Autonomous Resource Provision in Virtual Data Centers

Noha Elprince
D.R. Cheriton School of Computer Science, University of Waterloo, Canada.
Email: noha.elprince@uwaterloo.ca

***Abstract*—In recent years the advances in cloud computing and virtualization have created the need for autonomic resource provision. The correct provisioning of resources is a difficult task due to variations and uncertainty in workload demands. Most data center workload demands are very spiky in nature and often vary significantly during the course of a single day. Because the resource availability in a data center is generally unpredictable due to the shared feature of the cloud resources and because of the stochastic nature of the workload, severe service level agreement (SLA) violations may occur frequently. To overcome this problem, researcher's attention is diverted towards developing dynamic resource management strategies. In this paper, an autonomic resource controller is proposed that dynamically controls the resource allocation for data center's virtual containers. The controller has two parts: A resource modeler that models the non-linearity of the system by employing different Machine Learning techniques allowing the datacenter to allocate the appropriate resources and a resource fuzzy tuner that dynamically tunes the allocated resources using fuzzy logic to sustain the desired performance taking into consideration the enforcing of service differentiation among clients. Experimental results on a real data center dataset showed that the proposed resource controller can predict future resource needs while still sustaining performance goals stated in the SLA. Also, using the bagging and the boosting techniques along with model tree classifiers was demonstrated to be promising in terms of accuracy and performance.**

I. Introduction

Data Centers suffer from growing complexity and increased demand for its computational resources. Managing these resources is not a trivial task especially with the use of virtualization. Virtualization is a technique that allows a computational resource to be partitioned on multiple isolated execution environments, which are called Virtual Machines (VM). Adopting virtualization in a data center offers greater flexibility of resource allocation across physical resources. Also, it enables service providers to offer fast provision of VMs according to the need and charge clients in a "pay-as-you-go" basis, which gives the cloud an elastic nature [1]. So, due to the ongoing rapid growth in complexity and scalability, the IT world is motivated to focus on autonomic computing systems that aim to lower the human interaction in managing complex computing systems. In virtualized data centers, self-scaling is a desirable feature that guarantees gaining the desirable cloud's "elasticity" feature. This feature can be attained via dynamic virtualized resource provision; however, provisioning of scalable and virtualized resources is considered a complex task. This complexity is due to the uncertainty in predicting the resource demands. As a result, two problems appear which are: resource over-provision (anticipating more resources than the amount consumed) and resource under-provision (anticipating less resources than the amount consumed). Resource over provisioning may result loss in data center's profit margin, while resource under provisioning may result in paying penalties to clients by the data center as a result of violation of SLA (service level agreement). Another non-trivial problem that usually accompanies employing virtualization is service differentiation which means providing different service quality to different clients, each with different SLA.

In this paper, an autonomic resource management controller is proposed that effectively allocates the sufficient amount of resources needed for a given application without violating the SLA. Such violation may results in large penalities that should be given by the data center to the clients.The main motivation beyond this work is to minimize such penalities and improve data center management. The proposed prototype is tested using real traces of Los Alamos National Lab (LANL) [2].

The main contributions of this paper are:

- Design of an autonomic resource controller capable of learning adaptively.
- Performance comparison between different prediction Machine Learning techniques.
- Satisfying service differentiation and resource tuning via a fuzzy inference resource tuner.

The rest of this paper is organized as follows: Section II describes in detail the design of a proposed autonomic resource controller. Section III presents experimental evaluation of the proposed prototype. Section IV examines the related work. Section V concludes the paper and highlights the future work.

II. Architecture of Autonomic Resource Controller

Initially, the client has to provide the data center with the desired response time and type of application that needs to be hosted in the cloud. The main task of the proposed resource controller is to predict the set of resources needed by an application (job) running in a virtual container. Specifically, the system predicts the amount of CPU usage in seconds, memory used in KBs and latency expected in seconds. In order to perform this task, the controller needs to learn the behavior of the VM under various workloads. Also, predicted values should be tested in real time settings. Then, a feedback

978-3-901882-50-0

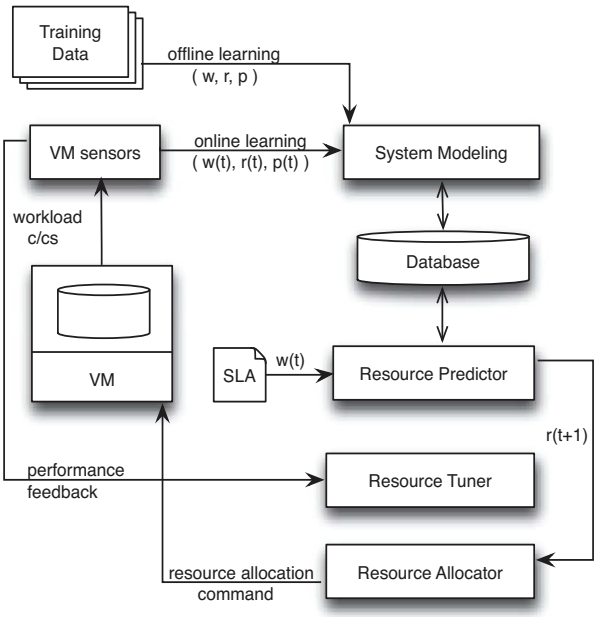

Fig. 1: Autonomic Resource Controller: w is the input workload, r is the input resource needed, p is the application performance results from executing w, and r(t+1) is the predicted resource usage.

performance report should be fed to the controller in order to decide whether resources need more tuning or are sufficiently allocated. Figure 1 shows the proposed resource controller model. The system consists of five main modules: (i) System Modeling: models the relationship between the workloads and the resources used. The model is built initially via offline training, then online training afterwards. (ii) Resource Predictor: dynamically predicts the required amount of resources needed to maintain the desired QoS that is requested by the client. (iii) Resource Allocator: allocates the adequate resources after being tuned by the resource tuner.(iv) Resource Tuner: takes the decision whether to tune resources or not, taking into consideration fullfilling service differentiation among different clients. (v) Adaptive learning via the VM sensor: monitors the workload achieved performance and its resource consumption. The next subsection explains the design of each of these module in more details.

A. System Modeling Using Machine Learning Techniques

The idea of learning from successful designs is already well known. Successful design is defined to be the successful jobs that have been executed with a certain amount of resources and proved to be successful in terms of both: normal termination due to sufficient allocated resources and fullfilling the client's anticipated performance (response time). So, by modelling successful jobs, the required amount of resources needed for a particular job can be predicted with a guarantee of fullfilling the defined success metric. Machine Learning (ML) is employed because it is able to deal with the complex nature of the data and to detect the dependency between the inputs and the outputs efficiently. However, careful selection of the right ML method along with its inputs and outputs is quite essential to gurantee good prediction results. The stages of the system modeling module will be as follows:

1) Dataset: The dataset used is a real computing center workload trace. This trace contains two years records produced by the DJM software running on the 1024-node Connection machine (CM-5) at Los Alamos National Lab (LANL). LANL is one of the largest science and technology institutions in the world. The log [2] contains detailed information about resource requests and use, including memory and CPU time. It also contains data on the user, executable, project, and submit, start, and end times. The cleaned version contains 201,387 jobs, collected throught two years. The hardware used was 1024-node Connection Machine CM-5 from Thinking Machines.

2) Data Preprocessing: Unsuccessful jobs were filtered with status: failed or aborted. Also, some resources were noticed to have unrealistic negative value (-1) resulting from missing values. These values have no significance in the job analysis and can have bad impact on the accuracy of the proposed model. The description of the dataset states that negative values and zeros were assigned to "run time" and "wait time" when their values are less than 0.5 sec. Also, stepwise regression was chosen for feature selection. It assisted us in identifying the most influential features in the model. The main advantage of stepwise technique is that it lowers the total number of potential features to a more mangeable number and provides a basis for examining the dependency relationship between different features [3]. In this work, the number of features were reduced from 18 to only 6 features. The selected features were: avg_cpu_time_used, no_allocated_procs, used_mem, run_time, wait_time, queue_no. Total CPU usage was the product of avg_cpu_time_used and no_allocated_procs. Also, total memory usage was the product of mem_used and no_allocated_procs in kilobytes(KB). The response time was calculated as the sum of run time and wait time (latency) in seconds. The latency represents the disk I/O wait time and network latency. Finally, data normalization was performed to reduce the noise in the dataset.

3) Statistical Analysis: In order to obtain a general overview of the relationship between workload performance characteristics and system resources: CPU-usage, memory usage and latency time, statistical analysis was performed as illustrated in Figure 2. In each subfigure, the x-axis illustrates the parameter under inspection and the y-axis illustrates the resource(s) allocated to the given setting. Subfigure 2(a) and 2(c) show how the system performance, in terms of the run time and response time affects the allocated resources in terms of CPU and memory. Intuitively, they show that whenever the run time and the response time decrease, there is a corresponding increase in resource utilization. Subfigure 2(b) shows how the workload type represented by the feature"queue no" in the dataset, affects the system resources. It is noticed

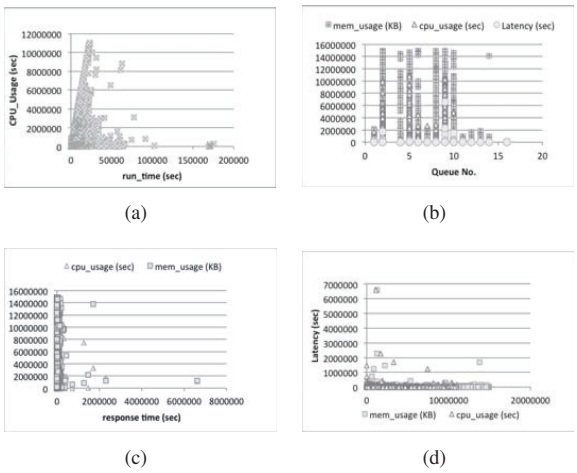

Fig. 2: Relationship between workload performance characteristics and system resources.

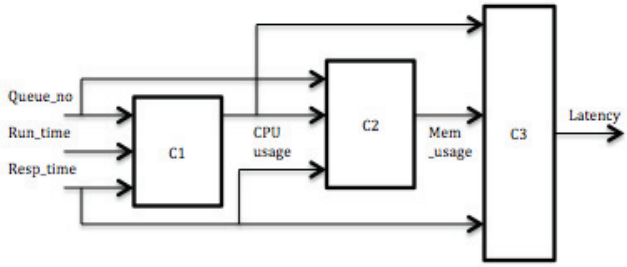

Fig. 3: Proposed Multi-stage Resource Prediction Model

that the greatest effect in resources was in the CPU-usage followed by memory usage then finally, the latency. Also, some correlation were noticed to exist between different system resources. For example, it is a good performance practice that the amount of memory allocated is synchronized with the amount of allocated CPU shares. Also the Latency time is correlated with CPU and memory usage as illustrated in subfigure 2(d). These resultant correlations are mainly used in designing the proposed prediction model discussed in the following subsection. Moreover, this analysis enforces the idea of using ML for building a prediction model due to the non-linear nature of the data and the inability to fit a standard curve for the data.

4) Prediction Model: The proposed cascaded multi-stage predictor model is illustrated in Figure 3. The predictor has three inputs that represent the workload characteristics: response time, run time and queue number. The later indicates the workload type (e.g. interactive, development, production, etc).The three predicted outputs are: CPU-usage (sec), Memory-usage(KB) and latency(sec). The idea beyond using cascaded classifiers is the dependency that exist between CPU usage, memory usage and latency. This dependency was demonstrated in subfigure 2(d). The proposed model is designed to have three working modes: offline learning, adaptive learning and predicting mode. Section II-E tackles the adaptive learning mode and the predicting mode is discussed in section II-B.

Offline Learning: The offline learning is initially started using the historical collected data of the successful jobs only after discarding the unsuccessful ones. The selection of an optimal classifier for modelling this complex data is non-trivial. So, several classifiers were investigated and reported their accuracy. In this work, different types of ML techniques were used: regression and classification. For the regression, the input "queue_no" is splitted into 16 inputs representing 16 different queues. Each queue no. represents a specific type of workload and takes a dual value of 1 or -1 which indicates it's existence or not respectively. The rationale beyond this, is that regression uses grouping which groups similar data together. So, approximately similar ranges of queue_no.s will be grouped together and may consequently lead to an inaccurate model.

B. Resource Predictor

The client requests hosting a specific type of application (workload) with a pre-specified performance metrics (response time and run time). The prediction helps the data center to allocate an initial estimate for the resources needed to fulfill the clients request. The rate of prediction is accompanied by the coming of the client to the data center. The prediction results are sent to the resource allocator to allocate the requested resources. However, the latency time stands for the disk I/O wait time and the round trip time taken by the network. Assuming that most data centers use low latency network ($< 25\text{msec}$). So, the main concern is to handle the disk I/O wait time. Therfore, latency time is used to calculate the rotational speed of the hard disk using the following formula:

$$LatencyTime(sec) = \frac{1}{RotationalspeedinRPM/60} \times 0.5 \quad (1)$$

Then, given the rotational speed, the IOPS (I/O requests per sec) for each VM can be determined using the data center hardware specifications [4], [5].

C. Resource Allocator

It calculates the normalized error in resource allocation (at time interval k) as follows:

$$RespTimeError(k) = \frac{RespTime(k)_{ref} - RespTime(k)_{obs}}{RespTime(k)_{ref}} \quad (2)$$

The RepTime(k)$_{ref}$ is the response time requested by the client, and the RepTime(k)$_{obs}$ is the monitored response time after resource allocation. K is the control time interval. Then, it feeds the error(if exist) into the resource tuner. After that, the resource allocator takes the feedback from the tuner "ResAdjustment" and sends a command to VM with the appropriate decision.

TABLE I: Resource Allocator I/O features and their member ship functions

I/O FEATURES	MF LABELS
RESPTIMEERROR	LOW, MED, HIGH
CLIENTCLASS	BRONZE,SILVER,GOLD
STATUS	UNDER PROVISION,OVER PROVISION
RESOURCEDIR	SDH,SDM,SDL,NA,SUL,SUM,SUH

TABLE II: Fuzzy Inference System Rules for (resDir) resource direction estimation

		RespTimeError		
		Low	Medium	High
Client Class	Gold	ϕ / SUL	SDM / SUM	SDH / SUH
	Silver	SDL / ϕ	SDM / SUM	SDH / SUH
	Bronze	SDL / ϕ	SDM / ϕ	SDH / SUH

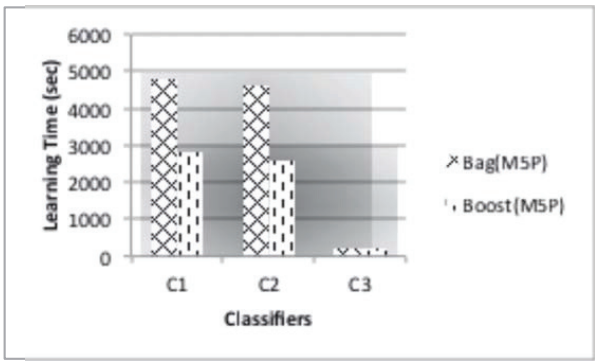

Fig. 4: Learning Time Comparison

D. Resource Tuner

The resource tuner is responsible of taking the decision whether to speed up or step down the performance by increasing or decreasing the amount of allocated resources. Also, it may decide to take no action if the system performs well. A resource tuner using fuzzy inference system rule-based system (FIS) has been implemented. One of the virtues of fuzzy logic is that it allows nonlinear relationships to be expressed by a set of qualitative "if-then" rules [6]. The FIS consists of four components: The fuzzifier receives the inputs (RespTimeError, ClientClass, Status) and maps them into fuzzy sets using membership functions (MFs) represented by gaussian functions. The grades of membership functions are qualitative and are set by the expert using linguistic variables. The output is a numeric real value for the "Resource direction". Table I shows the inputs and outputs along with their membership function labels (SDH = step down high, SDM = step down medium, SDL = step down low, SUL=step up low, SUM = step up medium, SUH = step up high). The rules are set by experts from the data center as shown in Table II where the total number of rules is 18. The upper and lower cell partition represents the status of "over provision" and "under provision". This status can be easily known via monitoring the sign of the difference in RespTimeError. If sign is positive , then status is "over provision" and if negative then status is "under provision". The grades of membership of each attribute (high, medium, low) are adjusted by experts in the datacenter. The fuzzy inference engine uses the rules stored in the rule-base to infer from input fuzzy sets to output fuzzy sets. The defuzzifier aggregates the fuzzy rule consequents and outputs the weighted average of all rule outputs. The output is a signed crisp value that represents the estimated resource direction (resDir) for the next control time interval. If the output is positive, then this means resource should speed up, and If negative, then resource is step down. The ϕ value in the table means a don't care state that requires no action to be taken.

a) Design Assumptions: (i) Each client is assigned a Virtual Container (VC) with a specified number of resources. These resources can be upgraded from the shared resource pool. (ii) The resDir ranges from -1 to +1 with MFs: low, med, high for both speed up (+ve side) and for step down (-ve side).This resDir is viewed as a percentage of the resource that should be utilized in the virtual container (VC). This means if resDir is 0.5, this means that 50% of the resources in the VC should be employed. (iii) Each resource has a specific weight (for example, Wcpu=1, Wmem = 0.5 , Wlatency=0.4). (iv) If the weighted (resDir) reaches 0.9 , then this means that the VC is currently utilizing 90% of its resources, then a command is sent to the resource global controller in the VM asking for more resources. The design of the global controller is outside the scope of this paper.

b) Actual resource Calculation: For example, assume that RespTimeError is medium and client class is gold and status is under provision then the fuzzy tuner will compute resDir is0.5 (Speed Up Medium). If the total CPU usage resources for the VM container VC_{res} is 100 shares and the current shares used (currDir) is 30 (monitored by VM sensor). Then, the weighted resDir can be computed as follows:

$$\begin{aligned} weightedresDir &= resDirection \times W_{res} \times VC_{res} \\ &= 0.5 \times 1 \times 100 = 50 \end{aligned} \quad (3)$$

Then, the amount of resource adjustment needed for the next control interval (k+1) can be calculated as follows:

$$\begin{aligned} resAdjustment(k+1) &= weightedresDir - currDir \\ &= 50 - 30 = +20CPUshares \end{aligned} \quad (4)$$

E. Adaptive Learning

New incoming data will be fed into the prediction model by two ways (depending on the prediction model used): (i) Online learning directly via incremental clustering or re-clustering (if clustering is used as in Sugeno-FIS). (ii) Active learning via storing new data in a database, until a certain threshold is reached. Then an ECA (event-condition action) rule is fired, initiating re-modeling using ML techniques. Also, new data center policies can be accomodated by the FIS user friendly rule editor, which allow the experts to modify rules online.

TABLE III: Performance measures for numeric prediction models

Prediction Error						
ML model	Type	RMSE	RAE	CC	RRSE	MAE
Linear Regression	C1	0.0024	50.33%	0.70	70.92%	0.0008
	C2	0.0023	57.29%	0.71	70.37%	0.0001
	C3	0.0026	58.15%	0.98	15.83%	0.0003
Sugeno FIS (FCM)	C1	0.0022	47.15%	0.63	65.06%	0.0007
	C2	0.0021	53.96%	0.63	64.85%	0.0009
	C3	0.0025	55.91%	0.72	66.59%	0.0009
Sugeno FIS (SUB)	C1	0.0021	44.89%	0.66	62.20%	0.0009
	C2	0.0012	51.06%	0.66	62.65%	0.0002
	C3	0.0011	53.93%	0.85	32.30%	0.0002
REPTree	C1	0.0019	33.78%	0.83	55.40%	0.0006
	C2	0.0018	38.56%	0.84	54.88%	0.0007
	C3	0.0026	39.39%	0.78	74.79%	0.0002
M5P (Model Tree)	C1	0.0020	35.31%	0.79	60.89%	0.0006
	C2	0.0018	39.23%	0.84	53.82%	0.0007
	C3	0.0003	11.82%	0.99	7.79%	0.0001
Boosting (M5P Model Tree)	C1	0.0020	34.59%	0.80	60.13%	0.0006
	C2	0.0018	39.20%	0.84	53.72%	0.0007
	C3	0.0003	10.99%	0.99	7.96%	0.0001
Boosting (REPTree)	C1	0.0019	33.78%	0.82	57.80%	0.0006
	C2	0.0017	38.43%	0.85	52.55%	0.0007
	C3	0.0028	73.89%	0.72	80.44%	0.0003
Bagging (REPTree)	C1	0.0018	32.56%	0.84	54.62%	0.0005
	C2	0.0017	36.38%	0.84	53.83%	0.0007
	C3	0.0027	45.89%	0.79	76.61%	0.0002
Bagging (M5P)	C1	0.0018	32.57%	0.84	54.62%	0.0005
	C2	0.0017	36.38%	0.84	53.83%	0.0007
	C3	0.0003	11.82%	0.99	7.79%	0.0001

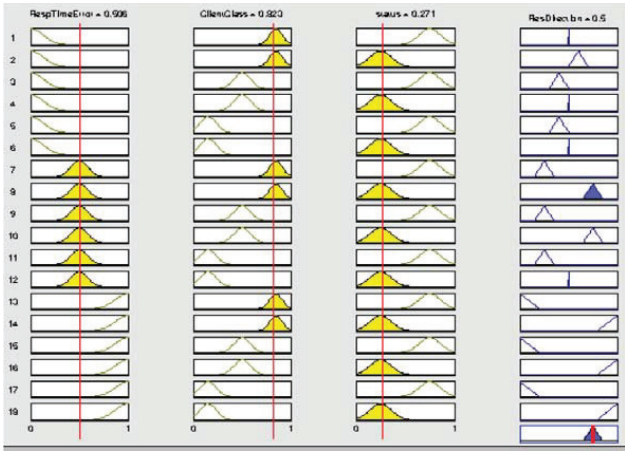

Fig. 5: Resource Tuner FIS: case of Golden client

III. Experimental Evaluation

A. Setup

In implementation, the WEKA package [7] is used for all the ML classifiers, except the Sugeno FIS, which is implemented using the fuzzy logic tool box under Matlab. All the experiments were carried on a MacBook Pro machine with Intel Core 2 Duo having a processor speed 2.4 GHz and memory 4GB.

B. System Model Validation

For validating our prediction model, the standard 10-fold cross validation technique is used.

C. Results and Discussion

The results are shown in Table III. In measuring the classifiers (C1, C2, C3) accuracy, Witten and Frank [8] advise to report the following metrics: RMSE = Root Mean Square Error, RAE = Relative Absolute Error, RRSE = Relative Root Mean Square Error, CC = Correlation Coefficient, MAE = Mean Absolute Error. A complete description of each metric is found in [8]. In Table III, it is clear that the best results for the predictors was achieved via the Bagging (M5P) that employs "Model tree". As expected previously, the bagging technique excels over the boosting due to its ability to handle the noisy data.The fuzzy models performs poorly probably because it employs clustering technique. Clustering may be unsuitable for our data set due to the concentration of our data set in specific regions and the low spread of the data over the entire space. So, a conclusion can be drawn from the obtained results, that the best ML technique investigated to model resources is classification using "Model trees" in terms of accuracy. However, decision which model to use should consider also time taken to build the model (learning time). A comparison between bagging and boosting was performed using M5P classifiers as illustated in Figure /reflearntime.

It is clear that the learning time for bagging technique is about double that of the boosting for C2 and C3. So, the boosting technique may be favored if one is keen to have a good learning performance rate.

D. Fuzzy Tuner Validation

The validation of the tuner is performed by testing cases. This can be tested using the fuzzy rule viewer in the fuzzy logic toolbox of the Matlab as shown in Figure 5. The first three columns represent the input values of the fuzzy rule-based inference system and the fourth column represent its output value. The shaded triangles in each column represent the specific gaussian MF fired among the 19 embeded rules, and the shaded triangle in the fourth column represent the weighted average of all the fired inputs. The red vertical line in the first three columns allows the adminstrator to adjust the inputs online by dragging it back or forward. For example, a fixed value is set for input RespTimeError is "medium" and the status is "underprovision". The client class is altered to test service differentiation. In case the client class "Bronze", then the output recieved of the ResDirection is approximately zero indicating no action should be taken. If the client class is "Silver", then the output recieved is +0.5 which means SUM (speed up medium). If the client class "Gold", then the output recieved is +0.5 which means SUM (speed up medium). The result is shown in the top of the fouth column in Figure 5. This confines with the rule table II column 2.

IV. Related Work

The related work can be divided into different methods used in autonomic resource allocation:

A. Resource management using Fuzzy Approach

Xu et al. [9] presented a two-layered approach to managing resource allocation to virtual containers sharing a server pool

in a data center and evaluated the scheme on a test bed running VMware ESX Server. The local controller offers a solution similar to the fuzzy controller studied in this article, however, a fuzzy controller is implemented using two different clustering approaches (FCM and SUB) and a comparison between them was created. Rao et al. [10] employs fuzzy rule-based inference system in tuning error resulted from deviation of response time using a feedback sontrol system. Our work is similar to that but differs in employing service differentiation through our fuzzy rule-based inference system instead of using a static objective function and employing optimization techniques under constraints.

B. Utility-based self-optimizing approach

Utility functions are traditionally used by many autonomic resource management systems as a way to measure the satisfaction of an application with respect to its level of service (SLA) satisfaction [11] proposes a generic architecture for using utility function with a two-level architecture that separates application-specific managers from a resource arbitration level. Menasce et al. [12] proposes a utility-based autonomic controller for dynamically allocating CPU power to VM. They attempt to maximize a global utility function. However, they do not take into account the possibility of provisioning additional VMs to an application and restrict their work to homogeneous application workloads modeled with a queueing network.

C. Model-based approach based on performance modeling

The objective is to find the optimal configuration for maximizing the global utility in the next interval under some constraints. Stochastic heuristic methods can be employed as Wang et al. [13] proposes a non-linear optimization strategy using simulated annealing to determine the configuration (i.e. VM quantity and placement) that maximize global utility.

D. Machine Learning approach

Xiong et al. [14] employs boosting trees for predicting total average cost. They associated the penalty cost with the response time and tried to minimize it using grid search optimization technique. In our view, associating the penalty cost with the response time is not efficient way of utilizing resources. The reason is that minimizing the cost means minimizing the response time as well even though the customer is satisfied with the current response time. Our work also uses boosting algorithm. However, the bagging algorithm was also employed and a comparison between them was conducted in terms of accuracy and learning time.

V. Conclusions and Future Work

Several sophisticated ML techniques were investigated aiming to find a good technique that achieves a good accuracy. The aim of this work is to aid the data center to allocate the right amount of resources needed to fullfill the clients requests. These requests are written in "SLA" between the client and the service provider. The design of the proposed resource tuner makes sure to handle any deviations that may occur in the allocated resources. This deviation may be due to prediction inaccuracy or an abrupt change in the workload characteristics. Also, the proposed FIS design for the tuner makes it a general design schema that can be employed easily in any data center due to two reasons: Its flexibility in modifying rules that is suitable to the buisness rules and the expressive power of fuzzy logic in employing liguistic variables that are easy to understand. The results show that the best ML technique to model resources is the classification using Model trees via the Bagging technique. Moreover, the proposed fuzzy tuner demonstrates the ability to offer service differentiation among different clients. Also, SLA penalities due to violations are expected to be reduced through adaptive learning and the resource online tuning. In future, a real time simulator may be used to test the performance our control system. Fuzzy petri net may be used for validating the rules in the rule base. Also, in future it will be interesting to implement an adaptive SLA that reflects the clients evolving needs over time in future and autonomously measure the customer satisfaction on the cloud.

References

[1] P. Marshall, K. Keahey, and T. Freeman, "Elastic site: Using clouds to elastically extend site resources," in *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, ser. CCGRID '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 43–52. [Online]. Available: http://dx.doi.org/10.1109/CCGRID.2010.80

[2] "Parallel Workloads Archive," http://www.cs.huji.ac.il/labs/parallel/workload/l_lanl_cm5/index.html, [online].

[3] J. Mark and M. A. Goldberg, "Multiple regression analysis and mass assessment: A review of the issues." *Appraisal Journal*, vol. 56, no. 1, p. 89, 1988. [Online]. Available: http://search.ebscohost.com.proxy.lib.uwaterloo.ca/login.aspx?direct=true&db=bth&AN=5362393&site=ehost-live&scope=site

[4] I. Ahmad, "Easy and efficient disk i/o workload characterization in vmware esx server," in *Proceedings of the 2007 IEEE 10th International Symposium on Workload Characterization*, ser. IISWC '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 149–158. [Online]. Available: http://dx.doi.org/10.1109/IISWC.2007.4362191

[5] S. Kaneko, "Evolution of magnetic disk subsystems," *Journal of Magnetism and Magnetic Materials*, vol. 134, no. 23, pp. 217 – 222, 1994. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0304885394001448

[6] D. Dubois, H. Prade, and R. R. Yager, Eds., *Fuzzy information engineering: a guided tour of applications*. New York, NY, USA: John Wiley & Sons, Inc., 1997.

[7] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009. [Online]. Available: http://doi.acm.org/10.1145/1656274.1656278

[8] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

[9] J. Xu, M. Zhao, J. Fortes, R. Carpenter, and M. Yousif, "Autonomic resource management in virtualized data centers using fuzzy logic-based approaches," *Cluster Computing*, vol. 11, no. 3, pp. 213–227, Sep. 2008. [Online]. Available: http://dx.doi.org/10.1007/s10586-008-0060-0

[10] J. Rao, Y. Wei, J. Gong, and C.-Z. Xu, "Dynaqos: model-free self-tuning fuzzy control of virtualized resources for qos provisioning," in *Proceedings of the Nineteenth International Workshop on Quality of Service*, ser. IWQoS '11. Piscataway, NJ, USA: IEEE Press, 2011, pp. 31:1–31:9. [Online]. Available: http://dl.acm.org/citation.cfm?id=1996039.1996074

[11] G. Tesauro and J. O. Kephart, "Utility functions in autonomic systems," in *Proceedings of the First International Conference on Autonomic Computing*, ser. ICAC '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 70–77. [Online]. Available: http://dl.acm.org/citation.cfm?id=1078026.1078411

[12] D. A. Menasce and M. N. Bennani, "Autonomic virtualized environments," in *Proceedings of the International Conference on Autonomic and Autonomous Systems*, ser. ICAS '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 28–. [Online]. Available: http://dx.doi.org/10.1109/ICAS.2006.13

[13] X. Wang, Z. Du, Y. Chen, S. Li, D. Lan, G. Wang, and Y. Chen, "An autonomic provisioning framework for outsourcing data center based on virtual appliances," *Cluster Computing*, vol. 11, no. 3, pp. 229–245, Sep. 2008. [Online]. Available: http://dx.doi.org/10.1007/s10586-008-0053-z

[14] P. Xiong, Y. Chi, S. Zhu, H. J. Moon, C. Pu, and H. Hacigumus, "Intelligent management of virtualized resources for database systems in cloud environment," in *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering*, ser. ICDE '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 87–98. [Online]. Available: http://dx.doi.org/10.1109/ICDE.2011.5767928