Muses: Enabling Lightweight and Diversity for Learning Based Congestion Control

Wei Wang, Yiyang Shao, Kai Zheng
Huawei

CCS CONCEPTS

• Networks → Transport protocols;

KEYWORDS

Congestion Control, Machine Learning

ACM Reference Format:

Wei Wang, Yiyang Shao, Kai Zheng. 2018. Muses: Enabling Lightweight and Diversity for Learning Based Congestion Control. In *SIGCOMM Posters and Demos '18: ACM SIGCOMM 2018 Conference Posters and Demos, August 20–25, 2018, Budapest, Hungary*. ACM, New York, NY, USA, 3 pages. <https://doi.org/10.1145/3234200.3234202>

1 INTRODUCTION

Nowadays learning based congestion control (CC) algorithms ramp up quickly (e.g., Remy, Indigo, PCC) and are very promising to increase the average bar of CC performance. These methods leverage the ideas of machine learning technique and train the control model based on big network data. They enable more accurate and fine-grained network state perception, and have more general and powerful representation ability which could deal with complex network states. Based on the performance ranking results on Pantheon platform [1] shown in Fig. 1(a), the learning based CC algorithms have shown their obvious performance improvement in some scenarios.

However, except the common interest (e.g., valuable data and labels, suitable learning algorithms) that all machine learning based methods focus on, there are still three practical challenges in learning based CC

Figure 1: Learning based congestion control algorithms are promising but challenging in practice

algorithms: 1) **High resource consumption**. We reimplement the neural-network-based congestion control model named Indigo [1] in C++, and measure the total CPU usage of the file download client-server application in two Intel Xeon 2*2.40GHz-CPU servers which are directly connected by a 1Gbps link. In the experiments, the model is invoked to update the window in a fixed interval ranging from 10ms to 1ms respectively, and we find the CPU usage increases from 27% to 98% for about 936Mbps throughput as shown in Fig. 1(b), while Cubic only costs 18% for the same throughput. For large numbers of concurrent flows, this is definitely not acceptable. 2) **Long response time**. We measured the response time of calling the model: it takes about 450us-1ms to infer the window size once, which is too long in a high-speed network like data center network. 3) **Weak robustness**. We tested the learning based algorithm in various network paths, and find that in some unknown network paths, i.e., bandwidth or RTT is very different from the training data set, the performance degradation is obvious and the control action can not converge.

We propose a novel system named Muses to solve above problems. Instead of using a heavy and ensemble model for all scenarios, Muses leverages a two-stage learning method (called teacher-student learning): firstly training an ensemble *universal model* using as many scenarios of data as possible, then training a set of lightweight *specialist models* of which the knowledge is distilled from the *universal model* for each specific scenario. Different specialist models are orchestrated for different scenarios, and are able to achieve comparative performance with the universal model and gain both lower resource consumption and better robustness.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SIGCOMM Posters and Demos '18, August 20–25, 2018, Budapest, Hungary

© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5915-3/18/08...\$15.00
<https://doi.org/10.1145/3234200.3234202>

Figure 2: Muses Architecture

2 DESIGN AND ARCHITECTURE

Why teacher-student learning is necessary? As mentioned in previous section, the key challenge is how to achieve the lightweight model as well as keeping high performance. A straightforward answer is “divide and conquer”, i.e., we divide the congestion control problem into several sub-scenarios, then in each sub-space we can have a simpler machine learning model as a feasible solution. However, this is not practical for the reasons that: 1) It is quite hard to directly train a simple model in some complex network scenarios due to the weak representation ability of simple models. 2) Training steps, including collecting data and labels, tuning model parameters, etc., are hard to reuse across various scenarios. 3) In some scenarios it is difficult to get enough training data to let the model converge.

In Muses, we collect as many scenario data as possible together to enrich semantics in training data and leverage the powerful representation ability of universal model by using deep neural network (DNN). Compared with the origin label which is normally called “hard target”, the output of tempered softmax layer in neural network, which is defined as “soft target”, contains more class similarity information in nature. Leveraging soft targets and so called knowledge distillation technique [2], specialist models could converge easily even using simple models to perform light-weight inference at runtime. Meanwhile, the mechanism of transferring knowledge from teacher model to student model increases efficiency a lot since only training data varies across different scenarios. Additionally, this mechanism is able to offer better generalization ability to avoid overfitting even on limited data by controlling the shape of posterior probability distribution of the teacher model.

Training Procedure of Teacher. First, we construct various typical network paths (e.g., data center network, cellular network) and collect packet traces in each path. Second, using these traces we build the network emulators each of which can reproducibly emulate corresponding network environment, and we can calculate and obtain the best congestion control action (as the

target in learning phase) from the emulators. Finally, the universal model is trained on all of the emulators using reinforcement learning algorithm.

Training Procedure of Student. We leverage knowledge distillation technique to train the specialist model to imitate the universal model in a specific domain. Once we have the universal model $\theta_{teacher}$, and a set of specific domain data $\{x, y\}$, in which x is the input features corresponding to observed network state and y is the hard target which indicates best congestion control actions, we can train the specialist model $\theta_{student}$ by minimizing following loss function:

$$L = (1 - \rho)C_{hard}(x, y) + \rho C_{soft}(x, q)$$

where ρ is the weight of the hard and soft cross entropy losses, q is the soft target outputted by $\theta_{teacher}$ under the given input x , C_{hard} and C_{soft} respectively represent the hard and soft cross entropy losses [2].

The first term of the equation above constrains the specialist model $\theta_{student}$ to be trained so that it tends to accurately classify in-domain data $\{x, y\}$, while the second term drives the specialist model to acquire generalizability by learning the class similarity information from the universal model $\theta_{teacher}$. Note that the domain data set $\{x, y\}$ can be reasonably small and reuse the training data of universal model, thus the training procedure of the students can be quite efficient.

Congestion Control Orchestrator. The specialist models are set adaptively by the orchestrator according to the communication peers’ network environments. The two peers’ network information (e.g., network type and bandwidth) is exchanged through the handshake packets of transport protocol, and then each peer’s orchestrator individually decides its model instance. Based on the exchanged information, the orchestrator will query all specialist models’ specification to find the most suitable one. If it does not exist, the traditional algorithms like Cubic are used. The orchestrator enables diversity and avoid the performance degradation of learning based models in unknown environment.

3 DEMONSTRATION

The demo will present how to build the training system, examples of training universal model and specialist models, and evaluate their performance in typical usecase. In the usecase, the universal model is built as a DNN model by reinforcement learning, while three specialist models are taught respectively for LAN, cellular Internet and cable Internet. We showcase the adaptation of specialist models in different networks, then compare their performance improvement and the resource usage with universal model and traditional algorithms.

REFERENCES

- [1] Y. Yan Francis, Ma Jestin, etc., and Winstein Keith. 2018. Pantheon: the training ground for Internet congestion-control research. In *USENIX Annual Technical Conference*.
- [2] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).