UDC 004.75

# Administration of Virtual Data Processing Center over OpenFlow

**V. M. Solovyev, A. A. Belousov**

Vladimir M. Solovyev, https://orcid.org/0000-0003-3778-8201, Volga Region Centre of New Information Technologies in Volga Region, Saratov State University, 83 Astrakhanskaya St., Saratov 410012, Russia, svm@sgu.ru

Aleksandr A. Belousov, Saratov State University, 83 Astrakhanskaya St., Saratov 410012, Russia, tortyt1@gmail.com

This paper researches the building principles and administration of virtual data processing centers based on hyper-converged systems over OpenFlow. We provide the implementation features of such virtual centers on the basis of software-defined networking that is managed by a dedicated controller (a server). We suggest the graph administration model of hyper-converged system resources compliant with required performance on the one hand and economic requirements on the other. Based on the proposed model, the implementation of a greedy control algorithm for the virtual data processing center over OpenFlow was examined. This algorithm assigns the requests to physical resources by using of dedicated server software. The advantages of such hyper-converged system model on performance issues were outlined, e.g., multi-threaded routing and security, elimination of the majority of current threats. We summarize the possibilities of transition to network infrastructure in these virtual data processing centers. Such infrastructure is focused on data and usage of blockchain technology providing high reliability and content protection.

*Keywords:* converged infrastructure, hyper-converged infrastructure, software defined networks, OpenFlow, virtual data center, service level agreement, multi-threaded routing, quality of service, Data Oriented Network Architecture (DONA), blockchain.

Received: 29.05.2018 / Accepted: 05.09.2018 / Published online: 28.05.2019

DOI: https://doi.org/10.18500/1816-9791-2019-19-2-226-232

Virtual data processing center (VDPC) refers to hyper-converged infrastructure (HCI) that allows us to create virtual machines, data warehouses, switchers and routers, and communication channels. The main task of VDPC is to accept a client connection (tenant[1]) and update it with the help of virtualization technology in network topology. The basis of the mechanism of VDPC resources administration refers to a model extended for HCI tasks [1]. The network topology in this model is represented by graph $T = (C \cup M \cup K \cup L)$, $C$ is a plurality of computing nodes, $M$ is a plurality of data warehouses, $K$ is a plurality of switching elements, $L$ is a plurality of communication channels. Each of plurality has its own vectors of scalar argument defined. This argument sets up the parameters of computing nodes — $c \in C$, of data (memory) warehouses — $m \in M$, of crosspoints — $k \in K$ and of communication channels — $l \in L$ respectively.

$$\begin{gathered} fct(c) = (ct_1(c), ct_2(c), \ldots, ct_n(c)), \\ fmt(m) = (mt_1(m), mt_2(m), ..., mt_n(m)), \\ fkt(k) = (kt_1(k), kt_2(k), ..., fk_n(k)), \\ flt(l) = (lt_1(l), lt_2(l), ..., lt_n(l)). \end{gathered} \tag{1}$$

[1]A tenant represents the requests for virtual machines, data warehouses, switchers, routers, communication channels, and all virtual communication channels.

© Solovyev V. M., Belousov A. A., 2019

In this model VDPC resources are specified by graph $R = (V \cup S \cup D)$,$V$ is a plurality of applications deployed in virtual machines, $S$ is a plurality of virtual data warehouses, $D$ is a plurality of communication channels between virtual machines and data warehouses. Each plurality has its own vectors of scalar argument defined. This argument sets up the parameters of virtual machines — $v \in V$, virtual data warehouses — $s \in S$, communication channels including switching elements that provide required service level agreement (SLA[2]) — $d \in D$ respectively:

$$\begin{aligned} fvr(v) &= (vr_1(v), vr_2(v), ..., vr_n(v)), \\ fsr(s) &= (sr_1(s), sr_2(s), ..., sr_n(s)), \\ fdr(d) &= (dr_1(d), dr_2(d), ..., dr_n(d)). \end{aligned} \tag{2}$$

The parameters (2) providing SLA coincide with corresponding parameters (1) and are represented by mapping of resource requests to HCI topology:

$$O : R \to T \cup \{\oslash\} = \{V \to C \cup \{\oslash\}, \quad S \to M \cup \{\oslash\}, \quad D \to K \cup \{\oslash\}, \quad L\{\oslash\}\}. \tag{3}$$

Resource requests from the expression(3) determine three relationship types between request parameters $r_i$ and physical resources $t_i$, based on HCI topology:

– the requested resources correspond to resources identified by topology $r_i = t_i$;

– overload of physical resources $r_i > t_i$ that violates SLA;

– underload of physical resources $r_i < t_i$ that requires a topology reconfiguration for economic reasons.

In the last case available resources can be represented by residual graph $T_{res} = (\cup M \cup K \cup L)$ that redefines the parameters as follows:

$$\begin{aligned} fct_{res}(c) = fct(c) - \sum_{v \in V} fvr(v), \quad fmt_{res}(m) = fmt(m) - \sum_{s \in S} fsr(s), \\ fkt_{res}(k) = fkt(k) - \sum_{d \in D} fdr(d), \quad flt_{res}(l) = flt(l) - \sum_{l \in L} fdr(d). \end{aligned} \tag{4}$$

Automatic migration of HCI structures managed by controllers over OpenFlow enables us to meet both SLA and economic requirements. Migration is carried out even if it is not possible to assign the warehouse on demand, and data is added to multiple warehouses. In that case, one part of the applications can work with data warehouse, meanwhile the other part can work with data located in different physical storage. In accordance with migration plan, virtual structure relocation should comply with the following requirements:

– there is no SLA violation during relocation;

– relocation is implemented at given time constraints. Automatic operation of controllers enables us to achieve that.

Input to migration is a plurality of incoming requests $Z = \{R_i\}$, a plurality of queried requests $W = \{R_i\}$, a graph of remaining resources $T_{res}$, and time constraint on migration $\tau$. During migration, a new node $s'$ and a virtual communication channel between nodes $s$ and $s'$ are added to the graph of requested resources $R$.

[2]SLA (Service Level Agreement) is a formal contract between a service provider and a client that sets out agreed service quality, service description and the rights of the parties. Such agreement serves as an assessment tool for quality of provided network services.

The administration of VDPC over OpenFlow is based on a greedy algorithm[3] of request assignment to physical resources, with the use of controller (server) software. Expression (4) describes such a greedy algorithm. As an optimization criterion, the most compact allocation of request elements (2) is applied. A similar approach is widely used in data processing centers and by cloud providers [2, 3]. The quality of VDPC administration depends on selected greedy criteria: next request — $K_R$, virtual node — $K_V$, physical node — $K_C$. Criteria $K_V$ and $K_C$ rely on the cost function defined as a weighted sum of required parameters considering the resource deficit. This function is represented as follows: $d(i) = (\sum_R \sum_{e \in R} r_{r,i} - \sum_{c \in C} r_{c,i}) / \sum_R \sum_{e \in R} r_{e,i}$. Then the cost function of an assignment of element e will appear as $r(e) = \sum_{i=l}^{n} d(i) r_{e,i}$. In this equation the selected element HCI is characterized by a vector of values of required resource parameters $(r_{e,1}, r_{e,2}, \ldots, r_{e,n})$. To calculate the measure of the resource deficit, firstly, it is necessary to subtract the values of available physical resources for the required resource parameter from the common value of this resource parameter in all requests. Then the measure is calculated as a quotient of this difference by total sum of required resources. We can define the cost function as weighted sum of required resource parameters considering the resource deficit. According to criterion $K_V$, the HCI virtual element with maximum cost function is chosen. This allows us to assign primarily the most resource-deficient elements and then assign all the other virtual elements. According to criterion $K_C$, the HCI physical element with minimum cost function is chosen. By this, we can ensure maximal utilization (loading) of computing resources. According to criterion $K_C$, the query with the maximum weighted sum of requested resources is chosen.

The general framework of administration algorithm will be as follows.

1. Scheduler[4] analyses incoming requests of resources $Z = \{R_i\}$.
2. If plurality $\{R_i\} \notin \oslash$ is not empty, the program selects another request $R_i$ according to greedy criterion $K_R$. Otherwise, algorithm terminates its functioning.
3. Using the elements of request $R_i$, the program forms a plurality of virtual nodes $U = \{V \cup S\}$. Where it is not possible to form a plurality of virtual nodes $U$, it proceeds to step 14.
4. Scheduler selects another element $N$ from formed plurality of virtual nodes $U$ on the basis of greedy criterion $K_V$. Then this element is placed in queue $Q$ which contains the elements awaiting an assignment.
5. Using the elements $C_i$, scheduler forms a plurality of physical nodes $\{C_i\} \notin \oslash$. It is possible to assign the element $N$ to these nodes based on correct accomplishment of mapping(3). Otherwise, if $\{C_i\} \in \oslash$, program calls the procedure of limited enumeration.
6. The program selects a physical resource from the formed plurality of physical nodes on the basis of greedy criterion $K_C$. It redefines the values of physical resources parameters according to functions (4).
7. Scheduler selects all virtual channels $D_i$ that link element $N$ to elements of request $R_i$ to be assigned.
8. Scheduler sorts a plurality of channels $\{D_i\} \notin \oslash$ by the value of the capacity in ascending order.

[3]The greedy algorithm is an optimization algorithm based on locally optimal decisions that are made at each stage. Whereby, we assume that the final decision will also prove optimal.

[4]Scheduler is a program (service) driven by controller software. The principal scheduler function is to start other programs.

9. The program selects a virtual channel $L_i$ from a plurality of channels $\{D_i\}$. It should ensure the shortest route that links element $N$ to elements of request $R_i$. Where it is not possible to plot the route, it calls the procedure of virtual channel assignment on a physical resource. It redefines the values of physical resources according to functions (4).
10. Scheduler adds the virtual nodes linked with $N$ to queue $Q$. By this, it follows the order of virtual channels from sorted plurality $\{D_i\}$. These channels connect the nodes.
11. Scheduler deletes $N$ from $U$ and $Q$.
12. If $Q$ is not empty, program proceeds to step 4.
13. If $U$ is not empty, program proceeds to step 3. Otherwise, if $U$ is empty, program proceeds to step 1.
14. The program cancels all assignments of the elements of request $R_i$ and removes the request from a plurality $Z = \{R_i\}$. Then it proceeds to step 2.

This algorithm contains two procedures described in [2]. The first one is a procedure of limited enumeration; the second one refers to a procedure of virtual channel assignment to a physical resource. A scheduler calls the procedure of limited enumeration if it is not possible to assign the next virtual node $N$ from a plurality of requests to any physical resource. This procedure analyses a subset of a plurality of physical nodes $\{C_i\}$ from a graph of physical resources. Specified enumeration depth determines the subset capacity; the quantity of viewed subsets is limited. The program views only subsets whose total quantity of nodes' remaining resources allows us to assign the current element $N$. The procedure ensures the execution of step 5 if the program changes (selects) the enumeration depth and quantity of viewed subsets. Scheduler calls the procedure of virtual channel assignment to a physical resource when it is not possible to plot a route that links element $N$ to element of request $R_i$ via virtual channel. The route searching mechanism is based on modified Dijkstra's algorithm [4]. However, it can include only switching elements and communication channels of the physical network to which ratios of mapping accuracy are applied (3). If it is not possible to assign a virtual channel that connects the storage element, the storage search is accomplished. This storage should have the resources to create storage element replication. The replication requires the quantity of resources equivalent to the quantity of storage element resources. All storages selected for replication creation are considered in ascending order of total route length. Furthermore, the possibility of creation of communication channel $l$ for replication is considered. This channel provides capacity and required data-flow intensity. If the communication channel $l$ can not provide the required parameters, program considers another variation of replication mapping. The result is the route that provides coherence between element $N$ and replication. The parameter variations of route and communication channel provide favorable result. The same approach to virtual machines has been widely recognized and studied [5].

Analyzed HCI control algorithm over OpenFlow enables us to plan the computing resources, resources of data storage and network resources of self-organizing cloud platform, by using of SDN technological solutions. This algorithm mechanism also complies with SLA. The algorithm allows us to use physical resources rationally by eliminating their segmentation, with the help of virtual resources migration. The algorithm enables us to administrate the hyper-converged system by specifying the data flows routing

policies. Whereby, it uses the virtual network control functions of virtual and physical devices from different manufacturers. We refer to devices that support OpenFlow protocol. The proposed solution allows us to integrate different networks administrated over OpenFlow and transfer data flows between them effectively, by means of multi-threaded routing (MRT).

We can consider hyper-converged systems as applied to any computing platforms (e.g., hard, programming, cloud, neuromorphic, quantum) which provide user an access to various services. These systems should be user-friendly and support multiple infrastructure layers, surely including layers of safety, reliability, communication services; providing QoS for various data. Furthermore, the network behind HCI should have the opportunities to work with different types of terminals (mobile, desktop, active network, advanced UX/UI[5] etc.). This network should also have single management platform (controller, server) for the full package of services, applications, hardware, and data transfer channels. Whereby, it should select data transfer channel in real time based on QoS and applications needs for capacity and nature of traffic. Convergent technologies are not the endpoint in evolution of the next-generation computing systems. These technologies already allow us to take a content-centric approach onto prevalidated HCI infrastructure. They enable us to create computing systems that leapfrog over end-to-end paradigm towards content or data addressing paradigm (Information Centric Networking, or ICN). This paradigm implies data organization, regardless of location (server, host), through distributed network caching. Expected benefits of this approach include more efficient use of expensive network resources, scalability of computing systems and their adaptability to volatile QoS. The paradigm is based on the primitives *publish/subscribe*, that is to publish the content (make it available) and declare it. These primitives are realized in Data Oriented Network Architecture (DONA). It works as follows: the element of such system receives a request from a similar element or host. Whereby, two scenarios are possible. If the element contains required data in cash, it will implement the request. If the DONA element does not contain the content, it will request similar elements which have data. When it gets a response, it caches the content and implements the request. This universal mechanism is applicable to any protocol, forming a global single mechanism of caching and content delivery. In addition, this mechanism is supported by all network nodes and aimed at all users, not just ICN users. Such a network ensures content security, not security of its delivery. It relies on a content-based model and draws on the concept of reputation, because the provider must sign the content, so users can always define it. Data Oriented Network Architecture interacts well with blockchain[6] technology that provides the high reliability of content storage and protection. Network entry is protected cryptographically. Unauthorized entry requires enormous computing resources proportional to the network size. It allows us to exclude

[5]UX/UI (User Experience/User Interface) refers to an interface design that meets current requirements.

[6]The revolutionary technology of blockchain was created by Satoshi Nakamoto. This technology helps to allocate the digital content without copying it. Pertinently, it resembles a digital book data of which and their modifications are duplicated in the network for several thousands of times and are regularly updated. This distributed database without a central storage node is stored in the network. It provides its users the hosting, such as Google Docs during collective work. Each group of blockchain transactions is a block, and miners conduct the audit of them (digital content). Therefore, this technology operates with chain of blocks created by complex cryptographic algorithms.

human or machine error, missed operations, unauthorized entry etc. In future, over the course of evolution, HCI will employ other network technologies.

## References


1. Clos C. A study of non-blocking switching networks. *The Bell System Technical Journal*, 1953, vol. 32, iss. 2, pp. 406–424. DOI: https://doi.org/10.1002/j.1538-7305.1953.tb01433.x
2. Zotov I. A., Kostenko V. A. Resource allocation algorithm in data centers with a unified scheduler for different types of resources. *Journal of Computer and Systems Sciences International*, 2015, vol. 15, no. 1, pp. 59–68.
3. Meng X., Pappas V., Zhang L. Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement. *2010 Proceedings IEEE INFOCOM*. San Diego, CA, 2010, pp. 1–9. DOI: https://doi.org/10.1109/INFCOM.2010.5461930
4. Cormen T. H., Leiserson C. E., Rivest R. L., Stein C. *Introduction to Algorithms*. Cambridge MA, MIT Press and McGrawHill, 2001. 1202 p.
5. Zhao M., Figueiredo R. J. Experimental Study of Virtual Machine Migration in Support of Reservation of Cluster Resources. *VTDC '07 Proceedings of the 2nd international workshop on Virtualization technology in distributed computing*. New York, NY, USA, ACM, 2007, pp. 1–8. DOI: https://doi.org/10.1145/1408654.1408659


**Cite this article as:**
Solovyev V. M., Belousov A. A. Administration of Virtual Data Processing Center over OpenFlow. *Izv. Saratov Univ. (N. S.), Ser. Math. Mech. Inform.*, 2019, vol. 19, iss. 2, pp. 226–232. DOI: https://doi.org/10.18500/1816-9791-2019-19-2-226-232

УДК 004.75

# Управление виртуальным центром обработки данных по протоколу OpenFlow

**В. М. Соловьев, А. А. Белоусов**

Соловьев Владимир Михайлович, кандидат технических наук, доцент кафедры математической кибернетики и компьютерных наук, начальник Поволжского регионального центра новых информационных технологий, Саратовский национальный исследовательский государственный университет имени Н. Г. Чернышевского, Россия, 410012, Саратов, ул. Астраханская, д. 83, svm@sgu.ru

Белоусов Александр Александрович, магистрант, Саратовский национальный исследовательский государственный университет имени Н. Г. Чернышевского, Россия, 410012, Саратов, ул. Астраханская, д. 83, tortyt1@gmail.com

В работе рассмотрены принципы построения виртуальных центров обработки данных на основе гиперконвергентных вычислительных систем и управление ими по протоколу OpenFlow. Приведены особенности реализации таких виртуальных центров на основе программно-конфигурируемой сети, управляемой выделенным контроллером (сервером). Предложена графовая модель управления ресурсами гиперконвергентной вычислительной системы, отвечающая требованиям заданного качества обслуживания, с одной стороны, и экономическими требованиями, с другой. На основе предложенной модели рассмотрен вариант реализации жадного алгоритма управления виртуальным центром обработки данных по протоколу OpenFlow и осуществляющего назначение запросов на физические ресурсы, используя программное обеспечение выделенного сервера. Показаны преимущества такой модели гиперконвергентной вычислительной системы в вопросах производительности за счет много-