A cloud computing based system for cyber security management

Guobin Xu, Wei Yu, Zhijiang Chen, Hanlin Zhang, Paul Moulema, Xinwen Fu & Chao Lu

To cite this article: Guobin Xu, Wei Yu, Zhijiang Chen, Hanlin Zhang, Paul Moulema, Xinwen Fu & Chao Lu (2015) A cloud computing based system for cyber security management, International Journal of Parallel, Emergent and Distributed Systems, 30:1, 29-45, DOI: 10.1080/17445760.2014.925110

To link to this article: http://dx.doi.org/10.1080/17445760.2014.925110
A cloud computing based system for cyber security management

Guobin Xu\textsuperscript{a}, Wei Yu\textsuperscript{a,}\textsuperscript{*}, Zhijiang Chen\textsuperscript{a}, Hanlin Zhang\textsuperscript{a}, Paul Moulema\textsuperscript{a}, Xinwen Fu\textsuperscript{b} and Chao Lu\textsuperscript{a}

\textsuperscript{a}Department of Computer and Information Sciences, Towson University, 7800 York Road, Towson, MD 21252, USA; \textsuperscript{b}Department of Computer Science, University of Massachusetts Lowell, 1 University Ave, Lowell, MA 01960, USA

(Received 30 April 2014; final version received 6 May 2014)

The exponential increase of cyber security has led to an ever-increasing accumulation of big network data for cyber security applications. The big data analysis for cyber security management presents challenges in data capturing, storing and processing. To address these challenges, in this paper we develop a cloud computing based system for cyber security management to fasten the analysis process of big network data. Our developed system is built on the \textit{MapReduce} framework and consists of end-user devices, cloud infrastructure and a monitoring centre. To make our proposed system efficient, we introduce two key function modules of our system: data storage module and task scheduling module. We conduct the system implementation using Apache Hadoop, and our implemented system consists of data collection, data normalisation, data computation and data visualisation. Using ranking and aggregation as primitives for performing cyber security management, we conducted extensive experiments to show the effectiveness of our developed system. We also discuss how to extend our proposed system to other applications.

\textbf{Keywords:} big data; cloud computing; cyber security management; \textit{MapReduce}

1. Introduction

The exponential increase of Internet-based applications has reshaped how people interact with each other and enabled the efficient network-based business. Nonetheless, the rapid growth of the digital world also leads to an extremely large accumulation of data from hosts and network devices. For example, according to the \textit{Infosys} labs report, \cite{1} web users generate about 200 exabytes data each year. Intel statistics \cite{2} reveal that for every Internet minute, we account for more than 204 million emails exchanged, around 20 million photos are viewed on \textit{Flickr}, 6 million \textit{Facebook} pages are viewed around the world and more than 1.3 million video clips are watched on \textit{YouTube}.

Analysing large data or big data piling up in organisations has become a hot topic that affects a variety of domains, including cyber security, healthcare, manufacturing, network design and so on. To conduct cyber security operations, the effectively processing big data from hosts and network devices will facilitate the detection of cyber threats and help security administrators respond to cyber attacks in a timely manner. Nonetheless, big data presents serious challenges for cyber security situation awareness, which needs to capture, store, manage and process an ever-growing large and complex security related data from cyberspace. Cyber security applications such as network threat monitoring, detection and analysis are characterised by a very high volume of data streams, as big data, and real-time...
processing requirements. With the continuous, unbounded, rapid and time-varying data streams generated by numerous hosts and network devices, the complexity to store and process big network data for cyber security applications will significantly increase. Hence, there is an urgent need to develop an efficient and scalable solution to process and transform complex and often unstructured large amounts of data into useful information.

In this paper, we propose a cloud computing based system using the MapReduce framework for conducting cyber security management, which can fasten the analysis of large network traffic data. To be specific, we first introduce the MapReduce framework and review its structure and functions, along with Apache Hadoop. It is worth noting that MapReduce is a parallel programming model, which is primarily designed for the batch processing over big data in a distributed computing environment. We then present our system architecture, which consists of the following components: end-user devices, cloud infrastructure and monitoring centre. To make our system efficient, we also introduce the two key function modules in our system: data storage module and task scheduling module.

To evaluate our proposed system, we implemented a prototypical system using Apache Hadoop, an open-source software framework. In our developed system, we use Cloudare to efficiently manage the overall system and present the system set-up and implementation, including data collection, normalisation, computation and visualisation. To validate the effectiveness of our developed system, we use the ranking and the aggregation as the primitive functions for performing cyber security management. Through extensive experiments, our data show that our developed system can efficiently process and analyse large network traffic data. We also discuss how to extend our proposed system to other applications and how to make the proposed cloud infrastructure secure.

The rest of the paper is organised as follows: we introduce the MapReduce framework, including the map function, the reduce function and Apache Hadoop in Section 2. We present a cloud-based system for cyber security management, including system architecture, data storage module, task scheduling module, data analysis module and system implementation in Section 3. We show experimental results to validate the effectiveness of our developed system in Section 4. We discuss how to extend our developed system to other applications in Section 5. We conduct the literature review in Section 6. Finally, we conclude the paper in Section 7.

2. MapReduce framework

Generally, MapReduce is a parallel programming model primarily designed for batch processing over big data in a distributed computing environment.[3] It is worth noting that MapReduce is designed using the concept of divide-and-conquer and follows the master/slave paradigm. MapReduce can take advantage of the locality of data, processing data on or near the storage assets to reduce the overhead of transmitting data. To address the big data issue, a master node will divide the task into a number of small subtasks, which are independently executed in parallel on multiple slave nodes. Slave nodes can be individual threads, processes or individual computers. Intermediate results from slave nodes will be further integrated to obtain the final results.

As shown in Figure 1, the MapReduce framework consists of user, master, map worker and reduce worker. The user provides a set of data $A_{ij}$ to the MapReduce framework for processing. The data-set $A_{ij}$ is then split into $n$ chunks and stored in the distributed file system. The MapReduce is based on key/value tuples and relies on two built-in functions: the map function and the reduce function. The map and reduce functions can be defined by the user with a set of key/value pairs $K_i \rightarrow A_{ij}$. The key/value pairs can be various data...
types (e.g. string and integer). For example, the source IP address and the destination IP address pair can be defined as the key with string data type and the information related to the source IP address while the destination IP address pair can be defined as a value. The detail of map and reduce functions will be introduced next.

2.1 Map function

For the map function, after data are input into the MapReduce framework, the master will manage and maintain data in the distributed file system. Based on the defined map function with key/value pairs by users, the master will divide the processing task into multiple subtasks and distributes them to map workers. After receiving assigned tasks and data locations, the map worker will read the data from the distributed file system and process data. Then, the map worker will scan the input data and perform key matchings to list associated key/value pairs. The map workers can run subtasks concurrently and the master will keep tracking the progress of individual subtasks. Subtasks in the waiting queue will then be assigned to the map workers when they become available. The output of the map worker will be a set of intermediate key/value pairs $K_i \rightarrow A_{ij_1} \cdots A_{ij}$, which will be output into intermediate files and stored locally. The intermediate key/value pair is the list of values related to defined keys. The locations of intermediate key/value pairs will be feedback to the master. In the following, we illustrate an example with the three traffic inputs to the map worker.

1. Oct 16 03:07:38 192.168.1.1 192.168.1.2 80
2. Oct 16 03:07:42 192.168.1.23 192.168.1.40 80
3. Oct 16 03:07:42 192.168.1.5 192.168.1.96 23

We can see that the traffic input 1 and input 2 are from the same port 80. We define the key as the port number and the value as the recorded information associated with the
selected port number. The aggregated results of intermediate key/value pairs are listed as follows:

2. 23: ["Oct 16 03:07:42 192.168.1.5 192.168.1.96"]

2.2 Reduce function

In the reduce function, prior to the computation, the intermediate results need to be shuffled or sorted to group the identical intermediate key/value pairs located in different intermediate files. Based on the key sorting results $K_1, \ldots, K_i$, the master will assign different tasks to the reduce workers, along with the intermediate key/value pair locations. For example, the intermediate key/value pairs with the key ranging from $K_1$ to $K_x$ will be assigned to one reduce worker and the intermediate key/value pairs with the key ranging from $K_x+1$ to $K_i$ will be assigned to another reduce worker. The reduce worker will locally or remotely retrieve the intermediate results and perform the key/value computation. The output of the reduce function will be returned to the distributed file system and reported to the master. After all the map and reduce workers complete the assigned subtasks, the master will return final results to the user.

2.3 Apache Hadoop

Generally, Hadoop [4] is an open-source software framework licensed under the Apache v2 licence. As a MapReduce implementation, Hadoop supports data-intensive distributed applications and can work with a number of computation-independent computers and deal with petabytes of data. To perform the network traffic analysis for cyber security situation awareness, Hadoop consists of the following functions:

- **Network data capture**: It is responsible for capturing network traffic data. Once network data are captured, this function computes flows and exports data to specified collectors. To capture network traffic data, software tool such as nProbe can be used. The traffic capture can be performed by a high-end system with dedicated hardware if needed.
- **Netflow collection**: It is performed by software collectors such as nfCapd, which reads the netflow data collected from the network and stores it into binary files.
- **Traffic information storage**: The software tool such as nfdump can be used to read the netflow data from files and dump them to store as plain-text files.

In Hadoop, there are several important components such as the distributed file system, the database management system and the user interface, which all are important to support the above functions. In the following, we will describe these components.

- **Hadoop Distributed File System (HDFS)**: It is responsible for data management and manipulation. HDFS provides a reliable storage of both input and output data required by MapReduce tasks. In HDFS, data are stored as files that can be split and distributed across multiple nodes. Unlike other distributed file systems, HDFS is explicitly designed for applications with large data-sets. It features a high fault-tolerance through data replication, concurrent access to files, cross-platform portability and low-cost deployment.
Hbase: It is an open-source, non-relational, distributed and column-oriented database management system that runs on top of HDFS. To enable scalable parallel processing of data, Hadoop integrates a tool named Cloudera Impala, an open-source Massively Parallel Processing query engine. Cloudera Impala enables the capability for users to issue low-latency structured query language (SQL) queries to the data stored in HDFS and Apache Hbase without the physically moving or transforming the data.

Pig and Hive: To interact with users, HDFS provides two non-programmatic interfaces Pig and Hive to process queries from users and to present data-sets in a standardised way. Pig and Hive receive queries from users, compile queries and execute them on nodes. HiveQL is a query language provided by the hive interface. Similar to the well-known SQL, HiveQL presents data as tables to perform the basic SQL operations such as select, join and insert.

3. A cloud-based system for cyber security management

In this section, we first show the system architecture, which consists of several key components: end-user devices, cloud infrastructure and a monitoring centre. To make the system efficient, we then introduce the two key function modules in our system: data storage module and task scheduling module. Finally, we introduce the implementation of our proposed system.

3.1 System architecture

3.1.1 End-user devices

End-user devices consist of end hosts (e.g. computers, mobile devices and others) and network devices (e.g. routers, firewalls and others). As the resources of end-user devices are limited, they lack the computation ability and storage capacity in comparison with high-performance computers. The cloud infrastructure will provide a huge storage capacity and computation power to perform cyber security management. Hence, the data from end-user devices are continuously streamed up to the cloud infrastructure for threat monitoring and analysis purposes. For example, various logs (e.g. system logs, security logs, and application logs) on mobile devices and computers can be used to perform both static and dynamic behaviour analyses of malware on end-user devices.[5] With the help of network devices (e.g. routers, firewalls and sniffers), the network traffic data (e.g. the number of scans from designated sources and destinations,) can be collected and used to detect attacks. To provide an efficient and fast data retrieval with processing for threat detection, the collected data shall be normalised in a specific format.

3.1.2 Cloud infrastructure

The cloud computing infrastructure is composed of multiple distributed servers, which are responsible for provisioning storage and computing resources to cyber security applications. Pushing storage, computation and analysis to the cloud will not only resolve the issue of the limited resources on an end-user device, but also significantly improve the efficiency of overall cyber security situation awareness through fast data retrieval and processing. More importantly, the implementation of a large data-processing technique such as the MapReduce will make the system more efficient through eliminating operation delays and enabling real-time processing of data streams. In addition, the reliability of the
system can be improved as servers in the cloud infrastructure are immune to the single node of failure.

### 3.1.3 Monitoring centre

In our system, the monitoring centre holds the intelligent role and is responsible for analysing the stream data stored in the cloud to perform cyber security situation awareness. The monitoring centre interacts with end-user devices and cloud servers by pushing cyber operation policies and configuration to the monitored end-user devices and cloud servers. To compound this system, the data visualisation will provide the monitoring centre to effectively conduct the cyber security situation awareness to deal with emergent and dangerous cyber threats.

### 3.2 Data storage module

In the cloud infrastructure, a number of cloud storage servers are distributed across multiple locations to collect a large amount of raw data on dispersed end-user devices. The distributed cloud storage can avoid the bottleneck of the centralised storage. For example, the cloud storage server provides bi-directional data synchronisation capability to end-user devices. The data stream can be transmitted to the storage server based on an optimal route, reducing the time needed for storage. Distributed cloud storage servers can store data based on the locations of end-user devices over time. With the contiguous data stream stored in the cloud, the status of storage server can be formulated as

\[
\int_{t_n}^{t_{n+1}} B_j(x) \, dx + Q_j(t_n) = Q_j(t_{n+1}),
\]

where \(0 \leq Q_j(t_{n+1}) \leq \delta \cdot Q_j^{\max}\), \(\delta\) is the alert threshold and \(Q_j^{\max}\) is the maximum capacity of storage server \(j\). When the \(Q_j(t_{n+1}) > \delta \cdot Q_j^{\max}\), the alert will be issued and broadcasted to the application servers if there is a little storage space on server \(j\).

To rapidly and efficiently store a large amount of data and minimise the delay of storage process, we need to consider the following constraints: (i) the delay for the big data storage mainly consists of a data propagation delay and a transmission delay; (ii) the data propagation delay is affected by the distance \(L_{ij}(t)\) between the end-user device \(i\) \((i \in [1, m])\) and the cloud storage server \(j\) \((j \in [1, n])\); (iii) the data transmission delay relies on the available link bandwidth \(B_{ij}(t)\); (iv) the sum of supported bandwidth to each device cannot exceed the total available link bandwidth \(B_j\) on the storage server; (v) the size of data in the storage server \(j\) cannot exceed the maximum storage capacity; (vi) the data-set from a user device \(i\) shall be split into at least \(S_i\) chunks and stored in different storage servers; (vii) all the distributed stored data chunks associated with the user device \(i\) can restore the whole data-set associated with the user device \(i\).

Based on the above constraints, we formalise the optimal data storage process as an optimisation problem, which is listed as follows:

\[
\text{Objective.} \quad \min \left\{ \sum_{i \in [1,m]} (T_i^1 + T_i^2) \right\}
\]
\[
\begin{align*}
\text{S.t.} & \\
\forall i \in [1, m], \ \forall j \in [1, n], & \\
\sum_{j \in [1, n]} (\alpha \cdot L_{ij} \cdot x_{ij}) &= T^1_i, \\
\sum_{j \in [1, n]} \left( \frac{d_{ij}}{B_{ij}} \cdot x_{ij} \right) &= T^2_i, \\
\sum_{i \in [1, m]} (x_{ij} \cdot B_{ij}) &\leq B_j, \\
\sum_{i \in [1, m]} (x_{ij} \cdot d_{ij}) + Q_j(t) &\leq Q^{\text{max}}_j, \\
\sum_{i \in [1, m]} x_{ij} &\geq S_i, x_{ij} \in \{0, 1\}, \\
\sum_{j \in [1, n]} (x_{ij} \cdot d_{ij}) &= d_i,
\end{align*}
\]

where \( \alpha \) is a constant, \( d_i \) is the size of data collected from user device \( i \) and \( d_{ij} \) is the data chunk size of user device \( i \). When the network traffic is low, few data streams share the bandwidth concurrently, so that the propagation delay \( T^1_i \) plays a key role in the total delay of the storage process. Hence, using the shortest path to route the traffic can incur the smallest propagation delay because \( L_{ij} \) is the main factor. When the network traffic is higher, the transmission delay incurs a larger impact on the total delay of the storage process in comparison with the propagation delay. The idle or not fully loaded storage server will be selected to receive data in order to reduce the delay in the data transmission process. Hence, the storage server selection should be determined by the available bandwidth \( B_{ij} \) to the user device.

Nonetheless, the computation overhead for the optimisation increases rapidly with an increase in the scale of cloud. To overcome this, we consider dividing the large network into multiple regions based on the locations of user devices. The optimal data storage process can be performed in each region and then the optimal data storage process can be performed across multiple regions.

### 3.3 Task scheduling module

When there are large amounts of data stored in the cloud, how to effectively schedule the data processing is a challenging issue. To this end, an optimal task scheduling model should be developed to assign tasks optimally to slave nodes with an objective of achieving the shortest data-processing time. To minimise the data-processing time, all the tasks shall be balanced to different servers that perform the computation. In an ideal condition, when all the servers concurrently complete the computations of tasks, the time taken to process all tasks shall be the shortest. Nonetheless, in reality, all the servers cannot complete all computations of tasks simultaneously.

As such, the problem of the optimal task scheduling can be formalised by considering the following constraints: (i) the objective is to minimise the variance of data-processing time on all servers; (ii) the data-processing time of each server is affected by the server computation speed, data-set size and available bandwidth of the server; (iii) the server computation speed is related to the status of CPU and memory; (iv) each data shall be processed in one server. With the aforementioned constraints, the optimal task scheduling
model can be formalised as follows:

\[
\text{Objective.} \quad \text{Min} \left\{ \text{Max} \left\{ \sum_{s \in [1,z]} \left( T_s - \bar{T}^2 \right) \right\} \right\} \\
\text{S.t.} \\
\forall k \in [1,w], \quad \forall s \in [1,z], \\
\sum_{s \in [1,z]} T_s \cdot \frac{1}{z} = \bar{T}, \\
\sum_{k \in [1,w]} \left[ x_{ks} \cdot \left( \frac{d_k}{v_s} + \frac{d_k}{B_s} \right) \right] = T_s, \\
\omega_1 M_s + \omega_2 C_s = v_s, \\
\sum_{s \in [1,z]} x_{ks} = 1, \quad x_{ks} \in \{0, 1\},
\]

where \( w \) is the number of tasks, \( z \) is the number of servers, \( T_s \) is the data-processing time on the server \( s \), \( \bar{T} \) is the average data-processing time on all servers, \( d_k \) is the data size of the task \( k \), \( B_s \) is the available bandwidth of the server \( s \), \( v_s \) is the computation speed of the server \( s \), \( M_s \) is the memory status of the server \( s \), \( C_s \) is the CPU status of the server \( s \) and \( \omega_1 \) and \( \omega_2 \) are the weight coefficients. Based on the modelling, we can see that when \( \bar{T} = T_1 = T_2 \cdots = T_z \), the objective function can be minimised such that all the servers and all the tasks can be computed in the shortest time.

### 3.4 Implementation

We implemented the cloud computing based system for cyber security management, which consists of one master node and four slave nodes, as shown in Figure 2. The slave node can support both the map worker and reduce workers. In each node, DELL Optiplex 9010 computer with Intel Core i7 3.40 GHZ 8 processors and 16 GB RAM and 2 TB hard drive is used. We use the Cloudare Manager as a single and central interface to carry out the configuration, management and monitoring of the designed system.

We downloaded the cloudera-manager-installer.bin from the Cloudera website. We configure it with the executable permission by the command ‘chmod u+x cloudera-manager-installer.bin’ and the installer can be executed with the command ‘sudo./cloudera-manager-installer.bin’ to install the Cloudera Manager. The server is set on port 7180. To install Cloudera Manager on hosts in the cloud, the Cloudera Manager Admin Console is used to install and configure CDH (Cloudera Distribution including Apache Hadoop). It is worth noting that CDH is an open-source powerful management and automation tool to deploy Apache Hadoop widely.

We now use the scenario of analysing network traffic data to find anomalous behaviour as an example to show how our developed prototypical system performs cyber security management. Our developed system workflow consists of four steps: data collection, data normalisation, data computation and data visualisation, which will be explicitly introduced as follows:

- **Step 1: Data collection.** The network monitoring agent is distributed in the network to collect the traffic data and monitor the suspicious activities from the end-user...
devices. It is worth noting that the collected data are unstructured and unreadable, which needs to be normalised. For example, the collected traffic data are stored in the network traffic trace files with the PCAP format, which is binary and not readable, as shown in Figure 3.

- **Step 2: Data normalisation.** The collected data are normalised and stored in HDFS. To filter out the useless information and normalise the collected data, we use a tool called tshark, which is a wireshark network analyser, which integrates the packet capture function and the PCAP format data-reading function. tshark normalises the unstructured data into the structured data with a specific format. For example, the network traffic data can be normalised based on the characteristics of frame number, source IP address, destination IP address and port number, as shown in Figure 4.

- **Step 3: Data computation.** In this step, the normalised data are computed using the MapReduce framework introduced in Section 2.

- **Step 4: Data visualisation.** To help the network security administrator to detect anomalies in the network and present useful information to defend against cyber attacks, the data visualisation is implemented to transform the network traffic data such as IP addresses and other features (e.g. time, ports and others) into images, which can better present the attack consequence and scenes.
4. Performance evaluation

To evaluate the effectiveness of our developed system, we use the network traffic data from [http://www.caida.org/home/](http://www.caida.org/home/) to perform experiments. The data consist of a 160.8 GB data file and split into 3 GB per chunk. In our evaluation, we consider two representative scenarios: ranking and aggregation, as examples to demonstrate the effectiveness of our developed system.

To identify whether or not end-user devices have high scan traffic rates, we implemented the ranking primitive, which has been widely used for cyber security situation awareness. Figure 5 illustrates the workflow of the ranking primitive implementation in the MapReduce framework, in which the host IP address and destination IP address pair is defined as the key and the count as the value. Users define the map function to list the count for the key and the reduce function to rank the count. Based on the map function and the reduce function defined by users, the master will schedule tasks to the map worker and track the status of tasks. With the assigned tasks, the map worker will read the data from the distributed file system and match the data to the defined key when the count is recorded. The shuffling and sorting operation will group all counts with the same key in intermediate results and the sort results are based on the key. Then, the reduce worker will count the ranking results to identify the host IP addresses, which have the high scan traffic rate. Figure 6 illustrates the result for conducting port ranking. As we can see, the port 80 is most scanned, which is about 138 millions.
In cyber security situation awareness, the aggregation is another important primitive, which can be used to consolidate the analysed results and conduct statistical analysis, as a critical part of intrusion detection. The traffic data features (e.g. time, source address, destination address and port number) can be used as the key to perform the aggregation. Using the port number as the aggregation key, the information related to the key will be defined as the value, which presents the volume of data associated with the designated port. The workflow of traffic aggregation is similar to the ranking in the MapReduce framework. The large amount of network traffic data will be split and assigned to the map worker. The map worker finds the information related to the port number and lists them. The reduce worker will then aggregate all the related information with the same key and then generate the final aggregation results.

Figure 7 shows a screenshot of the ranking in the MapReduce framework evaluation result. We summarise all the evaluation results in Table 1. We can see that in order to process 160.8 GB of data, the processing speed of the ranking and the aggregation can be improved after the MapReduce framework is introduced. With an increase in the data size, the time for processing data will increase as well. Table 2 presents the relationship between the processing time and the number of slave nodes. As we can see, with more slave nodes used in the system, more computation resources can be applied to process data, leading to a declined trend of data-processing time. For example, to process...
160.8 GB of data for the ranking and the aggregation using four slave nodes, the processing time are 1043 and 842.15 s, respectively.

To present useful information to security administrators, we also implemented the network traffic visualisation using the MapReduce framework as shown in Figure 8. Here, the horizontal axis represents the source IP address and the vertical axis represents the destination IP address. The data point in this figure represents the communication density of a pair of nodes. As we can see, the address 0.3.143.223 has a high scan traffic and is highly likely to be a worm propagation host.

Table 1. Time versus the size of data.

<table>
<thead>
<tr>
<th>Data size (GB)</th>
<th>Ranking algorithm with <em>MapReduce</em></th>
<th>Ranking algorithm without <em>MapReduce</em></th>
<th>Aggregation algorithm with <em>MapReduce</em></th>
<th>Aggregation algorithm without <em>MapReduce</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>39.8</td>
<td>280</td>
<td>1401.76</td>
<td>205.44</td>
<td>1047.53</td>
</tr>
<tr>
<td>63.1</td>
<td>433</td>
<td>2222.382</td>
<td>333.46</td>
<td>1660.79</td>
</tr>
<tr>
<td>87.9</td>
<td>598</td>
<td>3095.8</td>
<td>460.35</td>
<td>2313.5</td>
</tr>
<tr>
<td>112.3</td>
<td>741</td>
<td>3930.5</td>
<td>588.14</td>
<td>2955.7</td>
</tr>
<tr>
<td>139.2</td>
<td>913</td>
<td>4902.6</td>
<td>728.02</td>
<td>3663.7</td>
</tr>
<tr>
<td>160.8</td>
<td>1043</td>
<td>5663.4</td>
<td>842.15</td>
<td>4232.2</td>
</tr>
</tbody>
</table>

Table 2. Time versus number of slave nodes.

<table>
<thead>
<tr>
<th>Number of slave nodes</th>
<th>Ranking with <em>MapReduce</em></th>
<th>Aggregation with <em>MapReduce</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4200</td>
<td>3346.7</td>
</tr>
<tr>
<td>2</td>
<td>2061</td>
<td>1644.69</td>
</tr>
<tr>
<td>3</td>
<td>1364</td>
<td>1094.21</td>
</tr>
<tr>
<td>4</td>
<td>1043</td>
<td>842.15</td>
</tr>
</tbody>
</table>

5. Discussion

In addition to performing the cyber security management, our proposed system is generic and can be extended to other applications, including mobile security and transportation systems.

Due to the popularity of use, smart mobile devices have become burgeoning targets for the cyber adversary. By compromising mobile devices, the adversary can steal private information from organisations.[6] Hence, the development of an effective threat monitoring system for monitoring mobile devices becomes an important issue. To deal with this issue, in our previous work,[5,7] we investigated the malware detection on mobile devices using machine learning schemes. Nonetheless, mobile devices have very limited computing and storage resources, our cloud-based system can be extended as a mobile cloud system to detect malicious activities on a large number of mobile devices.

In transportation systems, constantly monitoring variations in traffic characteristics (e.g. densities and speeds,) and processing the collected massive data-set are critical to
improving highway safety. To tackle this issue, our proposed cloud-based system can boost the efficiency of information management for transportation systems as well. As a proof-of-concept, we use our developed system to perform traffic analysis on the Maryland highway traffic data. In our experiment, 128 GB of data-set was split into 650 MB per chunk. We performed the rank primitive that computes the average speed in different zones to show the performance of our developed system, the data were distributed to the map worker to compute the average speed based on zone identifier. Then, the reduce worker will help to sort the average speed on the zone. Our experimental data show that the MapReduce framework can significantly improve the efficiency of data processing and more nodes can improve the processing time.

6. Related work

In the realm of cyber security, big data is a challenging issue for preventing and detecting network attacks.[8–12] For example, to detect botnets, a large amount of network traffic data, which include botnet traffic data and benign traffic data, need to be monitored,
collected and processed, which will face the computational bottleneck.[10] To quickly discover the insightful information of attacks, analysing a large number of data patterns in real time becomes a necessary need for intrusion detection systems.[8] In addition, Lee and Lee [11] noted that with the network traffic growing up, how to efficiently analyse a huge amount of traffic within the affordable response time becomes a challenging issue for a defence system to deal with the distributed denial-of-service attacks.

To provide efficient data storage in the cloud, a number of research efforts have been performed.[1,10,14,15,22,35,40] For example, Zeng et al. [19] proposed a layered and cooperative architecture of the cloud storage system. Abu-Libdeh et al. [13] proposed to apply Redundant Array of Inexpensive Disks (RAID) techniques to store data across multiple providers, which can reduce the storage-switching costs and achieve a high resilience against outages and failures. Kamara and Lauter [16] proposed a combination of recent and non-standard cryptographic primitives to build a secure cloud storage on top of a public cloud infrastructure. Harnik et al. [15] investigated the side channels in a cloud environment. Wu et al. [14] provided a comprehensive review of the key technologies in cloud computing and cloud storage and the types of cloud services, and described the advantages and challenges of cloud storage. Wang et al. [18] proposed a scheme for confidential data sharing on cloud servers.

To conduct big data analysis, there are a number of research efforts using the MapReduce framework.[3,20–33] For example, Dean and Ghemawat [22] introduced the MapReduce programming model and demonstrated how the MapReduce works with the implementation in a large cluster. Lin and Dyer [3] investigated the MapReduce algorithm design with a focus on text processing (e.g. natural language processing, information retrieval and machine learning). Morken [29] studied the MapReduce model and the frameworks for netflow data processing. Ebrahimi [24] investigated how to solve linear programs using MapReduce. Alves et al. [20] investigated an elastic streaming MapReduce for distributed data stream processing. Halim et al. [26] designed and implemented a MapReduce-based max-flow algorithm to process large small-world graphs. Osterman et al. [30] investigated the potential and the usability of cloud computing for the scientific community. Gunarathne et al. [25] studied the implementation of the MapReduce in the cloud for science applications. Shivhare et al. [31] investigated the pros and cons of cloud computing in storing and processing big data and discussed potential solutions. Chen et al. [21] evaluated a cloud-based security centre for traffic data forensic analysis and proposed a collaborative cyber security management system for data collection and analysis using a parallel programming paradigm in a cloud computing platform. Liu and Orban [27] leveraged the MapReduce technology to build a highly scalable system on top of a scalable cloud. Tan et al. [32] developed a tool, namely Kahuna, to diagnose performance problems in MapReduce systems. Zhang et al. [33] developed a simple Matlab-to-MapReduce translator for cloud computing, namely M2M, for the basic numerical computations capable of translating a Matlab code with up to 100 commands to a MapReduce code in just a few seconds.

As an open-source software framework and an implementation of MapReduce algorithm, a number of research efforts have been conducted to use Hadoop to process large-scale data-sets on large clusters.[4,34–41] For example, White [4] introduced and demonstrated how to process big data using Hadoop. Borthakur [34] introduced the distributed file system architecture and designs in Hadoop. Shvachko et al. [39] described the architecture of a Hadoop File system and provided an implementation report on managing 25 petabytes of enterprise data. Kambatla et al. [35] investigated the performance of existing solutions in optimally provisioning the MapReduce job in the
Hadoop File system environment. Xie et al. [41] proposed a data placement scheme, which balances data across nodes before processing the load on a heterogeneous Hadoop MapReduce cluster. Sandholm and Lai [37] proposed a dynamic priority parallel task scheduler for Hadoop. Their scheduler enables the user to control their allocated capacity and dynamically adjust their spending based on the current demand for cloud services. Shafer et al. [38] investigated the root causes of Hadoop performance bottleneck and discussed the tradeoffs between portability and performance in the Hadoop-distributed file system.

Different from the existing research efforts, in this paper we leverage a cloud computing based system to assist cyber security management, which can improve data storage efficiency, speed up access to data and eliminate operational delays in handling real-time data stream for cyber security situation awareness.

7. Conclusion

In this paper, we investigated a cloud computing based system for cyber security management to fasten the analysis of large network traffic data. Using the MapReduce framework, we designed and implemented a system consisting of end-user devices, cloud infrastructure and a monitoring centre. To make our proposed system efficient, we introduced two key function modules of our system: data storage module and task scheduling module. We performed the MapReduce implementation using Apache Hadoop and the implemented system consists of data collection, data normalisation, data computation and data visualisation. Using ranking and aggregation as primitive functions for cyber security management, we carried out extensive experiments and our data show that our system can efficiently analyse a large network data. Finally, we discussed how to extend our proposed system to other applications.

References


