International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-2, December, 2019

Performance Testing in A Multi Tenant Cloud Architecture using Genetic Algorithm

Vishnu Shankar.S, Sajidha.S.A, Nisha.V.M, Sathis kumar.B

Abstract:*Recent researches in cloud discusses about the application response testing, performance testing, security testing and many more, but still there is a lack of researches addressing issues like resource utilization and user interactions in cloud SaaS testing. The load on the cloud, SaaS instance keeps varying dynamically with respect to time, it is difficult to find the exact load at a particular interval of time. One does not know where to look for the solution and where to start, this made SaaS instances non deterministic in nature. In order to find a solution for such non deterministic problems, we make use of Genetic Algorithm which is considered as a good solution for non-deterministic problems.We determine the optimized resources that a cloud instance, would need to manage the dynamic load at all times. Toaddress the resource utilization of a group of users in Multi-Tenant Architecture (MTA), we adopt Genetic Algorithm which uses a popular technique, called neighborhood search and instance ranking policy. The basic concept of this paper is to explore the neighbors of an existing solution, that is considered as the solutions which can be obtained with a specific operation on the base population. In addition to that,this paper discusses about the ranking of all the available population and select the most highly ranked one. Instance ranking policies are aimed at minimizing the number of nodes in use or maximize the resources available to each node in an instance.*

Keywords: *Software-as-a-Service (SaaS), Virtual Machines, Cloud Multi Tenant Architecture (MTA), Genetic Algorithm (GA), Non-determinism.*

I. INTRODUCTION

Fast advances in cloud computing have brought many new business opportunities globally. As a result, cloud resource allocation and their effective utilization have become a challenge. Although there are a number of recent researches addressing cloud testing issues, like application performance testing, response testing and security testing.But there are very few researches focusing on specific problems like effective resource utilization, user data synchronization, in SaaS testing. To address the resource utilization in the cloud, we adopt Genetic Algorithm (GA) in our work.
In Genetic Algorithm, neighborhood search, has been used to explore a lot of combinatorial optimizations. The basic idea is to explore 'neighbors' of an existing population by applying a specific operation on it.

Revised Manuscript Received on December 30, 2019.
* Correspondence Author
Vishnu Shankar*, GE Power, Bangalore, India.
Email:vishnushankar.sf@gmail.com
Sajidha.S.A*, Vellore Institute of Technology, Chennai, India.
Email:sajidha.sa@vit.ac.in
Nisha. V.M*, Vellore Institute of Technology, Chennai, India.
Email:nishavm@vit.ac.in
Sathis kumar. B,Vellore Institute of Technology, Chennai, India.
Email:sathiskumar.b@vit.ac.in

Variousapproaches for the cloud resource allocation using time scheduling are discussed in[1], [2], [3]. Most of these algorithms do not consider the dynamically varying load on cloud instances. But the major complexity lies here. That has made cloud resource allocation non-deterministic in nature. Non-deterministic problems are known to be NP-Hard. Many heuristic algorithms like genetic algorithm are there to solve such problems to a near-optimal solution. However, there is a lack of practical solution for cloud computing systems because most of them are are multiple-QoS constrained. The existing genetic algorithms discussed in [4] are used in the proposed research work because that may provide an acceptable solution and an optimal solution. The method discussed in [5] on the other hand, uses slightly more sophisticated algorithms for cloud instance optimization, where for each immediate request, it ranks all the available instances and selects the most highly ranked one. Instance ranking policies are aimed at minimizing the number of nodes in use or maximize the resources available for each node in an instance. Finding the optimum configuration for nondeterministic entity is one of the biggest challenges. If this is not configured properly, it might lead to either under-utilization or overloaded. This may interrupt the user experience. Optimization in cloud services [6], [7] Hence an efficient strategy has been proposed and implemented using Genetic Algorithm [8] to identify the optimum solution for the resource space. Proving the real optimum is not possible every time and hence the solution found by genetic algorithm is considered as a good solution.

II. PROPOSED APPROACH

The aim of the proposed work is to provide an efficient algorithm to monitor the performance and resource allocation of cloud, SaaS Instances [9],[10]. The main idea behind multi-tenancy in a SaaS system is to provide a single code-based polymorphic application that is customizable and scalable to meet the demands of different types of applications in multi-tenant environment. It is a trend for SaaS vendors to construct SaaS programs with high multi-tenancy to achieve cost-reduction in software production and maintenance.One does not know where to look for the solution and where to start, to find a solution for such problem we make use of Genetic Algorithm which is often considered as a good solution, to determine the optimized resources that a cloud SaaS application would need. Dynamically varying load in the cloud is discussed in research work mentioned in [11], [12], [13], [14]. To test the cloud instance performance over dynamically varying load, we create three separate tenants where each tenant has an application instance running for its user, on this instance, we apply virtual load by making use of a tool named "WapT pro" to calculate

various parameters likeAverage response time, overall bandwidth, successful hits, errors and etc., these results are tabulated and used as input for the genetic algorithm.
Inorder to relate with the useage of the biological terms, herein our case, we make an assumption between actual cloud instance test results and genetic algorithm biological values, like successful hits per second is considered as uniform rate, average response time is considered as mutation rate, active users are considered as tournament Sizeas shown inTable 1. The above obtained values are given as an input to our genetic algorithm and optimal performance of each tenant is obtained, and the results are compared with three other tenants. Finally the tenant that achievesthe best fit in the least number of generations is considered as the best working tenant.This assumption is made by considering the instance parameters that best fits the biological value genetic algorithm.

Table 1:

Actual Values Obtained	**Biological consideration**
Successful Hits/ Sec (SFH)	Uniform Rate (SFH)
Average Response Time (ART)	Mutation Rate (ART/2)
Active Users (AU)	Tournament Size (AU)

These values are passed as input to our genetic algorithm and optimal performance of each tenant is obtained, and the same is repeated for other two tenants also, and the results are compared with each other tenant. Finally the tenant that achieve the best fit in the least number of generations is considered as the best working tenant.

III. DESIGN OF THE SYSTEM

A. System Architecture

The structure, behavior and overall views of the system is represented using the system Architecture which is a conceptual model.
Fig.1. shows the architecture where multiple tenants are created on "Mirantis Openstack provider" where each tenant has its own instance running for a dedicated application, and Software testing tools are used on these instances to test their application performance and their values are represented using graph charts.From these charts we derive the input assumptions to our Genetic Algorithm.

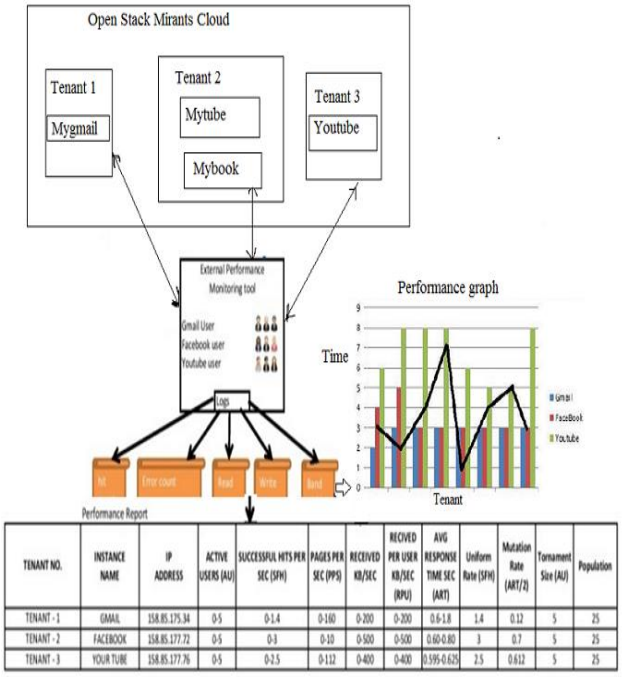

Fig.1

B. Calculating the performance of each Virtual Machine

Here we make use of a tool named "WapT pro" for Load testing on each of the virtual machines, where Load testing with "WAPT Pro" is simple and straightforward. It can emulate the activity of hundreds or even thousands of real users from a single testing machine. The program replaces human users with virtual users.
Here each tenant, initially starts with a minimum user load of 5 users/second that eventually increases to 25 users/second (i.e. 5x5 users) and each user to keep accessing the instance critical memory, while we count the different parameters like applications such as average response time, overall band width, successful hits, errors and etc. and these values are used as input to our genetic algorithm.

IV. IMPLIMENTATION OF THE SYSTEM

A. Introduction

The implementation is done in the Mirantis Cloud platform by creating tenants with Virtual Instance and the required load testing graphs are obtained using "WapT Pro" tool, and JAVA IDE is used to implement Genetic Algorithm.

B. Tools Used

Mirantis Cloud Openstack Platform: The below Fig.2 shows Mirantis free Openstack Cloud service provider where they provide us with enterprise-grade OpenStack environment for our development and building and assigns a dedicated technical account manager to facilitate us. Along with real-time monitoring and support includes SLAs that ensure our cloud SaaS is available for users.

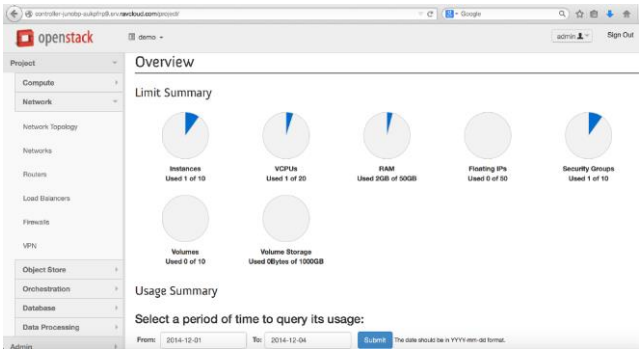

Fig.2. Mirantis Cloud Openstack Platforms

Creating a test scenario:We use "WapT pro" tool and floating IP address is given as an input to the "WapT pro" tool with prefixed virtual load on each cloud instances.When we start executing the test, on instance, simultaneously graphs are generated which states the different parameters like applications average response time, overall bandwidth, successful hits, errors and etc. and these values are used as input to our genetic algorithm.

C. Implementation of Genetic Algorithm

Genetic Algorithm is proposed to better support the software instance testing in the cloud. The proposed architecture, mainly contains application running on virtual instances and their response to the dynamically varying load with respect to time and optimization its resource utilization in the cloud.

i) Procedure for Implementing Genetic Algorithm: In order to improve the overall performance of the cloud application, without need of additional resources and providing best end user experience is the most challenging aspect. Even though many methods and techniques were proposed to compensate the demand, none could give a lifetime solution to the challenge; here we make an approach by implementing genetic algorithms to cloud applications that produce an optimal solution.

a) Initial Population: The keyfactors that are consideredarethe population sizeand howto choose the individuals. A meaningful search is possible only if the size of the population isminimum.

For finding the right population size-

(a) Create an initial population, which can be generated randomlywith a desired size.

(b) Selecting initialization is random from only a few individuals to thousands.

(c) Consider a population of size in multiples of 5,and that each gene has 5 alleles say *'m'*, labeled 0…4, and the values of alleles in each 'column' are generated using randompermutation
after whichmodulo 5 is computed on each value to get values in the range 0 to $m-1$.

b) Termination: GA isa stochastic search technique that would run for ever compared to simple neighborhood search methods that terminate when a local optimum is reached. In real time cases,a termination criterion is needed; In order to select the termination criterion, the usual practices are based on clock time of the computer or setting up of a limit based on the number of fitness evaluations or based on population's diversity track. The search is stopped when this value falls below a fixed threshold. Here the diversity could be relatedto either the genotype or the phenotype or to the fitness's.This is mostly measured by the genotype statistics. For example, termination of a run can be done when the fraction of one particular allel at every locum exceeded above 90%.

c)Ranking: Ordering or ranking the chromosomes based on the fitness may cause some information to be lost. Hence, we use linear ranking to select the k^{th} rank string from the population which is represented as$P[k]$ given by the equation (for linear ranking), where α and β are constants.

$$P[k] = (\alpha + k\beta) \quad \text{Eq. 1.}$$

Let the probability distribution be denoted by $P[k]$, the other parametercan be chosen in a way we tune the *selection pressure.*

Selection pressure,

$$\emptyset = \frac{Prob[selecting\ Fittest\ String]}{Prob[Selecting\ Average\ String]} \quad \text{Eq.2.}$$

In the case of linear ranking the average is interpreted as meaning the median string, so tha

$\Phi = \frac{\alpha+\beta M}{\alpha+\beta(M+1)/2}$Eq.3

(So the population size is assumed as odd, but for the even number, the analysis holds mutatis mutandis) that can be defined with simple algebra

$$\beta = \frac{2(\phi+1)}{M(M-1)} \text{ and } \alpha = \frac{2M-\phi(M+1)}{M(M-1)} \quad \text{Eq. 4.}$$

that implies that $1 \leq \Phi \leq 2$ With this framework, it is observed that, in terms of the sum of an arithmetic progression, the cumulative probability distribution can be stated, so that identifying appropriate value for$k$ for a given pseudo-random number $r$will become simple, i.e. Eq.5 gives the quadratic equation to solve k, which can be computed in in O(1) time.

$$k = \frac{-(2\alpha+\beta)\pm\sqrt{(2\alpha+\beta)^2-4\beta r}}{2\beta} \text{ Eq.5}$$

Other than linear ranking, many other functions can be used. But most of the applications usethe above mentionedtechniqueas it is more flexible.

d) Tournament Selection:One majorbenefit of tournament selection above all the remaining techniques is that it only needs the pairs or groups of strings be orderd according to a specific preference, and hence, it can cope up with resource optimization situations.

However, it is noted thatarbitrary stochastic effects have an impact onthe tournament selection in the same way as roulette-wheel selection, there is no guarantee that every string will appear in a given cycle. In fact, using sa mpling with replacement there is a probability of approximately0.368($\sim e^{-1}$) at a given string will not appear at all.

To string missing we need τ items to be drawn Mtimes, so we simply construct τ random permutationsof the numbers $1, \ldots . M$ i.e. the indices of the individuals in the population. These are concatenated into one long sequence which is then chopped up into M pieces, each one containing the τ indices of the individuals to be used in the consecutive tournaments.

$$P[k] = \alpha + \beta\, k$$

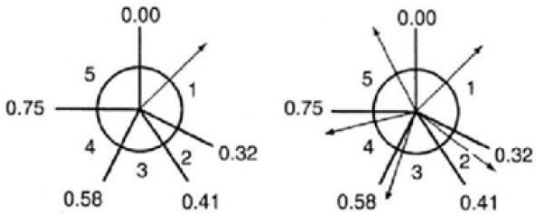

Fig.3 Roulette Wheel

The M needs to be an exact multiple of τ otherwise, there is the small possibility for some distortion in the permutations join, but this is a relatively minor problem.

e) Crossover: The process of replacing some of the genes in one parent by the corresponding genes of the other is termed as crossover. One-point crossover (1X) or two point crossover (2X) or uniform crossover(X).

In our approach we make use of uniform crossover, where the obtained new off springs will contain 50% characters if 1st parent string and 50% characters of the 2nd parent string. An indistinguishable prescription can be given for*m*-point crossover where $m > 1$.

f) Mutation: It is used for the genetic diversity from one generation to the successor generation. Assuming that the concept of mutation is much simpler compared to crossover, and that the chromosomescan be represented as a bit-string,usingBernoulli distribution, a mask such as

0 1 0 0 0 1

can be generated at each locus. In the above example new allele values are assigned for the 2nd and 6th gene.Apart from this technique that are other ways that can be used to implement this idea which can significantly improve the performance of a GA. A simple idea is to select a random number for *every* gene in the string which is compared with µ. The computational complexity of this method increases when the strings become longer and when there is huge population.

For m mutations, we draw m random numbers (without replacement) uniformly distributed between 1 and l in order to specify the loci where mutation is to producing new allele for the farther crossover to produce the fittest.

g) New Population: A generative approach is used here: Until a new set of M individuals were generated, the process of selection, recombination and mutation were executed on a population of M chromosomes,.This set is considered as the new population. To acheive optimization, a considerable amount of effort has been spent to achieve a good solution, to avoid the risk of it from participating in further reproduction.Hence, concepts of *elitism* and *population overlaps*were introduced.This elitist strategy is simple andguarantees the best individual got so far to survive by preserving it. Only the remaining $(M - 1)$ members of the population are replaced with new strings. Populationsoverlaps replaces only a fraction$G$ (the generation gap) of the population at each generation which is much more simplified. By applying these steady-state or incremental strategies,at each stage one new chromosome (or a pair) is generated.

There are different strategies, commonly used in the GA community, which traditionally designates them either $\lambda.\mu + (\lambda + \mu)$.In the first case,µ($> \lambda$)offspring is generated from λparents, and the best $\lambda$of these offspring are chosen to start the next generation. For the + strategy,µoffspring is generated and the best λindividualsare chosen from the combined set of parents and offspring.In the case of incremental reproduction it is also necessary to select members of the population for deletion. Some GA's assumed that parents are replaced by their children and delete the worst member of the population, this exerts a very strong selective pressure on the search, which may need fairly large populations and high mutation rates to prevent a rapid loss of diversity. A milder prescription is to delete one of the worst P% of the population (for example, Reeves used P = 50i.e., selection from those worse than the median). This is used when rank-based selection is used.

V. PERFORMANCE EVALUATION OF GENETIC ALGORITHM

The implementation of the GENETIC ALGORITHM is done using JAVA IDE. We analyze the performance of the proposed Genetic Algorithm; by using tools like "WapTpro" to test application response of the dynamically varying load and the results are recorded, which are then, used as input in our GENETIC ALGORITHM and JFreeChart graph is used to display the output in graph between Generation and Fittest.

Environment Parameters

Various parameters are considered in the design of the GENETIC ALGORITHM. All those parameters are tabulated in table 2.

Table 2 :Parameters and Values

Parameters	Values
Operating System	CentOS 6.5 x64
RAM	2GB and 512MB
Virtual Central Processing Units(VCPU)	1
Virtual Disk Drive	20GB and 10GB

VI. RESULTS AND DISCUSSION

As we observed in the previous sections, the implementation of proposed approach is done using NET BEANS-IDE. In this project, we implemented Genetic Algorithm on the obtained load test values The test is performed on 3 Tenants, where each tenant consist of minimum one Virtual Machine with above specified configurations.

i) TENANT 1 - Gmail:

Table 3: Tenant 1 Gmail VM Test results

INSTANCE NAME	IP ADDRESS	ACTIVE USERS (AU)	SUCCESSFUL HITS PER SEC (SFH)	PAGES PER SEC (PPS)	RECEIVED KB/SEC	RECIVED PER USER KB/SEC (RPU)	AVG RESPONSE TIME SEC (ART)
GMAIL	158.85.175.34	0-5	0-1.4	0-160	0-200	0-200	0.6-1.8

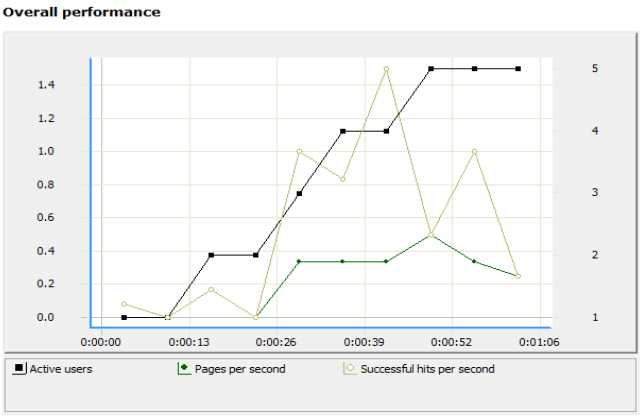

Fig.4. Overall Performance of Gmail VM

The test results are obtained from "WapT Pro" tool are plotted in Table 3 are used as input to the Genetic Algorithm, where the results shows the Overall Performance like active users, pages per second and successful hits per second and etc. it is shown in Figure.4.

Genetic Algorithm on TENANT 1 Gmail

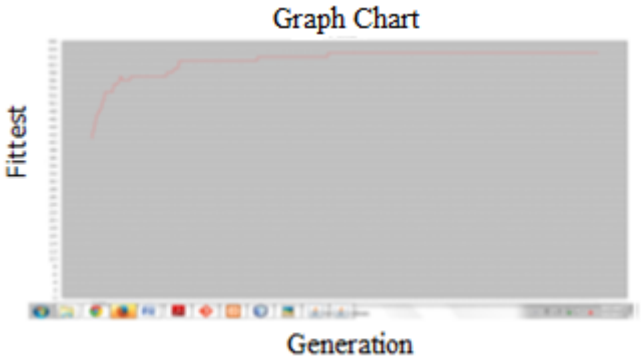

Fig.5. Graph is plotted between Generation and Fittest

Figure 5 represents the Fittest and number of generations obtained.Generations -141 and the Fittest - 63.

ii) TENANT 2 - Facebook:

Table 4: Tenant 2 Facebook VM Test results

INSTANCE NAME	IP ADDRESS	ACTIVE USERS (AU)	SUCCESSFUL HITS PER SEC (SFH)	PAGES PER SEC (PPS)	RECEIVED KB/SEC	RECIVED PER USER KB/SEC (RPU)	AVG RESPONSE TIME SEC (ART)
FACEBOOK	158.85.177.72	0-5	0-3	0-10	0-500	0-500	0.60-0.80

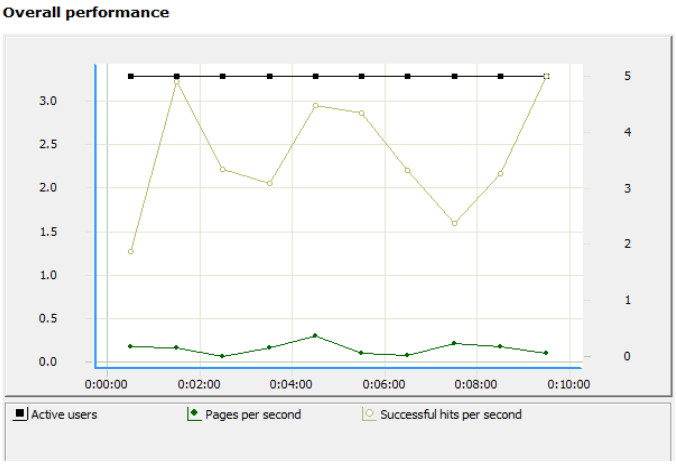

Fig.5. Overall Performance of Facebook VM

The test results are obtained from "WapT Pro" tool are plotted in Table 4 and Fig.5 are used as input to the Genetic Algorithm, where the results shows the Overall Performance like active users, pages per second and successful hits per seccond and etc.

Genetic Algorithm on TENANT 2 Facebook

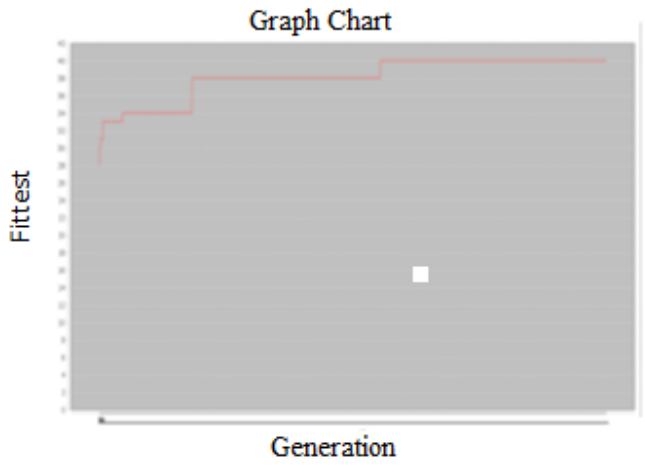

Fig.6. Graph is plotted between Generation and Fittest

Figure-6 represents the Fittest and number of generations obtained.Generations -587 and the Fittest - 40.

iii) TENANT 3 YouTube:

Table 5: Tenant 3 YouTube VM Test results

INSTANCE NAME	IP ADDRESS	ACTIVE USERS (AU)	SUCCESSFUL HITS PER SEC (SFH)	PAGES PER SEC (PPS)	RECEIVED KB/SEC	RECIVED PER USER KB/SEC (RPU)	AVG RESPONSE TIME SEC (ART)
YOUR TUBE	158.85.177.76	0-5	0-2.5	0-112	0-400	0-400	0.595-0.625

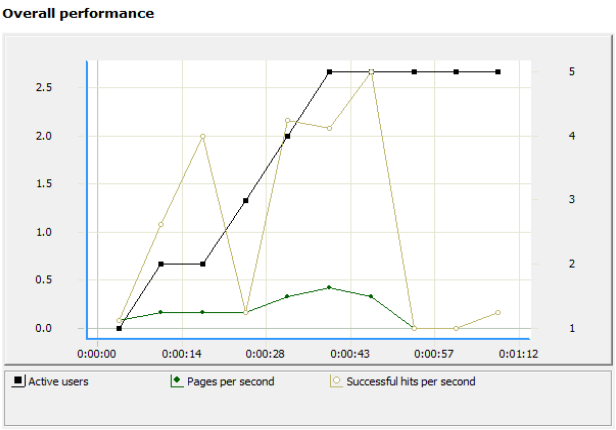

Fig.7. Overall Performance of YouTube VM

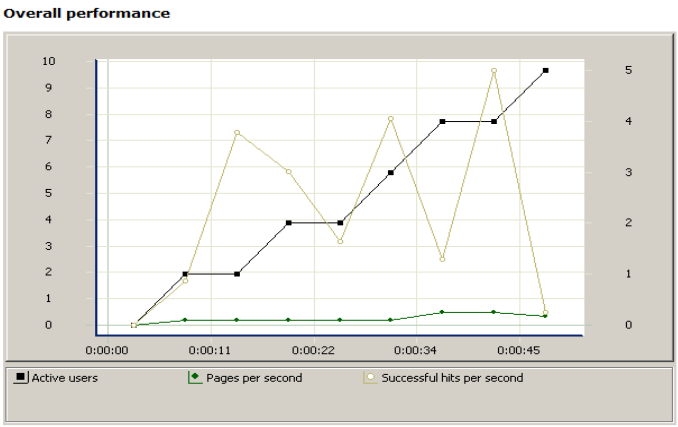

Fig.9. Overall Performance of Genetic YouTube VM

The test results are obtained from "WapT Pro" tool are plotted in Table 5 and Fig.7 are used as input to the Genetic Algorithm, where the results show the Overall Performance like active users, pages per second and successful hits per second and etc.

Genetic Algorithm on TENANT 3 YouTube

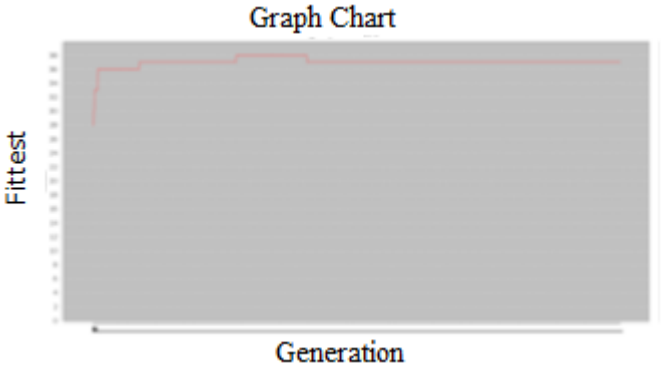

Fig.8. Graph is plotted between Generation and Fittest

Figure- 8 represents the Fittest and number of generations obtained.Generations -809 and the Fittest - 39.

iv) Validation of Proposed System: From the above test we have determined that Tenant 3 is working with minimal resource utilization under dynamic load. As it could achieve fittest generation at 39th generation which has less generations compared to others.

Now tenant 3 characteristics were applied for the new instance named "*Genetic YouTube*" in the cloud, which is provided with only one fourth resource of our previous "*YouTube*" instance i.e. 2GB RAM is reduced to 512MB and 20GB Hard Disk space is reduced to 10GB leaving the Operating System as CentOS 6.5 x64 and number of Virtual CPU as 1. Now we validate our new instance by executing all the above test cases on it, the output gives the desired application performance at less resource utilization as shown below 5.3.5.

Table 5 Tenant 1 Genetic YouTube VM Test results

INSTANCE NAME	IP ADDRESS	ACTIVE USERS (AU)	SUCCESSFUL HITS PER SEC (SFH)	PAGES PER SEC (PPS)	RECEIVED KB/SEC	RECIVED PER USER KB/SEC (RPU)	AVG RESPONSE TIME SEC (ART)
GENETIC YOUTUBE	169.53.135.55	0-5	0-2.1	0-10	0-650	0-600	0.6-1.6

The test results are obtained from "WapT Pro" tool are plotted in Table 5 and Figure 9 are used as input to the Genetic Algorithm, where the results show the Overall Performance like active users, pages per sec and successful hits per sec and etc.

v)Genetic YouTube tenant response: And tenant 3 characteristics were applied for the new tenant named "*Genetic YouTube*" in cloud and all the above test cases are executed on it as shown below,

Genetic Algorithm on Genetic YouTube tenant

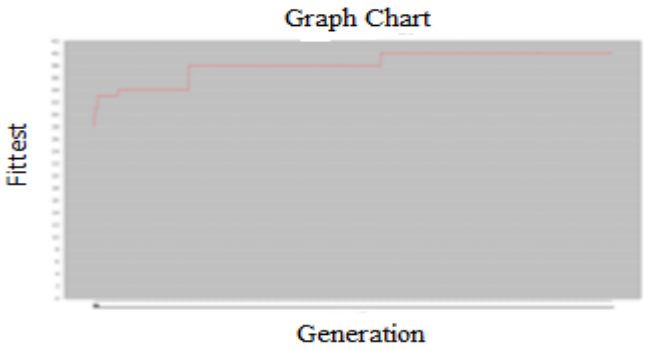

Fig.10. Graph is plotted between Generation and Fittest

Figure-10 represents the Best Fit and number of generations obtained-overall Generations-178 and the Fittest 46. From the obtained graphs and results we can say that our tenant delivers desired performance even when the providedresources for this tenant are less than the actual provisioning.

VII. CONCLUSION AND FURURE WORK

This paper explain about various perspectives of SaaS testing at the practical level. By highlighting the special testing features in genetic algorithm. we believe there is an urgent need to develop new techniques, standards, platforms, solutions for cloud SaaS instance, where it is difficult to find the exact load at a particular instance at a given time Probably GA is unique for the above need, one does not know where to look for the solution and where to start, as SaaS instances is non-deterministic in nature, to find solution for such problem we make use of Genetic Algorithm which is often considered as a good solution, to determine the optimized resources in cloud application,

which would need the implementation of variations in population size, in initialization methods, in fitness definition, in selection and replacement strategies, in crossover and mutation are obviously possible. Parallel processing is also possible with genetic algorithm. Parallelization can be extended to very sophisticated implementations that add spatial aspect to the algorithm.

REFERENCES

1. Chana, Inderveer, and Ajay Rana. "Empirical evaluation of cloud-based testing techniques: a systematic review." ACM SIGSOFT Software Engineering Notes 37.3 (2012): 1-9.
2. Gao, Jerry, Xiaoying Bai, and Wei-Tek Tsai. "Cloud testing-issues, challenges, needs and practice." Software Engineering: An International Journal 1.1 (2011): 9-23.
3. Booker, Lashon B., David E. Goldberg, and John H. Holland. "Classifier systems and genetic algorithms." Artificial intelligence 40.1-3 (1989): 235-282.
4. Rechenberg .Evolutions strategic: Optimierung technischer Systeme nachPrinzipen der biologischen Evolution, Frommmann-Holzboog Verlag, Stuttgart(2nd edition 1993).
5. Mitchell, Melanie, John H. Holland, and Stephanie Forrest. "When will a genetic algorithm outperform hill climbing." Advances in neural information processing systems. 1994.
6. H.-P.Schwefel (1999).Numerische Optimierung von Computer-modellen mittelsder Evolutionsstrategie, Birkhäuser Verlag, Basel. (English edition: Numerical Optimization of Computer Models, John Wiley & Sons, Chichester, 1990).
7. Agrawal, Dipanshu, et al. "An Evolutionary Approach to Optimizing Cloud Services." (2012).
8. Portaluri, Giuseppe "A power efficient genetic algorithm for resource allocation in cloud computing data centers." 2014 IEEE 3rd International Conference on Cloud Networking (CloudNet). IEEE, 2014
9. Maqableh, Mahmoud, and Huda Karajeh. "Job scheduling for cloud computing using neural networks." Communications and Network 6.03 (2014): 191
10. Patil, S. D. and S. C. Mehrotra. "Resource allocation and scheduling in the cloud." Int J Emerg Trends Technol Comput Sci (IJETTCS) 1.1 (2012): 47-52.
11. Li, Li Erran, and Thomas Woo. "Dynamic load balancing and scaling of allocated cloud resources in an enterprise network." U.S. Patent Application No. 12/571,271.
12. Chen, Shang-Liang, Yun-Yao Chen, and Suang-Hong Kuo. "CLB: A novel load balancing architecture and algorithm for cloud services." Computers & Electrical Engineering 58 (2017): 154-160.
13. Ningning, Song, et al. "Fog computing dynamic load balancing mechanism based on graph repartitioning." China Communications 13.3 (2016): 156-164.
14. Mesbahi, Mohammadreza, and Amir Masoud Rahmani. "Load balancing in cloud computing: a state of the art survey." Int. J. Mod. Educ. Comput. Sci 8.3 (2016): 64

AUTHORS PROFILE

Mr. Vishnu Shankar, has 4+ years industry experience as enterprise application engineer, currently working for GE Power, India. Implemented Business solutions for clients across Asia Pacific, US and Nordics regions on Salesforce.com platform. Completed master's in computer science with specialization in cloud Computing from VIT University, India.

Prof. Sajidha S.A, has completed her M.E (CSE) from Anna University andsubmitted a research thesis fromfromVIT University. She has about 20 years of teaching experience at various levels and presently working as an Assistant Professor(selection grade) in the School of Computing Science, VIT University, Chennai. She has published several papers inreputed International Journals and her area of interest includesdata analytics, cloud Computing, machine Learning and data mining.

Prof. Nisha V.M, has completed her M.E (CSE) from Anna University and submitted a research thesisfrom VITUniversity . She has about 14 years of teaching experience at various levels and presently working as an Assistant Professor(senior grade) in the School of Computing Science, VIT University, Chennai. She has published several papers in International Journals and her research interest includes cellular automata, cloud Computing anddata analytics.

Dr. B. Sathis Kumar, has completed his M.E (CSE) from Anna University and Ph.D from Anna University . He has about 20 years of teaching experience at various levels and presently working as an Associate Professor in the School of Computing Science, VIT University, Chennai. He has published 15 papers at International Journals and International Conferences. His research interest includes Software Engineering, Big Data analytics, Cloud Computing, Machine Learning and IoT.