Real-time Analysis of NetFlow Data for Generating Network Traffic Statistics using Apache Spark

Milan Čermák, Tomáš Jirsík, Martin Laštovička
Institute of Computer Science, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
E-mail: {cermak, jirsik, lastovicka}@ics.muni.cz

Abstract—**In this paper, we present a framework for the real-time generation of network traffic statistics on Apache Spark Streaming, a modern distributed stream processing system. Our previous results showed that stream processing systems provide enough throughput to process a large volume of NetFlow data and hence they are suitable for network traffic monitoring. This paper describes the integration of Apache Spark Streaming into a current network monitoring architecture. We prove that it is possible to implement the same basic methods for NetFlow data analysis in the stream processing framework as in the traditional ones. Moreover, our stream processing implementation discovers new information which is not available when using traditional network monitoring approaches.**

I. Introduction

Monitoring high-speed networks puts high demands on storage and computation capabilities. Thus, flow aggregation and export compatible with IPFIX IETF standards (based on NetFlow v9) has become widely used for traffic monitoring and analysis. Flow monitoring, however, only allows batch data processing where data is processed in separate, typically 5 minute intervals. This approach introduces inconvenient delays to data analysis which decreases the efficiency of such factors as network anomaly detection or threat identification. A solution to the inconvenient delays is to process all data in real-time which is possible due to the use of contemporary distributed stream processing systems. Our previous results [1] showed that modern stream processing systems are able to process large amounts of data in real time and provide sufficient throughput to monitor network traffic.

However, analysing data using a stream processing paradigm is different from the traditional approaches and has its own challenges:

- stream processing cannot recalculate old data,
- goals have to be specified in advance,
- all results are available in (near)real-time,
- scaling allows the processing of a higher amount of data and more complex calculations.

In this paper, the suitability of a stream processing systems is demonstrated for network traffic analysis. We have chosen the Apache Spark system for the demonstration as it offers easy management and high versatility in terms of the running environment and proprietary processing methods [2]. Apache Spark was integrated into a contemporary system for network data monitoring and the specifics of the integration will be described in this paper. Furthermore, network data analysis was implemented into the stream paradigm with respect to its specifics and the results are discussed.

978-1-5090-0223-8/16/$31.00 © 2016 IEEE

II. System Architecture

Traditional flow monitoring architecture consists of network probes, which generate flows from network data, and a collector for data storage. Our system architecture for the demonstration is based on the interconnection of two basic components: IPFIXcol [3] and the stream processing environment of Apache Spark. An overall view of the components and their connections is depicted in Figure 1.

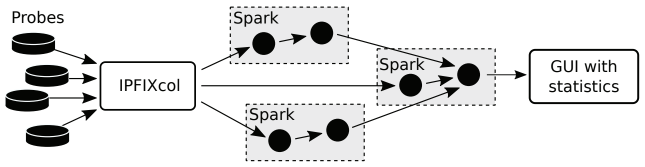

Fig. 1. Network flow data analysis framework architecture.

IPFIXcol is a flexible IPFIX flow data collector designed to be easily extensible by plugins. In our demonstration, we use only part of its wide functionality – data acquisition from multiple network probes and their transformation into a data stream. The data are not stored, only forwarded for further processing to Apache Spark. The collector to the Spark connection is made via the TCP socket, into which the collector feeds its data in JSON format. As the IPFIX protocol [4] supports variable fields, the JSON messages format is not fixed but adapted to it.

Details about Apache Spark stream processing and its deployment are available in our paper [1] about a performance benchmark of distributed stream processing systems. In contrast with the benchmark we have used Spark's component, named "accumulator" to hold data and share them between processing nodes. The other change is in the input data serialisation. In our demonstration, the input is IPFIX data which is serialised into POJO (Plain Old Java Object).

The last part of the framework architecture is a graphical interface for presenting statistics. The computation results from Spark are loaded into a web server and visualised by various types of charts. With the architecture described above, the statistics are calculated in parallel and results are displayed as a statistics graph which refreshes with real-time data.

The demonstration cluster consists of 7 virtual machines, one is dedicated to IPFIXcol, five to Spark and one to the web server. The following configuration is the same for all machines:

- Intel(R) Xeon(R) CPU E5-2680 0 @ 2.70GHz,
- 4 GB 1600M MHz DIMM DRAM EDO,
- 85GB SCSI Disk with 53c1030 PCI-X Fusion-MPT Dual Ultra320 SCSI,
- 10 Gbit/s network connection, 1 Gbit/s virtual NICs.

The software used for the demonstration is the following:

- VMware vSphere 6.0
- Ubuntu 14.04.2 LTS
- Oracle Java 1.8.0
- Scala 2.9.2
- Apache Spark 1.4.1

III. System Demonstration

We demonstrate the utilisation of stream processing in network flow data analysis to generate and report statistics of network traffic. These statistics provide basic information about the volume of flows, packets or bytes transported via an observation point in a network. The aggregation interval for the statistics generation is usually set to five minutes [5] and so is the refresh rate for the statistics. However, the five minute lag is too long in critical situations when network security is at stake and real time data is needed. Moreover, aggregation over such a long period may hide important events that would be observable without the aggregation. Therefore, stream processing comes as a natural choice for the generation of statistics that need to be available in real-time. The reporting provides a periodic overview of the network traffic. A report usually consists of selected information gathered over a certain amount of time. When implementing the reporting in stream processing paradigm, it is important to realise that data are processed on the fly and no historical data analysis is available. Therefore, the report structure (i.e. the computed statistics) needs to be defined in advance and it is not possible to create historical report using stream data processing.

The statistics we chose to implement and demonstrate in stream paradigm are:

- **Basic statistics** – the total number of flow, packets and bytes transported via an observation point in a network.
- **Traffic over HTTP and HTTPS** – a simple overview of the network traffic that shows only connections with web servers.

For generating reports we chose to demonstrate reports with two different frequencies (short term – one minute, and long term – one hour). Each of the reports provides the following information:

- **Basic statistics** – the time series of the number of flow, packets and bytes observed over the report period.
- **TOP 10 statistics** – we present a chart of the TOP 10 most frequently used ports for TCP/UDP network traffic.
- **Host statistics** – for a selected host in the network, we provide the number of communication partners, volume of transmitted data, and most used ports.

The analysis of the statistics generated by the stream processing showed increased volatility in results compared to the results of traditional approaches. The increased volatility is caused by a shorter interval for collecting the statistics. Nevertheless, this approach provides more accurate information about the network. We were able to observe short, but strong bursts of the network traffic that were lost due to the aggregation used in traditional batch approaches.

The results of our demonstration show that it is possible to use stream processing to analyse network data. We were able to successfully implement basic network data analytic methods used in traditional batch network data analysis. We have observed that using stream processing for data analysis reveals new information from the network data as we can increase the granularity of observation. The increased granularity introduces more volatility in the data and may discover information that are lost in traditional approaches.

IV. Conclusion

This paper has presented a pilot exploitation of data stream processing systems for NetFlow data analysis. We were able to interconnect state-of-the-art probes for network data monitoring with the Apache Spark system. The interconnection allows us to process the same data as in traditional approaches. Next, we implemented basic methods for network traffic monitoring. The results show that stream processing approach is suitable for network traffic analysis. It is capable of performing the same analyses as traditional approaches and even opens new ways how to explore data thanks to the increased granularity of the observations. Our future work will further explore the utilisation of stream processing in network monitoring. We plan to transform current batch-based detection methods into the stream paradigm.

Acknowledgement

This research was supported by the Technology Agency of the Czech Republic under No. TA04010062 *Technology for processing and analysis of network data in big data concept*.

References

[1] M. Čermák, D. Tovarňák, M. Laštovička, and P. Čeleda, "A Performance Benchmark for NetFlow Data Analysis on Distributed Stream Processing Systems," in *Network Operations and Management Symposium (NOMS), 2016 IEEE*, 2016, [To appear].

[2] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster Computing with Working Sets," in *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, ser. HotCloud'10. USENIX Association, 2010.

[3] CESNET, z. s. p. o., "IPFIXcol," Web page, 2015, accessed January 10, 2016. [Online]. Available: https://www.liberouter.org/technologies/ipfixcol/

[4] B. Claise, B. Trammell, and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information," RFC 7011 (INTERNET STANDARD), Internet Engineering Task Force, Sep. 2013. [Online]. Available: http://www.ietf.org/rfc/rfc7011.txt

[5] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow Monitoring Explained: From Packet Capture to Data Analysis with NetFlow and IPFIX," *Communications Surveys Tutorials, IEEE*, vol. PP, no. 99, pp. 2037–2064, 2014.