Server Infrastructure Virtualization for Data Centers

Valeriy Mochalov¹, Gennadiy Linets² and Ilya Palkanov³,⁴

¹Department of infocommunications, North-Caucasus Federal University, 355028, 2 (building 9), Kulakov Avenue, Stavropol, Russia
²Department of infocommunications, North-Caucasus Federal University, 355028, 2 (building 9), Kulakov Avenue, Stavropol, Russia
³Department of infocommunications, North-Caucasus Federal University, 355028, 2 (building 9), Kulakov Avenue, Stavropol, Russia
⁴E-mail: ilya0693@yandex.ru

Abstract. This paper considers cluster systems of data centers containing a definite set of application servers, file servers, data storage systems, and an input-output system that are interconnected by a switching system and communication channels. The goal of research is to increase the efficiency of virtualized cluster systems using a new rational method to distribute virtual machines among the physical elements of data centers. This method is based on an iterative greedy algorithm and a limited enumeration procedure, which improve the performance of a virtualized system due to rational data management. A simulation model for the operation of a virtualized system is developed, and its experimental study is carried out. The approaches presented below determine the required number of cluster nodes in a virtualized system under an unpredictable load rate and an unpredictable level of resources requested by virtual machines.

1. Introduction

Modern data centers involve software-defined networking technologies and server infrastructure virtualization tools such as FirstFit, RandomFit, Min-Min, Max-Min, CPU Load, and Windows Azure, as well as support various IT applications, in order to provide users with numerous utilities and network services in accordance with Service Level Agreement (SLA). Server infrastructure virtualization, aimed at improving the efficiency of data center resource management and based on hypervisor support technologies, distributes virtual machines (VMs) among physical data center servers using algorithms to optimize their load. Modern hypervisors (for example, VMware Infrastructure, VMware ESXi Server, Microsoft Hyper-V) adopt direct application distribution methods, in which each application is provided with a separate VM and redistribution is possible only during implementation, as well as experimental settings selected empirically, which are often based on statistics and resource utilization forecasting [1–3, 7]. If the load rate and volume exceed a given threshold, the server is considered overloaded, and its load is migrated to less loaded hosts. A data center resource scheduler based on the OpenStack platform algorithms, which randomly distribute all incoming applications among the set of available resources, neither properly matches the dynamically changing and unpredictable load rate nor fully reflects the real-time dynamics of all ongoing processes. In these algorithms, control actions are formed simultaneously with the appearance of overload, thereby not solving the optimal allocation problem of physical server resources among virtual machines and network applications in real time [4–6, 9]. Therefore, the SLA requirements are not reliably satisfied. For a higher efficiency of the hardware-software platform of data centers, it is...
necessary to implement optimization algorithms for placing VMs on physical servers in real time based on load balancing.

2. **Server infrastructure virtualization algorithm for Data Centers**

A modern data center includes a server complex, data storage devices (a network storage system), telecommunications equipment that integrates all devices into a single system and communicates with external systems, and a data center management system.

When choosing an optimal distribution of VMs among the physical servers of the hardware-software platform of a data center, it is necessary to consider not only the volume of resources of this equipment and its load, but also the design specifics of data storage systems (DAS, NAS, SAN, CAS, HSM, etc.), which have a significant effect on the parameters, volume and speed of accessing information [10–12, 8]. Here static and dynamic approaches can be used. The static approach is applicable in the case of a sufficiently stable and predictable load rate (volume), when a hypervisor is configured using statistical methods. However, this approach neglects the real-time dynamics of all ongoing processes, detecting an overload in the data center hardware platform as soon as it actually occurs. Some methods for solving such problems were suggested by Gimadi and other researchers; see [13–15]. The corresponding models are based on exact and approximate heuristic solutions of the multidimensional packing problem and linear integer optimization methods.

The dynamic approach is used under the following conditions: unpredictable load rate and volume, a significant increase in the volume of tasks being solved, an increase in the number of users, and other factors that are difficult to formalize. This approach is based on optimization algorithms for the long-term forecasting of resource utilization. It should allocate in real time the available data center resources for each VM without prior planning and distribution, determine the correspondence of each VM to a minimum possible number of specific physical servers, and maintain the performance levels of VMs in accordance with the SLA requirements [16–18]. (For example, the VMware vSphere virtualization platform implements Assignable Hardware, a dynamic load distribution technology, or Live Migration technology). The structural diagram of the management system for the hardware-software platform of a data center is shown in Figure 1.

![Figure 1. Management system for hardware-software platform of data center.](image-url)
At the request of the data center users, this system creates a VM with definite resource parameters. The local manager controls the load of physical servers and the placement of requests for VM resources. The data center monitoring system transfers the information received to the global manager; the latter migrates VMs and manages the redistribution of physical servers. The main problem in implementing the dynamic approach is to determine when data center servers will be overloaded in the future, in order to perform real-time application migration, i.e., to adapt the allocation of physical server resources to VM loads. The long-term forecasting of data center resource utilization can be based on regression analysis (Box–Jenkins model), variational neural networks, machine learning, and decision theory. The suboptimal solutions of short- and long-term forecasts are obtained using various optimization algorithms, heuristic algorithms, genetic algorithms, and neural networks [19].

The static approach to allocate data center resources in accordance with application requirements can be implemented in stages. In the first stage, the required number and type of VMs are determined to implement application requests. For example, a typical range of VMs from Amazon EC2, a leading supplier of virtualization tools, can be considered for the technical characteristics of VMs. In the second stage, the required number of physical servers is determined to provide resources to the specified number of VMs in accordance with the SLA requirements. Data center applications are implemented by forming a set of VMs with definite resource characteristics. The local resource scheduler determines the current load of the data center equipment and distributes VMs to computing devices, provided that they have the required performance, memory capacity, switching system throughput, communication channels, and network interfaces. The necessary parameters of the data center equipment are obtained by combining the characteristics of VMs for each resource type.

The set of software applications \( F = \{ f_1, f_2, \ldots, f_n \} \) is distributed among VMs taking into account the limited resources of data centers. A rational number of VMs involved and the software applications distributed among them are determined under the following constraints:

- An application can be implemented on a single VM only, i.e.,

\[
\sum_{n} a_{jn} \leq 1, \quad (1)
\]

where \( a_{jn} = \begin{cases} 1 & \text{if application is executed on } n \text{ VMs}, \\ 0 & \text{otherwise}, \end{cases} \quad j = 1, n, \quad n = 1, k \)

- The applications distributed to VMs request the total volumes of resources not exceeding the available ones, i.e.,

\[
\sum_{n} a_{jn} C_j \leq C_n, \quad \sum_{n} a_{jn} m_j \leq M_n, \quad (2)
\]

where \( C_n \) and \( M_n \) are the performance and memory capacity of the \( j \) th VM.

- The applications are indivisible for all time intervals \( t_n \) of their execution, i.e.,

\[
\sum_{t=0}^{t_n} a_{jn} = t_n \sum_{n} a_{jn}. \quad (3)
\]

This problem belongs to the class of NP-complete ones. The complexity of its solution using complete enumeration is \( k^n \) operations. In order to reduce computational time, the problem can be solved approximately using the following heuristic greedy algorithm [12].

1) Rearrange all software applications \( j \in r \) in the descending order of their requests for the resources of VMs.

2) Rank all VMs \( k \in K \) by their performance, top to bottom. Performance is measured in EC2 Compute Units (ECU). 1 ECU corresponds to 1-1.2 GHz AMD Opteron or Intel Xeon.

3) For a next application \( j(k) \) on each time interval \( t_n \), select a VM being able to execute this application.
4) If such a VM is not selected, the application will not be executed. Otherwise the required volumes of resources are reserved on the VM. This algorithm has a polynomial complexity and yields an almost optimal distribution; see [15].

A data center contains a definite set of application servers and a definite set of file servers, $S_i$, $i = \{1, L\}$, and $H_j$, $j = \{1, M\}$, respectively. It is required to distribute among them the set of all $VM_k$, $k = \{1, N\}$, under the following constraints:

- Each VM can be placed on a single server only, i.e.,
  \[ \sum_{i=1}^{L} \sum_{j=1}^{M} \sum_{k=1}^{N} x_{ijk} = 1, \]  
  \[ \text{where } x_{ijk} = \begin{cases} 1 & \text{if VM}_k \text{ is placed on } S_i \text{ or } H_j, \\ 0 & \text{otherwise.} \end{cases} \]  

- The memory resources (RAM) requested by VMs from physical servers satisfy the inequality
  \[ \sum_{i=1}^{L} \sum_{j=1}^{M} \sum_{k=1}^{N} RAM_k x_{ijk} < RAM_{ij}. \]  

- The resources utilized by all VMs do not exceed the resources of servers.

- The server performance constraint, which is written as
  \[ \sum_{i=1}^{L} \sum_{j=1}^{M} \sum_{k=1}^{N} CPU_k x_{ijk} < CPU_{ij}. \]  

- The disk storage capacity constraint, which is written as
  \[ \sum_{i=1}^{L} \sum_{j=1}^{M} \sum_{k=1}^{N} S_k x_{ijk} < S_{ij}. \]  

- The performance constraint on the data input-output system, which is written as
  \[ \sum_{i=1}^{L} \sum_{j=1}^{M} \sum_{k=1}^{N} IOPS_k x_{ijk} < IOPS_{ij}. \]  

The Windows Performance Monitor tool platform can be used to estimate the computing load of data center equipment. The optimal ratio of the efficiency of resource utilization and the redundancy of resources for load variations is 70–80%; for details, see [14].

Physical servers and storage systems are interconnected by a switching system. Therefore, when allocating the resources of physical servers, it is necessary to consider the main parameters of this system and virtual communication channels between the VM and the data storage system. The total throughput of all communication channels and switching systems must satisfy the conditions

\[ \sum_{i=1}^{n} \sum_{l=1}^{L} r_{il} < \Pi, \quad \sum_{i=1}^{n} \sum_{l=1}^{L} r_{il} < K \]

where $r_{il}$ is the required throughput of $VM_i$, under the utilization of $l$ virtual channels; $n$ and $K$ are the total throughputs of physical channels and the switching system.

Choose the following criteria of optimality:

- The maximum load of data center equipment,
The maximum performance of the input-output system,

\[ K_2 = \sum_{i=1}^{L} \sum_{j=1}^{M} \sum_{k=1}^{N} IOPS_k x_{ijk} \]  

- The maximum performance of the switching and data transfer system,

\[ K_3 = \sum_{i=1}^{\alpha} \sum_{l=1}^{L} n_i l \]  

The optimal allocation problem of VM resources is NP-hard; therefore, it should be solved using non-polynomial algorithms for offline or online placement [17]. Such approaches guarantee an acceptable allocation accuracy in comparison with the highly complex polynomial algorithms.

The most widespread non-polynomial algorithms for solving such problems are combined in Table 1.

It is known [17] that the next-fit (NF) algorithm ensures the placement accuracy is 2 times different from the optimal one, respectively, the first-fit (FF) and best-fit (BF) algorithms are 1.7 times, the next-fit decreasing (NFD) algorithm is 1.691 times, the first-fit decreasing (FFD) and best-fit decreasing (BFD) algorithms are 1.222 times.

| Algorithm | NF | FF | BF | NFD | FFD | BFD |
|-----------|----|----|----|-----|-----|-----|
| Accuracy  | 2  | 1.7| 1.7| 1.691| 1.222| 1.222|

For example, the first-fit decreasing algorithm (FFD) [4] rearranges all VMs in the descending order of requested resources and sequentially distributes them to the least busy servers.

This algorithm solves the placement problem with an accuracy of no more than 22% different from the optimal one. None of the known approximate algorithms will guarantee a significantly better result [4].

Such algorithms are reasonable if the load of data center resources does not exceed 50%. In the case of higher loads, these algorithms become unacceptable, and heuristic algorithms should be developed. The algorithm proposed in this paper performs the sequential processing of all requests for VMs placement. They are ranked in the descending order of the requested resources. The external data center scheduler allocates the maximum volume of all resources (CPU, RAM, disk, IOPS, etc.) to each VM. A physical resource with a minimum residual sum of its parameters is determined, and a VM is placed on it using the following scheme:

1) All VMs [4] are ranked by the value of the characteristic

\[ V_i = \sum_{j=1}^{K} C_j V'_j, \]

where \( V'_j \) is the request of VM\(_{i}^{j}\), for resource \( j, j = 1,K \) and \( C_j \) denotes the significance of this resource.

2) In accordance with the initial scheme, a VM with the highest resource requirements is selected and placed on a physical element with a minimum resource. Here a most reasonable method is to use an iterative algorithm based on dynamic programming.

For example, the iterative greedy algorithm runs as follows [14]. As the input processes \( a_i \) of the set \( S = \{ a_1, a_2, ..., a_n \} \) the final times \( f_i \) of their implementation are used. The sequentially selected processes are combined by the Greedy Activity Selector procedure \( (s, f) \) into a set \( A \) in which \( f_i \) is
the maximum final time of all processes: \( f_i = \max \{ f_k : a_k \in A \} \). Greedy Activity Selector \((s, f)\) can be described by the code

```plaintext
n ← length[s];
A ← \{a_i\};
i ← 1;
for m ← 2 to n;
do if S_m ≥ f_i;
then A ← A \cup \{a_m\};
i ← m;
return A.
```

3) The external data center scheduler estimates the performance of the VM placed to the server.

4) If the VM’s performance is below the specified level, the external scheduler moves it to a physical element with a higher level of resources and goes back to Step 3.

5) If the VM’s performance is higher than the specified level, the external scheduler moves it to a physical element with a lower level of resources and goes back to Step 3.

6) If it is impossible to distribute VM, the transition to the limited enumeration procedure is performed. The enumeration depth, which determines the maximum number of servers for which the distribution is assigned, must guarantee the required balance between the quality of service and the VM distribution time (find an acceptable solution in an allowed time). The quality of service can be estimated by the probability of blocking for service requests; otherwise, by the probability of no admissible servers per unit time. This can be done using the Erlang formula

\[
P_\delta = \frac{a^n/n!}{\sum_{k=0}^{n} a^k/k!},
\]

where \( P_\delta \) denotes the probability of blocking; \( n \) is the number of servers; \( a \) is the rate of requests.

This probability can be calculated iteratively [8]:

\[
P_\delta = B(n, a) = \frac{B(n-1, a)}{B(n-a)+n/a},
\]

where \( n=1,2,..., \) and \( B(0, a) = 1 \).

In the case of large values of \( n \) and \( a \), the computing time of \( P_\delta \), can be reduced by choosing an required deviation \( P_\delta (i) - P_\delta (i-1) < \delta \) and terminating the enumeration procedure as soon as the latter value is achieved [10]. Here \( \delta \) specifies the accuracy of calculations.

7) If the resource is not found, then the next VM is placed, the resource requested by the previous VM is decreased, and it is included in the common ordered queue.

8) If the resource is found, then the VM is eliminated from the queue.

9) The time-optimal placement is selected under the existing constraints.

3. Numerical experiment

The new virtualization method was simulated for a cluster based on the Huawei 2488 platform, Intel (R) Xeon (R) Gold 6154 processor, 512 Gb RAM, containing 10 hosts and 40 VMs. The volume of requested VM resources was uniformly distributed in the range of 20-80% of the server resources. The ratio of CPU time and RAM was uniformly distributed. The switching system throughput was limited to 1 Gb/s. The duration of the VM migration was defined by the Gaussian distribution law.
Tables 2–4 show the simulation results of placing VMs on this computing cluster in a data center.

**Table 2. Percentage of used hosts.**

| Numbers of hosts | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 |
|------------------|----|----|----|----|----|----|----|----|----|----|
| Used hosts before placement, % | 85 | 63 | 100| 81 | 42 | 37 | 34 | 45 | 30 | 40 |
| Used hosts after placement, %   | 73 | 75 | 70 | 75 | -  | -  | -  | -  | -  | -  |

**Table 3. Percentage of used RAM.**

| Numbers of hosts | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 |
|------------------|----|----|----|----|----|----|----|----|----|----|
| Used hosts before placement, % | 60 | 65 | 70 | 45 | 27 | 25 | 20 | 32 | 40 | 43 |
| Used hosts after placement, %   | 80 | 65 | 40 | 70 | -  | -  | -  | -  | -  | -  |

**Table 4. Running time of algorithm.**

| Number of VMs | 5  | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 50 |
|---------------|----|----|----|----|----|----|----|----|----|----|
| Placement time, s | 12 | 19 | 25 | 31 | 50 | 75 | 85 | 90 | -  | -  |

In accordance with experimental evidence, the algorithm proposed in this paper guarantees an acceptable resource allocation quality if the number of VMs does not exceed 20. With an increase in the number of VMs, the placement time grows exponentially. This algorithm reduces the number of data center hosts used by 60%, provided that the number of VMs being placed is not greater than 20.

4. Conclusions

This paper has considered a topical problem, the design of a mechanism for distributing virtual machines among physical elements of a data center. A method for solving this problem has been proposed, which combines a heuristic greedy algorithm with a limited enumeration procedure. As it has been demonstrated, this problem can be solved in two stages as follows: apply an algorithm for the optimal allocation of software applications among virtual machines, and to place virtual machines on the physical nodes of a data center cluster, taking into account the statistics of requests. A simulation model for the operation of the virtualization management system of a software-defined network has been developed (certificate of state registration of a computer program No. 2019664647 dated November 11, 2019). This model has been studied experimentally to obtain and analyze performance characteristics such as the percentage of host resources used before and after placement, as well as the predicted running time of the distribution algorithm. The method described in this paper can be used for distributing software applications among the VMs of a corporate data center in a reasonable way, for selecting an appropriate set of virtual machines, and for choosing their rational placement on physical servers in the data center.

5. References

[1] Hewitt C 2010 ORGs for scalable, robust, privacy-friendly client cloud computing *The Private Cloud Report* iss 3 (Berkshire) 42 p

[2] Foster I and et al 2008 Cloud Computing and Grid Computing 360-Degree Compared *Grid Computing Environments Workshop* (Chicago) pp 1–10

[3] Timm S and et al 2015 Cloud services for the Fermilab Scientific Stakeholders *J Phys Conf Ser* vol 664 no 2

[4] Baranov A V, Balashov N A, Kutovskiy N A and Semenov R N 2016 JINR cloud infrastructure evolution *Physics of Particles and Nuclei Letters* vol 13 no 5 pp 672–675
[5] Feller E, Rilling L and Morin C 2011 Energy-aware ant colony based workload placement in clouds Proceedings of the 12th IEEE/ACM International Conference on Grid Computing (Lyon, France)

[6] Farahnakian F, Liljeberg P and Plosila J 2013 LiRCUP: Linear regression based CPU usage prediction algorithm for live migration of virtual machines in data centers 39th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA) pp 357–364

[7] Beloglazov A and Buyya R 2012 Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers Concurrency and Computation: Practice and Experience (CCPE) vol 24 no 13 pp 1397–1420

[8] Mastroianni C, Meo M and Papuzzo G 2013 Probabilistic consolidation of virtual machines in self-organizing cloud data centers IEEE Transaction of Cloud Computing vol 1 pp 215–228

[9] Mosa A and Paton N W 2016 Optimizing virtual machine placement for energy and SLA in clouds using utility functions Journal of Cloud Computing: Advances, Systems and Applications vol 5

[10] Monil M A H and Rahman R M 2016 VM consolidation approach based on heuristics, fuzzy logic, and migration control Journal of Cloud Computing: Advances, Systems and Applications vol 5

[11] Guenter B, Jain N and Williams C 2011 Managing cost, performance, and reliability tradeoffs for energy-aware server provisioning Proc of the 30st Annual IEEE Intl Conf on Computer Communications (INFOCOM) pp 1332–1340

[12] Balashov N, Baranov A and Korenkov V 2016 Optimization of over-provisioned clouds Physics of Particles and Nuclei Letters vol 13 no 5 pp 609–612

[13] McNab A, Stagni F and Luzzi C 2015 LHCb experience with running jobs in virtual machines J Phys: Conf Ser vol 664

[14] Computing Center of the Institute of High Energy Physics (IHEP-CC) “VCondor – virtual computing resource pool manager based on HTCondor” 2020 Available at: https://githubcom/hep-gnu/VCondor

[15] McNab A, Love P and MacMahon E 2015 Managing virtual machines with Vac and Vcycle J Phys: Conf Ser vol 664

[16] Feller E, Rilling L and Morin C 2012 Snooze: A scalable and autonomic virtual machine management framework for private Clouds Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid) pp 482–489

[17] Beloglazov A and Buyya R 2015 OpenStack Neat: A framework for dynamic and energy-efficient consolidation of virtual machines in OpenStack clouds Concurrency and Computation: Practice and Experience (CCPE) vol 27 no 5 pp 1310–1333

[18] Orgerie A-C and Lefèver L 2009 When clouds become green: the Green Open Cloud Architecture International Conference on Parallel Computing (ParCo) pp 228–237

[19] Ward J S and Barker A 2014 Observing the clouds: a survey and taxonomy of cloud monitoring Journal of Cloud Computing: Advances, Systems and Applications vol 3