International Journal of Control Theory and Applications

ISSN : 0974-5572

Volume 9 • Number 45 • 2016

Task Allocation Using Big Bang-Big Crunch in Cloud Infrastructure

Punit Gupta[a] and S.P. Ghrera[a]

[a]Department of Computer Science Engineering, Jaypee University of Information Technology Himachal Pradesh, India

E-mail: punitg07@gmail.com

Abstract: Cloud computing is now trending and more popular in these days for the computation and adopted by many companies like Google, Amazon, Microsoft etc. As the cloud size increases with increase in number of data center power consumption over a data center increases. As number of request over the data center increase with increase in load and failure probability over a data center. So the requests need to be balanced in such manner which having more effective strategy for resources utilization, request failure and improved system reliability. A recent survey on cloud computation shows that the failure probability of a server, increasing in a linear way due to increase in number of independent resource (processors) resulting in request failure at data centers. So to overcome these issues in cloud Infrastructure as a service (IaaS), we have proposing a BB-BC (Big Ban-Big Crunch) algorithm for task allocation to minimize the request scheduling time and improve QoS (Quality of Service) over a data center. Proposed algorithm has proven to have better performance in term of load and request execution rate as compared to previously proposed task allocation algorithm for cloud IaaS.

Keywords: *Cloud computing, QoS, Resource Utilization, Failure probability, Reliability, Cloud Infrastructure as a service.*

1. INTRODUCTION

Cloud computing is more popular and promising epitome for both consumer and provider in different field like engineering mathematics and highly used in business industries in homogeneous and heterogeneous way. Request from different kind of area are served by data centers in cloud environment also increase power consumption. However, to keep computing large scale requests on data centers required huge amount of power, which leads to high power consumption. Request type also affect the services *i.e.*, private or public requests. As per survey in 2006, data center consumed around 4.5 billion kWh, which is equal to 1.5% of total power consumed by USA, and increasing 18% yearly [1]. In general Cloud Computing deals with various issues listed as follows: 1) As cloud computing adopted by industry and number of user also rapidly growing with number of data centers with increasing power consumption. 2) Task allocation among data centers without having information of QoS provided by them.3) Current task allocation algorithms focus on load balance when request load increases but not on failure probability of server. 4) High loaded data centers has high failure probability to compute requests and may be due to high load theses data center slow down which is not good for user as well cloud provider.5) Some data centers having less load compare to highly loaded data center and they are under maximum load but high loaded data center computing them with high failure rate.6) Some request are need to be computed with QoS but due to high load and fault rate they may the QoS promised which is not appropriate to user and will be a critical issue. 7) As per recent study [4-7], utilization of data centers is major problem because 60% data centers are idle and most of 20% data centers are fully utilize and wastes of resource. This show poor utilization of resources but this shows importance of new approach that have sufficient strategy to minimize wastes of resources as much as and increasing reliability by allocating task over resources which in case of Cloud is VM with low failure probability to provide high QoS to users.

This paper focus on reduction the scheduling delay in task allocation algorithm that emphasis hardware capability of cloud. Proposed algorithm lead computation with execution time in new directions and improved utilization of all resources accordingly. There are many factors of cloud computing which are untouched and can give best results and optimize the computation schema, request failure is one of them. VM or request failure at a data center depends on configuration of server hardware, high configuration of hardware and most priory load on a data center, that can be network load, storage load, computational load. Data centers are processing the requests which are scheduled by scheduler using a scheduling algorithm and at the end they need to be computed on allocated resources which may be faulty or over loaded. There is always a fixed capacity of data center to serve request beyond that data center need extra resource to do extra computation with the same hardware configuration, here is chance of task failure or some interruption during the process due to limited resource. This paper proposed an approach to avoid such situation with efficient manner as well continue task computing without any interruption.

Data center applies an algorithm to allocate resource to complete request with some configuration which makes it computable as soon as possible to serve request and response them, but some time due to more than its request server capability request comes and queued and data center compute the task with its maximum serve capability and finish the computation and it's increase the load and there are chances of system failure which is not appropriate and cause loss of money. This scenario may be coming into existence and decrease the productivity and down the system performance. There are many algorithms to find least loaded server, which can provide service efficiently and economically.

This paper proposes an dynamic way to minimize the computation and maximize utilization of resource by allocation resource to request by getting centre of mass from previous iteration and improve in next integration which make it more efficient and reliable computation over cloud environment. Proposed algorithm emphases on learning base algorithm to find an appropriate solution which cannot be achieved using static algorithm like max-min, min-min and ant colony optimization.

2. RELATED WORK

Cloud Computation is a new domain and need more research to make it more reliable, in order to have a better user experience in term of task computation.

We have proposed a scheduling algorithm which is based BB-BC to complete task scheduling with least scheduling time and better utilization.

Many researchers have done research and introduce us some beneficial and optical scheduling algorithm. [1] Proposed and algorithm Min-min algorithm, in this choose least the completion time of task and schedule to serve accordingly. In this paper they proposed load balance Min-min algorithm which having basic properties of Min-min algorithm and consider minimizing completion of all request. In this proposed three level service model used.

1. **Request manager :** To take request and forward to service managers.
2. **Service manager :** Various manger works or task and dispatch them to respective service node.
3. **Service Node :** Service node provide service to request which came to request mode

They have merged two approaches (OLB Opportunistic load balancing and load balance min-min) scheduling algorithms in this model. The main focus of combined approaches is to distribute the request or dispatched task basis of their completion time to suitable service node via an agent. This approach not saying about main system, suppose if request are somehow moving or scheduled in the same server and due to lots of load sever need more power to complete these request and more physical heat will generate and to stop heating system need an external cooling system which also lead to extra power source and one more important thing is due to overheating system performance slow down The same way [2] proposed and another algorithm for task scheduling, this paper proposed VM resource allocation basis on genetic algorithm to avoid dynamic VM migration to completion of request. They have proposed a strategy to share or allow resource equally to VM so it can work fast and minimize response time to subscribe. They also proposed hotspot memory (virtual memory) assignment and dispose that after completion of request via remapping of VM migration. Here VMware distribution tool is used to schedule computation work in a virtual environment. As genetic algorithm characteristics is to find best, fittest VM in terms of Cloud computation.

This paper checks fitness of each VM and schedule task accordingly. When creating a VM a process executes to create that and increase process work that also lead to more process and increase energy consumption.

Hu, Jinhua et al.[3] Proposed another scheduling algorithm, this paper proposed an approach for collective collaborative computing on trust model. The trust value taking as a factor for task scheduling, trust value mutually took from consumers as well service provider, which make it fail free execution environment. Here they have proposed a mathematical equation to calculate the Reputation point which enhances the reputation of VM in terms of fast execution and type of task. If the reputation of VM is high them more task allocation will be happening to that VM. To calculate Reputation many factors have to consider which also reflect QoS of cloud computing. This paper also proposed a way to serve a request reliability, as well trust management with a reputation of VM factor which are lead to trustworthy. Trust has calculated by a mathematical equation and schedule accordingly.

Hu, Jinhua et al. [4] proposed a live VM migration algorithm, this paper proposed a method for VM live migration with various resource reservation system. VM migration is taking place on the basis of source machine load, if the load is high then it can wear, during execution of the request it migrates the VM to another server or data centers to complete the task without interruption for better performance. Resource reservation done both sides, i.e., Source machine and target machine as well will in such manner CAP (maximum availability of CPU) allocate them and adjust memory resource dynamically. At the end of target machine, they properties time bound program which will keep monitoring for cup resource utilization. Memory Reservation done by allocating crating certain number of VM and when the migration process comes into existence these VM got shut down to evacuate the space to migrate VM. Sometime it may be possible that target machine not having enough space to migrate in such condition that physical machine should remove from candidate machine for migration and which physical machine having the capability or enough space will lead to migrate VM. This paper implemented and simulated using Xen Virtualization.

Barroso et al.[5] This paper proposed an algorithm, dynamic and integrated resource scheduling algorithm for Cloud Data center which balance load between servers in overall run time of request, here they are migrating an application from one data center to another without interruption. Here they are introducing some measurement to ensure load balancing. They have given a mathematical reputation to calculate imbalance load to calculate average utilization to its threshold value to balance load. To implement DAIRS they have used physical server with physical cluster and Virtual servers with virtual cluster. Application migration saves time instead of migrating whole VM data.

Barroso et al. [6] - proposed a most known base scheduling algorithm ACO (ant colony optimization) they proposed ant colony optimization algorithm to load balance by distributing request in a cloud computing environment. This paper proposed LBACO with dynamic load balancing strategy to distribute load among the node.

The problem with traditional ACO in cloud is that it's a schedule task to most frequent (high pheromone intensity) node, if what if node is bearing heavy load in such situation may create a problem of overhead. This paper proposed and LBACO algorithm to reduce such problem. In this algorithm decrease the time of computation and monitor load on each VM with tracking previous scheduling.

Xiaobo et al. [7] proposed and Real-time VM provisioning model, which is based on energy models which follow a Min-Price RT-VM Provisioning to allocate VM.Suraj, S. Rin et al. [14] proposed a genetic algorithm for task allocation in cloud environment with least execution time and maximum resource utilization. Above proposed algorithms are either general scheduling algorithms or proposing a strategy to improve resource utilization and has assumed the cloud to be non-faulty. In real world faults occur at data center due to hardware failure or software failure, which may decay the reliability of system to process requests.

So in next section we have proposed an algorithm, which will deal with such problem in cloud, and requests have to be completed within less execution time and best start time. Proposed algorithm tries to insure lease scheduling delay to find global best solution.

3. PROPOSED MODEL

In above section existing proposed algorithms are either discus about task scheduling or resource utilization and some of them talk about task or VM migrating to fulfil requests but the existing algorithm are static or dynamic in nature, and they may suffer from local minima solution considering that as the best solution. But a better solution for task allocation may be possible. So to overcome these learning based algorithms were proposed like Genetic algorithm and PSO (particle swarm optimization). The issue with these algorithms is that they have very high time complexity more over they depend upon the iteration and the initial; population size, which affects their solution. If the population size of the iterations/generation are less then there less probability to get best solution.

To overcome these problems and find the same best solution is less time complexity, we have proposed a task allocation algorithm based on Big Bang-Big Crunch (BBC) algorithm. This algorithm is motivated from the physics behind creation of universe theory in astrology. BBC algorithms refers to the evolution of universe and end of universe which says universe is a finite space which once expanded with force binding it and will end into a single point referred as a black hole. Algorithm also suggests that any element of universe cannot be suggested as center of universe. Similar to this we have proposed a task allocation algorithm to find a single best solution from a large set of solutions. Where generation of universe is referred as Big Bang phase and dissipation of universe in black hole near the center is said to big crunch phase.

Proposed algorithm uses Poisson probability distribution for random request at virtual machine i.e. at host and data center level. Based on computing capability of a system, we have proposed a task allocation policy to minimize the total makespan over the system and reduce time complexity of solution. According to algorithm collect the information of data center resources and its capability. Proposed algorithm is similar to Genetic algorithm (GA) but the problem size reduces after each phase and will give you a single point solution i.e. the global solution. But in existing GA the population size remain same and there is no guarantee that the global best is achieved.

Proposed algorithm is divided into four phases which are as follows:

1. Big Bang / Initialization phase
2. Evaluation phase
3. Crossover / Center of mass
4. Big Crunch phase

1. **Initialization:** In this phase we have a set of tasks (T1, T2, T3, T4, T5, T6…. T *n*) and a set of resources in term of virtual machine (VM1, VM2, VM3, VM4, VM5…. VM m) are pre allocated on hosts in distributed data centers. Here we initialize asset of sequences or schedules allocated randomly, each sequences act a chromosomes for genetic algorithm. The complete set of chromosomes is said to be a population, acting as a input for algorithm.

   In this phase we evaluate the fitness value for each set of sequence or chromosome, which depends up on the computing capability, total time taken to complete the schedule.

   Where If

   VM_MIPS *i* : MIPS of i^{th} virtual machine

   T_Leng *i*: Length of i^{th} Task

   Fitness_chomosomei : Fitness value of chromosome/sequence *i*

   Then the predicted time to complete a task Ti is defined:

$$\text{T\_Exei} = \left(\frac{\text{T\_Lenght } i}{\text{VM\_MIPS}i}\right) \quad (1)$$

$$\text{Total\_time} = \sum_{i=1-n} \frac{\text{T\_Lenght } i}{\text{VM\_MIPS}i} \quad (2)$$

   The fitness value for a chromosome is defined by the fitness function gives as:

$$\text{Fitness\_chromosome}_i = \alpha(\text{Total\_time } i) \quad (3)$$

   Based on the fitness value of chromosome the fittest one is selected having least fitness value. The population is sorted based on the fitness value and best two are selected from next phase.

3. **Crossover:** In this step two fittest solutions based on least make span are selected based on the center of mass and the population sequence near to center of mass are selected for cross over. The steps for selection are as follows:

   a) Find Center of mass of fitness values of the sequences in population using mean.

   b) Find the sequence having fitness value with least difference from the center of mass.

   We have used multi point crossover to generate new fittest sequences/ chromosome. Steps to generate crossover are as follows.

   - The two fittest chromosomes are selected with least difference from center of mass.
   - A new fittest chromosome is generated using multi point cross over by interchanging the set of schedules between two chromosomes.
   - The new chromosome replaces the chromosome with highest fitness value.

$$\text{C\_Mass} = \frac{\sum_{i=0}^{n} \text{Fitness\_chromosome}_i}{\text{Population Size()}} \quad (4)$$

4. **Big Crunch phase :** In this phase new merging the new offspring, which can be better solution from all other chromosomes/sequences. A new population is generated with new offspring generated and removing two chromosomes with least fitness value i.e. the worst solution from the population, decreasing the population size by one.

After specific count of iteration of proposed algorithm stop the iterations, when the population size is one. This is said to be the stopping condition of BBC and the last solution is the best solution for a definite time interval and iteration. Each iteration can also be referred as “generation” to create new fittest solution.

3.1. Proposed algorithm

Big Bang-Big Crunch Algorithm Task Allocation

```
Algorithm:-BBC (VM List VM_l , Task list T_i, population
size Po , Iteration Itr)
//Input : Po, VM_l, Itr and T_i
1.  VM_i ← VM_List() ;
2.  i ←No. of VM
3.  T_i ← Task_List();
4.  C ← BBC_algo(Vmi ,Ti, Po, Itr);
5.  Allocate_Resource( C);    // processing the client
request.
7.  End
```

Figure 1: Proposed Algorithm initialization

BBC Algorithm

```
BBC_algo(VMi ,Ti, Po, Itr)
//Input : Po, VM_l, Itr and T_i
1.  Po ←  Initiate_Population( Ti,) ;
2.  Evaluation();
3. CenterMass();// find mean of all fitness values
4.C1 ← getFittest1();
5.C2 ← getFittest2();
6. Po ← Crossover(C1,C2)
7.  Big_Crunch(Po , C1, C2);
8.  Return( getFittest());
6.  End
```

Figure 2: Proposed Algorithm

```
Evaluation
1. Evaluation(){
2. For each Ci   i=0 - po
3.       For each Ti
4.           temp=  (Ti/Vmi )
5.           Fitnessi = Fitnessi +temp
6.       End
7. END
8. }
```

Figure 3: Proposed Evaluation Phase

```
Big_Crunch

1. Big_Crunch(Po , C1, C2)
2. {
3.   Delete_worst(); // delete element with
     least fitness value.
4.   Delete_worst();
5. }
```

Figure 4: Big Crunch Phase

```
1. Allocate_Resource(C){
2. Ci ← Getchromosomes();
3. For each Ci
4.       Allocate(Ci);
5. END
6. }
```

Figure 5: Proposed Allocation Phase

Figure 1-5 shows pseudo code for various phased of proposed algorithm. Proposed algorithm provides a benefit over existing static scheduling algorithm, that it can search for best global solution rather than assuming the local best solution as the best solution. Moreover, the proposed algorithm takes into consideration center of mass of the existing solution to find new solution in next iteration so helps is finding new best solution by shifting the center of mass toward the best fit., which helps in find a solution with similar high utilization and less time complexity as compared to Genetic algorithm.

4. EXPERIMENTAL RESULT

For simulation we CloudSim 3.0 power module is used. CloudSim 3.0 provides cloud simulation and predefined power model simulation. We have simulated proposed BB-BC algorithm in CloudSim. The proposed algorithm is being tested over various test cases with 10 servers D0-D9 and Poisson distribution model for random request in distributed environment.

Testing of proposed algorithm is done with basic Genetic algorithm proposed by Suraj, S. Rin [14]. Testing is done for 1000, 1500, 2000, 2500, 3000, 3500 requests with population size been 100, 200, 300, 400. Iteration for simulation of each simulation is 100. Table 1 show the simulation setup and server configurations.

Table 1
Stimulation settings

Server	*RAM(Mb)*	*MIPS*	*Storage (Gb)*	*Core*	*PE*	*HOST*
D0	2000	10000	100000	4	6	2
D1	2000	10000	100000	4	6	2
D2	2000	10000	100000	4	6	2
D3	2000	10000	100000	4	6	2
D4	2000	10000	100000	4	6	2
D5	2000	10000	100000	4	6	2
D6	2000	10000	100000	4	6	2
D7	2000	10000	100000	4	6	2
D8	2000	10000	100000	4	6	2
D9	2000	10000	100000	4	6	2

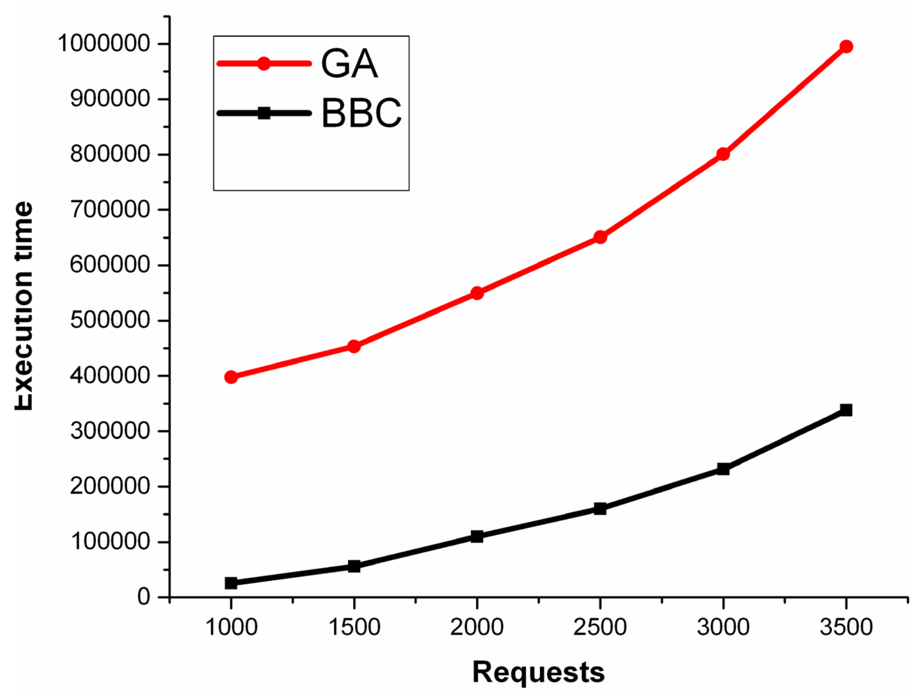

Figure 6: Comparison of improvement in execution time

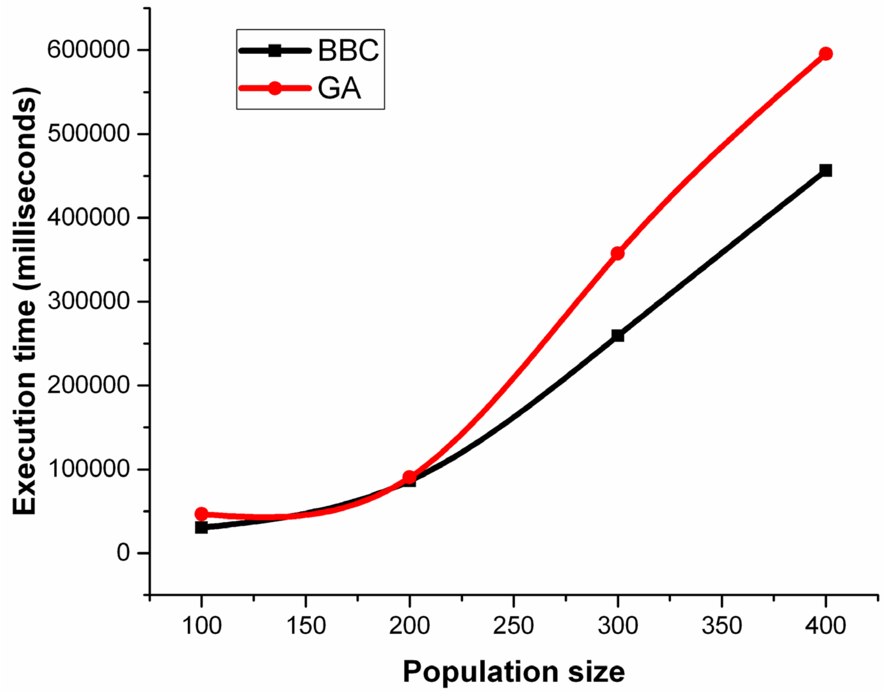

Figure 7: Comparison of improvement in execution time with changes in population size

Figure 6 compares the improvement in execution time with increase in number of requests over the system. Figure 7 shows improvement in execution time which reduces over the proposed system with increasing population size over the system.

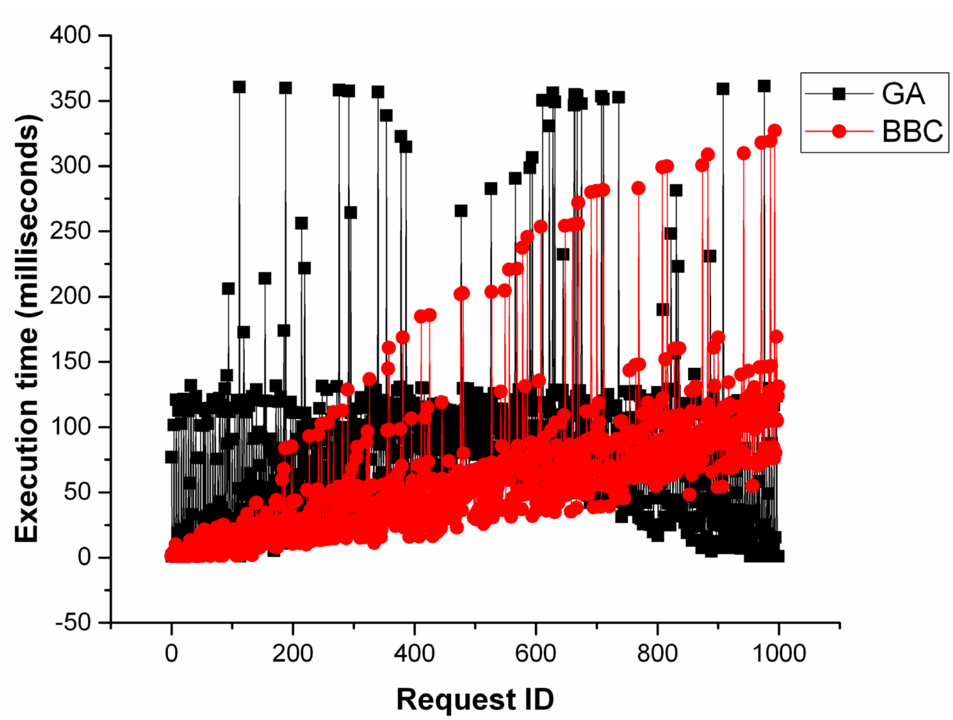

Figure 8: Comparison of execution time of individual requests. For 1000 request count

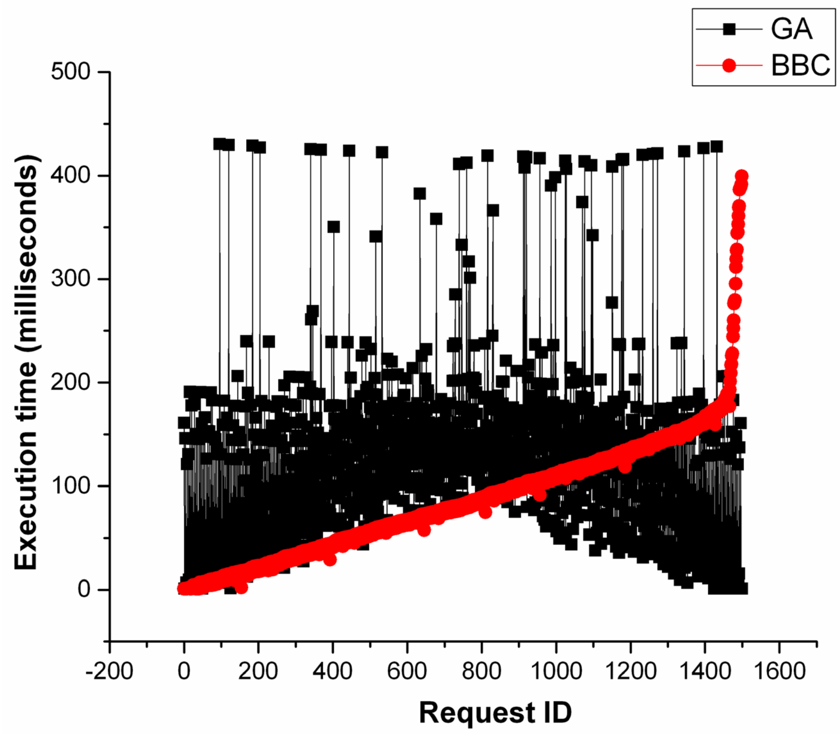

Figure 9: Comparison of execution time of individual requests. For 1500 request count

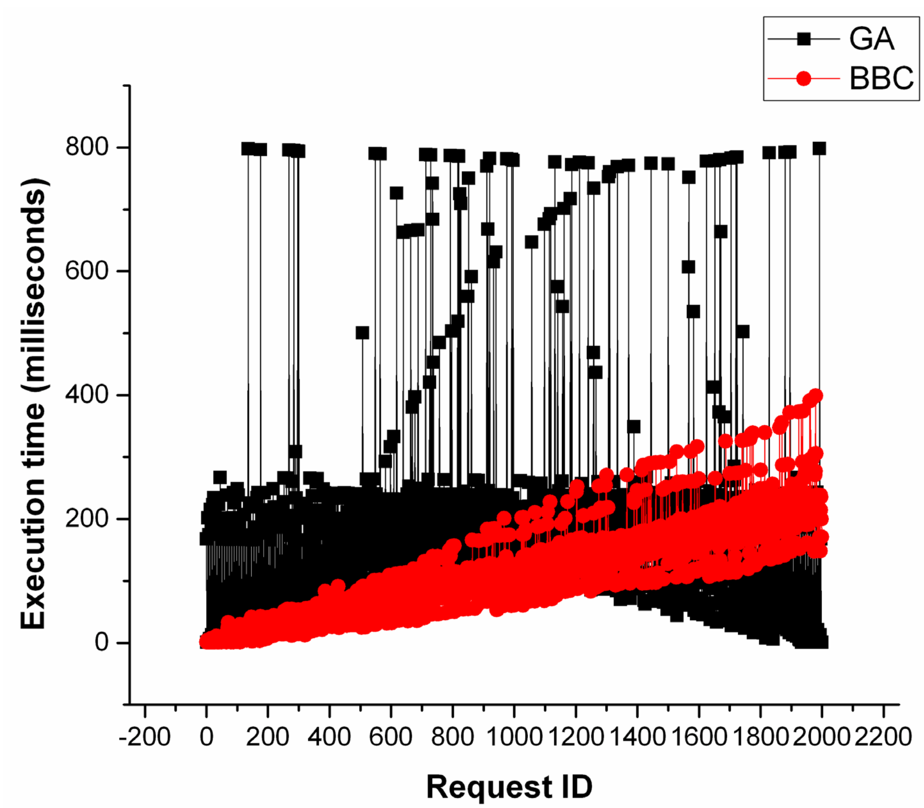

Figure 10: Comparison of execution time of individual requests. For 2000 request count

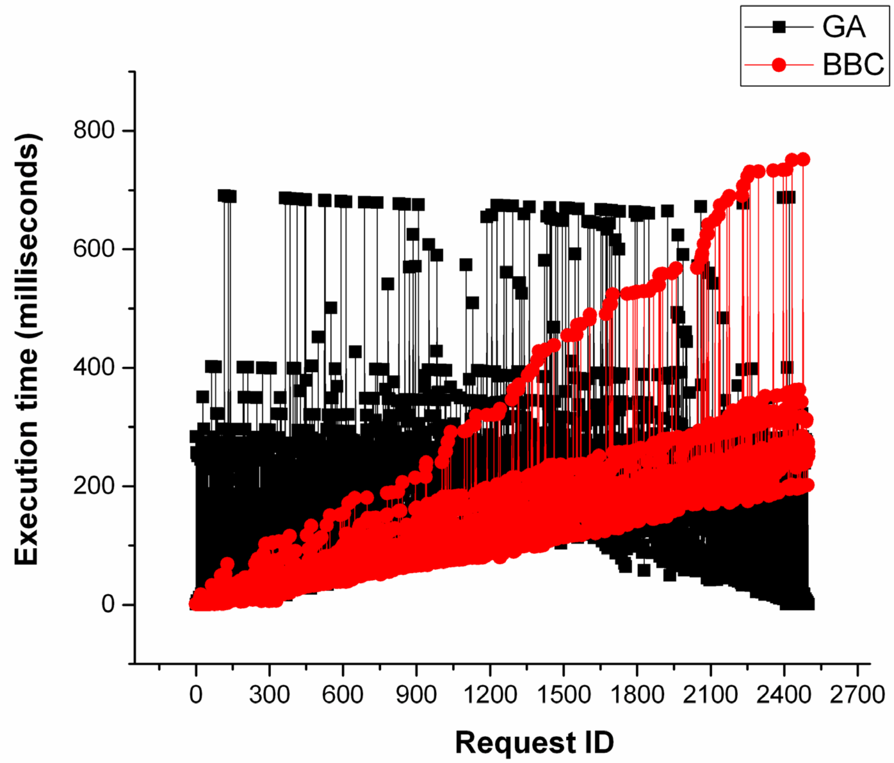

Figure 11: Comparison of execution time of individual requests. For 2500 request count

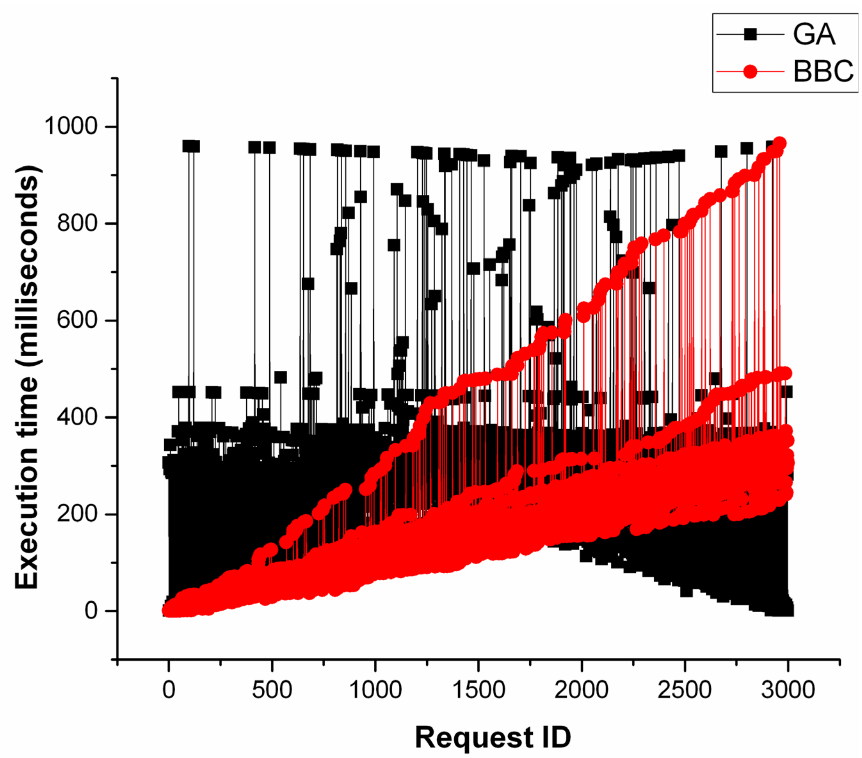

Figure 12: Comparison of execution time of individual requests. For 3000 request count

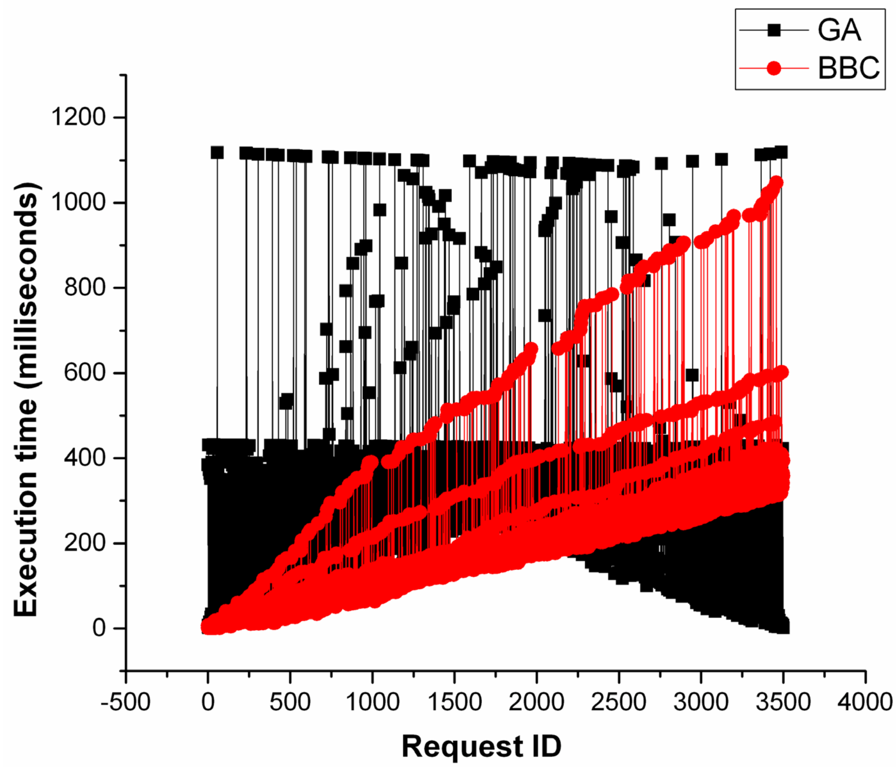

Figure 13: Comparison of execution time of individual requests. For 3500 request count

Figure 8-13 shows the improvement in distribution of execution time for varying request count using proposed BB-BC algorithm for task allocation over cloud. The execution time of the requests has improved and majority of requests are completed in small execution time.

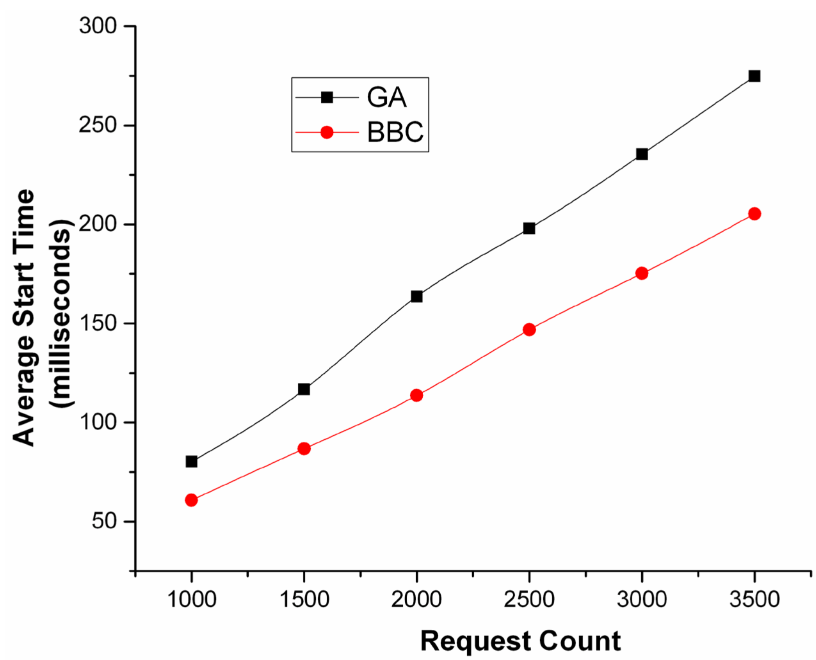

Figure 14: Comparison of Average Start time of system with increase in request count

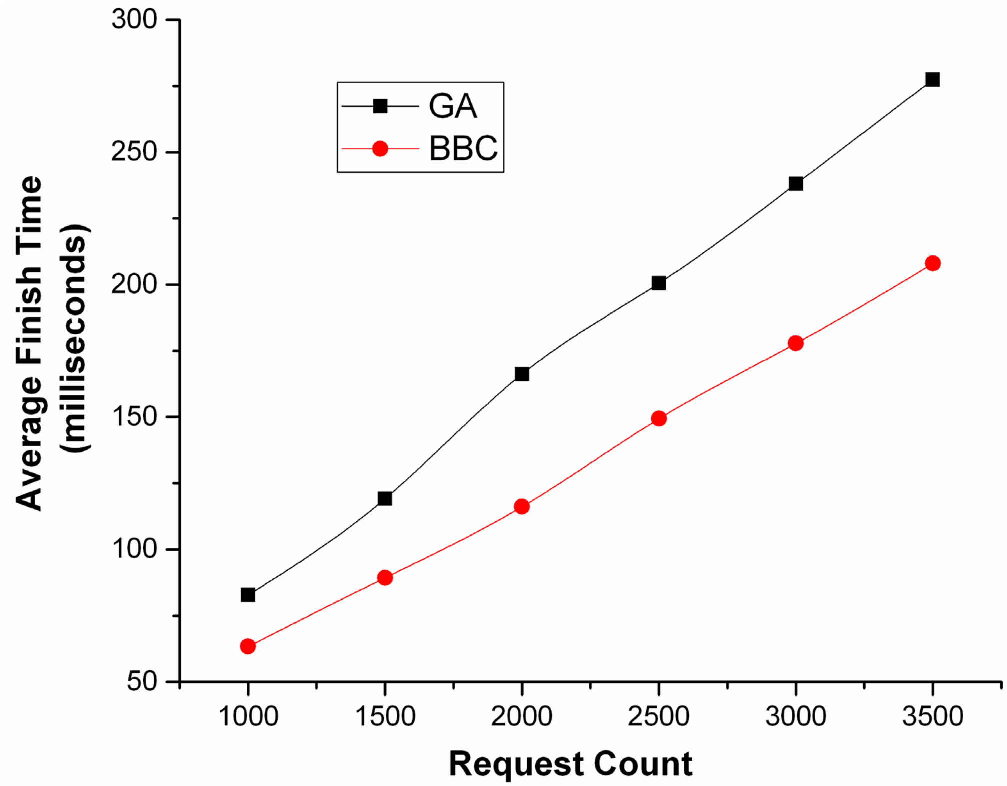

Figure 15. Comparison of Average Finish time of system with increase in request count

Figure 14 discourses the improvement in average start time with increase in number of request over the system, which shows the proposed algorithm proves to provide better start time them the conventional genetic algorithm. Figure 15 discourses the improvement in average finish time which reduces with increasing requests over the system the experiment has been performed over 1000, 1500, 2000, 2500, 3000 and 3500 request count. Proposed algorithm proves to provide reduced finish time as compared to existing genetic algorithm.

5. CONCLUSION

From experimental result section, it is clear that proposed BB-BC provides better QoS (Quality of service) as compared to previous proposed GA algorithm. The main idea of this algorithm in cloud computing is to complete maximum number of requests with least execution time, proposed algorithm shown that it can provide better execution time over large requests with reduces average start time and average finish time over the system. Proposed algorithm reduces the number of iteration required to achieve a global best solution with least scheduling time. This strategy has proven that it provides better QoS in term of high reliability with increase in number of requests and resources with least scheduling time with decrease in execution time with increase in population size and number of requests. Proposed algorithm insures the schedule achieve is global best solution.

REFERENCES

[1] Brown, Richard. "Report to congress on server and data center energy efficiency: Public law 109-431." Lawrence Berkeley National Laboratory (2008).

[2] Wang, Shu-Ching, Kuo-Qin Yan, Wen-Pin Liao, and Shun-Sheng Wang. "Towards a load balancing in a three-level cloud computing network." InComputer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on, vol. 1, pp. 108-113.

[3] Hu, Jinhua, Jianhua Gu, Guofei Sun, and Tianhai Zhao. "A scheduling strategy on load balancing of virtual machine resources in cloud computing environment." In Parallel Architectures, Algorithms and Programming (PAAP), 2010 Third International Symposium on, pp. 89-96.

[4] Hu, Jinhua, Jianhua Gu, Guofei Sun, and Tianhai Zhao. "A scheduling strategy on load balancing of virtual machine resources in cloud computing environment." In Parallel Architectures, Algorithms and Programming (PAAP), 2010 Third International Symposium on, pp. 89-96.

[5] Barroso, Luiz André, and Urs Hölzle. "The case for energy-proportional computing." Computer 12 (2007): 33-37.

[6] Bohrer, Pat, Elmootazbellah N. Elnozahy, Tom Keller, Michael Kistler, Charles Lefurgy, Chandler McDowell, and Ram Rajamony. "The case for power management in web servers." In Power aware computing, pp. 261-289.

[7] Fan, Xiaobo, Wolf-Dietrich Weber, and Luiz Andre Barroso. "Power provisioning for a warehouse-sized computer." In ACM SIGARCH Computer Architecture News, vol. 35, no. 2, pp. 13-23.

[8] Lefurgy, Charles, Xiaorui Wang, and Malcolm Ware. "Server-level power control." In Autonomic Computing, 2007. ICAC'07. Fourth International Conference on, pp. 4-4.

[9] Mehta, Hemant Kumar, Manohar Chandwani, and Priyesh Kanungo. "On trust management and reliability of distributed scheduling algorithms." In Advanced Computing (ICoAC), 2010 Second International Conference on, pp. 46-50.

[10] Ye, Kejiang, Xiaohong Jiang, Dawei Huang, Jianhai Chen, and Bei Wang. "Live migration of multiple virtual machines with resource reservation in cloud computing environments." In Cloud Computing (CLOUD), 2011 IEEE International Conference on, pp. 267-274.

[11] Tian, Wenhong, Yong Zhao, Yuanliang Zhong, Minxian Xu, and Chen Jing. "A dynamic and integrated load-balancing scheduling algorithm for Cloud datacenters." In Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on, pp. 311-315.

[12] Li, Kun, Gaochao Xu, Guangyu Zhao, Yushuang Dong, and Dan Wang. "Cloud task scheduling based on load balancing ant colony optimization." In Chinagrid Conference (ChinaGrid), 2011 Sixth Annual, pp. 3-9.

[13] Kim, Kyong Hoon, Anton Beloglazov, and Rajkumar Buyya. "Power-aware provisioning of cloud resources for real-time services." In Proceedings of the 7th International Workshop on Middleware for Grids, Clouds and e-Science, p. 1-6.

[14] Suraj, S. R., and R. Natchadalingam. "Adaptive Genetic Algorithm for Efficient Resource Management in Cloud Computing." International Journal of Emerging Technology and Advanced Engineering 4.2 (2014): 350-356.