Load balancing method for KDN-based data center using neural network

Alex M. R. Ruelas
alrodrig@dca.fee.unicamp.br / University of Campinas. Campinas, Brazil

Christian E. Rothenberg
chesteve@dca.fee.unicamp.br / University of Campinas. Campinas, Brazil

Receipt: 3-7-2018 / Acceptance: 20-8-2018

Abstract. The growth of cloud application services delivered through data centers with varying traffic demands unveils limitations of traditional load balancing methods. Aiming to attend evolving scenarios and improve the overall network performance, this paper proposes a load balancing method based on an Artificial Neural Network (ANN) in the context of Knowledge-Defined Networking (KDN). KDN seeks to leverage Artificial Intelligence (AI) techniques for the control and operation of computer networks. KDN extends Software-Defined Networking (SDN) with advanced telemetry and network analytics introducing a so-called Knowledge Plane. The ANN is capable of predicting the network performance according to traffic parameters paths. The method includes training the ANN model to choose the path with least load. The experimental results show that the performance of the KDN-based data center has been greatly improved.

Keywords: OpenFlow, sFlow, data center, artificial neural network, Knowledge-Defined Networking

Método de equilibrio de carga para un centro de datos basado en KDN mediante la aplicación de una red neuronal

Resumen. El crecimiento de los servicios de aplicaciones en la nube entregados a través de centros de datos con diferentes demandas de tráfico revela las limitaciones de los métodos tradicionales de equilibrio de carga. Con el objetivo de asistir a escenarios en evolución y mejorar el rendimiento general de la red, este documento propone un método de equilibrio de carga basado en una Red Neuronal Artificial (ANN) en el contexto de las Knowledge-Defined Networking (KDN). Las KDN buscan aprovechar las técnicas de Inteligencia Artificial (AI) para el control y operación de redes de computadoras. Las KDN amplían las Redes Definidas por Software (SDN) con telemetría avanzada y análisis de red al introducir el así llamado Plano de Conocimiento. La ANN es capaz de predecir el rendimiento de la red de acuerdo con los parámetros de tráfico. El método incluye el entrenamiento del modelo ANN para elegir el camino con menos carga. Los resultados experimentales muestran que el rendimiento del centro de datos basado en las KDN ha mejorado enormemente.

Palabras clave: OpenFlow, sFlow, centro de procesamiento de datos, red neuronal artificial, Knowledge-Defined Networking

1. Introduction

Data centers are the main hosting infrastructures of Internet applications and services (i.e., multimedia content, Internet banking, and social networks). Traditional load balancing methods in such data center networks use dedicated hardware devices to distribute the network traffic in different server replicas. Although this approach achieves high performance in general, it is expensive and lacks flexibility in its configuration, which cannot be dynamically adjusted based on real-time network state or other information.

As an innovative networking technology that offers (logical) centralization of network control and introduces the ability to program the network, Software-Defined Networking (SDN) has been applied to many load balancing systems (Al-Fares, Loukissas, & Vahdat, 2008; Akyildiz, Lee, Wang, Luo, & Chou, 2014; Kreutz et al., 2015), based on decoupling of the data and control planes in which the forwarding state in the data plane is managed by a remote control plane. OpenFlow version 1.3 (O.N.F., 2013) is currently the most well-known SDN protocol. OpenFlow-based SDN can be used for monitoring network switches, sending policy commands to each switch, programming routes with flow tables that can define the planned path, and allowing the dynamic reprogramming of network devices through an external controller that contains the control logic with global knowledge of the network state.

Knowledge-Defined Networking (KDN) (Mestres et al., 2017) is a recent networking paradigm that allows the application of Artificial Intelligence (AI) techniques for controlling and operating the network. KDN relies on SDN, telemetry, and network analytics, and introduces a so-called Knowledge Plane for network control and management operations.

In this paper, we propose a KDN approach based on a load balancing method which uses performance metrics (bandwidth and latency) of each path of the network. Applying an Artificial Neural Network (ANN) to model the system behavior, the proposed load balancing method selects the least loaded path for each flow, altogether improving the SDN routing capabilities.

The rest of this paper is organized as follows: Section II gives an overview of the related works. Section III presents the proposed load balancing scheme, and describes the method to acquire two load features and the major mechanism of Artificial Neural Network to achieve an integrated path load. Section IV presents the experimental results compared with other load balancing schemes. Finally, Section V concludes with final remarks and future work.

2. Related work

Many methods and mechanisms have been proposed to implement load balancing in SDN environments using the OpenFlow protocol (Akyildiz et al., 2014; Andersson & Termander, 2015; Al-Fares, Radhakrishnan, Raghavan, Huang, & Vahdat, 2010; Li & Pan, 2013; Yang, Su, Liu, & Yang, 2014; Rotsos, Mortier, Madhavapeddy, Singh, & Moore, 2012). In Equal-Cost

MultiPath (ECMP) (Hopps, 2000) and Valiant Load Balance (VLB) (Zhang-Shen, 2010), the controller is used to analyze the replying information from OpenFlow switches and modify the flow tables by specific load balancing strategy, in order to efficiently plan the data transmission path and achieve load balancing in SDN. However, these strategies belong to the static load balancing method which cannot make a dynamic routing plan, according to real-time network load condition. In their thesis, Andersson & Termander (2015) propose a dynamic load balancer design which uses performance management (PM). Thus, PM measures key indicators of network performance such as latency and packet loss for all paths (end-to-end) in the network. This module can detect performance fluctuations in the network and, combined with an SDN controller, makes it possible to reroute the traffic around links that have reduced capacity, with the goal of balancing the load and improving the overall performance. Al Sallami & Al Alousi (2013) propose an ANN-based load balancing technique. To distribute equal load among all servers, this technique uses a back propagation algorithm. The demand of each user is predicted and the resources are allocated according to the predicted demand, but the active servers at any given time depend on the demand of users at that specific time. As a result, active servers are minimized, which leads to low energy consumption. Furthermore, the relation between energy consumption and carbon emission is highlighted in this paper.

Yang et al. (2014) implement an OpenFlow load balancer module and establish a data mining algorithm to predict the future user traffic. In this work, the proposed load balancer is compared with others', measures the influence of flows, and tests the effect of delay times. The experimental results show the proposed module is faster than the general load balancer.

The dynamic load balancing algorithm proposed in Du & Zhuang (2015) is based on OpenFlow and sFlow to efficiently distribute the traffic among the servers of the cluster. The algorithm makes decisions based on real-time traffic statistics obtained via the sFlow protocol.

Research on load balancing in KDN-based data center networks lacks specific load balancing methods. There are several tools to collect information from each path, e.g., the sFlow protocol, performance management tools, OpenFlow protocol, etc. This paper studies a KDN-based method using the global view of the network to collect two features, i.e., the available bandwidth and the observed patch latencies, to feed an Artificial Neural Network capable of suggesting load balancing decisions of the least loaded paths.

3. Load balancing method using an artificial neural network

3.1 Load balancing system design

The network architecture for the proposed load balancing method in the KDN-based data center is shown below.

Figure 1 shows the sFlow-RT that gathers the metrics of multiple paths of the data center. Then the data is sent to the ANN module to be processed by the ANN. The result will be a route with the least load. This route will be sent to the SDN controller. After that SDN controller chooses the network devices of the resulting route, it will send flow tables for OpenFlow switches to achieve the plan of data flow transmission.

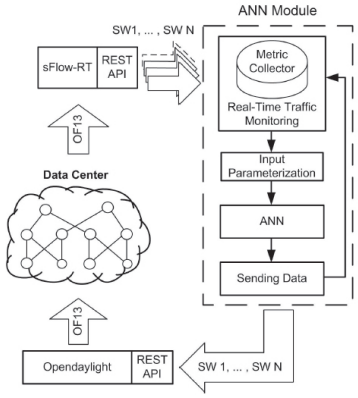

Figure 1. Proposed load balancing method
Elaborated by the author

The proposed load balancing algorithm is as follows:

i. If the SDN controller finds only a single path for data transmission, then the SDN controller will create flow tables and allocate them to OpenFlow switches for active data transmission.

ii. If the SDN controller finds multiple paths for data transmission, then the SDN controller will transmit multiple-path load information to the ANN module. Besides, the sFlow-RT gathers the metrics (bandwidth and transmission latency) of multiple paths of the data center.

iii. The ANN module processes the metrics and chooses the least loaded path, resulting in the return to the SDN controller.

iv. The SDN controller receives the chosen path from the ANN module and creates flow tables to allocate to OpenFlow switches.

3.2 Path features extraction using sFlow-RT

sFlow is a general-purpose network-traffic measurement technology. The sFlow agent uses the statistical packet-based sampling of switched packet flows to capture traffic statistics from

the switch. Therefore, the traffic can be accurately identified and monitored (Phaal, 2004). In order to acquire the load condition of each path, sFlow-RT should be used to obtain the link load condition between two switches. We implemented the application created in the sFlow-RT real-time network weather map example[1], which allowed us to know the available bandwidth in each link between switches.

This paper utilizes the available bandwidth (BW) and transmission latency (Latency) as the evaluation criteria for the path load condition.

i. Available bandwidth (BW): It can reflect the load condition of one link. The sFlow-RT sends the bandwidth of each link employing the REST API to the ANN module. If one path contains several links L_1, L_2, ..., L_n, with the corresponding bandwidth utilization ratio BW_1, BW_2, ..., BW_n, the available bandwidth ratio of this path can be calculated by equation (1):

$$BW = \frac{1}{n}\sum_{i=1}^{n} BW_i$$

ii. Transmission latency (Latency): It is the time spent by the host switch on data transmission (i.e., the amount of time required to push all the packet's bytes into the wire). It can indicate the congestion status of a link and the load situation of the switch in some way. The SDN controller can collect the transmitted bytes Num_Byte in this period and the transmission rate tx_{Rate} at the corresponding OpenFlow switches port. Then, the transmission latency can be calculated by equation (2):

$$Latency_i = \frac{Num\_Byte}{tx_{Rate}}$$

For several links L_1, L_2, ..., L_n with transmission latency $Latency_1$, $Latency_2$, ..., $Latency_n$, respectively, then the total latency of this path is as follows (equation 3):

$$Latency_i = \frac{1}{n}\sum_{i=1}^{n} Latency_i$$

We collected the load features from each communication path by random and ECMP load balancing methods.

1 https://github.com/sflow-rt/svg-weather

3.3 Artificial neural network module

We used a multilayer perceptron (MLP) (Bishop, 1995): a class of feed-forward artificial neural network. An MLP has three types of layers: input, hidden, and output. The number of input and output units in a neural network is generally determined by the dimensionality of the data set, while the number M of hidden units is a free parameter that can be adjusted to give the best generalization performance, corresponding to the balance between underfitting and overfitting. Therefore, we have 64 input nodes and 4 output nodes as shown in Figure 2. Each node is a neuron that uses a nonlinear activation function. The inputs are the available bandwidth (BW) from each link of the data center topology and the outputs are the transmission latency (Latency) in each route. The MLP utilizes a learning technique called "backpropagation" for training. In our experimental setup, we set the value of 1000 as the number of training times, and 0.01 as the learning rate. The structure of the MLP is depicted in Figure 2.

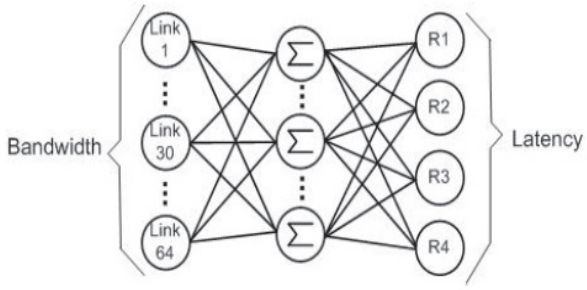

Figure 2. Path load balancing neural network design
Elaborated by the author

There are many rule-of-thumb methods (Heaton, 2008) for determining the correct number of neurons to use in the hidden layers, such as the following:

i. The number of hidden neurons should be between the size of the input layer and the size of the output layer.

ii. The number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer.

iii. The number of hidden neurons should be less than twice the size of the input layer.

3.4 SDN controller

We assume an SDN control platform (that could be implemented as a distributed system) to obtain the global view of the SDN network and discover all paths between the network

device. To update the state of each datapath device, the SDN controller uses a protocol like OpenFlow.

4. Experimental evaluation

We use the Mininet network emulator (Mininet, 2017) and the OpenDaylight SDN controller (Opendaylight load balancer, 2017) to run experiments on multipath SDN environments and evaluate the performance of a proposed load balancing method. The emulated SDN topology is shown in Figure 3.

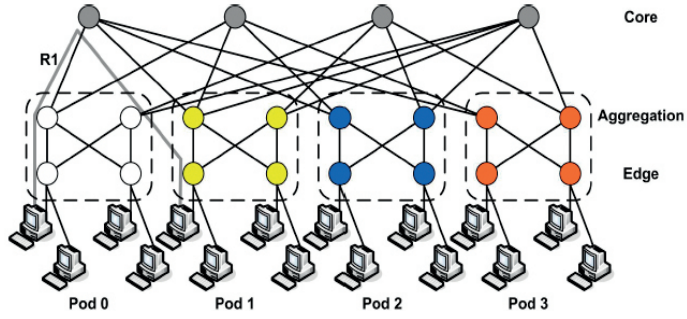

Figure 3. Experimental data center topology. Simple k-ary fat-tree topology (k = 4)
Source: Al-Fares, et al., 2010

Figure 3 shows a multiple path network topology. This topology contains four paths between the pod from source node to destination node. Example of Pod 0 to Pod 1:

Route 1: *Switch1_1→Switch1_2→Switch5_1→Switch2_2→Switch2_1*

Route 2: *Switch1_1→Switch1_2→Switch5_2→Switch2_2→Switch2_1*

Route 3: *Switch1_1→Switch1_3→Switch5_3→Switch2_3→Switch2_1*

Route 4: *Switch1_1→Switch1_3→Switch5_4→Switch2_3→Switch2_1*

In a real network environment, the network traffic presents strong randomness and uncertainty. So, in this paper, the experiment is designed to simulate the network environment as close possible to reality.

The traffic will be transmitted from Host1 to Host5, Host2 to Host6, ..., Host 15 to Host3, Host16 to Host4. During the 16 traffic transmissions, the load conditions of four

paths from Host1 to Host5 are quite different. Then, we will start to control Host1 to send traffic to Host5, which is regarded as the new incoming dataflow in the SDN domain.

4.1 Experimental results

4.1.1 Training results of MLP

Before using the MLP to indicate the integrated path load, it is indispensable to train the neural network with a large number of datasets in order to achieve the least error Artificial Neural Network. We used Mininet to emulate the KDN topology as shown in Figure 3. Hosts in the topology will transmit traffic randomly to other hosts to simulate the network traffic, while the SDN controller utilizes OpenFlow and sFlow protocol to record the path load features. We collected path load features in 480 sec to generate the datasets and train the MLP. In this work, the dataset of inputs and outputs was divided randomly into three subsets:

- Training set (75 %)
- Valediction set (15 %)
- Testing set (15 %)

As depicted in the previous section on the ANN module, the number of neurons (M) in the hidden layer cannot be easily determined. In the procedure of training, we trained the MLP, so we could get the results of these different neural networks to evaluate the performance of the MLP. The experimental results are shown in Table 1.

Table 1
Different number of hidden nodes in MLP

Number of hidden node		Error
1 layer	2 layer	
12	3	0.52
12	6	0.45
20	6	0.22
30	6	0.11
30	12	0.002

Elaborated by the author

We may choose the option that has the least errors. The last option may be preferred, since it has a reasonable error of 0.002. It gives the best generalization performance corresponding to the optimum balance between underfitting and overfitting. Figure 4 shows the decrease of the error by the number of epochs.

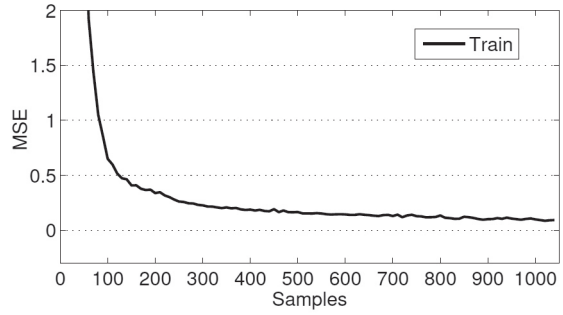

Figure 4. Evolution of the MSE (mean squared error) as a function of the size of the training set
Elaborated by the author

4.1.2 Load balancing experiments

The last experiment tests the performance of our load balancer with some realistic workloads. In order to evaluate the load balancing method based on MLP, we compared it with two other load balancing methods, namely, the ECMP and a random load balancing method. There are 16 traffics among hosts: each one generates an MLP according to the data entered by the load balancers (ECMP and random). Therefore, we have 16 MLP which choose a path with less traffic among the 4 possible routes. Figure 5 shows the results of the proposed method compared with the other two methods used for training the ANN.

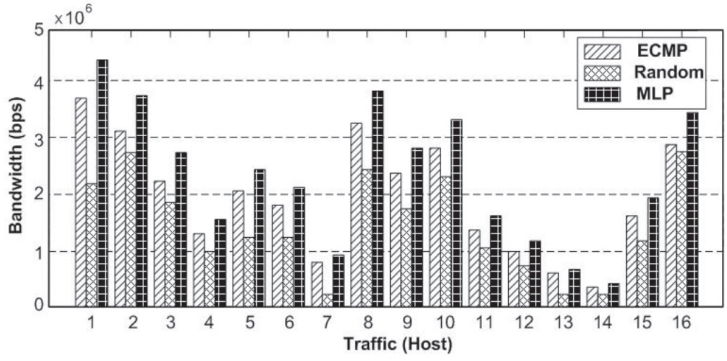

Figure 5. Results obtained with the three load balancing methods
Elaborated by the author

5. Conclusions and future work

We present a load balancing method based on MLP to predict path traffic demands, change the routes according to the proposed traffic metrics, and choose the least loaded path between source and destination nodes. Bandwidth utilization ratios and path latencies are collected and integrated into the Artificial Neural Network to represent the path load condition. Experimental results from the Mininet emulator and the OpenDaylight controller point to the performance opportunities of applying KDN-based methods in data center scenarios.

In future works, we intend to run larger scale experiments with diverse topologies. In addition to applying KDN to data center scenarios, we will also consider video streaming scenarios over wireless access networks using Mininet-WiFi.

References

Akyildiz, I. F., Lee, A., Wang, P., Luo, M., & Chou, W. (2014). A roadmap for traffic engineering in sdn-openflow networks. *Computer Networks, 71*, 1-30.

Al-Fares, M., Loukissas, A., & Vahdat, A. (2008). A scalable, commodity data center network architecture. *ACM SIGCOMM Computer Communication Review, 38*(4), 63-74.

Al-Fares, M., Radhakrishnan, S., Raghavan, B., Huang, N., & Vahdat, A. (2010). Hedera: Dynamic flow scheduling for data center networks. In Nsdi, 10).

Al Sallami, N. M., & Al Alousi, S. A. (2013). Load balancing with neural network. (IJACSA) *International Journal of Advanced Computer Science and Applications, 4*(10).

Andersson, J., & Termander, E. (2015). Performance management in software defined networking.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford University Press.

Du, Q., & Zhuang, H. (2015). Openflow-based dynamic server cluster load balancing with measurement support. *Journal of Communications, 10*(8), 16-21.

Heaton, J. (2008). *Introduction to neural networks with java*. Heaton Research, Inc.

Hopps, C. (2000, November). Analysis of an equal-cost multi-path algorithm. https://tools.ietf.org/html/rfc2992. ([accessed: 9 Aug. 2018])

Kreutz, D., Ramos, F. M., Esteves Verissimo, P., Esteve Rothenberg, C., Azodolmolky, S., & Uhlig, S. (2015). Software-defined networking: A comprehensive survey. proceedings of the IEEE, 103(1), 14-76.

Li, Y., & Pan, D. (2013). Openflow based load balancing for fat-tree networks with multipath support. In Proc. 12th ieee international conference on communications (icc'13), Budapest, Hungary (pp. 1-5).

Mestres, A., Rodriguez-Natal, A., Carner, J., Barlet-Ros, P., Alarcón, E., Solé, M.,... others (2017). Knowledge-defined networking. ACM SIGCOMM *Computer Communication Review, 47*(3), 2-10.

Mininet. (2017). An Instant Virtual Network on your Laptop (or other PC).Retrieved from http://mininet.org/

O.N.F. (2013). *Openflow switch specification, tech. rep., oct. 2013*. Retrieved from https://www. opennetworking.org/images/stories/downloads/sdn-resources/onf-specifica

Phaal, P. (2004). *sflow version 5*. Retrieved from http://www. sflow. org/sflow_version_5. txt

Rotsos, C., Mortier, R., Madhavapeddy, A., Singh, B., & Moore, A. W. (2012). Cost, performance & flexibility in openflow: Pick three. In *Communications (ICC), 2012 IEEE International Conference on* (pp. 6601-6605).

Yang, C.T., Su, Y.W., Liu, J.C., & Yang, Y.-Y. (2014). Implementation of load balancing method for cloud service with open flow. In *Cloud computing technology and science (cloudcom),2014 IEEE 6th International Conference* (pp. 527-534).

Zhang-Shen, R. (2010). Valiant load-balancing: Building networks that can support all traffic matrices. In *Algorithms for next generation networks* (pp. 19-30). Springer.