Contents lists available at ScienceDirect

Big Data Research

www.elsevier.com/locate/bdr

A Cloud Computing Based Network Monitoring and Threat Detection System for Critical Infrastructures [☆]

Zhijiang Chen, Guobin Xu, Vivek Mahalingam, Linqiang Ge, James Nguyen, Wei Yu [*], Chao Lu

Department of Computer and Information Sciences, Towson University, Towson, MD 21252, United States

ARTICLE INFO

Article history:
Received 2 June 2015
Received in revised form 15 September 2015
Accepted 2 November 2015
Available online xxxx

Keywords:
Network monitoring
Threat detection
Cloud computing

ABSTRACT

Critical infrastructure systems perform functions and missions that are essential for our national economy, health, and security. These functions are vital to commerce, government, and society and are closely interrelated with people's lives. To provide highly secured critical infrastructure systems, a scalable, reliable and robust threat monitoring and detection system should be developed to efficiently mitigate cyber threats. In addition, big data from threat monitoring systems pose serious challenges for cyber operations because an ever growing number of devices in the system and the amount of complex monitoring data collected from critical infrastructure systems require scalable methods to capture, store, manage, and process the big data. To address these challenges, in this paper, we propose a cloud computing based network monitoring and threat detection system to make critical infrastructure systems secure. Our proposed system consists of three main components: monitoring agents, cloud infrastructure, and an operation center. To build our proposed system, we use both Hadoop MapReduce and Spark to speed up data processing by separating and processing data streams concurrently. With a real-world data set, we conducted real-world experiments to evaluate the effectiveness of our developed network monitoring and threat detection system in terms of network monitoring, threat detection, and system performance. Our empirical data indicates that the proposed system can efficiently monitor network activities, find abnormal behaviors, and detect network threats to protect critical infrastructure systems.

1. Introduction

A critical infrastructure system, as a typical cyber–physical system (CPS), is a system that integrates computation, networking, and physical elements together to support different applications [1]. It covers smart transportation, smart electrical power grids, smart medical systems, smart manufacturing systems, etc. In a critical infrastructure system, a huge amount of data will be collected from physical and cyber components and transmitted to the computing core through communication networks. The collected real time data leads to efficient and secured operations of a critical infrastructure system [2,3]. For example, in the smart grid, renewable energy sources, distributed energy storage, and generation need to be efficiently integrated and managed through complex and computationally intense models, real-time analysis, and visualization. Then, massive amount of data will be generated from the power grid and transmitted to the energy management system (EMS) in order to enable efficient system operations [4].

Similarly, in a safe and reliable transportation system, various sensors will be installed on vehicles and deployed on roadsides to collect information and transmit collected data to the operation center. With a large number of vehicles dynamically running in a transportation system, huge volumes of streaming data (big data) are generated by monitoring variations in traffic characteristics (e.g., traffic densities, speeds, vehicles, etc.), over time for timely processing and analysis [5]. For example, real-world SHRP2 dataset is over a petabyte in size [6]. Thus, the mounting volume of stored and processed data, along with the continuously increasing requirements of storage and processing capacity pose significant challenges, which hinders the effectiveness of critical infrastructure systems.

To support highly secured critical infrastructure systems, a generic threat monitoring and detection system will be developed to efficiently mitigate cyber threats. Effectively monitoring data from both physical and cyber components will facilitate the detection of cyber threats and help security administrators respond to cyber-

[☆] This article belongs to SI: Big Data Networking.
[*] Corresponding author.
E-mail address: wyu@towson.edu (W. Yu).

http://dx.doi.org/10.1016/j.bdr.2015.11.002
2214-5796/© 2015 Elsevier Inc. All rights reserved.

threats in a timely manner. Nonetheless, developing a scalable, reliable and robust defense system for protecting critical infrastructure systems is a challenging issue. First, it is challenging to quantify the impact of threats as they may come from various sources. Second, it is difficult to detect threats because the detection system has to inspect various data sources, which are always in large-scale with different formats and semantics. Monitoring different applications and detecting threats can be characterized as high volume data streams and real-time processing requirements. In addition, resources in critical infrastructure systems (e.g., bandwidth, storage, etc.) are also limited. Thus, how to efficiently store and process such big data to ensure security and efficient operations of critical infrastructure systems become a challenging and urgent issue.

To address the aforementioned challenges, in this paper, we developed a cloud-computing based network monitoring and threat detection system for critical infrastructures, which can efficiently enhance the security of critical infrastructures. The proposed system consists of monitoring agents, cloud infrastructure, and an operation center. Monitoring agents can be deployed on devices in a critical infrastructure system to collect the threat monitoring information and transmit the information to the cloud infrastructure. A cloud infrastructure is a distributed system that is deployed with a number of servers, providing both storage and computation resources to efficiently process the large scale of collected data. In the cloud infrastructure, we used both Hadoop MapReduce [44] and Spark [45] to speed up data processing by separating and analyzing the data streams concurrently. The operation center plays the intelligence role that dynamically updates system operation policies and configuration, and monitors the system security.

To demonstrate the effectiveness of our developed network threat monitoring and detection system, we conducted experiments from three aspects: network monitoring, threat detection, and system performance. In network monitoring, we designed several scenarios to find the outgoing traffic volume from each server, the traffic volumes based on source and destination IP address, the incoming traffic volume to each server, and the port access count on each server in a given time duration. We implemented k-mean clustering algorithm in our proposed system to fast classify data into different groups based on their similarities. With dynamic thresholds, we conducted threat detection on emulated distributed denial-of-service (DDoS) network traffic and measured both detection rate and false positive rate. In addition, we compared the efficiency of Hadoop and Spark for the data processing in network monitoring and threat detection. Through our extensive evaluations, our result shows that our proposed system can efficiently help the system administrator to monitor network activities and identify abnormal behaviors. Moreover, our proposed system can accurately and dynamically detect network threats. Our experimental data shows that, with the implementation by Spark, the system performance is almost 30 times better than that of the implementation by Hadoop.

The remainder of the paper is organized as follows: In Section 2, we conduct a literature review of big data in critical infrastructure and cloud computing to assist network monitoring and threat detection. In Section 3, we introduce our system architecture, components, and features. In Section 4, we propose to present the system implementation in detail. In Section 5, we demonstrate the experimental results to validate the effectiveness of the developed threat monitoring system. In Section 6, we show the extension of our proposed system. Finally, we conclude the paper in Section 7.

2. Related work

In this section, we conduct the literature review on big data issues in critical infrastructure systems, cloud computing to assist network monitoring and threat detection.

2.1. Big data issue in critical infrastructure system

Governments, research communities, and enterprises can all make use of the overwhelming amounts of digital data, which is available, evidently creating new opportunities and nurturing powerful business intelligence for decision support [7,8]. Big data can be used by the organizations in creating real-time solutions to the challenges put forth by healthcare, agriculture, transportation, and more. The relentless growth of data will not only challenge information technology engineers, but also researchers in various fields. In order to process massive amounts of data that have been collected, there have been a number of studies on critical infrastructure systems in the past [7,8,20–26,36–40]. For example, Nakazato et al. [9] designed a prototypical system related to big data, in which the driver can use his smartphone to view traffic conditions on an expressway in real time. Chen et al. [10] developed a system called DiabeticLink, which offers electronic health record search, diabetic health indicator tracking, Q&A forums, diabetic medication side effect reporting, and diet recommendations for diabetic patients. The researchers utilized the modern data, text, and web mining algorithms that are relevant to healthcare decision support. In addition, the rapid growth of smart phones has apparently increased the usage of low cost sensors that detect environment and user interaction. For example, Billen et al. [11] presented a framework that stores, fuses and processes smartphone sensor data. Smart phone sensor data can be used in various areas, notably to detect the drunkenness of the driver [12], and pothole detection [13]. Detecting potholes (road surface defects) on real-time can avoid accidents and higher costs.

2.2. Cloud computing to assist network monitoring

In critical infrastructure systems, network traffic has always been the primary resource for network security enthusiasts [15–19, 26–28,35,38,42,43]. There have been several MapReduce based algorithms implemented to monitor network traffic [16,17]. Albeit there are several programming models for parallel processing, MapReduce is a generic mechanism to perform challenging computing tasks [14]. As indicated in [14,17], the key features of MapReduce include cost effectiveness, extreme scalability, high throughput and high performance. Vieira et al. [15] evaluated the efficiency of MapReduce in packet level analysis and DPI (Deep Packet Inspection) and verified that packet level analysis and DPI are Map-intensive. Their study implied that block, input and cluster sizes play a vital role in defining the job completion time and efficiency of MapReduce [15]. Lee and Lee [16] analyzed multi-terabytes of network traffic in a scalable manner. They have devised TCP, IP and HTTP traffic analysis using MapReduce algorithms to improve the scalability of analysis. In addition, a web-based interface to execute Hive queries on NetFlow data was developed.

2.3. Cloud computing to assist threat detection

In order to quickly and efficiently detect threats, integrating detection algorithms with cloud computing has been one of the most active research areas [17–19,28–31,33–35,41–43]. For example, Aljarah and Ludwig [17] proposed an intrusion detection system that uses MapReduce to analyze network traffic. They proposed an algorithm IDS-MRCPSO, which is based on the particle swarm optimization method and clustering based on MapReduce

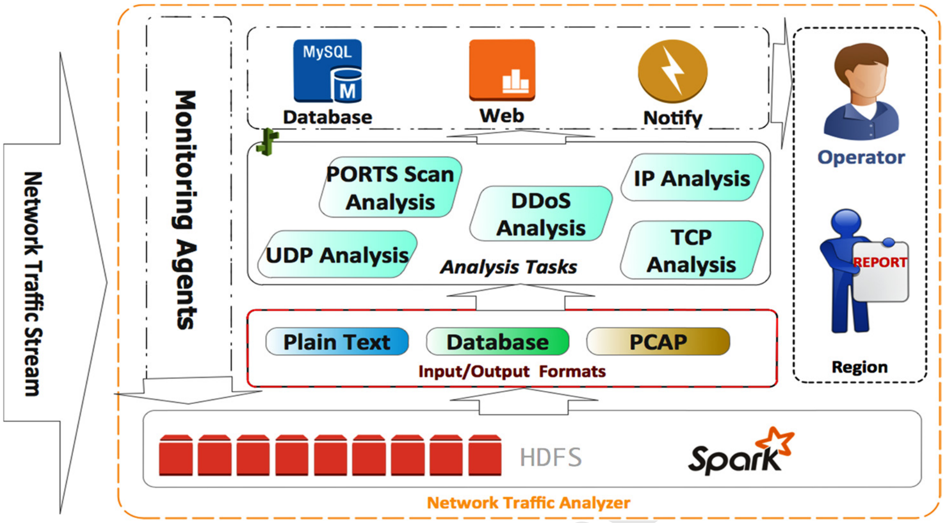

Fig. 1. System architecture.

method. Their research results showed that their developed system is efficient in terms of scalability and detection accuracy. Lee and Lee [18] implemented a DDoS detection method based on Hadoop that leverages HTTP GET flooding detection mechanism by checking both number of accesses and patterns. The findings imply that performing detecting through multiple nodes in parallel could improve the performance. Narang et al. [19] proposed a Hadoop-based framework for the detection of P2P botnets by using the host-aggregation based mechanism to aggregate behavioral metrics for P2P hosts in the network.

Different from existing research efforts, our proposed cloud computing based threat monitoring and detection system can enable secured and efficient operations of critical infrastructure systems, which is novel. The developed cloud based defense system is generic and can integrate various security mechanisms to effectively conduct security situation awareness of critical infrastructure systems. In addition, the large storage and high computational resources in the cloud can handle big data processing and computation, which could further improve the efficiency of critical infrastructure systems.

3. System architecture

In this section, we first give an overview of our cloud based threat monitoring and detection system. We then present the detailed components and features of the system, respectively.

3.1. Overview

The cloud based threat monitoring and detection system can effectively collect, store, and process large traffic data for critical infrastructure systems. As shown in Fig. 1, it consists of the monitoring agent, the cloud infrastructure, and an operation center. The monitoring agent is the threat monitoring software deployed on nodes in the critical infrastructure system to collect threat related information and transmit information such as raw data to the cloud. With a large number of servers in a data center, a cloud infrastructure is a distributed system, which provides large data storage space and high computation facilities. The real time monitored stream of data can be fed and stored in the storage server waiting for threat analysis and detection. In addition, Hadoop MapReduce and Spark, as the implementation of cloud infrastructure, are leveraged to separate and process the data concurrently, aiming to speed up data processing and to increase the efficiency of threat monitoring and detection. In addition, the detection center can dynamically provision the cyber operation policies on components in the critical infrastructure system to enhance the system security.

3.2. Components

We now introduce the main components of the proposed threat detection system in detail.

3.2.1. Monitoring agents

Monitor agents can be deployed on components in a critical infrastructure system to collect its components' information and transmit the information to the cloud. To secure the cyber operation in the critical infrastructure, a large volume of data will be continuously collected for the purpose of threat analysis. For example, different log files (e.g., system logs, firewall logs, web logs, and router access logs) associated with each component can record its activities, which can be used for conducting behavior analysis finding the malicious behaviors. In addition, network traffic data is important to detect normal and malicious behaviors through analyzing traffic data patterns (e.g., the traffic volume per minute between the source and destination IP addresses, the traffic volume per minute from each server, the scan rate of each port, etc.).

3.2.2. Cloud infrastructure

All the data will be transmitted and stored in the cloud infrastructure. To build the cloud infrastructure, we use two types of framework: Hadoop MapReduce and Spark. (i) *MapReduce:* MapReduce [44] is a framework that allows programs to process massive amounts of unstructured data in parallel across an array of computers or servers. Generally speaking, MapReduce comprises a Map() procedure that performs filtering and sorting operations and a Reduce() procedure that performs a summary operation. Map() procedure concurrently processes data based on key/value pair. Assuming that the key/value pair is set as (k, v), Map() procedure can search and filter data set to list all keys and its corresponding values, which is Map(k, v) ⇒ List(k, v). Then, Reduce() procedure takes the intermediate output from Map() procedure to aggregate them to generate the final result, which can be viewed as Reduce(k, List (v)) ⇒ FinalList(v). Notice that, MapReduce is executed in sequence that Reduce tasks will be always executed after Map tasks. (ii) Spark: Spark [45] is a framework developed by Apache for cluster computing, which can be 10–100 times faster than Hadoop MapReduce, and easy to code if we use Scala programing language [45]. Spark can run on different platforms like

Hadoop, Mesos, and standalone. Spark also has machine learning libraries build-in. In addition to Hadoop MapReduce, we also use Spark as a big data processing platform after carefully evaluating the performance. Our experimental data shows that Spark, in our developed system, is almost 30 times faster than Hadoop MapReduce. We also implemented k-means detection algorithm on the Spark framework using Scala. Notice that other detection algorithms can also be easily integrated to our developed system. After the detection process, the results will be stored in MySql database.

3.2.3. Operation center

The operation center can dynamically enforce monitoring policies on data generated by cloud infrastructure. The detection results can be visualized in a web application to help the system administrator efficiently monitoring the system operations. If an alarm signal is received in the operation center, the system administrator can reconfigure the system to ensure security.

3.3. Features

In our design, we consider several features to ensure secured data transmissions, and effective operation of the proposed threat monitoring system.

3.3.1. Modular

Our developed system is based on classic modular design, which is loosely coupled. Each component of the system can be individually plugged into any system by matching the APIs. Modular design can ensure system scalability and reliability that new modules can be integrated into the existing system to enhance its power, features and security.

3.3.2. Secured communication

To guarantee secured data transmission, all sensitive data are stored in a database that is separated from other modules. The communication between database and other modules are encrypted. In addition, we involve HTTPS to help secured data transaction and check session token for each request to make sure the authentication of access.

4. A prototypical system

In this section, we show the implementation of a prototypical system to demonstrate our system design. We first introduce Hadoop and Spark engines to implement the cloud infrastructure. Then, we introduce the system workflow. Finally, we describe the system database and user interface design, as well as threat detection.

4.1. Hadoop

Generally speaking, Hadoop is an open-source framework, which is mainly used to process massive amounts of data in a distributed fashion [44]. Hadoop distributed environment is built by clustering commodity hardware rather than using a supercomputer. Apparently, the use of commodity hardware makes Hadoop prone to hardware failures. Nonetheless, the underlying architecture of Hadoop is built in such a way that makes it highly fault tolerant. Hadoop is written in Java. It comes under the license of Apache Software Foundation. The two important components of Hadoop are MapReduce and Hadoop Distributed File System (HDFS).

4.1.1. Hadoop Distributed File System (HDFS)

Hadoop Distributed File System also known as HDFS is the distributed file system. HDFS is highly scalable and portable. HDFS is not one hundred percent POSIX-compliant. Nonetheless, HDFS is capable of performing most of the basic POSIX-compliant file system functionalities. HDFS is designed in a way that data access (reads) will be extremely high and writes will be minimal. HDFS uses TCP/IP for communication purposes. The two main components of HDFS are data node and name node. Because, Hadoop is based on master/slave architecture. Only one name node will be presented in a cluster, i.e., master. Each slave will have a data node. Name node serves as the metadata store for the file system. HDFS also features triple redundancy and automatic failover, making Hadoop highly available and fault tolerant.

4.1.2. MapReduce

MapReduce is the programming model that Hadoop uses to process massive amounts of data with a parallel algorithm on a cluster. Map phase is used to extract the desired data from each record. Reduce phase is used to aggregate, summarize, filter, or transform the mapped data. Any number of MapReduce jobs can be executed on the cluster of data. Hadoop moves the MapReduce jobs across the cluster and processes the data efficiently. In order to achieve this, it uses job tracker and task tracker. The job tracker will be presented only on the master node, the sole purpose is to track and schedule the MapReduce jobs. Each slave node will have a task tracker, which reports to the job tracker on a regular basis.

4.2. Spark

Apache Spark is an open-source framework, which uses in-memory primitives to process massive amounts of data [45]. Spark is blessed with advanced directed acyclic graph execution engine, which supports in memory computing and cyclic data flow. Spark does not comply with the MapReduce paradigm. It has the ability to process the same data much faster. Spark is optimal for iterative machine learning algorithms as machine learning requires repetitive iterations and communication of shared state. Spark uses the concept of RDD (Resilient Distributed Datasets), which is an optimal way to collect data that can be processed in parallel. Spark streaming also allows modifying the data in real time. Spark provides the ease of managing tasks such as streaming, machine learning, and batch processing.

In Spark, once mapping is completed, the output of map procedure is stored in the buffer cache. Nonetheless, the operating system has complete control over data. The output data may be moved to the disk if necessary. Shuffle spill files are created by the map task, which is directly proportional to the number of reducers. Unlike Hadoop, Spark does not combine and partition shuffle spill files into one huge spill file. Spark writes the shuffle files directly onto the memory. Nonetheless, there must be enough memory to hold the data in order to achieve significant performance gains. Finally, the reducer procedure is executed on the intermediate shuffle files that reside in the memory.

We now compare Spark with Hadoop MapReduce. First, Spark is significantly faster than Hadoop MapReduce, because it runs in the main memory, whereas Hadoop MapReduce runs on data that resides in disks. Second, when compared to Hadoop MapReduce's API support, the API support for Spark is versatile and user-friendly. Third, Spark can perform well on both real-time systems and batch processing, whereas Hadoop is good for batch processing only. Forth, for data integration and data transformation, Hadoop MapReduce can achieve optimized, where Spark is the solution for massive iterative computations. Fifth, to efficient schedule jobs, Spark has an inbuilt job scheduler, whereas Hadoop MapReduce had to use an external job scheduler such as Oozie. Moreover, unlike Hadoop MapReduce, Spark is not bound to a single file system, which can access multiple data sources such as HDFS, Cassandra, Hbase, Amazon S3 and OpenStack Swift. Further, for data recovery,

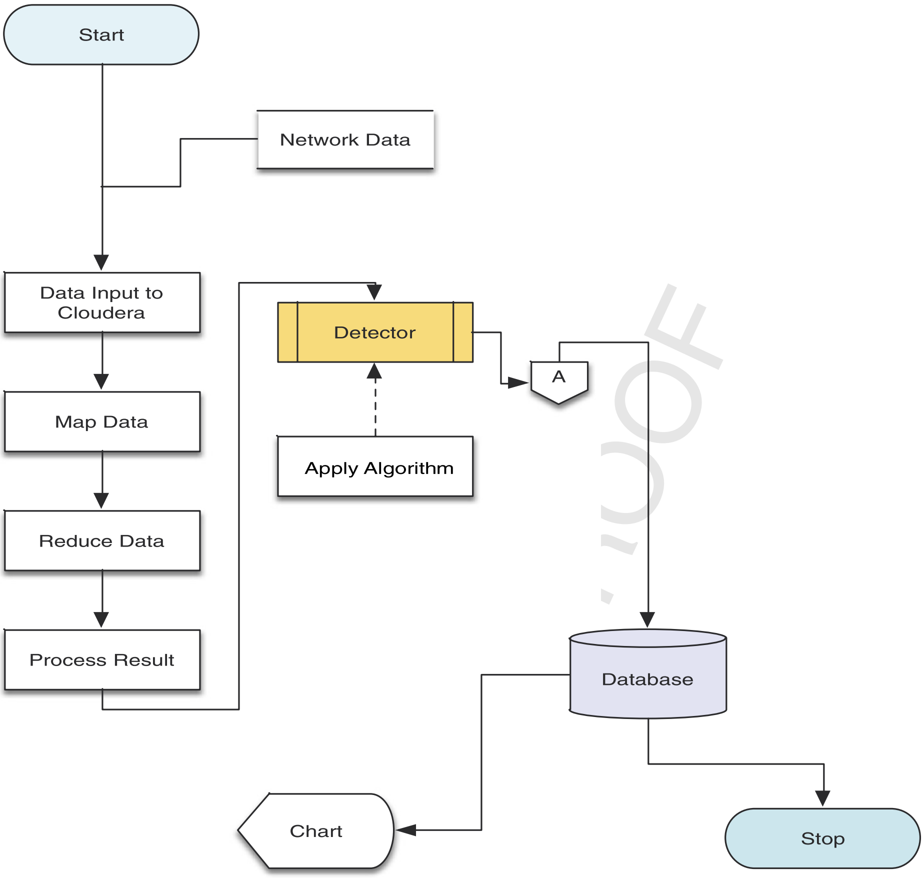

Fig. 2. System workflow.

Hadoop MapReduce uses check-pointing methods, whereas Spark's RDD handles recovery of failed nodes. Hadoop MapReduce is much secure compared to Spark. In addition, both Hadoop MapReduce and Spark comply with speculative execution to provide fault tolerance. Nonetheless, Hadoop MapReduce has the capability to resume a crashed process.

4.3. System workflow

Fig. 2 illustrates the workflow of our system. To conduct threat monitoring and detection for critical infrastructure systems, we collect real time data within the system (e.g., network traffic, system logs) as an input for MapReduce framework, which integrates with the predefined java programming code to execute *Map*() and *Reduce*() job accordingly. After data processing by MapReduce framework, the output result will be stored in a MySql based database server. Based on the schedule, the detector, which is a java application running on a separated Linux server, can start to conduct malicious behaviors detection on the data in MySql database. Finally, the detection result will be sent back to database preparing for visualization purpose. If any malicious activity is detected, an alarm will be automatically generated and inform the system administrator immediately by email and the monitoring system will pop up a warning window to ask for addressing the designated threats. The detailed workflow of the system is described as follows:

Step 1: data collection: Data can be collected from each physical component or throughout a network. For example, a network sniffer like Wireshark software can be used to capture network traffic data. In addition, the collected data need to be normalized prior to entering the MapReduce framework. For example, network traffic data captured by Wireshark is binary data. We should convert the package data to plain text file based on the package characters (e.g., source IP, destination IP, port number, protocol type, package length, package size, and package timestamp).

Step 2: data mapping: The system administrator can define the key/value pair to filter data based on the requirements. For example, in network traffic analysis, if a system administrator intends to find traffic volume of each server, the server address shall be set as key and its corresponding traffic volume shall be set as value. After parallel processing, intermediate files will be generated to store in local disk, which will be used for the input data of *Reduce*() process.

Step 3: data reducing: With shuffle and sorting processes, the intermediate file will be further aggregated and fed to *Reduce*(). Based on the defined key/value pair in *Reduce*() by the system administrator, individual result in each intermediate file can be combined to generate the final result. For example, the traffic volume of the same server address in different intermediate files will be summarized to illustrate the total traffic volume.

Step 4: data storage: After data is processed, the final result will be stored in distributed databases, which can be easily retrieved and used for detection and virtualization purposes. All the data stored in distributed databases will be made into three copies to store in three different physical servers in order to make the system fault-tolerant.

Step 5: notification: The detection results can be delivered to the administrator through email or web display. Once abnormal activities are determined, actions can be taken immediately, such as blocking ports, isolating or cutting off malicious components from network to protect from any possible loss, etc.

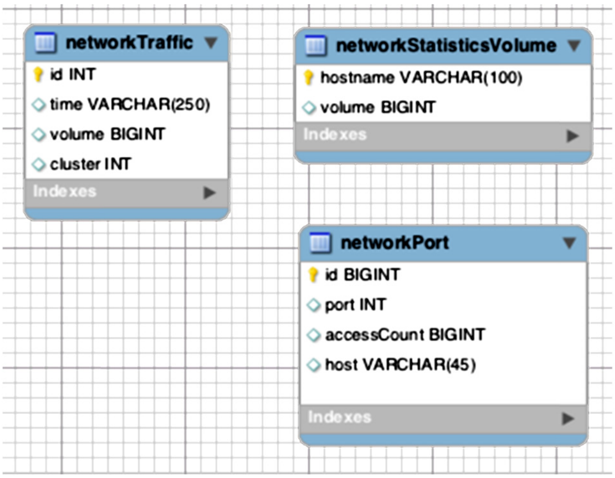

Fig. 3. Database Entity Relationship Diagram (ERD).

4.4. Database design

Our threat monitoring system is developed using the JavaEE/Scala and PHP languages, running on the Apache web server. Our system can integrate various databases (e.g., Oracle, MSSql, MySql, and others). To be cost-effective, we choose MySql as the system database. Fig. 3 illustrates the database design for our developed system, which includes core database tables of networkTraffic, networkStatisticsVolume, netoworkPort to support the virtualization. The networkTraffic table stores the analyzed result for network traffic, which is categorized by cluster. The networkStatisticVolume table stores all the traffic to/from each host server. The networkPort table stores the port access count for each server. The database design is generic and can be extended to integrate more tables when additional modules are integrated.

4.5. User interface design

To help the operation center to conduct threat monitoring and detection, we developed a web based virtualization module with a dashboard. As shown in Fig. 4, the administration dashboard, which is designed by PHP with AJAX (Asynchronous JavaScript and XML), will be automatically refreshed every 5 seconds to retrieve latest data in database. In the dashboard, different types of diagrams help the administrator monitor the network activities (e.g., network traffic, port access, system performance, and others). Each diagram on dashboard is intractable that the detailed information can be displayed with clicking. When threats detected, the system warnings will pope up in the dashboard to notify the administrator to take actions.

4.6. Threat detection using k-mean clustering algorithm and naïve Bayes

To efficiently detect abnormal behaviors, in this paper, we leverage the k-mean clustering algorithm [46] in our system to rapidly classify data into different groups based on their similarities. Generally speaking, clustering is one of the popular and important data mining methods to form a group for the data that have similar feature patterns. We also implemented Naïve Bayes algorithm to compare with K-mean Clustering implementation.

The basic idea of k-mean clustering algorithm is to separate a set of data into k clusters. The process of partition includes two steps: (i) *Computing distance between data point and cluster center:* Initially, with predefined number of clusters k, the k number of cluster center can be randomly generated. Then, the distance between data point and cluster center can be computed and each data point can be assigned to its nearest cluster. (ii) *Updating cluster center*: Based on the newly created clusters, the cluster center will be re-computed and updated. With the help of updated cluster center, by iterating between these two parts repeatedly, the clusters can be updated. The objective of k-mean clustering is to find the minimal sum of square errors to achieve high accuracy clustering. Algorithm 1 shows the detail procedure.

Notice that our proposed monitoring and threat detection system is generic and applicable for different detection algorithms. For example, we can implement other machine learning algorithms

Algorithm 1 K-mean clustering algorithm.

```
Input: $X = \{x_1, x_2, \ldots, x_n\}$ – a set of data
       $k$ – the number of clusters
       $m$ – the number of iterations
Output: $C = \{C_1, C_2, \ldots, C_k\}$, $X \subset C$ – set of clusters
        $E = \{E_1, E_2, \ldots, E_k\}$ – set of clusters' center
1. For each $C_j$
2.    Random generate cluster center $E_j^1$
3. End
4. $D_i = 0$ – the minimal distance of $x_i$ to a cluster center
5. For each $x_i$
6.    Compute distance $d_{ij}$ between data point $x_i$ and current cluster center (e.g.,
      $E_j^1$, $E_j^2$)
7.    If $D_i = 0$
          $D_i = d_{ij}$
8.    Else If $D_i > d_{ij}$
          $D_i = d_{ij}$
9.    Assign $x_i$ to cluster $C_j$
10. End
11. For each $C_j$
12.    Re-compute cluster center $E_j$
13.    Update new $E_j$
14. End
15. Repeat from step 5 to step 14
16. Until cluster center is $E_j^m$
```

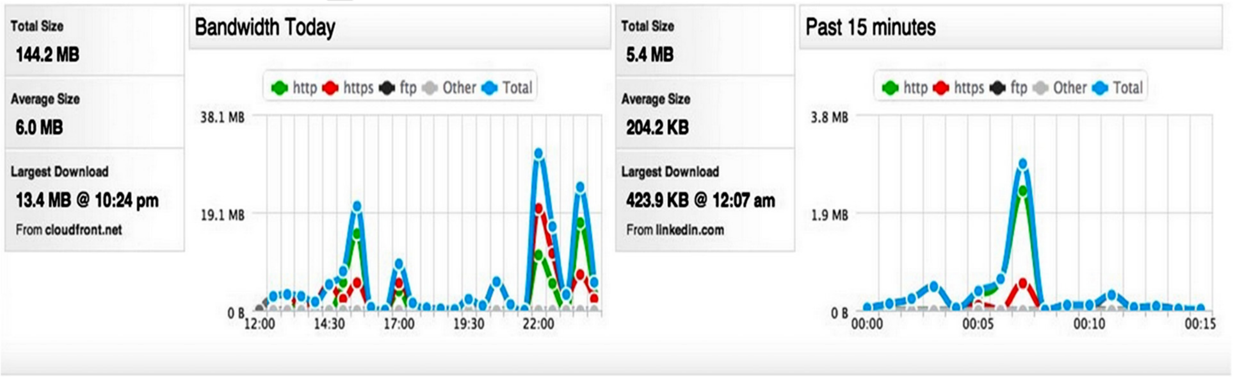

Fig. 4. Example of virtualization.

Please cite this article in press as: Z. Chen et al., A Cloud Computing Based Network Monitoring and Threat Detection System for Critical Infrastructures, Big Data Research (2015), http://dx.doi.org/10.1016/j.bdr.2015.11.002

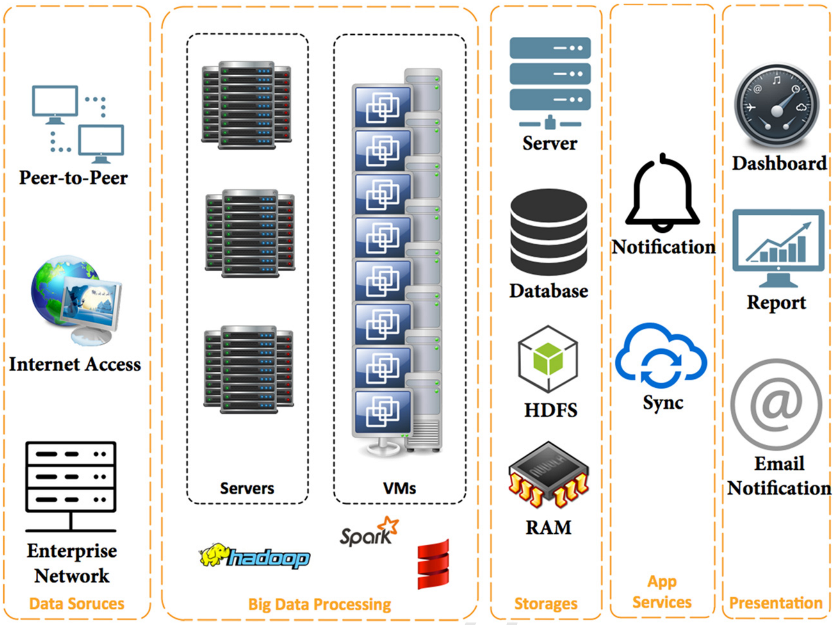

Fig. 5. Testbed setup.

such as Logistic Regression and Naïve Bayes to profile the dynamic characteristics of traffic flows and then to detect anomalies based on learned classifiers. For different critical infrastructure systems, we can apply different detection mechanisms. For example, we can use on-line nonparametric cumulative change detection mechanism to detect threats, which marginally manipulate data over time in the smart grid [51].

Besides K-mean clustering algorithm, Naïve Bayes algorithm is another well-known machine-learning method. The Naïve Bayes classifier can achieve a good classification performance when dataset size is large and the attributes of data are independent from each other. With the support of Naïve Bayes algorithm, we selected five important features, including source IP address, destination IP address, port, packet length, and protocol, to detect network threats. With applying Bayes' theorem and the assumption that features are independent, the classifier can be trained in a supervised learning process.

5. Performance evaluation

We conduct extensive experiments to evaluate the effectiveness of our developed system. In the following, we first present the evaluation methodology and the show the experiment results.

5.1. Evaluation methodology

To evaluate the effectiveness of our proposed system, as shown in Fig. 5, we developed a threat monitoring and detection testbed, which consists of three DELL PowerEdge T420 servers. Each server is configured with 2X Intel Xeon E5-2400 processors with 128 GB ECC RAM and a 1.5 TB hard drive. Each server has two CPUs and each CPU has six cores, which can support up to twelve threads. Because of this, deploying VMs (virtual machines) can effectively utilize computing resources of servers. After installing Esxi 6.0 (enterprise-class, type-1 hypervisor developed by VMware), we created ten VMs on each server. Each VM is installed with Ubuntu 14.04 server operating system and configured with 10 GB RAM, 100 GB disk space, while static IP address and hosts table are configured to achieve reliable communication among all VMs. We deployed a gigabyte switch to ensure a high performance communication among servers. In total, we deployed thirty VMs installed with Cloudera CDH 5.4 for threat analysis. We have one master server to manage thirty computing nodes. We implemented both Hadoop MapReduce (with java language) and Spark (using Scala language) as the cloud infrastructure in our system, which are bundled in Cloudera CDH 5.4.

For the data set, we used real-world network traffic data from Chicago Equinix Data Center [47]. The size of the data set is around 200 GB, which includes more than 50 network traffic related features (e.g., source IP address, destination IP address, source port, destination port, sequence number, acknowledgment number, data offset, reserved, 9 bits flags, windows size, checksum, padding, and others). Notice that, for all the features, only parts of them are useful for detecting potential threats. The selected features include source IP address, destination IP address, source port, destination port, package length, and package timestamp. The data set size is then reduced to 50 GB after partially selecting the features. In addition, in order to simulate abnormal network traffic data, we used the LOIC software [48] to simulate DDoS attacks. In our simulation, we isolated two physical machines to establish a separate lab network. One of them is installed with LOIC simulator [48] to simulate DDoS attacks and the other one is configured as web server, ftp server and SSH server, where all ports are opened and it can be set as a target for attacks. Therefore, we collect DDoS attack data from an isolated network so that other machines will not be affected and the security of our testbed can be ensured. After that, the simulated attack data will be embedded to the normal network traffic data set and will be further used to evaluate the effectiveness of detection algorithms. In our experiments, we used Wireshark software to capture network traffic data.

We evaluated the effectiveness of our developed system in the following three aspects: network monitoring, threat detection, and system performance. For the network monitoring, we designed four scenarios: (i) Finding out the outgoing traffic volume from each server for every 10 seconds, (ii) Finding out the traffic volume based on source and destination IP address for every 10 seconds, (iii) Finding out the incoming traffic volume to each server for every 10 seconds, and (iv) Finding out the port access count on each server for every 10 seconds. Notice that these scenarios are useful and important for detecting potential DDoS attack, port scanning

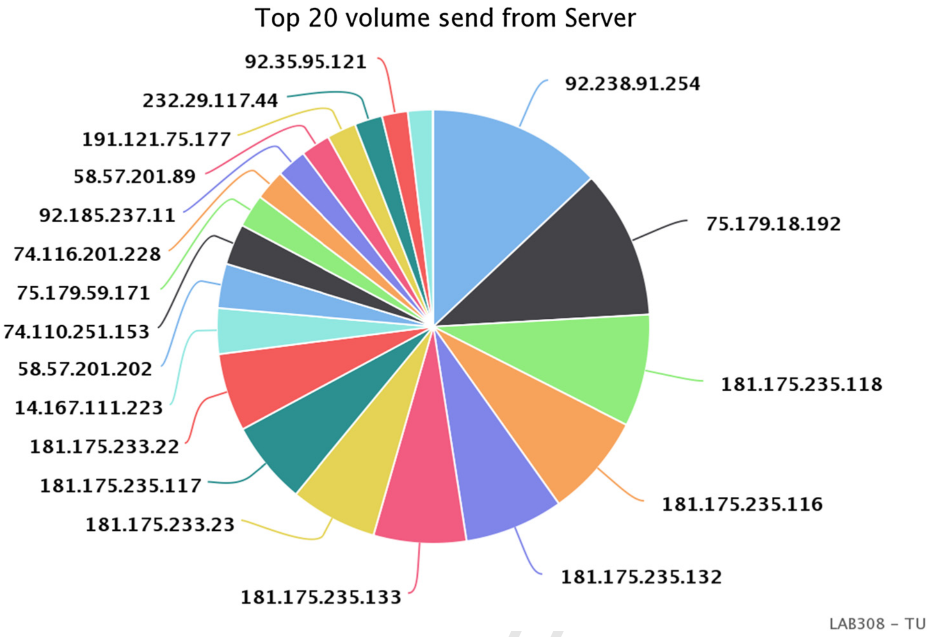

Fig. 6. Top 20 outgoing traffic servers.

attack, and more. Using the detection of DDoS attack as an example, if the attack occurs, usually, the traffic volume to the victim server is high. Similarly, a large number of port access can reflect potential port scanning attack.

For the threat detection, we simulated DDoS and port-scanning activities that last for around 25 minutes. The collected dataset was used for the training process of k-mean clustering. By applying the k-mean clustering algorithm, we established two clusters for the collected data, where one cluster contains the normal traffic data and the other contains the anomalous data. In k-mean clustering, we define the distance from the test data to the cluster center associated with the normal traffic data in a high-dimensional feature space as a threshold for the detection process. For example, if the measured difference is larger than the defined threshold, the test data will be classified as a member in the anomaly cluster. The selection of threshold is a trade-off between the detection accuracy and the false positives. We run simulations for 1000 times without attacks and compute the total number of false alarms for a certain threshold. Then we control the threshold to maintain a low false positive rate. We implemented k-mean clustering algorithm in both Hadoop and Spark. For threat detection, we simulated the DDoS attack through HTTP requests and UDP fragments in the network. To validate the effectiveness of our proposed approach, we compared the efficiency of Hadoop and Spark with respect to network monitoring and threat detection.

To measure the performance, we used the traffic volume and its processing time as metrics for network monitoring. Specifically, the total number of package length in a time interval (e.g., we set it as 10 seconds) and its processing time is the time taken for processing a certain amount of data (e.g., we set data size as 50 GB). To measure the detection accuracy, we use the receiver operating characteristic (ROC) curve, which is plotted with the detection rate versus the false positive rate at various threshold settings. In our case, the detection rate is defined as the probability of correctly classifying the malicious traffic cluster and is the ratio of the number of malicious traffic cluster correctly classified versus the total number of malicious traffic clusters. False positive rate is defined as the probability of falsely classifying non-malicious traffic clusters and is the ratio of the number of non-malicious traffic clusters falsely classified as malicious traffic flows versus the total number of non-malicious traffic clusters.

5.2. *Evaluation results*

In the following, we show the evaluation results from network monitoring, threat detection, and system performance aspects.

5.2.1. *Network monitoring*

Fig. 6 shows the top 20 active servers that have high outgoing traffic. We can see that the larger area in the figure means the higher network traffic volume. For example, as shown in Fig. 6, the server with IP address 92.238.91.254 has highest outgoing traffic. The unusual high outgoing traffic of the server means the server may be compromised as a new attack source. Similarly, Fig. 7 presents the top 20 active servers that have high incoming traffic. As we can see, the server with IP address 73.168.190.18 takes around 25% of incoming traffic in the network, which indicates that the server has possibly been attacked. In addition, the network traffic status between servers is shown in Fig. 8. We observe that the server 92.238.91.254 and the server 73.168.190.18 have highly active communication.

Fig. 9 shows the overall ports access count is larger than 10,000 in a given time from all servers. It shows that the port number 1935 is the most accessed port, which has been accessed 11,069,411 times in an hour. With this figure, the system administrator can easily find the most active port in the network.

5.2.2. *Threat detection*

Fig. 10 shows our simulated DDoS attack network traffic data captured by Wireshark in 20 minutes. Specifically, the traffic volume fluctuations in the figure show the DDoS attack occurred in the network. The significant traffic volume changed in 145–152 and 183–191 denotes the occurrence of strong DDoS attack. Fig. 11 presents the result of k-mean clustering based threat detection. In training process, we carried out extensive experiments and evaluated different thresholds and dataset in our developed testbed. The results on our testbed show when the threshold is set to nine, it can achieve a lowest false positive rate and a highest detection rate. It is worth noting that, for the threat detection used for different networks, the threshold needs to be adjusted to ensure the detection accuracy. The red line in the figure is the detection threshold. The black line represents the distance to anomalous cluster and the blue line is the distance to normal cluster. As we

Please cite this article in press as: Z. Chen et al., A Cloud Computing Based Network Monitoring and Threat Detection System for Critical Infrastructures, Big Data Research (2015), http://dx.doi.org/10.1016/j.bdr.2015.11.002

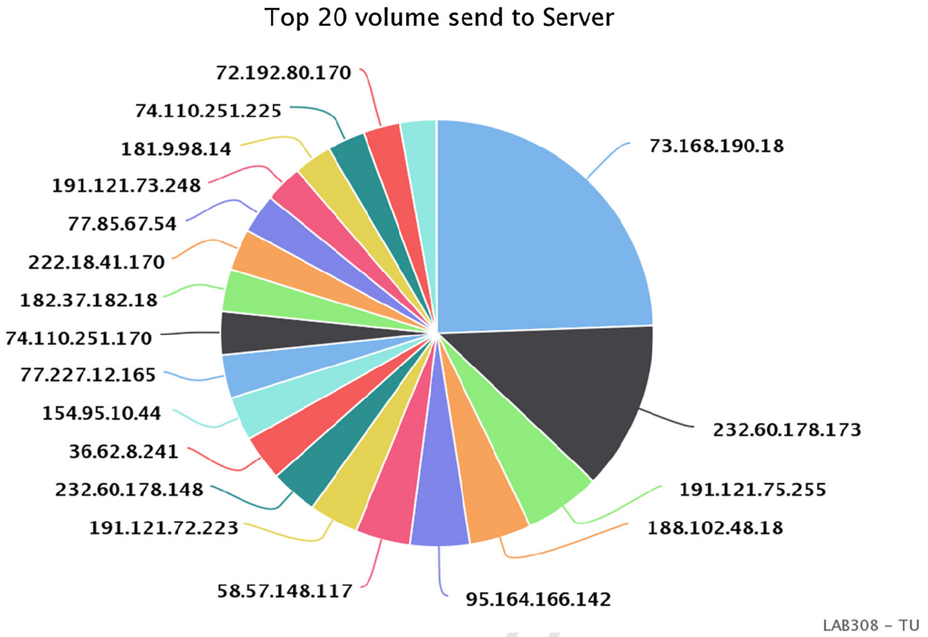

Fig. 7. Top 20 incoming traffic servers.

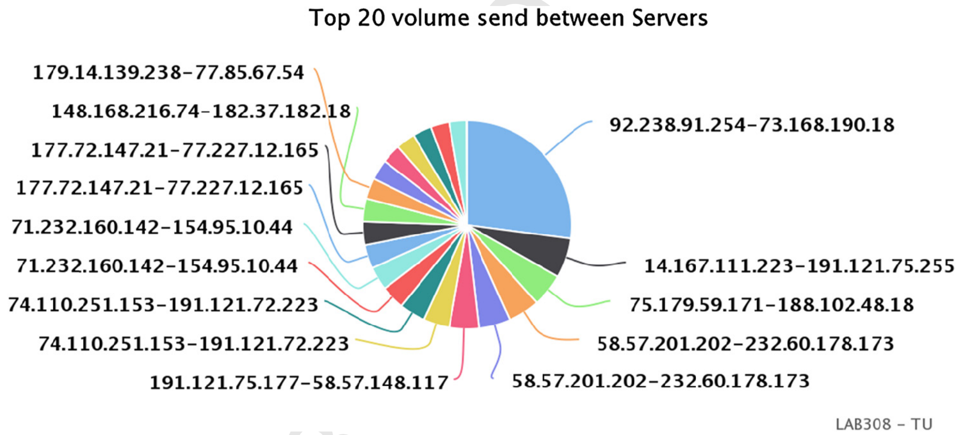

Fig. 8. Network traffic between servers ranking.

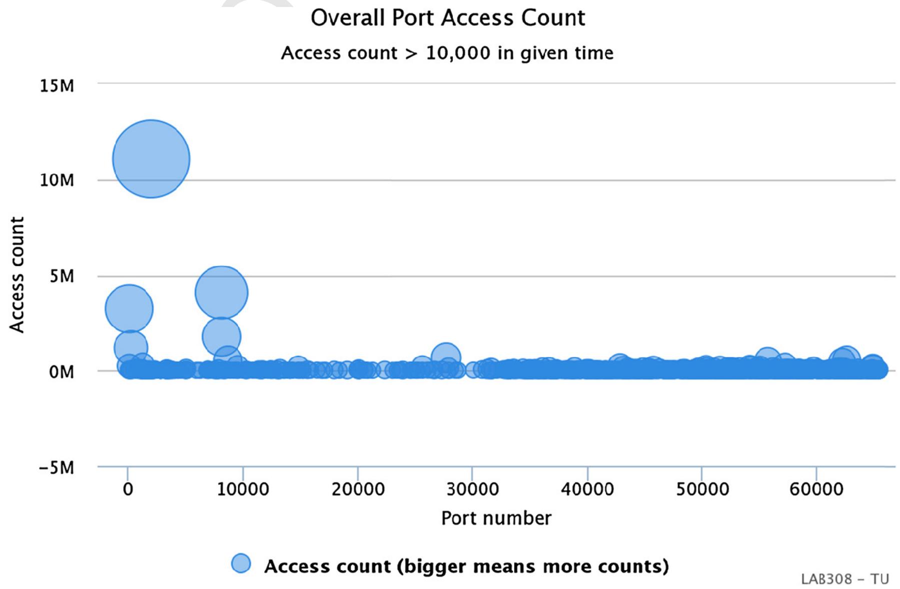

Fig. 9. The number of access in each port.

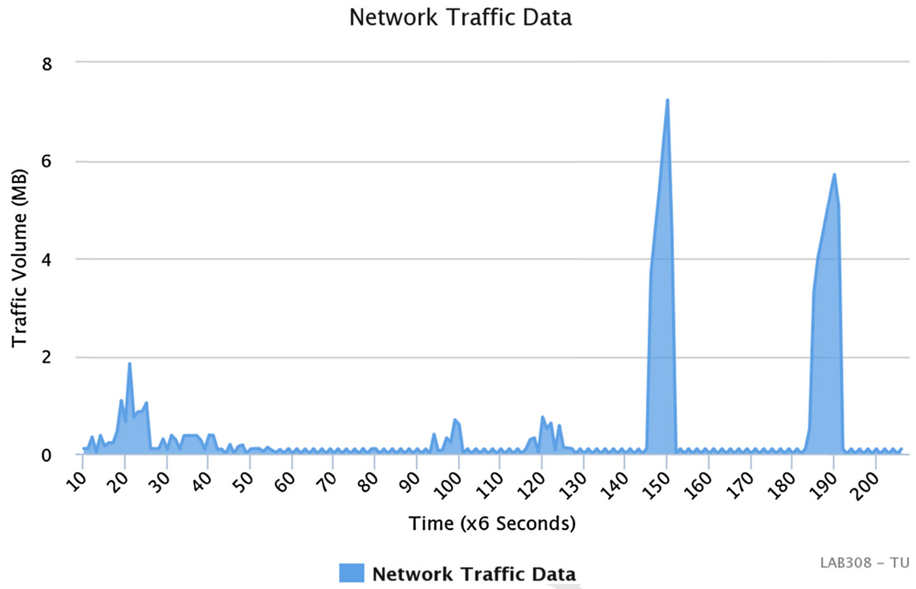

Fig. 10. Network traffic of simulated DDoS attacks.

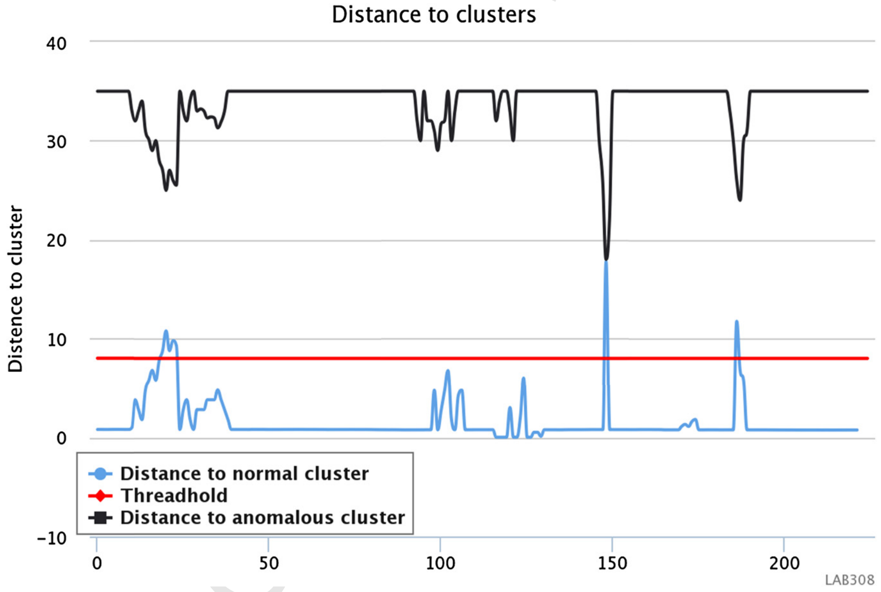

Fig. 11. K-mean clustering based threat detection. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

can see, when a DDoS attack is launched, the traffic volume suddenly spikes. Then, the distance between the node and anomalous cluster will decrease and the distance between the node and normal cluster will increase.

Fig. 12 shows the detection results when Naïve Bayes algorithm is in place. We trained and tested data with the same dataset as K-mean algorithm. In the detection process shown in the figure, we can observe that 338,374 out of 1,636,748 packets have been identified as DDoS attack.

Fig. 13 shows the ROC curve. Our data shows that K-mean clustering scheme achieves a high detection rate (say 90%) and a suppressed false positive rate (say 0.5%). In comparison with K-mean clustering scheme, when Naïve Bayes scheme achieves 90% of detection rate, its false positive rate is around 1.8%, which is higher than that of K-mean clustering scheme. This is because in Naïve Bayes scheme, the training data is randomly selected. While in K-mean clustering scheme, we trained the classifier with history dataset, which can lead to a higher detection accuracy.

Fig. 14 shows the number of ports accessed by server 192.203.28.207 in a time interval. The bigger bubble of a port number means more number of port access. As we can see from the figure, there are six ports accessed during a period of 20 minutes. If an adversary is trying to access a system through port scanning (assuming that, at most, only three ports can be accessed according to the trained data within normal system operations), the alarm will be triggered to notify the abnormal activity.

5.2.3. System performance

Fig. 15 illustrates the processing time with different number of computing nodes on Spark implementing k-mean clustering algorithm. When 50 GB data set is processed in a local machine, which can be considered as one node, the processing time is 16.04 s. By increasing the number of nodes to twenty, the processing time

Please cite this article in press as: Z. Chen et al., A Cloud Computing Based Network Monitoring and Threat Detection System for Critical Infrastructures, Big Data Research (2015), http://dx.doi.org/10.1016/j.bdr.2015.11.002

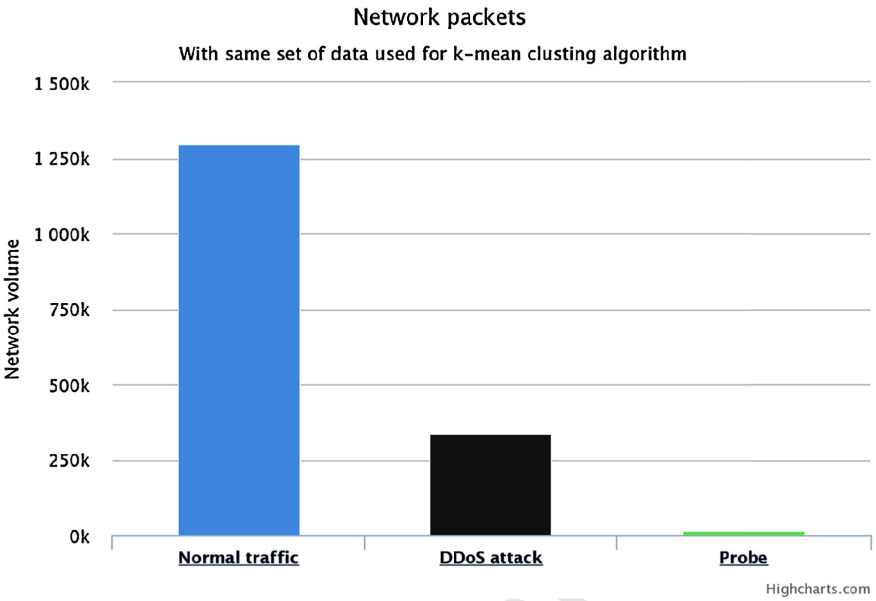

Fig. 12. Naïve Bayes based threat detection.

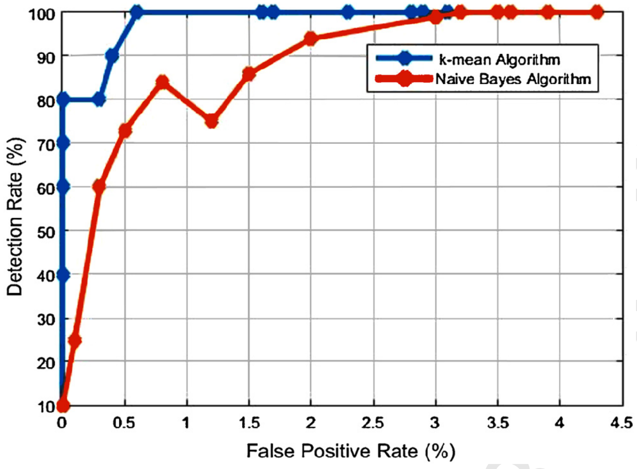

Fig. 13. ROC curve of our detection methods.

will increase. This is because the overhead incurs significantly when the system assigns tasks to node, for example, schedule the resources and perform communication. When more than twenty nodes are used in the system, the processing time is efficiently reduced. With more computing nodes, substantial computation time will be saved even though the overhead still exists.

Fig. 16 shows the performance of Hadoop and Spark to process 50 GB of network traffic data with 30 nodes. As expected, we can see that the Spark's performance is almost 30 times better than Hadoop MapReduce. That is because the Spark runs in the main memory, whereas Hadoop MapReduce runs on data that resides in disks.

Fig. 17 shows the performance comparison between k-mean algorithm and Naïve Bayes algorithm. As shown in the figure, with the same number of computing nodes involved (i.e., 30 computing nodes in our experiments) and different sizes of dataset, k-mean algorithm can achieve fast data processing in comparison with Naïve Bayes algorithm.

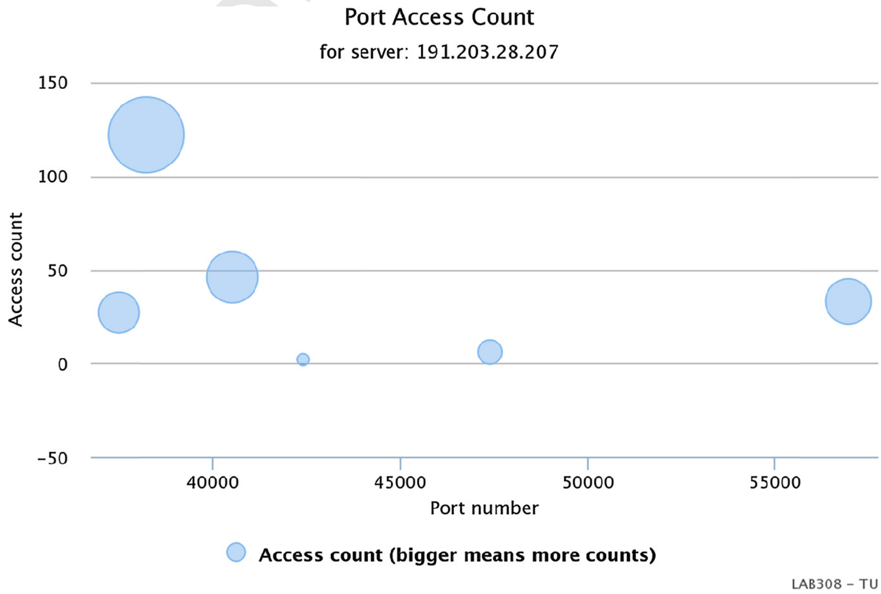

Fig. 14. Detection of port-scanning attack.

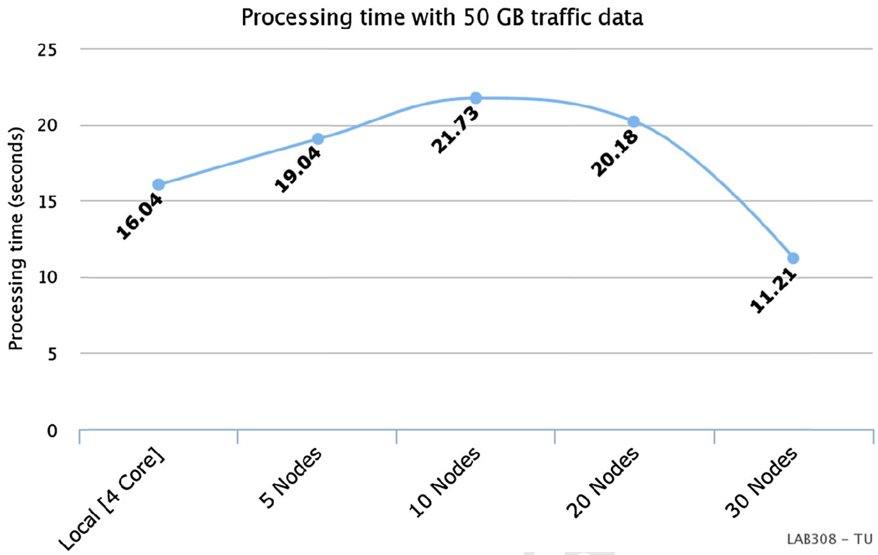

Fig. 15. Processing time of Spark.

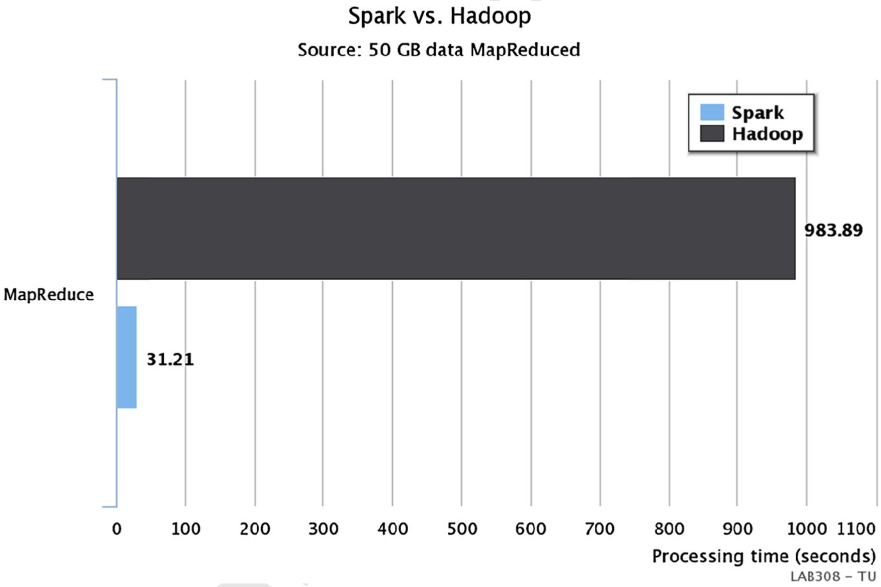

Fig. 16. Processing time comparison between Spark and Hadoop.

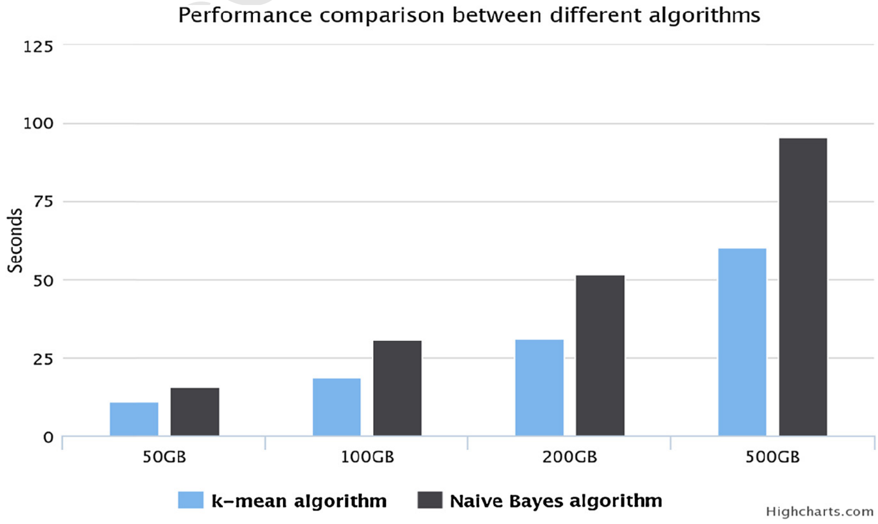

Fig. 17. Performance comparison between k-mean and Naïve Bayes.

6. Discussion

We now discuss the extension of our work from the following aspects: enhance critical infrastructure security with proactive and reactive defense strategies and an extension to handle live streaming data.

6.1. Enhance critical infrastructure security with proactive defense strategies

As ongoing work, we will enhance the cloud based threat monitoring and detection system to secure critical infrastructure systems with proactive defense strategies. We will develop techniques in both data level and system level and integrate them to our system. In data level, we will develop a data self-correction mechanism to detect and recover compromised data. In system level, we will implement and deploy monitoring and detection tools in critical infrastructure systems to effectively discover exploitable vulnerabilities proactively to make the system secure. We will deploy various system management and security tools to establish a trustworthy critical infrastructure. For example, monitoring and detection tools (e.g., Fuzzing [49], SYSSTAT [50], etc.) can be leveraged.

6.2. Enhance critical infrastructure security with reactive defense strategies

In addition to proactive defense strategies, we will systematically integrate more reactive defense strategies in our system (e.g., memory authentication, etc.) to detect compromised components remotely by effectively and efficiently checking whether memory codes are modified. The challenge is performing detection remotely by decoupling detection from network delays. We will also adopt both statistical-based and machine learning-based techniques (e.g., support vector machine, neural networks, etc.), to detect and foresee impeding system anomalies and improve the ability of defense systems to predict behaviors of new attacks. To improve detection efficiency, we will develop a MapReduce based machine-learning scheme to monitor threats on big data sets efficiently. The main idea is to speed up the machine learning process using cloud computing and MapReduce. The first step is to collect the characteristics of threat monitoring data from threat monitors in critical infrastructure system. To accurately and rapidly detect anomalies, techniques (e.g., MapReduce-based machine learning (MML) schemes, etc.) can be used to profile the dynamic characteristics of collected threat monitoring data and then used to detect anomalies. In this way, the computational burden of the learning process is spread across multiple machines and the learned computational results from multiple machines are then integrated into a single learned classifier. Finally, the learned classifier will be used to recognize whether a newly observed data is normal or abnormal. We will evaluate the effectiveness of these techniques in terms of learning accuracy, training set size, and training and detection processes overhead.

6.3. Extension to handle streaming data

Currently, our developed system is intended to analyze static network data. Our system is capable of capturing and storing network traffic data in the database, waiting for batch processing. Nonetheless, in order to monitor and detect imminent threats, we will enhance our developed system to handle streaming data from all sources. Thus, the abnormal behaviors can be detected in near real-time. To achieve this, we will leverage the Spark streaming technique to our system. Spark streaming can use high-level functions like map, reduce, join and window to process live streaming data (e.g., network traffic, active system logs, network device logs, etc.) [45]. The basic idea of Spark streaming is that the streaming data from different sources can be continuously fed to Spark streaming API, and then divided and combined into a small batch for Spark engine processing, which is called DStream (discretized stream). In addition, we will apply a high performance database in our system to handle extreme database input and output operations in order to increase the stability of system.

7. Conclusion

In this paper, we proposed a cloud computing-based network monitoring and threat detection system to secure critical infrastructure systems. The three main components of the proposed system are monitoring agents, cloud infrastructure, and an operation center. With distributed deploying in critical infrastructure systems, monitoring agents play a role in data collection. A cloud infrastructure can provide both large storage space and high computation resources. The operation center can dynamically update system operation policies, configuration, and monitor the system security. We leveraged both Hadoop MapReduce and Spark to speed up data processing by separating and processing data streams concurrently. To evaluate the effectiveness of our developed network threat monitoring system, we evaluated the effectiveness of our developed system with respect to network monitoring, threat detection, and system performance. Through our extensive evaluations, our data shows that the proposed system can efficiently help the system administrator to monitor network activities and find abnormal behaviors. Moreover, the proposed system can accurately and dynamically detect network threats. Our experiments also show that there is a significant performance gain when Spark is used over Hadoop MapReduce.

Uncited references

[32]

References

[1] Wikipedia, Cyber-physical system, available at http://en.wikipedia.org/wiki/Cyber-physical_system, 2013.
[2] Abhishek B. Sharma, Franjo Ivančić, Alexandru Niculescu-Mizil, Haifeng Chen, Guofei Jiang, Modeling and analytics for cyber–physical systems in the age of big data, in: Proceedings of ACM SIGMETRICS Performance Evaluation Review, 2014.
[3] Lu-An Tang, Jiawei Han, Guofei Jiang, Mining sensor data in cyber–physical systems, in: Proceedings of Tsinghua Science and Technology, 2014.
[4] Ching-Han Chen, Ching-Yi Chen, Chih-Hsien Hsia, Guan-Xin Wu, Big data collection gateway for vision-based smart meter reading network, in: Proceedings of Big Data, BigData Congress, 2014.
[5] Tao Qu, Steven T. Parker, Yang Cheng, Bin Ran, David A. Noyce, Large-scale intelligent transportation system traffic detector data archiving, in: Proceedings of Transportation Research Board 93rd Annual Meeting, 2014.
[6] Transportation Research Board, Strategic Highway Research Program: SHRP 2, available at http://www.trb.org/StrategicHighwayResearchProgram2SHRP2/Blank2.aspx.
[7] G. Kim, S. Trimi, J. Chung, Big-data applications in the government sector, Proc. Commun. ACM 57 (3) (2014) 78–85.
[8] J. Bertot, H. Choi, Big data and e-government: issues, policies, and recommendations, in: Proceedings of the 14th Annual International Conference on Digital Government Research, 2013.
[9] Naoto Nakazato, Takuji Narumi, Toshiki Takeuchi, Tomohiro Tanikawa, Kyohei Suwa, Michitaka Hirose, Influencing driver behavior through future expressway traffic predictions, in: Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing, 2014.
[10] Hsinchun Chen, Sherri Compton, Owen Hsiao, DiabeticLink: a health big data system for patient empowerment and personalized healthcare, in: Proceedings of Smart Health, 2013.
[11] Nicolas Billen, Johannes Lauer, Alexander Zipf, A mobile sensor data acquisition and evaluation framework for crowd sourcing data, in: Proceedings of the Second ACM SIGSPATIAL International Workshop on Crowdsourced and Volunteered Geographic Information, 2013.

[12] Jiangpeng Dai, Jin Teng, Xiaole Bai, Zhaohui Shen, Dong Xuan, Mobile phone based drunk driving detection, in: Proceedings of Pervasive Computing Technologies for Healthcare, PervasiveHealth, 2010.

[13] Jakob Eriksson, Lewis Girod, Bret Hull, Ryan Newton, Samuel Madden, Hari Balakrishnan, The pothole patrol: using a mobile sensor network for road surface monitoring, in: Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services, 2008.

[14] Hung-chih Yang, Ali Dasdan, Ruey-Lung Hsiao, D. Stott Parker, Map–reduce–merge: simplified relational data processing on large clusters, in: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, 2007.

[15] Thiago Pereira de Brito Vieira, Stenio Flavio de Lacerda Fernandes, Vinicius Cardoso Garcia, Evaluating mapreduce for profiling application traffic, in: Proceedings of the first Workshop on High Performance and Programmable Networking, 2013.

[16] Yeonhee Lee, Youngseok Lee, Toward scalable internet traffic measurement and analysis with Hadoop, in: Proceedings of ACM SIGCOMM Computer Communication Review (43), 2013, pp. 5–13.

[17] I. Aljarah, A.S. Ludwig, Towards a scalable intrusion detection system based on parallel PSO clustering using mapreduce, in: Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation, 2013.

[18] Yeonhee Lee, Youngseok Lee, Detecting DDoS attacks with Hadoop, in: Proceedings of the ACM CoNEXT Student Workshop, 2011.

[19] Pratik Narang, Abhishek Thakur, Chittaranjan Hota, HaDeS: a Hadoop-based framework for detection of peer-to-peer botnets, in: Proceedings of the 20th International Conference on Management of Data, 2014.

[20] Lin Dai, Xin Gao, Yan Guo, Jingfa Xiao, Zhang Zhang, Bioinformatics clouds for big data manipulation, Biol. Direct 7 (2012).

[21] Randal Bryant, Randy H. Katz, Edward D. Lazowska, Big-data computing: creating revolutionary breakthroughs in commerce, science and society, available at www.cra.org/ccc/files/docs/init/Big_Data.pdf.

[22] Viktor Mayer-Schönberger, Kenneth Cukier, Big Data: A Revolution that Will Transform how We Live, Work, and Think, Houghton Mifflin Harcourt, 2013.

[23] Danah Boyd, Kate Crawford, Critical questions for big data: provocations for a cultural, technological, and scholarly phenomenon, Inf. Commun. Soc. 15 (5) (2012) 662–679.

[24] Chris Yiu, The big data opportunity: making government faster, smarter and more personal, in: Policy Exchange, 2012.

[25] Lev Manovich, Trending: the promises and the challenges of big social data, available at http://manovich.net/index.php/projects/trending-the-promises-and-the-challenges-of-big-social-data.

[26] Hsinchun Chen, Roger H.L. Chiang, Veda C. Storey, Business intelligence and analytics: from big data to big impact, Manag. Inf. Syst. Q. 36 (4) (2012) 1165–1188.

[27] Jiaqi Zhao, Lizhe Wang, Jie Tao, Jinjun Chen, Weiye Sun, Rajiv Ranjan, Joanna Kołodziej, Achim Streit, Dimitrios Georgakopoulos, A security framework in G-Hadoop for big data computing across distributed cloud data centres, J. Comput. Syst. Sci. 80 (5) (2014) 994–1007.

[28] Eric Bloedorn, Alan D. Christiansen, William Hill, Clement Skorupka, Lisa M. Talbot, Jonathan Tivel, Data mining for network intrusion detection: how to get started, MITRE Technical Report, 2001.

[29] Jerome Francois, Shaonan Wang, Walter Bronzi, R. State, Thomas Engel, Botcloud: detecting botnets using mapreduce, in: Proceedings of IEEE International Workshop on Information Forensics and Security, WIFS, 2011.

[30] Anna Koufakou, Jimmy Secretan, John Reeder, Kelvin Cardona, Michael Georgiopoulos, Fast parallel outlier detection for categorical datasets using MapReduce, in: Proceedings of IEEE International Joint Conference on Neural Networks, 2008.

[31] Vibhore Kumar, Henrique Andrade, Buğra Gedik, Kun-Lung Wu, DEDUCE: at the intersection of MapReduce and stream processing, in: Proceedings of the 13th International Conference on Extending Database Technology, 2010.

[32] Zhifeng Xiao, Yang Xiao, Accountable MapReduce in cloud computing, in: Proceedings of IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS, 2011.

[33] Marcelo D. Holtz, Bernardo M. David, Rafael Timóteo de Sousa Jr., Building scalable distributed intrusion detection systems based on the mapreduce framework, in: Proceedings of Revista Telecommun., vol. 1, 2011.

[34] Junho Choi, Chang Choi, Byeongkyu Ko, Dongjin Choi, Pankoo Kim, Detecting web based DDoS attack using MapReduce operations in cloud computing environment, J. Internet Serv. Inf. Secur. 3 (4) (2014) 28–37.

[35] Alvaro A. Cárdenas, Pratyusa K. Manadhata, Sreeranga P. Rajan, Big data analytics for security, in: Proceedings of the IEEE Computer and Reliability Societies, 2013.

[36] Stephanie E. Hampton, Carly A. Strasser, Joshua J. Tewksbury, Wendy K. Gram, Amber E. Budden, Archer L. Batcheller, Clifford S. Duke, John H. Porter, Big data and the future of ecology, in: Proceedings of Frontiers in Ecology and the Environment, 2013.

[37] Stephen Kaisler, Frank Armour, J. Alberto Espinosa, William Money, Big data: issues and challenges moving forward, in: Proceedings of 46th Hawaii International Conference on System Sciences, HICSS, 2013.

[38] Emilio Corchado, Álvaro Herrero, Neural visualization of network traffic data for intrusion detection, in: Proceedings of Applied Soft Computing, 2012.

[39] Yuri Demchenko, Zhiming Zhao, Paola Grosso, Adianto Wibisono, Cees de Laat, Addressing big data challenges for scientific data infrastructure, in: Proceedings of CloudCom, 2012.

[40] Wei Yu, Guobin Xu, Khanh D. Pham, Erik P. Blasch, Genshe Chen, Dan Shen, Paul Moulema, A framework for cyber–physical system security situation awareness, Foundational Methods for Cyber–Physical Systems, 2015, in press.

[41] Linqiang Ge, Hanling Zhang, Guobin Xu, Wei Yu, Chen Chen, Erik P. Blasch, Towards MapReduce Based Machine Learning Techniques for Processing Massive Network Threat Monitoring Data, Networking for Big Data, CRC Press & Francis Group, 2015.

[42] Guobin Xu, Wei Yu, Zhijiang Chen, Hanlin Zhang, Paul Moulema, Xinwen Fu, Chao Lu, A cloud computing based system for network security management, Int. J. Parallel Emerg. Distrib. Syst. 30 (1) (2015) 29–45.

[43] Wei Yu, Guobin Xu, Zhijiang Chen, Paul Moulema, A cloud computing based architecture for cyber security situation awareness, in: Proceedings of the 4th International Workshop on Security and Privacy in Cloud Computing (SPCC), 2013.

[44] Hadoop, What is Apache Hadoop, available at https://hadoop.apache.org/, 2015.

[45] Spark, Spark: lightning-fast cluster computing, available at https://spark.apache.org/, 2015.

[46] Ahmad Amir, Lipika Dey, A k-mean clustering algorithm for mixed numeric and categorical data, in: Proceedings of Data & Knowledge Engineering, 2007.

[47] CAIDA Data, available at http://www.caida.org/data/, 2015.

[48] troyhunt, What is LOIC and can I be arrested for DDoS'ing someone, available at http://www.troyhunt.com/2013/01/what-is-loic-and-can-i-be-arrested-for.html, 2013.

[49] Anna-Maija Juuso, Ari Takanen, Kati Kittilä, Proactive cyber defense: understanding and testing for advanced persistent threats (APTs), in: Proceedings of the European Conference on Informations Warfare, 2013.

[50] SYSSTAT, http://sebastien.godard.pagesperso-orange.fr/, 2015.

[51] Qingyu Yang, Jie Yang, Wei Yu, Dou An, Nan Zhang, Wei Zhao, On false data-injection attacks against power system state estimation: modeling and countermeasures, IEEE Trans. Parallel Distrib. Syst. 25 (3) (March 2014) 717–729.

Please cite this article in press as: Z. Chen et al., A Cloud Computing Based Network Monitoring and Threat Detection System for Critical Infrastructures, Big Data Research (2015), http://dx.doi.org/10.1016/j.bdr.2015.11.002