Network Optimizations on Prediction Server with Multiple Predictors

Kaho Okuyama, Yuta Tokusashi, Takuma Iwata, Mineto Tsukada,
Kazumasa Kishiki, and Hiroki Matsutani
Dept. of Information and Computer Science, Keio University
3-14-1 Hiyoshi, Kohoku, Yokohama, Japan
Email: {okuyama,tokusasi,iwata,tsukada,kishiki,matutani}@arc.ics.keio.ac.jp

Abstract

Toward machine learning based prediction services, the prediction server has multiple predictors and selects an appropriate one based on past feedbacks from the clients. In this case, three messages including request, reply, and feedback, are required for each prediction request. Packets are typically transmitted and received via a network protocol stack in OS kernel, and performance improvement can be expected by avoiding the protocol stack since it degrades the communication performance especially for small packets. We implement the prediction server using network optimization approaches including kernel-bypassing and in-NIC processing approaches. Evaluation results show that these network optimizations are beneficial to improve the prediction server performance compared to a baseline prediction server using a standard network protocol stack.

1. Introduction

Machine learning is widely used in various applications especially that require classification, regression, and clustering tasks. Since machine learning requires a huge amount of data and computation cost for learning, the prediction services that receive requests, perform predictions, and reply the results will play an important role. Since an appropriate predictor differs depending on a given environment, the prediction server is required to select an appropriate predictor based on weights of predictors. Then it performs a prediction with the selected predictor. The weights of predictors are updated based on feedbacks from clients. In [1], a prediction server, called Clipper, that handles multiple models at the same time and dynamically selects an appropriate one from them is proposed. Inspired by [1], we also focus on the prediction server with multiple predictors.

In this case, as shown in Figure 1, three messages are required for each prediction request: (1) prediction request from a client to the prediction server, (2) prediction result from the server to the client, and (3) feedback (e.g., correct or not) from the client to the server. Such messages are typically transmitted and received via a network protocol stack in OS kernel, and performance improvement can be expected by avoiding the protocol stack since it degrades the communication performance especially for small messages. In such prediction servers, we expect that optimizations on the communication are beneficial.

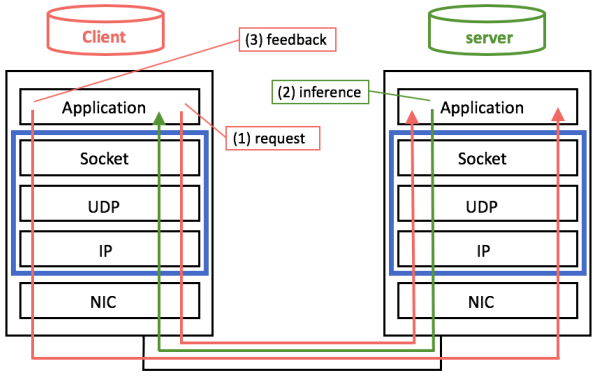

Figure 1. Three messages for prediction server

2. Network Optimization Approaches

In this paper, we optimize the network processing of the prediction servers with multiple predictors. The network optimization can be classified into three approaches: kernel-bypassing, in-kernel processing, and in-NIC processing. Figure 2 illustrates these three approaches. In the kernel-bypassing, the network protocol stack of OS is bypassed and application-specific packet processing is done at user space. This approach can be implemented with DPDK (Data Plane Development Kit) [2]. In the in-kernel processing, application-specific packet processing is performed at the entrance of the network protocol stack. In the case of Linux, it can be implemented with Netfilter framework [3]. In the in-NIC processing, application-

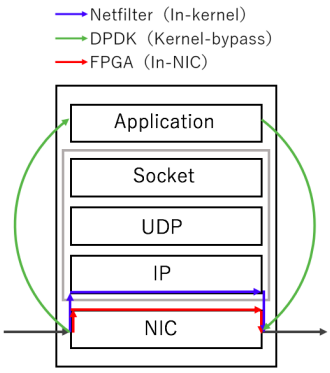

Figure 2. Three network optimization approaches

specific packet processing is implemented in Network Interface Cards (NICs) as dedicated hardware logic. It can be implemented with FPGA (Field-Programmable Gate Array) based NIC that has high-speed network interfaces, such as NetFPGA-SUME [4].

The kernel-bypassing and in-kernel processing are software-based approaches, while the in-NIC processing is a hardware-based approach. In this paper, the prediction server with multiple predictors is optimized with the kernel-bypassing and in-NIC processing approaches.

3. Design and Implementation

Baseline. We assume a simple image classification is performed as a prediction service. In this case, a prediction request includes an image and the reply includes the classification result. Then, as a feedback, the client notifies the server whether the prediction result is correct or not. An appropriate predictor is selected from ten predictors based on such feedbacks by using Exp3 or Exp4 algorithm [5]. The above-mentioned service is implemented as a user-space application in the case of baseline server.

Kernel-Bypassing. In the kernel-bypassing version, the prediction server with ten predictors and the predictor selection function based on Exp3 or Exp4 algorithm is implemented as a user-space application, and its communication is optimized by DPDK. DPDK version 17.05.2 is used to bypass the network protocol stack of Linux kernel.

In-NIC Processing. For the in-NIC processing version, the prediction server that has a prediction logic and the predictor selection logic based on Exp3 or Exp4 algorithm is implemented for NetFPGA-SUME board that has a Xilinx Virtex-7 690T FPGA and four 10Gbit Ethernet (10GbE) interfaces. Based on Reference NIC design of NetFPGA-SUME [4], the prediction server module is inserted between Input Arbiter and Output Port Lookup modules. It is implemented with C++ and synthesized with Xilinx Vivado HLS version 2016.4. Due to a resource limitation, only a single predictor was implemented.

Table 1. Specification of server machine

CPU	Intel(R) Core(TM) i5-3470S CPU @ 2.90GHz × 4
OS	Ubuntu 17.04
NIC	Intel Corporation 82599ES 10-Gigabit
FPGA NIC	NetFPGA-SUME (Xilinx Virtex-7 xc7vx690tffg1761-3)
Cable	SFP+ network connection

4. Experimental Results and Summary

Table 1 shows the machine where the baseline, kernel-bypassing, and in-NIC prediction servers are implemented. Ten predictors are trained with combinations of five optimizers (SGD, Adam, RNSProp, AdaGrad, and Nadam) and two epoch numbers (1 and 10). We use MNIST dataset for simplicity. For Exp3 and Exp4 algorithms, the γ parameter that adjusts the balance between utilization and search of information is set to 0.1. We confirmed that an appropriate predictor is dynamically selected based on feedbacks from clients by using Exp3 and Exp4 algorithms.

The baseline prediction server and those optimized by the kernel-bypassing approach and in-NIC processing approach are evaluated in terms of prediction throughput. Because we could not implement the ten predictors in the FPGA NIC due to a resource limitation, throughput of the in-NIC processing version is evaluated based on simulations. As a result, the kernel-bypassing version is 1.9 times faster than the baseline in both the Exp3 and Exp4 cases. The in-NIC processing version is 4.6 times faster than the baseline in the Exp3 case and 3.4 times faster in the Exp4 case. It is corresponding to 39.6% of the 10GbE line rate in the case of Exp3 algorithm.

These results show that the network optimizations are beneficial to enhance the prediction server performance compared to the baseline. As future work, we are still optimizing the three implementations for higher performance.

Acknowledgements This work was supported by JST CREST Grant Number JPMJCR1785, Japan.

References

[1] D. Crankshaw, X. Wang, G. Zhou, M. J.Franklin, J. E.Gonzalez, and I. Stoica, "Clipper: A Low-Latency Online Prediction Serving System," in *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI'17)*, Mar. 2017, pp. 613–627.

[2] "DPDK.org," http://dpdk.org/.

[3] "The netfilter.org Project," https://www.netfilter.org/.

[4] "The NetFPGA Project," https://netfpga.org/.

[5] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The non-stochastic multi-armed bandit problem," *SIAM Journal on Computing*, vol. 32, no. 1, pp. 48–77, 2002.