Accelerating Online Change-Point Detection Algorithm using 10GbE FPGA NIC

Takuma Iwata✉, Kohei Nakamura, Yuta Tokusashi, and Hiroki Matsutani

Keio University, 3-14-1 Hiyoshi, Kohoku-ku, Yokohama, Japan 223-8522
{iwata,nakamura,tokusasi,matutani}@arc.ics.keio.ac.jp

Abstract. In statistical analysis and data mining, change-point detection that identifies the change-points which are times when the probability distribution of time series changes has been used for various purposes, such as anomaly detections on network traffic and transaction data. However, computation cost of a conventional AR (Auto-Regression) model based approach is too high and infeasible for online. In this paper, an AR model based online change-point detection algorithm, called ChangeFinder, is implemented on an FPGA (Field Programmable Gate Array) based NIC (Network Interface Card). The proposed system computes the change-point score from time series data received from 10GbE (10Gbit Ethernet). More specifically, it computes the change-point score at the 10GbE NIC in advance of host applications. This paper aims to reduce the host workload and improve change-point detection performance by offloading ChangeFinder algorithm from host to the NIC. As evaluations, change-point detection in the FPGA NIC is compared with a baseline software implementation and those enhanced by two network optimization techniques using DPDK and Netfilter in terms of throughput. The result demonstrates 16.8x improvement in change-point detection throughput compared to the baseline software implementation. The throughput achieves 83.4% of the 10GbE line rate.

1 Introduction

Due to advances in information and communication technology, data sets exchanged over networks are growing rapidly in size and the number. As the data sets grow, high-bandwidth becomes more important for data analysis and pattern recognition. Change-point detection is a method to identify the change-points which are times when the probability distribution of time series changes. Popular applications of the change-point detection are related to a security field [13], such as detecting a sudden increase in traffic volume by computer virus and worm. It is also used in other applications fields, such as transaction data, resource management, and trend analysis [3].

In a conventional change-point detection algorithm [5], the computational cost is too high to use it as an online algorithm. ChangeFinder algorithm [8] solves this issue and can be used as an online change-point detection. However, its computational cost is still high to detect change-points from data received via

high bandwidth networks, such as 1Gbps and 10Gbps, due to heavy workload imposed to the host.

In this paper, change-point detection using ChangeFinder algorithm is implemented on an FPGA (Field Programmable Gate Array) based NIC (Network Interface Card). The proposed system computes the change-point score from time series data received from 10GbE (10Gbit Ethernet). More specifically, ChangeFinder algorithm implemented in the FPGA NIC computes the score in advance of host applications. This paper aims to reduce the host workload and improve change-point detection performance by offloading ChangeFinder algorithm from host to the NIC. As evaluations, change-point detection in the FPGA NIC is compared with a baseline software implementation and those enhanced by two network optimization techniques using DPDK and Netfilter in terms of throughput. The result demonstrates 16.8x improvement in change-point detection throughput compared to the baseline software implementation, while keeping the same change-point detection accuracy.

The rest of this paper is organized as follows. Section 2 introduces ChangeFinder algorithm and related FPGA-based accelerators. Section 3 designs the ChangeFinder module and Section 4 integrates it in the FPGA NIC. Section 5 evaluates area and throughput. Section 6 concludes this paper.

2 Background

In statistical analysis and data mining, change-point detection has been used for various purposes, such as step detection, edge detection, and anomaly detection. Since AR model is a primary approach to describe time-varying process, in this section, we will start with a conventional change-point detection based on AR model.

2.1 AR Model: A Conventional Way

Let $x_1^n = x_1, ..., x_n$ denote a time-series, and it is divided into x_1^t and x_{t+1}^n by a time point t, where$x_1^t = x_1, ..., x_t$ and $x_{t+1}^n = x_{t+1}, ..., x_n$. Assuming the k-th order AR model, the conditional probability density function of x_t is given as follows.

$$p(x_t|x_{t-k}^{t-1}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left[-\frac{(x_t - \omega_t)^T \Sigma^{-1}(x_t - \omega_t)}{2}\right], \tag{1}$$

where d and Σ denote the number of data dimensions and a covariance matrix, respectively.

ω_t is given as follows.

$$\omega_t = \sum_{i=1}^{k} \alpha_i (x_{t-i} - \mu) + \mu, \tag{2}$$

where $\alpha_1, ..., \alpha_k$ and μ are model parameters.

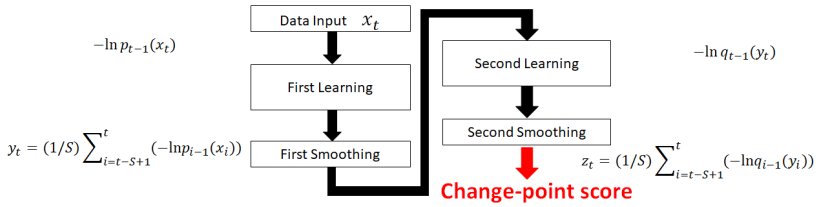

Fig. 1. Flowchart of ChangeFinder

Let $\hat{\omega}_t$ denote an estimated ω_t calculated by Equation 2 using estimated model parameters. The model fitting error for x_1^n is thus given as follows.

$$I(x_1^n) = \sum_{t=1}^{n} ||x_t - \hat{\omega}_t||^2 \tag{3}$$

Here, time t is detected as a change-point when $I(x_1^t) + I(x_{t+1}^n)$ is sufficiently small compared to $I(x_1^n)$. Although this method is simple, computation cost is $O(n^2)$ and thus cannot be used for online change-point detection.

2.2 ChangeFinder Algorithm

The above mentioned problem is addressed by SDAR (Sequentially Discounting Auto-Regression model learning) algorithm [15]. ChangeFinder algorithm employs SDAR algorithm for the online change-point detection. It has been proven to be efficient. As one of promising applications, for example, [11] utilizes the SDAR-based change-point detection for detecting fraudulent calls. Apache Hivemall [1], which is a machine learning library on Apache Hive, releases a software module of ChangeFinder. But its hardware design has not been discussed.

Overview Figure 1 shows the ChangeFinder algorithm that consists of two learning phases. Each step is described below.

Step 1 (Data Input) x_t is received at time point t.

Step 2 (First Learning) For each t, an AR model is built. More specifically, a sequence of probability density functions $p_t(x) : t = 1, 2, ...$ is obtained by the SDAR model, which will be explained later. Please note that p_{t-1} is learned based on x^{t-1}. The "outlier" score at x_t is calculated as follows.

$$Score(x_t) = -\log p_{t-1}(x_t) \tag{4}$$

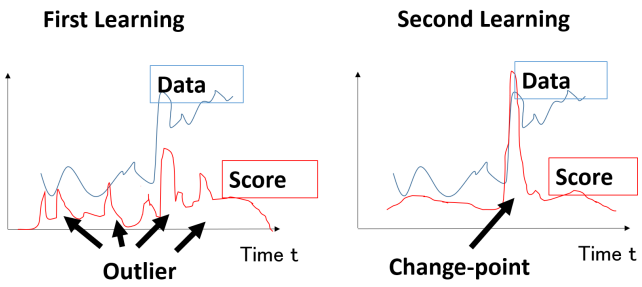

Fig. 2. Two-phase learning of ChangeFinder

Step 3 (First Smoothing) For each t, a moving average of the outlier scores (obtained in Step 2) in a time window is calculated, More specifically, a sequence of moving averages of the outlier scores $y_t : t = 0, 1, 2...$ is obtained as follows.

$$y_t = \frac{1}{T} \sum_{i=t-T+1}^{t} Score(x_i), \tag{5}$$

where T is the length of a time window.

Steps 4 & 5 (Second Learning & Smoothing) For each t, an AR model is built for the new time-series data $y_t : t = 0, 1, 2, ...$ (obtained in Step 3), and a sequence of new probability density functions $q_t(x) : t = 1, 2, ...$ is obtained by the SDAR model as well as Step 2. A smoothing step is also applied as well as Step 3. Thus, a sequence of the moving averages $z_t : t = 0, 1, 2, ...$ is obtained as follows.

$$z_t = \frac{1}{T} \sum_{i=t-T+1}^{t} (-\ln q_{t-1}(y_t)) \tag{6}$$

Here, z_t is denoted as the "change-point" score at time t. A higher change-point score z_t indicates a higher possibility of change-point at time t. As shown in Figure 2, by using the two-phase learning, outliers are eliminated by the first smoothing step and thus only the change-points where the probability distribution of time series changes are extracted.

SDAR Model SDAR model is used for online discounting learning that relies on AR model. ChangeFinder algorithm uses SDAR model to obtain the sequences of probability density functions $p_t(x)$ and $q_t(x)$. These probability density functions are derived from ω_t and Σ in Equation 1. To obtain these

parameters, SDAR model is used as follows.

$$\hat{\mu} := (1-r)\hat{\mu} + rx_t \tag{7}$$

$$C_j := (1-r)C_j + r(x_t - \hat{\mu})(x_{t-j} - \hat{\mu})^T \tag{8}$$

$$\hat{x}_t := \sum_{i=1}^{k} \hat{\omega}_i (x_{t-i} - \hat{\mu}) + \hat{\mu} \tag{9}$$

$$\hat{\Sigma} := (1-r)\hat{\Sigma} + r(x_t - \hat{x}_t)(x_t - \hat{x}_t)^T \tag{10}$$

Here, r is a discounting rate. A smaller r indicates a greater influence on past data. For each t, an weighted average $\hat{\mu}$ is updated using r and x_t in Equation 7. Based on $C_j : j = 1, ..., k$ obtained in Equation 8, estimated $\omega_1, ..., \omega_k$ (denoted as $\hat{\omega_1}, ..., \hat{\omega_k}$) are derived so that the following equation is satisfied.

$$\sum_{i=1}^{k} \omega_i C_{j-i} = C_j \tag{11}$$

Then $\hat{\omega_1}, ..., \hat{\omega_k}$ are used for Equation 9.

By introducing the discounting effect, SDAR model can be used for online learning on non-stationary time-series data. In addition, the computation cost is reduced down to $O(n)$ and thus it is preferred for online change-point detection.

2.3 Related Work

In this paper, change-point detection using ChangeFinder algorithm is implemented on an FPGA NIC. NPCUSUM (Non-Parametric Cumulative SUM) is a classic and simple change-point detection algorithm. In [4], it is implemented on a high-speed FPGA NIC in order to detect attacks from network. The network attack detection using NPCUSUM is illustrated below.

$$S_0 = 0 \tag{12}$$

$$S_n = max\{0, S_{n-1} + X_n - \hat{\mu} - \epsilon\hat{\theta}\}, \tag{13}$$

where X_n denotes input data. $\hat{\mu}$ is an estimated value of X_n before an attack, $\hat{\theta}$ is that after the attack, and ϵ is a tuning parameter. An attack from the network is detected when S_n becomes unstable and changes drastically. Although it is quite simple to implement, $\hat{\mu}$ and $\hat{\theta}$ must be known in advance, which limits the applications of NPCUSUM.

There are some prior works that present FPGA-based outlier detection that detects anomaly values (not change-points). In [6], LOF (Local Outlier Factor) algorithm is accelerated by using an FPGA. Normal data are filtered at the NIC and only anomaly data are transferred to the host machine to reduce data size.

Although our target is change-point detection to detect trend changes, ChangeFinder algorithm can be used for both the change-point detection and outlier detection. Actually, the result of the first learning phase $Score(x_t)$ is used as outlier score, while the final output z_t is used as change-point score. Please note that this paper is the first work that accelerates ChangeFinder algorithm that supports both the change-point and outlier detections by using FPGA NIC.

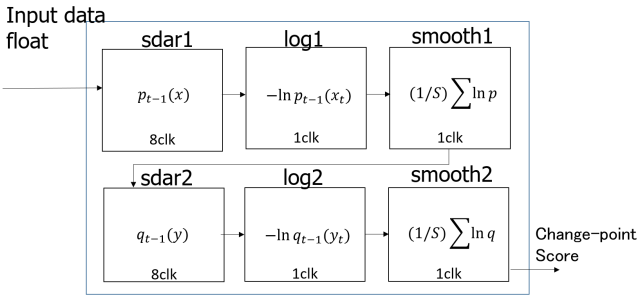

Fig. 3. Pipeline of ChangeFinder module

3 ChangeFinder on FPGA

In this section, ChangeFinder module on FPGA is illustrated. It is integrated into an FPGA NIC in Section 4. ChangeFinder module is written in C. As a high-level synthesis tool we use Xilinx Vivado HLS for the implementation.

3.1 Pipeline Structure

Figure 3 illustrates an overview of ChangeFinder module. It consists of pipelined six stages as mentioned in Section 2.2. As input data, a 32-bit float value is fed to the module. It is processed as follows.

- *sdar1*: A probability density function $p_t(x)$ for input data x_t in the first learning phase is computed.
- *log1*: A logarithmic loss of the probability density function is computed as an outlier score.
- *smooth1*: A moving average y_t of the outlier scores is computed as a result of the first learning phase.
- *sdar2, log2, and smooth2*: A change-point score z_t is computed by the same operations as the first phase.

These stages are operated at 125MHz. In Figure 3, the number in each pipeline stage indicates the minimum interval between two input data in the stage. For example, "1clk" indicates that new data can be accepted in every cycle. Thus, *log1*, *smooth1*, *log2*, and *smooth2* can accept new data every cycle, while *sdar1* and *sdar2* accept new data in every eight cycles. Please note that *sdar1* and *sdar2*, *log1* and *log2*, and *smooth1* and *smooth2* are identical, respectively. In the following, *sdar1*, *log1*, and *smooth1* modules are illustrated.

3.2 Detail of Each Module

Figure 4 shows *sdar* module. Its inputs are r and x_t. r is a discounting parameter. Based on it, $(1-r)$ is computed. x_t is an input float value. The outputs are $\hat{x}$ and $\hat{\Sigma}$. $\hat{x}$ is an estimated value of x_t and $\hat{\Sigma}$ is that of Σ_t.

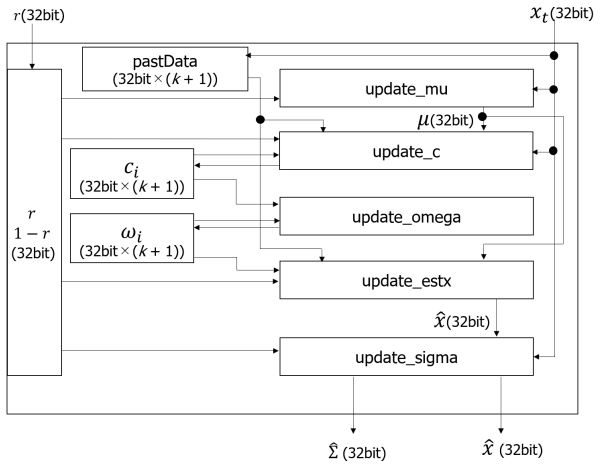

Fig. 4. *sdar* module

As shown, *sdar1* is further divided into five pipelined submodules: *update_mu*, *update_c*, *update_omega*, *update_estx*, and *update_sigma*. x_t is stored in $(k+1)$ 32-bit registers (pastData in the figure) to refer to past k data, where k is the order of AR model. C_i and ω_i are accumulated in $(k+1)$ 32-bit registers, respectively.

x_t, r, and $(1-r)$ are fed to *update_mu* submodule. *update_mu* submodule is corresponding to Equation 7 and computes μ. *update_c* submodule is corresponding to Equation 8 and updates C_i registers. *update_omega* submodule updates ω_i registers based on Equation 11. *update_estx* submodule is corresponding to Equation 9. It computes $\hat{x}_t$. Finally, *update_sigma* submodule is corresponding to Equation 10. It computes $\hat{\Sigma}$.

These five submodules work in a pipelined manner. As a result, *sdar1* module accepts new data x_t in every eight cycles.

Log module performs a logarithmic computation as in Equation 4. It is fully pipelined and can accept new data in every cycle.

Then *smooth* module computes a moving average of recent T data as in Equation 5. The maximum T is set to 16 in our design. It is also fully pipelined and can accept new data in every cycle.

4 ChangeFinder on FPGA NIC

ChangeFinder module is implemented on a 10GbE FPGA NIC. It is denoted as ChangeFinder NIC in this paper. It performs change-point detection for each numerical value coming from the 10GbE network. The change-point score computed at the NIC is passed to a host application so that it can identify changes in given time series data.

In this paper, NetFPGA-SUME [17] is adopted as a 10GbE FPGA NIC. It has four 10GbE interfaces. Packets received by these interfaces are processed at

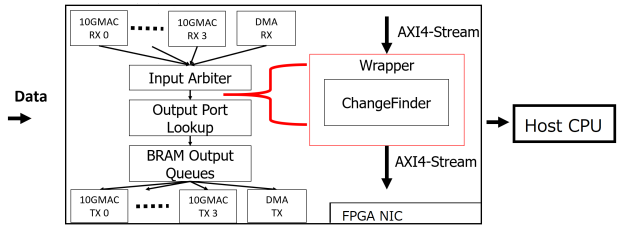

Fig. 5. ChageFinder on FPGA NIC

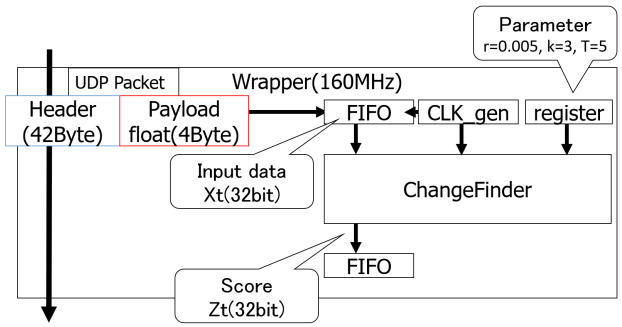

Fig. 6. Connection between wrapper and ChangeFinder modules

an on-board FPGA and the results are transferred to a host machine via a PCI-Express Gen3 x8 interface. We use 10GbE MAC IP core provided by Xilinx. We also use Reference NIC design provided by NetFPGA project [2] as a standard 10GbE NIC function.

We implemented a wrapper module along the datapath of Reference NIC design so that all the received packets go through the wrapper module. Then ChangeFinder module designed with Xilinx Vivado HLS is implemented inside the wrapper module. Figure 5 shows a block diagram of ChangeFinder NIC consisting of ChangeFinder module and Reference NIC. In Reference NIC, packets received by the four 10GbE interfaces (i.e., RX0 to RX3) and host DMAC are arbitrated at Input Arbiter module. Then, an output port is selected among the four 10GbE interfaces (i.e., TX0 to TX3) and host DMAC for each packet. Packets are stored and transmitted via BRAM Output Queues corresponding to the selected output ports. Packets are transferred between these modules as AXI4 stream [14]. The wrapper module is implemented between Input Arbiter and Outport Lookup modules. We use UDP/IP as transport/network layer protocols. ChangeFinder module computes a change-point score for each incoming packet destined to a specific UDP port. All the other packets including ARP and ICMP just skip the wrapper module without any additional delay.

Figure 6 illustrates the wrapper module and input/output signals of ChangeFinder module. A clock generator of 125MHz and parameter registers are im-

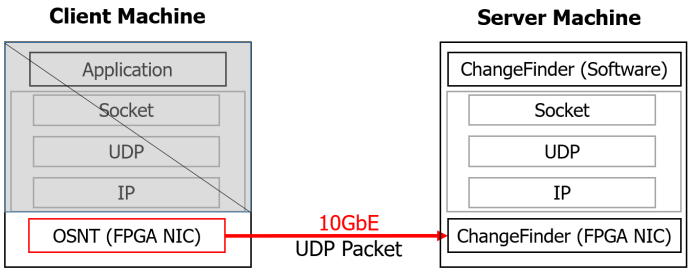

Fig. 7. Evaluation environment for throughput

plemented for ChangeFinder module. In addition, an input asynchronous FIFO buffer is inserted between them, because ChangeFinder module is operating at 125MHz and Reference NIC is operating at 160MHz.

The wrapper module identifies packets that contain sample data. Then it extracts the sample data and feeds them to ChangeFinder module. The packet conveys sample data x_t in a 32-bit float format in a UDP payload. UDP packets with a specific destination port number are extracted as sample packets and they are fed to the input FIFO buffer. As tuning parameters, AR model order k, discounting rate r, and smoothing window size T are stored in the parameter registers. They are fed to ChangeFinder module in addition to input data x_t when ChangeFinder module is ready. Then the change-point score z_t is computed and fed to an output asynchronous FIFO buffer. The score z_t can be embedded in the original packet and passed to host application.

5 Evaluations

5.1 Evaluation Environment

The target 10GbE FPGA NIC is NetFPGA-SUME that has a Xilinx Virtex-7 XC7VX690T FPGA and four SFP+ 10GbE interfaces. It is mounted to a host machine via PCI-Express Gen3 x8 interface. We use Xilinx Vivado HLS version 2016.4 for the implementation. Reference NIC part is operating at 160MHz, while the proposed ChangeFinder module is running at 125MHz.

Figure 7 shows the evaluation environment using two machines and Table 1 shows their specification.

The client and server machines are connected by a SFP+ direct attached cable for 10GbE. The client machine has an FPGA NIC with OSNT (Open Source Network Teste) installed, which is a hardware packet generator, and sends packets to the server. In the server machine, the proposed ChangeFinder module is implemented on the FPGA NIC and processes incoming time series data. We measured the number of sample data processed at the ChangeFinder module per a second as throughput.

Table 1. Machines used in the environment

	Server (host) machine	Client machine
CPU	Intel Core i5-4460	Intel Core i5-4460
OS	Ubuntu 14.04	CentOS 6.6
NIC	NetFPGA-SUME (Proposal) Intel X520-DA2 (Software)	NetFPGA-10G for OSNT

5.2 Area Utilization

Table 2 shows area utilization of ChangeFinder NIC including ChangeFinder module and Reference NIC. As shown in Table 2, ChangeFinder module consumes 5.1 to 12.1% of the FPGA resources. Even with 10GbE NIC functionality, the entire resource utilizations are less than or equal to 18.8%.

Table 2. Resources used in ChageFinder NIC

	ChangeFinder	ChangeFinder + Reference NIC
DSP	437 (12.1%)	437 (12.1%)
FF	44,519 (5.1%)	100,403 (11.6%)
LUT	45,836 (10.6%)	81,517 (18.8%)

5.3 Throughput

As mentioned above, OSNT at the client machine transmits time series data at 10GbE line rate to the server machine, and the number of sample data processed in one second at the server machine is measured as throughput.

The proposed ChangeFinder NIC is compared with three software-based counterparts implemented in C: Baseline, DPDK, and Netfilter. In Baseline, a ChangeFinder program is running on the application layer. In DPDK, although the ChangeFinder program is running on the application layer, the program directly accesses the NIC without kernel UDP/IP stack. In Netfilter, the ChangeFinder program is implemented as a kernel module.

Figure 8 shows their throughput. The proposed ChangeFinder module is denoted as FPGA(sim) and the ChangeFinder NIC consisting of ChangeFinder and Reference NIC modules is denoted as FPGA(actual). FPGA(sim) throughput is derived by the number of cycles, pipeline structure (i.e., interval), and operating frequency of the ChangeFinder module. FPGA(actual) is the measured throughput. The proposed FPGA(actual) achieves 16.8x throughput improvement compared to Baseline. It is much higher than those with software-based optimizations by DPDK and Netfilter.

In practical use cases, a specific field of received packets is extracted and fed to ChangeFinder module. In this experiment, we used 46-Byte UDP/IP packets containing a single 32-bit float value. This assumption is pessimistic in terms of

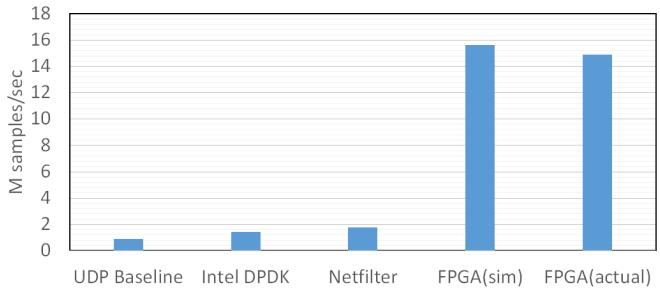

Fig. 8. Throughput of change-point detection [samples / sec]

throughput. Since internal data width of Reference NIC is 256 bits, these *sdar* modules are not bottleneck when packet length is greater than or equal to 256 Bytes. Considering the packet length of 46 Bytes[1], the proposed FPGA(actual) achieves 83.4% of 10GbE line rate.

6 Summary

Toward anomaly detection, change-point detection is used to look for change in a probability distribution of time series, while outlier detection is used to look for entity being away from the mean of a probability distribution. ChangeFinder algorithm based on SDAR model supports both the outlier and change-point detections and can be used for online use. This paper is the first work that accelerates ChangeFinder algorithm using FPGA and integrates it into NetFPGA-SUME for high-speed change-point detection at 10GbE NICs. The proposed ChangeFinder NIC is compared to a UDP baseline and two software-based optimizations, i.e., DPDK and Netfilter. The throughput is much higher than these counterparts and it is 16.8x higher than the UDP baseline. The throughput is corresponding to 83.4% of the 10GbE line rate. To achieve full 10GbE line rate or more, as future work, we are considering the possibility to use multiple ChangeFinder modules while keeping their consistency. A demonstration video of current design can be found in [16].

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

References

1. Apache Hivemall, `http://hivemall.incubator.apache.org/`
2. The NetFPGA Project, `http://netfpga.org/`

[1] In addition to the packet length, Ethernet preamble, FCS, and IFG are also considered.

3. Aminikhanghahi, S., Cook, D.J.: A Survey of Methods for Time Series Change Point Detection. Knowledge and Information Systems 51(2), 339–367 (May 2017)
4. Benacek, P., Blazek, R.B., Cejka, T., Kubatova, H.: Change-Point Detection Method on 100 Gb/s Ethernet Interface. In: Proceedings of the ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS'14). pp. 245–246 (Jun 2014)
5. Guralnik, V., Srivastava, J.: Event Detection from Time Series Data. In: Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD'99). pp. 33–42 (Aug 1999)
6. Hayashi, A., Matsutani, H.: An FPGA-Based In-NIC Cache Approach for Lazy Learning Outlier Filtering. In: Proceedings of the International Conference on Parallel, Distributed, and Network-Based Processing (PDP'17). pp. 15–22 (Mar 2017)
7. Hayashi, A., Tokusashi, Y., Matsutani, H.: A Line Rate Outlier Filtering FPGA NIC using 10GbE Interface. ACM SIGARCH Computer Architecture News 43(4), 22–27 (Sep 2015)
8. Jun-ichi Takeuchi and Kenji Yamanishi: A Unifying Framework for Detecting Outliers and Change Points from Time Series. IEEE Transactions on Knowledge and Data Engineering 18(4), 482–492 (Apr 2006)
9. Kawahara, Y., Sugiyama, M.: Change-Point Detection in Time-Series Data by Direct Density-Ratio Estimation. In: Proceedings of the SIAM International Conference on Data Mining (SDM'09). pp. 389–400 (Apr 2009)
10. Pu, Y., Peng, J., Huang, L., Chen, J.: An Efficient KNN Algorithm Implemented on FPGA Based Heterogeneous Computing System Using OpenCL. In: Proceedings of the International Symposium on Field-Programmable Custom Computing Machines (FCCM'15). pp. 167–170 (May 2015)
11. Saaid, F., Nur, D., King, R.: Change Points Detection of Vector Autoregressive Model using SDVAR Algorithm. In: Proceedings of the 5th Annual ASEARC Conference. pp. 18–21 (Feb 2012)
12. Urabe, Y., Yamanishi, K., Tomioka, R., Iwai, H.: Real-Time Change-Point Detection Using Sequentially Discounting Normalized Maximum Likelihood Coding. In: Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'11). pp. 185–197 (May 2011)
13. Wang, H., Zhang, D., Shin, K.G.: Change-Point Monitoring for the Detection of DoS Attacks. IEEE Transactions on Dependable and Secure Computing 1(4), 193–208 (Oct 2004)
14. Xilinx: AXI Reference Guide (2011)
15. Yamanishi, K., ichi Takeuchi, J.: A Unifying Framework for Detecting Outliers and Change Points from Non-Stationary Time Series Data. In: Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD'02). pp. 676–681 (Jul 2002)
16. YouTube: Accelerating ChangeFinder using 10Gbps FPGA NIC, `https://www.youtube.com/watch?v=wgTcBfkE5hY`
17. Zilberman, N., Audzevich, Y., Covington, G.A., Moore, A.W.: NetFPGA SUME: Toward 100 Gbps as Research Commodity. IEEE Micro 34(5), 32–41 (Sep 2014)