# Virtual Machine Consolidation using Roulette wheel selection Strategy

K.R. Raghi
Computer Science and Engineering
CEG, Anna University, Chennai, India

DOI : 10.55041/IJSREM11419

***Abstract*— Cloud computing data centers are growing rapidly in both number and capacity to meet the increasing demands for highly-responsive computing and massive storage. Such data centers consume enormous amounts of electrical energy resulting in high operating costs and carbon dioxide emissions. The reason for this extremely high energy consumption is not just the quantity of computing resources and the power inefficiency of hardware, but rather lies in the inefficient usage of these resources. Virtual Machine [VM] consolidation involves live migration of VMs hence the capability of transferring a VM between physical servers with a close to zero down time. It is an effective way to improve the utilization of resources and increase energy efficiency in cloud data centers. VM consolidation consists of host overload/under load detection, VM selection and VM placement. In Our Proposed Model We are going to use Roulette-Wheel Selection Strategy, Where the VM selects the Instance type and Physical Machine [PM] using Roulette-Wheel Selection Mechanism**

***Keywords—searchable encryption, dynamic update, cloud computing***

## I. INTRODUCTION

Virtual Machine (VM) consolidation is an effective technique to improve resource utilization and reduce energy footprint in cloud data centres. It can be implemented in a centralized or a distributed fashion. Virtual Machine (VM) consolidation is one of the key mechanisms of designing an energy-efficient dynamic Cloud resource management system. It is based on the premise that migrating VMs into fewer number of Physical Machines (PMs) can achieve both optimization objectives, increasing the utilization of Cloud servers while concomitantly reducing the energy consumption of the Cloud data centre. However, packing more VMs into a single server may lead to poor Quality of Service (QoS), since VMs share the underlying physical resources of the PM. To address this, VM Consolidation (VMC) algorithms are designed to dynamically select VMs for migration by considering the impact on QoS in addition to the abovementioned optimization objectives. VMC is a NP-Hard problem and hence, a wide range of heuristic and meta-heuristic VMC algorithms have been proposed that aim to achieve near-optimality by Researchers. Cloudsim Setup: - CloudSim is a framework written in Java Programming language is a toolkit for modelling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. CloudSim developed by Buyya began in 2009 in the laboratory of Grid Computing and Distributed Systems 2 (GRIDS) University of Melbourne. CloudSim has classes that serve as a simulator of cloud computing components such as broker, CIS (Cloud Information Service), VM, cloudlet (job or task in cloud computing), datacentre and PMs. Scheduling the cloudlet on VMs and in datacentre is based on two policies 1. Time Shared 2. Space Shared. In Time-Shared scheduling policy, allocates one or more processing elements to a VM and allows sharing of processing elements by multiple VMs. This can be explained as sharing of executing powers such as CPU, logical processor etc. It processes several requests at a time and shares the computing power of that machine, so they affect each other's processing time which results in performance degradation. In Space Shared policy allocates one or more processing elements to a VM, and doesn't allow sharing of processing elements. If there are no free processing elements then the allocation fails. In other words, Space Sharing refers to sharing of memory space such as hard disk, RAM etc.Ease of Use

## II. RELATED WORK

Yousefipour A et al [1] proposed a mathematical model aimed at reducing power consumption and costs by employing an effective VM consolidation in the cloud data center. Subsequently, they proposed a genetic algorithm–based meta heuristic algorithm, namely, energy and cost-aware VM consolidation for resolving the problem. Amany Abdelsamea et al [2] proposed the usage of hybrid factors to enhance VM consolidation. Specifically, they developed a multiple regression algorithm that uses CPU utilization, memory utilization and bandwidth utilization for host overload detection. The proposed algorithm, Multiple Regression Host Overload Detection (MRHOD), significantly reduces energy consumption. Seyed Saeid Masoumzadeh et al [3] proposed a model to concentrate on the VM selection task and proposed a Fuzzy Qlearning (FQL) technique so as to make optimal decisions to select virtual machines for migration. They validated their approach with the CloudSim toolkit using real world PlanetLab workload. Giuseppe Portaluri et al [6] compared a set of Virtual Machine (VM) allocators for Cloud Data Centers (DCs) that perform the joint allocation of computing and network resources. 5 Tanasak Janpan et al [4] proposed a VM consolidation framework for Apache CloudStack, which is a popular open source cloud platform software suite. Md Anit Khan et al [7] proposed a novel heuristic Dynamic VM Consolidation algorithm, RTDVMC, which minimizes the energy consumption of CDC through exploiting CSU provided information. Zoltan Adam Mann [5] investigated the impact of the choice of cloud simulator on the implementation of the algorithms and on the evaluation results

## III. PROBLEM FORMULATION

Cloud computing data centres are growing rapidly in both number and capacity to meet the increasing demands for highly-responsive computing and massive storage. Such data centres consume enormous amounts of electrical energy resulting in high operating costs and carbon dioxide emissions. The reason for this extremely high energy consumption is not just the quantity of computing resources and the power inefficiency of hardware, but rather lies in the inefficient usage of these resources. Virtual Machine [VM] consolidation involves live migration of VMs hence the capability of transferring a VM between physical servers with a close to zero down time. It is an effective way to improve the utilization of resources 3 and increase energy efficiency in cloud data centres. VM consolidation consists of host overload/underload detection, VM selection and VM placement. In Our Proposed Model We are going to use Roulette-Wheel Selection Strategy, Where the VM selects the Instance type and Physical Machine [PM] using Roulette-Wheel Selection Mechanism.

## IV. THE PROPOSED SCHEMES

Is to show how genetic algorithms can be used in solving VM consolidation problem. To understand how genetic algorithm works and to find the best algorithm in terms of power and cost utilization. There are 3 major roles in the system model including user, cloud provider, and cloud broker. The cloud broker is the main part of the system model, Which acts as an intermediary between the user and providers to handle the use and delivery of cloud services, considering the performance requirements. The cloud broker can provide the system transparency in which the cloud providers are invisible to the users and a user interacts with the broker instead of communicating directly with the providers. User sends request in form of cloudlets to the broker and the broker assigns it to the cloud provider. The virtual infrastructure manager component periodically asks the registry component about any updates on the availability of resources. The registry obtains the required information from each particular cloud provider. A new provider interested in joining the environment, registers its own information in the registry

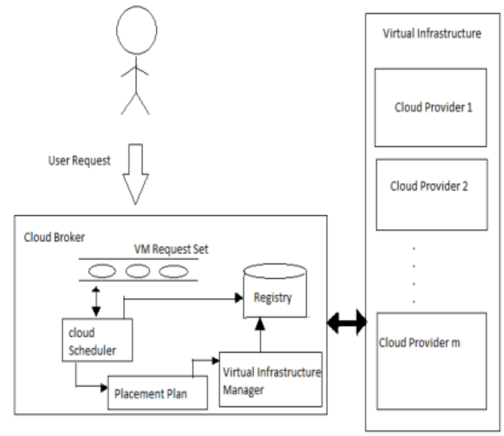


Fig. 1 Architecture Diagram

*POWER MODEL* is the total power consumption of physical servers is considered as the power consumption of data centres. The power consumption of each server is calculated by considering 2 states of the server, ie static and running state. The static state represents an idle state in which no VM exits on the server and the server is active. The running state involves the allocation process of VMs. Therefore, Pi is the power consumption of a server si at a data centre that can be modelled by a power function as

$$Power = P\_idle + P\_placement + P\_vm + P\_switch$$

8 Power is the total power consumption of server si at a data centre, P_idle is the power consumption of server si in the idle state. P_placement is the power consumed by running different VM instances on server si, P_vm is the outcoming results on running a new vm on a server without processing the workload, P_switch is the power Consumption on system when server switch between on and off states.

### *a. ENCODING SCHEMA*

One of the most important elements of the Genetic Algorithm is the chromosome. Representation or encoding of the chromosome affects the performance. In our proposed project, each chromosome consists of 2 segments and the length of it is $2N + L + M-2$. Since each VM has to be assigned to one instance type and each VM has to be assigned to PM on the last stage of the problem, integer encoding was used to define this situation. The first segment of the chromosome is comprised of $|N + L – 1|$ genes that is the permutation of integer numbers from 1 to $N + L – 1$. Numbers larger than N are used as delimiters. Hence the decoding schema of the first segment, from the beginning of the array to first delimiter, indicates which of the VM selected the first instance type for execution. This method of selecting the instance type continues 9 with respect to delimiter. Here N denotes the total number of VMs and L denotes the total number of instance types. If 2 of the delimiters appear consecutively, it shows that none of the VMs select the specified instance type.

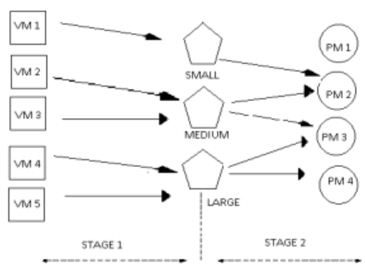


Fig. 2 A Sample VM allocation problem

STAGE 1: VM CHOOSE THE INSTANCE TYPE
STAGE 2: VM CHOOSE THE PHYSICAL MACHINE

The initial population is randomly generated by considering the number of servers, number of virtual machines, and number of virtual machine instance types. In order to reduce the computation time of GA execution, the range of values for each gene can be predetermined based on the Integer encoding schema.

### b. *EVALUATION FUNCTION*

The generated Chromosome will be evaluated against a power and cost function, lower objective value implies the higher performance of the chromosome. 11 The performance of each chromosome in the population is proportional to a placement plan, the objective or fitness function determines the power consumption and cost of placement plan based on information received from the chromosome.

$$v_i^{cpu}=\max\left(\frac{\sum_{j=1}^{L}\sum_{k=1}^{N}R_j^{CPU}Xijk}{Yi*C_i^{CPU}}-1.0\right),\qquad \forall i\in\{1,2,\dots,M\}$$

$$v_i^{Memory}=\max\left(\frac{\sum_{j=1}^{L}\sum_{k=1}^{N}R_j^{Memory}Xijk}{Yi*C_i^{Memory}}-1.0\right),\qquad \forall i\in\{1,2,\dots,M\}$$

$$v_i^{Disk}=\max\left(\frac{\sum_{j=1}^{L}\sum_{k=1}^{N}R_j^{Disk}Xijk}{Yi*C_i^{Disk}}-1.0\right),\qquad \forall i\in\{1,2,\dots,M\}$$

$$v_T=\frac{\sum_{i=1}^{M}\left(v_i^{CPU}+v_i^{Memory}+v_i^{Disk}\right)}{M},\qquad \forall i\in\{1,2,\dots,M\}$$

$$F=(0.5*F2)+(1+\beta*v_T)$$

### c. *NOTATION USED IN THE MODEL*

| Indices | Notations |
|---|---|
| i | Server index, i=1,2,…,M |
| j | VM instance type index, j=1,2,…,L |
| k | VM index, k=1,2,…,N |
| Parameters | |
| $Xijk$ | 1, if VM k can be done on VM instance type j, else 0 |
| $Yi$ | 1, if atleast one VM instance type runs on server i, otherwise 0 |
| $Rj^{CPU}$ | CPU requests of each VM instance type j |
| $Rj^{Memory}$ | Memory request of each VM instance type j |
| $Rj^{Disk}$ | Disk request of each VM instance type j |
| $Ci^{CPU}$ | CPU capacity of each server i |
| $Ci^{Mem}$ | Memory capacity of each server i |
| $Ci^{Disk}$ | Disk capacity of each server i |

### d. *CROSSOVER OPERATOR*

The crossover operator in GA is used to produce new individuals from 2 parents. In each generation, the number of offspring that are added to the population by this operator is controlled by a crossover percentage. We employed a segment based crossover operator that was based on a single- point crossover. At first a gene is selected randomly in each segment of the chromosome, and then the created fragments of each parent to produce offspring are combined together.

*e. ALGORITHM*

INPUT: Two parent chromosomes with q genes, $CHb = a1\ ba2\ b\ldots aq\ b$ and $CHc = a1\ ca2\ c\ldots aq\ c$

OUTPUT: two offspring chromosomes, $CHd = a1\ da2\ d\ldots aq\ d$ and $CHe = a1\ ea2\ e\ldots aq\ e$

For each segment in chromosome segments do

1. $CH'$ <- CH;
2. Randomly generate two integer values between 1 and segment size called r1, r2.
3. Exchange the elements in a $CH'$ at indexes r1 and r2.

End For Each
Return $CH'$ ;

INPUT: VMList, PMList, InstanceTypeList
OUTPUT: allocation of VMs
1. Initially Pop Size, CP, MP /* num of population, crossover percentage,
mutation percentage */
2. 2000 iterations (t>2000) // Termination Condition
3. nc<- Pop size * CP; // number of offspring
4. nm<- Pop size * MP; // number of mutants
5. population <- InitializePopulation(Pop Size); /* Generation of initial random
population */
6. EvaluatePopulation(population);
7. While the termination condition not true do
8. t<- t+1;
9. FOR i=1…nc /2 // Apply selection and crossover
10. parents <- selection(population,2);
11. offsprings<- Crossover(parents);
12. EvaluatePopulation(offsprings);
13. Population.add(offsprings);
14. End FOR
15. FOR i=1…nm // Apply mutation
16. parents <- selection(population,1);
17. offsprings<- muatation(parent);
18. EvaluatePopulation(offsprings);
19. Population.add(offsprings);
20. End FOR
21. End While
22. Population.sort(); // sort the individuals according to their fitness
23. Allocation<-population.get(first); //selection of best individual
24. Return allocation;

TABLE 1.TEST CASE

| S.No | Module Name | Input | Expected Output |
|---|---|---|---|
| *1* | Crossover | Two parent chromosomes A and B | Two offspring's with gene get shifted from one chromosome to another after segment size 'r'. |
| *2* | Mutation | A chromosome 'CH' | Exchange the elements of chromosome, at randomly generated two points r1 and r2 (1<=r1, r2<=segment size). |
| *3* | Overall Roulette Wheel Algorithm | VMList, PMList, Cloudlet List | Allocation of VMs |

## V. EXPERIMENTAL RESULT

**Simulation configuration**

The characteristics of the VM, Host and cloudlet used in the simulation are detailed below:-

**VM requirements(Each host has a VM)**

IPS rating: 250

Image Size on Disk: 10000 MB

RAM: 512 MB

Band width: 1000

Number of Requirement Pes: 1

Hypervisor: Xen

**Characteristics of Host (each datacentre has a host)**

RAM: 200000

Storage: 1000000

Bandwidth: 100000

MIPS rating of processing entity: 1000

**Characteristics of the cloudlets**

Length (in terms of instruction):40000

Input file size: 300

Output file size: 300

The cloudsim 3.0.3 toolkit is used for simulation and measuring the performance of large scale virtualized datacentre. The simulation is done with 3 datacentre, 4 host and 4 VM.

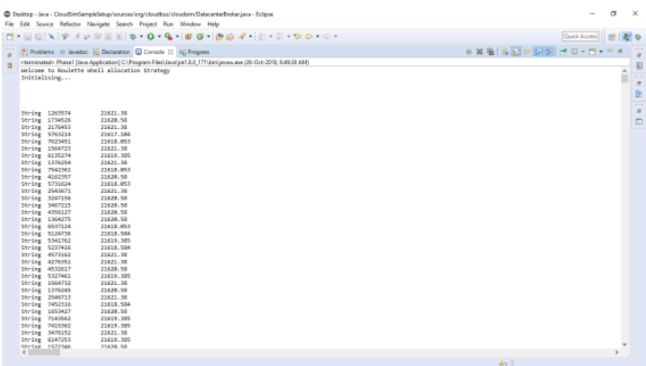

Fig. 3 Initial Population with its Power value

Figure 3 shows how initial population set is generated with different chromosome values and its corresponding power values which is calculated from power calculation function. Figure 4 shows how the best chromosome returned by the system and how the allocation of VM to cloudlet is done.

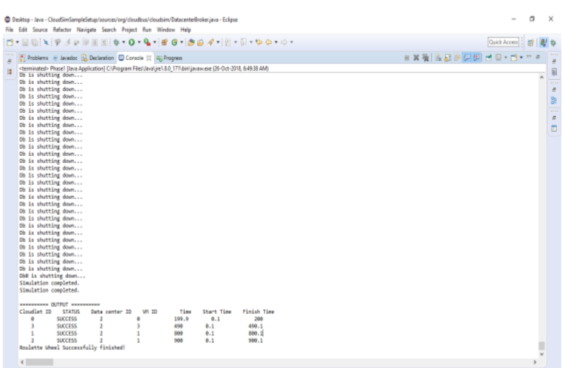

Fig. 4 Initial Population with its Power value

*f. EVALUATION METRICS*

The objective or fitness function determines the power consumption and cost of placement plan based on information retrieved from the chromosome. The graph is plotted as number of generations verses fitness value, where the fitness value is obtained as

$$\text{Fitness value} = \frac{\text{Total fitness of the population}}{\text{number of generation (population)}}$$

Total fitness of the population = sum of fitness values of all individual chromosomes in a generation Where, Individual chromosome fitness value is calculated based on formula's

$$v_i^{cpu} = \max\left(\frac{\sum_{j=1}^{L}\sum_{k=1}^{N} R_j^{CPU} Xijk}{Yi * C_i^{CPU}} - 1.0\right), \quad \forall i \in \{1,2,\dots,M\}$$

$$v_i^{Memory} = \max\left(\frac{\sum_{j=1}^{L}\sum_{k=1}^{N} R_j^{Memory} Xijk}{Yi * C_i^{Memory}} - 1.0\right), \quad \forall i \in \{1,2,\dots,M\}$$

$$v_i^{Disk} = \max\left(\frac{\sum_{j=1}^{L}\sum_{k=1}^{N} R_j^{Disk} Xijk}{Yi * C_i^{Disk}} - 1.0\right), \quad \forall i \in \{1,2,\dots,M\}$$

$$v_T = \frac{\sum_{i=1}^{M}\left(v_i^{CPU} + v_i^{Memory} + v_i^{Disk}\right)}{M}, \quad \forall i \in \{1,2,\dots,M\}$$

$F = (0.5 * F2) + (1 + \beta * vT)$

*g. GRAPH REPRESENTING GENERATIONS VS FITNESS VALUE*

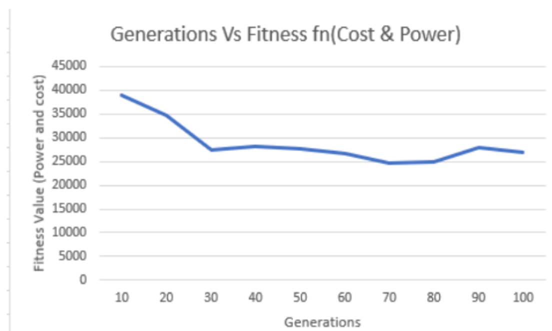


Fig. 5 GENERATIONS VS FITNESS VALUE

The graph represents the number of generations (population) and its impact on fitness value. X-axis in the graph (figure 4.1) represents the Number of generation and y-axis represents the fitness value. The number of VM and cloudlet is set to be constant as 4 VM and 4 Cloudlets. From the graph we could infer as the generation (population) increases the fitness value that is the power and cost is reduced.

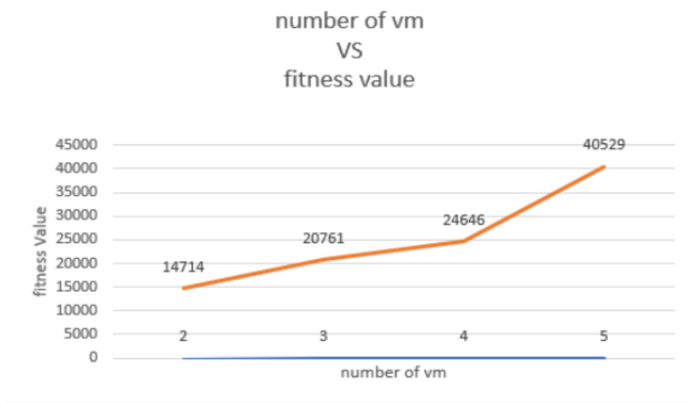


Fig. 6 NUMBER OF VM VS FITNESS VALUE

Here the number of cloudlets is kept to be constant value of 4 and number of generations(population) is kept to be constant of 100, and the system is evaluated varying the number of VM counts, and result was found that the fitness value is increasing in increase of VM count. Figure 4.2 shows the increasing fitness value with respect to increasing VM count.

## REFERENCES


[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, "Title of paper if known," unpublished.

[5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.