Monitoring and Analyzing Big Traffic Data of a Large-Scale Cellular Network with Hadoop

Jun Liu, Feng Liu, and Nirwan Ansari

Abstract

Network traffic monitoring and analysis is of theoretical and practical significance for optimizing network resource and improving user experience. However, existing solutions, which usually rely on a high-performance server with large storage capacity, are not scalable for detailed analysis of a large volume of traffic data. In this article, we present a traffic monitoring and analysis system for large-scale networks based on Hadoop, an open-source distributed computing platform for big data processing on commodity hardware. This system has been deployed in the core network of a large cellular network and extensively evaluated. The results demonstrate that the system can efficiently processes 4.2 Tbytes of traffic data from 123 Gb/s links with high performance and low cost every day.

With the proliferation of powerful mobile devices, innovative mobile applications, and increased spectrum availability, cellular networks are emerging as the primary means for accessing the Internet. In order to support the explosive growth of data volume, cellular network operators need to manage and plan their network resources accordingly. Hence, the ability to accurately and extensively monitor and analyze network traffic is a fundamental necessity for network operations and optimization.

Over the years, a number of methodologies and tools have been developed for traffic monitoring and analysis. They can be categorized into two disciplines: active and passive approaches. The active approach generates probe traffic to induce measurable effects for extracting information of network status. Drawbacks of the active approach include interference to the normal traffic by injected probes and the questionable precision of inferred conclusions about the overall network. In contrast, the passive approach studies the network as an unperturbed system. Passive tools do not generate probes into the network or change the original traffic, but silently observe and analyze the traffic in the network. They are preferred in managing and planning commercial networks.

Essentially, the enablers for passive monitoring and analysis of commercial networks are high-speed traffic acquisition devices, large-capacity storage hardware, and high-performance computing servers. In general, the traffic acquisition devices collect traffic data at two levels, the packet and flow levels. Both of them require the network management system to employ a large storage system and high-performance computing server to store and analyze the traffic data from a huge number of network devices. However, as networks grow exponentially, administrators face the big data challenge, that is, managing and processing a huge amount of traffic data (e.g., terabyte or petabyte packets or flow records). Traditional network measurement systems, which usually rely on a centralized high-performance server, can no longer handle this huge amount of data. An efficient and scalable traffic monitoring and analysis system is thus called for.

In contrast, the explosive growth of the Internet's population and data-intensive applications has advanced the research and development of scalable platforms and technologies for data mining, commonly used by large Internet companies. Two of the major emerging technologies are GFS [1] and MapReduce [2] developed by Google. The combination of GFS and MapReduce allows developers to manage and process a massive amount of data in a distributed manner over thousands of commodity computers. Inspired by GFS and MapReduce, the open source community produces the Apache Hadoop [3] sponsored by Yahoo!. Due to its low cost and high efficiency, Hadoop is widely used as a distributed computing framework to conduct large-scale data processing. Hadoop possesses several valuable features: efficient distributed parallel computing, low-cost scale-out capability, and high fault tolerance. These features enable Hadoop to be a suitable platform for large-scale network measurements. However, it is a great challenge to tailor Hadoop for network measurements because it was originally developed for batch-oriented tasks, such as text mining and web page indexing. Nevertheless, some efforts have been made toward this goal. For example, Lee and Lee [4] developed a Hadoop-based tool to process *libcap* packet trace files in a parallel manner. RIPE [5] aims to provide a library based on Hadoop to analyze *PCAP* files. They made the first attempt to process the raw packet data in Internet. In addition, authors in [6, 7]

Jun Liu and Feng Liu are with Beijing University of Posts and Telecommunications.

Nirwan Ansari is with the New Jersey Institute of Technology.

0890-8044/14/$25.00 © 2014 IEEE

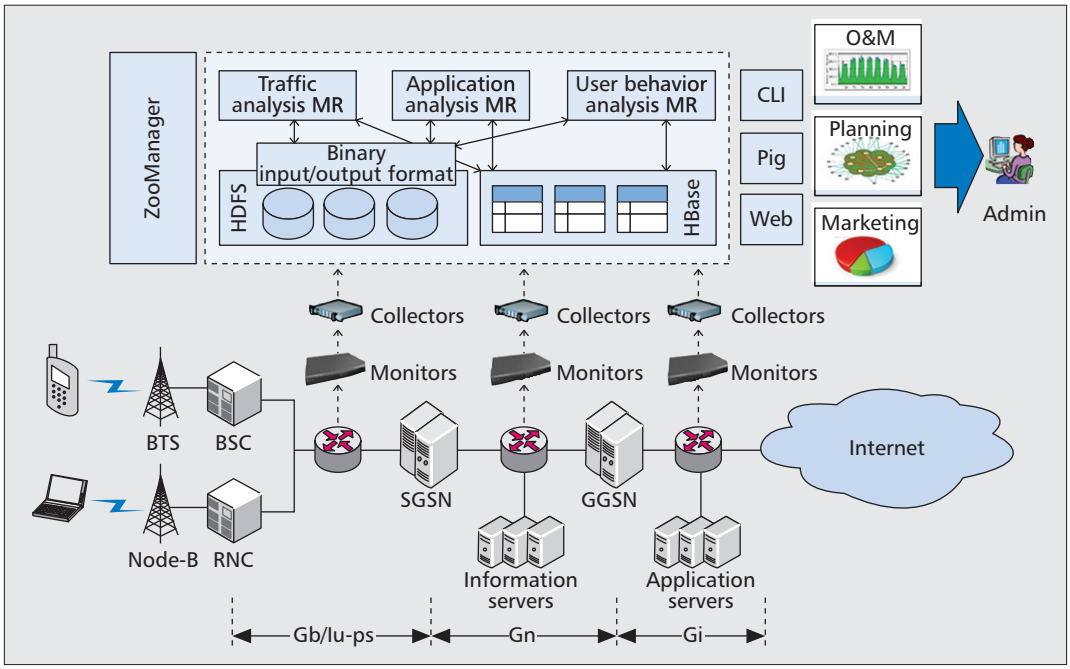


Figure 1. System architecture.

investigated how to analyze large volumes of network data by Hadoop-based tools like MapReduce and Pig. Although these initial works have demonstrated the viability of using Hadoop for analyzing network traffic data, some important issues in large-scale commercial telecommunication networks have not been addressed.

In this article, we propose a Hadoop-based scalable network traffic monitoring and analysis system for big traffic data. The system is designed and implemented following a multi-layer architecture with functional components including high-speed traffic monitors, traffic collectors, data store, MapReduce analysis programs, result presentation interfaces, and a cluster manager. To prove the viability of the proposed system, we deploy the system into the core network of a large-scale second/third generation (2G/3G) cellular network. The results demonstrate that Hadoop is a promising enabler for building an efficient, effective, and cost-efficient large-scale network traffic monitoring and analysis system.

Hadoop-Based Large-Scale Network Traffic Monitoring and Analysis System

Figure 1 shows the architecture of our proposed Hadoop-based network traffic monitoring and analysis system. An exposition of its major components is offered next.

Cellular Data Network

Generally, the cellular data network can be divided into two domains: access and core. The access is composed of a number of radio access networks (RANs) including Global System for Mobile Communications (GSM)/Enhanced Data Rates for Global Evolution (EDGE, 2G) and Universal Mobile Telecommunications System (UMTS)/High-Speed Data Packet Access (HSDPA, 3G) RANs. A mobile device communicates with a transceiver station in the access network, which forwards its data service traffic to a serving general packet radio service (GPRS) support node (SGSN). The SGSN establishes a tunnel on the Gn interface with a gateway GPRS support node (GGSN) that provides connectivity to the Internet via the Gi interface. Through this path, the requesting message of a mobile device enters the Internet and reaches the serving server. Data responding from the server to the mobile device traverse in the reversed path.

Traffic Monitor

To meet the increasing bandwidth requirement, mobile operators are deploying more network lines at the gigabit level, such as 10G (OC-192) and 40G (OC-768). In such a high-speed environment, even with offloading network interface cards and optimized kernel code, software-based traffic monitoring systems are still inadequate to achieve real-time monitoring. The hardware-based tailor-made collector system is the only option for networks with multiple 10 Gb/s links.

We design and implement a flexible monitoring system based on a scalable hardware/hardware-combined architecture to cope with the modification and addition of monitoring requirements as well as future rate increase. The hardware component, named *Traffic Monitor*, is in charge of the performance-critical tasks such as deep packet inspection, flow record composition, and basic packet statistics. Other resource-intensive tasks like application-specific record generation are executed by software components. We utilize the reconfigurable capabilities of field programmable gate arrays (FPGAs) and parallel processing techniques to achieve both flexible and high-speed concurrent packet processing.

Key features of Traffic Monitor are:

1. Line-speed packet parsing: The objective of packet parsing is to extract expected attributes of header and contents of payload. We implement a set of parsing algorithms with interpretation rules for different types of protocol formats like PPPoE and GRE to capture packets at various network interfaces, such as Gb, Gn, and Gi.
2. Accurate flow composition: A flow is a collection of packets between two end nodes defined by specific attributes, such as the well-known TCP/IP five-tuple. Metrics of flows can illustrate more useful information than individual packets. The flow manager component generates and updates the flow records stored in the flow cache. Completed flows, which are terminated by TCP FIN packets or time-out events, are exported by the flow exporter as *Flow Record Frame* in a predefined binary format.

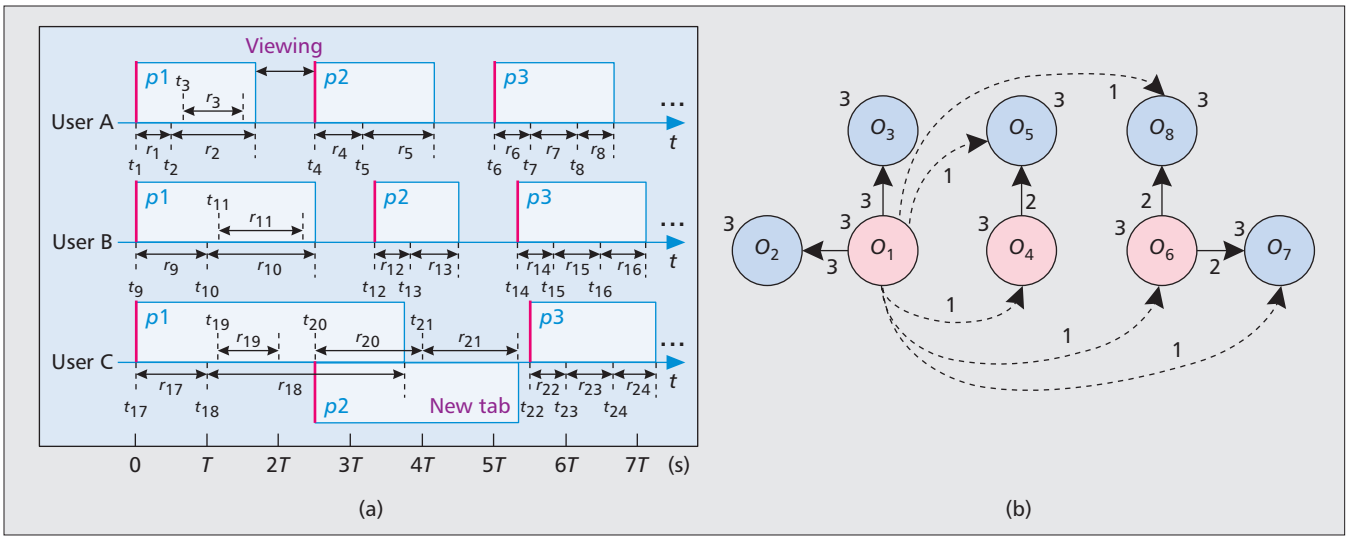


Figure 2. HTTP click request identification: a) web browsing behavior; b) HTTP request dependence graph.

3. Real-time traffic classification: Traffic classification is fundamental to numerous network management tasks. The traffic classifier component provides real-time traffic classification with 14 service classes and 129 sub-service classes by combining port-based application deduction, deep packet inspection, deep flow inspection, and statistical patterns-based technologies. Classification results are exported into two fields of a Flow Record Frame.
4. Multi-level traffic statistics: The Traffic Monitor supports traffic statistics at three granularities: packet, flow, and aggregated. At the packet level, information on each packet, such as bytes and arrival time, is accumulated to produce the statistics report. The flow-level statistics provide detailed information of each observed flow like duration and number of packets. Statistics at the packet and flow levels can be aggregated from different perspectives to generate various aggregated reports, such as by user identity and service class.
5. Fine-grained traffic shaping: To support QoS differentiation, traffic shaping can be conducted from different aspects, such as subscriber, IP range, and service class, which can be configured through a web interface. Hardware-based cascade token bucket architecture and algorithms facilitate real-time traffic shaping with 1 kb/s granularity at a multi-gigabit line speed.
6. Packet mirror: To support offline traffic analysis, packet passing through the traffic collector can be mirrored to other devices like our *Traffic Collector* via a cable splitter or dumped into trace files.

Traffic Collector

The Traffic Collector receives two kinds of input from the traffic monitor via Gigabit Ethernet: flow record frames as UDP stream data and specific mirrored packets as encapsulated Ethernet frames. With these inputs, the Traffic Collector performs three functions: extracting flow records from flow record frames, generating application-layer session records, and uploading records to the *Data Store*.

To improve the performance of processing and transmission, flow record frames produced by the traffic monitor are encapsulated in binary format at the bit level; that is, a field of a flow record may reside in one or multiple bits. Moreover, each flow record frame consists of 16 flow records with a common header, which includes several common attributes like sampling rate and link identity. This compact format helps save the transmission bandwidth, but presents challenges for subsequent analysis tasks. To tailor the flow record frame for Hadoop processing, the traffic collector extracts information from the flow record frame and translates each of them into 16 flow records. Each flow record consists of a fixed length of 92 bytes of flow information, such as sample rate, timestamp, TCP/IP five-tuple, and number of packets.

As network applications become increasingly complex and heterogeneous, there is an increasing need for application-oriented traffic analysis. However, most existing network traffic monitoring systems just capture and store raw packets. The complex application-layer operations like HTTP session reassembly are left to back-end analysis. This leads to increased analysis complexity and reduced performance due to excessive packet copies between the traffic probe and the analysis subsystem. To bridge this gap, we develop a pluggable generator to produce application-layer session records in Length-Value (LV) binary format. It addresses the following key challenges:

1. It can handle massive traffic data generated from multi-gigabit network links in real time. We use the PF_RING, a technology that can dramatically improve the packet capture speed, to achieve high-speed packet capturing.
2. It supports parallel processing of packets and sessions on multi-core machines. The record generator transparently creates a number of worker threads, which equal the number of allocated cores, to process packets and application sessions.
3. It flexibly supports variable application-layer protocols. To efficiently keep track of known and potentially growing application-layer protocols, the session assembling rules of different protocols are defined in configuration files, which are loaded into memory at runtime. Multiple protocol modules can process the same packet in a shared memory to avoid unnecessary data copies and improve the overall performance. Based on this scalable and flexible architecture, the record generator can produce HTTP session records for a 10 Gb/s network line using an 8-core machine.

The last function of the Traffic Collector is to transmit extracted flow records and generated application-layer session records to HDFS. The transmission task is performed by an HDFS uploader, which is an isolated Java process. The uploader reads the buffered flow records and application-layer session records, and exports them to HDFS by using the Hadoop stream writer. To expedite the parallel processing of traffic data, the flow and session records are aggregated every five minutes and saved into HDFS.

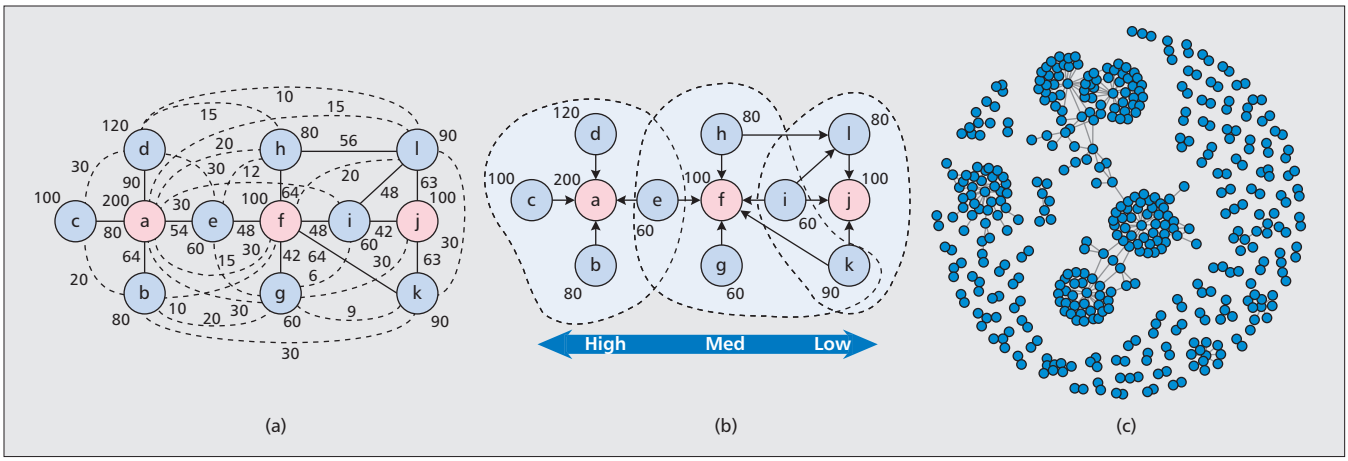


Figure 3. Website community identficiation: a) example of user distribution among websites; b) example sparsified affinity graph; c) identified website communities.

Data Store

In addition to real-time capturing packets, the scalability of storing a huge amount of traffic data is another significant requirement for large-scale network monitoring and analysis. To address this issue, we build a distributed data store by leveraging HDFS and HBase for managing unstructured and structured data, respectively.

HDFS, a core component of Hadoop, is designed for storing data in low-cost hardware with scalability and fault tolerance. It provides high-throughput and reliable access to petabytes of big data on thousands of commodity machines. HDFS is well suited to process the massive network traffic data. In our system, flow record and application-layer session record files are saved into HDFS via an HDFS stream writer application programming interface (API). These files are parallel processed by MapReduce programs, and some of the analysis results are written back into HDFS for presentation.

Although HDFS is suitable for storing large files, its documentation oriented design does not support fast random read and write records in files. For network monitoring and analysis, this limitation is unacceptable in some cases, such as searching flow records by TCP/IP five-tuple and updating traffic counting records of a given user indexed by the user identity. That is why we take HBase as another data store module to manage structured data. HBase is built on top of HDFS by storing data in indexed HDFS StoreFiles. Data stored in HBase is organized as key/value pairs in a columnar manner. It supports a flexible data model, fast access to rows indexed by the row key, and low-latency update to given records within a large data set.

Analysis Programs

To analyze the captured massive traffic data, we develop a set of programs based on the powerful MapReduce programming model. Each MapReduce program consists of a set of jobs composed by distributed Map and Reduce tasks. These tasks take a number of key/value pairs as inputs and produce other key/value pairs by user defined logics.

The developed MapReduce programs analyze traffic data from four aspects:

1. Network traffic statistics. We compute the traffic statistics in terms of byte, packet, and flow number according to different grouping indices, such as IP addresses, transport-layer protocols, and TCP/IP five-tuple.
2. Application-layer analysis. The traditional network-layer and transport-layer traffic analysis cannot keep pace with the growing emerging networked applications. The generated flow records and application-layer session records provide us an opportunity to investigate some data mining algorithms for extracting application layer information and study the traffic characteristics from the application perspective.
3. Web service provider analysis. We investigate web traffic over cellular data networks from the service provider perspective, which plays a critical role in the mobile Internet.
4. User behavior analysis. Improving user experience is the ultimate goal of network operators, and thus it is critical to understand user behavior and mobile service usage. Therefore, we characterize and model the traffic from the perspectives of the mobile client and the end user.

Result Presentation Interfaces

To present the analysis results in a flexible and user-friendly manner, we provide three kinds of interfaces: command line, Pig Latin scripting, and web graphic user interfaces (GUIs). Experienced developers who are familiar with Hadoop can use Hadoop commands to access results stored in HDFS files. If they want to investigate results from MapReduce programs running in different conditions, they can execute the programs by command line with various parameters. For analysts with a certain degree of scripting skills, they can explore the data by Pig Latin scripting with built-in functions and extended user defined functions, thus simplifying application development in the MapReduce framework by high-level data processing task abstractions. The last presentation interface is a web GUI for ease of viewing, comparing, and analyzing a data set with various interactive options.

Cluster Manager

Surveillance of Hadoop nodes to keep track of cluster health and maintain fault alert is critical to deploying Hadoop components into the production environment for running mission-critical applications. Although there are a number of experimental tools for Hadoop cluster management like Apache Ambari, providing an easy-to-use integrated Hadoop management system is still an open issue. We develop a tool named ZooManager based on two open source tools, Ganglia and Nagios, to ease the most important management task, Hadoop cluster monitoring. Ganglia can collect and aggregate performance metrics of servers such as CPU, memory, and disk and network utilization. We extend Hadoop's built-in plugins to publish performance metrics of Hadoop components like NameNode and DataNode to a Ganglia daemon. Based on these metrics of individual components, we define

Metric	Value
Link speed	12*10G POS + 3*GE~
Average traffic rate	24.5 Gb/s
Max traffic rate	50 Gb/s
~Bytes of flow record frame per day	875 Gbytes
Number of flow records per day	13.3 billion
Number of HTTP records per day	17.3 billion
Size of HDFS files per day	4.2 Tbytes
Accuracy of traffic statistics	100%
Accuracy of HTTP session records	100%
Average CPU Util of traffic collector	40.6%
Average Mem Util of traffic collector	21.9%
Average time for uploading 5 min records	98 s
Max time for uploading 5 min records	290 s (10 PM)
MapReduce jobs per day	18
Computing time per day	11 h 15 min
Cost of per gigabyte link speed	$201.2

Table 1. Overview of workload and performance.

several aggregated indicators to represent the overall status of the whole cluster and display them as time series via the web GUI. In addition, key metrics and indicators are fed into Nagios to produce real-time alerts to administrators via emails and SMS.

Traffic Analysis Algorithms

Application-Layer Analysis

Mining the logs of HTTP, the most widely used application layer protocol, is an effective way for service providers to understand user behavior and interests. For modern websites, a click usually triggers a number of HTTP requests for downloading embedded objects. Therefore, identifying clicks from a large number of captured HTTP requests is the prerequisite for web usage mining. However, the prevalent parallel web browsing behavior caused by multi-tab web browsers, shown by an example in Fig. 2a, presents great challenges to clicks identification. Hence, we propose a novel click identification algorithm based on dependency graph.

We model the complex browsing behavior as a *dependency graph*, which is a directed and weighted graph $G = (O, S, E, W)$. Each node o_i in the graph represents a web object identified by a URL. We divide the request sequence of each user into a number of sub-sequences. The rule of dividing the sequence is to make the interval between the first request in a sub-sequence and the last request in the previous sub-sequence bigger than τ, called *lookahead time window*. Then we can construct a set of edges E pointing from the first node to other nodes in each sub-sequence. The weight $w(o_i, o_j)$ of the edge $e(o_i, o_j)$ is the occurrence count of the pair $<o_i, o_j>$ appearing in all sub-sequences. Figure 2b shows the dependence graph derived from the browsing behavior example shown in Fig. 2a. For each node in the dependence graph, we define a dependence probability, which is the value of the weighted in-degree divided by the occurrence count of this node. We can determine whether a request is a click by comparing the dependence probability with a threshold, which is self-learned from the graph structure.

The algorithm is implemented as a MapReduce program executed on the cluster once every day for HTTP data preprocessing. It outputs URLs marked as click or non-click and their relationships into HBase for further web usage mining.

Web Service Provider Analysis

A website community structure is an essential means to describe the relationships among websites. Although there are a number of solutions for identifying website communities based on hyperlink structure [9] and content similarity measurement [10], the relationships between user behaviors and community structures are far from being understood.

We develop a three-step algorithm: measuring affinity, sparsifying a graph, and identifying communities, to extract communities by affinity measurement derived from user access information [12]. Formally, we model the mobile Internet websites as an *affinity graph* $G = (O, E, W)$, where O is the set of nodes representing websites and E is the set of directed edges with weights W denoting the affinity relationships between websites. The affinity weight from o_i to o_j is derived by dividing the number of common users between them by the total number of users of o_i. To reduce the computational workload and improve identification quality, the original dense affinity graph is transformed into a directed and unweighted $G' = (O, E')$ called a *sparsified affinity graph*. The structure of G' is determined by a rule of satisfying a scale-free fitting index *fit*, which is computed by the frequency distribution of node in-degree d_{in}^i to be larger than a threshold τ. Figure 3a shows an example of user distribution among websites. Nodes from *a* to *l* represent 12 websites. The number in bold close to each node is the number of users accessing this website. The number in italic close to each dashed line between two nodes is the number of common users of the two websites. This example can be transformed into a sparsified graph with $\tau = 0.69$, as shown in Fig. 3b. After generating the sparsified affinity graph, the *affinity rate* from node o_j to o_i is defined as the value of the number of users accessing website o_j divided by the out-degree of o_j if there is an edge from o_j to o_i. Then the influence score $inso_i$ of node (o_i) can be calculated by summing the affinity rates $affr(o_j, o_i)$ of all neighbor nodes o_j around o_i in the sparsified affinity graph. At last, all nodes are ranked by the influence score, and the top-k scorers are selected as the seed set R. Each node o_i in R and the set of nodes o_j having a directed edge from o_j to o_i in the sparsified affinity graph G' are grouped together as the ith community C_i.

A MapReduce program is implemented based on this algorithm with several parameters to be set by operators, such as scale-free index threshold τ and the number of top communities to be produced k. The identified website communities are presented to data analysts as graphs by a GUI tool.

User Behavior Analysis

With the high penetration rate of cellular devices and increasing number of fancy mobile applications, cellular devices are becoming indispensable to our daily lives; a cellular device and its usage data can thus be regarded as the natural extension of an individual [13]. Therefore, mobile operators are interested in knowing the attributes of cellular devices including models, prices, and features, and understanding related user behaviors. As the first step to address this challenge, it is imperative to extract the model information of cellular devices to determine features and device prices. Therefore, we design a novel Jaccard-based learning method to recognize cellular device models to help the operator build a model database. The main idea is to identify an appropriate keyword that represents a cellular device model from unformatted textual headers of HTTP requests in three steps:

1. Extract all keywords pertaining to the description of a device model.
2. Filter candidate keywords by evaluating the conditional probability value between each keyword and device model.
3. Calculate the Jaccard coefficient index using statistical information, and select the keyword with the highest Jaccard index to represent the device model.

To handle billions of HTTP records, we implement the proposed method as a MapReduce program running on the cluster. Recognized results and related information are stored in HBase for further analysis.

Experimental Results

System Evaluation

To evaluate the accuracy and performance of the proposed system, we constructed a cluster with only six servers at the most economical cost and deployed it in a commercial cellular network of a leading mobile operator in China. All machines are connected by 1 Gb/s Ethernet lines and configured with two 8-core 2.4 GHz Intel Xeon processors. Memory and hard disks of the NameNode and two traffic collectors are configured with 16 Gbytes and 4 × 2 Tbytes, respectively. Three DataNodes are configured with 16 Gbytes of memory and 8 × 2 Tbyte hard disks. The software environment of all nodes employs Linux CentOS 6.4, the 64-bit version, and Hadoop 1.0.4 with a block size of 64 Mbytes and replication parameter of two. Table 1 shows the overall workload and performance in this environment.

Before the deployment, we use a high-speed packet simulator to generate massive traffic for each traffic monitor to test the monitoring and analysis functions. From 100 Mb/s to 1 Gb/s generating speed, we examined the accuracy by comparing logs of the simulator and records in HDFS, and observed 100 percent accuracy for traffic statistics and HTTP session record generation.

Then the cluster is deployed into the production environment which covers 12 × 10G POS and 3*Gigabit Ethernet ports in the core network of a 2G/3G cellular network. The average and maximum traffic rates going through these links are 24.5 Gb/s and 50 Gb/s, respectively. Each link is attached to a traffic monitor. There are two traffic collectors that handle all of the traffic to generate and upload flows and HTTP records into HDFS every 5 min. The execution time for uploading records every 5 min is 98 s on average, which is linearly proportional to the size of a data set to be uploaded. The average CPU and memory utilization of the traffic collector are 19.9 and 21.9 percent, respectively. The traffic generates 4.2 Tbyte HDFS files every day. They are processed by 18 MapReduce jobs that require 11 h and 15 min to complete all analysis works. The performance result and the cost per gigabyte link speed of $201.2 have substantiated the high performance and cost efficiency of the Hadoop-based solution.

Website	Method	9–10	11–12	13–14	15–16	17–18	19–20	Avg
sina.com	DC	0.32	0.36	0.31	0.37	0.31	0.25	0.32
	DG	0.94	0.91	0.97	0.92	0.85	0.93	0.92
sohu.com	DC	0.28	0.27	0.33	0.33	0.19	0.29	0.28
	DG	0.96	0.89	0.95	0.79	0.90	0.90	0.90
ifeng.com	DC	0.34	0.37	0.38	0.29	0.35	0.33	0.34
	DG	0.85	0.73	0.88	0.79	0.77	0.89	0.82

Table 2. $F1$ scores of click identification results.

User Click Identification

We measured the accuracy of our proposed user click identification algorithm by $F1$ score, which considers both the precision p and the recall r as $F1 = 2 \times p \times r/(p + r)$ [14]. $F1$ scores of the identification results of three well-known websites in China are shown in Table 2. The results show that our method (DG) achieves higher accuracy than the widely used data cleaning (DC)-based method [15].

Website Community Identification

The website communities identified by our affinity measurement-based method are presented as a community graph like Fig. 3c by a GUI tool using the R code [11]. Based on this tool, we observed some community structures that cannot be identified by traditional link- and content-based website community discovery methods. For example, we found a community with the core qq.com surrounded by paipai.com, pengyou.com, tenpay.com, and so on, which are websites owned by the same big Internet company, Tencent, in China. Another interesting community is composed by a number of mobile game websites like angrybirds.com and zeptolab.com surrounding the core flurry.com, a traffic statistical service provider.

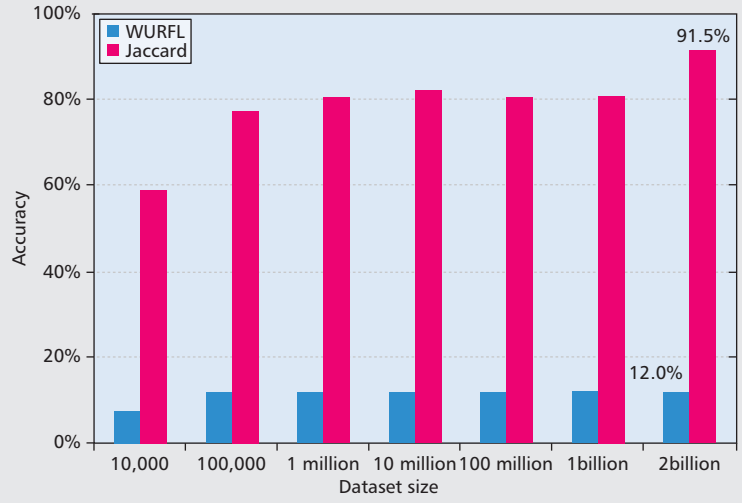
Mobile Client Recognition and User Behavior Analysis

To evaluate the accuracy of our mobile client recognition method, we employ the widely used WURFL [16] method for comparison. We conducted experiments on datasets with seven different sizes, from 10k to 2 billion HTTP records. Figure 4b shows the accuracy evaluation results. Note that our method produces a much better recognition result with 91.5 percent accuracy rate than WURFL's 12 percent accuracy rate. Some of the recognition results in HBase are shown in Fig. 4a.

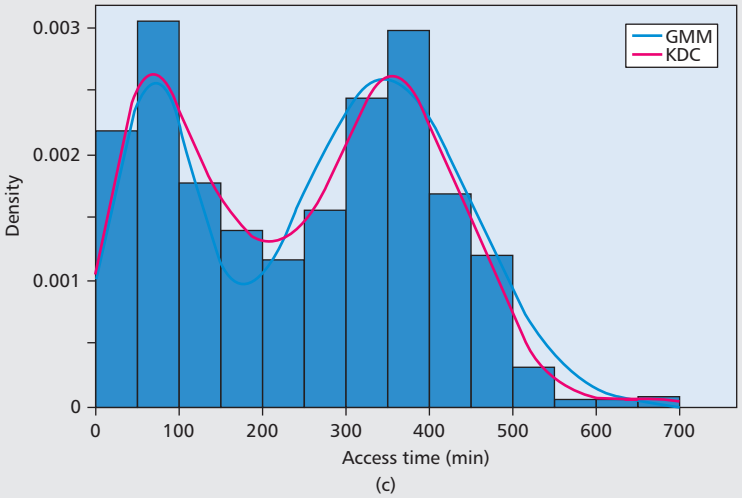
We further investigate a number of metrics for characterizing mobile user behavior. For example, we study the network access time, defined as the number of minutes in a day that a user accesses cellular data services. The density function estimated by the kernel density estimator (KDE) is shown in Fig. 4c. It is interesting to see that there are two humps with

TAC	Model	Jaccard
...	...	...
01284800	iphone	0.00597244
01285000	iphone	0.00622219
01285100	iphone	0.00605268
...	...	...
35201204	nokia5800	0.11106830
35201404	nokia5230	0.03753425
35201804	nokia5000	0.22978130
...	...	...
35172500	sonyericssonk700c	0.00919308
35238704	samsung-gt-s3650c	0.009820201
...	...	...

(a)



(b)



(c)

Figure 4. Mobile client model identification and user behavior analysis: a) example of identification results; b) accuracy of identification; c) density of network access time.

almost the same density value at around 70 and 350 min instead of a single peak. This kind of bimodal distribution implies that we can use the Gaussian mixture model (GMM) to characterize the distribution. We employ the Expectation-Maximization (EM) algorithm to estimate the parameters of the GMM. Figure 4c shows the original kernel density curve and the estimated GMM curve with the histogram. The fitting of these two lines indicates the distribution of network access time of cellular devices can be described well by GMM.

Conclusions

We have proposed a novel system for monitoring and analyzing large-scale network traffic data. The system utilizes the advanced distributed computing platform of Hadoop MapReduce, HDFS, and HBase. Based on this system, we have designed algorithms and implemented MapReduce programs for network traffic analysis from four different perspectives. The system was deployed into the core network of a commercial cellular network as a production system covering 123 Gb/s links and handling 4.2 Tbytes of input files every day. Evaluation results have demonstrated that the Hadoop-based solution is suitable for building a high-performance, cost-efficient, and scalable network traffic monitoring and analysis system. Moreover, the capability of this system to process big traffic data enables us to reveal a number of network traffic and user behavior phenomena not shown before. For future work, we plan to improve the performance of analysis programs by optimizing the MapReduce algorithm and tuning parameters of the Hadoop environment. In addition, we also plan to extend the current batch processing oriented system to support real-time traffic measurements by leveraging stream processing technologies.

References

[1] S. Ghemawat, H. Gobioff , and S. T. Leung, "The Google File System," *ACM SIGOPS Operating Systems Rev.*, vol. 37, no. 5, 2003.

[2] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Commun. ACM*, vol. 51, no. 1, 2008, pp. 107–13.

[3] Apache Hadoop, http://hadoop.apache.org/, accessed 9 Nov., 2013.

[4] Y. Lee and Y. Lee, "Toward Scalable Internet Traffic Measurement and Analysis with Hadoop," *ACM SIGCOMM Comp. Commun. Rev.*, vol. 43, no. 1, 2012, pp. 5–13.

[5] Large-Scale PCAP Data Analysis Using Apache Hadoop, https://labs.ripe.net/Members/wnagele/large-scale-pcap-data-analysis-using-apache-hadoop, accessed 9 Nov., 2013.

[6] T. Samak, D. Gunter, and V. Hendrix, "Scalable Analysis of Network Measurements with Hadoop and Pig," *IEEE Network Operations and Management Symp.*, 2012, pp. 1254–59.

[7] T. P. D. B. Vieira, S. F. D. L Fernandes, V. C. Garcia, "Evaluating MapReduce for profiling Application Traffic," *Proc. 1st ACM Wksp. High Performance and Programmable Networking*, 2013, pp. 45–52.

[8] Cisco Visual Networking Index: Forecast and Methodoloy, 2012-2107 http://www.cisco.com/en/US/soltions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf.

[9] J. M. Kleinberg, "Authoritative Sources in a Hyperlinked Environment," *Proc. 9th Annual ACM-SIAM Symp. Discrete Algorithms*, 1998, pp. 668–77.

[10] F. Ricca *et al.*, "An Empirical Study on Keyword-Based Web Site Clustering," *Proc. 12th IEEE Int"l. Wksp. Program Comprehension*, 2004, pp. 204–13.

[11] The R Project for Statistical Computing, http://www.r-project.org/, accessed 9 Nov., 2013.
[12] J. Liu and N. Ansari, "Identifying Website Communities in Mobile Internet Based on Affinity Measurement," *Computer Commun.*, vol. 41, 2014, pp. 22–20.
[13] G. Chittaranjan, J. Blom, and D. Gatica-Perez, "Who's Who with Big-Five: Analyzing and Classifying Personality Traits with Smartphones," *IEEE 15th Annual Int'l. Symp. Wearable Computers*, 2011, pp. 29–36.
[14] C. J. Rijsbergen, *Information Retrieval*, 1979.
[15] D. Tanasa D and B. Trousse, "Advanced Data Preprocessing for Intersites Web Usage Mining," *IEEE Intelligent Systems*, vol. 19, no. 2, 2004, pp. 59–65.
[16] Wireless Universal Resource FiLe Open Source Project (WURFL), http://wurfl.sourceforge.net/, [Online; accessed 9-Nov.-2013].

Biographies

JUN LIU (liujun@bupt.edu.cn) is the leader of the Network Monitoring R&D Base in the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications (BUPT). He received his B.E. and Ph.D. degrees from the Department of Information Engineering, BUPT, in 1998 and 2003, respectively. His research interests include network traffic monitoring and telecom big data analysis.

FENG LIU (liufeng2009@bupt.edu.cn) is a lecturer in the School of Information and Communication Engineering, BUPT. He received his B.E. and Ph.D. degrees from BUPT in 2004 and 2009, respectively. His research interests include network traffic data processing and big data analysis.

NIRWAN ANSARI (nirwan.ansari@njit.edu) is a Distinguished Professor of Electrical and Computer Engineering at the New Jersey Institute of Technology. He has (co)-authored over 450 publications, over one third in widely cited refereed journals/magazines. He has been granted more than 20 U.S. patents. Some of his recognitions include a couple of Best Paper awards, several Excellence in Teaching Awards, the Thomas Alva Edison Patent Award (2010), the New Jersey's Inventors Hall of Fame Inventor of the Year Award (2012), the NCE Excellence in Research Award (2014), and designation as an IEEE Communications Society Distinguished Lecturer.