Multi-objective Virtual Machine Placement for Load Balancing

Feng FANG[1] and Bin-Bin Qu[1,a]

[1]*School of Computer Science & Technology, Huazhong University Of Science And Technology, Wuhan, China*

Abstract. The virtual machine placement is closely related to the efficient and balanced utilization of physical resources. In this paper, the influence of two scenarios about resource utilization on load balancing is analyzed. A multi-objective ant colony optimization algorithm is proposed to solve the virtual machine placement problem, which balances the load among physical machines and the internal load of physical machine simultaneously. The proposed algorithm is compared with two single objective ant colony optimization algorithms, first fit algorithm and greedy algorithm under some instances. The results show that the proposed algorithm can search and find solutions that exhibit good balance among objectives while others cannot. This demonstrates the proposed algorithm can balance the load in the process of mapping virtual machines to physical machines.

1. Introduction

Cloud computing [1] as a new service model can effectively cope with mass data processing and computing needs by integrating Internet resources. Cloud computing [2] can be roughly classified into three types according to the service type: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). The virtualization [3] technology can divide physical resources into isolated virtual machines, which meet the demand of users to improve the utilization of resources and reduce the investment in infrastructure. The isolated virtual machines make it possible for different applications to run on the single server. The high load will affect the performance of upper applications, and the low load will not make full use of the limited resources, so optimal virtual machine placement closely related to the balanced utilization of resources is very important.

A lot of research has been devoted to solve the virtual machine placement problem in a data center. Virtual machine placement is often modeled as bin packing problem [4], and some solutions combine the classic algorithms for bin packing problem, such as First Fit Decreasing (FFD) [5] and Best Fit Decreasing (BFD) [6]. In [7], the relationship among power consumption, resource utilization and performance has been studied. Power consumption optimization algorithm is proposed through modeling as bin packing problem. Beloglazov et al. [8] proposed Modified Best Fit Decreasing (MBFD) algorithm to solve the virtual machine placement problem based on CPU utilization. Virtual machine placement belongs to combinatorial optimization problem, so the improved algorithm can combine with genetic algorithm [9], ant colony algorithm [10] or particle swarm optimization algorithm [11] which is effective to the problem. In [12], a two-level control system is proposed to manage the mappings of workloads to virtual machines and virtual machines to physical resources. An improved multi-objective genetic algorithm is proposed to minimize total resource wastage, power

[a] Corresponding author: bbqu@hust.edu.cn

consumption and thermal dissipation costs. In [13], a prototype virtual machine packing optimization mechanism on Grivon is implemented. Genetic Algorithm (GA) method is employed to avoid SLA (Service level agreement) violation, reduce number of real nodes in use and reduce virtual machine migrations. Feller et al. [14] proposed Energy-Aware ACO-based Workload Consolidation algorithm minimizing the number of physical machines required. In [15], a multi-objective ant colony system algorithm for virtual machine placement in cloud computing is proposed to minimize resource wastage and power consumption. In [16], to reduce energy consumption in cloud data center, an energy efficient virtual machine allocation algorithm is proposed based on a proposed energy efficient multi-resource allocation model and the particle swarm optimization (PSO) algorithm.

Most research on virtual machine placement only considers the initial scenario that the resources of physical machines are all idle. In the real scenario, the load of physical machines being dynamic, the virtual machine would be deployed to the physical machine with low load preferentially. Under the condition of a certain number of virtual machine requests, minimizing the number of physical machines to achieve energy efficient goal will also affect the balanced utilization of resources. This paper will study the problem in the general situation, namely the part of physical machine resources is already used. The designed algorithm in this paper is to make the utilization of physical machine resources as balanced as possible, so more needs are met under the condition of limited resources.

This paper is organized as follows. The second part makes a brief introduction on the ant colony optimization algorithm and multi-objective optimization. The third part describes and formulates the virtual machine placement problem. In the fourth part, multi-objective ant colony algorithm for load balancing is proposed in detail to solve the problem. In the fifth part, the proposed algorithm is compared with two single objective ant colony optimization algorithms, first fit algorithm and greedy algorithm to verify the effectiveness of the algorithm. The sixth part is the conclusion of this paper.

2 Backgrounds

2.1 Ant colony optimization algorithm

Inspired by the collective behavior of real ant colony, Dorigo proposed ant colony optimization algorithm [10] systematically. The mechanism of ant colony optimization algorithm consists of two basic stages: adaptation phase and cooperation stage. In the adaptation phase, each candidate solution according to the accumulated pheromone adjusts the structure itself. On the one hand, the amount of pheromone will be greater if more ants pass through the path, and the probability of the path selected will be larger. On the other hand, the pheromone will evaporate over time. In the collaboration phase, candidate solutions communicate through pheromone to get the desired solution with better performance. The self-organization mechanism of the algorithm does not need to understand every aspect of the problem in detail, so it is effective to solve many combinatorial optimization problems.

2.2 Multi-objective optimization

Many scientific and engineering problems can be modeled as a multi-objective optimization problem [17] which is different from the single objective optimization problem. Performance improvement of one objective may result in performance degradation of other objectives, so it is very difficult or impossible to optimize multiple objectives simultaneously. The feasible solutions of multi-objective optimization problem form a Pareto [18] set. Generally speaking, the multi-objective optimization problem with n decision variables and m objective functions can be expressed as follows.

$$\begin{aligned} &\min \ \boldsymbol{y} = f(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_m(\boldsymbol{x})] \\ &s.t. \quad g_i(\boldsymbol{x}) \le 0, \quad i = 1,2,\cdots,p \\ &\qquad h_j(\boldsymbol{x}) = 0, \quad j = 1,2,\cdots,q \end{aligned} \tag{1}$$

In expression (1), the decision vector is $\boldsymbol{x}=(x_1,x_2,\cdots,x_n)\in X$, and the objective vector is $\boldsymbol{y}=(f_1,f_2,\cdots,f_m)\in Y$. X is the decision space of decision vector, and Y is the objective space of objective vector. $g_i(\boldsymbol{x})\le 0(i=1,2,\cdots,p)$ defines p inequality constraints, and q equality constraints are defined by $h_j(\boldsymbol{x})=0(j=1,2,\cdots,q)$. The following concepts [18, 19] is often used.

Pareto dominance: $\boldsymbol{x}^0$ dominates $\boldsymbol{x}^1$ ($\boldsymbol{x}^0\succ\boldsymbol{x}^1$), if and only if

$$\begin{aligned} &f_i(\boldsymbol{x}^0)\le f_i(\boldsymbol{x}^1), \quad \forall i\in\{1,2,\cdots,m\} \\ &f_j(\boldsymbol{x}^0)< f_j(\boldsymbol{x}^1), \quad \exists j\in\{1,2,\cdots,m\} \end{aligned}.$$

Pareto optimality: $\boldsymbol{x}^0$ is Pareto optimal if and only if $\neg\exists\boldsymbol{x}^1:\boldsymbol{x}^1\succ\boldsymbol{x}^0$.

Pareto optimal set: The set of all Pareto optimal solutions is Pareto set $\mathrm{P}=\{\boldsymbol{x}^0|\neg\exists\boldsymbol{x}^1:\boldsymbol{x}^1\succ\boldsymbol{x}^0\}$.

3 Problem description and formulation

3.1 Problem description

Considering two scenarios about resource utilization, one scenario is that the utilization of one physical machine is far greater than the utilization of another for a long time, and the virtual machine migration [20] is usually used to balance load for such cases. The number of migrations should be reduced as much as possible because of the high costs. Such result is described as the load imbalance among physical machines. Another scenario is that the utilization of one certain resource is much larger than other resources' in a physical machine. This would lead to the fact that the physical machine cannot satisfy the virtual machine resource requirements, resulting in a waste of resources. Such result is described as internal imbalance load in a physical machine.

The virtual machine placement problem is actually to determine the mapping relationship between virtual machines and physical machines, and the mapping relationship between virtual machines and physical machines is multi-to-one. This paper will study that the multiple virtual machines are placed on a certain number of physical machines in the general situation. The goal is to make the load among physical machines and internal load as balanced as possible to achieve efficient and balanced utilization of physical resources so that more needs are met under the condition of limited resources.

3.2 Problem formulation

Virtual machine set is defined as $\mathrm{VM}=\{vm_1,vm_2,\cdots,vm_M\}$. Physical machine set is defined as $\mathrm{PM}=\{pm_1,pm_2,\cdots,pm_N\}$. M is the number of virtual machines and N is the number of physical machines. The types of resources include CPU, memory, storage and bandwidth. The resource request vector of virtual machine vm_i is defined as $R_i=(R_i^1,R_i^2,R_i^3,R_i^4)$. The available resource vector of physical machine pm_j is defined as $A_j=(A_j^1,A_j^2,A_j^3,A_j^4)$. The total resource vector of physical machine pm_j is defined as $S_j=(S_j^1,S_j^2,S_j^3,S_j^4)$. The resource utilization vector of physical machine pm_j is defined as $U_j=(U_j^1,U_j^2,U_j^3,U_j^4)$, where $U_j^d=(S_j^d-A_j^d)\neq S_j^d \quad (d \quad 1,2,3,4)$.

When all virtual machines are placed on a certain number of physical machines, the resource utilization of each physical machine forms a matrix defined as $\mathrm{U}_{N\times 4}=\left(U_1,U_2,\cdots,U_N\right)^{\mathrm{T}}$. The average

utilization of each dimension is defined as the Eq. (2), and d represents the dimension of resource.

$$Avg^d = \frac{1}{N}\sum_{j=1}^{N} U_j^d, \quad (d = 1,2,3,4) \tag{2}$$

The average resource utilization vector of all physical machines is defined as Eq. (3).

$$Avg = (Avg^1, Avg^2, Avg^3, Avg^4) \tag{3}$$

To measure the load balancing degree of physical machines in data center comprehensively, the load among physical machines and the internal load are considered. In order to reflect the degree of load balancing among the physical machines, the **O**uter load **B**alancing **D**egree(OBD) is defined as the average of Euclidean distance between each physical machine resource utilization vector and the average resource utilization vector of all physical machines. Details are shown in Eq. (4).

$$OBD = \frac{1}{N}\sum_{j=1}^{N}\sqrt{\sum_{d=1}^{4}(U_j^d - Avg^d)^2} \tag{4}$$

In order to reflect the load balancing degree of different resources in the physical machine, the **I**nternal load **B**alancing **D**egree(IBD) is defined as the average value of the standard deviation of the resource utilization of each physical machine. Details are shown in Eq. (5).

$$IBD = \frac{1}{N}\sum_{j=1}^{N}\sqrt{\frac{1}{4}\sum_{d=1}^{4}(U_j^d - \frac{1}{4}\sum_{d=1}^{4}U_j^d)^2} \tag{5}$$

Based on the above analysis and parameter definition, the problem can be formulated as follows.

Goals: $\min OBD \; and \; \min IBD$ (6)

Constrains: $\forall pm_j \in \mathrm{PM}, \sum_{i=1}^{M} R_i^d \delta_{ij} \leqslant A_j^d \quad (d = 1,2,3,4)$ (7)

$$\delta_{ij} = \begin{cases} 1, & \text{if } vm_i \text{ is placed on } pm_j \\ 0, & \text{else} \end{cases} \tag{8}$$

$\forall vm_i \in \mathrm{VM}, \sum_{j=1}^{N} \delta_{ij} \quad 1$ (9)

$R_i^d \leq A_j^d \quad (d = 1,2,3,4)$ (10)

Expression (6) is to optimize two objectives simultaneously. Constraint (7) and (8) indicate that for each physical machine, the total resources of virtual machines placed on the physical machine do not exceed the available resources. Constraint (8) and (9) indicate that a virtual machine will eventually be placed on a physical machine. The virtual machine can be placed on the physical machine on condition that the Constraint (10) is satisfied. For each virtual machine, a corresponding set of candidate physical machines is established, and each physical machine in the set satisfies the constraint condition (10). Once a physical machine is selected, the available resource is updated until all virtual machines are placed. A feasible solution of the problem is the mapping of all virtual machines and their corresponding physical machine.

4 Multi-objective ant colony optimization algorithm

4.1 Heuristic function and selection strategy

In the process of virtual machine placement, the heuristic function can help the virtual machine select the appropriate physical machine. Eq. (11) defines the matching distance between the virtual machine and the physical machine, and Eq. (12) defines the heuristic function.

$$d_{ij} = \cos < R_i, A_j > \tag{11}$$

$$\eta_{ij}(t) = (1 - \frac{1}{4}\sum_{d=1}^{4} U_j^d) \cdot d_{ij} \tag{12}$$

There are two reasons about constructing the heuristic function. On the one hand, when the value of d_{ij} is greater, the proportion in all dimensions between R_i and A_j is more similar, so that can make the internal load more balanced, and the cosine of vectorial angle can eliminate the resource dimension. On the other hand, the heuristic function tends to choose the underloaded physical machine, so that can make the load among physical machines more balanced.

For virtual machine vm_i, ant k selects the physical machine pm_j with the probability p_{ij}^k in the set of candidate physical machines $Allowed_i$. p_{ij}^k is defined as Eq. (13).

$$p_{ij}^k(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}}{\sum_{s \in Allowed_i}[\tau_{is}(t)]^{\alpha}[\eta_{is}(t)]^{\beta}}, & j \in Allowed_i \\ 0, & j \notin Allowed_i \end{cases} \tag{13}$$

In Eq. (13), $Allowed_i$ is the set of candidate physical machines of virtual machine vm_i. $\tau_{ij}(t)$ is the amount of pheromone between the virtual machine vm_i and the physical machine pm_j, and $\tau_{ij}(0) = C$ where C is a constant. $\eta_{ij}(t)$ is the heuristic function value between vm_i and pm_j. α is the pheromone heuristic factor, and β is the visibility heuristic factor, which indicate the relative importance of pheromone and heuristic function respectively. The virtual machine vm_i selects the physical machine pm_j by the roulette wheel algorithm. $r \in [0,1)$ is generated randomly, and physical machine pm_j is selected if the cumulative probability $\sum_{s}^{j} p_{is}^k(t), s \in Allowed_i$ is not less than r.

4.2 Maintenance of Pareto optimal set and pheromone updating

At the end of each cycle, the number of feasible solutions obtained is equal to the number of ants at most, and each feasible solution S_i should be judged by the following steps to obtain a temporary Pareto optimal set. For each element in the temporary Pareto optimal set, the same method is used to maintain the global Pareto optimal set. Figure 1 is the main process for maintaining Pareto optimal set.

```
1.    boolean flag=false;
2.    for S_j in P    /* P is a Pareto optimal set */
3.        if S_i dominates S_j
4.            Remove S_j from P;
5.        else if S_j dominates S_i
6.            flag=true;
7.            break;
8.        end if
9.    end for
10.   if(!flag)
11.       add S_i to P;
12.   end if
```

Figure 1. The process for maintaining Pareto optimal set.

$\Delta\tau_{ij}^{k}(t)$ is the increment of pheromone between vm_i and pm_j defined as the Eq. (14) which considers the two objectives IBD and OBD .

$$\Delta\tau_{ij}^{k}(t)=\begin{cases}1/IBD+1/OBD, & \text{if the feasible solution is Pareto optimal}\\ 0, & \text{else}\end{cases} \tag{14}$$

$$\tau_{ij}(t+1)=(1-\rho)\tau_{ij}(t)+\sum_{k=1}^{A}\Delta\tau_{ij}^{k}(t) \tag{15}$$

The pheromone is updated after the completion of one cycle by Eq. (15). In Eq. (15), $\rho\in(0,1)$ is pheromone evaporation coefficient, and A is the total number of ants.

4.3 Deterministic virtual machine placement

For multi-objective optimization problems, the number of solutions is usually more than one. Considering the target weight is not easy to determine, this paper uses the stratified sequencing method to obtain the deterministic solution. The method is to rank all the objectives according to their importance, and then to obtain the set of optimal solutions for the most important objective, and to obtain the set of optimal solutions for the next objective on the basis of the previous set until the last objective. In the process of selecting the deterministic solution, the importance of the objective OBD is higher than that of the objective IBD, so the solution with minimum objective OBD is selected as the deterministic solution of the problem when the stratified sequencing method with two objectives is used. Figure 2 is the algorithm description. The deterministic solution is defined as S.

Input: VM, PM, α , β , ρ , A, G ,C
Output: S

```
for i=1 to G
    for j=1 to A
        for vm in VM
            select pm according to Eq.(13)
            update available resources of pm
            maintain the set of candidate physical machines for virtual machines
        end for
        calculate the value of objective functions
    end for
    maintain global Pareto optimal set
    update the pheromone according to Eq.(14) and Eq.(15)
end for
return S    /* the deterministic solution*/
```

Figure 2. The algorithm for virtual machine placement.

VM is virtual machine set. PM is physical machine set. α is pheromone heuristic factor. β is visibility heuristic factor, ρ is the pheromone evaporation coefficient. A is the number of ants. G is cycle times for algorithm. C is the amount of initial pheromone. In each cycle, the complexity of each ant selects the physical machine for the virtual machine is O(N), and the complexity of maintaining the set of candidate physical machines is O(M), so the complexity of generating a feasible solution is O ($M(M+N)$). Because the number of solutions in Pareto optimal set is uncertain, the complexity of maintaining the Pareto optimal set is not analyzed. The algorithm will generate the number of $G*A$ feasible solutions at most, so the complexity of generating feasible solutions is O($GAM(M+N)$).

5 Experimental results

To verify the effectiveness of the proposed algorithm MOACO (Multi-Objective Optimization Based on Ant Colony Optimization), it is compared with two single objective ant colony optimization algorithms which are SACO-OBD and SACO-IBD, FF(First Fit algorithm) and GS(Greedy Scheduling) under some instances. The paper use java programming language to implement all algorithms. Because of the inherent parallelism of ant colony algorithm, the parallel computing framework Fork/Join in Java7 is used to reduce the running time of the algorithm.

(1) MOACO: The algorithm is a multi-objective ant colony optimization algorithm for OBD and IBD.
(2) SACO-OBD: The algorithm is a single objective ant colony algorithm for objective OBD, which is used to measure the optimization of MOACO for objective OBD.
(3) SACO-IBD: The algorithm is a single objective ant colony algorithm for objective IBD, which is used to measure the optimization of MOACO for objective IBD.
(4) GS: For each virtual machine, the physical machine with the maximum matching distance as Eq. (11) defined is selected.
(5) FF: For each virtual machine request, the physical machine for the first time to satisfy the resource constraints as Constraint (10) defined is selected.

5.1 Experiment parameters

In this paper, the virtual machine template and the physical machine template are set up in advance as shown in Table 1 and Table 2. The physical machines and virtual machines are generated randomly by the templates. The results of five algorithms are compared in the same scale. The parameters of ant colony algorithm are set by several experiments: $\alpha = 1$, $\beta = 5$, $\rho = 0.5$, $A = M$, $G = 50$,C=1.

Table 1. Virtual machine template

CPU/core	Memory/GB	Storage/GB	Bandwidth/Mbps
1	1	100	100
1	2	200	100
2	2	200	100
2	1	300	200
1	2	100	100
2	2	100	200

Table 2. Physical machine template

CPU/ core	Memory /GB	Storage/ GB	Bandwidth/ Mbps	available CPU/core	available Memory/GB	available Storage/GB	Available Bandwidth/Mbps
8	16	750	750	5	10	500	550
4	16	1000	1000	3	12	700	600
8	8	1000	750	4	6	650	400
4	8	750	1000	2	5	450	600

5.2 Comparison of different algorithms

Every test was repeated with 10 runs for each instance and the average result of MOACO is compared with other algorithms. Figure 3 shows the experimental results of two objectives in different scales.

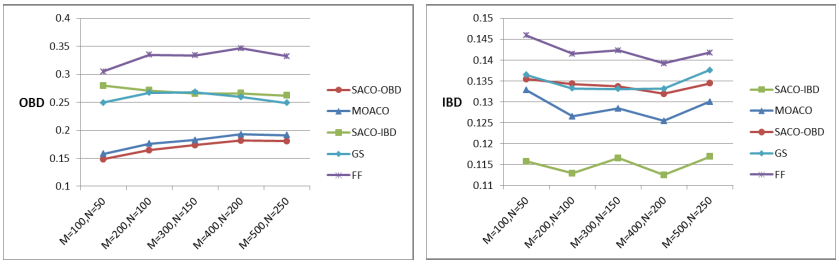

Figure 3. The results of different algorithms in different scales.

Figure 3 indicates that SACO-OBD performs best on the objective OBD, but it performs poor compared with MOACO and SACO-IBD on the objective IBD. SACO-IBD performs best on the objective IBD, but it performs poor compared with MOACO and SACO-OBD on the objective OBD. The experimental results of GS are similar to single objective ant colony algorithms on the objective OBD and IBD respectively, and that indicates the heuristic information is helpful for load balancing. The experimental results of MOACO are obviously better than GS and FF on the objective OBD and IBD. The results show that the proposed algorithm MOACO can search and find solutions that exhibit good balance among objectives while others cannot.

Conclusion

For the problem, this paper analyzes the influence of two scenarios about resource utilization on load balancing. Two objectives are proposed to measure the load balancing comprehensively as possible. A multi-objective ant colony optimization algorithm for virtual machine placement in the general situation is proposed to balance load by optimizing the proposed two objectives. The proposed algorithm is compared with two single objective ant colony optimization algorithms, first fit algorithm and greedy algorithm under some instances. The experimental results show that the algorithm can effectively optimize multiple objectives to achieve the goal of load balancing in different scales.

Acknowledgement

The work presented in this paper is supported by national ministries project which is the research on intelligent scheduling of cloud computing resources based on load balancing.

References

1. S. Yu, C. Wang, K. Ren ,W. Lou, *IEEE INFOCOM*, 1-9(2010)
2. M. Alhamad, T. Dillon, E. Chang, *IEEE International Conference on Digital Ecosystems and Technologies*, 606-610(2010)
3. P. Barham , B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A.Warfield, *ACM SIGOPS Operating Systems Review*, **37(5)**, 164-177(2003)
4. S. Martello, D. Pisinger, D. Vigo, Operations Research, **48(2)**, 256-267(2000)
5. B. S. Baker, Journal of Algorithms, **6(1)**, 49-70(1985)
6. C. Kenyon, *Proceedings of the seventh annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics*, 359-364(1996)
7. S. Srikantaiah, A. Kansal, F. Zhao, *Proceedings of the 2008 conference on Power aware computing and systems*, **10**, 1-5(2008)
8. Beloglazov, J. Abawajy, R. Buyya, Future generation computer systems, 2**8(5)**, 755-768(2012)
9. P. C. Chu, J. E. Beasley, Computers & Operations Research, **24(1)**,17-23(1997)
10. M.Dorigo, M.Birattari, T.Stutzle, IEEE computational intelligence magazine, **1(4)**, 28-39(2006)
11. R. Poli, J. Kennedy, T. Blackwell, Swarm intelligence, **1(1)**, 33-57(2007)
12. J. Xu, J. Fortes, *IEEE/ACM International Conference on Green Computing and Communications & 2010 IEEE/ACM International Conference on Cyber, Physical and Social Computing*, 179–188(2010)
13. H. Nakada, T. Hirofuchi, *International Work Conference on Artificial Neural Networks: Part 2:Distributed Computing, Artificial Intelligence Bioinformatics Soft Computing and Ambient Assisted Living*, 651-654(2009)
14. E. Feller, L. Rilling, C. Morin, *IEEE/ACM International Conference on Grid Computing (GRID)*, 26–33(2011)
15. Y. Gao, H. Guan, Z. Qi, Y. Hou, L. Liu, Journal of Computer and System Sciences, **79(8)**, 1230-1242(2013)
16. A. Xiong, C. Xu , Mathematical Problems in Engineering, **2014**, 816518 (2014)
17. K. Deb, *Search methodologies*, 403-449(Springer US, 2014).
18. P. Ngatchou, A. Zarei, A. El-Sharkawi, *International Conference on Intelligent Systems Application to Power Systems*, 84-91(2005)
19. M. Gong, L. Jiao, D. Yang, W. Ma, Journal of Software,**20**,271-289(2009)
20. J. Zhao, L. Hu, Y. Ding, G. Xu, M. Hu, PloS one, **9(9)**, e108275 (2014)