ISSN 1064-2307, Journal of Computer and Systems Sciences International, 2020, Vol. 59, No. 3, pp. 400–405. © Pleiades Publishing, Ltd., 2020.
Russian Text © The Author(s), 2020, published in Izvestiya Akademii Nauk, Teoriya i Sistemy Upravleniya, 2020, No. 3, pp. 95–100.

Effect of Virtual Machine Deployment Policies on the Efficiency Data Processing Centers

Yu. O. Vasil'eva[a],*, V. A. Kostenko[a],, and A. A. Chupakhin[a],*****

[a]*Moscow State University, Moscow, 119991 Russia*

**e-mail: vasilyeva.jo@gmail.com*

***e-mail: kost@cs.msu.su*

****e-mail: andrewchup@lvk.cs.msu.ru*

Received November 12, 2019; revised November 26, 2019; accepted January 27, 2020

Abstract—The use of deployment policies with the development of virtual machines (VMs) allows improving the reliability of applications, reducing the latency in interactions between VMs, and simplifying the administration at data processing centers. Setting VM deployment policies places additional requirements on physical resources. This can result in reduced physical resource utilization and a lower percentage of deployed requests. The article proposes an algorithm for mapping requests (for creating virtual resources) for data centers' (DCs) physical resources with the optional consideration of VM deployment policies and presents the results of an experimental study of the effect of deployment policies on the load on the physical resources and the percentage of deployed requests.

DOI: 10.1134/S1064230720030168

INTRODUCTION

The article considers infrastructure-as-a-service (IaaS) data centers (DCs). An IaaS DC receives requests to create virtual resources: virtual machines (VMs), virtual data storage systems (VDSSs), and virtual links (VPLs). The requirements for virtual resources are specified in the service level agreement (SLA) for each request element. Virtual resources deployed at a DC require the guaranteed fulfillment of the requirements requested in the SLA. VM deployment policies can be set as such requirements. Setting policies makes it possible to increase the reliability of executing applications on a VM, reduce the latency of VMs' interaction with each other, and simplify DCs' administration. The request also specifies whether the policy is mandatory or optional. A request with a mandatory policy is deployed only when the policy can be implemented. A request with an optional policy is deployed even if it cannot be implemented.

Setting VM deployment policies places additional requirements on the VMs' deployment on physical resources. This can result in a lower efficiency of the DC: reduced physical resource utilization and a lower percentage of deployed requests.

The efficiency of cloud platforms and the use of DCs' physical resources depends largely on the algorithm for mapping the DCs' physical resource requests. Greedy algorithms and algorithms that combine greedy strategies and a limited search [1, 2], heuristic algorithms [3, 4], and modified bin-packing algorithms [5] are used for planning calculations in a DC. Ant algorithms [6], simulated annealing algorithm [7], genetic algorithms [8], and immune algorithms [9] are also applied. In practice, greedy algorithms are mainly used. The reason is that they have low computational complexity and can be easily adjusted to the features of deploying requests for creating virtual resources at specific DCs. The disadvantage of greedy algorithms is that their accuracy can vary greatly on different classes of source data. Algorithms of other classes do not allow their adaptation to the optional consideration of VM deployment policies. Therefore, adaptation is reduced to developing a new algorithm.

The algorithm proposed in this article is based on a combination of greedy strategies and limited search [10]:

The local optimal selection is used at each algorithm step according to the greedy strategy.

At each algorithm step, it is checked that the "greedy selection does not block the way to the best solution."

If the “greedy selection does not block the way to the best solution” test is negative, carry out limited search.

The article proposes an algorithm for mapping requests for DC physical resources with optional consideration of VM deployment policies and provides data from the experimental study on the extent to which the load on physical resources is reduced and the percentage of deployed requests decreases if policies are taken into account.

1. PROBLEM OF MAPPING DC PHYSICAL RESOURCE REQUESTS WITH OPTIONAL POLICY CONSIDERATION

We specify the *model of DC physical resources* by the graph $H = (P \cup M \cup K, L)$, where P is the set of computing nodes, M is the set of data storage, K is the set of DC network switching elements, and L is the set of physical data channels. Vector functions of the scalar argument which specify the characteristics of computing nodes, data storages, switching elements, and data channels are defined on the sets P, M, K, and L, respectively.

The *request for creating virtual resources* is specified by the graph $G = (W \cup S, E)$, where W is the set of VMs used by the applications, S is the set of virtual data storage (storage elements), E is the set of virtual links between VMs and storage elements of the request. Vector functions of the scalar argument which specify the characteristics of the requested SLA virtual element are defined on the sets W, S, and E.

VM deployment policies can be defined in the request. Policies can be of four types:

(1) VM-to-PM affinity, a VM is assigned to a set of physical servers on one of which it should be deployed.

(2) VM-to-PM anti-affinity, a VM is assigned to a set of physical servers on one of which it cannot be deployed.

(3) VM-to-VM affinity, the set of VMs that should be deployed on the same physical server is specified.

(4) VM-to-VM anti-affinity, the set of VMs that should be deployed on different physical servers is specified.

Policies are defined for each request independently. It should be specified for each policy whether the policy is mandatory or optional.

A request with a mandatory policy is either deployed under it or not deployed at all. Optional policies are taken into account whenever possible.

The mapping A is called the *request purpose*:

$$\begin{gathered} A : G \to H \cup \{\varnothing\} = \{W \to P \cup \{\varnothing\}, \\ S \to M \cup \{\varnothing\}, \quad E \to \{K, L\} \cup \{\varnothing\}\}. \end{gathered} \tag{1.1}$$

Let us introduce three types of relationships between request characteristics and the corresponding physical resource characteristics. We denote the request’s characteristic as x and the corresponding physical resource characteristic as y. Let us write these relationships as follows:

(1) The physical resource capacity should not be overloaded: $\sum_{i \in R_j} x_i \le y_i$. Here, R_j is the set of requests assigned to be handled on the physical resource j.

(2) Compatibility of the types of the requested and physical resources: $x = y$.

(3) The physical resource has the required characteristics: $x \le y$.

Mapping (1.1) will be called *correct* if relations (1)–(3) and the defined policies for VMs hold for all physical resources and all their characteristics.

The graph H_{res} is called the residual graph of available resources. It is derived from the graph of physical resources H, for which the values of functions by characteristics which should satisfy the ratio of correctness (1) are redefined. The values of the characteristics are redefined by the following rule: the value of the characteristics of the physical resource is reduced by the sum of the values of the characteristics of the virtual resources assigned to this physical resource.

The *source data* of the problem of assigning resource requests to physical resources are set as follows:

(1) The set of requests (assigned, partially assigned, and pending) $Z = \{Gi\}$ received by the scheduler.

(2) The residual graph of available resources H_{res}.

Required: out of the set of requests received by scheduler Z, deploy the maximum number of requests at the DC in such a manner that the new mapping of physical resource requests is correct.

2. ALGORITHM FOR MAPPING DC PHYSICAL RESOURCE REQUESTS WITH OPTIONAL POLICY CONSIDERATION

The requirements for the algorithm coincide with the requirements formulated in [11], and the requirement of optional consideration of VM deployment policies is added to them.

The algorithm combines greedy strategies and limited search and consists of the following successive steps.

Step 1. If the set $\{G_i\}$ is not empty, select the next request G_i according to the greedy criterion K_G, otherwise the algorithm stops.

Step 1.1. Create the set of VMs from the W request for which VM-to-VM affinity or VM-to-VM anti-affinity policies are specified. Create the set R the elements of which are sets of VMs that are associated with one of the deployment policies.

Step 1.2. If the set R is empty, go to step 2. Otherwise, select any element r from R. Sort the rest of the VMs from the set in nondecreasing order according to the greedy criterion K_V.

Step 1.3. If the set r is empty, remove it from R and go to step 1.2. For the first element from the sorted set, create the set of physical nodes Ph to which this element can be assigned.

Step 1.4. If the set Ph is empty: if it is specified for at least one VM out of r that the policy is mandatory, remove the request from $\{G_i\}$, remove the assigned request elements, and go to step 1; otherwise remove all VMs out of r from the set W, remove the current set from R, and go to step 1.2.

Step 1.5. Select the physical resource ph from the set Ph according to the greedy criterion K_P. For VM-to-VM affinity: if all VMs out of r can be deployed on ph, then assign them to ph. For VM-to-VM anti-affinity: if no VM out of r is assigned to ph, then assign the selected VM to ph. If no element is assigned, remove ph from Ph and go to step 1.4. Go to step 3.

Step 1.6. Remove the current element from r. Go to step 1.3.

Step 2. If the set G_i is empty, remove it from $\{G_i\}$ and go to step 1. Select element e from G_i according to the greedy criterion K_e.

Step 3. Create the set of physical nodes Ph to which this element can be assigned.

Step 3.1. If the set Ph is empty, activate limited search.

Step 3.2. If a limited search was unsuccessful, remove the request from $\{G_i\}$, remove the assigned request elements, and go to step 1; otherwise proceed with the assignment and go to step 1.

Step 3.3. If the set Ph is not empty, select the physical resource ph from the set Ph according to the greedy criterion K_P and proceed with the assignment. Go to step 2.

Limited search is carried out when it is impossible to assign some element N from the request G_i to any physical resource ph. The procedure searches through the set of assigned request elements of such a subset A_d that element N can be assigned after reassigning elements from A_d. The search is limited to number d that specifies the maximum number of physical elements that can be reassigned. Reassignment is the removal of all virtual elements and reassigning them according to the algorithm. Only the subsets of physical resources for which the total number of remaining resources is sufficient to reassign the current virtual element are considered during the search.

3. RESEARCH ON THE EFFECT OF VIRTUAL MACHINE DEPLOYMENT POLICIES ON THE USE OF DC RESOURCES

Criteria for DC resource efficiency:

(1) Load on physical DC resources.

(2) Percentage of deployed requests.

Research objectives are to identify the following points:

(1) Effect of taking into account virtual resource deployment policies on the number of deployed requests and DC load.

(2) Effect of mandatory policies on the number of undeployed requests.

(3) Effect of taking into account policies on the algorithm's execution time.

Experimental method

The research was performed on a computer system with the following characteristics:

(1) Ubuntu 16.04 LTS.

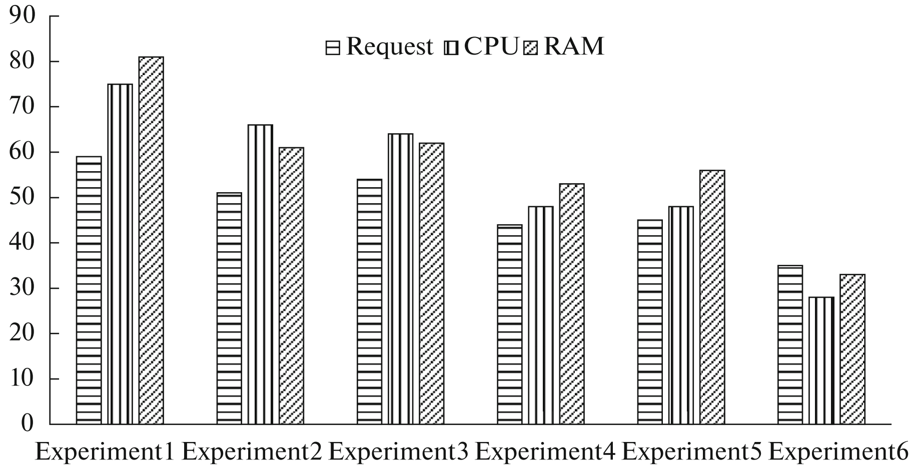

Fig. 1. Effect of taking into account policies on the percentage of deployed requests and DC load.

(2) 8 GB RAM.

(3) Intel(R) Core(TM) i7-6500U CPU 2.50GHz, 2 cores.

Six experiments were conducted as part of the study. The same physical topology was used for all experiments. It was generated with the following parameters.

1. Topology: tree.

2. Number of physical servers: 256.

3. Parameters of the physical nodes:

(a) Number of cores: 16–32,

(b) RAM: 32–64 GB.

4. Bandwidth of physical network channels: 200–300 Mb/s.

Different sets of requests were generated for each experiment. The requests were generated with the following parameters.

1. Topology: tree, star, ring. Selection is made at random.

2. Number of requests: 190–200.

3. Number of VMs in the request: 6–8.

4. Parameters of VMs:

(a) Number of cores: 1–8.

(b) RAM: 2–16 GB.

5. Virtual channel bandwidth: 2–8 Mb/s.

6. Percentage of VMs for which policies are set:

(a) Experiments 1–2: 20%.

(b) Experiments 3–4: 50%.

(c) Experiments 5–6: 80%.

7. Percentage of mandatory policies:

(a) Experiments 1, 3, 5: 20%,

(b) experiments 2, 4, 6: 80%.

Figure 1 shows how taking the request deployment policies into account affects the percentage of deployed requests and the DC load. The percentage of deployed requests of the received (horizontal shading), the percentage of used cores of the requested (vertical shading), and the percentage of used RAM of the requested (diagonal shading) are given for each of the six experiments. The histogram below shows that if the policies are taken into account, it has a negative effect on the number of deployed requests. Moreover, the higher the percentage of requests with policies and the higher the percentage of mandatory policies the lower the percentage of deployed requests and the DC load.

Figure 2 shows how the number of mandatory policies affects the number of undeployed requests. The percentage of requests with mandatory policies of the number of undeployed requests is shown for each of

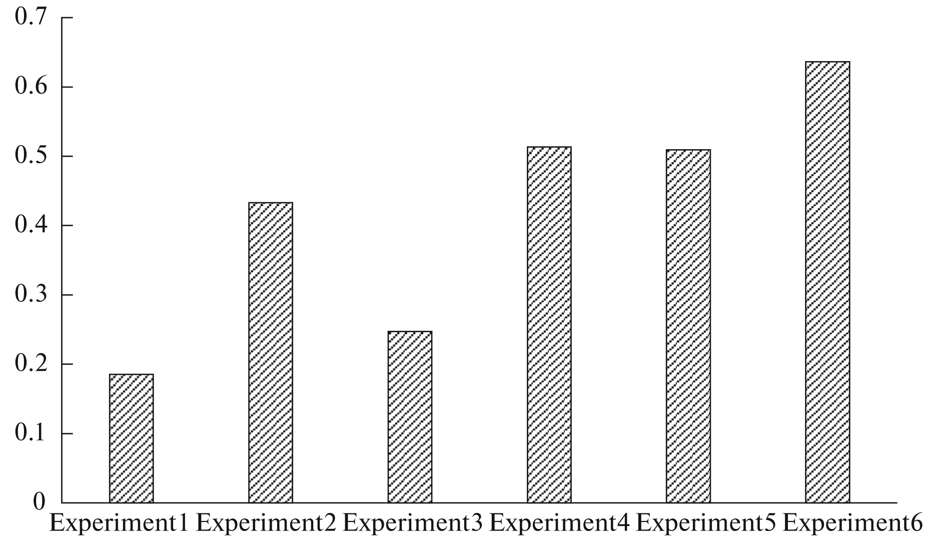

Fig. 2. Effect of mandatory policies on the number of undeployed requests.

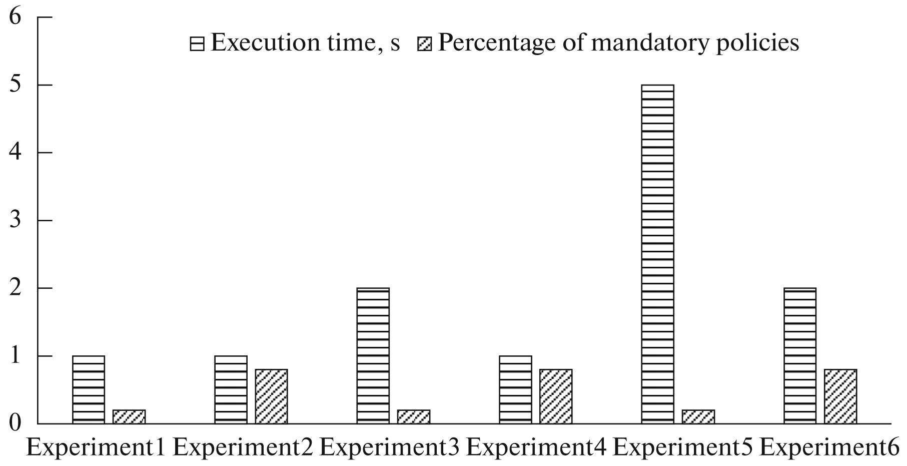

Fig. 3. Effect of taking into account policies on the algorithm execution time.

the six experiments. The histogram above shows that mandatory policies have a rather negative effect on the number of mandatory policies.

Figure 3 shows how the number of mandatory policies affects the algorithm's execution time. The algorithm's execution time in seconds (vertical shading) and the percentage of requests with mandatory policies of requests with a policy (diagonal shading) is shown for each of the six generated tests. From this histogram, we can conclude that if the percentage of optional request deployment policies is high the algorithm's execution time increases.

CONCLUSIONS

The algorithm proposed in this paper allows to optionally take into account the VM deployment policies depending on the indication whether a policy is mandatory or optional in creating a mapping of requests for DCs' physical resources.

In the course of the experimental research of the developed algorithm, it was found that taking the deployment request policies into account has a negative effect on the DC load, the number of deployed requests, and the algorithm's execution time. The DC load and the percentage of deployed requests are negatively affected by the high percentage of mandatory policies, because the number of deployment options of a VM with a policy significantly reduces. An algorithm's execution time is negatively affected by a high percentage of optional policies, because if it is impossible to deploy according to the policy, the algorithm attempts to deploy a VM twice.

Therefore, it should be taken into account that a request with a mandatory policy may not be deployed and may be in the queue of requests waiting to be deployed on DC resources for a longer period of time than without policies or with optional deployment policies.

FUNDING

This work was supported by the Russian Foundation for Basic Research, project no. 19-07-00614.

REFERENCES

1. I. A. Zotov and V. A. Kostenko, "Resource allocation algorithm in data centers with a unified scheduler for different types of resources," J. Comput. Syst. Sci. Int. **54**, 59–68 (2015).
2. P. M. Vdovin and V. A. Kostenko, "Algorithm for resource allocation in data centers with independent schedulers for different types of resources," J. Comput. Syst. Sci. Int. **53**, 854–866 (2014).
3. M. Rabbani, *Resource Management in Virtualized Data Center* (Waterloo, Ontario, Canada, 2014).
4. T. Benson, A. Akella, A. Shaikh, and S. Sahu, *CloudNaaS: A Cloud Networking Platform for Enterprise Applications* (Madison, WI, USA, 2011). http://pages.cs.wisc.edu/~tbenson/papers/CloudNaaS.pdf. Accessed April 11, 2016.
5. A. Ngenzi, R. Selvarani, and S. Nair, *Dynamic Resource Management in Cloud Data Centers for Server Consolidation* (Bangalore, India, 2015).
https://arxiv.org/ftp/arxiv/papers/1505/1505.00577.pdf.
6. A. Ashraf and I. Porres, "Multi-objective dynamic virtual machine consolidation in the cloud using ant colony system," Int. J. Parallel. Emergent Distrib. Syst. **33** (1) (2018).
7. R. Ricci, C. Alfeld, and J. Lepreau, "A solver for the network testbed mapping problem," ACM SIGCOMM Comput. Commun. Rev. **33**, 65–81 (2003).
8. X. Mi, X. Chang, J. Liu, L. Sun, and B. Xing, "Embedding virtual infrastructure based on genetic algorithm," in *Proceedings of the 13th International Conference on Parallel and Distributed Computing, Applications and Technologies, Bejing, 2012,* pp. 239–244.
9. Z. Zhang, S. Su, Y. Lin, X. Cheng, K. Shuang, and P. Xu, "Adaptive multi-objective artificial immune system based virtual network embedding," J. Network Comput. Appl. **53**, 140–155 (2015).
10. V. A. Kostenko, "Combinatorial optimization algorithms combining greedy strategies with a limited search procedure," J. Comput. Syst. Sci. Int. **56**, 218–226 (2017).
11. V. A. Kostenko and A. A. Chupakhin, "Approaches to improving the efficiency of data centers," Programm. Comput. Software **45**, 251–256 (2019).

Translated by O. Pismenov