

# Cluster-based Online Monitoring System of Web Traffic\*

Yun Mao Kang Chen Dongsheng Wang Weimin Zheng  
Dept. of Computer Science and Technology, Tsinghua Univ.  
Beijing, 100084 P.R.China

{maoyun00,ck99}@mails.tsinghua.edu.cn {wds,zwm-dcs}@tsinghua.edu.cn

## ABSTRACT

Web traffic has been increasing and evolving rapidly in recent years. It is important to measure the volume and characteristic of such dominant traffic to understand large-scale user access pattern and analyze performance of Web applications. Among the common methods of Web measurements, the passive way using packet monitoring is more advantageous since it provides comprehensive information and is transparent to end-users. However, the throughput of current packet monitoring system is limited by the bandwidth of network adapters. Computational capacity and buffer size are also potential performance bottlenecks for monitoring high-speed links. This paper proposes a cluster-based online monitoring system, which generates rich logs by packet sniffing for Web analysis in a cluster environment. Powered by cluster computing technologies, the system achieves high performance and availability. Other techniques, such as online reconstruction in memory and kernel packet filter, are also adopted to improve the system performance. The experimental results indicate that it is feasible to trace in high-speed links with the cluster-based system.

## 1. INTRODUCTION

Web applications are evolving rapidly and become more dominant and popular in today's Internet. To improve the performance of Web servers and study large-scale user access pattern, it is important to characterize the Web servers and clients by utilizing all information initiated by Web including both HTTP level and TCP level events. Among the common methods of Web measurements, only the passive way using packet monitoring can satisfy this requirement and has more advantages than others do. However, sustained, rapid growth, increased e-commerce services, and proliferation of new applications have combined to change the characteristics of the Internet in recent years. The sheer volume of the traffic and the high capacity of the trunks have rendered the Web traffic measurement a more challenging endeavor. Existing passive measurement systems, such as BLT [5] and HTTPDump [17] are not well suited for monitoring on high-speed links. Their throughput is limited by the bandwidth of network adapters or low system performance. It is impossible to

\* This work is supported in part by National Science Foundation (grant No.F020303) and National Key Project of Basic Research G1999032702.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WIDM '01, Nov 9, 2001, Atlanta, Georgia, USA.

Copyright 2001 ACM 1-58113-000-0/00/0000...\$5.00.

trace continuously on a link of more than 100Mbps bandwidth. CPU, I/O bandwidth and buffer size are also potential performance bottlenecks.

This paper proposes a cluster-based online monitoring system. The system monitors Web traffic on a high-speed link by means of a cluster of workstations, and reconstruct HTTP and TCP events from sniffed raw packets. These events, such as HTTP request and response headers, source and destination IP addresses and port numbers, TCP SYN, FIN and RST packets and their timestamps, are beneficial to Web analysis. By distributing entire traffic to every node in the cluster environment, the system gains high throughput. Hardware and software redundancy in the cluster also provide our system high availability. Meanwhile, other techniques, such as kernel pre-filter and online reconstruction in memory, improve the performance of a single node.

The rest of this paper is organized as follows. Section 2 introduces the background of Web measurements and the advantages of packet monitoring in contrast to other measurement methods. In section 3, the design and implementation of cluster-based online monitoring system is described in detail. Experimental results are illustrated in section 4. Section 5 summarizes previous work. Finally, section 6 concludes this paper.

## 2. BACKGROUND INFORMATION

There are several ways to measure the Web access information of users' activities:

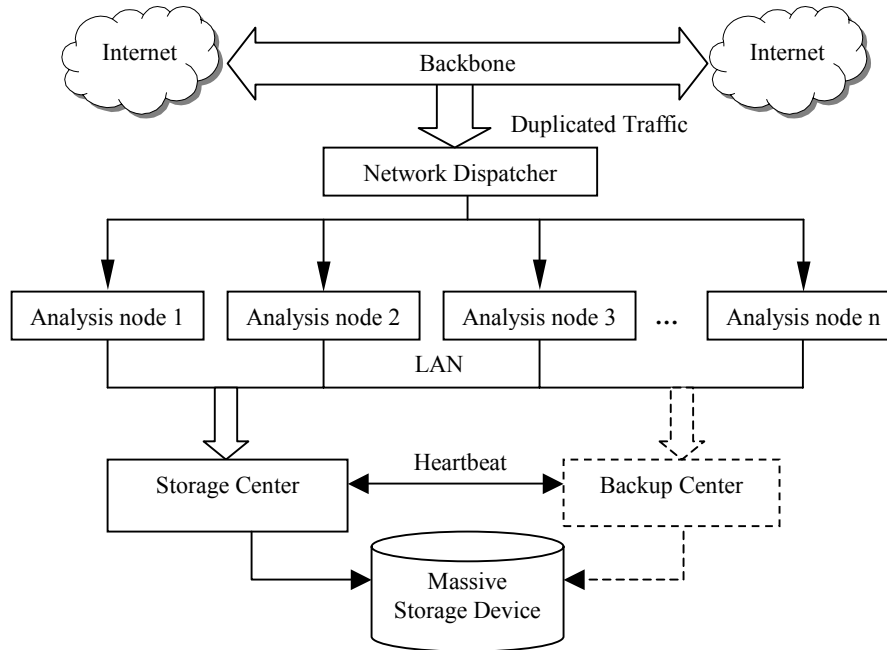
- (1) From the log-files of Web servers provided by ICPs (Internet Content Provider); [11, 1]
- (2) From users running modified Web browsers; [3]
- (3) From the log-files of Web proxies; [12, 4]
- (4) From the log-files generated by packet monitoring. [5, 17, 9, 2]

Each of these methods has its advantages and disadvantages:

The log-files from Web servers are very useful to evaluate the performance of the specific Web server. However, what ICP is offering will heavily influence the user's access patterns to a particular server. Therefore, many Web server logs have to be analyzed in order to generalize the entire Web. The work is not trivial since not all the log formats are identical. For example, IIS, Apache and Netscape Enterprise Server have different log formats.

It is also impractical to persuade many users to use a modified Web browser instead of their favorite Internet Explorer or Netscape Navigator and let them know the divulgence of their privacy about browsing behavior. The comparison between the research in Boston University by modifying Mosaic browser [3] and research by packet monitoring in [9] shows the limitation of lacking enough volunteers in this method.

Using the Web proxies for logging information can be a good method. Nevertheless, the availability of proxy server is crucial



**Figure 1. Architecture of parallel HTTP reconstruction**

and the proxy will become the bottleneck of the network (LAN or campus network) if the population is large enough. Moreover, the default level of detail collected in the logs is often inadequate to perform the necessary analysis. Modifying the source code to perform detailed logging can severely influence the performance, especially when the level of detail required is extensive.

There are several advantages in using information gathered from tracing HTTP level and TCP level information via packet monitoring. i) The strength of packet monitoring ensures that all useful information can be provided, such as full HTTP headers, TCP events and precise timestamps. ii) This passive monitoring methodology would never impact the performance of the Internet, and it is transparent to the end users. iii) The information is more general than the log of Web servers since data set of Web access will be very large if the packet monitoring system is installed on backbone of the Internet. These benefits make us to choose packet monitoring as our measurement method. The disadvantage is that the design and implementation of the system is complicated. HTTP reconstruction has almost to run a TCP and HTTP stack. Packet losses, retransmission and corruption are inevitable due to the extremely heavy traffic in the Internet and should be considered carefully. Moreover, we cannot make any assumptions about the compliance of either the clients or the servers with the TCP and HTTP specifications. Multiple HTTP requests can be pipelined on a TCP connection. Along with the bandwidth of the network increases, it is also difficult to develop a high performance monitoring system with high throughput.

### 3. DESIGN AND IMPLEMENTATION

#### 3.1 Architecture and Design

The system architecture is designed for high bandwidth network flows such as the Internet backbone. The network dispatcher dispatches the network data packets according to a specific algorithm, which will be discussed in Section 3.2. The analysis

nodes in cluster are interconnected with high-speed network. They reconstruct HTTP level and TCP level information and record them in the storage center with a massive storage device. Matching of HTTP requests with their responses is done offline. A backup center is prepared for handling unexpected fault events of main server. The overall architecture is illustrated as Figure 1.

The detail design of the analysis node is showed in Figure 2. When a node gets packets from the dispatcher, the network driver passes them to the kernel filter directly. After eliminating useless packets, the packets are sent to the buffer in the user space of the reconstruction process. Then the packets are forwarded one by one in the reconstruction hierarchy. Link-level frame layer simply removes the frame headers from the raw packets; IP extraction layer is responsible for IP fragmentation reconstruction; TCP extraction layer takes charge of making connectionless, out-of-order IP packets into connection-oriented TCP streams; HTTP extraction layer answers for HTTP request and response header extraction. This model is similar to the TCP/IP stack. After all the reconstruction and extraction works are finished, the process will store the information into local hard disk. Migration process runs periodically to move the information from local disk to the storage center.

#### 3.2 Dispatch Algorithm

The network dispatcher plays an important role in our system. All Internet traffic will pass through it, so it should have high performance and throughput. Common workstations are not suitable. We adopt a router as our dispatcher. Common routers and Layer-3 switches have gigabit bandwidth and are appropriate to distribute packets in time.

Dispatch algorithm must be selected with caution to satisfy two basic requirements: convenience of HTTP reconstruction and approximate load balance. The HTTP reconstruction requires that

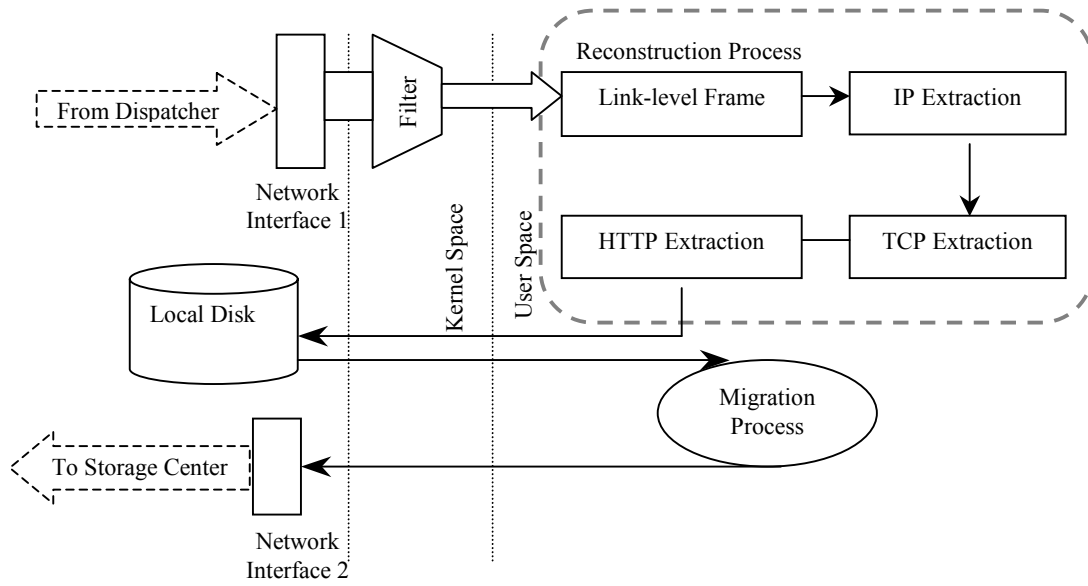


Figure 2. Design of HTTP reconstruction in analysis node

net-flows from one direction of a TCP connection is dispatched to the same node in the cluster. It is essential to the scalability of the cluster. If the dispatch algorithm did not satisfy this requirement, severe overhead would be paid for the complex communication in cluster and overall performance would drop down.

There are some algorithms that routers provide:

- (1) Round Robin. The dispatcher sends every packet with round-robin manner to every node.
- (2) Static Route. The router dispatches the packets only according to the destination IP address and the next-hop address in the routing table.
- (3) Dynamic Route. Packets are dispatched intelligently according to various network conditions.

Although Round Robin and Dynamic Route algorithm provide fairly good dataflow balance support, only Static Route algorithm is reasonable to meet the requirement of HTTP reconstruction.

We design a Semi-Static Route algorithm to ensure more load balance than Static Route algorithm. Routing table is not fixed in the algorithm. It varies every 12 hours since frequent table variation will also violate the requirement of reconstruction. The change policy is based on the statistic study of the network traffic distribution. First, assuming that  $n$  analysis nodes are in the cluster, we divide the whole network into  $m$  segments manually, as  $\{S_1, S_2, \dots, S_m\}$ ,  $m \geq n$ . For instance,  $S_1$  denotes the segment from 210.78.128.0 to 210.78.255.255;  $S_2$  denotes the segment from 202.204.68.0 to 202.204.68.255. Second, for each segment  $S_j$ , its traffic of the  $i$ -th 12-hour period is calculated as  $T_{j,i}$ . Third, at the beginning of  $k$ -th 12-hour period, we estimate the traffic  $E_{j,k}$  of every segment  $S_j$  as  $E_{j,k} = (T_{j,k-2} + T_{j,k-4} + T_{j,k-6})/3$ . This empirical equation is deduced from traffic volume similarity in consecutive days. [16] Finally, routing table is decided according to the estimation. Obviously, it is a NP-Hard problem to get the best solution. Here we present a greedy algorithm in polynomial

execution time. The algorithm of routing table generation is showed in Figure 3.

- 1)  $d_i = 0, i=1, 2, \dots, n$ .  $D = \{d_i\}$
- 2)  $G = \{E_{j,k}\}, j=1, 2, \dots, m$
- 3) Repeat
  - a) Let  $T_{c,k} = \max_{arg j} G$ .
  - b)  $G = G - \{T_{c,k}\}$
  - c) Select  $i$  that  $d_i = \min D$
  - d) Add a rule to routing table: segment  $S_c \rightarrow$  node  $i$
  - e)  $d_i = d_i + T_{c,k}$
- 4) Until  $G = \Phi$

Figure 3. The greedy algorithm of routing table generation

### 3.3 Availability Issue

High availability is important in many situations. Extremely busy access to a Web site may happen in only a few minutes and no one wishes his monitoring system to crash at that time. One of the advantages of a cluster-based system is its hardware and software redundancy. High availability can be provided by detecting node or daemon failures and reconfiguring the system appropriately so that the workload can be taken over by the remaining nodes in the cluster.

We deal with the analysis node failures as follows: There is a monitor daemon running on the storage center. This daemon is configured to detect whether the analysis nodes are alive. The service daemons running on all the nodes report their working conditions every  $t$  seconds. If there is an emergent exception reported or even no reports at all from a node, the node is considered to be down and routing table is changed immediately. All rules about that node are deleted and new rules are generated to distribute traffic to the left  $n-1$  nodes.

Now, the storage center becomes a single failure point of the whole system. In order to mask the failure of the primary storage center, we need to setup a backup server of the storage center. The

"fake" [10] technique is used for the backup to takeover the IP addresses of the storage center when primary center fails, and the "heartbeat" [15] technique is used to detect the status of the storage center to activate/deactivate the "fake" on the backup server. Two heartbeat daemons run on the primary and the backup. They heartbeat the message like "I'm alive" each other over network periodically. When the heartbeat daemon of the backup cannot hear the "I'm alive" message from the primary in the pre-defined time, it activates the fake to take over the Virtual IP address to provide the storage and monitoring service. When it receives the "I'm alive" message from the primary later, it deactivate the fake to release the Virtual IP address, and the primary comes back to work again.

### 3.4 Online Reconstruction in Memory

Most of online monitoring systems choose hard disks as their temporary buffers to keep the incoming packets. The more bandwidth they monitor, the more I/O operations they involve and the overall performance is degraded. Furthermore, all the browsing contents of users, including passwords, credit card numbers and private letters, are temporarily logged into disks, which lead to privacy and security problems. Monitoring in memory will not induce these problems. Although the memory resources are limited, we believe that memory online reconstruction is feasible for following reasons:

(1) Not all the Internet traffic is Web data. The useless data contain: (a) None TCP packets, such as UDP and ICMP, etc. Due to the popularity of streaming playback of music and news, as well as real-time voice telephony and conferencing, the number of UDP packets is increasing rapidly. (b) Traffic of familiar applications from well-known ports, such as port 23 for telnet, port 25 for SMTP/E-SMTP, port 110 for POP3, port 143 for IMAP, port 21 and 20 for ftp and ftp-data, etc. These applications occupy a considerable proportion in the entire Internet traffic. We adopt Linux Socket Filter, a Linux implementation of BPF-like kernel filter, to pre-filter these useless packets in the kernel space, which is efficient and able to avoid many unnecessary kernel-user space switches made by system calls [13]. We do not filter packets in the dispatcher because other information, such as multimedia traffic in UDP and generic TCP traffic may be useful in other research area and may be analyzed by other applications running on the cluster environment.

(2) Protocol headers, such as TCP headers, IP headers, and link-level frame headers are not necessary to save. Our experimental

results in section 4 illustrate the traffic composition in bytes. It shows that the Web traffic without overhead of TCP/IP protocol headers occupies about 65.0% of entire network traffic.

(3) It is arbitrary to filter out all information other than port 80 because not all the Web servers serve on port 80 and proxy servers have their own ports. However, Web traffic can be easily identified from their first 200 bytes according to the specification in RFC 2616 [7]. Therefore, TCP connections of other protocols can be identified as useless flows and removed from memory in their early stages of appearance.

By adopting memory online reconstruction, we remove the I/O bottleneck and speedup our system.

## 4. EXPERIMENTAL RESULTS

In this section we present preliminary results obtained from our use of our system. In section 4.1 we present results in a synthetic environment, while section 4.2 presents measurement results from a probe point in Beijing-TELECOM's commercial IP network. The results presented are meant to show the feasibility and performance of our system rather than conclusive results about the use of HTTP protocol in the Internet. Since common routers or switches are capable for dispatching gigabit traffic and there is no interaction between each node after dispatch, we mainly concern with single node performance of processing and the result of dispatch algorithm.

We choose 8 nodes cluster as our experimental environment. The configuration of each node is dual Pentium III 650E CPU, 512MB memory, 40GB SCSI hard disk and two 100Mbps Fast Ethernet adapters. The nodes in cluster are interconnected with a 100Mbps switch. The dispatcher is a layer-3 switch with a gigabit port for backbone connection and 100Mbps ports for cluster nodes. We select Linux with kernel version 2.2.16-smp as our operating system. All system code is implemented in C.

### 4.1 Tests in Synthetic Environment

In this case, all traffic that traverses the measured link is synthesized by means of Apache Bench Tool [18] to make concurrent requests. This test is necessary to obtain the max performance of a single node because in a real environment we only know how many requests and responses our monitoring system record, but how many Web pages users actually view is unknown. We can avoid this situation in synthetic environment. Results in synthetic environment are shown in table 1. When the

**Table 1. Single node testing results in synthetic environment**

Page Size (Bytes)	Concurrent Connections	Bandwidth (Mbps)	Requests Generated	Requests Caught	Accuracy
6519	100	33.1	2,000,000	2,000,000	100.0%
6519	600	43.9	6,000,000	6,000,000	100.0%
6519	2000	50.2	2,000,000	1,992,136	99.61%
6519	4000	55.8	4,000,000	3,613,871	90.34%
18156	100	67.2	2,000,000	2,000,000	100.0%
18156	600	86.5	6,000,000	6,000,000	100.0%
18156	2000	88.6	2,000,000	1,999,204	99.96%
18156	4000	91.1	4,000,000	3,824,178	95.60%

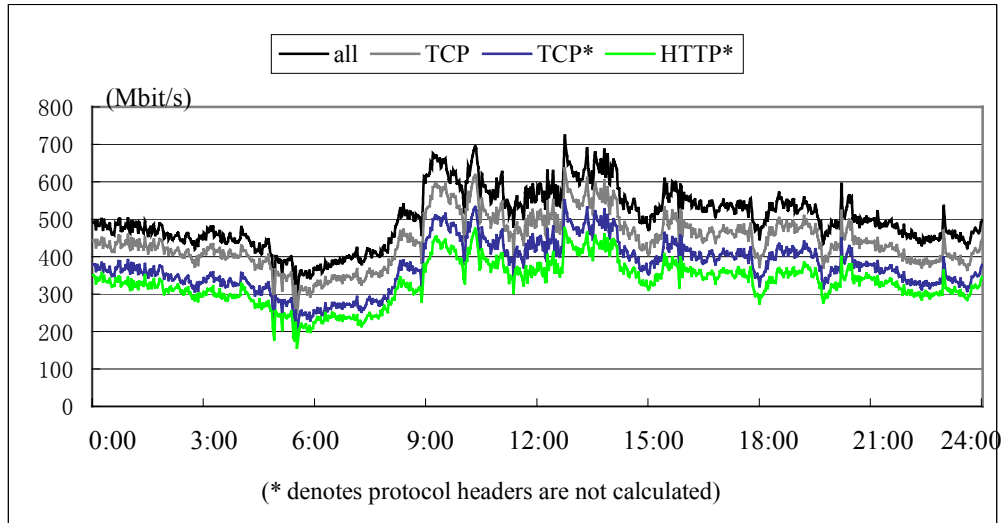


Figure 4. Traffic composition of common protocols in bit-rate over 24-hour period

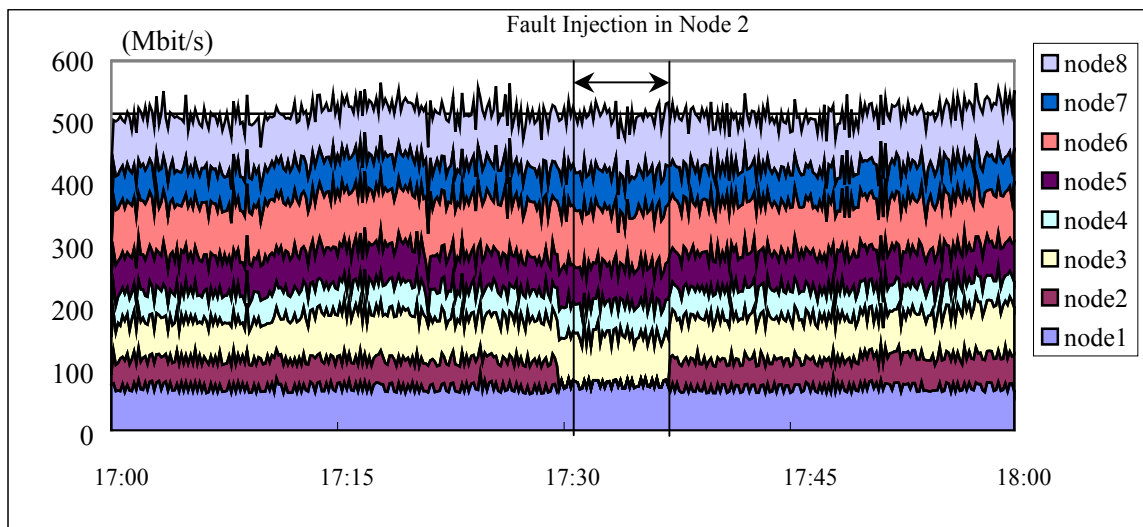


Figure 5. Traffic distribution in bit-rate during 1-hour period

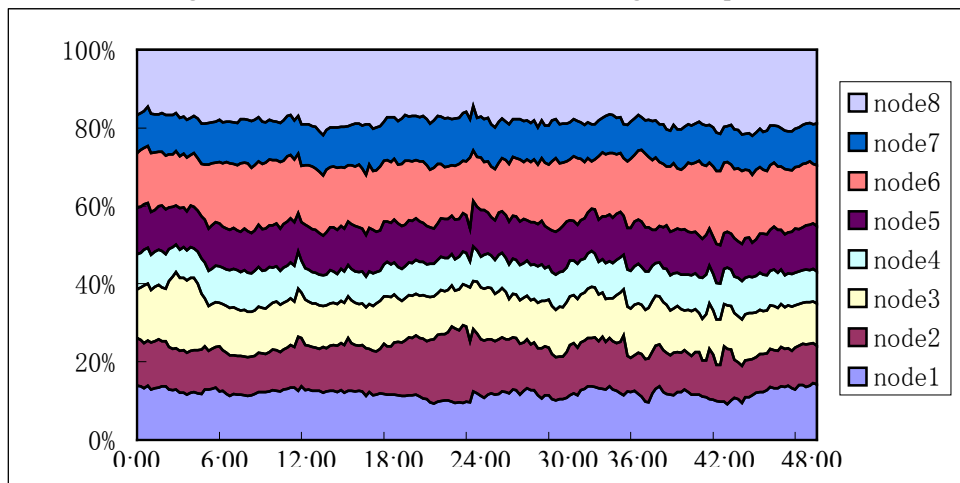


Figure 6. Traffic distribution in proportion during 2 consecutive days

number of concurrent HTTP connections is less than 2000, the system shows perfect accuracy. Along with bit-rate and number of concurrent connections increasing, packet loss happens and not all Web requests and response are caught. Since we do not save and parse HTTP contents and only care for headers, the ratio of header size to content size in entire traffic is also related to max performance. This is why performance increases when page size increases. Because all synthetic loads are Web traffic, we believe that our system can get better performance in real environment.

## 4.2 Tests on the Internet Backbone

We have been using our system to monitor Web traffic on BJ-TELECOM's Internet backbone of gigabit bandwidth. We divide the entire IP address domain into 50 segments according to their geographical locations and their traffic volume. While our system is running, the number and byte volume of packets that traverse each node are recorded. Figure 4 depicts bit-rate and traffic composition for this measurement point. The average ratio of Web traffic to entire traffic is 65.0%.

To test the effect of the data distribution, we record the bit-rate in every node. Their comparison in different time scales is showed in Figure 5 and 6. We conclude from them that although the characteristic of Internet traffic is bursty and self-similar, the ratio of traffic volume in each node to entire volume is relatively steady and is fairly smooth during a short period of time, which proves the feasibility of our traffic distribution strategy. We also tested the availability of our system. In Figure 5, we manually halted node 2 at 17:30, and turned it on at 17:35. When node 2 was off, the dispatcher redistributed the workload to the left 7 nodes. The throughput of system was not influenced by this fault injection.

## 5. PREVIOUS WORK

Some universities and research institutes have used packet level data to extract Web data. A group at Virginia Tech University has developed the HTTPDump [17] to extract HTTP headers from Tcpdump [19] traces. The performance of HTTPDump is not capable to sustain continuous tracing of 10Mbps Ethernet. AT&T Research Lab has the BLT (Bi-Layer Trace) [5] project, in which the challenges and advantages of packet monitoring are discussed in detail. The log file format is designed carefully and precisely to meet many analysis requirements. BLT has provided the foundation of several Web performance studies [6, 14]. Some researchers at IBM Research Institute have studied their Web server during Olympics with partial packet level data combined with server log files [2] and analyzed the TCP behavior in Web server. A group at Berkeley has developed a HTTP tracing module based on ISPE (Internet Protocol Scanning Engine) [8] to monitor the HomeIP network (A network center with 10Mbps bandwidth) continuously. Client's activities are traced and analyzed. [9]

## 6. CONCLUSION AND FUTURE WORK

After analyzing the characteristics and limitations of some related works, this paper provides an applicable and scalable solution, cluster-based online monitoring system. Sniffing on a high-speed link, our system provides abundant and useful information for Web analysis. In contrast to other related work, the throughput of our system is increased substantially with the help of cluster computing techniques instead of developing special purpose hardware. Meanwhile, heartbeat and Fake IP techniques ensure

the high availability of the cluster-based system. Other key techniques including kernel pre-filter, make the memory online tracing feasible so that some privacy problems are avoided and the disk I/O operations are reduced dramatically. Our architecture can also be applied to many other network research areas besides HTTP, such as real-time transport protocols, etc.

In terms of ongoing and future work, we have recently added more heuristics in network segmentation and traffic volume estimation. We are experimenting with using it to achieve better load balance. We also plan to encode IP addresses and cookies in log-files to achieve higher privacy level so that we can share the log-files to the public. Finally, we continue to use our system to monitor Web traffic on public Internet and plan to perform a thorough analysis of this traffic's growth and characteristics.

## 7. REFERENCES

- [1] Martin F. Arlitt and Carey L. Williamson, *Internet Web Servers: Workload Characterization and Performance Implications*, in IEEE/ACM Transactions on Networking, VOL.5, No.5, October 1997.
- [2] Balakrishnan, V.N. Padmanabhan, S. Seshan, M. Stemm, and R.H. Katz. *TCP behavior of a busy Internet server: Analysis and improvements*. In Proc. IEEE INFOCOM, April 1998.
- [3] Carlos R. Cunha, Azer Bestavros, and Mark E. Crovella. *Characteristics of www client-traces*. Technical Report BU-CS-95-010, Computer Science Department, Boston University, July 1995.
- [4] B.M. Duska, D. Marwood, and M.J. Feeley. *The measured access characteristics of World Wide Web client proxy caches*. In Proceedings of the Usenix Symposium on Internet Technologies and Systems, page 23-36, December 1997.
- [5] A. Feldmann, *BLT: Bi-Layer Tracing of HTTP and TCP/IP*, 9<sup>th</sup> International World Wide Web Conference, Amsterdam, May, 2000.
- [6] A. Feldmann, R. Caceres, F. Douglass, G. Glass, and M. Rabinovich. *Performance of Web proxy caching in heterogeneous bandwidth environments*. In Proc. IEEE INFOCOM, 1999.
- [7] R. Fielding, J. Gettys and J. Mogul, *RFC2616: Hypertext Transfer Protocol – HTTP/1.1*, Network Working Group, 1999.
- [8] Ian Goldberg. *The internet protocol scanning engine*. <http://www.cs.berkeley.edu/~iang/isaac/ipse.html>
- [9] S.D. Gribble and E.A. Brewer. *System design issues for Internet middleware services: Deductions from a large client trace*. In Proc. USENIX Symp. on Internet Technologies and Systems, December 1997.
- [10] Simon Horman, *Creating Redundant Linux Servers*, 4<sup>th</sup> Annual Linux Expo, May, 1998  
<http://www.au.vergenet.net/linux/fake/>
- [11] Arun K. Iyengar, Mark S. Squillante and Li Zhang, *Analysis and Characterization of Large-Scale Web Server Access Patterns and Performance*. World Wide Web, Volume 2, Number 1-2, Pages 85-100, 1999.

- [12] Tom M. Kroeger, Jeff Mogul, and Carlos Maltzahn. *Digital's Web proxy traces*.  
<ftp://ftp.digital.com/pub/DEC/traces/proxy/webtraces.html>,  
aug 1996.
- [13] S. McCanne and Van Jacobson, *The BSD Packet Filter: A New Architecture for User-level Packet Capture*. 1993  
Winter USENIX conference, January 25-29, 1993, San Diego,  
CA.
- [14] J.C. Mogul, F. Douglis, A. Feldmann, and B. Krishnamurthy.  
*Potential benefits of delta encoding and data compression  
for HTTP*. In Proc. ACM SIGCOMM, pages 181-194,  
September 1997.
- [15] Alan Robertson, *Linux-HA Heartbeat System Design*, 4<sup>th</sup>  
Annual Linux Showcase & Conference, Atlanta, Oct. 2000.
- [16] K. Thompson, G. J. Miller, and R. Wilder, *Wide-Area  
Internet Traffic Patterns and Characteristics*, IEEE Trans.  
on Networking, 1997.
- [17] R. Wooster, S. Williams and P. Brooks. *HTTPDUMP: a  
network HTTP packet snooper*. 1996.
- [18] Apache Software Foundation, *Apache Benchmark Tool*.  
<http://httpd.apache.org/docs/programs/ab.html>
- [19] LBNL's Network Research Group, *TCPDump - dump traffic  
on a network*. UNIX man page.