TIFA: Enabling Real-Time Querying and Storage of Massive Stream Data

Jun Li[1,3], Shuai Ding[1,3], Ming Xu[1,3], Fuye Han[1,3], Xin Guan[1,3], Zhen Chen[2,3]
Department of Computer Science & Technologies[1]
Research Institute of Information Technology[2]
Tsinghua National Laboratory for Information Science and Technology (TNList)[3]
Tsinghua University, Beijing 100084, China
Email: flank920@gmail.com

***Abstract*—With the proliferation of network applications and user groups, the network stream data scale are getting larger and larger. To manage and analyze the massive data to get better service for end users (a.k.a. network optimization) or understanding of impacts of new network applications are attracting many researchers. This paper focuses on the storage and real-time retrieval of network traffic for analysis. We address this problem by integrating Time Machine and FastBit as a real-time querying and storage of massive stream data system, named *TIFA*. TIFA is designed for operation in Gbps network, accompanied with UTM as a module or an independent device. It leverages multiple large volume storages, and can achieve Ten TB level in volume. It has been proved useful in the performance of querying and storage of massive stream data in the experiments.**

Keywords—TIFA, Time Machine, FastBit, Query, Storage, Massive Stream Data

I. INTRODUCTION

Recently, the rapid development and expand of computer networks leads to the increasing of network interactive data scale. As a result, traditional database technology is stretched and powerless in the face of massive data storage and query. Researches have been carried out on theories and technologies in online data flow query and rapid archiving [7-16], and get valuable research results. However, these results are always limited to one area to solve specific issues, and the performance of these technologies meets new bottlenecks with the explosive growth of streaming data. On one hand, some methods improve the speed of storage at all costs, even at the expense of storage space. On the other hand, in order to meet the needs of data analysis, not only larger scale of achieving data should be supplied, but also query response time should be reduced. This paper aims to design flow records storage system, enabling continuous recording up to 8TB flow data (as the speed of 1MB per second, equivalent to 90 days continuous recording). Meantime, we improve the flow query system to get the result in large capacity storage in a shorter time. Our system runs on commercial UTM [19] device with 1Gbps port flow through the integration Time Machine and FastBit. Time Machine supplies the flow acquisition and record functions, while FastBit supplies flow record index and query functions.

As the system integrates Time Machine and FastBit, we name it *TIFA*. In this work, our main contributions are concluded as the following four points, i.e., (1) Propose a practical, scalable massive data storage system; (2) Time Machine and FastBit achieve effective integration and complementary; (3) Historical archived data can still be used effectively even after restarting the system; (4) The experiment results show the effectiveness of massive data storage query.

This paper is organized as follows: Section 2 and Section 3 introduce background and related work. Section 4 and Section 5 present the architecture and implementation of TIFA. Experiments and results are presented in Section 6. Finally, we give our conclusion in Section 7.

II. BACKGROUND

Nowadays, applications systems are generating large scale flow data in many fields, which is up to TB even PB. It prompts the emerging and development of massive flow data query and archiving technologies. The motivation of massive data storage and query is to analyze user behaviors and network communication status in massive network communication data. The massive data storage and query system in real time aims to realize online data storage and query in a long period and in large volumes.

TIFA emphases three characteristics, i.e., (1) Stability: Integrates Time Machine and FastBit effectively to update system interface and reliability; (2) Scalability: Realizes storage of multi-disks, the system default storage volume is 12TB, and the maximum is 24TB even 180TB (where multi-storage-servers and MB switches are needed); (3) Performance: TIFA can achieve TB level data storage. Users can query conveniently with SQL statements supported. It builds index for original records of 1,000,000 packets in 1 minutes.

III. RELATED WORK

Currently, there are some valuable research results in archiving and checking online high-speed stream of massive data. For querying continuous flow of online data, there is also a stream database management system (DSMS) [7]. A typical one is GigaScope, which is a distributed network monitoring framework. In this framework, some of the queries are added to the data source (routing etc.), and it can query multiple data stream from high-speed connections. PIER [8] and MIND [9] support queries of the historical monitoring data in a distributed environment, but their support for high-speed archiving and packet-level index is lacked. StreamBase [10] is a general database, which can support archiving and providing high-

speed data stream querying like GigaScope, but the database index is B-tree or the traditional hash. Literature [11] proposes Hyperion system, which is a solution based on a log file system derived from a LFS and can optimize archiving, indexing and online searching of the flow of multiple data. But the Hyperion does not take account of the scalability of flow continuous queries and storage.

Kwon et al. [12] have done more research on fault-tolerant, and have tried a number of scalable storage solutions. They propose a rollback recovery mechanism (Borealis [13]) which is implemented by asynchronous. The asynchronous checkpoint is stored in a distributed copy file system (implementing through HDFS [14]). However discussion of the scalability lacks detail. Applications about big file stream data processing of numerous writing and continuous reading are typically web search and Map-Reduce computing model [15]. It also helps the Google File System (GFS) [16] born. GFS has done a lot in high-throughput optimization, but not considered the reducing of latency.

In terms of real-time storage and online querying to network data stream, S. Kornexl and V. Paxson et al. designed Time Machine [1] to store and query high-speed network flow data in 2005. On the basis of data flow heavy-tailed characteristics, they used reasonable truncate data flow storage policy, which can significantly reduce the amount of data to be stored, while retain the necessary network information. In 2007, Frederick Reiss et al. [3] proposed a method that implemented fast real-time access to historical traffic data. Kesheng Wu et al. [2] proposed the system FastBit which used bitmap index system with SQL interface. It can reduce response time of large-scale data query.

IV. ARCHITECTURE

Time Machine can cache a few days' massive data flow. It stores the flow data after truncated. It also provides an efficient query interface to retrieve real-time data packets and manages available storage space automatically. It depends on the strict surveillance of characteristics associated with heavy-tailed network traffic, and can record the most completed connections.

Time Machine has good realization of traffic archiving, flow truncation, and the query operation in network trace file. There are still some improvements will make it more useful and popular: (1) It is better to support multiple hard devices as the surplus of SATA interfaces; (2) Packets stored in the raw files is returned by Time Machine when “to-file” query result is conducted, and it is better to give some quantitative analysis of the results; (3) It is simple enough to make its stability very well. However this would miss some considerations in practical use, such as it does not check hard disk space in runtime implementation; (4) It is better to support SQL queries statement because most of database users are already accustomed with SQL queries operation; (5) The query process of Time Machine is to read packet in the trace file one by one, and then compare them with the query for locating the matched packet. It is better to maintain a meta-data database to improve the efficiency of query.

Time Machine architecture and applications used by TIFA has been introduced in IMC 2005 [1] and SIGCOMM 2008 [4]. We have sought to avoid the above imperfections of Time Machine via integration of Time Machine and FastBit. FastBit uses bitmap index system with SQL interfaces. The key to its fast archiving of data is the vertical structure and the compressed bitmap directory. On one hand it is designed to be a column storage database. Each column is contiguously stored in a separated area of disk, which reduces the I/O overhead and improves I/O efficiency. Since the data packets are stored by column, and each packet is high correlation, which makes FastBit easier to achieve compression. On the other hand, FastBit implements a wealth of bitmap indexing algorithm [2], and is very efficient for massive read-only data query.

The commercial UTM we used is implemented based on Untangle [21]. We design the query using postgreSQL [22] database which is installed in Untangle. Grouping in the storage network, TIFA records relevant information of the groups using FastBit database, including the source address, destination address, source port, destination port, and protocol. Users should specify a time period when querying traffic streams, and TIFA analyzes the results returned using ntop [23] and provide a visual statistical report to users. TIFA architecture and work process are shown in Fig. 1 and Fig. 2.

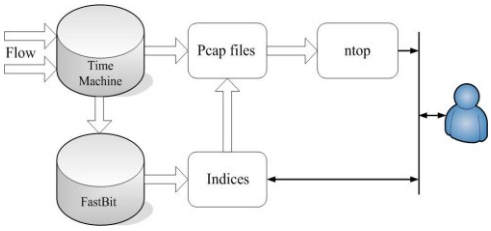

Figure 1. TIFA integrates the functions of TimeMachine: traffic collection and recording & FastBit: traffic records index and query.

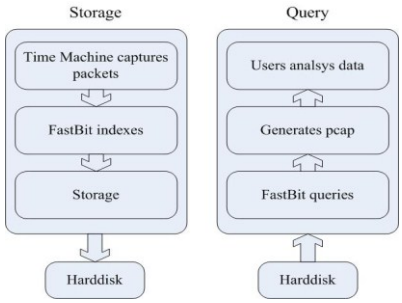

Figure 2. TIFA work process

V. IMPLEMENTATION

A. Storage

To make the UTM cost-efficient, the storage of a single UTM is expanded to avoid using special NAS or SAN in design. The UTM is equipped with 3 SATA disks. Time Machine stores the captured packets in the dump files. We limit the size of each dump file to 500 MB. TIFA creates a directory for each day and a subdirectory for each dump file. Different dump files are labeled with numbers in the same day. For example, the first dump file in a day is named 1, the second one is named 2, etc. TIFA creates a thread named movethread, which runs every 5 minutes. While running, the thread checks the dump files that haven’t been checked and move them to the

corresponding disk. As illustrated in Fig. 3, The disks are used in a circular way and mounted a Disk0, Disk1, Disk2,…, Disk n respectively. When a disk is full, its subsequent disk is used. Limited by the number of SATA interfaces, the current number of disks is 3 in TIFA. If more SATA interfaces can be supplied, the more disks can be mounted.

B. Index Creation

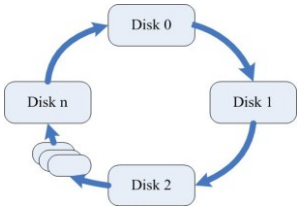

Figure 3. Disks are used in a circular way.

FastBit can build indices for data provided by columns, each column is a file. Table 1 indicates the attributes indexed for each packet.

TABLE I. ATTRIBUTES INDEXED FOR EACH PACKET

Sip	*Dip*	*Sport*	*Dport*	*Protocol*	*Offset*	*Time*

Sip, dip, sport and dport represent source IP address, destination IP address, source port and destination port respectively. Protocol is used to identify it is TCP or UDP. We limit the size of each dump file up to 500 MB. Offset indicates the offset of each packet in the file. Time expresses when a packet is captured. Usually, users search packets according to addresses and time, which is accurate in minute. Thus we make time of accuracy to one minute, enabling FastBit to build indices quickly.

Upon receiving a packet, it is feed to FastBit. A thread named buildindexthread is created to run periodically every 5 minutes. While running, it checks the pcap files that haven't been indexed and builds indices for them.

C. Queries

A postgreSQL database is installed in UTM, where we create a table named fileinfo, in which we record the information of each dump file. The fileinfo table is illustrated in table 2.

TABLE II. INFORMATION RECORDED FOR EACH DUMP FILE

column	usage
day	On which day the dump file is created
id	The number of files TIFA has created during the day
starttime	The number of seconds elapsed since the beginning of 1970 when the file is created
endtime	The number of seconds elapsed since the beginning of 1970 when the file is closed
index	Indicates whether the file has been indexed, 1 for yes, 0 for no.
directory	Indicates the disk where the file is placed

When users query the traffic between two hosts during a specific time range, TIFA queries fileinfo and finds which dump file may contain the traffic. For a selected dump file, TIFA queries FastBit and obtains the offset of corresponding packets in the file. TIFA uses libpcapnav [24] library to extract the packets according to the available offset, and dump them to a dump file called "result.pcap", which can be analyzed by ntop and produces a visual statistical report for users.

VI. EXPERIMENTS

In this section, we evaluate the performance of TIFA through deliberately designed experiments. We first test the key components of TIFA separately, which consist of packet capturing, index creation and query execution. Finally, we test the end-to-end throughput of TIFA based on realistic network traffic in the production network. All of our experiments are conducted on a UTM machine with one quad-core processor, 3G main memory, and 3 disks with SATA interfaces that support 33MB/sec reading and writing .In our current implementation, the capturing rate is a little slower than Time Machine, yet we have a great improvement in query response time compared to Time Machine.

A. Packet Capturing

Time Machine exploits the heavy-tailed characteristic of network traffic, and can truncate the long tail of a stream. If we enable Time Machine to cut off traffic, it will be difficult to measure the rate. As a result, we disable Time Machine ability to truncate traffic by setting its configuration file. Time Machine can also extract packets in a pcap trace file, which enables us to measure packets accurately. We start Time Machine with a trace file larger than 500 MB, and then we incrementally set the file size from 50 to 100 MB and measures the time that Time Machine takes to produce one dump file. The relation between average time and the size of dump file is illustrated in Fig. 4. We find that the average time is linear to the size of dump file with a ratio of 46MB/sec.

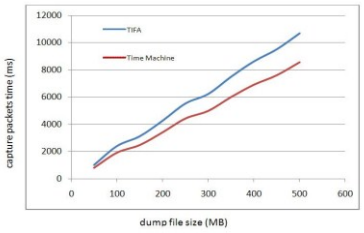

Figure 4.Throughput comparing between TIFA and Time Machine

B. Index Generation

FastBit can build index for batches of data, according to [3] the optimal batch size is about 1 million. The average length of Internet packet is about 512 bytes [19], thus a 500 MB dump file contains about 1 million packets. For each packet, we provide 7 attributes to FastBit and build indices for all attributes except offset. We find that the time to build index for one column is related to data diversity of the column. The higher data diversity is, the longer it takes to build index for the column, which is one of the reasons why we make the time attribute to an accuracy of one minute. We measure the time to build indices for each attribute for one dump file, and find that in many cases it's less than 5 seconds. An extreme case is when the elements in one column are different from each other. It takes 10 seconds to build index on our machine. The result indicates that FastBit can create indices efficiently.

C. Query

When a user submits a query based on time range, TIFA executes it in the following steps. First, it accesses fileinfo and finds out which dump file may contain the expected packets. Second, FastBit returns the offset of expected packets. Finally, corresponding packets are extracted to a dump file using libpcapnav library. We find that the first step can usually finish within one second. The second and third steps are time consuming. We also find that different queries have different time cost. For ease of expression, we divide queries into the following categories: (1) Accurate IP addresses and ranged time; (2) Ranged IP addresses and ranged time.

Since large time ranges can be subdivided into small time ranges which correspond to only one dump file, we mainly focus on testing queries related to one dump file with its size being 500 MB. For the first category, we find that the average response time is 0.8 seconds, while the average response time of Time Machine is 0.95 seconds. For the second category, we set IP address ranges being 10 to 100 and TIFA query average response time is illustrated in Fig. 5.

Furthermore, we evaluate the CPU usage of TIFA querying. In order to demonstrate the CPU usage change in various conditions, we set IP address ranges being 4,8,16,32,…,1024 and the result is shown in Fig. 6.

To summarize this section, we find TIFA gains 23% improvement in query response time with 20% decrease in throughput. In addition, TIFA can support IP addresses ranged query which is very import in post-attack analysis. Finally, we run TIFA continuously for 90 days in a real work network, which demonstrates that TIFA is a reliable system.

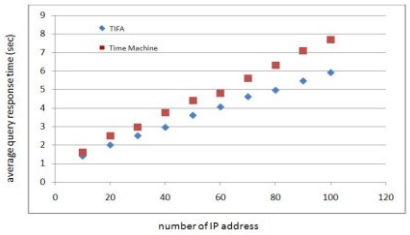

Figure 5. Average query response time within 10 to 100 IP

Figure 6. CPU usage in TIFA querying

VII. CONCLUSIONS

To the best of our knowledge, TIFA is the first system that integrates Time Machine and FastBit. TIFA exploits Time Machine ability to record high-volume traffic, and incorporates FastBit to gain an advantage in traffic query. We demonstrate the reliability and efficiency of TIFA through a series of deliberately designed experiments. To further improve TIFA, we think that new creative query technology is a tendency and parallel computation may also be of great importance.

REFERENCES

[1] Kornexl S, Paxson V, Dreger H, et al. “Building a TimeMachine for Efficient Recording and Retrieval of High-Volume Network Traffic,” [C]//Proc ACM IMC, October 2005

[2] Kesheng Wu, Sean Ahernm, et al. “FastBit: Interactively Searching Massive Data,” Journal of Physics, Vol 180, 2009

[3] Reiss F, Stockinger K, Wu Kesheng, et al. “nabling Real-time Querying of Live and Historical Stream Data,” E [C]//SSDBM 2007

[4] Gregor Maier, Robin Sommer, et al. “Enriching Network Security Analysis with Time Travel,” SIGCOMM 2008

[5] C. Shannon, D. Moore, K. Keys, M. Fomenkov, B. Huffaker, and kc claffy. “The Internet Measurement Data Catalog,” ACM Computer Communication Review, 35(5), Oct. 2005

[6] C.Fraleigh, S.Moon, B.Lyles,et al. “Packet-Level Traffic Measurements from the Sprint IP Backbone,” IEEE Network,2003,17(6):6-16

[7] Nicolas Hohn, Darryl Veitch, “Inverting sampled traffic. IEEE/ACM Traansactions on Networking,” 2006,14(1):68-80

[8] Shivanath Babu, “Adaptive Query Processing in Data Stream Management Systems,” Phd. Thesis, Stanford University, 2005.

[9] C. D. Cranor, T. Johnson, O, “Spatscheck, and V. Shkapenyuk.Gigascope: A Stream Database for Network Applications,” In SIGMOD, 2003.

[10] R. Hubesch et al. , A. R. The Architecture of PIER: an Internet-Scale Query Processor. In Proc. Conf. on Innovative Data Systems Research (CIDR) (Jan. 2005).

[11] LI, X., BIAN, F., ZHANG, H., DIOT, C., GOVINDAN, R., HONG, W., AND IANNACCONE, G, “Advanced Indexing Techniques forWide-Area Network Monitoring,” In Proc. 1st IEEE Intl. Workshop on Networking Meets Databases (NetDB) (2005).

[12] P. Desnoyers, P. Shenoy, “Hyperion: High Volume Stream Archival for Retrospective Querying,” in Proceedings of USENIX Annual Technical Conference, Santa Clara, CA, 2007.

[13] S. Kwon, M. Balazinka, A. Greenberg, “Fault-tolerant stream processing using a distributed, replicated filesystem,” in Proceedings of the VLDB Endowment (1): 574-585, 2008.

[14] U. Cetintemel et al., “The Aurora and Borealis Stream Processing Engines,” in Data Stream Management: Processing High-Speed Data Streams, , Springer-Verlag, July 2006.

[15] Apache Hadoop Project, “The Hadoop Distributed File System: Architecture and Design,” http://hadoop.apache.org, 2007.

[16] J. Dean, S. Ghemawat, “Map-Reduce: Simplified Data Processing on Large Clusters,” OSDI'04, San Francisco, CA, December, 2004

[17] S. Ghemawat, H. Gobioff, S. T. Leung, “The Google File System,” in Proceedings of the 19th ACM Symposium on Operating Systems. Principles, Lake George, NY, October, 2003

[18] C. Shannon, D. Moore, K. Keys, M. Fomenkov, B. Huffaker,and kc claffy, “The Internet Measurement Data Catalog,” ACM Computer Communication Review, 35(5), Oct. 2005

[19] M. F. Caetano, P. Vieira, J. L. Bordim, P. S. Barreto, “Throughput Bounds for HTTP Services Over WiFi Networks,” IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.8, pp. 17-18, August 2010

[20] UTM, at http://en.wikipedia.org/wiki/Unified_threat_management

[21] untangle, at http://www.untangle.com

[22] postgreSQL, at http://www.postgresql.org/

[23] ntop, at http://www.ntop.org/overview.html

[24] libpcapnav,at http://netdude.sourceforge.net/doco/libpcapnav/index.html