Future Generation Computer Systems 89 (2018) 563–574

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Why wait? Let us start computing while the data is still on the wire☆

Shilpi Bhattacharyya [a,*], Dimitrios Katramatos [b], Shinjae Yoo [b]

[a] Stony Brook University, Stony Brook, NY 11790, USA
[b] Brookhaven National Laboratory, Upton, NY 11973, USA

HIGHLIGHTS

- A framework namely Analysis on Wire (AoW) capable of computing on the wire.
- AoW can help save resources in the data centers and/or cloud.
- AoW can help in making earlier decisions in relevant scenarios as trading.
- Computations include analysis, visualization, pattern recognition, and forecasting.
- We present three examples - forex trading, media publishing, and monitoring sensors.
- AoW can be deployed anywhere in the network between the source and the destination.

ARTICLE INFO

Article history:
Received 1 February 2018
Received in revised form 9 June 2018
Accepted 14 July 2018
Available online 17 July 2018

Keywords:
Software defined networking
Service function chaining
Big Data
Streaming data
Machine learning in networks
Pattern recognition
Analysis on wire
Intelligent networks
Internet of Things (IoT)

ABSTRACT

In this era of Big Data, computing useful and timely information from data is becoming increasingly complicated, particularly due to the ever increasing volumes of data that need to travel over the network to data centers to be stored and processed, all highly expensive operations in the long haul. This is a strong motivation to explore how to perform computing and analysis of data "on the wire", i.e., while the data is still in transit. The nature of these computations include analysis, visualization, pattern recognition, and prediction on the streaming data. In this paper we present the idea of a framework capable of analyzing data in transit based on the principles of a service function chaining architecture. This framework can be deployed at any practical location within the network where computation on data flows is desirable. We further describe an all-virtual implementation of the framework as a worst-case scenario and present an early investigation of its capabilities with three examples — pattern recognition and data visualization on streaming Forex data, targeted advertising from clickstream data, and processing of monitoring data from solar sensors for maintenance decisions. Our results indicate that performing computations on live data flows to provide immediate perspective on the data is possible and attractive, but also that performance heavily depends on the amount and capabilities of the dedicated resources.

1. Introduction

With the Internet revolution, globalization, and digitization of everything and anything possible, there is a tremendous increase in the data volumes moving through the network e.g., an institute like CERN produced 73 PB of data in the year 2016 running their Large Hadron Collider [1]. To make all this colossal amounts of data useful, we perform custom computations on them. Conventionally, when data sent from one end is received at the other end — any kind of computation is performed on it at specific data centers or by utilizing newer computing paradigms as cloud, edge and mist computing to quench the computational needs. In cloud computing, such as in Amazon AWS [2], computation and storage is performed in virtual data centers that are put together dynamically. It is a popular computing paradigm which works well on streaming data and performs all kinds of computations under the flavors of SaaS (Software as a service), PaaS (Platform as a service) and IaaS (Infrastructure as a service). Nevertheless, computation (and storage) still takes place in data centers which can be far away from the data sources and data is subjected to significant latencies, overheads, and delays before any meaningful computation can be performed on it. Edge computing [3–5] facilitates the operation of compute, storage, and networking services between end devices and data centers; computation is performed at the physical "edge" of the network. This paradigm is also called "fog computing". Mist computing [6,7] is the latest paradigm involving computing in the

☆ This project has been funded by Brookhaven National Laboratory, USA (contract number: DE-SC0012704).

* Corresponding author.
E-mail addresses: shbhattachar@cs.stonybrook.edu (S. Bhattacharyya), dkat@bnl.gov (D. Katramatos), sjyoo@bnl.gov (S. Yoo).

https://doi.org/10.1016/j.future.2018.07.024

very end devices found at the edge of the network to assist in the motion of data towards the fog and the cloud. All these paradigms are intimately linked to the Internet of Things (IoT) [8] with the huge numbers of data sources and destinations at the extremes of the network and were conceived to facilitate the operation and reduce the volume of traffic through the network that could eventually "drown" data centers with data tsunamis. Here, however, we suggest that it is feasible to start computations on the data as soon as it arrives at a specific point on the wire by deploying the AoW framework there and this point can be at the edge of the network, close to the data center, or anywhere applicable in between depending on problem scope and data availability. We believe this framework can enable researchers to go beyond the IoT horizon because of its inherent flexibility to be deployed anywhere between the source and the destination.

We follow the Service Function Chaining (SFC) architecture [9] to build the AoW framework. SFC emphasizes that the functionality of some legacy hardware devices can be implemented with the concept of Software Defined Networking (SDN). To the best of our knowledge, throughout the literature the emphasis of SDNs has always been towards a better network management [10,11]. But, we present a high performance computing perspective for SDNs here to enable "on the wire" computation. We present our idea with examples to show how we can compute on streaming data to inspect, analyze, forecast and recognize specific patterns in the data.

We started the AoW framework as an uncomplicated setup [12], where we send a simple string as "hello world" from one end to the other, with and without the SFC architecture, and compute the overhead of sending it through the chain, which becomes almost negligible with increasingly more data being sent. That implementation deployed seven virtual machines, configured with the Vagrant [13] environment, which made the framework quite slow for Big Data.

The current implementation is based on the Docker [14] environment, which is relatively lighter and faster. We design a SDN framework to do computations on the streaming data on the network. We run algorithms on this framework to visualize, forecast and analyze the incoming data into the network to infer useful information. This could help us not only in saving resources in the data centers or alternatively any kind of cloud storage, but additionally in early decision making since data is processed in flight instead of having to wait for its arrival and accumulation at a data center before processing can begin on it. With the help of AoW, we can even prevent or be prepared for impending disasters such as device breakup; we demonstrate this for solar sensors.

We present the functionality and "on the wire" compute potential of the AoW framework with three examples. We run a pattern recognition algorithm on Forex data to plan a better trade investment. We analyze clickstream data from multiple streaming websites to identify user buying patterns. And, finally, we process a streaming data from twenty three solar sensors in the AoW framework to detect which of them is down and possibly schedule maintenance, requisite repairs or replacement for them.

The paper is organized as follows. We discuss the conceptual design of the AoW framework in Section 2.1. The implementation details of the preliminary framework is discussed in Section 2.2, describing the functionality and capabilities of all of its components. The operation of the framework is described in Section 2.3. We start discussing various algorithms which we are executing on the AoW framework under three subsections of Section 3. The observations and the results of the experiments are presented in Section 4. We highlight the related research work in this area and some potential use cases in Sections 5 and 6 respectively. We analyze the trade-off and limitations of the current implementation in Section 7. Finally, we conclude in Section 8, reflecting the future scope of the work as well as our current progress therein.

2. Analysis on Wire (AoW) Framework

2.1. Conceptual design

The AoW framework is designed on the principles of Software Defined Networking [10] where we separate the intelligence (control plane) of the network from its forwarding capability (data plane). The conceptual design of the AoW framework can be seen in Fig. 1 and as depicted, we envisage deployment of AoW-enhanced nodes to multiple locations in a network, spanning multiple domains. These nodes can be coordinated through a layer of middleware to form a distributed computer that can be used to process streaming data originating at multiple geographically distributed sources. The distributed aspect of AoW is beyond the scope of this paper as here we focus on the concept of a single node.

2.2. Preliminary implementation

The preliminary implementation as depicted in Fig. 2, is essentially a Docker-based service function chaining architecture with an OpenDaylight (ODL) [15] controller. We started developing this framework on top of the OpenDaylight Service Function Chaining demo [16]. The controller is a Vagrant virtual machine (VM), which has other nodes as docker containers. All the docker containers are Open vSwitches [17]. We feed the controller with the necessary path information for our framework to perform a desired computation on the streaming data by passing json payloads through a Representational State Transfer (REST) Application Programming Interface (API). The controller, in turn, prepares all other nodes for computation. The controller is also responsible for monitoring the good operation of all other nodes in the framework. The modules in the framework are described as follows.

Traffic Checker - An Open vSwitch, which decides whether the incoming data packet is destined to enter the chain framework. It makes decisions according to the rules created in its flow table as shown in Fig. 4. The rules have been created in the traffic checker by the controller when we feed json payloads through REST API for the required configuration of the framework, once the controller on Vagrant VM starts. Rules are open ended and can be configured based on the demands of the data computing design framework. For our experiments, we have rules based on acceptable ip addresses and packet type. Once a data packet is accepted into the AoW framework at the traffic checker, the data packet needs to move further towards its destination. The traffic checker encapsulates the incoming data packets with Network Service Headers (NSH) [18] and transports them over User Datagram Protocol (UDP) [19] for further movement in the network. We discuss more about NSH towards the end of this section. Traffic Checker is used interchangeably with Checker hereafter.

Forwarder - Once, a packet has been classified to enter the AoW framework, it means it has been encapsulated with NSH, which guides it further in the network. It reaches a Forwarder now, with a specific Computing Unit (CU) attached to it. The Forwarder redirects the packet to the CU, which parses the packets to extract the payload and executes the desired algorithm on the data payload to do any kind of computation and/or analysis.

Computing unit (CU) - This is our data computing unit in the chain. It has two modules: (a) **(Streaming) Data processor module** - which extracts the payload from the incoming data packets, and converts them in a format which the corresponding Algorithm module can accept as an input, (b) **Algorithm module** - which is the custom algorithm module, wherein a plethora of algorithms such as pattern recognition, forecasting, and in general any form of streaming computation can be executed.