Genetic Algorithm and Gravitational Emulation Based Hybrid Load Balancing Strategy In Cloud Computing

Santanu Dam[*], *Gopa Mandal*[†], *Kousik Dasgupta*[†] *and Paramartha Dutta*[‡]
[*]*Department of CSE,Future Institute of Engineering & Management,Kolkata-700 150, India*
Email:sntndm@gmail.com
[†]*Department of CSE,Kalyani Govt. Engineering College,Kalyani-741 235, India*
Email:gopa.mandal@gmail.com
Email:kousik.dasgupta@gmail.com
[‡]*Department of CSS,Visva- Bharati University,Shantiniketan-731 235, India*
Email:paramartha.dutta@gmail.com

***Abstract*— Cloud computing enables a new supplement of consumption and delivery model for internet based services and protocol. It helps to provide software, hardware and data in form of collaborative services on the demand of the end user. To meet the QoS and ensure high interoperability and scalability is one of the most challenging tasks for cloud service provider. However, there are also several technical challenges that need to be tackled before the benefits can be fully realized. Among them reliability, resource provisioning, and efficient resources consuming etc are major concern. Load balancing also one of them. It includes selecting a proper node that must be full filled end user demand and also distribution of dynamic workload evenly into the multiple nodes. So load balancing can be described as an optimization problem and should be adapting nature due to the changing needs. In this paper we suggest a novel load balancing strategy to search under loaded node to balance load from overwhelmed node. CloudAnalyst used as a simulation tool for the proposed load balancing strategy. Experimental results of the sample application are really very encouraging. Significantly the results of the proposed algorithm are compared and outperformed the traditional strategy like First Come First Serve(FCFS), local search algorithm like Stochastic Hill Climbing(SHC) and soft computing approaches like Genetic Algorithm (GA) and Ant Colony Optimization(ACO).**

Keywords— Cloud Computing; CloudAnalyst; Load balancing; Genetic Algorithm.

I. INTRODUCTION

Internet technology is growing very firstly, due to advent of internet, cloud computing it has become one of the exponentially growing technologies in this decade. Today cloud computing becomes very hot topic by providing computing as utility to meet the everyday needs of the users. Different companies are trying to use cloud based system very extensively, due to its simple and flexible architecture and service that is provided on demand anywhere from the world. This has results exponentially increase in number of cloud users. It represents as a new paradigm for the dynamic provisioning of computing services, typically supported by state-of-the-art data centers containing ensembles of networked Virtual Machines[1].Cloud computing is based on distributed computing mechanism that use high speed internet to disperse the job from local PC to remote computer clusters (big data centers owned by cloud service providers) for data processing. Cloud computing has been adopted widely by organization which includes, social networking websites like Facebook, online application design by Google app managers and by Google doc etc. Some cloud based application is specially designed for online software testing. This all suggests that cloud computing not only change the way we interact with the resources through Internet but also beginning of the glorious future of cloud computing. It uses virtualization as technology that helps in slicing a single data centre or high power server to act as multiple machines. Despite glorious future of the cloud computing, many crucial problem needs to be addressed for the actual realization of the new technology. Load balancing is one of in them which play a very important role. The main objective behind load balancing is to distribute the local workload evenly to the entire cloud to ensure at any instant moment of time no overloaded processor or resources is present in entire network. It is used by Cloud service provider (CSP) in its own cloud computing platform to provide a high efficient solution for the user. Also, a inter CSP load balancing mechanism is needed to construct a low cost and infinite resource pool for the consumer [2]. Thus load balancing helps to distribute any number application requests, into any number of resources as service located in data centers via through cloud service providers. For this reason a good load balancing algorithm must be dynamic and adapt to the environment. Load balancing strategy can be broadly categorized as centralize or decentralized, static or dynamic and periodic or non periodic. Several approaches of Load balancing are found in literature Armstrong et. al. in [3] uses Minimum Execution Time (MET) assign each job in arbitrary manner to a particular node on which it is expected to be executed fastest, regardless of the current load on that node. Some existing scheduling techniques

978-1-4799-4445-3/15/$31.00 ©2015 IEEE

like Min-Min, Round Robin and FCFS are also used for load balancing also exist in literature. An intelligent method for load balancing has been proposed by Yang Xu et. al. [4]. A few soft computing techniques like Stochastic hill Climbing, Genetic Algorithm, Ant Colony [6] were also reported. In this paper a hybrid approach base on Genetic Algorithm (GA) and the gravitational emulation local search (GELS) has been proposed for load balancing of VMs in Cloud. GEL has been used as local search algorithm which imitates gravitational pool therefore it's strong for local searches and weak for global searches. GA is totally based on biological rule and strong for global searches. Combination of these two algorithms can solve the load balancing problem by the help of dual optimization strategy. CloudAnalyst a CloudSim based visual modeler has been used for simulation and analysis of the proposed technique. The experimental result remarkably optimizes the entire system load. The rest of paper is organized as follows. Section II Introduces the CloudAnalyst toolkit. Section III Load Balancing of VM's using Gravitational emulation and Genetic Algorithm in Cloud Computing. Section IV details the proposed GA-GEL algorithm. Section V presents the simulation results. Finally, Section VI concludes this paper

II. CLOUD ANALYST

We find most of time it's become so difficult and time consuming to measure the performance of the application or proposed policies, whenever we are going to test in real world environment. Simulation in this consequence becomes very much helpful for users or researchers because its enables with practical feedback without having real environment. This section briefly portray the simulation in cloud to support application level infrastructure, services arises from cloud computing paradigm. It includes modeling of on demand virtualized resources, which supports cloud infrastructure. Practically in real world scenario to get number of user's application and their impact on performance is very hard to acheive. To test these new issues, testbed are required which supports Cloud infrastructures. Different simulators right now are available to adapt the real world situation like CloudSim [5] and CloudAnalyst [6]. In this paper Cloud Analyst has been used as a simulation tool, developed at the University of Melbourne. Main objective of the CloudAnalyst to evaluate social networks tools according to geographic distribution of users and data centers. Communities of users and data centers supporting the social networks are characterized based on their location. Cloud Analyst developed on CloudSim, it's a GUI based simulation tool. CloudSim facilitates modelling, simulation and other experimentation on cloud programmatically. CloudAnalyst uses the functionalities of CloudSim and does a GUI based simulation. It allows setting of parameters for setting a simulation. CloudAnalyst enables developers to evaluate the large scale application in terms of both computing servers and user's workload. A snapshot of the GUI of CloudAnalyst simulation toolkit is shown in figure 1(a) and its architecture in depicted figure 1(b). CloudAnalyst[6] developed as a visual modeler tool on CloudSim[5]. Maintaining the Integrity of the Specifications

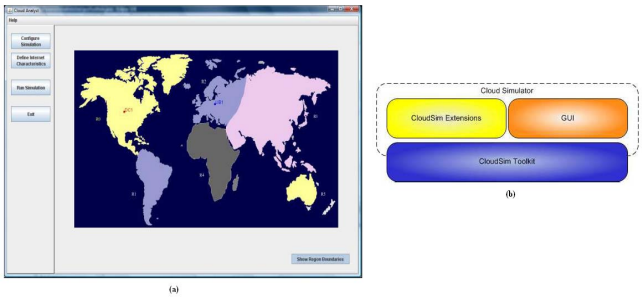

Fig.1. Snapshot of CloudAnalyst (a) GUI of CloudAnalyst (b) Architecture of CloudAnalyst builds on CloudSim

III. LOAD BALANCING OF VM'S USING GRAVITATIONAL EMULATION ALGORITHM AND GENETIC ALGORITHM IN CLOUD COMPUTING

GA was first proposed by John Holland et al.[7] in 1975 at Michigan University.GA is basically inspired by the nature and considered as most natural evaluation method for existence.GA selects the most suitable strings from organized stochastic information that is searched and gathered by any living being. In each generation a new set of string is produced to survive in changing world. The new set is tested for their strength or fitness level is evaluated. The basic form of GA is as follows.

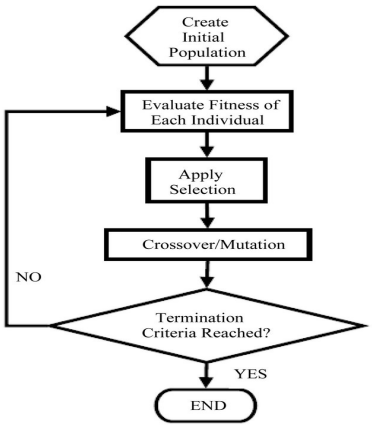

Fig.2. Basic Genetic Algorithm

3.2 Gravitational Emulation Local Search

In 1995, Voudouris and his colleagues [7] proposed the Guided Local Search (GLS) algorithm with an NP-hard solution for searching in a search space. In 2004, Vebster[8] presented GLS as a strong and effective algorithm, the GELS algorithm. GEL mimics gravitational attraction for searching within a defined search space. Each response has different neighbours and they may be grouped based on problem-dependent criteria. The neighbours obtained in each neighbour group are called a dimension. A primary velocity is defined for every dimension. For every individual dimension, if we found with greater velocity for a particular dimension it is considered more responsive for the problem [10]. The GEL algorithm accounts for gravitational force in the responses in a search space by the help of two methods. In case of first

method a response is selected from local neighbour space of the current response rather than being limited to one response. GELS allow movement into the search space by the help of two methods. The first method allows movement from the current response towards the response of the current response in local neighbour spaces. In second method movements are allowed from the responses outside of the current response local neighbour spaces, in addition to the neighbour responses of the current response. Each of these movement methods has been used in combination with each gravitation force calculation method .GELS algorithm used by Blachandar [10], to solve the Travelling Salesman Problem and compared it with other algorithms such as Hill Climbing and Simulated Annealing. It is observed that initially the algorithm works same as like the other algorithm whenever the problem size is small. But it remarkably outperforms the other existing algorithm when problem size is large. A primary response and a primary velocity vector are used at the beginning of the algorithm which consist a primary velocity that may be specified by the user or randomly generated. After the primary velocity vector has been examined, the responsive dimension with the greatest primary velocity among the neighbour dimensions is selected for movement. A pointer object is used to move within the search space which basically points response with the most weight. In the first iteration of the algorithm using the first method, a dimension is selected for obtaining a neighbour response from the current response. Candidate response is selected from the local neighbour space of the current response in terms of this dimension. Then the gravitational force between the current and candidate responses is calculated and added to the primary velocity of the dimension of the candidate response, thus updating primary velocity. In the next iteration, a new movement direction is selected for continuing the response search for this reason primary velocity vector is examined. Each iteration of the algorithm using the second method is generally similar to the first method. Only difference is that instead of calculating gravitational force and updating the primary velocity vector for just one candidate response in the current dimension, gravitational force is calculated and the primary velocity updated for each candidate response in the current dimension. In this algorithm, the gravitational force between two entities is calculated using Equation (1):

$$f = \frac{G(CU-CA)}{R^2} \qquad (1)$$

Where CA is the candidate response, CU is the current response and R is the constant or may change on each iteration. The terminating condition set here primary velocity for all dimension equal to zero or maximum number of iteration is reached.

3.3.1 First Fitness Function

Cloud computing is dynamic but at any particular instance the said problem of load balancing can be formulated as allocating N number of jobs submitted by cloud users to M number of processing units in the Cloud. Each processing unit vector (PUV) that is indicating current status of processing unit utilization. This vector consists of MIPS, indicating maximum number of instructions can be executed by that machine per second. The delay cost L is an estimate of penalty, which Cloud service provider needs to pay to customer in the event of job finishing actual time being more than the deadline advertised by the provider. NIC represents the number of instructions present in the job, this is count of instruction in the job determined by the processor. Here $w_1 = 0.8$ and $w_2 = 0.2$ are considered such that we get their summation as one.

$$fit1 = w_1 * \propto (NIC \div MIPS) + w_2 * L \qquad (2)$$

3.3.2 Second Fitness function

With respect to the basic purpose of load balancing, minimizing makespan, several chromosomes may be found that have similar makespans but don't all balance the load among the VMs. Hence, our proposed algorithm is static, we assume expected completion time of user task are determined and set in the Expected Time of Completion table (ETC) where ETC [i,j] represents each task i on each vm j. Also, the ready time(Ready[j]) means when a VM is ready to take the next task. Makespan indicates maximum completion time, Comp_Time[i,j].

$$\text{makespan}=\text{Max}(\text{Comp\_Time}[i,j]) \quad \{1\le i \le N,\ 1 \le j \le M\} \qquad (3)$$

Comp_Time [i,j] is the time at which task i ends on vm j and is calculated according to Equation(4)

$$\text{Comp\_Time}[i,j]=\text{Ready}[j]+ \text{ETC}[i,j] \qquad (4)$$

The second fitness function considers this factor after obtaining solutions with similar makespans, to find the most appropriate solution with respect to load balance. If the execution time for any user job is E_time[Ji], the average execution time (avg) for all users jobs is as shown in Equation (3)

$$avg = \sum_{i=1}^{num\ of\ jobs} (E_{time}[J_i] \div \text{num of VMs}) \qquad (5)$$

Load balance for vm_i , CPU_LB_i is obtained from Equation 6 given below

$$CPU\_LB_i = \text{makespan} \div \text{avg} \qquad (6)$$

Hence second fitness function taken as

$$\text{fit2}= \frac{1}{CPU\_LBi} \qquad (7)$$

Initial population generation: We use GEL algorithm to select the total population here high primary velocity taken into concern. We select the population with the possibility that

chromosome may not initially have a good fitness value but turn out to be better after the mutation and crossover operations and converted into binary string pair.

Crossover : Our proposed algorithm uses a two- point crossover operator. Which are selected randomly from the previous phase. Within these two points of the first and second chromosomes are removed

Mutation: Now a point on each chromosome from previous step is selected randomly the bits of the chromosomes, are toggled from 1 to 0 or 0 to 1.

Force Calculation: After crossover and mutation the gravitational force of the current chromosome and candidate chromosome are calculated using Equaion 1, then the gravitational force is added to the velocity of that dimension. This leads to no copying in the candidate population.

Terminating Conditions: The algorithm terminates when the primary velocity is equal to zero for all dimensions or the maximum number of iterations has been reached.

IV. PROPOSED ALGORITHM

Input: Population of VMs

Output: Scheduled VMs based on fit1 and fit2.

STEP 1: Randomly initialize a population of processing unit after encoding them into binary strings [Start].

STEP 2: Set the Velocity value for each dimension.(1,2,….upto k)

STEP 3: While (i<=max_iteration or Velocity i ≠0) then /* select current_chromosome1 and current_chromosome2 such that the velocity is larger and it will generate two offspring, candidate_ch1 and candidate_ch2, by crossover and mutation*/

STEP 4: If (Fit1(candidate_ chromosome1) >Fit1(current_ chromosome 1)) then

current_chromosome1=candidate_cchromosome;

STEP 6: If (Fit1(candidate_ chromosome2)> Fit1(current_ chromosome2)) then

current_ chromosome2= candidate_chromosome2;

STEP 7: Calculate force from Equation1

STEP 8: UPDATE the velocity for each dimension by gravitational force.

STEP 9: END WHILE

STEP10: If many chromosomes with same fit1 exist then SELECT best chromosome using fit2 for crossover

V. SIMULATION RESULTS AND ANALYSIS

Our proposed algorithm is simulated in ClouAnalyst[] where we consider the real world scenario of "social networking site like Facebook ". Suppositional configuration partitioned the world into six continents given in table.

TABLE I: Configuration of simulation environment

S.No	User Base	Region	Simultaneous Online Users During Peak Hrs	Simultaneous Online Users During Off-peak Hrs
1	UB1	0–N. America	4,70,000	80,000
2	UB2	1–S. America	6,00,000	1,10,000
3	UB3	2 – Europe	3,50,000	65,000
4	UB4	3 – Asia	8,00,000	1,25,000
5	UB5	4 – Africa	1,15,000	12,000
6	UB6	5 – Oceania	1,50,000	30,500

All user bases (UB) are set to a single time zone and a sample online and offline user during peak and off peak hour has been considered. We considered only one tenth of the online user available during off peak hours. Each data center host has a particular amount of virtual machine (VMs). For simulation each of the Machines has been considered of 4GB of RAM, 100 GB storage and 1000MB of available bandwidth. Each Datacenter (DC) is assumed to be having 4 CPUs with a capacity of 10000 MIPS. Simulated hosts have x86 architecture, virtual machine monitor Xen and Linux operating system. Each user request (jobs) has been considered to be requiring 100 instructions to be executed. The proposed algorithm is executed in several setup that's are tabulated in Table 2, here we considered one DC with 25, 50, 75 VMs, and Table 4, 5, 6 and 7 considers three, four, five and six DCs respectively with a mixture of 25, 50 and 75 VMs for all DCs. Average response time of the jobs are calculated for the proposed algorithm and tabulated. The performance of te propose algorithm is really encouraging after comparing the result with some existing techniques described earlier. Figures 2, 3,4,5,6 and 7 make a comparative analysis of the proposed technique for the different scenarios and techniques. The novelty of the work confirmed by comparative analysis

VI. CONCLUSION

This paper presents an algorithm for solving load balancing problem among VMs through a combination of a GA and GEL. GA has global nature towards the problem space where GELS searches locally. Basically the proposed algorithm tries to minimizing the make span as well as reduce the number of VMs who are going to miss their deadlines. Though GEL algorithm use some random elements that does not move always among them in the same way for this reason it does not stop always with the best possible solution. However combination of both techniques improved the response time of VMs that is compared and verified with other existing technique like ACO, GA, SHC and FCFS which is

outperformed by the algorithm and guarantees the QoS requirement of the customer job. Though fault tolerance issues and priority of the job has not been consider here, which may not be the actual scenario. Researchers can proceed to include the fault tolerance and priority of the job can be used for further research work.

REFERENCES

[1] R. R.Buyya, R.Ranjan, Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services,in: ICA3PP 2010, Part I, LNCS 6081., 2010, pp. 13–31.

[2] "Load Balancing in Cloud Computing using Stochastic Hill Climbing-A Soft Computing Approach", Brototi Mondal,Kousik Dasgupta and Paramartha Dutta, in Proc. of (C3IT-2012),Elsevier, Procedia Technology 4(2012), pp.783-789, 2012.

[3]. T.R. Armstrong, D. Hensgen, "The relative performance of various mapping algorithms in independent runtime predictions", in proc. Of 7th IEEE Heterogeneous computing workshop (HCW 98). Pp. 79-87,1998

[4] Yang Xu, Lei Wu, liying Guo, Zheng Chen, Lai Yang, Zhongzhi Shi, "An Inteleegent Load Balancing Algorithm Towards Efficient Cloud Computing", in proc. Of AI for Data Center Management and cloud Computing: Papers from 2011 AAAI worfshop (WS-11-08), pp. 27-32, 2008

[5] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. Rose and R. Buyya, "Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, in Software: Practice and Experience (SPE), Volume 41,Number 1, ISSN: 0038-0644, Wiley Press, New York, USA., pp. 23-50,2011.

[6] B.Wickremasinghe, R. N. Calheiros and R. Buyya, "Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications", in Proc. of Proceedings of the 24th International Conference on Advanced Information Networking and Applications (AINA2010), Perth, Australia, pp.446-452, 2010.

[7] J. Holland, "Adaptation in Natural and Artificial Systems," University of Michigan Press, Ann Arbor, ISBN: 0262-58111-6, 1975.

[8] C. Voudouris and T. Edward,"Guided Local Search.Technical Report CSM-247," Department of Computer Science, University of Essex, UK,August 1995.

[9] L. W. Barry, "Solving Combinatorial Optimization Problems Using a New Algorithm Based on Gravitational Attraction," Ph.D. Thesis, Florida Institute of Technology Melbourne, FL, USA, May 2004.

[10] S. R. Balachandar and K. Kannan, "Randomized gravitational emulation search algorithm for symmetric traveling salesman problem," Applied Mathematics and Computation, 192(2), pp. 413–421, 2007.

TABLE II: Simulation scenario and calculated overall average response time (RT) in (ms) using one DC

Serial No	Cloud Configuration	Data Center specification	RT in ms for GA-GEL	RT in ms for ACO	RT in ms for SHC	RT in ms for GA	RT in ms for FCFS
1	CC1	OneDC with 25 VMs .	326	328.98	329.02	329.01	330.11
2	CC2	OneDC with 50 VMs.	324.27	327.63	329.01	329.01	329.65
3	CC3	OneDC with 75 VMs.	232.52	238.12	329.34	329.34	329.44

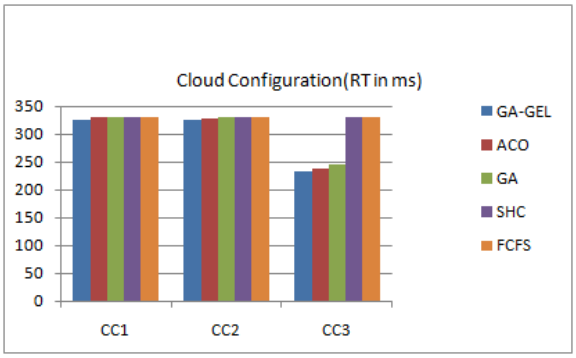

Fig.3. Performance analysis of proposed GA-GEL, with ACO, GA,SHC, FCFS Result using one Datacenter.

TABLE III: Simulation scenario and calculated overall average response time (RT) in (ms) using two Data Center

Sl No	Cloud Configuration	Data Center specification	RT in ms for GA-GEL	RT in ms for ACO	RT in ms for SHC	RT in ms for GA	RT in ms for FCFS
1	CC1	Two DCs with 25 VMs each.	351.32	354.72	360.77	365.44	351.32
2	CC2	Two DCs with 50 VMs each.	346..89	349.89	355.72	360.15	346..89
3	CC3	Two DCs with 75 VMs each.	345.98	348.68	355.32	359.73	345.98
4	CC4	Two DCs with 25, 50 VMs.	343.27	346.57	350.58	356.72	343.27
5	CC5	Two DCs with 25, 75 VMs.	342.66	347.86	351.56	357.23	342.66
6	CC6	Two DCs with 75, 50 VMs.	340.37	350.47	352.01	357.04	340.37

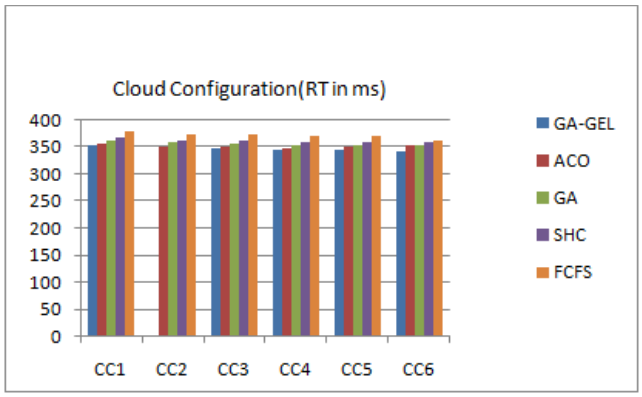

Fig. 4. Performance analysis of proposed GA-GEL, with ACO, GA,SHC, FCFS Result using two Datacenter.

TABLE IV: Simulation scenario and calculated overall average response time (RT) in (ms) result using three Data Centers

Sl No	Cloud Configuration	Data Center specification	RT in ms for GA-GEL	RT in ms for ACO	RT in ms for SHC	RT in ms for GA	RT in ms for FCFS
1	CC1	Each with 25 VMs .	343.25	345.68	350.32	356.82	363.34
2	CC2	Each with 50 VMs .	341.43	344.86	350.19	355.25	363.52
3	CC3	Each with 75 VMs	338.52	340.62	346.01	350.73	361.56
4	CC4	Each with 25, 50 ,75VMs.	343.25	345.68	350.32	356.82	363.34

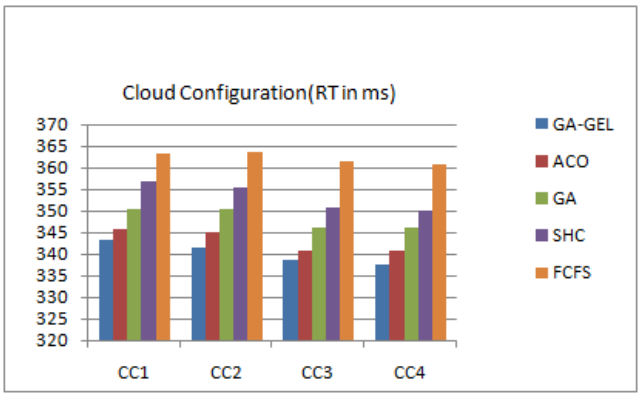

Fig. 5. Performance analysis of proposed GA-GEL, with ACO, GA,SHC, FCFS Result using three Datacenter.

TABLE V: Simulation scenario and calculated overall average response time (RT) in (ms) using four Data Center

Serial No	Cloud Configuration	Data Center specification	RT in ms for GA-GEL	RT in ms for ACO	RT in ms for SHC	RT in ms for GA	RT in ms for FCFS
1	CC1	Each with 25 VMs .	337.51	341.46	348.85	354.35	359.35
2	CC2	Each with 50 VMs .	336.38	339.78	345.54	350.71	356.93
3	CC3	Each with 75 VMs	335.49	336.56	340.65	346.46	352.09
4	CC4	Each with 25, 50 ,75VMs.	332.61	334.32	337.88	344.31	351

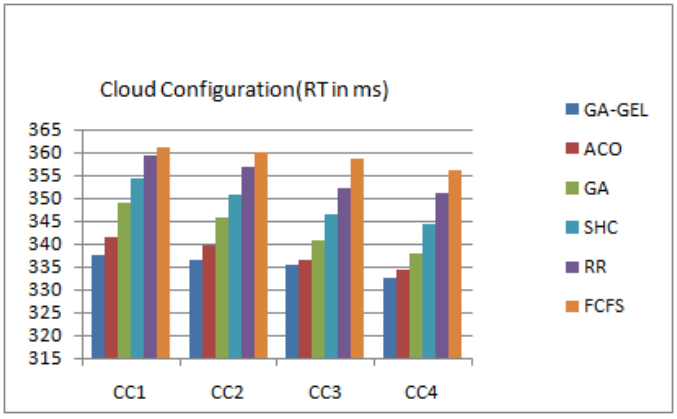

Fig. 6. Performance analysis of proposed GA-GEL, with ACO, GA,SHC, FCFS Result using four Datacenter.

TABLE VII: Simulation scenario and calculated overall average response time (RT) in (ms) result using Five Data Center

Sl No	Cloud Configuration	Data Center specification	RT in ms for GA-GEL	RT in ms for ACO	RT in ms for SHC	RT in ms for GA	RT in ms for FCFS
1	CC1	Each with 25 VMs .	329.34	331.45	335.64	342.86	352.05
2	CC2	Each with 50 VMs .	317.38	321.12	326.02	332.84	345.44
3	CC3	Each with 75 VMs	316.43	319.89	322.93	329.46	342.79
4	CC4	Each with 25, 50 ,75VMs.	315.53	317.65	319.98	326.64	338.01

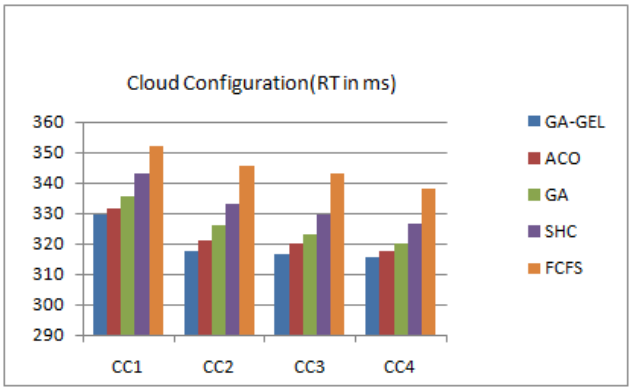

Fig. 7. Performance analysis of proposed ACO with GA, SHC, FCFS and RR for five data centre

TABLE VIII: Simulation scenario and calculated overall average response time (RT) in (ms) result using six Data Center

Sl No	Cloud Configuration	Data Center specification	RT in ms for GA-GEL	RT in ms for ACO	RT in ms for SHC	RT in ms for GA	RT in ms for FCFS
1	CC1	Each with 25 VMs .	321.26	323.98	330.54	336.96	349.26
2	CC2	Each with 50 VMs .	314.23	316.48	323.01	331.56	344.04
3	CC3	Each with 75 VMs	311.34	313.56	321.54	327.78	339.87
4	CC4	Each with 25, 50 ,75VMs.	302.5	309.66	315.33	323.56	338.29

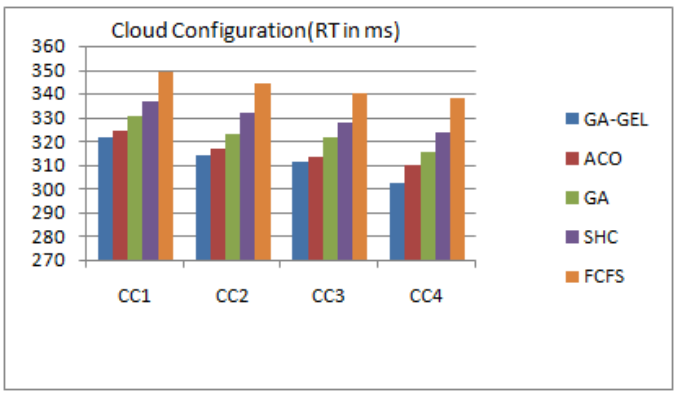

Fig. 8. Performance analysis of proposed GA-GEL, with ACO, GA,SHC, FCFS Result using six Datacenter