Application of Computational Intelligence to Virtualized Data Center Management

Eshrak Assaf Amr Badr Ibrahim Farag

Department of Computer Science, Faculty of Computers and Information,
Cairo University, Cairo, Egypt

ABSTRACT

Server Virtualization is a growing trend in almost all the critical IT infrastructures all over the world. Apart from the cost savings involved with such approach, it is even useful in increasing the infrastructure operational efficiency as it speeds up the operation, enhances the services availability and minimizes the downtimes. But it is actually worthless if the available resources are not well managed, that's why data center management is really crucial to ensure that the virtualization applied is beneficial. In this paper, we propose a new representation for the problem of finding the best allocation for the virtual machines on the physical hosts. We also compare the performance of four types of Genetic algorithms that were used to solve this problem. These are: Steady State (ssGA), Generational (genGA), Cellular (cGA) and distributed (dGA).

General Terms

Genetic Algorithms, Computational Intelligence.

Keywords

Virtualization, Cellular Genetic Algorithms, Distributed Resource Scheduling.

1. INTRODUCTION

Virtualization is a term that is often used these days in a large number of companies all over the world. It is used to abstract applications and their components away from the hardware layer and present a logical view of the resources. The most popular goals of virtualization are: easier management, security, scalability, reliability [1]. Virtualization is most probably used nowadays as it is a popular way of reducing the costs accompanied with a project as it provides a consolidated view of resources at many levels. For example server virtualization maybe used to reduce the number of hardware machines required for a project. This in turn reduces the costs accompanied with cooling, hardware maintenance and datacenter footprint of the servers. This also reduces the administration effort accompanied with the introduced servers [2]. It may also be used to provide a consolidated view of the resources in which server resources like CPU, memory, storage and network are treated as a pool. Each machine is then assigned a customized set of resources according to its processing needs. The resource assignments are highly scalable as they can be easily adjusted by increasing or decreasing them according to the demands of the application deployed. This scalability is very useful as it increases the agility and responsiveness of the organization as it ensures that arising business requirements and market needs are fulfilled in the least time possible.

Virtualization makes it a lot easier to create testing and development environments as it provides a very flexible environment that allows you to make full clones of servers with minimal administrator intervention. Another benefit is that you can use certain procedures to capture the state of the server or virtual machine and return to this state when this is required.

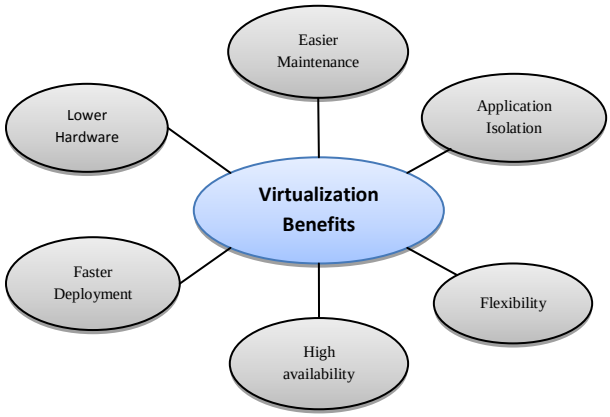

Figure 1. Virtualization Benefits

It is also very useful in production environments as it can be used to respond to emerging business needs in a much faster ways than the old approaches. A pool of resources is always available and if the required resources are well predicted and maintained, the needed servers can be made available upon request. In the era of economic crisis the concept of virtualization has been proven to be very beneficial in saving a lot of costs for companies and generate a better value from their IT investments. It is useful in saving both capital expenditure (CAPEX) and operational expenditure (OPEX). The capital expenditure is reduced as virtualization is used in making a better utilization of the company's infrastructure. The operational expenditure is also reduced as virtualization can be used to reduce the overall number of servers in the infrastructure. It is used to make a better usage of existing underutilized computing resources. This directly reduces the data center costs like power, cooling and datacenter footprint which helps corporates to move to a greener data center. It also provides centralized and easier administration for the servers. Virtualization helps to eliminate the physical hardware dependencies from server operating systems and allows the servers to be moved and recovered in a very efficient way. Ziff-Davis's Research, February 18, 2008, shows some common drivers for virtualization [3]. The scale of the companies that can benefit from virtualization can vary from small enterprises to large scale enterprises, each according to its needs.

There are many levels of virtualization [1, 4] as shown in figure 2, they can be applied individually or combined. The type of virtualization used depends on the purpose, scope and budget.

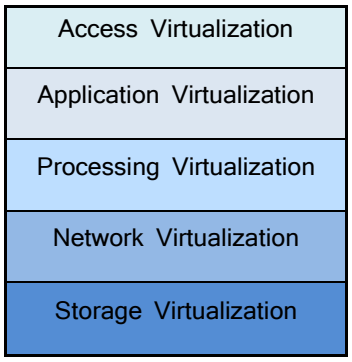

Figure 2. Virtualization Levels

One of the forms of Processing Virtualization is Server virtualization [1]. Server virtualization is a software layer that gives us the ability to expose the server's physical resources like memory, processing power, storage and network bandwidth to make them available to several different virtual machines at the same time. In this way a single physical server can support multiple independent workloads that can be of many different operating systems (Windows, Linux, UNIX, etc.).

Server Virtualization techniques are used for dividing each physical server into many virtual machines where each is run and managed separately from the others. This separation is very useful for the organizations that consider using this approach due to many reasons like security, supportability of the applications and also maximizing the utilization of the physical servers. Just like the physical servers are maintained, virtual machines are created, managed and maintained from a virtual datacenter. In order to be able to deploy, configure and manage virtual machines on a server, a bare metal hypervisor is installed first on the physical hosts. Virtual clusters are then created in the datacenter and the hosts are added to those clusters according to the design that is set. The physical resources of each cluster are treated as a resource pool to be available for all the virtual machines, the main resources to maintain are: memory, CPU, network and storage. Virtual machines are then created and assigned the relevant resources according to the requirements and the recommended sizing of the applications that will be placed on them.

Data center management is used in order to manage the virtual datacenter in an efficient way and to reduce the costs associated with the servers by making an efficient use of the resources available. But in order to achieve this target, special tools are used to monitor the running servers and make on demand-provisioning of the shared resources in real time, while maintaining the operational efficiency and Quality of service (QOS) guarantees of the applications at the least cost possible. When the workloads on the servers are fluctuating, building such tools gets challenging. This is due to the fact that the resources must be ready for unexpected and peak workloads. Dynamic Resource allocation is used in order to create an adaptive, real-time infrastructure where resources are dynamically managed and intelligently allocated to meet the demands of the business. This dynamic resource allocation is applied by using artificial intelligent techniques and methods and is done based on the on-going performance utilization. Different types of evolutionary algorithms were used in calculating the best placement for the virtual machines on the physical hosts. Using performance measures from the hosts, the initial placement can change. This change occurs by moving the virtual machines from host to another without requiring a downtime.

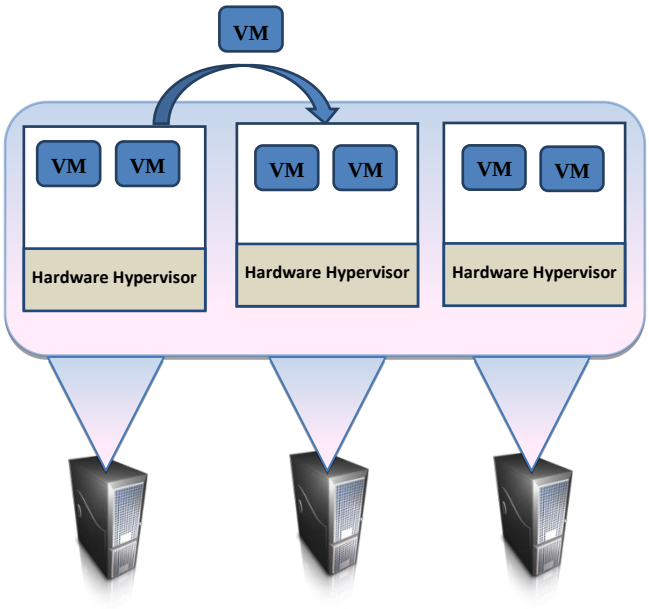

Figure 3. Distributed Resource Scheduling (DRS)

Artificial intelligence techniques are used to determine the best allocation for the virtual machines on the hosts. The best solution should maximize the utilization of the physical resources and minimize the costs associated with moving the virtual machines from one host to another. In this research we used different variants of Genetic algorithms to find a solution to the proposed problem.

2. RELATED WORK

Virtualized data center management is an emerging topic that many researches are currently investigating. The research done in [5] proposed a dynamic processor resource configuration to load balance the virtual environment. It introduced a system called VScheduler. This system is used to adjust the amount of processor resources allocated to virtual machines in order to improve the overall resource utilization of the systems. It proposed a two level configuration scheme: local resource configuration (LRC) for adjusting the resources assigned to each individual virtual machine and global resource configuration (GLC) for the data center or the cluster under consideration. Research results show that the system didn't only help in satisfying the resource demands of the systems but it also ensured the stability of the virtual infrastructure by minimizing the virtual machines migrations that were needed to achieve a proper placement for the virtual machines on the physical hosts.

Another approach was used in [6] where it considered the historical data to find the best solution to the problem using the basic genetic algorithm. This strategy also computed the effect that each solution would have on the system in order to find a solution which also minimizes the number of dynamic

migrations as much as possible. The research in [7] used the same approach to load balance the virtual machines in a large scale cloud computing environment. The model in [8] also used genetic algorithms approach to find a solution to the Virtual Machine allocation problem in a multi-tier distributed environment. The model that was used allows for both quantitative and qualitative resources. It also captured the structure of the distributed infrastructure smoothly and handled multiple SLAs. Unlike other prediction based approaches another model in [9] used the approach of Lyapunov Optimization [10] to get the optimal resource allocation for virtual machines with time-varying workloads and heterogeneous applications. The approach used system queuing information to make online control decisions. In [11] two techniques were proposed to control the CPU resource allocation of the servers. The first approach is a Single Input Single Output (SISO) first-order Kalman filter that dynamically allocates the CPU of individual Virtual Machines. The Second uses a Multiple Input Multiple Output (MIMO) feedback controller which allocates CPU resources dynamically for multi-tier server applications. This approach makes global decisions by coupling the CPU resource usage of all components.

3. THE USED ALGORITHMS

Different variants of Genetic Algorithms were used in this research, these are:

3.1 Steady State Genetic Algorithm (ssGA)

In ssGA there is only one population, the generated individuals replace the older ones in the population. The replacement/deletion strategies determine which individuals are replaced by the newly generated individuals [12]. When only one or two individuals are replaced at each generation then it is called Incremental GA [13]. In the ssGA at each generation two individuals are selected according to the selection criteria. Crossover is then applied to the two individuals and one of them is mutated. The fitness of the resulting individual is evaluated and it replaces the worst individual in the population [14]. The work in [15] compares between the behavior of the generational and the steady state genetic algorithms. The equations in the study show how steady state GA balances its elite selection and at the same time ensures the need for diversity.

3.2 Generational Genetic Algorithm (genGA)

In the genGA at each generation a newly generated offspring is used to form a new population that completely replaces the previous population. This means that individuals can only reproduce with other individuals only if they are from the same generation [16].

3.3 Cellular Evolutionary Algorithms

Cellular Evolutionary Algorithms (cEAs) [14] are one type of Evolutionary algorithms in which the population is represented as a connected graph. Each individual has a certain position in the grid and it communicates with its neighbors. Recombination operators are applied only between each individual and its neighbors. This leads to the separation of the population into islands or subpopulations. The Pseudo code for a simple cEA [17] is shown below:

```
1. proc Increment(cEA)
2. for k=1 to MAX do
3.     for i=1 to Width do
4.         for j=1 to Height do
5.             neig= Get_Neigh(cEA,pos(i,j));
6.             selected_ind=Selection(neigh);
7.             pop= Reproduction(Selected_ind);
8.         end_for;
9.     end_for;
10. cEA= Replacement(cEA,pop);
11. Evaluate_Population(cEA);
12. end_for;
13. end_proc Incerement;
```

Line2: The cEA algorithm runs either until the best solution is found or till it reaches a maximum number of generations (MAX). Lines 3 and 4: The population is represented as grid with Width and Height. Line 5: The neighboring individuals are computed and placed in a list (neigh). Line 6: Perform Selection on the neighboring individuals that were computed in Line 5. Line 7: Apply the Reproduction operators on the selected individuals. Line 10: Replace the old generation by the newly created one. Line 11: Evaluate the fitness of the individuals of the new population. For each individual the surrounding individuals are its neighborhood. Those neighborhoods overlap and the always have the same size and shape. The types of neighborhoods are:

3.3.1 Linear (Ln)

In this type the breeding is done with the neighboring individuals in the north, south, east and west directions only.

3.3.2 Compact (Cn)

The breeding occurs with all the (n-1) nearest individuals.

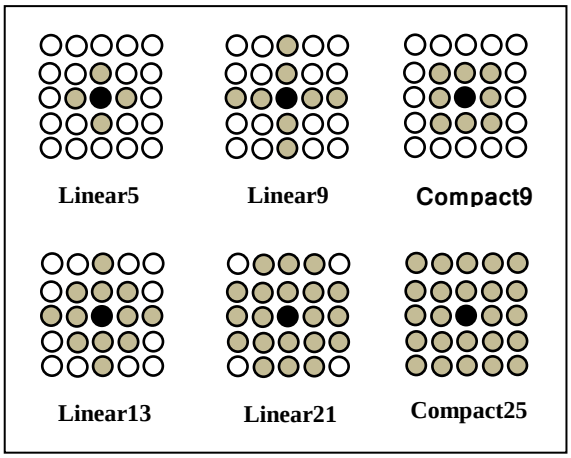

Figure 4. cGA Variations

3.4 Distributed Genetic Algorithm

Distributed (or multiple-deme) GA was proposed as a way of parallelizing the standard GA (sGA). The algorithm is more sophisticated since the population is divided into sub populations where each parent breeds only with the

individuals from its sub-population. Unlike the cellular GA (CGA) that has one string in each sub-population, the dGA has a relatively large sub-population (>>1). Individuals migrate between sub-populations every certain number of generations. Parallel GA's allows for further exploitation of genetic information [18-21].

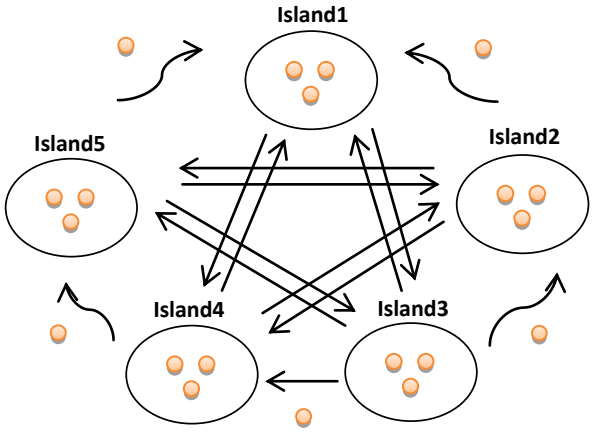

Figure 5. Island Model

4. REPRESENTATION SCHEME

The problem is simply represented so that each individual in the population is a one dimensional array of a possible allocation for the virtual machines on the hosts. The optimal solution should be the one that gets the optimal way to place the virtual machines on the hosts in a way that satisfies the resource requirement of each individual virtual machine and also minimizes the differences of the utilization between the hosts. Virtual machines in a cluster are given numbers from VM number 1 to n, where n is the total number of virtual machines in the cluster. The index of each cell in the individual represents the virtual machine under consideration. The allele values represent the hosts where each virtual machine in the cluster will be placed. H represents the hosts where the host values can be any number from one to k, where k is the total number of hosts in the cluster.

$VM1$	$VM2$	...	VMn
$H(1..k)$	$H(1..k)$		$H(1..k)$

Figure 6. Individual Representation

5. FITNESS EVALUATION

In order to calculate the fitness of each individual in the population, two factors are taken into consideration. The first one is the deviation of the individual from the calculated average load of the hosts. The second factor that we consider is the migration cost. This value represents how different the individual is from the initial allocation of the virtual machines on the hosts.

5.1 Deviation from the Average load

Apart from the fact that the model implemented in [6] uses a different representation scheme for the problem but in this approach, the deviation from the average load is also used to define the fitness of each individual. Assume that the VM is relatively stable in each period of time. The average load of load of the Virtual machine i at time T in n time periods is defined as below:

$$\overline{M(i,T)} = \frac{1}{n}\sum_{x=1}^{n} M_i(x,T) \quad (1)$$

The load of each given host H is equal to the sum of the average utilizations of the virtual machines residing on it at time T.

$$H(i,T) = \sum_{y=1}^{k} \overline{M_i}(y,T) \quad (2)$$

The mean square deviation of a node to the average load is the root of sum of the square differences between the sum of the utilization of each host and the host average load divided by the number of hosts N.

$$\sigma_i(T) = \frac{1}{N}\sqrt{\sum_{i=1}^{N}(\overline{H(i,T)} - H(i,T))^2} \quad (3)$$

Where the average load of each host is defined as the sum of the load of all the hosts in the cluster divided by the number of the hosts at time T as shown below:

$$\overline{H(T)} = \frac{1}{N}\sum_{i=1}^{N} H(i,T) \quad (4)$$

5.2 Migration Cost

In order to reach a solution that has the least migrations possible, the delta between each individual and the initial solution is calculated as in the below equation.

$$\delta = \sum_{i=1}^{N} \frac{1}{N} \times m_i \quad (5)$$

Where N is the size of the solution, for each individual in the population the migration cost δ is the delta between the values of the individual and the initial allocation of the virtual machines on the hosts. In order to calculate the delta, for each index i in the individual, the allele value is compared to that of the initial solution. The value is m_i acts as a penalty value where it is equal to one if the allele at this index is the different from the allele at the same index in the initial allocation of the virtual machines. This means that the virtual machine at this index has been migrated to a different host which implies extra overhead on the environment. This value is set to zero if the allele value is the same as the initial solution as this means that no migration would need to take place.

5.3 The Fitness Function

The best solution is the one that minimizes each host's deviation from the average load in order to meet the required load constraints. The solution needs to ensure at same time using the least number of migrations possible. In order to reach this the mean square deviation and the migration cost needs to be minimized. This makes the fitness function as follows:

$$f_i(S,T) = \frac{1}{\sigma_i \times \delta} \quad (6)$$

The best solution in the population is the one that gets the best fitness value $f_i(S,T)$ after the specified number of iterations.

6. VM SCHEDULING ALGORITHM

6.1 Experiment Setup

This experiment was carried on an existing VMware environment. The virtual datacenter contains a cluster of four

physical servers were used where each server has 8 dual core Intel® Xeon(R) CPU L755, 1.86 GHZ, 128 GB of RAM. Each physical server has an operating system version of VMware ESXi 5 [22]. The hosts have 51 virtual machines with different operating system versions. Each virtual machine is assigned a specific amount of memory, CPU, storage and networking resources according to its needs in the environment. The Fitness Evaluation of the individuals depends on the historical performance data of the virtual machines in the cluster. So, one month CPU utilization for the virtual machines on the host was collected and used for calculating the fitness of the proposed solutions. The scheduling algorithm in this model was implemented by introducing a new problem called "VMBest" to the Java object oriented framework JCell [14, 23]. The framework was further extended by adding the necessary classes in order to find a custom solution to the virtual machine allocation problem.

6.2 Algorithm Analysis

This research provides a new strategy that uses cellular genetic algorithms (cGA), steady-state GAs, generational GAs, and distributed GAs algorithms for load balancing the virtual machines residing in the virtual datacenter. Three different variants of cGAs were considered: Linear5 (cGA-L5), Compact9 (cGA-C9) and Compact13 (cGA-C13). The initial allocation of the virtual machines on the hosts is taken from an already existing VMware environment. One month performance data were collected from all the machines in order to be used for the fitness evaluation. The details for the algorithm that was used in this approach are:

1. Collect one month historical data for the CPU utilization for all the virtual machines on the hosts.

2. Calculate the fitness of the initial solution by calculating the deviation of each host from the average load and summing up these values. The migration cost in this case is equal to zero as no migrations occurred yet.

3. Each algorithm is used to find a solution to the problem in order to find a solution that defines the optimal allocation for the virtual machines on the hosts.

4. A Random population of individuals is generated where each individual represents a proposed solution for allocating the virtual machines on the hosts.

5. The average load that is supposed to be on each host is calculated by calculating the sum of the utilization of all the virtual machines on the hosts and dividing the resulting value by the total number of virtual machines.

6. The fitness values are defined for each individual by calculating the deviation of each host in the solution from the average load. The migration cost is also taken into consideration while evaluating the fitness so as to find a solution with the least migrations possible from the initial allocation. The best solution in is one that minimized the deviation of the CPU utilization of the host.

7. The specific algorithm operators are applied on all the individuals in the population until the maximum number of iterations is reached.

6.3 The Mapping Relationship

In order to analyze the performance of the algorithm a sample of four clustered hosts was used. 51 virtual machines reside on these hosts. Predefined data for one month is used in order to reflect the real resource utilization of the virtual machines. The mapping between each host and its average CPU utilization is shown below in Table 1.

Table 1. The mapping relationship before using the algorithm

Host 1		**Host 2**		**Host 3**		**Host 4**	
VM	**CPU**	**VM**	**CPU**	**VM**	**CPU**	**VM**	**CPU**
V1	1.5	V17	8	V30	22.9	V41	2.2
V2	8.2	V18	2.1	V31	2.1	V42	0.9
V3	9	V19	8.1	V32	2	V43	2
V4	4.1	V20	0.8	V33	4.1	V44	7.3
V5	1.5	V21	1.3	V34	2.1	V45	0.6
V6	5.5	V22	18.8	V35	2.5	V46	2.3
V7	0.5	V23	5	V36	5.9	V47	2.9
V8	11.8	V24	8.6	V37	1	V48	3.4
V9	0.8	V25	7.3	V38	2.8	V49	1.1
V10	12.9	V26	25.6	V39	3.1	V50	3
V11	6.7	V27	3.8	V40	8.1	V51	0.8
V12	10.1	V28	3.9				
V13	6.6	V29	24.4				
V14	0.5						
V15	8.6						
V16	1.4						

Population scale is 10 and the mutation probability applied is 0.2 while the crossover probability is equal to one. The stopping condition is to reach the allowed maximum number of iterations. The average load of each host is= 72.625. The fitness of this initial solution is equal to its deviation from the average load which is = 124.3. The resulting mapping relationship after using the algorithm is shown in Table. 2.

Table 2. The mapping relationship of the best solution

Host 1		Host 2		Host 3		Host 4	
VM	**CPU**	**VM**	**CPU**	**VM**	**CPU**	**VM**	**CPU**
V3	9	V2	8.2	V1	1.5	V7	0.5
V4	4.1	V5	1.5	V10	12.9	V11	6.7
V6	5.5	V8	11.8	V17	8	V26	25.6
V9	0.8	V13	6.6	V18	2.1	V28	3.9
V12	10.1	V15	8.6	V22	18.8	V29	24.4
V14	0.5	V21	1.3	V30	22.9	V35	2.5
V16	1.4	V24	8.6	V33	4.1	V43	2
V19	8.1	V25	7.3	V34	2.1		
V20	0.8	V31	2.1	V42	0.9		
V23	5	V32	2	V44	7.3		
V27	3.8	V37	1	V47	2.9		
V36	5.9	V38	2.8	V51	0.8		
V39	3.1	V46	2.3				
V40	8.1	V48	3.4				
V41	2.2						
V45	0.6						
V49	1.1						
V50	3						

This solution was found by using the Compact13 Cellular GA. The fitness of the final solution is= 25.89, time= 459 ms.

6.4 Performance Evaluation

The below figure shows the average time in milliseconds that was taken by each of the considered algorithms to reach a near optimal solution.

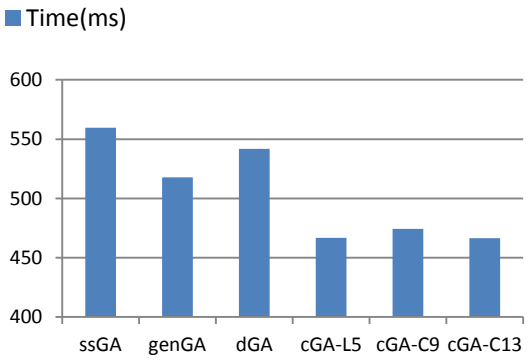

Figure 7. Average time taken by each algorithm to find the best solution

Steady state genetic algorithms took the longest time to reach the optimal solution, while the Cellular genetic algorithms were the fastest to converge to the solution. The Compact13 variation of cellular genetic algorithms was the first to get a near optimal solution. The effect of the population size on the performance of the considered algorithms is shown in Table 3.

Table 1. The effect of the population size on the performance

Pop.	ssGA	genGA	dGA	cGA-L5	cGA-C9	cGA-C13
(10,10)	25.960	25.952	26.004	25.963	25.920	25.946
(20,20)	25.957	25.955	25.999	25.965	25.920	25.937
(30,30)	25.961	25.957	25.994	25.968	25.923	25.943
(40,40)	25.970	25.960	25.994	25.957	25.920	25.948
(50,50)	25.972	25.959	25.999	25.962	25.917	25.950

The below Fig.8 and Fig.9 show that increasing the population size caused a slight degradation in the performance of the SSGA, the genGA and the dGA.

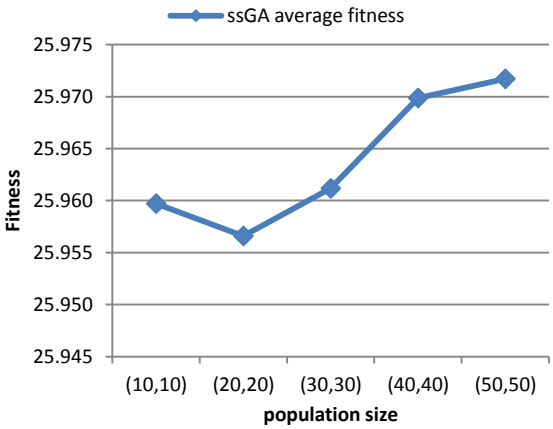

Figure 8. The effect of the population size on the ssGA

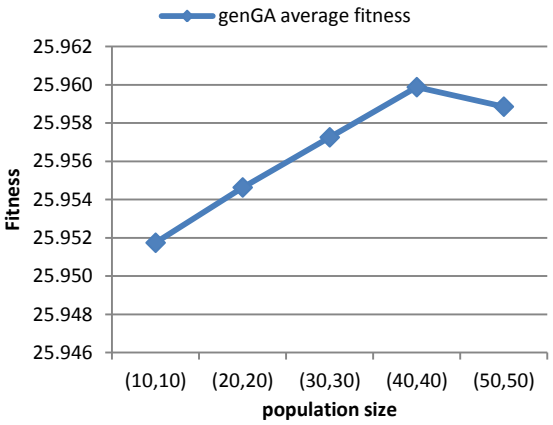

Figure 9. The effect of the population size on the genGA

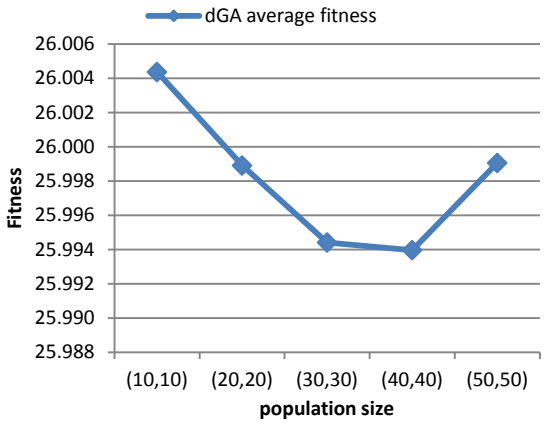

Figure 10. The effect of the population size on the dGA

The cGAs performance didn't vary much by changing the population size. The cGA-L5 and cGA-C9 showed degradation succeeded by a slight improvement while the cGA-C13 showed an improvement succeeded by some degradation in the performance.

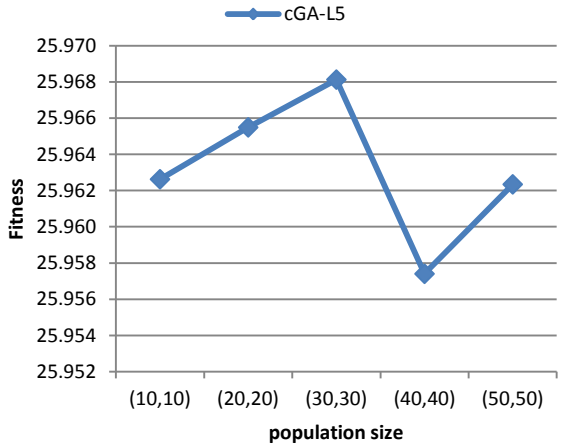

Figure 11. The effect of the population size on the cGA-L5

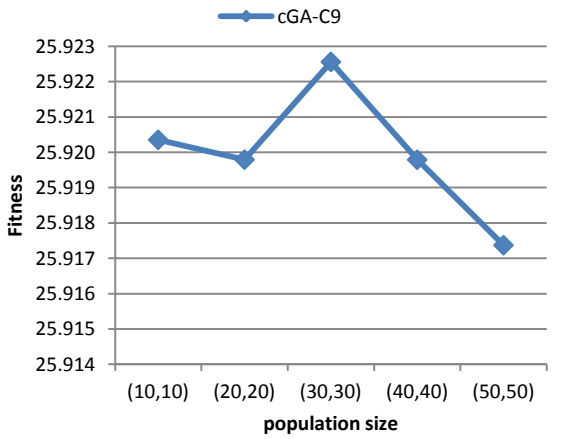

Figure 12. The effect of the population size on the cGA-C9

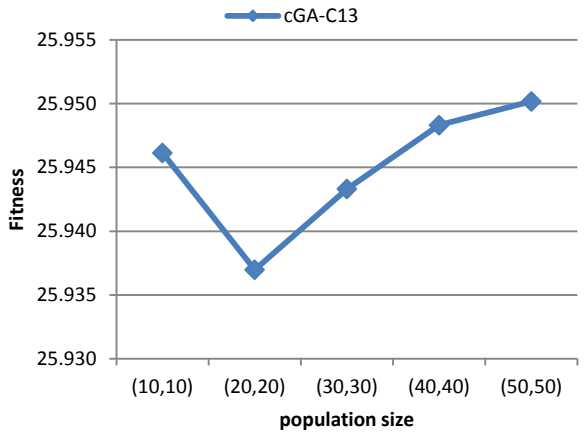

Figure 13. The effect of the population size on the cGA-C13

7. FUTURE WORK

The approach proposed in this research uses historical CPU utilization for the virtual machines to find the best allocation for the virtual machines on the hosts. It only considers the CPU resource utilization of the virtual machines and the migration cost of each solution. The research can be further elaborated by considering other performance metrics like: memory, network and I/O. The algorithms can also be integrated with the various hypervisors available in the market in order to test their applicability in real life environments. This would be useful in proving that the algorithm doesn't violate the applications predefined Service level agreements (SLAs). The table below shows the how the virtual machines were placed on the hosts before using the algorithm and the average CPU utilization of each VM on them.

8. CONCLUSION

This research proposed a new approach for load balancing the machines with their different resource allocation on the hosts of the virtual environment. The goal was to get the best allocation for the specified set of virtual machines on the physical hosts. In order to do that, a population of one dimensional integer individuals was randomly created where each individual represents a solution to the problem. In order to reflect real life implementations, performance data was collected from an existing virtual infrastructure over a one month period of time in order to serve as an input to the algorithm. The fitness of each solution was calculated based on the value of the deviation of the hosts from the average load that is accompanied with the proposed placement. The migration cost for each virtual machine was also taken into consideration while calculating the fitness for the individuals. This is in order to get the best solution with the least number of migrations possible. The research compared the performance of some forms of genetic algorithms in finding the best solution to the problem, these are: cellular GAs, steady-state GAs, generational GAs, and distributed GAs. It also considered three variations of cellular GAs that are determined by the neighborhood shape, these are: Linear5, Compact9, and Compact13.

9. ACKNOWLEDGMENTS

I would like to thank everyone that has directly or indirectly inspired me to pursue this research; my doctors, family, friends and colleagues.

10. REFERENCES

[1] Kusnetzky, D. 2007. Virtualization is More than Virtual Machine Software.

[2] Cerling, T., et al. 2009. Mastering Microsoft Virtualization: Sybex.

[3] Ruest, D. and Ruest, N. 2009. Virtualization: A Beginner's Guide: McGraw-Hill.

[4] Kusnetzky, D. 2011. Virtualization: A Manager's Guide. First Edition ed, United States of America: O'Reilly. 60.

[5] Jin, H., et al. 2011. Dynamic Processor Resource Configuration in Virtualized Environments, in Proceedings of the SCC '11 IEEE International Conference on Services Computing.

[6] Jinhua, H., et al. 2010. A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environment, in Proceedings of the PAAP '10 3rd International Symposium on Parallel Architectures, Algorithms and Programming.

[7] Sawant, S. 2011. A Genetic Algorithm Scheduling Approach for Virtual Machine Resources in a Cloud Computing Environment, in Department of Computer Science, San Jose State University SJSU ScholarWorks.

[8] Campegiani, P. 2009. A Genetic Algorithm to Solve the Virtual Machines Resources Allocation Problem in Multi-tier Distributed Systems. In Proceedings of VPACT'09 Second International Workshop on Virtualization Performances: Analysis, Characterization and Tools.

[9] Urgaonkar, R., et al. 2010. Dynamic Resource Allocation and Power Management in Virtualized Data Centers. In Proceedings of NOMS'10 IEEE Network Operations and Management Symposium

[10] Georgiadis, L., Neely, M., and Tassiulas, L. 2006. Resource allocation and crosslayer control in wireless networks. Foundations and Trends in Networking, Hanover, MA, USA: Now Publishers Inc.

[11] Kalyvianaki, E. and Charalambous, T. 2008. On Dynamic Resource Provisioning for Consolidated Servers in Virtualized Data Centers. In Proceedings of PMCCS'08 the 8th Int. Workshop on Performability Modeling of Computer and Communication Systems.

[12] Vavak, F. and Fogarty, T.C. 1996. A Comparative Study of Steady State and Generational Genetic Algorithms for Use in Nonstationary Environments. Evolutionary Computing (Lecture Notes in Computer Science), Brighton, UK: Springer.

[13] Ma, T. and Abdulhai, B. 2002. Genetic Algorithm-Based Combinatorial Parametric Optimization for the Calibration of Microscopic Traffic Simulation Models. ieee.

[14] Alba, E. and Dorronsoro, B. 2008. Cellular Genetic Algorithms: Springer.

[15] Noever, D. and Baskaran, S. 1992. Steady-state vs. generational genetic algorithms: A comparison of time complexity and convergence properties, Santa Fe Institute.

[16] Sivanandam, S.N. and Deepa, S.N. 2008. Introduction to Genetic Algorithms: Springer.

[17] Dorronosoro, B. 2004. Cellular Evolutionary Algorithms Site. 2004 [cited Access; Available from: http://neo.lcc.uma.es/cEA-web/index.htm.

[18] Yi, W., Liu, Q., and He, Y. 2000. Dynamic Distributed Genetic Algorithms In Proceedings of Congress on Evolutionary Computation. IEEE.

[19] Belding, T.C. 1994. The Distributed Genetic Algorithm revisited. In Proceedings of the 6th International Conference on Genetic Algorithms.

[20] Mcmahon, M.T. 1998. A Distributed Genetic Algorithm With migration for the design of composite laminate structures, in Computer Science and Applications, the Faculty of the Virginia Polytechnic Institute and State University: Blacksburg, Virginia.

[21] Alba, E. and Troya, J.M. 1999. A Survey of Parallel Distributed Genetic Algorithms.

[22] VMware ESXi 5. 2012 [cited Access; Available from: http://www.vmware.com/products/vsphere/esxi-and-esx/index.html.

[23] JCell Framework. [cited Access; Available from: http://jcell.gforge.uni.lu/.