Intelligent Resource Allocation Method Using an Innovative Algorithm for Virtualized Data Center

M. R. Ahamdi, S. Ebrahimi, and F. Ebrahimi

Abstract— **Increasing demand for storage and computation has driven the growth for the environment with tremendous computer resources. To make huge resources available, virtualization technology provide flexible and manageable execution environment that is specialized for variety of applications using shared and manageable resources [1]-[3]. One of the major problems in computing systems and service processing is managing the available resources to support more demands. In this paper we have introduced some advantages of virtualization technology and proposed an intelligent virtual resource allocation method for virtualized data center. Since this problem is NP-hard, the solution has been focused on approximate methods based on genetic algorithm to find an optimum solution in a multi-tier distributed environment. Here, some mathematical analysis and technical evaluation *have* been presented to certify the advantage of the proposed method. Results of our evaluation compare with other methods appear to perform well in finding an approximate solution with less resource consumption ratio.**

Index Terms—**Virtualization, data center, resource allocation, genetic algorithm.**

I. Introduction

Modern data centers are comprised of tens of thousands of dedicated servers, and perform the processing for many applications. An alternate approach is to maintain a pool of resource capacity and allocate virtual resources to different applications. Virtualization is a technology that combines computer resources and provides different operating environments using methodologies like hardware and software partitioning or aggregation, partial or complete machine simulation, emulation, time-sharing, and more. More recently, virtualization at all levels has become important again as a way to improve system security, increase reliability and availability, reduce costs, create better adaptability to workload variations, support easier migration of virtual machines among physical levels has become important again as a way to improve system security, increase reliability and availability, reduce costs, create better adaptability to workload variations, support easier migration of virtual machines among physical machines, and prepare easy coexistence of legacy applications in data center applications. Virtualization provides flexible and manageable execution environments that are specialized for different applications using share resources and delivering expected performance, security and isolation [2]-[4]. One of the important challenges in datacenters is resource allocation and dynamic resource management for virtualized resources. Resource allocation needs to not only guarantee enough virtual resources to meet the performance goals, but also prevents over-provisioning in order to reduce cost and allows concurrent hosting of many applications. Resource control functions are integrated in a data center at two different levels of abstraction: virtual machine and virtual resource pools. The central management is responsible for determining the necessary resources which are needed by each service based on framework of service level agreement. By doing so, the local controller minimizes leasing costs by avoiding over-provisioning for the applications running on the virtual machines. The key to service oriented resource allocation is the ability to efficiently find the minimum amount of resources that an application needs to meet the desired quality constraint [5]-[7]. We can consider the procedure as a classical 0/1 knapsack problem which could be represented as a specific instance of resource allocation. Consider a 0/1 knapsack problem with knapsack capacity, weight item and service quality. This is a multidimensional problem with different weight items and knapsack capacity with different vectors. We can add more dummy knapsacks to have a multiple knapsacks generalization, and these knapsacks can be described via different capacity vectors. As a result, the problem to be tackled is NP-hard. Here an approximate way to solve the formal problem is proposed [8]. To solve this problem, a dynamic resource management approach based on genetic algorithm that enables automatic and adaptive resource provisioning in accordance with quality of service and Service Level Agreements (SLA) specifying dynamic tradeoffs of service quality and cost has been considered. An intelligent technique to characterize the relationship between application workload and available virtual resources has been introduced. A prototype of the proposed resource management system has been deployed on a simulated test bed. The rest of this paper is organized as follows. Section 2 provides optimization methods for data center resources. Section 3 presents resource management methods. Section 4 presents the experimental results. Finally we have concluded the paper in section5.

Manuscript received September 20, 2011; revised October 8, 2011.

M.R. Ahamdi is a member of Information Technology Department in Institute of ICT, Tehran (e-mail: M.ahmadi@itrc.ac.ir).

S. Ebrahimi and F. Ebrahimi is a MSC candidate in Azad University, Research and Science Branch Tehran, (e-mail: s.science.e@gmail.com, h.farzanehebrahimi1982@gmail.com).

II. Data Center Optimization Methods

In this section we introduce three major techniques which are effective for resource optimization in data centers.

A. Parallelism Processing

In Parallelism Processing, separable and independently practicable pieces from an algorithm or even independent practicable parts from a series of attached orders are distributed among the independent processors and they are calculated simultaneously. The Parallel architects in

programs that have dealings with huge mass of data, such as data mining programming. They are categorized in different classes. For example, the architects in "Shared–Memory", all processors use a shared main memory. In "Shared–Disk", the disk memory is the same among the processors. In architect "Shared–Nothing", every processor has its own main memory and related disk. Hence, increasing the resources is more convenient than other methods. The architect "Shared–Something", is a combination of "Shared–Memory" and "Shared – Nothing".

B. Clustering

A cluster includes several machines connected to each other in a local network that perform the processing functions by a timing program and harmonizing among the machines of the network [9]. In this design, the dynamic cluster-wide resource management problem is a constrained optimization problem, with the resource allocations on individual machines as independent variables, and the desired cluster-wide resource allocations as constraints. Periodically collected resource usages serve as further inputs to the problem. It is certified that, in a number of different scenarios, cluster reserves are effective in ensuring performance isolation while enabling high utilization of the server resources.

C. Virtualization

Virtualization is the process of presenting a logical grouping or subset of computing resources so that they can be accessed in ways that give benefits over the original configuration. It is the most significant progress since the microprocessor introduction, in providing information and business secure systems. The idea of virtualization can be used for most of the components in IT infrastructure such as networks, storages, servers, operating systems and applications. In server virtualization, the virtual environment allows for creation of multiple independent occurrence of an operating environment (logical or virtual servers) to run on a single physical server. Virtualization allows physical servers to be carved into multiple virtual machines, and enabling a virtualized data center where applications hosts and managed in their dedicated virtual environment. Storage virtualization implements a virtual layer on top of the physical storages so that all physical devices hid from clients and prepare a pooling environment for using virtual spaces [10].

III. Resource Management Methods

In resource management technique, it's not sufficient to guarantee that virtualized environments always provide the necessary resources for all application programs, but we should decrease the costs by optimum allocating of resources until more application programs can use the available resources. The static methods can't be used to attain this aim because the combination of the workload is changing continuously. For this reason, the system behavior must trace the actual needs at any time [11]. To manage this procedure, a proposed method based on genetic algorithm has been introduced.

A. Genetic Algorithm

In this technique, the genetic algorithm for allocating the virtual resources in multilayer distributed is suggested. This problem is from the ones in the group of NP-hard, for this reason, using the genetic algorithm to find the optimum response is a successful solution. Here, an important point is the understanding concept of the workload to serve the current and future services by identifying it worthily using collection of the virtual resources or pooling system. The noticeable aim in this research is the autonomous recognition of the optimum uses of the existing resources in the pool, when various hosts want to employ different services [12]. To solve this problem, we have proposed the genetic algorithm as a successful solution. Generally, a genetic algorithm is a search technique used in computing to find exact or approximate solutions to search and optimization problems. The GA algorithm repeatedly modifies with genetic operators and seeking for the answer with the best fitness. Since problem to be tackled is NP-hard. Here an approximate way to solve the formal problem has been proposed.

B. Capacity Management Processing

In this section, we deal with the processes that we consider for managing the capacities and also its analogous services. These processes are based on the combination of the sub-processes that take place for the resource pool in various manners. To protect these cases, we need to define the concept and the meaning of the needed capacity. RC is the minimum needed capacity in order to gratify the request of the implemented workload on the resources of the server [10]. To recognize the RC, following conceptions must be considered.

a) The Service of Permitting Control

This service decides whether the pool of services, has the adequate service to host the now workload or not? If so, which workload of the server must be granted? However, the workload extracted from several resources, will be implemented in from of a unique workload so that we may have a limitation for the place of the considered workload. The role of the service of appointing the workload is giving the address to this load.

b) The Service of Workload Placement

This service expresses a categorized technique to identify the requested resource and represent its solution. Fundamentally, each request demands different resources. The service of appointing the workload uses the greedy algorithms to unite the resources. To optimize these inquires, that causes these greedy solution to improve the genetic algorithm is used. In every condition, the algorithm simulated several scenarios. Each scenario determines the number of the workloads on each server. The entire workload request is resulted by the given rate to each server by the help of the different times of the previous workload. This service finds the best place for the workload in the entire server. In the end, it should be stated this service should accept a limitation for the place of the workload that involves the dependence between the workloads. As an example, workloads can or, can aversely, can't too reserve in a physical server together.

c) The Service of Prediction Request of the Workload

This service has a lot of aims.

Implementing a technique for the exploratory model.

- Recognizing the chief changing's in requesting the workload momentarily.
- Protecting the producing of combinative technique that the next requests show each workload and also protecting the similar plans for resources.

Processing the management is its capacity based on the meaning of the capacity management design. The capacity management design includes a memory that some cases include the type of the workload, predicting, the quality of the needed services and giving the workloads to the resources.

C. Algorithm *Functions*

Genetic algorithm is a successful method for optimization problems, although it may not directly lead to the best answer. GA population repeatedly modifies with genetic operators in a search space and seeking for an answer with the best fitness. GA initializes a population to random individuals of [0/1] and over successive generations, the population "evolves" toward an optimal answer. Moreover, this method includes several genetically isolated groups including the CPU, storage, network and I/O which evolves in a parallel model. Individual member from each group collaborate with other members through "representative population" and improves their fitness according to a specific objective function. This method can proceed in a competitive or cooperative model. In cooperative method the groups are encouraged to cooperate with one another by rewarding them based on how well they work together to solve a target problem. In competitive method, the amount of reward resulting from each success is a function of other members in the population who can defeat the same opponent. We have employed both methods in our proposed system [13].

IV. Experimental Result

In this section the proposed algorithm and evaluation scenario with the obtained results have been introduced.

A. *Problem Hypothesis*

There are certain common hypotheses about this research. Any request for using a data center services consist of demands with different resources, and the data center should satisfy those requests. Since the requested jobs are vital for business continuity of the organizations, a small interruption in system may create heavy loss and limitation. Characteristic of the proposed model:

- In this model, the resources have been unified in a pooling system.
- Four kinds of resources have been considered: CPU, I/O, NETWORK, and STOEAGE.
- 100 derivative contents have been used in this model with similar format.
- About 0-20 requests per second have been imported to data center for resources demands.
- Derivative content may share resources commonly or it may respond to request jointly.
- About 5-3 vectors from resources vector have been returned to resources and it has been transmitted to consider virtual container.

As mentioned before each virtual container has been considered as a vector with four resource fields as Fig.1:

$VC_i[1]$	$VC_i[2]$	$VC_i[3]$	$VC_i[4]$
CPU	STORAGE	NETWORK	I/O

Fig.1.Format of resource vectors

Each resource has measured with its standard measurement, for example the size of CPU is accounted in megabyte. Each unit may be defined based on predefined values. For example, one unit of memory can be defined 20 megabyte or 100 megabyte this kind of definition not only unified the values, but also it increases flexibility of model. Because, it is possible those values changes in different data centers, it is necessary that unified the measurement or define it for each data center differently. For example, in one datacenter, each unit of CPU may be regarded about 50 megabyte but in other one it may be regarded about 180 megabyte. It is noted that units are changeable for different data centers. But, in certain data center, size of each unit should be regarded unchangeable. In our proposed model, amount of each resource is assigned with value between 0-30 unite in each virtual container. In our proposed genetic algorithm, chromosomes with length of 400 genes have been used. In total state, 100 virtual containers are defined. Every vector includes four types of resources CPU, STORAGE, NTWORK, I/O. At first, we create primitive generation. In every generation crossing and mutation happen. 60% and 30% respectively and 10% is unchanged. We continue this algorithm for 2000 iteration in order to obtain optimum answer. Fig.2 shows the cost. Rank method was applied for selecting parents in crossing stage. In each repetition of the algorithm the chromosomes having less cost, is selected to virtual container. After end of the process, the chromosome which has less cost is regarded as the best answer.

TABLE I: The Weights Related to CPU

Unit	Weight	Unit	Weight
0≤Unit<1	10	15≤Unit<16	8
1≤Unit<2	9	16≤Unit<17	4
2≤Unit<3	9	17≤Unit<18	2
3≤Unit<4	9	18≤Unit<19	5
4≤Unit<5	5	19≤Unit<20	2
5≤Unit<6	3	20≤Unit<21	4
6≤Unit<7	6	21≤Unit<22	4
7≤Unit<8	7	22≤Unit<23	7
8≤Unit<9	3	23≤Unit<24	4
9≤Unit<10	2	24≤Unit<25	2
10≤Unit<11	7	25≤Unit<26	5
11≤Unit<12	2	26≤Unit<27	9
12≤Unit<13	3	27≤Unit<28	7
13≤Unit<14	4	28≤Unit<29	9
14≤Unit<15	2	29≤Unit<30	10

TABLE II: THE WEIGHTS RELATED TO STORAGE

Unit	Weight	Unit	Weight
0≤Unit<1	8	15≤Unit<16	4
1≤Unit<2	5	16≤Unit<17	7
2≤Unit<3	7	17≤Unit<18	8
3≤Unit<4	6	18≤Unit<19	2
4≤Unit<5	3	19≤Unit<20	2
5≤Unit<6	9	20≤Unit<21	7
6≤Unit<7	10	21≤Unit<22	4
7≤Unit<8	8	22≤Unit<23	5
8≤Unit<9	5	23≤Unit<24	7
9≤Unit<10	6	24≤Unit<25	2
10≤Unit<11	4	25≤Unit<26	3
11≤Unit<12	7	26≤Unit<27	2
12≤Unit<13	8	27≤Unit<28	3
13≤Unit<14	8	28≤Unit<29	6
14≤Unit<15	9	29≤Unit<30	7

TABLE III: THE WEIGHTS RELATED TO NETWORK

Unit	Weight	Unit	Weight
0≤Unit<1	6	15≤Unit<16	5
1≤Unit<2	5	16≤Unit<17	3
2≤Unit<3	8	17≤Unit<18	7
3≤Unit<4	6	18≤Unit<19	9
4≤Unit<5	5	19≤Unit<20	1
5≤Unit<6	3	20≤Unit<21	3
6≤Unit<7	4	21≤Unit<22	6
7≤Unit<8	4	22≤Unit<23	3
8≤Unit<9	8	23≤Unit<24	4
9≤Unit<10	2	24≤Unit<25	2
10≤Unit<11	10	25≤Unit<26	4
11≤Unit<12	4	26≤Unit<27	2
12≤Unit<13	7	27≤Unit<28	3
13≤Unit<14	7	28≤Unit<29	5
14≤Unit<15	7	29≤Unit<30	8

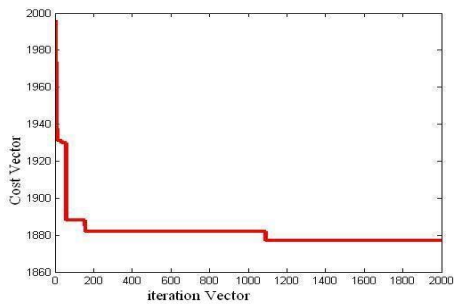

Fig. 2. Cost for each iteration

TABLE IV: THE WEIGHTS RELATED TO I/O

0≤Unit<1	5	15≤Unit<16	7
1≤Unit<2	9	16≤Unit<17	6
2≤Unit<3	9	17≤Unit<18	9
3≤Unit<4	6	18≤Unit<19	4
4≤Unit<5	7	19≤Unit<20	7
5≤Unit<6	6	20≤Unit<21	7
6≤Unit<7	3	21≤Unit<22	6
7≤Unit<8	4	22≤Unit<23	6
8≤Unit<9	3	23≤Unit<24	6
9≤Unit<10	8	24≤Unit<25	8
10≤Unit<11	10	25≤Unit<26	3
11≤Unit<12	6	26≤Unit<27	9
12≤Unit<13	5	27≤Unit<28	7
13≤Unit<14	4	28≤Unit<29	10
14≤Unit<15	3	29≤Unit<30	8

TABLE V IS SHOWN THE CUMULATIVE PROBABILITY DISTRIBUTION FOR CPU RESOURCES.

Unit	cpd	Unit	cpd
0≤Unit<1	*6.13*	*15≤Unit<16*	*54.54*
1≤Unit<2	*11.65*	*16≤Unit<17*	*56.99*
2≤Unit<3	*17.17*	*17≤Unit<18*	*58.21*
3≤Unit<4	*22.69*	*18≤Unit<19*	*61.27*
4≤Unit<5	*25.75*	*19≤Unit<20*	*62.49*
5≤Unit<6	*27.59*	*20≤Unit<21*	*64.94*
6≤Unit<7	*31.27*	*21≤Unit<22*	*67.39*
7≤Unit<8	*35.56*	*22≤Unit<23*	*71.68*
8≤Unit<9	*37.4*	*23≤Unit<24*	*74.13*
9≤Unit<10	*38.62*	*24≤Unit<25*	*75.35*
10≤Unit<11	*42.91*	*25≤Unit<26*	*78.41*
11≤Unit<12	*44.13*	*26≤Unit<27*	*83.91*
12≤Unit<13	*45.97*	*27≤Unit<28*	*88.22*
13≤Unit<14	*48.42*	*28≤Unit<29*	*93.74*
14≤Unit<15	*49.64*	*29≤Unit<30*	*100*

B. Cost *Function*

As mentioned, there is not any unique mathematical rank between requested and granted resources. For this purpose, we should try to create appropriate cost function in order to optimize the answer in every generation of chromosomes. After examine the quantities resources we found out, some of

quantities are more than other quantities, for this reason, we devote certain weight to each unites according to the previous requests for proper cost function. TABLE I and TABLE II and TABLE III and TABLE IV illustrates the weight of CPU, STORAGE, NETWORK and I/O.

C. The Cost Function Formula

Cost related to accounting weight of CPU is obtained by the following function,

$$COSTcpu(j) = \sum_{i=1}^{1/4} weight(4i-3) \quad (1)$$

Cost related to accounting weight of STORAGE in each line is obtained by following the function.

$$COSTcpu(j) = \sum_{i=1}^{1/4} weight(4i-2) \quad (2)$$

Cost related to accounting weight of NETWORK in each line is obtained by the following function.

$$COSTcpu(j) = \sum_{i=1}^{1/4} weight(4i-1) \quad (3)$$

Cost related to accounting weight I/O in each line is obtained by the following function:

$$COSTcpu(j) = \sum_{i=1}^{1/4} weight(4i) \quad (4)$$

For accounting total cost for every line, the following function is used:

$$COSTcpu(j) = \sum_{j=1}^{J} \sum_{i=1}^{1/4} weight(4i-3) + weight(4i-2) + weight(4i-1) weight(4i) \quad (5)$$

After calculation the cost of each line, we arrange available population according to the obtained cost in order to obtain more optimum chromosomes for the next generation.

After obtaining the optimum state running the algorithm, the best possible answers should be examined by different demand to indicate the algorithm efficiency. In other words, in real environment, algorithm should be proceed in such a way that it can select the best virtual container and in the shortest possible time, because in the case of being joint several environment, one request may be answered by two or several derivative contents. Two points should be noted to solve this problem

1) Each incoming request should be satisfied; otherwise, it may cause a big damage to an organization.

2) For the request with minimum resources, the cost should be minimized.

As mentioned before, it is a NP- hard problem and if we consider the first item as the weight and the second item as the value, it can be assume as a classical knapsack problem. A greedy algorithm is necessary to solve this problem and the innovative algorithms provide the best answer in this kind of problems. We have proposed the genetic algorithm to solve the problem.

V. The Algorithm Input

The input of the algorithm is the best solution matrix with size of 4×100 that every line is indicator of virtual container. Percentage of every source value is obtained by the following formula:

$$v_{i,j} = \frac{weigh_{(i,j)}}{\sum_{i=1}^{30}\sum_{J=1}^{4} weight_{(I,J)}} \times 100 \quad (6)$$

Then based the weight value which consider for each unit matrix 1*4 is regarded as an indicator of demand to data center. Percentage value of crossover and mutation are equal to 60% and 30% respectively.

To calculate the cost function, all four resources fields should be considered at the same time. As some possible cases, it may possible that three resources have optimum value, where the fourth one is not acceptable. As an example of this scenario:

CPU= 15	STORAGE=1 3	NETWORK=1 0	I/O= 25

And two virtual container devoted to this demand is:

CPU= 10	STORAGE =8	NETWORK= 3	I/O= 15

CPU= 7	STORAGE= 7	NETWORK= 9	I/O= 7

It will be consider that CPU, STORAGE, NETWORK is specified in optimum state, but the I/O request is not satisfied. We transmit each resource independently to the target function and then examine every state, in order to indicate whether it satisfies all demands or not. So the states that satisfy all demands are supplied in an array. After end of the process, we can find the best optimum state in the array. It should be noted that this procedure is a time consuming process.

To solve this problem, we propose the following procedure. At first, we arrange the sequence of incoming requests to the data center. Then, instead of sending all the requests, we only examine the request with maximum unite. After finding best solution from the maximum one, we examine whether all of demanded resources meet the satisfaction level or not. If the answer is negative we will go to the second row, and continue this procedure to find the best answer. We use cost function similar to cost function in classical knapsack problem. We use this method for calculating the cost of chromosome. To calculate the cost for chromosome "i", we continue the procedure as follows:

At first, we transmit address of blanks field with one to the item array:

```
P = Pop(ii,:);
Item = find(P==1);
```

Then we obtain TW (sum of the weights for selected address in the item) and TV (sum of value that their address is indicated in item) by the following relations:

$$TW = \sum_{k=1}^{size(Item)} \left(Weight_{(Item(k))}\right) \quad (7)$$

$$TV = \sum_{k=1}^{size(Item)} \left(Value_{(Item(k))}\right) \quad (8)$$

TABLE V: cumulative probability distribution for CPU resources cost (i) that is chromosome cost of "i" is obtained by following relations: Cost (i)=1/TV if ((TW>Request(1,andis(1,end))+5)) Cost(i) = Cost(i) * (10+(TW-Request(1,andis(1,end)))); end if (TW==Request(1,andis(1,end)))

```
      Cost(i)=Cost(i)/20;
   elseif
(((TW>Request(1,andis(1,end)))&&(TW<Request(1,andis(1,end))
+1)))
      Cost(i)=Cost(i)/10;
   elseif
(((TW>Request(1,andis(1,end)))&&(TW<Request(1,andis(1,end))
+2)))
      Cost(i)=Cost(i)/8;
   elseif
(((TW>Request(1,andis(1,end)))&&(TW<Request(1,andis(1,end))
+3)))
      Cost(i)=Cost(i)/6;
  end
   if ((TW<Request(1,andis(1,end))))
      Cost(i) = Cost(i) * (10+(Request(1,andis(1,end)))-TW);
```

VI. Evaluation Result

So many requests are entered to a data center in each second and because of jobs vitality, all requests should be responded and all the requested resources should be satisfied. On the other hand so many virtual containers may be rented in each second and after fulfilling their action, they are sent back to the relevant data center.

For investigation of Algorithms, efficiency, we should put them in conditions like the present conditions in data center. For this reason we have considered conditions that is close to the conditions of data center, in simulation of data center, we have simulated environment for 60 seconds and we have assumed that between 0 to 20 requests would enter to data center. In addition in each second (except first and second seconds) between 2 to 30 virtual containers accomplished their action and sent back to data center, that were taken in to service by hosts in past seconds. If the assembly of virtual containers for hosts is less that 30, the produced accidental numbers for returning virtual containers will be 2 of this assembly. Entered requests to data center were considered in simulation environment, and they were according to aforementioned probabilities in last sections. Entered requests with adopting request probability of each resource that mentioned in last section, were selected accidentally.

Fore representation of algorithms' efficiency, one output sample taken from Genetic algorithm indicated in TABLE VI and simulating algorithm of entered environment and algorithm.

In time period of 10 second, total consumption of resources is equal to 8002 unit. Also, numbers of demands imported to data center in this period is equal to 94.Average of resource devoted to each demand is equal to:

$$Average_{10} = \frac{\sum_{i=1}^{10} Resource}{\sum_{i=1}^{10} Request} = \frac{8002}{94} = 85.128 \quad (9)$$

In which "i" is count of time ranged from 1 to 10. If we assume efficiency of the algorithm rather than time in which size of every source per derivative contents is 30 unites, advantage rate is equal to:

$$\text{EfficiencyRate}_{10} = \frac{(120 - Average_{10})}{120} * 100 = 29.060 \quad (10)$$

In 60 second, the average resources assigned to the incoming requests:

$$Average_{60} = \frac{\sum_{i=1}^{60} Resource}{\sum_{i=1}^{60} Request} = \frac{47752}{565} = 84.517 \quad (11)$$

$$\text{EfficiencyRate}_{60} = \frac{(120 - Average_{10})}{120} * 100 = 29.569 \quad (12)$$

The results indicated that the proposed algorithms prevent the additional resource consumption for 20% to 30% rather than usual method. And it can create the best answer compare to the other innovative algorithms.

VII. Related Works

To the best of our knowledge there is no prior work using a genetic algorithm modeling approach to data center resource management. The following briefly summarizes other work with some common elements with this paper's approach.

Rule-based systems: This approach uses a set of event-condition-action rules (defined by system experts) that are triggered when some precondition is satisfied (e.g., when some metrics exceed a predefined threshold). For example, the HP-UX Workload Manager [17] allows the relative CPU utilization of a

resource partition to be controlled within a user specified range, and the approach of Rolia [14] observes resource utilization (consumption) by an application workload and uses some "fixed" threshold to decide whether current allocation is sufficient or not for the workload. With the growing complexity of systems, even experts are finding it difficult to define thresholds and corrective actions for all possible system states.

In [15] the CPU shares are dynamically allocated with the goal to optimize a global utility function, under varying workload levels, and in [16] the proposed architecture involves different Application Environments (AEs), each one comprising several physical machines bounded together.

Each AE serves different classes of transactions, and server could be moved from one AE to another, to optimize a global utility function which is based on the performance metrics of the AEs, like response time and throughput. The proposed solver searches for the optimal number of physical servers for each AE, with a beam search algorithm.

VIII. Conclusion

In this paper, we have studied virtualization technique in data centers and investigated a successful method for managing the resources in virtual environment. We have proposed a dynamic management technique for resource assignment in data center application. At first, we observed that due to complexity of the problem, resource assignment procedure is NP- Hard and we cannot solve the problem by an absolute mathematic formula. We considered an optimization technique as an effective solution for controlling the consuming resources. We have proposed an intelligent algorithm based on genetic functions to solve resource allocation. Some mathematical analyses to calculate the resource values and cost functions have been done. Then the average and efficiency rate of the results have been calculated to demonstrate the advantage of the proposed method. The results certify that proposed method reduce 20-30 percent saving in consuming resources.

TABLE VI: QUANTITIES OF VIRTUAL CONTAINERS

	CPU	STORAGE	NETWORK	I/O		CPU	STORAGE	NETWORK	I/O
VC_1	17	5	20	28	VC_{51}	23	30	13	20
VC_2	3	8	4	26	VC_{52}	27	10	20	6
VC_3	14	6	3	3	VC_{53}	15	15	27	14
VC_4	11	15	14	8	VC_{54}	28	7	14	24
VC_5	9	4	19	6	VC_{55}	5	27	3	5
VC_6	20	11	25	26	VC_{56}	16	11	1	14
VC_7	14	20	29	26	VC_{57}	6	16	15	20
VC_8	15	6	11	5	VC_{58}	13	21	20	7
VC_9	8	14	7	19	VC_{59}	24	1	22	9
VC_{10}	17	28	18	8	VC_{60}	19	6	5	13
VC_{11}	4	8	22	8	VC_{61}	8	20	6	9
VC_{12}	15	19	25	18	VC_{62}	21	14	13	2
VC_{13}	18	22	18	20	VC_{63}	17	5	9	23
VC_{14}	19	3	14	16	VC_{64}	12	5	14	12
VC_{15}	0	19	3	4	VC_{65}	20	13	4	1
VC_{16}	25	6	7	17	VC_{66}	12	2	17	14
VC_{17}	20	18	19	8	VC_{67}	15	16	29	7
VC_{18}	13	22	11	6	VC_{68}	12	25	20	19
VC_{19}	19	5	21	11	VC_{69}	18	24	23	19
VC_{20}	19	19	16	27	VC_{70}	19	8	3	12
VC_{21}	8	26	10	15	VC_{71}	12	14	28	23
VC_{22}	11	16	10	8	VC_{72}	10	19	6	12
VC_{23}	16	19	23	6	VC_{73}	18	29	10	20
VC_{24}	10	12	26	28	VC_{74}	0	13	24	0
VC_{25}	24	2	21	13	VC_{75}	16	24	18	7
VC_{26}	27	12	24	13	VC_{76}	17	7	4	15
VC_{27}	10	26	12	12	VC_{77}	29	22	23	14
VC_{28}	4	2	8	2	VC_{78}	3	26	24	14
VC_{29}	19	17	20	27	VC_{79}	6	18	26	19
VC_{30}	26	6	14	7	VC_{80}	14	23	29	20
VC_{31}	23	4	8	25	VC_{81}	19	5	8	11
VC_{32}	8	25	13	21	VC_{82}	25	15	6	26
VC_{33}	7	10	8	2	VC_{83}	28	24	9	18
VC_{34}	29	25	12	13	VC_{84}	24	3	23	17
VC_{35}	28	11	20	11	VC_{85}	2	26	17	22
VC_{36}	20	8	5	28	VC_{86}	19	25	13	26
VC_{37}	18	20	21	29	VC_{87}	12	16	20	25
VC_{38}	28	14	11	8	VC_{88}	26	21	19	13
VC_{39}	5	12	18	3	VC_{89}	30	25	18	24
VC_{40}	25	8	17	21	VC_{90}	9	12	11	15
VC_{41}	12	14	3	25	VC_{91}	25	30	6	23
VC_{42}	24	7	19	19	VC_{92}	23	10	27	5
VC_{43}	25	4	24	14	VC_{93}	27	13	4	25
VC_{44}	24	8	13	17	VC_{94}	11	20	7	29
VC_{45}	5	25	12	11	VC_{95}	9	9	16	17
VC_{46}	8	6	25	9	VC_{96}	11	3	2	18
VC_{47}	16	20	25	17	VC_{97}	22	24	26	2
VC_{48}	14	11	20	27	VC_{98}	10	25	1	12
VC_{49}	20	27	19	2	VC_{99}	25	3	10	14
VC_{50}	20	24	4	24	VC_{100}	15	17	11	13

REFERENCES

[1] P. X, Z. W, G. J, C. P, "Study on performance management and application behavior in virtualized environment," "*IEEE Network Operations and Management NOMS 2010"*, ISSN1542-1201, pp. 841-844, April 2010.

[2] E. Kalyvianaki, T. Charalambous and S. Hand: "Resource Provisioning for Multi-Tier Virtualized Server Applications," *Computer Measurement Group Journ*al" spring 2010.

[3] R. Perez, L. van Doorn, R. Sailer: "Virtualization and Hardware-Based Security. "IEEE *Security & Privacy Journal*", Vol. 6, No. 5, Sep. 2008.

[4] R. Perez, L. van Doorn, R. Sailer: Virtualization and Hardware-Based Security, "*IEEE Security and Privacy Journal*", Vol. 6, No. 5, Sep. 2008.

[5] A Kochut, K Beaty, "On Strategies for Dynamic Resource Management in Virtualized Server Environments," "*15th International Symposium on Modeling Analysis and Simulation of Computer and Telecommunication System"s*, 2007.

[6] N. Guelfi, E. Astesiano, and G. Reggio, "JGrid: Exploiting Jini for the Development of Grid Applications," University of Nottingham, Jubilee Campus, Wollaton Road, Nottingham, NG8 1BB, 2003, pp. 132 - 142.

[7] M. Glickman, J. Balthrop and S. Forrest, "A Machine Learning Evaluation of an Artificial Immune System," "*Evolutionary Computation Journal*", Vol 13, No 2, 2005, pp. 179-212.

[8] M. A. Potter and K. A. De Jong, "Cooperative Coevolution: An Architecture for Evolving Co-adapted Subcomponents," "*Journal Evolutionary Computation*", Vol. 8, No. 1, MIT Press, 2000, pp. 1-29.

[9] J .A. Hartigan, M. A. Wong, "A K- Means Clustering Algorithm," In "*Applied Statistic"s*, vol. 28, pp 100- 108, 1979.

[10] J. Hoopes, "Virtualization for security: including sandboxing, disaster recovery, high availability," Publisher: L. Colantoni, Published By Syngress Publishing, Inc, 2009.

[11] T. WOOD, "*Improving Data Center Resource Management, Deployment, And Availability with Virtualization*," Graduate School of the University of Massachusetts Amherst in partial fulfillment, 2009.

[12] P. Campegiani, "*A Genetic Algorithm to Solve the Virtual Machines Resources Allocation Problem in Multi-tier Distributed Systems*," Elsevier B.V, 2010.

[13] M. A. Potter and K. A. De Jong, "Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents," "*Journal Evolutionary Computation"*, Vol. 8, Issue 1, pages 1-29, MIT Press, 2000.

[14] J. Rolia "Configuring Workload Manager Control Parameters for Resource Pools," 10th "*IEEE/IFIP Network Operations and Management Symposium"*, 2006.

[15] D. A. Menasce and M. N. Bennani, "Autonomic virtualized environments" In ICAS '06: "*Proceedings of the International Conference on Autonomic and Autonomous Systems, IEEE Computer Societ"*, 2006.

[16] M. N. Bennani and D. A. Menasce, "Resource allocation for autonomic data centers uses analytic performance models," In ICAC '05: Proceedings of the Second International Conference on Automatic Computing, *"IEEE Computer Society"*, 2005.

[17] HP-UX Workload Manager, http://docs.hp.com/en/5990-8153/ch05s12.html

Mohammad Reza Ahmadi received the B.Sc. and M.Sc. degrees in Electrical Engineering and Communication Systems from K.N.T. University of Technology in 1986 and 1990 respectively. He received his Doctor degree in Communication Networks from Tokyo Institute of Technology, Tokyo, in 1997.

Currently he is the project manager and researcher in IT department of Research Institute of ITC, Iran Telecommunication Research Center.

His research interests are data center design and resource optimization, virtualization and cloud computing techniques, network security focus on intrusion detection systems.

Mr. Saeed. Ebrahimi was born in 7/3/1984 at Tehran, Iran. He received the M.Sc. degrees in Tehran Research and Science Branch in 2011. His thesis in M.Sc. was about virtualization of data centers and optimization of resource allocation in it, by extra innovative algorithms. He is working as Associate professor. His research interests include Genetic algorithm, Fuzzy logic and Control, Artificial Intelligence and Neural networks etc.

Mrs. Farzaneh. Ebrahimi was born in 25/2/1982 at Tehran, Iran. She is M.Sc. Software Engineering Student at Shahre Rey payame noor university. Her thesis in M.Sc are about cloud computing.