Reconfigurable FPGA-Based K-Means/K-Modes Architecture for Network Intrusion Detection

Lucas Andrade Maciel, Matheus Alcântara Souza, and Henrique Cota de Freitas

***Abstract*—Over the years, the amount of data shared between users from different areas have grown considerably. Consequently, so did network attacks. Security monitoring strategies must classify information types on networks quickly and effectively. Intrusion Detection Systems have been proposed with Machine Learning techniques and High-Performance Computing to avoid security anomalies. Thus, FPGA devices are good candidates to improve performance and energy efficiency. In this brief, we propose a reconfigurable FPGA-based K-means/K-modes architecture to accelerate the data clustering for network intrusion detection. We evaluated our approach over NSL-KDD data set and the results showed that K-means and K-modes can achieve up to 15x and 994x more operations per Watt than parallel software versions.**

***Index Terms*—K-means, K-modes, FPGA, energy efficiency, intrusion detection system.**

I. Introduction

THE INTERNET has become a great way to share information between many users and machines. Large data volumes from diverse areas are transferred on local and global networks every second, increasing data security concerns. This leads to investments in monitoring and security strategies. To reduce cybernetic attacks, Intrusion Detection Systems (IDSs) have been proposed [1]. Those systems protect the information on networks, identifying potential attacks. They alert system administrators to possible intrusions, or even prevent the arrival of malicious packets. Intrusion detection can be performed in two ways: (i) The signature-based detection identifies intrusions by comparing evaluated data and previously know stored ones. (ii) The anomaly-based detection identifies records that have different characteristics from normal behavior.

To improve IDS efficiency, Machine Learning techniques have been applied. Machine Learning unveiled a wide range of possibilities for detecting intrusions [2]. In this context, K-means and K-modes clustering methods are constantly used. The clusters formed by these algorithms can represent types of anomalies and normal accesses [3]. This allowed IDSs to evaluate new records according to similarities between data.

Manuscript received May 22, 2019; revised August 11, 2019; accepted September 1, 2019. Date of publication September 6, 2019; date of current version August 4, 2020. This work was supported in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, Brazil, under Grant 001, in part by the Conselho Nacional de Desenvolvimento Científico e Tecnológico, and in part by the Fundação de Amparo à Pesquisa do Estado de Minas Gerais. This brief was recommended by Associate Editor J. Wu. *(Corresponding author: Lucas Andrade Maciel.)*

The authors are with the Department of Computer Science, Pontifical Catholic University of Minas Gerais, Belo Horizonte 31980-110, Brazil (e-mail: lamaciel@sga.pucminas.br).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSII.2019.2939826

The large amount of numerical or categorical data from network traffic requires high performance to be processed. Powerful parallel architectures, such as Graphics Processing Units (GPUs) and Intel Xeon Phi have been widely used to accelerate clustering algorithms. However, these devices are composed by complex circuits to deliver high floating point performance, consuming considerable energy [4]. Thus, efficient architectures based on Field Programmable Gate Arrays (FPGAs) should be considered [5]. In this brief, our goal is to propose a high-performance and energy-efficient reconfigurable FPGA-based K-means/K-modes architecture for network intrusion detection.

In summary, our contributions to the state-of-the-art are:

- A K-means/K-modes architecture that supports numerical or categorical input data with 64 bits;
- Flexibility to change the parameters of both algorithms (number of centroids, points, iterations and type of algorithm) at runtime;
- An energy-efficient and high-performance reconfigurable architecture compared to the parallel software versions.

This brief is organized as follows. Section II provides an explanation of the algorithms. Section III presents related work. In Section IV, we show our proposed hardware architecture. The evaluation methodology is presented in Section V. The results obtained are shown in Section VI and in Section VII we present our final considerations.

II. Background

K-means is a clustering algorithm which groups data into k clusters, based on their similarities [6]. The basic idea is to, from a set of n data points with d attributes, iteratively associate these points to the centers of each cluster, named centroids, based on their Euclidean distance. The algorithm starts initializing k centroids with strategic values. Afterwards, the distance between each point and a centroid is calculated, and the point is assigned to its closest centroid [7]. In the last step, the centroids are updated, based on the arithmetic mean of the attributes from data points associated to them. The algorithm performs these steps while centroids continue changing their values or until it reaches a certain number of iterations. Thus, for each iteration t, all of these steps are executed with computational complexity of $O(tnkd)$. At the

1549-7747 © 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: PUC MG. Downloaded on March 06,2024 at 13:23:14 UTC from IEEE Xplore. Restrictions apply.

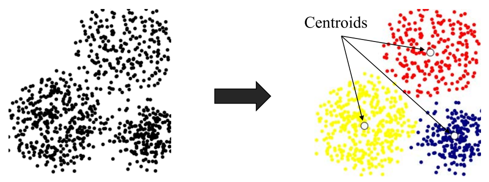

Fig. 1. Clustering example with k = 3.

end, the output is a mapping of centroids and their assigned data points, representing the generated clusters.

K-modes works similarly as K-means, but using categorical data, i.e., non-numeric data, as input [8]. However, the distances between points and centroids are defined by dissimilarity techniques (e.g., simple matching). Furthermore, the arithmetic mode is used to update centroids attributes.

Figure 1 shows a clustering example with $k = 3$. The K-means and K-modes algorithms are able to group data with similar characteristics generated by several contexts. Thus, if network information is used as input to them, the generated groups can be tagged as malicious or normal accesses. This way, Intrusion Detection Systems based on clustering methods are widely used to find new patterns of anomalies to classify and detect unknown intrusions.

III. Related Work

In this section, we present related work and a brief comparison in Table I with focus on network intrusion detection.

A method called MinMax increase the quality of initial centroids in K-means [9]. This improved the detection rate with a reduction in false positives. A multi-level model [10] for an IDS based on K-means clustering, Support Vector Machine (SVM) and Extreme Learning Machine (ELM), was proposed to cluster training data using K-means, producing new data sets to enhance SVM and ELM. Another work [11] performed anomaly detection in high-dimensional data sets using Bayes networks, Gaussian mixture model (GMM), K-modes and active learning. The authors identified subgroups of relevant dimensions, which could be classified as attacks.

A modified approach using K-means aimed to apply the clustering method for IDS [12]. After test their approach over the KDDCup'99 data set, the authors showed an efficiency of up to 95% in intrusion detection, presenting inter-cluster distances larger than the traditional version, thus forming more distinct and higher-quality groups. In other experiment [13], a clustering method for IDS using K-means with Mini Batch and Principal Component Analysis (PCA) was used. The approach developed was compared to traditional versions of K-means and PCA, also using the KDDCup'99 data set. The model was efficient in big data scenario, presenting faster execution times than classical ones. A Naive Bayes classifier [14] was also used along with K-means, and tested over KDDCup'99. The purpose was to categorize data into normal instances and potential attacks. Likewise, a Random Forest classifier [15] was also used with K-means, presenting an accuracy rate of 99.86% when detecting intrusions on the NSL-KDD data set.

Although those works achieved good results, they considered scenarios with low computational power and energy efficiency, which cannot be neglected currently. Whit that in mind, a hardware architecture proposal of the K-means algorithm was proposed [16]. This hardware was capable of forming clusters that identify characteristics of four types of attacks. Using a 40 MHz FPGA model, 32-bit input data and a fixed number of iterations, the authors showed that the proposed architecture is up to 300× faster than a software version when evaluating syntactically generated records existing in the DARPA data set. This result represents a reduction in processing time of 99%, similar to this brief that achieved a reduction of 91%. In terms of logic elements, this related work shows an FPGA occupation of 58484 elements (only K-means), value just over half of 106824 elements achieved by our hybrid K-means/K-modes implementation.

TABLE I
Related Work Overview

Work	Device	Methods	Characteristics
[9]	CPU	K-means	Evaluate Detection Rate, NSL-KDD data set
[10]	CPU	K-means SVM and ELM	Evaluate Detection Rate, KDDCup'99 data set
[11]	CPU	K-modes Active Learning	Evaluate 7 clusters, Synthetic data set
[12]	CPU	K-means	Evaluate 2 clusters, KDDCup'99 data set
[13]	CPU	K-means PCA	Evaluate Time, KDDCup'99 data set
[14]	CPU	K-means Naive Bayes	Evaluate 2 clusters, KDDCup'99 data set
[15]	CPU	K-means Random Forest	Evaluate 2 clusters, NSL-KDD data set
[16]	FPGA	K-means	32-bit data, FPGA with 40 MHz, Evaluate Time, DARPA data set
[17]	FPGA	K-means	16-bit data, 82mW and 162 MHz, Evaluate Energy, Not related to IDS
[18]	CPU + FPGA	K-means	128-bit data, 126W (CPU+FPGA), Evaluate Time/Energy, Not related to IDS
[19]	FPGA	K-means	FPGA with 250 MHz, Evaluate Time/Energy, Not related to IDS
[20]	65nm CMOS	K-means	16-bit data, 41mW and 250 MHz, Evaluate Energy, Not related to IDS
Our Work	FPGA	K-means K-modes	64-bit data, Reconfiguration Inputs, 1.5W and 50 MHz, Evaluate Cycles/Energy, Evaluate up to 8 clusters, NSL-KDD data set

Authorized licensed use limited to: PUC MG. Downloaded on March 06,2024 at 13:23:14 UTC from IEEE Xplore. Restrictions apply.

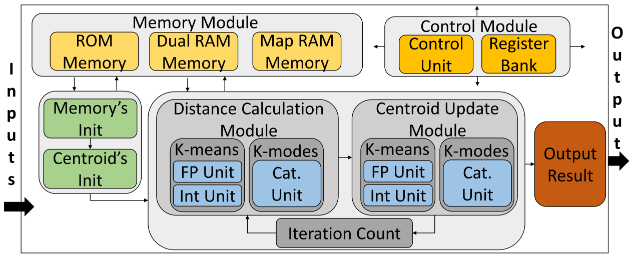

Fig. 2. Reconfigurable K-means/K-modes architecture.

There are other proposals related to K-means implemented on FPGA [17]–[19] and CMOS [20]. They are not related to IDS, but highlight the importance of hardware-based K-means as accelerator for different applications.

This brief differs from others, since we propose a reconfigurable FPGA-based K-means/K-modes architecture with 64-bit data. Our approach offers higher performance and lower power consumption in the intrusion detection scenario than software approaches. In addition, it is able to evaluate up to 8 clusters with both algorithms, showing a more robust FPGA solution than those presented in the previous works.

IV. FPGA-Based K-Means/K-Modes Architecture

Our reconfigurable K-means/K-modes architecture is based on interconnected blocks, managed by a central control unit, as shown in Figure 2. This architecture is described in VHDL (VHSIC Hardware Description Language) and its basic blocks are: (i) Register Bank which stores in internal registers the available operations; (ii) Iteration Count that determines the stop condition of the algorithms; (iii) Output Result that receives the mappings.

The inputs are clock, reset, the chosen algorithm (K-means-Integer, K-means-FloatingPoint or K-modes), number of points, centroids, features and iterations. The output is the mapping of centroids and their points. We use 64-bit data inputs with different meanings depending on the chosen algorithm. For K-modes, categorical attributes are used, thus the 64-bit word contains eight attributes of eight bits each. On the other hand, with K-means the attributes are numerical, thus we define 2 attributes per 64-bit words. Floating-point attributes are encoded in the IEEE 754 simple-precision standard.

In order to keep data and accelerate access during the execution of the algorithms, internal memories were used. Our prototype first uses a ROM to store the data set at compilation time. This memory simplifies the hardware prototype on FPGA. In spite of that, the future purpose is to use external memories to store the data set. In a second moment, the prototype uses a Dual RAM to dynamically store the points and centroids. This Dual RAM has two input and two output channels, enabling the processing of two information in the same clock cycle. Finally, a Map RAM is used to keep the final mapping of data points.

At each clock cycle, the control unit selects the next block that will be activated, following the execution flow depicted in Figure 3. The data initialization is performed by a module composed by Memory (1) and Centroid Initialization (2) blocks. The former reads an existing data in ROM and stores

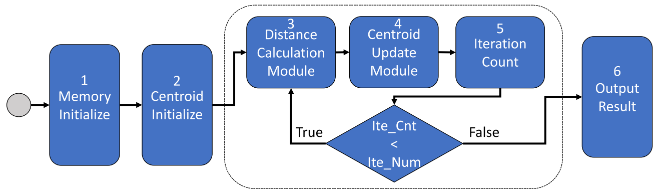

Fig. 3. Hardware state Diagram.

it in the first available address in the Dual RAM. The latter assigns the values of k initial points to centroids. This strategy reduces hardware complexity and latency.

Besides, K-means/K-modes architecture is divided into two main modules: distance calculation (3) and centroid update (4). The first calculates the distance d, in which n is the number of attributes, p_i and c_i are the i-th attributes of point p and centroid c. This block consists of three components, one for fixed-point data, another for floating-point data and another for categorical data.

$$d(p,c)^2 = \sum_{i=1}^{n} |p_i - c_i|^2 \tag{1}$$

Our implementation allows the parallel execution of two attributes when K-means is chosen and eight attributes for K-modes, reducing the complexity order of the algorithms to $O(\frac{tnkd}{a})$, where a represents the amount of attributes executed in parallel. They have a comparator unit which adds the difference between the attributes of points and centroids to an accumulator. Then, the result is stored in Map RAM. We adapted the Euclidean distance, so as to remove the square root computation, as shown in Equation 1. This computation adds greater complexity to the hardware, and removing it does not change the final mapping. With categorical data, numerical-based distance measurements cannot be used. Thus, we use a simple matching dissimilarity metric from Equation 2 to find the distance between such attributes. The metric assigns 0 when the attributes are equal and 1 when they are different, as Equation 3.

$$d(p,c) = \sum_{i=0}^{n} \delta(p_i, c_i) \tag{2}$$

where

$$\delta(p_i, c_i) = \begin{cases} 0, & if\,(p_i = c_i) \\ 1, & if\,(p_i \neq c_i) \end{cases} \tag{3}$$

The centroid update module has two blocks, one for K-means and other for K-modes. To compute the mean, the K-means block uses arrays to store the sum of attributes from points assigned to a given centroid. The K-modes block uses eight buffers of 256 positions to store the frequency of each possible value of attributes. The frequencies are used to find the largest element of each buffer and calculate the mode. After its execution, this module store updated centroids in the lowest region of the Dual RAM. These computations are carried out until centroids do not change anymore or until a given number of iterations have been reached (5). Lastly, the final mapping is sent to the output result block (6), which shows the clusters and their assigned points.

Authorized licensed use limited to: PUC MG. Downloaded on March 06,2024 at 13:23:14 UTC from IEEE Xplore. Restrictions apply.

TABLE II
FPGA OCCUPATION

Elements	Available	Used	Used (%)
Multipliers	532	65	12%
Registers	114480	31864	28%
Logics	114480	106824	93%
Memory bits	3981312	2270602	57%

TABLE III
NUMBER OF CYCLES (MILLIONS) PER NUMBER OF POINTS (PTS) AND CENTROIDS (CTR)

Ctr	Pts	K-means				K-modes	
		FPGA Int	Xeon Int	FPGA FP	Xeon FP	FPGA Cat	Xeon Cat
2	2^8	3.546	39.904	5.567	38.031	0.046	36.711
	2^{10}	2.884	35.755	6.724	43.754	0.056	39.676
	2^{12}	7.417	43.672	9.079	45.355	0.057	38.990
4	2^8	5.011	22.774	6.685	23.600	0.047	37.640
	2^{10}	3.158	34.730	8.679	39.706	0.056	38.293
	2^{12}	11.537	40.519	15.917	44.422	0.057	44.537
8	2^8	5.217	35.914	7.274	41.448	0.047	41.002
	2^{10}	9.055	43.926	14.851	39.212	0.057	41.097
	2^{12}	12.791	50.693	16.201	54.901	0.058	38.438

V. EVALUATION METHODOLOGY

We compared our architecture against OpenMP (Open Multi-Processing) parallel software versions with the same input sizes and operation definitions. The OpenMP versions were executed in a system with 6 threads on a 6-core 2.40GHz Intel Xeon E5-2620 processor. Our architecture was deployed in a DE2-115 board with Intel Cyclone IV-E FPGA EP4CE115F29C7, described in VHDL and synthesized by Quartus Prime Lite 16.1 software. OpenMP has a good abstraction for programmers and it is a easy way for a software solution focused on shared memory, but according to our results in Section VI, there is a considerable performance and energy improvement by the use of an FPGA as accelerator.

The NSL-KDD [21] was used in our experiments. It is a widely used data set for tests in network intrusion detection scenario. This data set is an update of the KDD'99 base solving the redundancy problems of the original set. It contains examples of 22 types of attack and normal access records, with a set of 125973 data for training and 22543 data for testing. Each registry has 41 numerical and categorical attributes, classified in *basic*, *content* and *traffic* resources.

Our architecture receives as input the data from NSL-KDD data set with 3 configured data groups. The first one contains integer attributes: the number of connections in the same destination and service; the number of bytes transferred from the data source to the destination; and from the destination to the source. The second group consists of floating-point attributes: the percentages of connections in the same service; and this percentage in different services. The third group is composed of categorical values: the protocol used; the destination network service; login; and connection status.

For every group, we set a maximum number of iterations equal to 5; a number of centroids varying between 2, 4 and 8; and total of points between 2^8, 2^{10} and 2^{12}. We set number of attributes as 4. Thus, we selected the most significant attributes from the records with more robust representation, good variability and higher non-zero values [22].

To evaluate the performance, we compared the clock cycles spent on running K-means-Int (Integer), K-means-FP (Floating Point) and K-modes-Cat (Categorical Data). The energy consumption for the FPGA was evaluated using the Intel PowerPlay EPE. For the Xeon processor, we used the PAPI framework [23]. The energy efficiency was measured using millions of operations per second (MOPS) per power consumption (Watts), i.e., MOPS/Watt. The number of operations was calculated using the product between number of centroids, points, attributes and iterations.

TABLE IV
ENERGY CONSUMED (mJ) PER NUMBER OF POINTS (PTS) AND CENTROIDS (CTR)

Ctr	Pts	K-means				K-modes	
		FPGA Int	Xeon Int	FPGA FP	Xeon FP	FPGA Cat	Xeon Cat
2	2^8	106.1	1521.7	166.6	1441.5	1.401	1393.8
	2^{10}	86.3	1341.8	201.2	1649.3	1.685	1506.9
	2^{12}	221.9	1674.0	271.7	1721.0	1.696	1484.6
4	2^8	149.9	877.7	200.0	900.2	1.405	1445.4
	2^{10}	94.5	1320.3	259.7	1508.8	1.689	1477.4
	2^{12}	345.2	1542.8	476.2	1682.4	1.696	1710.4
8	2^8	156.1	1365.5	217.7	1572.9	1.405	1565.2
	2^{10}	270.9	1677.7	444.4	1505.3	1.691	1557.6
	2^{12}	382.7	1924.6	484.7	2079.3	1.697	1474.5

VI. RESULTS

Table II shows the FPGA occupation, considering the total number of registers, multipliers, logic elements and memory bits of the synthesized hardware. The large number of operations required to execute the K-means (Int/FP) and K-modes (Cat) on the chosen FPGA explains the high occupation numbers in the logic elements.

The K-means columns in Table III present the execution cycles of the proposed architecture on the FPGA compared to the parallel software implementation on the Xeon processor. The results show that the FPGA performs better than Xeon for K-means application, carrying out operations with a smaller number of cycles, with a reduction between 72% and 91% for integer (Int) data and between 62% and 85% for floating-point (FP) data. The two last columns in Table III shows that the FPGA for K-modes (Categorical Data) requires up to 97% less cycles in comparison to Xeon processor. The number of clock cycles spent in each approach of this evaluation showed that increasing the input size does not reduce the performance of the FPGA approach, maintaining on average a proportional difference compared to OpenMP code.

The energy consumption, in millijoules, was evaluated, using the FPGA at 50MHz. FPGA-Int consumes between 78% and 94% less energy than the Xeon-Int, and FPGA-FP consumes between 72% and 88% less energy when compared to the Xeon-FP, as shown in Table IV. The same evaluation was performed to K-modes-Cat that the FPGA consumes up to

Authorized licensed use limited to: PUC MG. Downloaded on March 06,2024 at 13:23:14 UTC from IEEE Xplore. Restrictions apply.

TABLE V
MOPS/WATT PER NUMBER OF POINTS (PTS) AND CENTROIDS (CTR)

Ctr	Pts	K-means				K-modes	
		FPGA Int	Xeon Int	FPGA FP	Xeon FP	FPGA Cat	Xeon Cat
2	2^8	0.953	0.066	0.607	0.070	111.433	0.112
	2^{10}	4.687	0.301	2.010	0.245	188.401	0.210
	2^{12}	7.290	0.966	5.955	0.940	567.532	0.648
4	2^8	1.272	0.217	0.953	0.211	220.471	0.214
	2^{10}	8.074	0.577	2.937	0.505	369.937	0.422
	2^{12}	8.840	1.977	6.407	1.813	1110.783	1.101
8	2^8	2.370	0.270	1.700	0.235	439.094	0.394
	2^{10}	5.461	0.882	3.330	0.983	732.912	0.795
	2^{12}	15.464	3.075	12.210	2.846	2196.243	2.527

99% less energy than the Xeon. This is due to two factors: the difference in the frequency of operation and the number of cycles spent by each platform. This way, it is worth remembering that the software version uses more complex computational units and a frequency higher than the FPGA. Among them, the most robust systems of memory, multicore and multithread support increase energy consumption.

The energy efficiency of the proposed architecture is presented in Table V in number of million operations per second (MOPS) per Watt. A higher amount of MOPS/Watt is obtained as the workload increases. We evaluate our FPGA-based hardware model, which presented a total average power consumption of 1.50 Watts, while thermal design power (TDP) from Xeon E5-2620 it is on average 59 Watts. For the small data set, FPGA-based K-means performs up to 15x (integer) and 8x (floating point), more operations per Watt than the Xeon processor. With larger data set this gain reaches up to 8x and 7x. The results for K-modes (categorical data) shows the FPGA running 994x more MOPS/Watt than the Xeon thanks to the parallelization of operations during the calculation of simple matching. Finally, categorical data can give to an FPGA-based clusterization a performance advantage. Our results show that K-modes-Cat is up to 220x, 225x and 185x better than K-means-Int, which is more efficient than K-means-Float, for cycles, energy and MOPS/Watt, respectively.

VII. CONCLUSION

We presented the design of an architecture for K-means and K-modes clustering algorithms on an FPGA device. It supports 64-bit input data, with 32-bit for numerical attributes and 8-bit categorical attributes, allowing to change its main input parameters at runtime. We described an evaluation of the performance, energy consumption and energy efficiency of our hardware, when compared against an Intel Xeon processor. Our FPGA-based proposals proved to be better than OpenMP-based versions with an increase in MOPS/Watt of up to 15x for K-means and 994x for K-modes. The results showed that our hardware presented less number of cycles (up to 91%) and less energy consumption (up to 99%) than compared approaches. Thus, our architecture can enhance the efficiency required by intrusion detection systems. For future work, we intend to use more robust FPGA devices (e.g., Intel Arria 10); use external memories instead of ROM; and implement a hybrid model on the FPGA capable of classifying real-time data.

REFERENCES

[1] A. W. Al-Dabbagh, Y. Li, and T. Chen, “An intrusion detection system for cyber attacks in wireless networked control systems,” *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 65, no. 8, pp. 1049–1053, Aug. 2018.

[2] C. Yin, Y. Zhu, J. Fei, and X. He, “A deep learning approach for intrusion detection using recurrent neural networks,” *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[3] A. Bohara, U. Thakore, and W. H. Sanders, “Intrusion detection in enterprise systems by combining and clustering diverse monitor data,” in *Proc. Symp. Bootcamp Sci. Security (HotSoS)*, 2016, pp. 7–16.

[4] K. O’Brien, L. D. Tucci, G. Durelli, and M. Blott, “Towards exascale computing with heterogeneous architectures,” in *Proc. Design Autom. Test Europe (DATE)*, Mar. 2017, pp. 398–403.

[5] J. Kim and J. Park, “FPGA-based network intrusion detection for IEC 61850-based industrial network,” *ICT Exp.*, vol. 4, no. 1, pp. 1–5, 2018.

[6] S. Lloyd, “Least squares quantization in PCM,” *IEEE Trans. Inf. Theory*, vol. IT-28, no. 2, pp. 129–137, Mar. 1982.

[7] M. Estlick, M. Leeser, J. Theiler, and J. J. Szymanski, “Algorithmic transformations in the implementation of K-means clustering on reconfigurable hardware,” in *Proc. Int. Symp. Field Program. Gate Arrays*, 2001, pp. 103–110.

[8] N. Sharma and N. Gaud, “K-modes clustering algorithm for categorical data,” *Int. J. Comput. Appl.*, vol. 127, no. 17, pp. 1–6, Oct. 2015.

[9] M. Eslamnezhad and A. Y. Varjani, “Intrusion detection based on minmax K-means clustering,” in *Proc. 7th Int. Symp. Telecommun. (IST)*, Sep. 2014, pp. 804–808.

[10] W. L. Al-Yaseen, Z. A. Othman, and M. Z. A. Nazri, “Multi-level hybrid support vector machine and extreme learning machine based on modified k-means for intrusion detection system,” *Expert Syst. Appl.*, vol. 67, pp. 296–303, Jan. 2017.

[11] K. Pichara and A. Soto, “Active learning and subspace clustering for anomaly detection,” *Intell. Data Anal.*, vol. 15, no. 2, pp. 151–171, 2011.

[12] M. V. Rao, A. Damodaram, and N. C. B. Charyulu, “Algorithm for clustering with intrusion detection using modified and hashed k-means algorithms,” in *Advances in Computer Science, Engineering & Applications*. Heidelberg, Germany: Springer, 2012, pp. 737–744.

[13] K. Peng, V. C. M. Leung, and Q. Huang, “Clustering approach based on mini batch Kmeans for intrusion detection system over big data,” *IEEE Access*, vol. 6, pp. 11897–11906, 2018.

[14] Z. Muda, W. Yassin, M. N. Sulaiman, and N. I. Udzir, “K-means clustering and Naive Bayes classification for intrusion detection,” *J. IT Asia*, vol. 4, no. 1, pp. 13–25, 2016.

[15] M. K. Gambo and A. Yasin, “Hybrid approach for intrusion detection model using combination of k-means clustering algorithm and random forest classification,” *Int. J. Eng. Sci.*, vol. 6, no. 1, pp. 93–97, Jan. 2017.

[16] K. Labib and V. R. Vemuri, “A hardware-based clustering approach for anomaly detection,” *Int. J. Netw. Security*, pp. 1–12, Aug. 2005.

[17] R. Ratnakumar and S. J. Nanda, “A FSM based approach for efficient implementation of K-means algorithm,” in *Proc. 20th Int. Symp. VLSI Design Test (VLSI-DAT)*, May 2016, pp. 1–6.

[18] H. M. Kamali and A. Sasan, “MUCH-SWIFT: A high-throughput multicore HW/SW co-design K-means clustering architecture,” in *Proc. Great Lakes Symp. VLSI (GLSVLSI)*, 2018, pp. 459–462.

[19] Q. Y. Tang and M. A. S. Khalid, “Acceleration of K-means algorithm using altera SDK for OpenCL,” *ACM Trans. Reconfig. Technol. Syst.*, vol. 10, no. 1, pp. 1–19, 2016.

[20] L. Du, Y. Du, and M.-C. F. Chang, “A reconfigurable 64-dimension K-means clustering accelerator with adaptive overflow control,” *IEEE Trans. Circuits Syst. II, Exp. Briefs*, to be published.

[21] L. C. Hong. (2015). *NSL-KDD Dataset*. [Online]. Available: https://github.com/defcom17/NSL_KDD

[22] M. H. Aghdam and P. Kabiri, “Feature selection for intrusion detection system using ant colony optimization,” *Int. J. Netw. Security*, vol. 18, no. 3, pp. 420–432, May 2016.

[23] D. Terpstra, H. Jagode, H. You, and J. Dongarra, “Collecting performance data with PAPI-C,” in *Tools for High Performance Computing*. Heidelberg, Germany: Springer, 2010, pp. 157–173.

Authorized licensed use limited to: PUC MG. Downloaded on March 06,2024 at 13:23:14 UTC from IEEE Xplore. Restrictions apply.