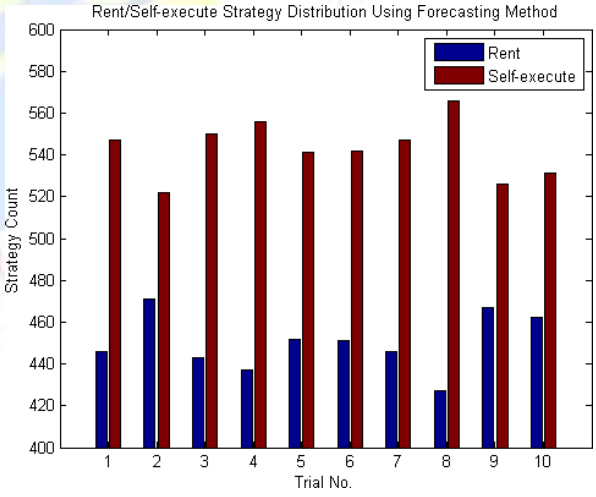
Further for game theory, the utility values for p_1 and p_2 probabilities are calculated for each batch and then by the selection process a 2×2 matrix is obtained which undergoes the mixed strategy calculation. The strategy with higher probability is the strategy using game theory.

From the above two approaches, two strategies are obtained. Whenever there is a conflict in the strategies obtained from the approaches, the strategy which gives minimum standard deviation with its respective means is finally chosen.

For example, consider that the cloud provider receives the i^{th} batch, then the forecast is done for the $(i+1)^{th}$ batch utility value which is the average of the utility values of all the jobs in the batch. If a trend is observed in these average values of $(i-7)^{th}$ batch to i^{th} batch, then the forecast for the $(i+1)^{th}$ batch utility is obtained using Holt's Method; else exponential smoothing is adopted. Then when the $(i+1)^{th}$ batch is received by the provider, the average actual utility of the $(i+1)^{th}$ batch is computed and is compared with the forecasted value previously obtained. Based on the algorithm for choosing batch strategy by forecasting, either Rent or Self-execute strategy is chosen. At the same $(i+1)^{th}$ batch, two utility values are then calculated for each job in this batch using p_1 and p_2 probabilities and a matrix is constructed. The selection process is then applied to this matrix to obtain the 2×2 matrix and finally the mixed strategy calculation is done to decide the Rent or Self-execute strategy using game theory. Thus the strategies for the $(i+1)^{th}$ batch is obtained by the two approaches. Consider a scenario whereby the Rent strategy is chosen through forecasting and Self-execute through game theory (or vice versa) for the $(i+1)^{th}$ batch, standard deviation is then applied and by the last algorithm on conflict of strategies, the final strategy to be adopted by the cloud provider is chosen for the $(i+1)^{th}$ batch.

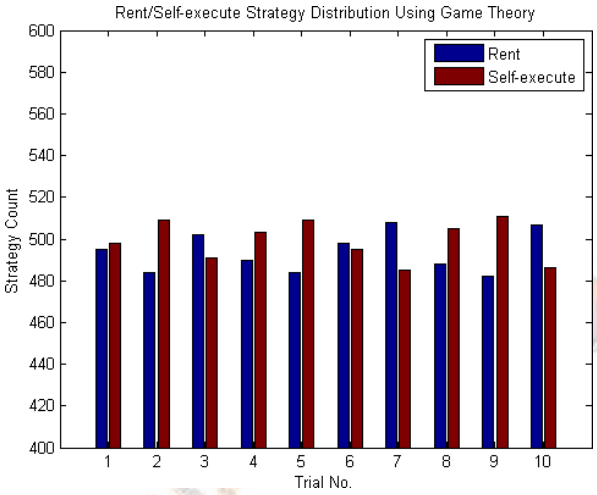
V. Results and Analysis

During forecasting, the strategy adopted is seen to be biased towards the Self-execute strategy. The utility values are observed to lie within the range 1 to 10 for the values generated. So when a sudden increasing trend in the utility values is seen (observed as a greater slope), then the next forecasted value will be much higher as it takes into consideration the difference between the values while forecasting in a linear trend. This will result in the Self-execute strategy being chosen frequently. The decreasing trend phenomenon does not happen frequently as this sudden decrease cannot be seen from a high value because of the common range observed. Hence however big a decreasing trend is observed, it still lies within the acceptable range.

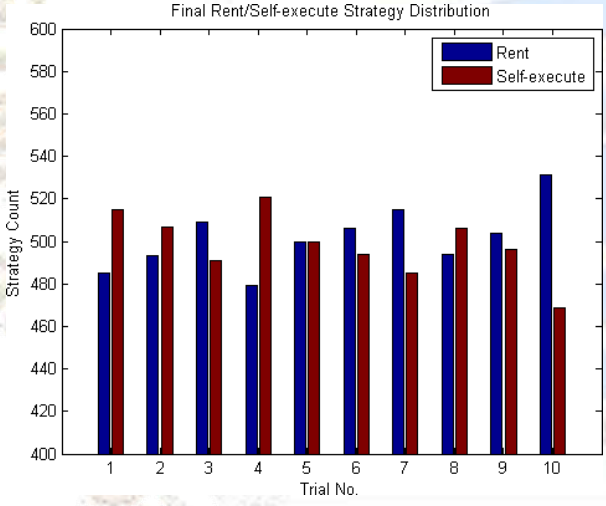


An almost equal distribution of strategies of Rent and Self-execute was observed with the game theoretic approach with no strategy overpowering the other. This is because

of the randomness in allocating the utility values to the strategies while constructing the matrix. This results in an unbiased evaluation of the matrix which gives an almost equal weightage to both the strategies.



The overall strategy also shows an almost equal distribution after resolving the conflicts. In case of conflicts, the final strategy obtained from the standard deviation method considers both forecasting and game theoretic approaches. In this case, the utility values of the jobs in the current batch is compared with the utility values obtained from forecasting and game theory. This method finally selects the strategy that does not deviate much from the expected value obtained from the above two approaches, thus making it easier for the cloud provider to provide the services required to execute the batch without deviating much from the services he already provides.



The results obtained above are with respect to the randomly generated data inputs. In the real environment, certain values like waiting probability depends on factors like network congestion, efficiency of the cloud, datacenter locations, etc. So during run-time, the results obtained may vary with respect to dynamic factors. The utility function can also be further enhanced to include economic factors for the cloud provider to get a clearer picture.

VI. Conclusion

In this paper, we have discussed one approach of how the illusion of infinite resources in Cloud Computing can be further optimized without incurring any loss for the cloud provider. QoS can be improved and thus it provides the user with better facilities. The cloud provider can decide whether he wants to pay rent for executing the load and prevent disappointing the user or just execute only what is possible. Forecasting followed by game theory gives a better approach of including the data available till the current point and will help in correctly deciding the strategy. From these results, a cloud provider can also check the number of times he is paying rent to execute the load and based on further calculations he can decide to own more facilities such that the frequency of renting will decrease and he may earn more profit. This can be modelled as a Ski-Rental problem and further worked out.

This paper provides one method of tackling the problem and it further opens up interesting avenues for improvement. The utility property can be changed to suit the agreement between the provider and the user. This approach considers execution time as one of the factors and this may change depending on the network traffic and other dynamic factors which will further complicate the utility property. There can be different utility functions that can be used considering more parameters. Other game-theoretic concepts and forecasting models can also be used to obtain the results. Since the avenues are deep and the domain is still growing, updates on this problem will keep on increasing until an optimal solution is reached.

References

[1] Mell, Peter, and Timothy Grance. "The NIST definition of Cloud Computing (draft)." *NIST special publication* 800 (2011): 145.

[2] Armstrong, Django, and Karim Djemame. "Towards Quality of Service in the Cloud." In *Proc. of the 25th UK Performance Engineering Workshop.* 2009.

[3] Endo, Patricia Takako, Andre Vitor de Almeida Palhares, Nadilma Nunes Pereira, Glauco Estacio Goncalves, Djamel Sadok, Judith Kelner, Bob Melander, and J. Mangs. "Resource allocation for distributed cloud: concepts and research challenges." *Network, IEEE* 25, no. 4 (2011): 42-46.

[4] Armbrust, Michael and Fox, Armando and Griffith, Rean and Joseph, Anthony D. and Katz, Randy H. and Konwinski, Andrew and Lee, Gunho and Patterson, David A. and Rabkin, Ariel and Stoica, Ion and Zaharia, Matei. "Above the Clouds: A Berkeley View of Cloud Computing." *Technical report,* no. UCB/EECS-2009-28 (2009).

[5] Vinothina Sr, V. "A Survey on Resource Allocation Strategies in Cloud Computing." *International Journal* (2012).

[6] Goncalves, Paulo, R. O. Y. Shubhabrata, Thomas Begin, and Patrick Loiseau. "Dynamic Resource Management in Clouds: A Probabilistic Approach." *IEICE Transactions on Communications* 95, no. 8 (2012): 2522-2529.

[7] Han, Zhu, Dusit Niyato, Walid Saad, Tamer Baar, and Are Hjrungnes. *Game theory in wireless and communication networks: theory, models, and applications.* Cambridge University Press, 2011.

[8] Kalekar, Prajakta S. "Time series forecasting using Holt-Winters exponential smoothing." *Kanwal Rekhi School of Information Technology* (2004).

[9] Ski Rental problem, http://en.wikipedia.org/wiki/Ski_rental_problem.

[10] Standard Deviation, http://en.wikipedia.org/wiki/Standard_deviation.

[11] Forecasting Method, http://nptel.iitm.ac.in/courses/110106045.

[12] SAS Products and Solution Documentation.