Traffic Analytics Development Kits (TADK): Enable Real-Time AI Inference in Networking Apps

Kun Qiu*, Harry Chang†, Ying Wang*, Xiaohui Yu*, Wenjun Zhu*, Yingqi Liu*, Jianwei Ma*, Weigang Li*
Xiao boa Liu†, Shuo Dai†

*Intel Corporation, †ZTE Corporation

{Kun.qiu,harry.chang,ying.wang,xiaohui.yu,wenjun.zhu,yingqi.liu,jianwei.ma,weigang.li}@intel.com,
{xiaoba2.dai.shuo}@zte.com.cn

Abstract—Sophisticated traffic analytics, such as the encrypted traffic analytics and unknown malware detection, emphasizes the need for advanced methods to analyze the network traffic. Traditional methods of using fixed patterns, signature matching, and rules to detect known patterns in network traffic are being replaced with AI (Artificial Intelligence) driven algorithms. However, the absence of a high-performance AI networking-specific framework makes deploying real-time AI-based processing within networking workloads impossible. In this paper, we describe the design of Traffic Analytics Development Kits (TADK), an industry-standard framework specific for AI-based networking workloads processing. TADK can provide real-time AI-based networking workload processing in networking equipment from the data center out to the edge without the need for specialized hardware (e.g., GPU, NPU, FPGA). We use TADK as a library for real-time AI-based traffic analytics. We design the overall pipeline of SQLi/XSS detection can achieve up to 3Gbps per core, which can fully support real-time classification in most cases. Meanwhile, the accuracy of traffic classification and SQLi/XSS detection is ≥ 95% in most cases [13]–[16].

To address above challenges, we have designed Traffic Analytics Development Kits (TADK), an industry-standard framework specific for AI-based networking workloads processing. TADK can provide real-time AI-based networking workload processing in networking equipment from the data center out to the edge without the need for specialized hardware [12]. Briefly speaking, TADK brings several advantages to AI-based networking processing:

- **High Throughput:** up to 3Gbps per core (rule-based level) for AI-based traffic classification [7], [8]
- **Low Latency:** 5 ~ 10µs per request for malicious traffic detection [9]
- **High Accuracy:** ≥ 95% accuracy
- **Easy Deployment:** deploying without the need for specialized hardware (e.g., GPU, NPU, FPGA)
- **Easy Development:** module-based development like DPDK [10] and VPP [11]

I. INTRODUCTION

Silicon and software technology advancements targeting AI inference have lowered the barrier (compute cost and R&D effort) to unleash the creativity and innovation of the network application developers on the use of AI-advanced techniques within their commercial solutions. Reports and analysis are projecting the use of AI in Enterprise SD-WAN deployments to increase from 5% in 2021 to 40% in 2025 [1].

Industry practices are introducing AI techniques using artificial intelligence (AI) and machine learning (ML) models across network analytics approach. Here are some examples of use cases: (1) **Traffic analytics:** Used to analyze encrypted network traffic, to identify anomalies within networks [2]; (2) **Malware Detection:** Detecting malicious traffic such as SQL injection or Cross-Site Script [3]; (3) **User Behavior analytics:** Detecting relationships, identifying anomalies, and conducting empirical assessments of security [4]–[6].

In order to support real-world workloads, an industry-standard framework for real-time AI traffic analytics has to meet the requirements for performance, accuracy, and scalability. Based on previous research and discussion with our customers and partners, we have identified several mutually challenging as follows:

1) **High Performance:** TADK provides highly-optimized library for real-time AI-based traffic analytics. We design several novel algorithms to increase the performance. From our benchmarking results, traffic classification can achieve up to 6.5Gbps per core, which can fully support real-time classification in most cases. Meanwhile, the overall pipeline of SQLi/XSS detection can achieve up to 4.5 ~ 6.1µs per HTTP request, which is 2x faster than the existing rule-based solution. Also, the accuracy of traffic classification and SQLi/XSS detection is ≥ 95% in most cases [13]–[16].

2) **Easy Deployment:** The application developed with TADK does not rely on any specialized hardware. TADK fully utilizes modern CPU features such as AVX512 to accelerate AI performance.

3) **Easy Development:** TADK offers a module-based development environment. Developers can implement their own AI-based traffic analytics application by combining TADK’s modules like building block bricks [17].

The rest of the paper is organized as follows: we first give the background and related work of AI-based traffic analytics in Section II. In Section III, we will give the overview design of TADK. Then, we will give some detail of our highly optimized feature extraction algorithms in Section IV, and we will evaluate TADK in two scenarios: traffic classification and...
SQLi/XSS detection in Section V. We conclude in Section VI.

II. BACKGROUND AND RELATED WORK

A. Data Collection

A systematic survey has concluded a general pipeline of AI-based traffic analytics [18]. The first step is data collection [4], [5]. The AI-based solution needs historical data as the input source to train the model. However, capturing and labeling enough traffics is hard to conduct, mainly due to accuracy and privacy concerns. It is reported that 60% of research is using public non-encrypted traffic [18] and using DPI tools to label traffic. In order to cover this issue, TADK provides a labeling helper that can help users to label non-encrypted and encrypted traffic with only one click.

B. Feature Extraction

The next step is called feature extraction. The most common trend uses statistical-based features (e.g., inter-arrival time and packet size with the minimum, maximum and average metrics) since they can be used both on non-encrypted and encrypted traffic analytics [19]–[21]. However, some open-source feature extraction libraries [22] whose performance is as not good as TADK’s library. Meanwhile, TADK can extract not only statistical features but also lexical features from encrypted traffic. It is proved that the combination of statistical and lexical features can significantly increase the accuracy. The flow extraction library of TADK has been utilized in AI traffic analytics [13]–[16]. TADK provides a tokenizer that is remarkably faster than existing solutions to extract lexical features.

C. AI Inference

At the last step, an AI model or an ensemble of models are needed for gathering analytics results [23], [24]. Both supervised and unsupervised methods are widely deployed in traffic analytics. Labeled datasets are used to train a supervised model such as SVM, decision tree, and random forest. Unsupervised models such K-Means are utilized in anomalous traffic detection. Meanwhile, most solutions use the unsupervised model to cluster encrypted traffics since labeling encrypted traffics [25]–[27] is difficult. In TADK, we provide an optimized random forest model for AI inference. We have compared a variety of models and found the random forest is well-balanced between accuracy and latency in traffic analytics workload.

III. THE OVERALL DESIGN OF TADK

A. Core Libraries

TADK is composed of a series of core libraries, which are corresponding to feature extraction and AI inference steps we mentioned before. We show each component in Fig. 1. Flow aggregator is used to aggregate traffics from packets (e.g., real-time packets or packet traces from PCAP files) by 5-tuples. Protocol detection is used to identify protocols such as TCP, TLS, QUIC, and so on. Feature extraction, which is the competitiveness of TADK, has been well-designed to support real-time AI-based traffic analytics. We will describe some core algorithms in Section IV. AI engine is a wrapper of a high-performance random forest, which is based on Intel oneDAL [28]. Our AI engine supports both training and inferencing, including automatic feature reduction.

B. Utilities

TADK brings some useful utilities for training. The data cleaner and labeling helper provide an one-click solution for traffic labeling. The user only needs to capture one or several packet traces (e.g., PCAP files) as input of the labeling helper, and the helper will cluster these packet traces into several clusters. Each cluster will have a labeling tip. The only work
for the user is to label each cluster with tips and use labeled traffic to train a model.

C. Reference Solutions

TADK provides some samples to show the reference usage of TADK core libraries. The traffic classification sample can monitor network traffic and identify different applications in encrypted traffic. Either packet traces (PCAP files) or real-time traffic can be the input of the traffic classification sample. The SQL injection (SQLi)/Cross-Site Script (XSS) detection sample can detect whether the payload of HTTP traffic contains malicious code. TADK also provides a VPP plugin for the traffic classification sample and ModSecurity plugin for the SQLi/XSS detection sample. With these plugins, users can directly integrate AI-based solutions into their existing pipeline without any modification. We give the integration points in Fig. 1.

IV. FEATURE EXTRACTION

A. SIMD-based Histogram

Histogram, such as the distribution characteristic of TCP packet header length, payload length, and arriving time intervals, etc., are mostly used as statistical features. Thus, designing an efficient histogram algorithm is a critical issue. We take histogram calculation of TCP packet payload length as an example to illustrate the detailed implementation. A buffer of lengths of packets as a shown example in Fig. 2 used to store the payload length of each packet in a network flow (for simplicity, 16 packets are considered here). The purpose of the histogram is to count the number of each element in the buffer belonging to a specific bin.

1) Existing Solution: Scalar Calculation (SC) is a widely utilized method. It has been implemented in most feature extraction libraries. SC is a loop-based method, which means they use huge amounts of loop and branch (it has to process and count each element one by one) for the histogram. In order to cover the disadvantage, a loop-free design such as a SIMD-based algorithm has been proposed.

2) Advanced Vector Calculation: We propose a SIMD-based algorithm called Advanced Vector Calculation (AVC). As shown in Fig. 2, we separate the input traffic into 4 categories:

1) **Category 1**: All elements are in different bins.
2) **Category 2**: All elements are random distribution.
3) **Category 3**: All elements are in one bin (except the biggest bin).
4) **Category 4**: All elements are in the biggest bin.

Since each category needs a different algorithm to calculate, we also propose a Vector Category Classifier (VCC) to identify the category of input data. In order to prevent VCC to be an overhead of histogram calculation, we use only up to 3 instructions to identify the category, which is also shown in Fig. 2. We give define SIMD intrinsics in TABLE I.

---

Fig. 2. The workflow of Advanced Vector Calculation with Vector Category Classifier
Briefly speaking, we first use a CMPGT to identify whether each element are larger than the biggest bin. If all elements are larger than the biggest bin, it is category 4. Then, we use CONFLICT to compute vec_conflict and msk_uni for checking whether each element is unique. If all elements are unique, it is category 1. At last, we can simply check whether the msk_uni with only one active bit. If there is only one bit in the msk_uni, it is category 3, otherwise, it is category 2.

Although it is easy to calculate the histogram in categories 1, 3, and 4 with up to 3 instructions, designing an algorithm for category 2 is the most challenging work. Thus, we propose a novel algorithm to calculate the histogram in category 2. We also give an example in Fig. 3. Algorithm 1 shows the pseudocode of AVC and VCC. We evaluate our proposed AVC for histogram calculation. AVC can achieve up to 11.73x, 4.38x, 1.33x and 1.47x faster than the existing solution in categories 1, 2, 3, 4 respectively.

**B. DFA-based Tokenization**

**TABLE I**

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPGT((a, b))</td>
<td>Compare (a) with (b) for greater-than</td>
</tr>
<tr>
<td>CONFLICT((a))</td>
<td>Test each element of (a) for equality</td>
</tr>
<tr>
<td>REDUCE_OR((a))</td>
<td>Reduce each element of (a) for equality</td>
</tr>
<tr>
<td>CMPEQ((a, b))</td>
<td>Compare (a) with (b) for equal</td>
</tr>
<tr>
<td>POPCNT((a))</td>
<td>Count the number of logical 1 bits in each element in (a)</td>
</tr>
<tr>
<td>ADD((a, b))</td>
<td>Add (a) with (b)</td>
</tr>
<tr>
<td>GATHER((a, b))</td>
<td>Load (b) to (a) with a specific order</td>
</tr>
<tr>
<td>PERMUTE((a, b))</td>
<td>Store (a) to (b) with a specific order</td>
</tr>
</tbody>
</table>

Most AI-based traffic analytics (e.g., Next Generation Web
Algorithm 1: Advanced Vector Calculation (AVC)

Input: length the buffer of payload length
Output: hist the histogram of payload length
1. Reset hist
2. Load length to vec_len
3. vec_bin ← \( \text{vec}\_\text{len} / 64 \)
4. msk_overflow ← CMPGE(vec_bin, 15)
5. if msk_overflow = 0xffff then
   6. Category 4: all in the biggest bin
   7. return hist
9. Remove elements that is larger than 15 in vec_bin
10. vec_conflict ← CONFLICT(vec_bin)
11. Use CMPEQ to convert vec_conflict to msk_uni
12. if msk_uni = 0xffff then
   13. Category 1: all in different bins
      GATHER(vec_cnt, hist)
      vec_cnt_added ← ADD(vec_cnt, 1)
      SCATTER(hist, vec_cnt_added)
   14. else if msk_uni BitAnd (msk_uni - 1) = 0 then
       Category 3: all in the same bin
       hist[vec_bin[0]] ← hist[vec_bin[0]] + 16
   15. else
       Category 2: random distribution
       msk_uni_rev ← BitNot
       REDUCE_OR(vec_conflict)
       vec_popcnt ← POPCNT(vec_conflict)
       GATHER(vec_cnt, hist)
       vec_cnt_tmp ← ADD(vec_cnt, 1)
       vec_cnt_added ← ADD(vec_cnt_tmp, vec_popcnt)
       SCATTER(hist, vec_cnt_added)

Application Firewalls) needs tokenization to convert lexical features (string-based information) into vectors as the input of the AI-model. For lexical features, most tokenizers (e.g., OpenNMT) are branch-based, which means they use huge amounts of IF-ELSE for tokenizing. A branch-based solution is easy to implement, but it is unfriendly to the CPU’s pipeline, and it may increase the number of cache misses. Thus, TADK uses a DFA-based tokenizer and provides a generator that can convert an easy-to-code profile into a specific DFA. We give an example of SQLi detection with a DFA-based tokenizer in Fig. 4. We also propose a training video [30] to describe how the tokenizer works.

1) Generator: In order to support multiple language/file formats, we propose a generator that can generate DFA from user-defined profiles. We defines a DFA profile language which can be easily maintained by our customers, and easy to extend to add new tokens for emerging threats, and to support more use cases. The generator also includes a DFA compiler to compile the user-defined profile into its corresponding DFA transition table. The DFA transition table is directly used by the Tokenizer.

Algorithm 2: DFA engine

Input: T the transition table
V the input string
A the state table
Output: R the accept state
1. \( S \leftarrow \text{initial state} \)
2. for \( i \in \text{the numbers of character C divide by V} \) do
   3. \( S \leftarrow T[S][C] \)
   4. if \( S \) is accept state then
      5. output \( A[S] \)

2) Tokenizer: The DFA transition table describes the transition behavior under every state and input character. Algorithm 2 shows how the DFA engine works. The engine does simple transitions in the main loop which makes it very fast.

V. Evaluation

A. Environment

We implement TADK using GCC 7.5. Since TADK has been deployed in several scenarios, such as WAF or 5G User Plane Function (UPF), we have different CPU and RAM environment. The 5G UPF uses ZTE 5300G4X, which is based on Intel Xeon Gold 6330N CPU (Icecake) with 512GB DDR4 RAM. Other evaluation is based on Xeon Gold 6148 CPU (Skylake) and Intel Xeon Platinum 8358 CPU (Icecake) with 32G DDR4 RAM. We integrate our reference traffic classification sample into ZTE 5G UPF to test its throughput and accuracy. We use IXIA as a traffic generator to generate traffic to test the maximum throughput with zero packet loss.

B. Data

Since we choose random forest as our AI inference model, we evaluated the accuracy of random forest in both traffic classification and malware detection. In traffic classification, we have collected top applications in China (BAIDU, TMALL, BILIBILI, TENCENT, TOUTIAO, KUAISHOU, QQ, HUOOS, QQNEWS, YOUKU, WECHAT) from the real-world, for both training and inferencing. In malware detection, we use SQLMAP [31] for SQLi and XSS TRIKE [32] for XSS to gather data for both training and inferencing. We also choose some public data for inferencing.

C. Traffic Classification

1) Accuracy: We give the confusion matrix of the model that can classify 9 applications in Fig. 5. All precision and recall are larger than 90%, and the average precision, recall, and f1-score are 0.936, 0.926, 0.918. From the evaluation result we can see that the accuracy for traffic classification is sufficient for most scenarios. We also train a model to classify WeChat image transfer traffic and WeChat video transfer traffic, which are UDP traffic. We prepare 70 image transfer flows and 100 video transfer flows to train, and we give the accuracy detail in TABLE II. The average precision, recall, and f1-score are 0.883, 0.884, 0.883.
shows that the performance will not reduce with the increasing number of applications. The reason why the throughput in 5G UPF cannot achieve our aforementioned throughput and it has jitter is our naive flow table implementation and other integration overhead.

## VI. Conclusion

In this paper, we proposed TADK as a solution to address real-time AI-based networking workloads processing. The evaluation result shows that the application implemented...
with TADK can meet the requirements for real-time performance (4.5µs per request on malware detection, 6.5Gbps and 35.3Gbps per core on traffic classification and feature extraction), accuracy (≥ 95%), and scalability without any specialized hardware. We have deployed our solution in WAF and 5G UPF and we have evaluated it for real-world usage. We are currently working with our partners to improve the reliability and missing features (e.g., GQUIC) required for real-world deployment, and will be examined and used by the public and community eventually.

REFERENCES


