A multi objective Genetic Algorithm for cloud service reservation

Mr. Monilal S*
[1*]Lecturer, Dept. of computer science
Govt Polytechnic College, Ezhukone, Kollam, India
monilal.s@gmail.com

***Abstract*—Cloud is one of the emerging technology in computer industry. Several companies migrates to this technology due to reduction in maintenance cost . Several organization provides cloud service such as SaaS, IaaS, PaaS. Different organization provides same service with different service charges and waiting time. So customers can select services from these cloud providers according to their criteria like cost and waiting time. By using 'demand pricing' strategy, providers can provide services with minimum cost without loosing any income or valuable resource time. But the existing system doesnot provide any automated job scheduling considering consumer cost, provider benefit , consumer waiting and provider idle time. This paper propose a multi objective genetic algorithm for solving this multivariable optimization problem. This system provides a new cloud brokering mechanism with cloud service discovery using this optimization technique. This paper consider IaaS. In this system user submit a job to cloud. Cloud provides infrastructure to run this job and gave output to user. Here aim of user is to obtain output with minimum time and minimum cost. At the same time aim of provider is to increase the income. For that provide run more job with in unit time. So We have to minimize consumer cost, consumer waiting time and provider idle time , and maximize provider benefit .**

***IndexTerms*— Cloud, Adhoc Genetic algorithm, Iaas.**

I. Introduction

Cloud is one of the emerging technology in computer industry. Several companies migrates to this technology due to reduction in maintenance cost . Several organization provides cloud service such as SaaS, IaaS, PaaS. Different organization provides same service with different service charges and waiting time. So customers can select services from these cloud providers according to their criteria like cost and waiting time. By using 'demand pricing' strategy, providers can provide services with minimum cost without loosing any income or valuable resource time. But the existing system does not provide any automated job scheduling considering consumer cost, provider benefit , consumer waiting and provider idle time. This paper propose a multi objective genetic algorithm for solving this multivariable optimization problem. This system provides a new cloud brokering mechanism with cloud service discovery using this optimization technique.

In practice, customers have to pay close attentions to the randomness of the duration of the activities since they directly impact on the total cost of completing a job and the completion time of the job. Also, there usually exists a trade-off relation between the resource allocation and the duration of the activities. In this study , we propose a genetic algorithm (GA) as a decision support tool so as to optimally allocate the available resources to minimize the expected total cost (which include resource usage cost an d tardiness cost) for finishing the job. In order to evaluate the expected total cost, one needs to calculate the mean of the job execution time. Though time cost negotiation mechanism could be a simple and straightforward approach, its total utility function become impractical when the number of jobs in the cloud is large. Therefore, we propose a new mathematical model for calculation job waiting time , cost of execution, provider idle time.

II. Related Work

In 2008, A heuristic method to schedule bag-of-tasks (tasks with short execution time and no dependencies) in a cloud is presented in so that the number of virtual machines to execute all the tasks within the budget, is minimum and the same time speedup. In 2009, Marios D. Dikaiakos and George Pallis realized the concept of organization of Distributed Internet Computing as Public Utility and addressed the several significant problems and unexploited opportunities concerning the deployment, efficient operations and use of cloud computing infrastructures . In 2009, Dr. Sudha and Dr. Jayarani proposed the efficient Two-level scheduler (user centric meta-scheduler for selection of resources and system centric VM schedular for dispatching jobs) in cloud computing environment based on QoS. In 2010, Yujia Ge and Guiyi Wei proposed a new scheduler which makes the scheduling decision by evaluating the entire group of tasks in a job queue. A genetic algorithm is designed as the optimization method for a new scheduler who provides better makespan and better balanced load across all nodes than FIFO and delay scheduling . In 2010, An optimal scheduling policy based on linear programming, to outsource deadline constraint workloads in a hybrid cloud scenario is proposed in . In 2011, Sandeep Tayal proposed an algorithm based on Fuzzy-GA optimization which evaluates the entire group of tasks in a job queue on basis of prediction of execution time of tasks assigned to certain processors and makes the scheduling decision . In 2011, Laiping Zhao, Yizhi Ren & Kouichi Sakurai proposed a DRR (Deadline, Reliability, Resource-aware) scheduling algorithm, which schedules the tasks such that all the jobs can be completed before the deadline, ensuring the Realiability and minimization of resources .In 2011, S. Sindhu & Saswati Mukherjee proposed two algorithms for cloud computing environment and compared it with default policy of cloudsim toolkit while considering computational complexity of jobs. This paper provided us a framework for our investigation .

This paper "A Price- and-Time-Slot-Negotiation Mechanism for Cloud Service Reservations" provides an efficient cloud resource provisioning using cost and free slots in cloud. Three types of utility functions are used to model time cost behavior preferences. They are

1. Price utility function :- consumers prefer the cheapest price for leasing a service providers want to sell their services at the highest prices.

2. Time-Slot Utility Function: a novel time-slot utility function is designed to model consumers' and providers' preferences for different time slots. A consumer can have multiple sets of acceptable time-slot preferences. A provider's time-slot preferences are based on the following:

a) service demand

b) temporal ordering

c) fitting job size

3. Aggregated total utility function :- this is a combined value of time and price utility function.

But this paper doesn't consider provider benefit and consumer waiting time. These issues are considered for our work. Here We present a new resource provisioning mechanism using Genetic Algorithm.

But this paper doesn't consider provider benefit and consumer waiting time. These issues are considered for our work. Here We present a new resource provisioning mechanism using Genetic Algorithm.

Cloud is one of the emerging technology in computer industry. Several companies migrates to this technology due to reduction in maintenance cost . Several organization provides cloud service such as SaaS, IaaS, PaaS. Different organization provides same service with different service charges and waiting time. So customers can select services from these cloud providers according to their criteria like cost and waiting time. By using 'demand pricing' strategy, providers can provide services with minimum cost without loosing any income or valuable resource time. But the existing system doesnt provide any automated job scheduling considering consumer cost, provider benefit , consumer waiting and provider idle time. This paper propose a multi objective genetic algorithm for solving this multivariable optimization problem. This system provides a new cloud brokering mechanism with cloud service discovery using this optimization technique.

III. SYSTEM MODEL

A. *Mathematical Modeling of Your Objective*

Cloud is one of the emerging technology in computer industry. Several companies migrates to this technology due to reduction in maintenance cost . Several organization provides cloud service such as SaaS, IaaS, PaaS. Different organization provides same service with different service charges and waiting time. So customers can select services from these cloud providers according to their criteria like cost and waiting time. By using 'demand pricing' strategy, providers can provide services with minimum cost without loosing any income or valuable resource time. But the existing system doesn't provide any automated job scheduling considering consumer cost, provider benefit , consumer waiting and provider idle time. This paper propose a multi objective genetic algorithm for solving this multivariable optimization problem. This system provides a new cloud brokering mechanism with cloud service discovery using this optimization technique.

Here we propose a model for broker mechanism , that allocate jobs to different VM according to our criteria. For that we use genetic algorithm. By using this algorithm We generate different job scheduling sequence and select best sequence. Best sequence selection is based on a rank. This rank calculation is shown below.

$$R = \sum_{1}^{Nj} \frac{\frac{Wi}{Max(Wi)}}{Nj} + \sum_{1}^{Nvm} \frac{\frac{Max(Pi) - Pi}{Max(Pi)}}{Nvm} + \sum_{1}^{i=Nvm} \frac{\frac{Ti}{Max(Ti)}}{Nvm} + \sum_{1}^{Nj} \frac{\frac{Min(Ci) - Ci}{Max(ci)}}{Nj}$$

R- Rank

Nj – Number of jobs

Wi – Weighting time of ith Job

Pi – Profit of ith VM

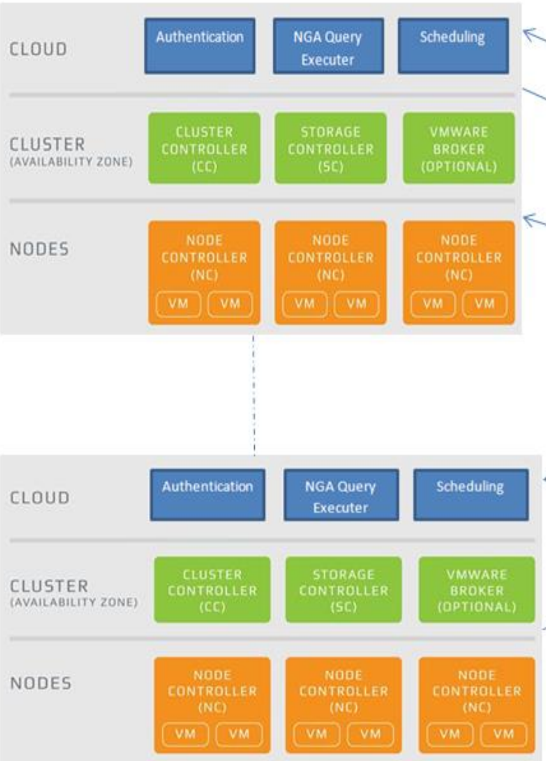
Nvm – Number of VM

Ti – Idle time ith VM

Ci- cost of ith Job

Here We select sequence with minimum rank. In this way, NGA generate new sequence. During this generation, NGA minimize rank. Here rank is the sum of waiting time, profit lose , consumer cost and cloud idle time. So these generations minimize above factors.

B. Architecture



This architecture is based on Eucalyptus cloud. We modify first layer of this architecture. In first layer we provider an authentication mechanism , NGA query executer and scheduler.

NGA query executer perform NGA algorithm based on user request and provides scheduling datas to user. Similarly same algorithm works on other clouds providing scheduling data. Based on these datas user select a choice and allocate job to that cloud using corresponding scheduler.

IV. METHODOLOGY

A. Genetic Algorithm

In a genetic algorithm, a population of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem is evolved toward better solutions. Each candidate solution has a set of properties (its chromosomes or genotype) which can be mutated and altered; traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible.

The evolution usually starts from a population of randomly generated individuals and is an iterative process, with the population in each iteration called a generation. In each generation, the fitness of every individual in the population is evaluated; the fitness is usually the value of the objective function in the optimization problem being solved. The more fit individuals are stochastically selected from the current population, and each individual's genome is modified (recombined and possibly randomly mutated) to form a new generation. The new generation of candidate solutions is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

A typical genetic algorithm requires:

1. a genetic representation of the solution domain,
2. a fitness function to evaluate the solution domain.

A standard representation of each candidate solution is as an array of bits. Arrays of other types and structures can be used in essentially the same way. The main property that makes these genetic representations convenient is that their parts are easily aligned due to their fi[redacted] facilitates simple crossover operations. Variab[redacted]esentations may also be used, but crossover [redacted]n is more complex in this case. Tree-like representations are explored in genetic programming and graph-form representations are explored in evolutionary programming; a mix of both linear chromosomes and trees is explored in gene expression programming.

Once the genetic representation and the fitness function are defined, a GA proceeds to initialize a population of solutions and then to improve it through repetitive application of the mutation, crossover, inversion and selection operators.

Initialization

Initially many individual solutions are (usually) randomly generated to form an initial population. The population size depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions. Traditionally, the population is generated randomly, allowing the entire range of possible solutions (the search space). Occasionally, the solutions may be "seeded" in areas where optimal solutions are likely to be found.

Selection

During each successive generation, a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through a fitness-based process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions. Other methods rate only a random sample of the population, as the former process may be very time-consuming.

The fitness function is defined over the genetic representation and measures the quality of the represented solution. The fitness function is always problem dependent. For instance, in the knapsack problem one wants to maximize the total value of objects that can be put in a knapsack of some fixed capacity. A representation of a solution might be an array of bits, where each bit represents a different object, and the value of the bit (0 or 1) represents whether or not the object is in the knapsack. Not every such representation is valid, as the size of objects may exceed the capacity of the knapsack. The fitness of the solution is the sum of values of all objects in the knapsack if the representation is valid, or 0 otherwise.

In some problems, it is hard or even impossible to define the fitness expression; in these cases, a simulation may be used to determine the fitness function value of a phenotype (e.g. computational fluid dynamics is used to determine the air resistance of a vehicle whose shape is encoded as the phenotype), or even interactive genetic algorithms are used.

Genetic operators

For each new solution to be produced, a pair of "parent" solutions is selected for breeding from the pool selected previously. By producing a "child" solution using the above methods of crossover and mutation, a new solution is created which typically shares many of the characteristics of its "parents". New parents are selected for each new child, and the process continues until a new population of solutions of appropriate size is generated. Although reproduction methods that are based on the use of two parents are more "biology inspired", some research suggests that more than two "parents" generate higher quality chromosomes.

These processes ultimately result in the next generation population of chromosomes that is different from the initial generation. Generally the average fitness will have increased by this procedure for the population, since only the best organisms from the first generation are selected for breeding, along with a small proportion of less fit solutions, for reasons already mentioned above.

It is worth tuning parameters such as the mutation probability, crossover probability and population size to find reasonable settings for the problem class being worked on. A very small mutation rate may lead to genetic drift . A recombination rate that is too high may lead to premature convergence of the genetic algorithm. A mutation rate that is too high may lead to loss of good solutions unless there is elitist selection. There are theoretical but not yet practical upper and lower bounds for these parameters that can help guide selection.

Termination

This generational process is repeated until a termination condition has been reached. Common terminating conditions are:

- A solution is found that satisfies minimum criteria
- Fixed number of generations reached
- Allocated budget (computation time/money) reached
- The highest ranking solution's fitness is reaching or has reached a plateau such that successive iterations no longer produce better results
- Manual inspection
- Combinations of the above

B. New Genetic Algorithm for Cloud Resource Provisioning

Initial Sequence -Here we generate n chromosomes randomly. Here We generate 4 sequences

Sequence	Job1	Job2	Job3	Job4	Job5	Job6	weight
1	2	4	1	4	4	4	109201
2	2	2	4	4	1	4	121743
3	2	2	3	1	3	1	106050
4	4	3	2	2	1	3	126608

Selection process-in this process we generate 2n sequences. for that we first copy n sequences from initial sequences and generate n remaining sequences as randomly select two sequences from initial sequences and select sequence with best fitness value. Repeat this n times then we get 2n sequences

Generate 8 Sequences by copying above 4 and Generate other 4 as follows

Generate two random number r1, r2 if rank of r1 < r2 select r1 otherwise select r2, this is based on another random number r3 whose value <.75.

For example r1=4 and r2=3 then select r2 because rank of r2 < r1 and set as sequence 5

Sequence	Job1	Job2	Job3	Job4	Job5	Job6	weight
1	2	4	1	4	4	4	109201
2	2	2	4	4	1	4	121743
3	2	2	3	1	3	1	106050
4	4	3	2	2	1	3	126608
5	2	2	3	1	3	1	106050
6	2	2	3	1	3	1	106050
7	2	4	1	4	4	4	109201
8	2	4	1	4	4	4	109201

Cross Over- here we generate 2n sequences by copying first n sequences from selection sequences. Remaining n sequences is generated as follows:

Randomly select two sequences from selection sequences and a new sequence is generated by combining genes from these selected sequences.

Copy first 4 sequence from cloning, next 4 sequences are generated as. Select three random numbers r1,r2,r3. New sequence are generated from r1 ,r2 by copying first r3 elements of r1 and balance job- r3 elements from r2 starting from r3

For example in the case of sequence 7, r1=7, r2=3 , r3=5 ,. Copy first 5 elements of sequence 7 in Table2 and last two elements of sequence 3.

Sequence	Job1	Job2	Job3	Job4	Job5	Job6	weight
1	2	4	1	4	4	4	109201
2	2	2	4	4	1	4	121743
3	2	2	3	1	3	1	106050
4	4	3	2	2	1	3	126608
5	2	2	3	1	3	1	106050
6	2	4	1	4	4	4	109201
7	2	4	1	4	3	1	103782
8	2	2	3	4	4	4	108956

mutation- here we generate 2n sequences by copying first n sequences from Cross over sequences. Remaining n sequences is generated as randomly select a sequence from above and perform some changes in sequence

copy first 4 sequnece, next 4 sequnce generated by r1,r2, r3. Select r1 sequnce, interchange r2 & r3 elements of r1

For example in the case of sequence 8, r1=3, r2=3 , r3=4

Sequence	Job1	Job2	Job3	Job4	Job5	Job6	weight
1	2	4	1	4	4	4	109201
2	2	2	4	4	1	4	121743
3	2	2	3	1	3	1	106050
4	4	3	2	2	1	3	126608
5	2	2	3	1	3	1	106050
6	2	2	3	1	3	1	106050
7	2	4	1	3	4	1	103782
8	2	1	3	2	3	1	106050

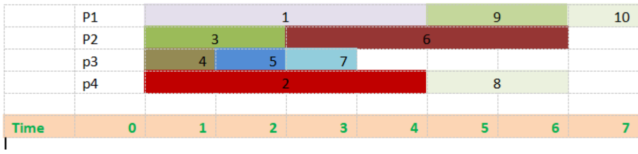
Select Best

Sort these 8 sequences on rank base , select first 4 sequences with min weight

Sequence	Job1	Job2	Job3	Job4	Job5	Job6	weight
1	2	4	1	3	4	1	103782
2	2	2	3	1	3	1	106050
3	2	2	3	1	3	1	106050
4	2	2	3	1	3	1	106050

How Weight Calculated

Consider Following Allocation



Waiting Time Of Job 9: 6

Here highest waiting time 7 (job 10)

PW (Percentage of waiting time)= Sum(waiting time for each job) /(heighst waiting time * num of jobs)

idle time of p4=1, Highest idle time=4 (p3)

PI (Percentage of idle time)= Sum(idle time for each node) /(heighst idle * num of nodes)

Suppose p3 is considered as costly node with 10 Rs/ sec , But Job j2 is allocated to p4 (have 5 Rs/Sec) and joblength=4 sec, Income= 4*5=20. Suppose this job run in p3 , income=4*10=40 . So profit lose=40-20=20

Max Benefit is obtained when all jobs are allocated to p3

PP (percentage of profit)=Sum(profit lose of each node) /(Maximum benefit * num nodes)

Similarly p1 is considered as low cost node (1Rs/ sec), But job j2 allocated to p4 (5Rs/Sec) and job length=4 , So cost=4*5=20. But it is run on p1 then cost=4*1=4 . So increase in cost= 20-4=16. Min Consumer cost obtained when all jobs executed on p1

PC (percentage of cost)=Sum(increase in cost for each job) /(Maximum benefit * num jobs)

Rank= PC + PP + PW + PI.

IV. Performance Evaluation

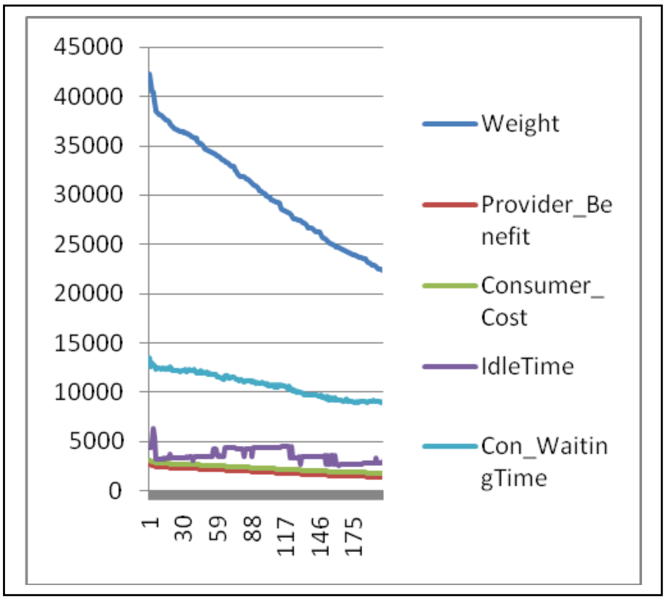
Developed a java based genetic algorithm. we got a scheduling details within 0.5 micro seconds for following inputs.

No : of jobs	2000
No: of nodes	30
No: of GA rounds	100
No : of chromosomes	20

From above data it is clear that this algorithm does not badly affect performance of brokering mechanism in cloud. after a no of rounds of various inputs we can realize that as no: of rounds increases we got better scheduling data. i.e Scheduling job with minimum waiting time, Consumer cost, provider idle time and maximum provider benefit. These results are shown in following table.

Iteration	Weight	Benefit Lose	Consumer_Cost	IdleTime	WaitingTime
1	42272	2680	3111	4404	13532
20	37036	2336	2769	3334	12478
30	36384	2293	2724	3387	12239
40	35770	2253	2679	3415	12095
50	34578	2174	2598	3480	12061
60	33960	2134	2555	3444	11579
70	33011	2070	2489	4363	11488
80	31916	1998	2415	4255	11325
90	31144	1947	2363	4303	11156
100	30102	1878	2295	4360	10835
110	29239	1821	2239	4402	10543
120	28258	1756	2170	4470	10619
130	27423	1704	2117	2722	9964
140	26612	1649	2062	3453	9733
150	25692	1588	2000	3505	9609
160	24827	1531	1938	3438	9452
170	24254	1495	1903	2716	9133
180	23691	1458	1863	2739	9092
190	23096	1418	1825	2774	9096
200	22382	1370	1779	2928	9087

This performance diagram is shown below.



Performance diagram for different parameters against Number of iteration

V. Conclusion

In existing cloud system job allocation does not consider consumer preferences like waiting time and cost. But our proposed NGA algorithm provides better negotiation between consumer and provider. this algorithm help us to provide an efficient resource provisioning with minimum waiting time, Consumer cost, provider idle time and maximum provider benefit. this scheduling algorithm does not delay scheduling. now we are not considering task classification and resource classification. this will be considered in our future work.

In IAAS, user submit a task to cloud. cloud perform this task. several cloud provides same services. user can select any of these clouds. these selection is based on criteria such as 1.waiting time 2.cost. At the same time provider attract consumer by reducing cost. For that provider use on demand pricing method. by using this method provider done more tasks per unit time. this will reduce provider idle time and increase profit. In the same time provider can reduce service cost due to high demand, but existing scheduling algorithm does not consider these factors. here we propose a new genetic algorithm based resource provisioning system with minimum waiting time, Consumer cost, provider idle time and maximum provider benefit.

References

[1] Ze Li, Student Member, IEEE, and Haiying Shen, Member, IEEE.” A QoS-Oriented Distributed Routing Protocol for Hybrid Wireless Networks”. IEEE transactions on mobile computing, vol. 13, no. 3, March 2014