Fall 2011

A Genetic Algorithm Scheduling Approach for Virtual Machine Resources in a Cloud Computing Environment

Shailesh Sawant

A Genetic Algorithm Scheduling Approach for Virtual Machine Resources in a Cloud Computing Environment

A Writing Project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Shailesh Sawant

Fall 2011

SAN JOSÉ STATE UNIVERSITY

The Undersigned Project Committee Approves the Project Titled

A Genetic Algorithm Scheduling Approach for Virtual Machine Resources in Cloud Computing Environment

by

Shailesh Sawant

Approved for the Department of Computer Science

Dr. Robert Chun	Department of Computer Science	Date
Dr. Soon Tee Teoh	Department of Computer Science	Date
Dr. Mark Stamp	Department of Computer Science	Date
Mr. Guru Parab	HP-Hawlett-Packard	Date

Approved for the University

Associate Dean Office of Graduate Studies and Research Date

A Genetic Algorithm Scheduling Approach for Virtual Machine Resources in a Cloud Computing Environment

by Shailesh Sawant

ABSTRACT

In the present cloud computing environment, the scheduling approaches for VM (Virtual Machine) resources only focus on the current state of the entire system. Most often they fail to consider the system variation and historical behavioral data which causes system load imbalance.

To present a better approach for solving the problem of VM resource scheduling in a cloud computing environment, this project demonstrates a *genetic algorithm* based VM resource scheduling strategy that focuses on system load balancing.

The genetic algorithm approach computes the impact in advance, that it will have on the system after the new VM resource is deployed in the system, by utilizing historical data and current state of the system. It then picks up the solution, which will have the least effect on the system. By doing this it ensures the better load balancing and reduces the number of dynamic VM migrations.

The approach presented in this project solves *the problem of load imbalance and high migration costs*. Usually load imbalance and high number of VM migrations occur if the scheduling is performed using the traditional algorithms.

ACKNOWLEDGEMENTS

I am grateful to my project advisor, Dr. Robert Chun, whose direction, support, encouragement, and dedication is valuable. Dr. Chun is an educationalist in the truest sense. I extremely appreciate Dr. Soon Tee Teoh's, Dr. Mark Stamp's and Mr. Guru Parab's input as my thesis committee members. My committee has shown helpful insight to polish my work which is presented in this report. The four of you are valuable assets to the computer world.

It has been a tricky, yet meaningful ride which I could not have completed without help and I am grateful for your support.

Thank you.

Shailesh Sawant

TABLE OF CONTENTS

1.0 INTRODUCTION……..1

1.1 CLOUD COMPUTING BACKGROUND…….....1

1.2 VIRTUALIZATION TECHNOLOGY…….....1

1.3. PROBLEM STATEMENT……..….2

2.0. RELATED WORK …….....3

2.1. NAÏVE STRATEGY……4

2.2. FIFO STRATEGY……4

2.3. OPTIMIZED STRATEGY ……..5

2.4. REQUIREMENT OF GENETIC ALGORITHM ……...6

3.0. THE MODEL DESIGN OF VM SCHEDULING ……6

3.1. VM MODEL……6

3.2. THE EXPRESSION OF LOAD……..7

3.3. MATHEMATICAL MODEL…….. ……..9

4.0. REALIZATION OF BALANCED SCHEDULING THROUGH GENETIC ALGORITHM……...10

4.1. STEPS IN GENETIC ALGORITHM……..10

4.2. POPULATION CODING ……...11

4.3 INITIALIZATION OF POPULATION ……12

4.4. FITNESS FUNCTION……12

4.5. SELECTION STRATEGY……..13

4.6. CROSSOVER OPERATION ……14

4.7. MUTATION OPERATION ……..15

4.8. SCHEDULING STRATEGY……16

5.0. ALGORITHM ANALYSIS ……...16

5.1. GLOBAL SCHEDULING ALGORITHM……..16

5.2. EXPERIMENT AND ANALYSIS OF GENETIC ALGORITHM……18

6.0. EXPERIMENT ……..20

6.1. HARDWARE SETUP ……20

6.2 SOFTWARE COMPONENTS……21

6.2.1. OPENNEBULA……21

6.2.2. HAIZEA ……24

6.2.3. ADVANCE RESERVATION……25

6.2.4. BEST-EFFORT LEASE……27

6.3. ANALYSIS ……...28

6.3.1. ALGORITHM EFFECT ANALYSIS ……....28

6.3.2. MIGRATION COST ANALYSIS ……..30

7. CONCLUSION ……..32

8. FUTURE WORK……32

9. REFRENCES ……34

LIST OF TABLES

TABLE 1 VM LOAD DATA ……18

LIST OF EQUATIONS

EQUATION 1. AVERAGE LOAD OF VM ON P IN TIME T 7

EQUATION 2. AVERAGE LOAD ON P IN TIME T 8

EQUATION 3. LOAD ON P IN TIME T WHEN VM V IS ARRANGED WITH P …….....8

EQUATION 4. LOAD DEVIATION OF THE SOLUTION SI 8

EQUATION 5. VM MIGRATION COST 9

EQUATION 6. FITNESS FUNCTION …….....13

EQUATION 7. FITNESS RATIO BASED SELECTION ALGORITHM.....14

EQUATION 8. MUTATION PROBABILITY 15

LIST OF FIGURES

FIGURE 1 SYSTEM STRUCTURE…….7

FIGURE 2 TREE STRUCTURE ENCODING IN GENETIC ALGORITHM……11

FIGURE 3 CROSSOVER OPERATION…….....14

FIGURE 4 MAPPING RELATIONSHIPS BEFORE USING ALGORITHM…….19

FIGURE 5 MAPPING RELATIONSHIPS AFTER USING ALGORITHM …….....19

FIGURE 6 CLOUD ECOSYSTEM…….....21

FIGURE 7 OPENNEBULA ARCHITECTURE…….22

FIGURE 8 COMPARISONS OF THREE SCHEDULING POLICIES WHEN SYSTEM LOAD IS STABLE…...29

FIGURE 9 COMPARISONS OF THREE SCHEDULING POLICIES WHEN SYSTEM LOAD IS VARIANT…29

FIGURE 10 NUMBER OF VM MIGRATIONS WITH STABLE SYSTEM LOAD……31

FIGURE 11 NUMBER OF VM MIGRATIONS WHEN THE SYSTEM LOAD IS MORE ……31

1.0.INTRODUCTION

This section addresses the benefit of VM load balancing with consideration of historical data and the current system state. The problem addressed by this project is discussed in section 1.3.

1.1.CLOUD COMPUTING BACKGROUND

Cloud computing is a new technology currently being studied in the academic world [1]. The definition of the cloud computing from the Gartner: "A style of computing where massively scalable IT-related capabilities are provided as a service across the internet to multiple external customers using internet technologies [2]."

The cloud computing platform guarantees subscribers that it sticks to the service level agreement (SLA) by providing resources as service and by needs. However, day by day subscribers' needs are increasing for computing resources and their needs have dynamic heterogeneity and platform irrelevance. But in the cloud computing environment, resources are shared and if they are not properly distributed then it will result into resource wastage. Another essential role of cloud computing platform is to dynamically balance the load amongst different servers in order to avoid hotspots and improve resource utilization. Therefore, the main problems to be solved are how to meet the needs of the subscribers and how to dynamically as well as efficiently manage the resources.

A layer in cloud computing is (IaaS) "Infrastructure-as-a-Service," for example with amazons' EC2, the cluster of virtual machines are deployed on the cloud providers' data-center.

1.2.VIRTUALIZATION TECHNOLOGY

On a cloud computing platform, dynamic resources can be effectively managed using virtualization technology. The subscribers with more demanding SLA can be guaranteed by accommodating all the required services within a virtual machine image and then mapping it on a physical server. This helps to solve problem of heterogeneity of resources and platform

irrelevance. Load balancing of the entire system can be handled dynamically by using virtualization technology where it becomes possible to remap virtual machines (VMs) and physical resources according to the change in load [3]. Due to these advantages, virtualization technology is being comprehensively implemented in cloud computing. However, in order to achieve the best performance, the virtual machines have to fully utilize its services and resources by adapting to the cloud computing environment dynamically. The load balancing and proper allocation of resources must be guaranteed in order to improve resource utility [4]. Thus, the important objectives of this research are to determine how to improve resource utility, how to schedule the resources and how to achieve effective load balance in a cloud computing environment.

1.3.PROBLEM STATEMENT

In the current cloud computing environments, VM resource scheduling only considers the current system condition and ignores the previous state of system which causes the system load imbalance. Number of VM migrations is more when most of the load balancing takes place [5]. The entire migration cost becomes a problem when all the VM resources are migrated. This is largely due to granularity of VM resources and the large amount of data transferred in the migration with suspension of VM service. This project provides a scheduling strategy to enable effective load balancing. The method used in this project will compute its influence on the system in advance, when current VM resources are allocated to every physical node and will opt for the deployment that will have the least load on the system. This is achieved using genetic algorithm, historical data and the current state of the system.

The current VM resource scheduling is explained in the Related Work, section 2.0. The design of VM scheduling model is explained in VM model, section 3.0. The genetic algorithm,

section 4.0, explains the genetic algorithm approach for the VM scheduling in a cloud computing environment. This report concludes with an experiment that analyzes the method and discusses possible future scopes.

2.0.RELATED WORK

The main objective of load balancing has always been to efficiently and fairly distribute every computing resource and improve resource utilization. Many researchers in the past have proposed different scheduling algorithms like static, dynamic and mixed scheduling strategies [6]. ISH [7], MCP [8] and ETF [9] are some of the static algorithms based on BNP which are suitable for high internet speed, ignorable communication delay and small distributed environments; while DSL [10] and MH [11] algorithms are based on APN that consider delays in communication and execution time. These approaches are suitable for largely distributed environments. Some algorithms in dynamic scheduling algorithms achieve effective load sharing and load balancing in task distribution by intelligent distribution and self-adapting distribution. Mixed scheduling algorithms' main goal is to provide equal distribution of assigned computing task and reduce the communication cost of all the distributed computing nodes. It also balances scheduling by computing the volume of every node. There have been other algorithms studied by researchers like autonomic scheduling, central scheduling, intelligent scheduling, and agent negotiated scheduling.

There are many differences and similarities between the scheduling of VM resources in the cloud computing environment and the traditional scheduling algorithm. Target of scheduling is the biggest difference between cloud computing and traditional computing environments. In the cloud computing environment, VM resources are the scheduled targets so the granularity and

transferred data is large; whereas in the traditional computing environment the target of scheduling is a process or task, so the granularity and transfer of data is small. Secondly, compared with the deployment time of the VMs, the time of scheduling algorithms can almost be neglected in a cloud computing environment. This project tries to make equal distribution of hardware resources of VMs in cloud computing environment so that the VM can improve its running efficiency while meeting the quality of service needs of the subscribers.

The resources can be allocated on demand when the application is in execution, but deployment of the resources takes time which directly impacts the performance of the application.

Below are the 3 strategies for allocation of the virtual resources:

1. Naïve Strategy
2. FIFO Strategy
3. Optimized strategy

2.1. NAÏVE STRATEGY

If we have *m* number of services in an application workflow and *t* is the execution time for each service. A set of *x* number of virtual computing resources may be allocated. The network bandwidth is also allocated based on the proportionality of the data transfers between two services [12]. This strategy considers only the single execution. It serves as a baseline for the performance.

2.2. FIFO STRATEGY

Assumption is made that on every computing resource, all the services can be deployed. The scheduler may ask any resource to compute any task. Due to infrastructure, redeployment of services is not necessary. It is considered as an optimal strategy. In this case, the same bandwidth

is allocated to all the links in the infrastructure. Due to data transfer time, when the bandwidth is small, the total cost is high [12]; but when the bandwidth increases, then both cost and execution time decrease. The optimization is used to approximate the optimal bandwidth. This strategy only works for the identical resources and bandwidth is not optimized between each pair of resources.

2.3. OPTIMIZED STRATEGY

The optimized strategy divides the execution of the workflow into multiple stages; and at each stage, bandwidth and resources are allocated independently. And also in every stage, the minimization algorithm is executed to allocate the optimum number of resources for the services involved in the current stage. An algorithm is required to determine the number of stages.

Directed execution graphs (DAG) are made up of the workflow of services. The DAG is divided into the execution stages [12]. A special virtual infrastructure executes the execution stages. In every execution stage, the infrastructure is reconfigured for deployment for services involved in the stage. The resources are allocated based on the number of invocations required for each service.

At present, many research studies on balanced VM resources scheduling are based upon dynamic migration of VMs. Sandpiper systems carry out hotspot probing and dynamic monitoring on the utility of system's memory resources, CPU and network bandwidth [13]. It also provides black box and white-box resource monitoring methods. The system's main focus is to determine how to dispose hotspots through the remapping of resources in VM migration and how to define hotspot memory. VMware Distributed Resource Scheduler (DRS) is a tool that balances and distributes computing volume by using available resources in a virtualized environment [14]. Thus, using dynamic migration all of the above systems can achieve system

load balance; but frequent dynamic migration would employ a large number of resources that might lead to degrading the entire system performance.

2.4.REQUIREMENT OF GENETIC ALGORITHM

Genetic algorithm [15] is a random searching method that has a better optimization ability and internal implicit parallelism. It can obtain, and instruct the optimized searching space and adjust the searching direction automatically through the optimization method of probability. With the advantages of genetic algorithm, this project presents a balanced scheduling strategy of VM resources in cloud computing environment [16, 17]. By considering the current states and historical data, this method will compute in advance its influence over the entire system.

3.0.THE MODEL DESIGN OF VM SCHEDULING

3.1.VM MODEL

The relationship between the physical machines and the VMs can be seen from figure 1. Consider *P* as a set of all the physical machines in the entire system, where P = {P_1, P_2, P_3 …. P_N }. *N* is total number of the physical machines and an individual physical machine can be denoted as P_i, where *i* denote the physical machine number and range of *i* is (1 <= *i* <= N). Similarly, we have a set of VMs on each physical machine P_i, V_i = {V_{i1}, V_{i2}, V_{im}} here m is the number of VMs on the physical server *i* [18]. If we want to deploy VM *V* on the present system, then we have a solution set denoted by *S* = {S_1, S_2, S_3 …. S_N}, it represents the mapping solution after VM *V* is assigned to each of the physical machines. When the *V* is arranged with the physical machine P_i we get the mapping structure denoted as S_i.

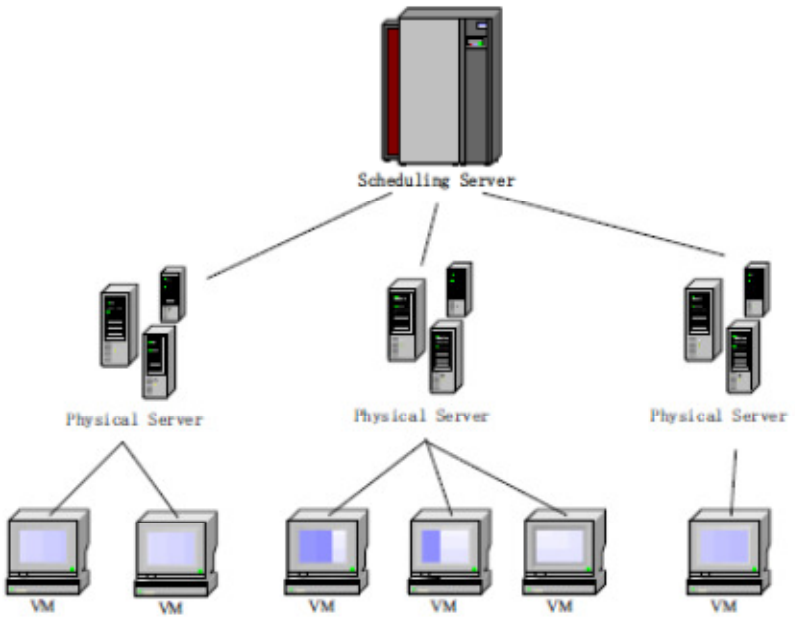

Figure 1. System Structure [18].

3.2.THE EXPRESSION OF LOAD

The summation of all the running VMs on a physical machine can be termed as the load of the physical machine. If we assume that the *T* is the best time duration to monitor historical data i.e., from current time to the last *T* minutes will be the historical data zone, which will be used to solve the load balancing problem.

By using the variation law of the physical machine load, we can split *T* into n subsequent time intervals. Therefore, *T* becomes $[(t_1 - t_0), (t_2 - t_1),\ldots,(t_n - t_{n-1})]$.

The time interval *k* can be defined as $(t_k - t_{k-1})$. If the load of a VM is stable in all the periods then *V(i, k)* refers to the load of VM *i* in the interval *k*.

By using the above definition, we can define the average load of VM on physical server P_i in time cycle *T* is $$\overline{Vı\,(ı,T)} = \frac{1}{T}\sum_{k=1}^{n} V(i,k) \times \left(t_k - t_{k-1}\right)$$

Equation 1. Average load of VM on P_i in time T.

By using the above average load definition of VM, we can calculate the load of the physical machine for last *T* interval by adding all the loads of the VMs present on the physical machine. So, the expression to compute the load of physical machine P_i is as follows:

$$P(i,T) = \sum_{j=1}^{mi} \overline{Vi(j,T)}$$

Equation 2. Average load on P_i in time T.

The virtual machine *V* needs to be deployed on the current system. We have the resource information for VM *V*, and from that we can estimate the load of the VM as *V'*. Therefore, when the VM *V* is joined to every physical server, we can calculate the load of each P_i as follows:

$$P(i,T)' = \begin{cases} P(i,T) + V' & After\ Deploy\ V \\ P(i,T) & Others \end{cases}$$

Equation 3. Load on P_i in time T when VM V is arranged with P_i.

Normally, when the VM *V* is set with P_i, there is a possibility of definite alteration in the system load. Thus, we need to compute a load adjustment factor to attain load balancing. After VM *V* is arranged to the physical machine P_i by using above equation, we can compute the load deviation $\sigma_i\ (T)$ of the solution S_i [18].

$$\sigma_i\ (T) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\overline{P(T)'} - P(i,T)')^2}$$

$$Where\ P(T)' = \frac{1}{N} \sum_{i=1}^{N} P(i,T)'$$

Equation 4. Load deviation of the solution S_i.

3.3.MATHEMATICAL MODEL

By using the previous analysis, the mathematical model can be defined as follows,

3.3.1. **Definition 1**

When the system mapping solution is S_i then to calculate the total load variation σ_i *(Si, T)* i.e., mean square deviation in time interval *T* with the load of each physical machine is *P(i, T)'* is defined in equation 4 as follows,

$$\sigma_i \text{ (Si, T)} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\overline{P(T)'} - P(i,T)')\text{^}2}$$

$$\text{Where} \quad \mathrm{P(T)}' = \frac{1}{\mathrm{N}}\sum_{\mathrm{i=1}}^{\mathrm{N}} \mathrm{P(i,T)}'$$

3.3.2. **Definition 2**

The final balanced mapping solution for the S_i is S_i*'* then *S* should refer to $\mathrm{S} = \{\mathrm{S}'_1, \mathrm{S}'_2, \mathrm{S}'_3 \ldots \mathrm{S}'_N\}$. S'_i is the top mapping solution to construct σ_i *(Si, T)* which satisfies the load constraints.

3.3.3. **Definition 3**

To calculate the migration cost, we use the ratio of VM number *M'* which needs to migrate for achieving load balancing in a specific solution to the total VM number as a cost divisor [18]. Then for each solution S_i the cost divisor $\rho(S_i)$ to attain balancing S'_i is as follows:

$$\rho(S_i) = \frac{M'}{M}$$

Equation 5. VM migration cost.

The main goal of this project is to achieve best system load balancing with the minimum migration cost by finding the optimal mapping solution S_i and also reducing the cost divisor $\rho(S_i)$ in the overall load balancing. We can get the best mapping result S'_i from mapping result S_i using the genetic algorithm approach.

4. REALIZATION OF BALANCED SCHEDULING THROUGH THE GENETIC ALGORITHM

Developed from the evolution law in the ecological world, the Genetic algorithm is a random searching method. After the first population is generated, it evolves better and better approximate solutions using the law of survival of the fittest from the generations. An individual is chosen in every generation based on the fitness of different individuals in certain problem domains. A new population representing a new solution set is produced when different individuals combine, cross, and vary by genetic operators in natural genetics. This project presents a scheduling strategy through the genetic algorithm based on the real situation of cloud computing.

4.1.STEPS IN THE GENETIC ALGORITHM

1. [**Start**] produce random population of n chromosomes (coding structure can be selected according to the problem domain) [19].
2. [**Fitness**] calculate the fitness value f(x) of every chromosome in the given population.
3. [**New population**] generate the new population by reiterating the following steps till the creation of new population is done.

3.1.[**Selection**] select two parent individuals from the population according to the fitness value.

3.2.[**Crossover**] by using the crossover probability, generate the new offspring by reforming the parents.

3.3.[**Mutation**] with the probability of mutation, mutate the new child at some positions.

3.4.[**Accepting**] now the new offspring the part of next generation of population [19].

4. [**Replace**] use the new generation as the current generation.

5. [**Test**] if the stopping condition is satisfied then end the algorithm and return the individual with the highest fitness value.
6. [**Loop**] goto step 2.

*4.2.*POPULATION CODING

To solve the problems using genetic algorithm, it is not to function on the result set but to create a specific coding denotations. We first need to do the coding to tackle the problem. Based on the design of genetic operators and the properties of the problem, we select our coding method. A classic genetic algorithm is a chromosome structure of genes composed of binary codes. The data model in this project has a one-to-many mapping relationship between VMs and physical machines. The project uses tree structure to mark the chromosome of genes [15]. This says that every mapping solution is marked as one tree; the root nodes on the first level are the managing and scheduling node of the system, while the second level stands for all of the N nodes of physical machines and third level stands for M nodes of VMs on certain physical machines. The main usage of tree encoding in the genetic algorithm is in the growing programs or expressions. In this encoding scheme, each chromosome is a tree structure of some objects as functions or commands e.g., delete leaf, search X.

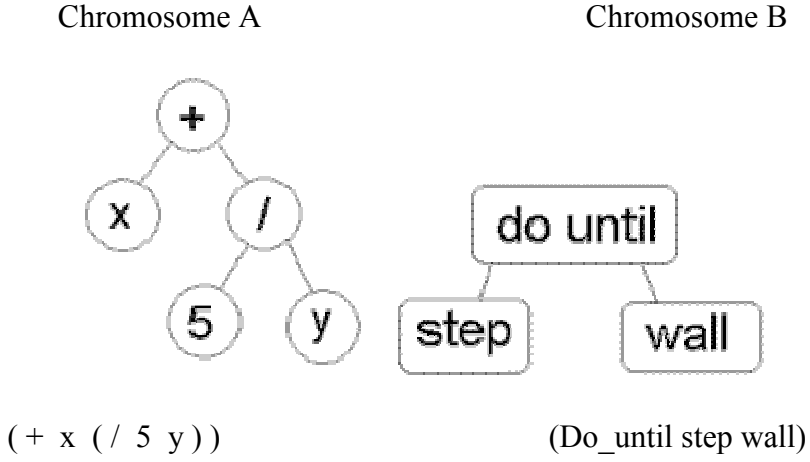

Figure 2 Tree structure encoding the genetic algorithm [15].

4.3.INITIALIZATION OF POPULATION

To initialize the population, we use the spanning tree method in this project. The tree will thus be defined as follows:

→This spanning tree is constructed using VM set and the elements in the physical machine set.

→ Predefined management and scheduling source node is the root node of this tree.

→ The tree also contains all of the VM nodes and physical machine nodes.

→ All the leaf nodes are the VM nodes.

The principle of the spanning tree method:

Through inheritance, spanning tree method should produce relatively superior descendants or it should meet the given load balancing conditions. This means the tree should itself be a comparatively superior individual. Thus the mapping relationship between VMs and the physical machines can be achieved by the following procedure. First, the selection probability *p* is computed for every VM from VM set. Probability *p* is the ratio of sum of single VM load and sum of all the VM loads). Then, based on probability *p*, we allocate all of the logical disks to the node, which has the smallest load in the physical machine set to produce the leaf node of the initial spanning tree. When the load of VM is high it generates more heat in the system. In this way, we increase the possibility of selecting VMs with high heat because; VMs with High heat can be allocated first on different physical machine to avoid a hotspot. Hotspot occurs when two or more computing intensive VMs are allocated on same physical machine.

4.4.FITNESS FUNCTION

In an ordinary world, the productivity of any individual depends on the fitness value which shows the number of decedents it will have. The fitness function is the measure for the superiority of an individual in the entire population. The fitness value shows the performance of

an individual. If the fitness is large, then the performance of an individual is better. Depending on the fitness function value, the individuals are determined to survive or die out. Hence, the fitness function is the motivating factor in the genetic algorithm. Following equation no. 6 is the fitness function used in this project.

$$f(S,T) = \frac{1}{A + B \times f_H}$$

$$f_H = \Phi(\sigma_i(S,T) - \sigma_0), \Phi(X) = \begin{cases} 1, & X \leq 0 \\ r, & X > 0 \end{cases}$$

Equation 6. Fitness Function.

The weighted coefficients *A* & *B* are predefined in the real applications.

σ_0 = a predefined heat variation constraints which is allowed in system load balancing.

$\Phi(X)$= penalty function value = 1 if individual meets the constraints otherwise *r* which is predefined in real application.

4.5.SELECTION STRATEGY

Selection mechanism is used to select an intermediate solution for the next generation based on the survival of the fittest law. This operation is the guiding channel for the genetic algorithm in terms of the performance. There are different selection strategies to select the best chromosomes e.g. roulette wheel, Boltzman strategy, tournament selection, rank based selection, etc. In this project, the *fitness ratio based selection algorithm* is used.

From the current population, first we calculate the fitness value of each individual and then keep the highest fitness value individual in the next generation. After that, we calculate the selection probability of every individual based on the fitness ratio as shown below.

$$p_i = \frac{fi(S,T)}{\sum_{i=1}^{D} fi(S,T)}$$

Equation 7. Fitness ratio based selection algorithm.

$f_i(S, T)$: for the fitness value of individual No. *i* in the population

D: scale of population.

After calculating the probability, we elect the individuals by performing the rotating strategy; because there will be a high probability for the selection of the high fitness value individual. In addition, those individuals who have a low fitness value are likely to be considered for selection.

4.6.CROSSOVER OPERATION

Crossover/hybridizing operation can be achieved by selecting two parent individuals and then creating a new individual tree by alternating and reforming the parts of those parents. Hybridization operation is a guiding process in the genetic algorithm and it boosts the searching mechanism. Since we are using tree coding in this project, in order to guarantee the legality of the chromosome of the new children, the hybridization does not work as to the binary encoding in which it exchanges the some parts of the genes. This project replicates the crossover process family tree to ensure that the child takes the same gene from the parent chromosome and also ensures the legitimacy of the tree leaf nodes. The crossover mechanism is explained as below;

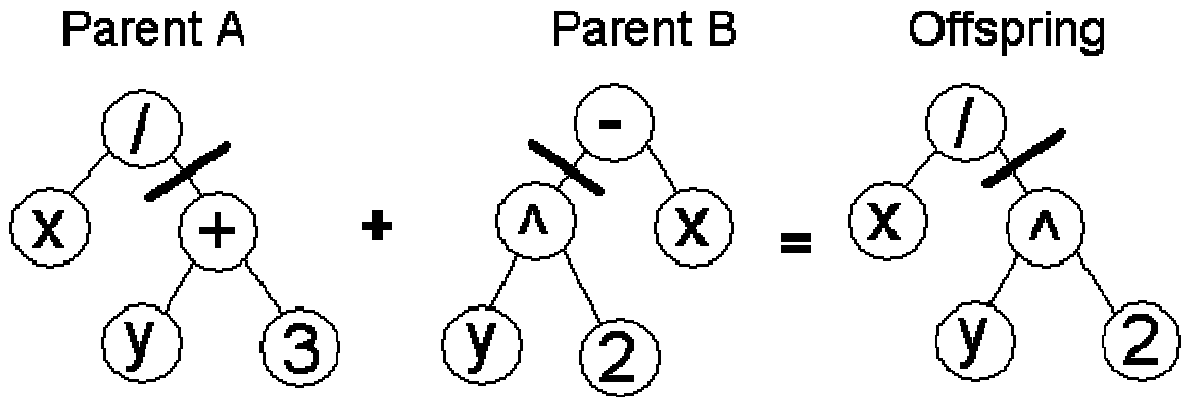

Figure 3 Crossover operations [15].

- Choose two parents *P1* and *P2* with the help of rotating selection algorithm.
- Join the two parents to structure a new child tree *P0,* which retains the nodes with the same leaf nodes in the two parents and discards the different ones.
- For the nodes which are discarded in the previous step, calculate their probability based on the load of every VM then based on probability distribute them on the least loaded physical machine set until the distribution process is completed.
- Repeat the above procedure until it reaches the desired crossover number.
- In both parent one crossover spot is chosen, parents are separated in that spot and swap part below intersect point to construct new children.

4.7.MUTATION OPERATION

To get the larger variation, variation operators at the foundation of the genetic algorithm preserve the diversity of population and keep away from prematurity. The variation operator is reduced to guarantee the local searching capability when the genetic algorithm gets close to the best result. This project uses the following self-adaptive variation probability.

$$P_m = exp\frac{(-1.5 \times 0.5t)}{D} \times \sqrt{M}$$

Equation 8. Mutation probability.

T = number of generations

D = scale of population

M = number of VMs

The individuals are arbitrarily selected to differ according to the deviation probability. To avoid the repetition of the same gene on the same chromosome, when the gene on one specific

location on the chromosome carries this chromosome, the leaf node must be different after variation.

4.8.SCHEDULING STRATEGY

The main aim of this project is to find the best mapping solution which meets the predefined system load constraint to the maximum level, or to make the migration cost for the load balancing as low as low possible. By using the genetic algorithm, the best scheduling solution can be found. The stopping condition for the algorithm is if there exists a tree which meets the heat limit requirement. After having set the best mapping solutions, we compare the current mapping relationship cost divisor with each of the solutions; then pick the solution, which has lowest cost. We choose the solution with the lowest cost because it will have the least effect on the current scheduling structure.

5. ALGORITHM ANALYSIS

5.1.GLOBAL SCHEDULING ALGORITHM

With the benefit of the genetic algorithm in the cloud computing environment for VM resource scheduling, this project presents an unbiased scheduling strategy for VM resources by using the genetic algorithm approach.

Starting from the initialization phase in the private cloud environment, we took the optimal solution provided by the genetic algorithm for each VM scheduling request. At the start when there are no VM resources present in the current system, we pick the solution based on the algorithm which uses the computed probability. With the increase in the number of VM resources and system run time, the algorithm computes the influence it will have on the system with the help of current state and the historical data. The algorithm will arrange the VM which

needs to be deployed with every physical machine and creates a solution set. After the solution set is ready the strategy chooses the optimal solution which has lower number of VM migrations. The Entire approach procedure is as follows:

Step 1: In the initial phase there are no VM resources that are present in the system, so no historical information can be obtained. If there is one VM resource that needs to be scheduled, based on the computed probability P (ratio of single VM load to the sum of all VM loads), the algorithm chooses the most suitable, free, smallest loaded physical server and then begins scheduling.

Step 2: As the number of VM resources in the system starts increasing with the running time, the algorithm computes the load and variance of every physical node in every solution from the solution set S by using historical and current state of the system.

Step 3: Then the algorithm uses the genetic algorithm approach to calculate the optimal mapping result for each solution in S. Whichever solution meets the predefined system variance constraints is referred as the best solution.

Step 4: The algorithm also calculates correspondingly the migration costs divisors of each result in S to attain the best mapping solution;

Step 5: By going through the cost divisor of each result, the algorithm chooses the one solution which has the lowest cost as the final scheduling solution and completes the scheduling.

Step 6: If there is new VM to Schedule then go back to step 2.

We have used the genetic algorithm in every solution and every scheduling to find the best mapping solution. To achieve the load balancing, we can take the best mapping solutions every time. Accumulation of the best solution can make it easier to find the best load balancing in quick time. One time scheduling cannot achieve the best load balancing because due to the load

variation over the time. We are able to find the load balancing for scheduling by having a low migration cost.

5.2.EXPERIMENT AND ANALYSIS OF GENETIC ALGORITHM

To analyze the behavior of the genetic algorithm with VM scheduling, the following experiment was carried out. We presume to have 5 physical machines with 15 VMs started on them. We can see the mapping relationship between VMs and physical machines in before section Figure 4. We used the predefined data for last 15 minutes, which is shown in the table 1. For the whole system condition we assumed the following configuration values.

Population scale is 50

Probabilities, Replication Pr = 0.1, Hybridization = Pc = 0.9 and variation is self-adaptive probability

Theoretically, Hybridization probability can be Pc = [0,1] and variation probability Pm = (0,1).

System load variation constraint σ_0 = 0.5, Stopping condition parameter.

By setting up the above parameters and running the algorithm, we got the following mapping relationship solution shown in Figure 5 after the section.

As an Example the next data exemplifies the mapping.

Virtual Machine	CPU Utility	Virtual Machine	CPU Utility
V1	28.8	V9	18
V2	23.4	V10	9.2
V3	17.9	V11	8.8
V4	16.8	V12	7.3
V5	12.6	V13	8.1
V6	22.3	V14	28.8
V7	13.9	V15	24
V8	40.2	V16	26.9

Table 1 Data for mapping with VM load.

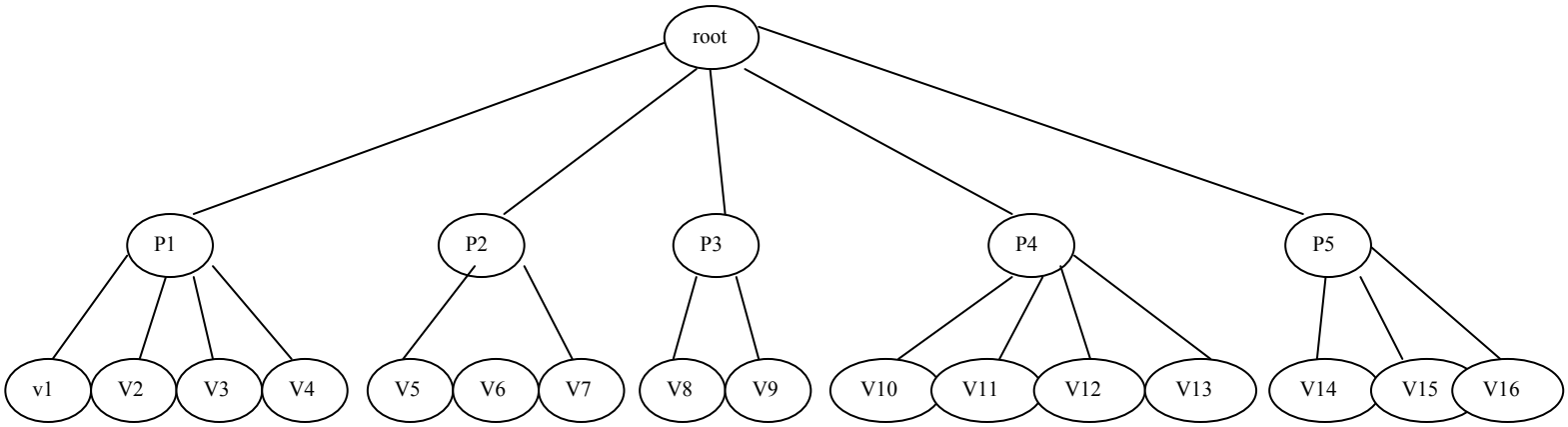

Figure 4 Mapping relationships before using algorithm.

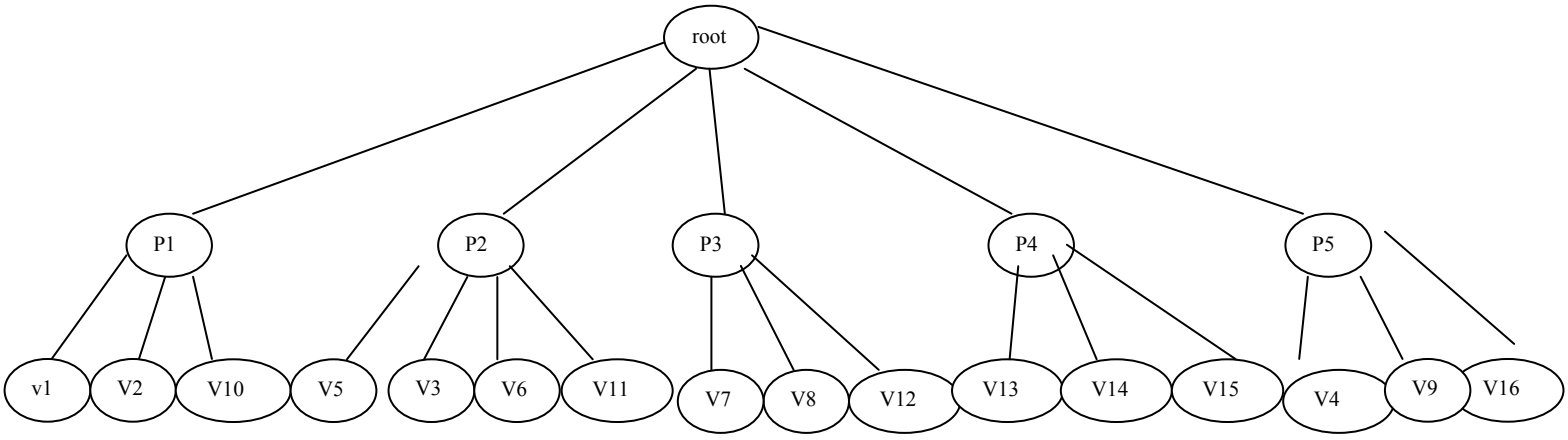

Figure 5 Mapping relationships after using algorithm

In the course of the experiment, we can see in Figure 4, the tree structure mapping relationships before using the algorithm. And in Figure 5, we can see the tree structure after the algorithm is applied on the Figure 4 tree structure. We can also see that VM loads are balanced on each physical machines i.e. every physical machine have approximately equal amount of load. The heat variation constraint, which is the stopping condition for the algorithm is also achieved i.e. overall system load variation, should be less or equal to 0.5. As a result, we can conclude that the above explained algorithm has improved the load balancing.

In the course of the experiment, we can see in Figure 4 and 5, the before and after effect on the tree structure mapping relationships using the algorithm. Figure 5, which shows the effect after using the algorithm, the VM loads are balanced on every physical node and also the system load variation parameter is also achieved. As a result, we can conclude that the above explained algorithm has quite a better globally astringency and it can come closer to be the best solution in very little time.

6. EXPERIMENT

6.1.HARDWARE SETUP

After performing the certification of the astringency behavior of the genetic algorithm, we carried out an experiment to test the performance of the overall strategy. We used most popular open source Virtual machine management infrastructure 'OpenNebula' [20]. For the OpenNebula front end, we chose a physical machine with the configured with operating system UBUNTU 11.10, Intel® Core™ 2 Duo 2.4GHz CPU, and 4.0GB of RAM. On this machine, OpenNebula front-end is installed to schedule and manage virtual machines. Along with the host machine, we chose 6 physical machines as client nodes on which we installed OpenNebula client platform with KVM VM. The client machines are configured with operating system Ubuntu 11.10, Intel® Core™ 2 Duo 2.4GHz CPU, 4.0GB of RAM, and 250GB disk capacity. The whole private cloud was created using the Local Area Network (LAN). The VM images were created using the Ubuntu –OpenNebula documentation [24]. The tree structure is created such that, the host physical machine will work as root/scheduler, the other client machines will work as the second level of nodes and the VM deployed on the client machines will work as the leaf nodes.

6.2.SOFTWARE COMPONENTS

The whole algorithm was realized using JAVA. For interaction between Java based scheduler and OpenNebula, we used Java OpenNebula Cloud API 3.0 [22]. By using this, all the RPC calls can be managed and VM information can be retrieve.

6.2.1. OPENNEBULA

In private or hybrid cloud, the important infrastructure component is VI management (Virtual Infrastructure). It is the mechanism, which allocates virtual machines dynamically on the set of resource that meets the given requirement. OpenNebula manages the virtual machine infrastructure which is used for variety of responsibilities

1. Deployment of virtual machines, which can be deployed as a group or individual or on a public cloud or on a local resource.
2. It helps to automate the setup process of virtual machines (setup network, create disk images as per requirement, etc.) without touching the lower level of virtualization layer (external cloud like EC2 or internal private cloud like KVM, Xen, VMWare).

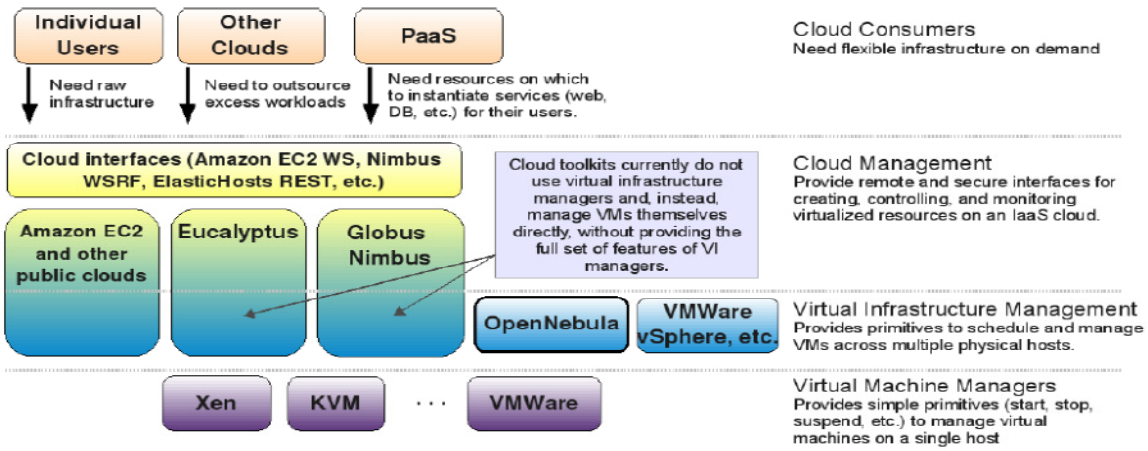

Fig. 1. The Cloud ecosystem for building private clouds

Figure 6 Cloud Ecosystems [20].

The figure 7 below shows the OpenNebula architecture which includes comprehensively several virtual infrastructure management key components which are focused in different aspects.

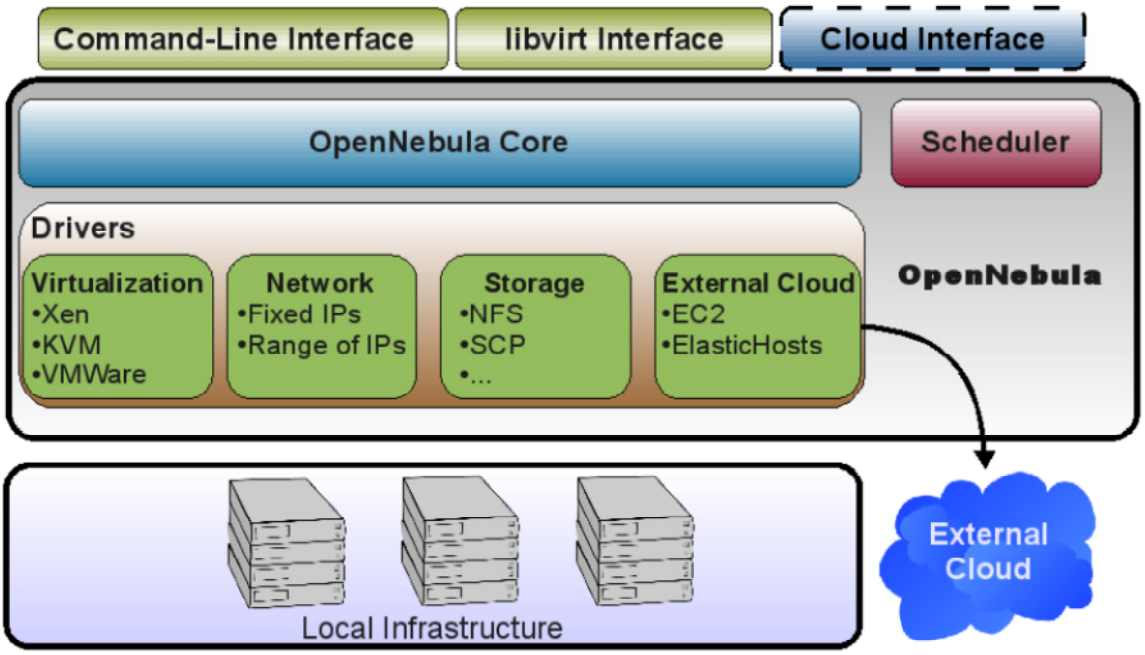

Figure 7 OpenNebula architecture [20]

OpenNebula controls the entire lifecycle of a VM in order to manage VM infrastructure, OpenNebula core infrastructure is organized into three different areas.

1. The network structure layer (e.g., DHCP servers, firewalls or switches).
2. To prepare disk images for virtual machines, it uses virtual appliance tools or distributed file systems as image and storage technologies.
3. To use hypervisors to create and control VMs.

The Core of the OpenNebula uses pluggable drivers and executes specific storage, network or virtualization operation. The OpenNebula provides a consistent management layer without considering the underlying infrastructure as it shows that it is not tied to any environment and it is portable.

The core component also takes care of the deployment of services; which includes a set of unified components (database backend, web servers, etc) that requires more than one VMs. Therefore, a related group of VMs is treated as an entity in OpenNebula. The core not only manages the VMs but also it holds the delivery of the interconnected context information (e.g., digital certificates, IP address of the servers, End user software licenses).

The VM placement in OpenNebula is decided by using the independent scheduling component. All the requests received by OpenNebula, the scheduler have every access of the information related to all the requests. By using this information the scheduler is solely responsible for

1. Creating and update the resource schedule
2. Sending suitable deployment commands to OpenNebula Core.

The default scheduler which comes with OpenNebula uses the ranking mechanism which allocates the VMs on the physical nodes based on the ranking strategy. This algorithm can be configured by the administrator. The configuration depends on the real-time VM as well as available Physical node data. The Haizea, the resource manager can be used in the place of this default scheduler.

OpenNebula also proposes management interfaces to combine the core mechanism within accounting or monitoring frameworks as data center management tools. OpenNebula implements the libvirt API; it is an open interface for VM management and CLI (command line interface). By using cloud interface a subset of libvirt API can be open to external users [20]. Finally, OpenNebula can maintain a hybrid cloud model with the use of Cloud Drivers to interface with outer clouds. By using this strategy the local infrastructure can be supplied with the calculating capacity from an outer cloud to gather high demands. OpenNebula included an EC2 driver which

can submit the requests to Amazon EC2 or an Elastic host driver or Eucalyptus. By doing this it better provides user access requests or implements the high accessibility approaches.

6.2.2. HAIZEA

Haizea [20] is the most popular open source resource lease manager. It can be used as a VM scheduler for OpenNebula backend. Haizea provides leasing capabilities which are missing in other cloud computing systems. Especially for private clouds, advance reservation and resource preemption techniques are designed.

An open source lease supervisor Haizea can be used in two purposes

1. It can be used as VM scheduler with OpenNebula.
2. It can be used as a simulator on its own to calculate the performance of various VM scheduling techniques.

The key feature that Haizea uses for resource provisioning is the lease. Naturally a lease can be seen as a contract between two parties, where one party acts as a service provider, which provides a set resources, and the other party act as a consumer who will use the set of resources provided by the service provider. Whenever a user requires computational resources, it can be requested through Haizea in the lease form. As per lease terms a VM is created for whole set of computational resources and that VM is then managed by OpenNebula system [20].

In Haizea, the lease terms include hardware resources, software environments and time period from start to end time in which all the requested hardware and software should be available.

At present the most effective reservation schemes supported by Haizea are advance reservation lease in which the resources must be available at the given time frame. The other lease scheme is the best-effort lease in which resources must be allocated as soon as possible. If

Haizea is busy then the requests are queued. There are immediate leases, in which resources are allocated upon request immediately or in other case resources won't be allocated.

6.2.3. ADVANCE RESERVATION

The most important feature in Haizea and OpenNebula is advance reservation for the computational resources. In the previous studies of parallel computing and the suspension or resumption capabilities it showed that advance reservation produces system underutilization since it has to vacate resources first before reserving then.

In the case of leases which can be implemented for virtual machines, advance reservation leases can work more competently by using resource preemption. Before starting the reservation, it suspends the VMs, which have lower-priority lease and resumes after the reservation process ends. It might need migration also for load balancing to other nodes or cloud infrastructure [21]. With previous studies of parallel computing in the context, resource preemption shows that VMs have a very eye-catching quality for suspension of computation without giving the knowledge that the applications inside the VM are going to be migrated, suspended or resumed.

The disadvantage of using VM is, it introduces an extra overhead, which causes the additional challenges in the scheduling mechanism. In particular, if the overhead is not managed effectively, the deployment of the VM image, which is required by the lease, can cause harmful impact on the performance.

Instead of assuming that the overhead will be deducted from a users request allocation, Haizea handles this overhead separately. The process is complex for supporting various types of leases with the different incompatible requirements which must be accepted. One transfer of a lease starting at 4pm can delay the transfers of other best-effort leases which results in long wait time.

Sample configuration of Advance Reservation

When you require your VM accessible at a precise time, this is called an advance reservation or AR.

```
HAIZEA = [
 Start       = "+00:00:20",
 Duration    = "00:01:00",
 Preemptible = "no"
]
```

However, in place of stating that you want your VM to begin after a definite amount of time has passed (20 seconds). You can also provide the exact start time you want:

```
HAIZEA = [
 Start       = "2008-11-04 11:00:00",
 Duration    = "03:00:00",
 Preemptible = "no"
]
```

Numbers of optimization techniques used by Haizea are as follows:

1. Disk images reusability: To minimize the time overhead in preparation of disk images, it reuses the disk images across leases.
2. Schedule run time overhead before a specific time: runtime overheads like VM suspension, migration and resumption can be scheduled before specific time which might be necessary for any new scheduling.

Haizea uses a resource slot table for scheduling which has all the information related to the physical clients as well as VMs. This resource slot table is updated by Haizea over the time.

6.2.4. BEST-EFFORT LEASE

Best-effort lease uses the FCFS (First Come First Serve) queue approach [21] with the backfilling technique (an optimization technique mostly used in queue-based systems).

Advance reservation uses the greedy approach which follows the selection strategy on physical servers that will minimize the no. of preemptions.

A sample configuration for Best-effort Provisioning

In this approach user instructs Haizea to determine the start time by using the best-effort provisioning mechanism. The request will complete as soon as the allocated resource becomes ready. This approach uses a request queuing mechanism. If the request cannot be fulfilled at that time, it will be placed in a queue and the request has to wait until its turn. We can use two commands to see the states of the queue (haizea-list-leases and haizea-show-queue).

The best-effort approach can be non-preemptible or preemptible. If it is non-preemptible, user has to wait in the queue to get the request done.

```
HAIZEA = [
  Start       = "best_effort",
  Duration    = "01:00:00",
  Preemptible = "yes"
]
```

The current version of Haizea has the hardcoded resource selection algorithm. In the future, developers may specify their own selection of decision policies (prioritize lease based on user or group). The selection policy may also be used in a decision making module which will decide whether lease should be accepted or not.

6.3.ANALYSIS

The analysis mainly observes the effect of load balancing by using the algorithm and the migration cost required to obtain the load balanced system after scheduling. It also compares the algorithm explained in this project with the current popular open source VM scheduling methods including Haizea leased based algorithms (Advance reservation and Best effort provisioning).

6.3.1. ALGORITHM EFFECT ANALYSIS

The experiments are performed using this strategy to analyze the effect of load balancing and then compare it with the Haizea scheduling strategies (Advance preemption and Best-effort provisioning).

The Following two different scenarios were considered for the experiment.

a. When the system load variation is stable for last T time interval.

b. When the system load variation is evident for last T time interval. .

In the following figures, Figure 8 and Figure 9, we can see that while system load variation is comparatively stable, all the strategies to a certain extent successfully achieve the system load balancing. But as we can see in Figure 9, when the system load variation is significant, the strategy from this project works better because of the consideration of historical data while scheduling.

The Advance preemption works better in parts than the best effort provisioning algorithm. This algorithm shows that it improves the way to achieve the system load balancing than the two other approaches.

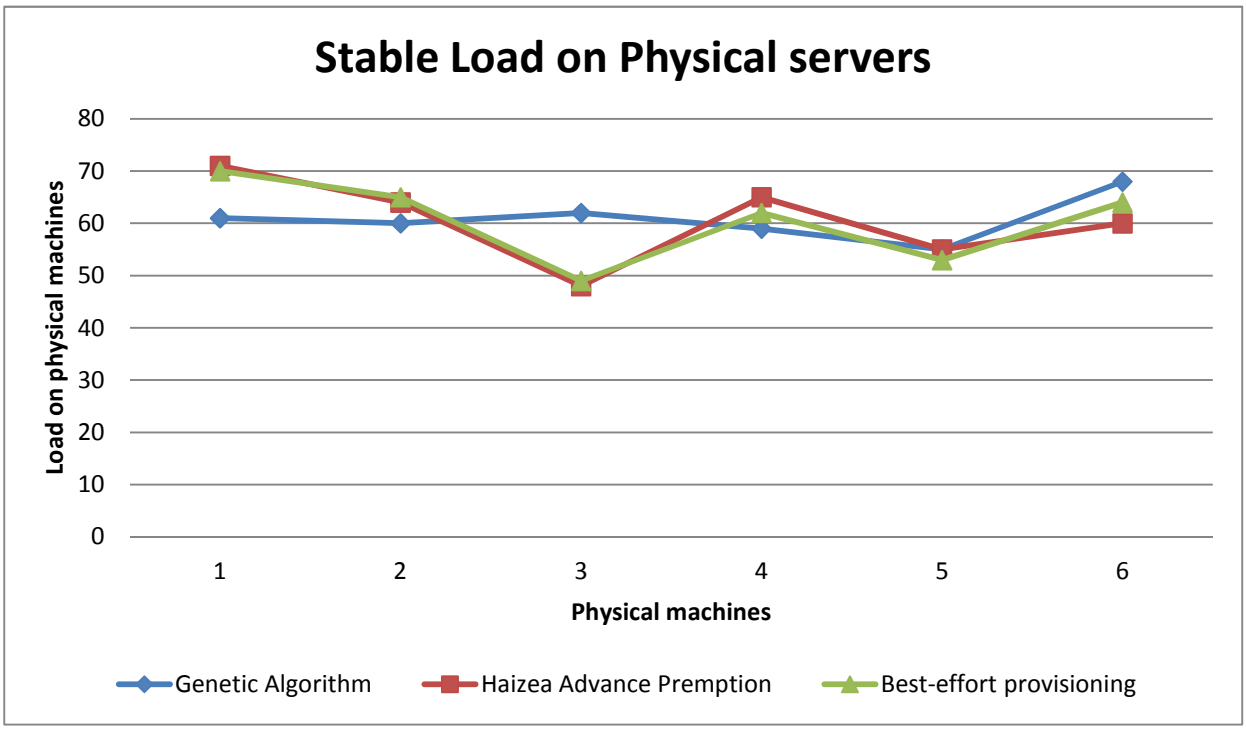

Figure 8 comparisons of three scheduling policies when system load is stable

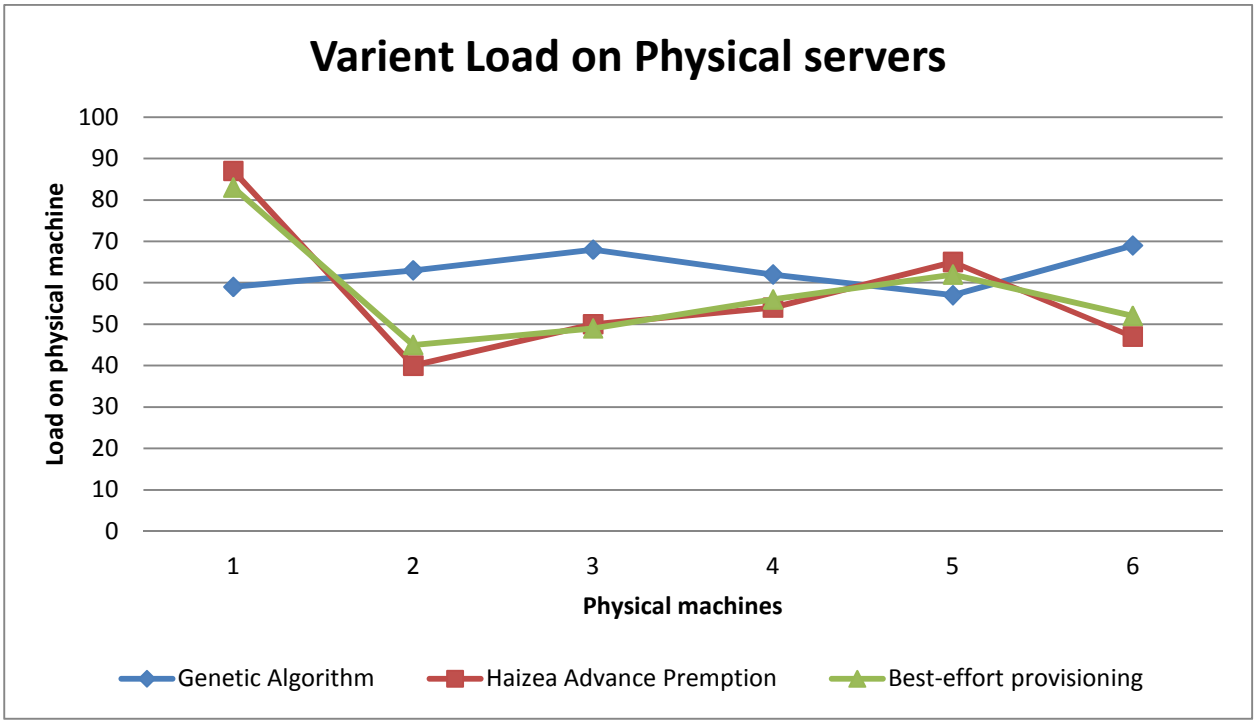

Figure 9 comparisons of three scheduling policies when system load is evident

6.3.2. MIGRATION COST ANALYSIS

In the current strategies, with some exceptional occurrences, there has been an evident increase in the load of the some of the nodes in the system, which was caused by the repeated usage, which led to the load imbalance in the entire system. That happened because normal system load balancing cannot be achieved in one time scheduling, so it requires doing the VM migration to perform the system load balancing. But VM migration cost cannot be ignored. The VM migration can be categorized in phases e.g. Stop the VM which needs to be migrated, send the data over the network which costs more, deploy the VM on the new physical machine, and then resume from the last state of the VM [21].

The problem to be considered is the VM scheduling along with finding the suitable location for the VM migration with the minimum number of VM migrations. The algorithm computes the influence it will have on the system before actual scheduling happens then it chooses the solution with less number of migrations. Following Figure 10 and Figure 11 show the no. of VMs migrating to attain load balancing after scheduling with the different numbers of VM running for the system load variation is stable and evident respectively.

When the system load is stable, the performance difference between all the algorithms is very little. But when the system load variation is evident, then the strategy in this project shows the notable benefit. The following trials show that the strategy in this project can reduce the migration cost significantly.

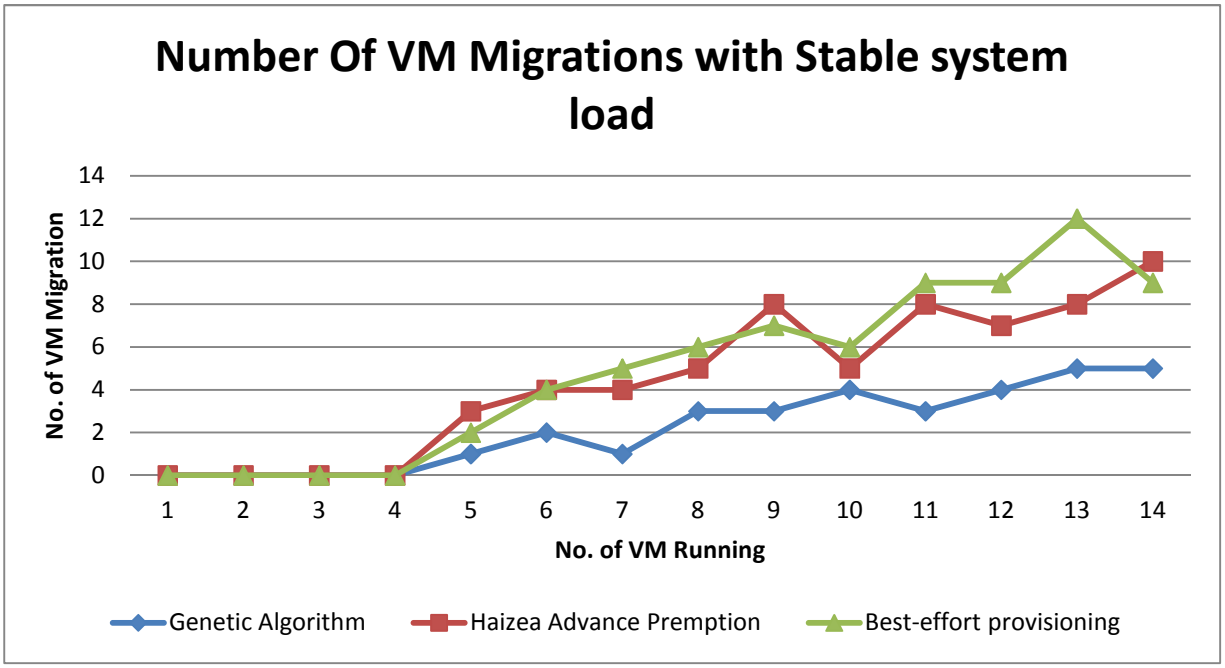

Figure 10 Number of VM migrations with stable system load

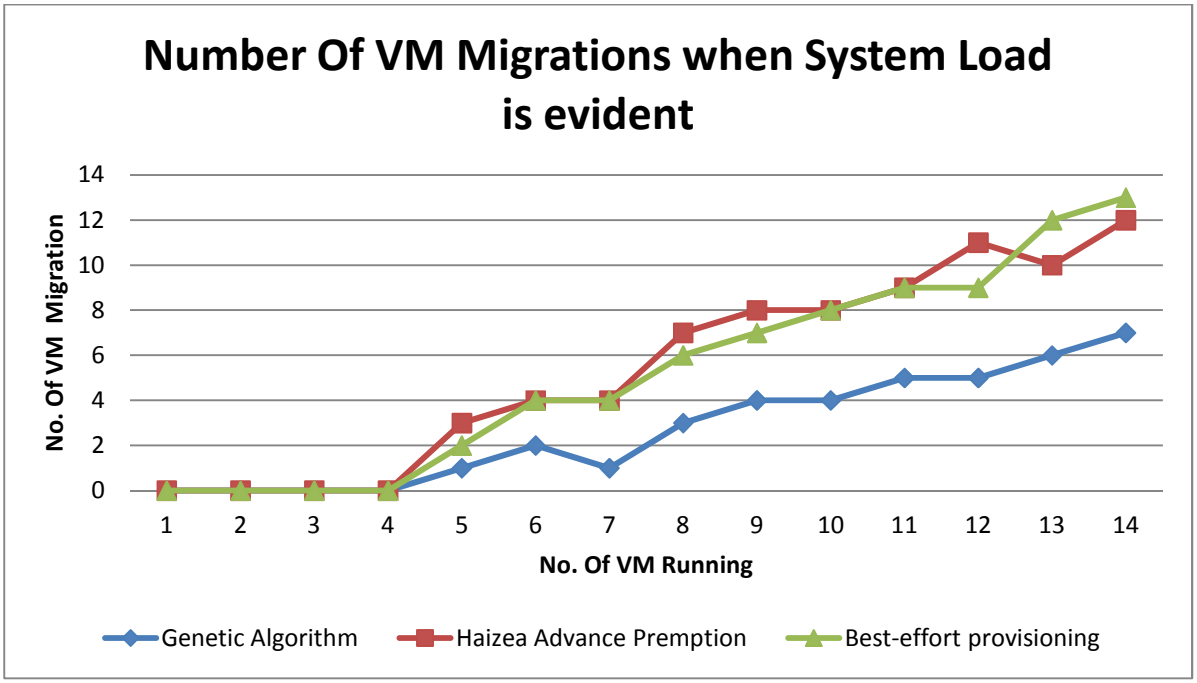

Figure 11 Number of VM migrations when the system load is more

7. CONCLUSION

To better present load balancing in VM resource scheduling approaches, this project shows a scheduling strategy based on the genetic algorithm. If we consider the problem on a large scale such as in the cloud computing environment, with the advantage of the genetic algorithm approach; this technique is able to compute the influence it will have on the entire system in advance with the help of historical data and systems current state. The VM resource that needs deploying is arranged to every physical server to create a potential solution set and finally chooses the solution which will have least affect on the system after VM is physically deployed. By doing it in this manner, the strategy achieves the best load balancing and keeps the number of dynamic migrations as low as possible to resolve the issues of migration cost as well as load unbalance which is caused by current scheduling approaches.

The investigational outcome shows that, this technique can better realize proper load balancing and system resource utilization.

This project builds a prototype for the strategy in the concrete cloud computing environment. The technique takes in account the historical data along with the current VM system state and also tree structure encoding, elitism selection strategy; a self adaptable variation used in the genetic algorithm puts control on the strategy so that it has improved astringency.

8. FUTURE WORK

This project shows a strategy of scheduling VM resources based on the historical data of the system and it tries to have the least number of VM migrations in the final solution. It will be interesting to find the enhancements for this approach to get the optimal way of solving load imbalance problem. But in the real world of cloud computing environment the scenario might be

different which could have dynamic changes in VM and computing costs. The virtualization software might as well cause unexpected load wastage because of the more number of VMs executed in each physical machine.

Furthermore, to analyze the unpredictability of the load wastage a monitoring mechanism will be very interesting research subject. It will help to solve the main issue of load balancing in better way.

9. REFRENCES

[1] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia, “Above the clouds: A berkeley view of cloud computing,” UCB/EECS-2009-28.

[2] Cloud Computing Definition Gartner.

http://www.gartner.com/it/page.jsp?id=1035013

[3] Borja Sotomayor, Kate Keahey, Ian Foster, and Tim Freeman, “Enabling cost-effective resource leases with virtual machines,” ACM/IEEE International Symposium on High Performance Distributed Computing 2007 (HPDC 2007), 2007.

[4] L. Cherkasova, D. Gupta, and A. Vahdat, “When virtual is harder than real: Resource allocation challenges in virtual machine based it environments,” Technical Report HPL 2007-25, February 2007.

[5] Clark C, Fraser K, Hand S, “Live Migration of Virtual Machines[C],” Proceedings of the 2nd Int’l Conference on Networked Systems Design & Implementation. Berkeley, CA, USA, 2005.

[6] Wei Wang, “A reliable dynamic scheduling algorithm based on Bayes trust model,” Computer Science, 2007.

[7] Rewinin H E, Lewis T G, Ali H H, “Task Scheduling in parallel and Distributed System Englewood Cliffs,” New Jersey: Prentice Hall,1994, pp. 401-403.

[8] Wu M, Gajski D, Hypertool, “A programming aid for message passing system,” IEEE Trans Parallel Distrib Syst, 1990, pp. 330-343.

[9] Guiyi Wei; Vasilakos, A.V.; Naixue Xiong; , "Scheduling Parallel Cloud Computing Services: An Evolutional Game," Information Science and Engineering (ICISE), 2009 1st International Conference on , vol., no., pp.376-379, 26-28 Dec. 2009.

[10] Sih G C, Lee E A, “A compile-time scheduling heuristic for Interconnection-constraint heterogeneous processor architectures,” IEEE Trans Parallel Distributed System, 1993, pp. 175-187.

[11] Rewinin H E, Lewis T G, “Scheduling parallel programs onto arbitrary target machines,” J Parallel Distributed Computing, 1990, pp.138-53.

[12] Tram Truong Huu, Johan Montagnat, "Virtual Resources Allocation for Workflow-Based Applications Distribution on a Cloud Infrastructure," 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, 2010.

[13] Wood T, “Black-box and Gray-box Strategies for Virtual Machine Migration[C],” Proceedings of the 4th Int’l Conference on Networked Systems Design & Implementation, IEEE Press, 2007.

[14] E. Goldberg, “The existential pleasures of genetic algorithms,” In: Genetic Algorithms in Engineering and Computer Science, Winter G ed. New York: Wiley, 1995, pp. 23-31.

[15] Introduction to Genetic algorithm
http://www.obitko.com/tutorials/genetic-algorithms/crossover-mutation.php

[16] Chen Ming; Li Mengkun; Cai Fuqin; , "A Model of Scheduling Optimizing for Cloud Computing Resource Sevices Based on Buffer-pool Agent," Granular Computing (GrC), 2010 IEEE International Conference, Aug. 2010.

[17] Exploiting Dynamic Resource Allocation for Efficient Parallel Data Processing in the Cloud
http://dsl.cs.uchicago.edu/TPDS_MTC/papers/TPDSSI-2010-01-0012.pdf

[18] Jinhua Hu; Jianhua Gu; Guofei Sun; Tianhai Zhao; , "A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environment” *Parallel*

Architectures, Algorithms and Programming (PAAP), 2010 Third International Symposium on, vol., no., pp.89-96, 18-20 Dec. 2010

[19] Basic genetic algorithm

http://www.obitko.com/tutorials/genetic-algorithms/ga-basic-description.php

[20] Open source solution to private cloud

http://www.mcs.anl.gov/uploads/cels/papers/P1649.pdf

[21] Haizea Documentation

http://haizea.cs.uchicago.edu/manual/node35.html#SECTION03451000000000000000

[22] Topology-Aware Resource Allocation for Data-Intensive Workloads

http://conferences.sigcomm.org/sigcomm/2010/papers/apsys/p1.pdf

[23] OpenNebula Web site for java cloud API

http://opennebula.org/documentation:rel3.0:java

[24] Spanning tree

http://en.wikipedia.org/wiki/Spanning_tree