Leveraging Renewable Energies in Distributed Private Clouds

Christian Pape[1,a], Sebastian Rieger[1] and Harald Richter[2]

[1]*Department of Applied Computer Science, Fulda University of Applied Sciences, Fulda, Germany*
[2]*Department of Informatics, Division Computer Engineering, Clausthal University of Technology, Germany*

Abstract. The vast and unstoppable rise of virtualization technologies and the related hardware abstraction in the last years established the foundation for new cloud-based infrastructures and new scalable and elastic services. This new paradigm has already found its way in modern data centers and their infrastructures. A positive side effect of these technologies is the transparency of the execution of workloads in a location-independent and hardware-independent manner. For instance, due to higher utilization of underlying hardware thanks to the consolidation of virtual resources or by moving virtual resources to sites with lower energy prices or more available renewable energy resources, data centers can counteract their economic and ecological downsides resulting from their steadily increasing energy demand. This paper introduces a vector-based algorithm for the placement of virtual machines in distributed private cloud environments. After outlining the basic operation of our approach, we provide a formal definition as well as an outlook for further research.

1 Introduction

Cloud infrastructures and the underlying virtualization technologies are building the foundation of modern data centers. These paradigms also offer potential for reducing the energy consumption of data centers representing most of their ongoing operational costs. In this paper, we seize an opportunity to increase the energy-efficiency of data center operation by introducing a vector-based algorithm to support virtual machine placement decisions. After a brief introduction of related work in Section 2, we outline the basic operation followed by the formal definition of our algorithm. Further, we evaluate our approach and discuss the impact of migration costs in Section 4. Finally, we give an outlook on future work in Section 5.

2 Related work

The placement of virtual resources in modern cloud-based environments is subject of current research. A project called CÆSARA [1], introduced an algorithm for the energy efficient placement of virtual machines by estimating the server's energy consumption based on the running virtual machines' characteristics. The cost of virtual machine migration operations and their energy consumption itemized by the different type of data center equipment, is described in [2]. In [3], a utility is described that allows to distribute virtual machines considering the migration cost and also a basic analysis of migration cost and the impact of live-migration on the running application is outlined. A distributed algorithm for placing virtual machines in large cloud environments is outlined in [4]. The basic approach is that each server knows the CPU load of the other physical servers and each server tries to comply with an upper and lower threshold for the CPU load and initiates the migration of virtual machines when these thresholds are violated. Also, the underlying mathematical challenges like the set-partitioning [5], [6] and bin-packing [7]-[9] problems are still subject of current scientific studies and research. There also exist vector-based approaches for VM placement, but these mostly focus on intra data center placement of VMs on physical machines (PM). In [10] a vector-based methodology to model VM resources and to place VMs on PMs is introduced. Another vector-based constraint programming approach is described in [11]. A routing centric placement algorithm is introduced in [12] and describes a combined optimization approach for data center traffic and VM placement. Furthermore, the communication demand is focused in [13] for VM placement. In contrast to the listed publications, our approach also incorporates the use of renewable energies and their fluctuating characteristics concerning availability and pricing.

3 Inter-DC energy-aware placement of virtual machines

This section outlines the basic idea and functionality of our algorithm. In this context the iterative approach of the algorithm causes its complexity to be relatively low compared to bin-packing or set-partitioning algorithms. Thus, each single iteration will lead to a better overall topology. The algorithm is run continuously, though in

[a] Corresponding author: Christian.Pape@informatik.hs-fulda.de

reality a delay or pause between individual runs might be reasonable, especially, if a predefined threshold of migrations across multiple runs has not been reached. This limits the resources and management traffic generated by the execution of the algorithm. The algorithm consecutively considers the optimal placement for each virtual machine with respect to its network flows, corresponding relationships among them and connections to external clients.

3.1 Energy-efficient placement considering renewable energies

The algorithm used in this paper is a vector-based approach to optimize scheduling and placement decisions in private clouds. In this context the dimensions of the used vector space specify the characteristics of virtual resources as entities in distributed data centers.

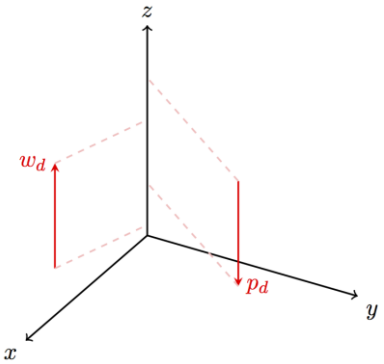

Figure 1. Example of the 3-dimensional vector space V.

To illustrate the basic operation of the algorithm, we will just use an example with three dimensions here. The dimension x and y depict the geographical location of the entity and the dimension z the availability of renewable energy sources for this data center site or, more generally speaking, location.

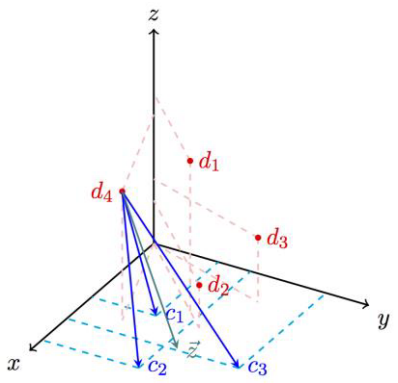

Figure 2. Computation of the destination vector $\vec{z}$.

The example shown in Figure 1 shows the data center w_d representing a site with available wind energy and p_d depicting a site with available photovoltaic energy. The shown arrows indicate the modification of the positional vector of the data centers w_d and p_d over the time span from summer to autumn. Figure 2 illustrates the computation of a destination vector. Thereby d_1, d_2, d_3 and d_4 represent data centers and c_1, c_2 und c_3 clients with a uniformly distributed volume of communication. The destination vector in this example will be computed for a virtual machine currently executed in data center d_4. In this case the destination data center d_2 is chosen for the migration since this data center location has the shortest distance to the destination vector $\vec{z}$. The algorithm can be adapted to consider the network location instead of a geographical location, e.g., by including weights regarding the latency or other quality of service metrics between the data center and the clients.

3.2 Definitions

Let V be a finite-dimensional vector space over $\mathbb{R}$ with the dimension v and P the set of properties with $\beta_P: P \rightarrow V$ the function to map properties of P to the vector space V. Furthermore, let m be a virtual machine defined by the tuple $m = (i_m, P_m)$ with i_m a unique identifier of the virtual machine and $P_m \subset P$ the set of the associated properties. Moreover, let $D = \{(i_d, c_d \in \mathbb{N}^+, M_d)\}$ be the set of available data centers with i_d the unique identifier of the data center, M_d the set of virtual machines currently executed at this data center and c_d the capacity of this data center, so it holds $|M_d| \leq c_d$. The complete set M_{all} of all virtual machines is defined as follows

$$M_{all} = \bigcup_{(i_d, c_d, M_d) \in D} M_d \tag{1}$$

Of course, each virtual machine $m \in M_{all}$ is restricted to only be executed at one data center for any moment in time. This means

$$\begin{gathered} \forall i_d \in D, j_d \in D | i_d = (i_{d_i}, c_{d_i}, M_{d_i}), \\ j_d = \left(i_{d_j}, c_{d_j}, M_{d_j}\right), \\ i_d \neq j_d : M_{d_i} \cap M_{d_j} = \emptyset \end{gathered} \tag{2}$$

Additionally, we define a function $\beta_D: D \rightarrow V$ by

$$\beta_D(x_d \in D) = \vec{v}_{x_d} \tag{3}$$

so that $\vec{v}_{x_d}$ is the positional vector of the data center x_d. One more function $\beta_M: M_{all} \rightarrow V$ is used to map a virtual machine to the positional vector of the data center within it is executed and is defined by

$$\begin{gathered} \beta_M(m \in M_{all}) = \beta_D(d) | d \in D, \\ d = (i_d, c_d, M_d), \\ m \in M_d \end{gathered} \tag{4}$$

Also, we define N as the set of network flows by $N = \{(s \in P, d \in P, n \in \mathbb{N}^+)\}$ so that s is a property of the communication source, d a property of a communication destination and n the metric over a defined observation time span. Furthermore the euclidian distance for V is used for the distance function $\delta: V \times V \rightarrow \mathbb{R}$ as follows:

$$\delta\big((x_1,\dots,x_v)\in V,(y_1,\dots,y_v)\in V\big) = \| x-y \| = \sqrt{(x_1-y_1)^2+\cdots+(x_v-y_v)^2} \quad (5)$$

Moreover, we define the following helper functions:

- The function $\sigma: M_{all} \to D$ maps each virtual machine to the data center it is currently executed in and is defined by:
$$\sigma(m \in M_{all}) = d \mid d \in D, d = (i_d, c_d, M_d), m \in M_d \quad (6)$$
- The function $\theta: M_{all} \to N$ maps each virtual machine to the set of network flows with matching properties:
$$\theta(m \in M_{all}) = \{(s,d,n) \in N \mid m = (i_m, P_m), s \in P_m\} \quad (7)$$
- The function $\phi: M_{all} \to \mathbb{N}^+$ defines for a virtual machine the sum of metrics of the associated network flows. It is defined by:
$$\phi(m \in M_{all}) = \sum_{(s,d,n)\in\theta(m)} n \quad (8)$$
- The function $\beta_A: P \to V$ maps a property to a vector. Respectively, for a property of a virtual machine the positional vector of the associated data center is given by:
$$\beta_A(p \in P) = \begin{cases} \beta_M(m), & \text{if } \exists m \in M_{all}, m = (i_m, P_m): p \in P_m \\ \beta_P(p), & \text{otherwise} \end{cases} \quad (9)$$

3.3 Continuous placement algorithm sequence

For the correct operation of the algorithm at least one data center of the set of data centers D must have spare capacity available. This means:

$$\exists (i_d, c_d, M_d) \in D \mid c_d > |M_d| \quad (10)$$

The sequence of the algorithm is as follows:

1) We define $T = M_{all}$
2) While $T \neq \emptyset$:
   a) We choose an arbitrary $t \in T$ with $t = (i_{m_t}, P_{m_t})$ and define $T = T \backslash t$. Furthermore, we set $t_d = \sigma(t)$ with $t_d = (i_{d_t}, c_{d_t}, M_{d_t})$ and define
$$D_t = \{(i_{d_x}, c_{d_x}, M_{d_x}) \in D \mid c_{d_x} > |M_{d_x}| \vee \exists m \in M_{d_x}: \phi(m) < \phi(t)\} \cup \{t_d\} \quad (11)$$
   For the case $D_t = \{t_d\}$ we can examine the next virtual machine and start over with step 2 of the algorithm.
   b) Now we can compute the destination vector
$$\vec{z} = \beta_M(t) + \sum_{(s,d,n)\in\theta(t)} \frac{n}{\phi(t)} \big(\beta_A(d) - \beta_M(t)\big) \quad (12)$$
   c) We choose the destination data center $z_d = \big(i_{d_z}, c_{d_z}, M_{d_z}\big) \in D_t$ with the shortest distance to the destination vector $\vec{z}$ so that
$$\nexists y_d \in D_t, y_d = \big(i_{d_y}, c_{d_y}, M_{d_y}\big), y_d \neq z_d: \delta(\beta_D(y_d), \vec{z}) < \delta(\beta_D(z_d), \vec{z}) \quad (13)$$
   For the case $z_d = t_d$ the virtual machine is already executed in the optimal data center, so we can examine the next virtual machine and start over with step 2 of the algorithm.
   d) For the case $|M_{d_z}| = c_{d_z}$, for the given destination data center $z_d = (i_{d_z}, c_{d_z}, M_{d_z}) \in D$, we choose the virtual machine with the lowest communication amount and move this machine to the nearest data center $s_d = (i_{d_s}, c_{d_s}, M_{d_s})$ with available capacity:
      i) We determine a virtual machine $x \in M_{d_z}$ so that $x = (i_{m_x}, P_{m_x})$ which holds
$$\nexists y \in M_{d_z}, x \neq y: \phi(y) < \phi(x) \quad (14)$$
      ii) Now, we identify $s_d = \big(i_{d_s}, c_{d_s}, M_{d_s}\big) \in D$ so that $c_{d_s} > \big|M_{d_s}\big|$ which holds
$$\nexists y_d \in D, y_d = \big(i_{d_y}, c_{d_y}, M_{d_y}\big), c_{d_y} > \big|M_{d_y}\big|, y_d \neq s_d: \delta(\beta_D(z_d), \beta_D(y_d)) < \delta(\beta_D(z_d), \beta_D(s_d)) \quad (15)$$
      iii) We define $M_{d_z} = M_{d_z} \backslash x$ and $M_{d_s} = M_{d_s} \cup \{x\}$
   e) Now, $|M_{d_z}| < c_{d_z}$, so we move the virtual machine t to the destination data center z_d. Finally, we alter the sets $M_{d_t} = M_{d_t} \backslash t$ and $M_{d_z} = M_{d_z} \cup \{t\}$.
3) The algorithm has now examined each virtual machine of M_{all}. After a new representative set of network flows is collected, we can start over with step 1 and a new iteration.

As outlined above, the algorithm starts over after all virtual machines were processed. Due to the online/iterative nature of this algorithm new network flows, historical data or even changes in the availability of renewable energy sources will be taken into account. This means, that this approach optimizes the overall topology continuously over time. It is obvious that the temporal resolution of the available data and its rate of change has to be considered when running the algorithm in a real private cloud environment to limit the amount of resources necessary to run the algorithm. Also, oscillations of virtual machines between data centers (e.g., due to similar clients or properties) should be prevented, e.g., using a hysteresis based on former placements of the virtual machine.

4 Evaluation of optimal VM placement and migration costs

The algorithm to optimize the placement of virtual machines described in the previous section is based on the minimization of operational characteristics of the VMs. In the outline of the algorithm described above, energy costs for the data centers hosting the VMs and the distance between the data centers and the clients are minimized. However, regarding the energy efficiency of the approach, the additional costs for the implementation of the algorithm have to be taken into account. While the energy consumption of the algorithm itself can be limited, e.g., by decreasing its resolution using longer intervals, the migrations resulting from the execution of the algorithm lead to additional energy costs. These migrations costs can be divided in direct and indirect costs. Direct migration costs arise from the effort to carry out the migration, i.e. using compute, storage and network resources across multiple data centers and links between them. Indirect costs are formed by the consequences of the migration and impacts on the operational characteristics of the virtual machine. For example, such indirect costs can result from a higher latency for some clients after the migration. The algorithm presented above does not consider possible service level requirements regarding the latency of all clients in favor of its primary goal, to benefit from the lowest energy price. Such constraints could be integrated in the optimization by using them as additional metrics of the virtual machines. This can be described as a the problem to minimize

$$C_{all} = C_{op}(M_{all}) + C_{alg}(M_{all}) + C_{mig}(M_{all}),\\ m \in M_{all}, m = (i_m, P_m),\\ \forall i_p \in P_m | P_{min} \leq i_p \leq P_{max} \quad (16)$$

whereas C_{all} are the overall energy costs to be optimized, $C_{op}(M_{all})$ the operational costs of the data centers based on the the properties and contraints of the virtual machines defined in P, $C_{alg}(M_{all})$ the costs to run the algorithm and the direct (mig_d) and indirect (mig_i) costs for the resulting migration and placement decisions of the algorithm $C_{mig}(M_{all}) = C_{mig_d}(M_{all}) + C_{mig_i}(M_{all})$. The minimization of C_{op} is subject of several projects and related work regarding the energy efficiency and power management of data center infrastructures. As stated above, C_{alg} can be limited by increasing the interval between iterations of the algorithm and its implementation. C_{mig_i} is primarily minimized by the defined constraints. C_{mig_d} is influenced by the migration technique used to transfer the virtual machines' resources between data centers. Based on related work in this area, in [2] we presented results from a simulation to leverage fluctuating renewable energies in northern, southern and central Germany. Also, we implemented an extension for OpenStack to use migrations to enhance the energy efficiency in distributed private clouds. Besides leveraging renewable energies, the testbed also consolidated the virtual machines across different data center sites to lower C_{op}. By integrating the algorithm described in this paper in this testbed and the previously presented simulation studies, also the placement of new virtual machines can be included in the optimization. This also includes the possibility to spawn multiple instances of a service provided by a virtual machine, e.g., to address the constraints regarding the latency between the service and the clients, hence increasing C_{op} in favor of a reduced C_{mig_i} and resulting lower C_{all}. This can solve situations in which the algorithm would migrate a virtual machine to a new site that is too far away from some connected clients and hence would violate predefined service level constraints (as, e.g., implemented by content delivery network and cloud providers).

5 Conclusions and future work

The algorithm described in this paper can be used to evaluate the use of renewable energies to enhance the overall energy efficiency across multiple data centers. Based on our previous research, we are evaluating to use the algorithm in simulations as well as integrate it in an extension for the OpenStack Nova scheduler, that we developed to leverage renewable energy sources and power management in distributed private cloud infrastructures. The scheduling extension can also consider the costs for placing or migrating virtual resources across multiple public or community clouds. Such hybrid cloud environments, can benefit from the energy efficiency and low energy prices in distant, supposedly public, clouds to lower the energy cost for workloads that are safe to be transferred to a third-party cloud provider. To benefit also from short-term fluctuations in the availability of renewable energy sources, e.g., on a daily basis, new live-migration and placement techniques for virtual resources have to be developed. We're currently working on container-based migration and service placement. This lightweight virtualization solution facilitates the transfer of the current state of the virtual resource and the underlying storage. Initial tests have shown that the amount of data that needs to be transferred is far less compared to full-sized virtual machines. However, the network virtualization, especially the data plane performance, and the effort to checkpoint and restore containers under load is still a challenge compared to existing and well-tested virtual machine live-migration techniques.

References

1. D. Versick, D. Tavangarian, in 6. DFN-Forum Kommunikationstechnologien, **31** (2013)
2. K. Spindler, S. Reissmann, R. Trommer, C. Pape, S. Rieger, T. Glotzbach, International Journal On Advances in Internet Technology, **8** (1 & 2), 13–28 (2015)
3. A. Verma, P. Ahuja, A. Neogi, Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware, ser. Middleware '08. Springer-Verlag New York, Inc., 243–264 (2008)

4. X. Wang, X. Liu, L. Fan, X. Jia, Mathematical Problems in Engineering, (2013)
5. D. Levine*, A Parallel Genetic Algorithm for the Set Partitioning Problem*, Meta-Heuristics, Boston, MA, Springer US, 23-35 (1996)
6. A. C.-S. Lin, International Journal of Electronic Commerce Studies **4**, 33-46 (2013)
7. A. Beloglazov, R. Buyya, Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, ser. CCGRID '10. IEEE Computer Society, 577-578 (2010)
8. T. Carli, S. Henriot, J. Cohen, J. Tomasik, *A packing problem approach to energy-aware load distribution in clouds*, arXiv.org (2014)
9. L. Shi, J. Furlong, R. Wang, IEEE Symposium on Computers and Communications (ISCC), 9-15 (2013)
10. M. Mishra, A. Sahoo, IEEE CLOUD, 275–282 (2011)
11. H. N. Van, F. D. Tran, J.-M. Menaud, CIT, 357–362 (2009)
12. J. W. Jiang, T. Lan, S. Ha, M. Chen, M. Chiang, INFOCOM, 2876–2880 (2012)
13. O. Biran, A. Corradi, M. Fanelli, L. Foschini, A. Nus, D. Raz, E. Silvera, CCGRID, 498–506 (2012)