A Genetic Algorithm Based Task Scheduling for Cloud Computing with Fuzzy logic

Avtar Singh and Kamlesh Dutta

Department of Computer Science, National Institute of Technology / Hamirpur, Himachal Pradesh, India
avtarz@gmail.com, kd@nith.ac.in

***** Corresponding Author: Avtar Singh

Received July 14, 2013; Revised July 29, 2013; Accepted September 12, 2013; Published December 31, 2013

** Short Paper*

***Abstract*:** Cloud computing technology has been developing at an increasing expansion rate. Today most of firms are using this technology, making improving the quality of service one of the most important issues. To achieve this, the system must operate efficiently with less idle time and without deteriorating the customer satisfaction. This paper focuses on enhancing the efficiency of a conventional Genetic Algorithm (GA) for task scheduling in cloud computing using Fuzzy Logic (FL). This study collected a group of task schedules and assessed the quality of each task schedule with the user expectation. The work iterates the best scheduling order genetic operations to make the optimal task schedule. General GA takes considerable time to find the correct scheduling order when all the fitness function parameters are the same. GA is an intuitive approach for solving problems because it covers all possible aspects of the problem. When this approach is combined with fuzzy logic (FL), it behaves like a human brain as a problem solver from an existing database (Memory). The present scheme compares GA with and without FL. Using FL, the proposed system at a 100, 400 and 1000 sample size*5 gave 70%, 57% and 47% better improvement in the task time compared to GA.

***Keywords*:** Design, Algorithms, Performance, Experimentation

1. Introduction

As with the evolution of information technology and system virtualization, Cloud computing has emerged as a new computing force. The cloud environment is used in many areas, such as IT industries, educational institutes and other industries [10-12]. Cloud computing is provides the virtualization of resources to a cloud user (CU) using information technology [1]. The CU establishes Service Layer Agreement (SLA) with the cloud provider that provides storage and severs etc. to the CU as services and pays for those services [2]. A cloud provider defines a computing system as a cloud with are nothing more than inter-connected virtual machines. The cloud provider earns profit by processing user tasks on the cloud. Therefore, determining how to allocate the tasks or jobs to the cloud is an important and challenging issue. The main aim of cloud computing is to satisfy the customer needs with respect to the QoS, as defined in the SLA and to enhance the cloud provider's profit [3]. This can be achieved by having a good scheduling procedure for different user tasks because a better scheduling procedure needs to minimize the waiting time there by processing more jobs in a given time span and earning good profit by satisfying the user in minimum time. The algorithm based on Fuzzy-GA optimization, which evaluates the entire group of tasks in a job queue based on a prediction of the execution time of tasks assigned to certain processors and makes the scheduling decision [8]. Chen et al. [9], proposed a genetic algorithm-based resource scheduling for the fitness function, which were sub divided further into three. A linear combination of these function values was performed to obtain the fitness value. Some issues, such as resource fragmentation and low utilization, are caused easily by system scale expansion, virtual machines and migrations etc., which consuming high energy within an Internet Datacenter. Genetic algorithms (GAs) are one possible solution for satisfying the cloud computing goal. GAs involve finding the best solution among the available solutions, deciding whether a newer solution is better a

previous one, and if yes, replace the current solution with the newer one. GAs for task scheduling require the use of a fitness function comprised of parameters, such as the cloud usage cost, execution time, memory etc., as defined in the SLA document. In general, the cost parameter of the cloud with higher priority means that a user who paid more money for cloud resources will be higher in the hierarchy, regardless of the other condition. Various cloud users (CUs) can be classified based on the cloud packages. CUs with a higher package will have higher priority for processing jobs, and CUs with a lower package will have lower priority for processing the jobs. On the other hand, the main problem arises when the cost parameter of fitness function are the same, which is where the GA provides better results than common task scheduling algorithms, such as FIFO, Round Robin, Shortest Remaining time etc. The general GA for task scheduling has certain limitations that can be overcome using this procedure with fuzzy logic. Fuzzy logic is a solution to a problem that involves uncertainty. The process involves the mapping of real world parameters with the given system. This mapping is performed with help of stored parameters in the memory that can be updated with time.

2. Techniques

This section reports the general view of GA, GA for task scheduling and GA for task scheduling along with fuzzy logic (FL).

2.1 General view of the Genetic Algorithm

GAs are search methods based on the principles of natural selection and genetics. GAs encodes the decision variables of search problems into finite-length strings of alphabets of certain cardinality. The strings that are candidate solutions to the search problem are referred to as chromosomes, the alphabets are referred to as genes, and the values of the genes are called alleles. For example, in a problem, such as the travelling salesman problem, a chromosome represents the route, and a gene may represent a city. A measure for distinguishing good solutions from bad solutions is needed to evolve good solutions and to implement natural selection [4]. The measure could be an objective function that is a mathematical model or a computer simulation, or it can be a subjective function where humans choose better solutions over poorer ones. In essence, the fitness measure must determine the relative fitness of a candidate solution, which will then be used by the GA to guide the evolution of good solutions. Once the problem is encoded in a chromosomal manner and a fitness measure for discriminating good solutions from bad ones has been chosen, *evolve* solutions to the search problem can be started using the following steps:

a) Initialization

The initial population of candidate solutions is normally generated randomly across the search space. On the other hand, domain-specific knowledge or other information can be incorporated easily.

b) Evaluation

Once the population is initialized or an offspring population is created, the fitness values of the candidate solutions can be evaluated.

c) Selection

Selection allocates more copies of those solutions with higher fitness values and imposes the survival-of-the-fittest mechanism on the candidate solutions. The main idea of selection is to prefer better solutions to worse ones.

d) Recombination

Recombination combines parts of two or more parental solutions to create new, possibly better solutions.

e) Mutation

While recombination operates on two or more parental chromosomes, local mutations randomly modify a solution.

f) Replacement

The offspring population created by selection, recombination, and mutation replaces the original parental population. Many replacement techniques, such as elitist replacement, generation-wise replacement and steady-state replacement methods, are used in GAs.

g) Repeat steps 2–6 until a terminating condition is met [5].

2.2 Genetic algorithm for task scheduling

I. Initialization, As soon as a cloud server (CS) finds a processes to execute in the Ready Queue (RQ), the CS extracts information on the classification of the user according to whether they are a higher or lower priority user depending on the cost of their packages. Higher priority user's processes going to be executed first regardless of the other condition. The GA is applied the CS finds conflicts in the user priority. Fig. 1 presents the cloud queuing model architecture using a GA.

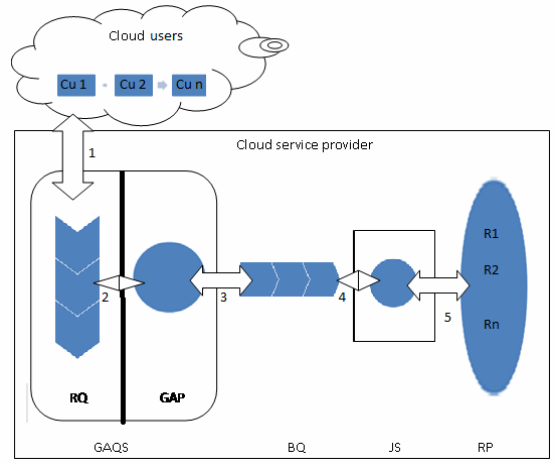

Fig. 1. Architecture of the Cloud queuing model using a Genetic Algorithm [6].

II. Evaluation/Selection

III. The CS arranges the processes in an order that leads to the minimum waiting time, which is done by going through the traditional task scheduling algorithms, such as round robin, Shortest First Time, etc. and

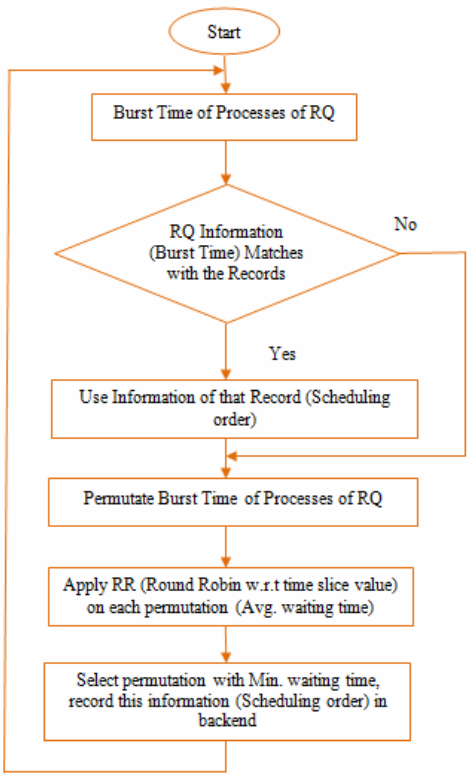

Fig. 2. Flowchart.

changing the sequence process in RQ then applying traditional algorithms one by one. The most common traditional method for a genetic algorithm for task scheduling is the Round Robin because it provides the best way of satisfying the cloud computing goal as described earlier [6, 7].

IV. Execution

V. Once a solution with a minimum waiting time is found, this procedure should be applied to the process in RQ. If the solution is similar to the previous one, no replacement is done else a replacement of the solution is required.

Fig. 2 presents a flow chart of the resource scheduling policy with the burst time of processes of a RQ information and matches with the records. If the record matches, then the information is used to create the scheduling order. the Burst time of process with RQ on both records is then permutated to determine if it a match or does not match. The average waiting time is then calculated by applying the round robin time with respect to time slicing. The permutation with the minimum waiting time is selected and this information (scheduling order) is recorded in Backend.

2.3 GA for Task Scheduling along with Fuzzy Logic

The use of fuzzy logic in this procedure is discussed. : This provides solution to a situation involving uncertainty [8]. Many real time applications exist, such as automatic coolant operation in mega industry. Fuzzy logic can be considered as a way of mapping a real world parameter with any given system. The fuzzy logic mechanism works on dividing the ranges of parameters into different classes, with each class having a value between [0, 1] (real values)

2.3.1 Mapping of FL with GA task scheduling

A simple GA is used for task scheduling as described above. The main concern is that when the RQ size is more than 4, the search time for determining the best scheduling order increases, and at each time, the same procedure is applied to the processes in the RQ. This searching overhead can be reduced significantly when the GA for task scheduling is embedded with FL as follows:

I. Starting with the same procedure described in section 2.2, independent of the size of the RQ, the scheduling order is retrieved

II. Keep the record of the burst time range (BTR) of the processes.

III. Dynamically divide the burst time into equivalent classes, such as processes with a burst time between (6-10) in class- 1, (11-15) in class-2, (16-20) in class-3, etc., until the end limit of the BTR is reached.

IV. Keeping records of the above information (I-III) having the BTR as a reference value for mapping as a single batch.

V. As the next job request job batch is received, find the BTR value for this batch, if this value is already in the records then apply the mapped information (scheduling order) else repeat step (I)

VI Repeat step (V)

3. Performance analysis

3.1 Genetic Algorithm for Task Scheduling

Suppose a CU sends *n* number of requests for the resources and those resources initially are stored in the RQ, such as P1, P2.... P*n* and requests come from the CU. The GA processor executes all possible sequences of task one by one using Round Robin scheduling. If there are *n* number of tasks ready to execute, the number of possible ways are *n*!. Here, three tasks are ready to be executed, so the possible way of performing the tasks into the Job scheduler (JS) are 3! Or 6 ways. All of them are discussed one by one. Note that the time quantum of the tasks is 3 and the burst time for [P1 = 3, P2 = 6, P3 = 4] for Round Robin scheduling operation.

According to Table 1 based on the processes using [P1, P3, P2], the average waiting time is minimal. Therefore, this particular sequence must be initially stored into the buffer queue and executing the processes in this order results in less waiting time. Therefore, the GA can reduce the waiting time of the overall system.

3.2 GA for task scheduling with Fuzzy Logic

According to Table 1 based on the GA for task scheduling n! (Size of RQ), the search needs to be done to

Table 1. Waiting Time for various execution orders of the processes.

Order of execution of process	Average waiting time
[P1, P2, P3]	5
[P2, P1, P3]	6
[P1, P3, P2]	4.3
[P2, P3, P1]	6
[P3, P2, P1]	6.3
[P3, P1, P2]	5.3

find the best order. The scheduling is going to be mapped with fuzzy logic; this information is kept as a record. Let the burst time for the process in RQ of size 3 be P1, P2, and P3 of 5, 10, and 15, respectively. The following steps are performed using this procedure:

I. Initially, the range is undefined, but when processes batch is received new BTR is 5-15.
II. The classes are made dynamically such as [5-10, 11-15]. Any burst time between [5-10] is going to be a part of class-1, so with class-2 (any burst time between [11-15])
III. By applying GA for task scheduling on the process processes batch of 3, suppose the best order with the minimum waiting time is P2, P3, P1.
IV. The BTR is recorded as a reference value for mapping the process execution order.
V. Next time a job batch of size 3, with a BTR ranging from 5 - 10, then use the already available mapping from the records. If mapping is unavailable then go to step (I)
VI Repeat step (V)

As the CS statistic background becomes stronger with time, the performance of this algorithm improves drastically compared to the conventional GA for task scheduling. In addition, the searching time can be speed up by having the recently used mapping in the higher priority for searching.

4. Result simulation

Here, the results were obtained from the time complexity of the conventional GA algorithm as O(n!), which is not applicable at any cost to finding best scheduling order each time respective to the size of the RQ. This factor can be compensated for using fuzzy logic with a conventional GA. Considering the case GA for task scheduling with fuzzy logic, whenever the same RQ occurred, instead of finding the same scheduling order, the previously recorded value respective to the BTR value is used, so running this case is O(1). The experiments were run on an Intel Celeron Core 2 Duo processor with a 1.60GHz and 4GB of RAM. Fig. 3 compares the GA (BLUE) and GAFL (RED). The data consisted of 5000 random sample of RQ of size 3. The burst time of the

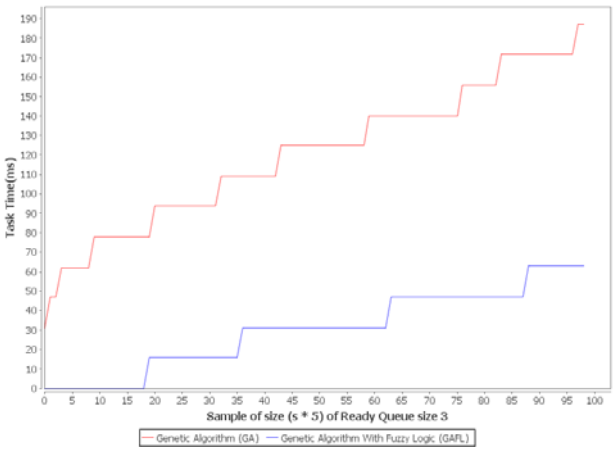

Fig. 3. Comparison of GA and GAFL with a sample size*5 for 100 versus the task time (Comparison using java programming language).

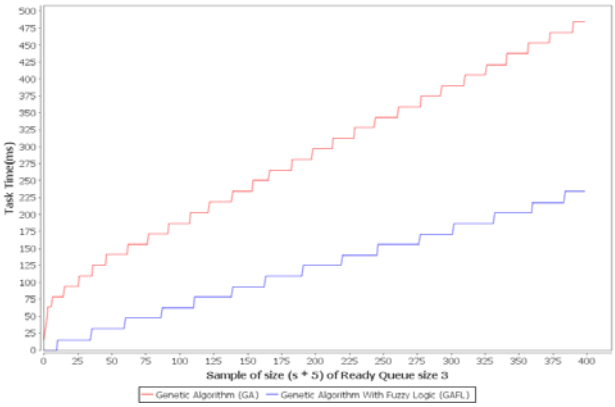

Fig. 4. Comparison of GA and GAFL with a sample size*5 for 400 versus the task time (Comparison using java programming language).

process ranged from [1, 100]. If the process burst time interval is [1, 25], the performance extends to 40% but the performance increases to 50%.with the initial start with interval [1, 10]. Fig. 3 shows that the sample size *5 versus the task time with GA and GAFL shows that the sample with size (s*5) of Ready Queue size 3 (25, 50, 100)* 5 = [125, 250, 500] had a task time of [15, 30, 55] ms in the case of GA with fuzzy logic, whereas the task time was [95, 120, 185] ms in GA. This shows that the GAFL produces 70% improvement over the GA with a sample size of 500.

Fig. 4 shows that the sample size *5 versus task time with GA and GAFL. The sample of size (s*5) of Ready Queue size 3 (100,200,400)*5 = [500, 1000, 2000] a task time of [55, 100, 200] ms in the case of GA with fuzzy logic, whereas it was [175, 275, 475] ms in the GA. This shows that GAFL give 57% improvement compared to GA with a sample size of 2000.

Fig. 5 compares GA and GAFL the sample size *5 versus the task time. The results showed that the sample of size (s*5) of a Ready Queue size 3 (100, 500, 1000)*5 = [500, 2500, 5000] had a task time of [50, 200, 500] ms in the case of GA with fuzzy logic, whereas it was [100, 450,

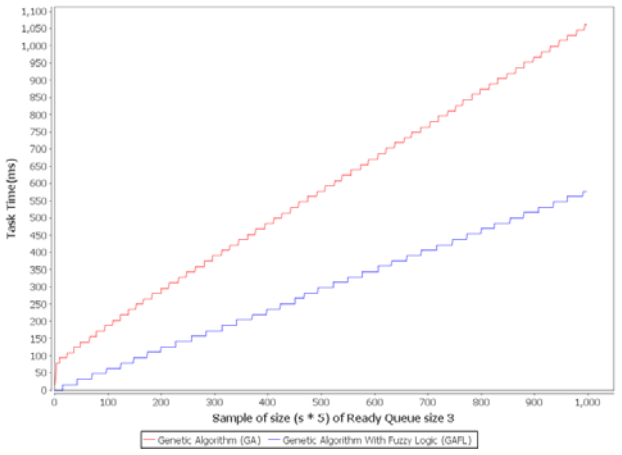

Fig. 5. Comparison of GA and GAFL with a sample size*5 for 1000 versus the task time (Comparison using java programming language).

1050] ms in GA. This shows that GAFL gave 47.61% improvement compared to GA at a sample size 5000.

These results show that as the sample size increases, the task time will also increase and the performance of the system will decrease.

5. Conclusion

This paper presents the results of a comparative analysis of a GA with and without fuzzy logic. GA is a heuristic search algorithm that always finds out a solution that takes the minimum time to execute or find the optimal solution from a set of possible solutions. The GA for task scheduling along with fuzzy logic enhances the performance of the conventional GA for task scheduling. Therefore, it provides benefits to the cloud user by satisfying the request for less time than actual and provides benefit to the cloud provider by allowing them to process more jobs than actual in a given time span. The performance of GAFL at a 100, 400 and 1000 sample size*5 gave 70%, 57% and 47% performance improvement in the task time compared to GA. The advantages of GA for task scheduling with fuzzy logic can enhance the speed of cloud computing significantly. This algorithm is applied only when there is a fitness function value conflict.

Acknowledgement

The authors are grateful to Director of National Institute of Technology for providing the facility to carry out this research.

References

[1] B. Hayes, “Cloud computing”, Communications of the ACM, vol. 51, no. 7, pp. 9-11, 2008. Article (CrossRef Link)

[2] P. Patel et al., “Service level agreement in cloud computing”, Cloud Workshops at OOPSLA09, 1: pp. 1-10, 2009. Article (CrossRef Link)

[3] R. Buyya et al., “Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility”, Future Generation Computer Systems, Vol. 25, pp.599-616, 2009. Article (CrossRef Link)

[4] M. Mitchell, “An Introduction to Genetic Algorithm”, the MIT Press, Cambridge, MA 1996. Article (CrossRef Link)

[5] Sung Ho Jang et al., “The Study of Genetic Algorithm-based Task Scheduling for Cloud computing”, International Journal of Control and Automation, Vol. 5, No.4, 2012. Article (CrossRef Link)

[6] Sourav Banerjee et al., “Advanced Task Scheduling for Cloud Service Provider Using Genetic Algorithm”, IOSR Journal of Engineering (IOSRJEN), Vol. 2, pp.141-147, July 2012. Article (CrossRef Link)

[7] Rich & knight, “Artificial Intelligence”, 3rd Edition, pp.445-455, 1998. Article (CrossRef Link)

[8] Tayal, S., “Tasks Scheduling Optimization for the Cloud Computing Systems”. International Journal of Advanced Engineering Sciences and Technologies (IJAEST), Vol. 5, pp.111-115, 2011. Article (CrossRef Link)

[9] Shi Chen, Jie Wu, Zhihui Lu, A Cloud Computing Resource Scheduling Policy Based on Genetic Algorithm with Multiple Fitness, IEEE 12th International Conference on Computer and Information Technology. Article (CrossRef Link) pp. 177-184, 2012.

[10] M. Armbrust et al., “A Berkeley View of Cloud computing”, Above the Clouds: Technical Report No. UCB/EECS-2009-28, University of California at Berkley, USA, Feb. 10, 2009. Article (CrossRef Link)

[11] Luqun Li, “An Optimistic Differentiated Service Job Scheduling System for Cloud Computing Service Users and Providers”, Third International Conference on Multimedia and Ubiquitous Engineering pp.295-299, 2009. Article (CrossRef Link)

[12] Francesco Maria Aymerich et al., “An Approach to a Cloud Computing Network”, IEEE Vol. 113, pp. 113-118, 2008. Article (CrossRef Link)

Dr. (Mrs) Kamlesh Dutta is the Associate Professor in Department of Computer Science Engineering, National Institute of Technology Hamirpur, Himachal Pardesh. Her area of research includes Artificial intelligence. She has more than 50 publications at her credit in the leading journals and conference, conducted many workshops and seminars. She is a life member of many professional organizations. Her area of interest is Artificial Intelligence. She is currently guiding research scholars at Ph.D levels.

Avtar Singh working as Assistant Professor in Department of Computer Science Engineering, Shoolini University, Solan Himachal Pardesh. He received the B.Tech and M.Tech degree in Computer Science Engineering from the Electro Technical University, Saint Petersburg Russia ("LETI") in 1999 and 2001 respectively. In 2001, he served IT industry in Bangalore and in 2006 joined academics served in leading educational institutions. He is authors of 4 research papers in the leading journals and conference, attended many workshops and seminars. His areas of interest are cloud computing, optimization, distributed computing, networking and Web Technologies. He is pursuing Ph.D in National Institute of Technology Hamirpur, Himachal Pardesh.