“Mathematical model of cloud computing data center based on OpenFlow”

P.N. Polezhaev, A.E. Shukhman, U.A. Ushakov

This paper describes adequate mathematical model of cloud computing data center based on OpenFlow. The model represents cloud computing data center as weighted undirected graph. Its vertices are network devices including computing nodes, OpenFlow switches, data storages and OpenFlow controller. Edges represent network links between them. Proposed model can be used to implement efficient traffic routing schemes which localizes data flows between virtual machines in the groups, decreases network contention and increases overall cloud efficiency.

Introduction

Cloud computing systems have become the de facto standard in many areas of advanced information technologies. Companies use clouds to deploy their scientific and business applications. They avoid the costs of creating and maintaining their own data centers. On the other hand, the owners of cloud computing data center by consolidating computing resources and storage systems are able to reduce the total cost of IT infrastructure ownership by serving a larger number of customers. Usage of effective technical tools for scheduling and load balancing, packet routing control and integration of several territorial disparate data center segments are another ways to reduce costs for computing cloud data center owners.

Existing network protocols of cloud computing data center are Fiber Channel, Infiniband and traditional Ethernet. They have limited traffic management and QoS. An improved versions of the Ethernet protocol - Converged Enhanced Ethernet and Cisco Data Center Ethernet include extensions for flow control based on priorities, bandwidth sharing, overloads and logical state band data control, lossless data transferring and simultaneous use of several parallel data paths between nodes. The main drawbacks of these solutions are complex decentralized flow control scheme based on a set of proprietary protocols, considerable costs of network equipment and the complexity of its modification. It should be noted that these solutions use reactive flow control scheme, which makes decisions on switching during the transmission of packets.

Software defined networks (SDN) are extremely attractive technologies which can be used in cloud computing data centers. SDN principles first emerged in the research laboratories at Stanford and Berkeley [1]. Now SDN are being developed by a consortium of Open Network Foundation and the European project OFELIA [2]. Known positive experience of Google and Amazon to implement SDN in cloud data centers.

The approach of SDN is the ability to dynamically control the data transmision over the network by using OpenFlow protocol. All active network devices work together under the network operating system, which provides applications with access to network management. Network operating systems can be centralized and use common abstraction for packet forwarding.

By controlling the packet forwarding, software defined networks of the cloud data centers can be used to implement simultaneous multipath data transfer schemes, priority based flow control, network virtualization, QoS mechanism or effectively distribute the load on the network. OpenFlow open standards and the shift of control logic to a separate controller simplifies the hardware and software of active network equipment. Centralization, standardization and transparency of SDN allows to use it to build flexible and efficient data centers which can adapt their infrastructure to the emerging needs of the business.

Currently, there are no solutions based on SDN for cloud computing data centers which can topologically localize data flows between virtual machines in the groups, reduce network contention and increase the overall data center efficiency. This is the novelty of the present work.

Implementation of such approach needs adequate mathematical model of cloud computing

data center based on OpenFlow.

Proposed mathematical model of cloud computing data center based on OpenFlow

Cloud computing data center can be represented as an weighted undirected graph of the form:

$$Cloud = (Devices, Links, type, w_{n.}, w_{sw.}, w_{st.}, w_{l.}), \tag{1}$$

where set of vertecies $Devices = Nodes \cup Switches \cup Storages \cup \{Cont_0\}$ denotes network devices including computing nodes $Nodes = \{N_1, N_2, ..., N_n\}$, OpenFlow switches $Switches = \{S_1, S_2, ..., S_m\}$, network storages $Storages = \{F_1, F_2, ..., F_r\}$ and OpenFlow controller $Cont_0$, edges $Links = \{L_{ij}\}$ represent bidirectional network links. Type of each network device $d \in D$ can be determined by function $type : Devices \to \{"node", "switch", "storage", "controller"\}$.

$Cont_0$ is a special computing node executing OpenFlow controller (network operating system).

Function $w_{n.}$ for each computing node N_i calculates the vector of its characteristics

$$w_{n.}(N_i) = (w_{n.stat.}(N_i), w_{n.dyn.}(N_i, t)), \tag{2}$$

where $w_{n.stat.}(N_i)$ and $w_{n.dyn.}(N_i, t)$ are respectively denote static parameters and dynamic characteristics of N_i. Node static parameters are represented by a vector

$$w_{n.stat.}(N_i) = (M_i, D_i, C_i, P_i) \tag{3}$$

of its RAM size M_i, local disk size D_i, computing cores count C_i and their performance characteristics $P_i = (P_{i1}, P_{i2}, ..., P_{iC_i})$. Dynamic characteristics can be described by vector function:

$$w_{n.dyn.}(N_i, t) = (m_i(t), d_i(t), u_{ik}(t), vm_i(t)). \tag{4}$$

Here $m_i(t)$, $d_i(t)$ are respectively denote available RAM and disk size at the time $t \geq 0$, $u_{ik}(t)$ - utilization of node N_i core k at the time t. $vm_i(t)$ is a set of virtual machine instances running at the time t. All this information can be collected at regular intervals by SNMP protocol.

Each OpenFlow switch $S_j \in Switches$ also has static parameters and dynamic characteristics:

$$w_{sw.}(S_j) = (w_{sw.stat.}(S_j), w_{sw.dyn.}(S_j, t)). \tag{5}$$

Static parameters of S_j include the following values:

$$w_{sw.stat.}(S_j) = (Tp_j, Pc_j, OF_j, Tc_j, Ts_j), \tag{6}$$

where $Tp_j \in \{"100\ Mbit\ Ethernet", "1\ Gbit\ Ethernet", "10\ Gbit\ Ethernet"\}$ denotes supported version of Ethernet protocol, Pc_j is a number of switch ports, $OF_j \in \{"1.0", "1.1", "1.2", "1.3"\}$ is a version of supported OpenFlow protocol. Tc_j denotes a number of flow tables in the switch S_j. Each table has maximum Ts_j flow entries.

Dynamic characteristics of the switch are represented by vector:

$$w_{sw.dyn.}(S_j, t) = (Ft_j(t), Q_j(t), Pt_j(t)). \tag{7}$$

Here $Ft_j(t) = (Ft_{j1}(t), ..., Ft_{jTc_j}(t))$ reflects the current state of all Tc_j flow tables.

At the moment t each flow table $Ft_{jk}(t)$ contains $Rc_j(t)$ OpenFlow rules (flow entries). Each of them has the following form:

$$R_l = (Mt_l, Cn_l, Ac_l). \tag{8}$$

Mt_l is a matching part of the rule, Cn_l - statistical counters and Ac_l represents the set of actions. All incoming packets are compared against flow entries of the flow tables. If a matching entry is found (its Mt_l part is matched against incoming packet), then all actions of Ac_l are performed on the packet and counters Cn_l are updated. Otherwise packet is forwarded to controller $Cont_0$. The

Table 1. Header fields from packets used to match against flow entries

Ingress port	Ether. src.	Ether. dest.	Ether. type	VLAN id	VLAN priority	IP src.	IP dest.	IP proto.	IP ToS bits	TCP/UDP src. port	TCP/UDP dest. port

contoller is responsible for determining how to process packets without flow entries in switches flow tables, it manages the switches flow tables by adding and removing flow entries.

In OpenFlow protocol of version 1.0 [3] matching part has the header fields described in the table 1.

Each header field can contain specific value or ANY value. This makes it possible to implement different switching, routing and firewall schemes of traffic control.

Ac_l can contain the following actions [3]:

- Forward all. Send packet out all ports except its incoming port.
- Forward controller. Encapsulate and send packet to the OpenFlow controller.
- Forward local. Send the packet to the local networking stack of the switch.
- Forward table. Perform actions in the table.
- Forward in port. Send packet out the input port.
- Forward normal. Process packet without OpenFlow.
- Forward flood. Send packet along the minimum spanning tree, not including incoming port.
- Enqueue. Send packet to concrete QoS queue.
- Modify field. Changes the specific field of packet's header.

Empty list of actions in Ac_l means that packet should be dropped. The most usual action is the forwarding of packet to specific port.

Counters are collected by OpenFlow for each flow entry. Cn_l contains the following values [3]:

- Received packets. Total number of packets matched against the flow entry.
- Received bytes. Total size of packets matched against the flow entry.
- Duration seconds. Duration in seconds of time the flow has been installed in the switch.
- Duration nanoseconds. Nanoseconds part of duration.

These metrics give the representation of the flow entries usage.

Also each flow table $Ft_{jk}(t)$ contains dynamic metrics set $CFt_{jk}(t)$ which incudes [3]:

- Active entries count.
- Packet lookups count.
- Packet matches count.

These values can be collected at regular intervals using OpenFlow protocol.

In (7) $Q_j(t)$ represents a set of packet queues $\{Q_{jkr}(t)\}$ associated with specific switch port and Type of Service (ToS) value. They are used to realize QoS mechanism, which maintains the minimum guaranteed bandwidth on the given network links.

$\{Q_{jkr}(t)\}$ contains queue metrics including [3]:

- Transmit packets count.
- Transmit bytes count.
- Transmit overrun errors count.

These metrics are also can be collected by OpenFlow. They help to understand the efficiency of QoS implementation in computing cloud.

$Pt_j(t) = (Pt_{j1}(t), Pt_{j2}(t), ..., Pt_{jPc_j}(t))$ (see (7)) is a set of dynamic port characteristics of switch S_j. Port k of switch S_j can be described by values [3]:

- Port status (on or off).
- Total received packets count.
- Total transmitted packets count.
- Total received bytes.

- Total transmitted bytes.
- Total received packet dropped notifications.
- Total transmitted packet dropped notifications.
- Total received errors count.
- Total transmitted errors count.
- Total received frame alignment errors.
- Total received overrun errors count.
- Total received CRC Errors count.
- Total collisions count.

These metrics are collected at regular intervals by OpenFlow protocol. They help to discover links and switches overloads, their failures.

Network storages keep virtual machine instances, application databases and computing cloud infrastructure database. Any storage F_k has vector of characteristics

$$w_{st.}(F_k) = (Vl_k, vl_k(t)), \tag{9}$$

where Vl_k - maximum size of storage, $vl_k(t)$ - its current size at the time t.

For each network link $l_{ij} \in L$ $w_{l.}(l_{ik})$ (see topology graph (1)) denotes the vector of its static parameters and dynamic characteristics:

$$w_{l.}(l_{ij}) = (B_{ij}, Lat_{ij}, b_{ij}(t), lat_{ij}(t)), \tag{10}$$

where B_{ij} is a maximum bandwidth of network link measured under condition of the contention absence, Lat_{ij} is maximum latency, $b_{ij}(t)$ - current bandwidth at the time t, $lat_{ij}(t)$ - current latency. The last two characteristics consider network contention. They can be measured at regular intervals by special software like ping and iperf.

Usage of proposed mathematical model

Advantages of the proposed cloud computing data center mathematical model:

- Detailed network topology description.
- Effective data structure representation.
- Support of multiprocessor nodes and virtual machine instances.
- Associating of cloud elements static parameters and dynamic characteristics.
- Support of OpenFlow protocol.
- Support of different OpenFlow counters.
- Ability of failure detection.
- Support of QoS mechanism.
- Ability of cloud's QoS implementation efficiency evaluation.

Unified cloud computing data center representation as weighted undirected graph can be used to solve different problems. Authors use it as a part of their scientific research related to the development of cloud computing system based on OpenFlow. System has multitenant architecture. Each tenant has a group of virtual machines connected by virtual network with minimum guaranteed link bandwidths. System should maintain efficient simultaneous work of many virtual machine groups on the same physical cloud hardware. OpenStack [4] is selected as the basis for development of such cloud computing system.

OpenFlow helps authors to solve localization problem of traffic generated by groups of virtual machines. After scheduling of virtual machines group cloud dispatcher transmits selected nodes and group topology information to OpenFlow controller (see figure 1). Controller installs routing rules to switches so as to localize data flows between virtual machines in the group. This leads to decreasing of network contention and computation time (or response time) of software programs executed in virtual machines. In this case cloud computing data center mathematical model is used to build efficient data flows.

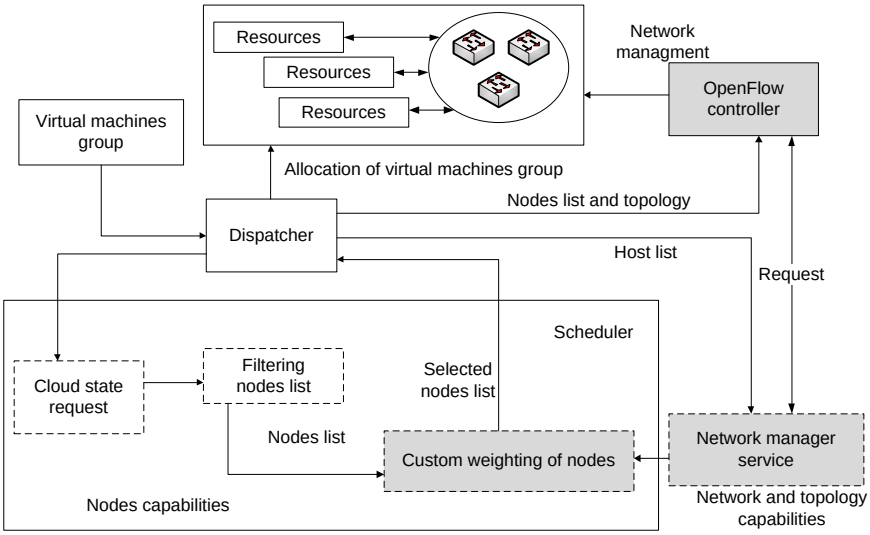

Figure 1. Principle scheduling scheme in the cloud computing system.

OpenFlow and presented model are also used for implementation of QoS mechanism, which allows to setup minimum guaranteed bandwidth for links.

Conclusion

Adequate mathematical model of cloud computing data center based on OpenFlow standard 1.0 is proposed. It can be used to implement efficient traffic routing schemes which localizes data flows between virtual machines in the groups, decreases network contention and increases overall cloud efficiency.

Proposed mathematical model will be used in cloud computing system based on OpenFlow and developed by authors.

Research was supported by the federal target program "Research and development in priority fields of scientific and technology complex of Russia in 2007 - 2012" (state contract no. 07.514.11.4153) and Russian Foundation for Basic Research (project no. 12-07-31089).

References

[1] OpenFlow - Enabling Innovation in Your Network. http://www.openflow.org/
[2] OFELIA: OpenFlow in Europe. http://www.fp7-ofelia.eu/.
[3] OpenFlow Switch Specification, Version 1.0.0. http://www.openflow.org/documents/openflow-spec-v1.0.0.pdf
[4] OpenStack Open Source Cloud Computing Software. http://www.openstack.org/.

Authors

Petr Nikolaevich Polezhaev — assistant, faculty of mathematics, Orenburg State University, Orenburg, Russian Federation; E-mail: *peter.polezhaev@gmail.com*

Alexander Eugeniyevich Shukhman — candidate of pedagogic sciences, associate professor, faculty of mathematics, Orenburg State University, Orenburg, Russian Federation; E-mail: *shukhman@gmail.com*

Yuriy Alexandrovich Ushakov — candidate of engineering sciences, associate professor, faculty of information technologies, Orenburg State University, Orenburg, Russian Federation; E-mail: *unpk@mail.ru*