Prediction Based Cloud Bandwidth and Cost Reduction System using Light Weight Chunking

P.Vedhalakshmi[1] U.Hema[2] V.Kavitha[3]
[1,2]Student [3]Assistant Professor
[1,2,3]Department of Information Technology
[1,2,3]Karpaga Vinayaga College of Engineering and Technology, Kanchipuram Dt.Tamil Nadu, India

Abstract— we present PACK, a novel end-to-end traffic redundancy elimination (TRE) system, designed for cloud computing customers. It can be used to identify previously received chunk chains, which in turn can be used as reliable predictors to future transmitted chunks. Cloud-based TRE needs to apply a judicious use of cloud resources so that the bandwidth cost reduction combined with the additional cost of TRE computation and storage would be optimized. PACK's main advantage is its capability of offloading the cloud-server TRE effort to end clients, thus minimizing the processing costs induced by the TRE algorithm. Unlike previous solutions, PACK does not require the server to continuously maintain clients' status. This makes PACK very suitable for pervasive computation environments that combine client mobility and server migration to maintain cloud elasticity. PACK is based on a novel TRE technique, which allows the client to use newly received chunks to identify previously received chunk chains, which in turn can be used as reliable predictors to future transmitted chunks. We present a fully functional PACK implementation, transparent to all TCP-based applications and network devices. Finally, we analyze PACK benefits for cloud users, using traffic traces from various sources.

Key words: Pack, Traffic Redundancy Elimination, Chunking, Novel End to End

I. Introduction

Cloud computing offers its customers an economical and convenient pay-as-you-go service model, known also as usage-based pricing. Cloud customers pay only for the actual use of computing resources, storage, and bandwidth, according to their changing needs, utilizing the cloud's scalable and elastic computational capabilities. In particular, data transfer costs (i.e., bandwidth) is an important issue when trying to minimize costs. Consequently, cloud customers, applying a judicious use of the cloud's resources are motivated to use various traffic reduction techniques, in particular traffic redundancy elimination (TRE), for reducing bandwidth costs Traffic redundancy stems from common end-users' activities, such as repeatedly accessing, downloading, uploading (i.e., backup), distributing, and modifying the same or similar information items (documents, data, Web, and video). TRE is used to eliminate the transmission of redundant content and, therefore, to significantly reduce the network cost.

In most common TRE solutions, both the sender and the receiver examine and compare signatures of data chunks. When redundant chunks are detected, the sender replaces the transmission of each redundant chunk with its strong signature. Cloud providers cannot benefit from a technology whose goal is to reduce customer bandwidth bills, and thus are not likely to invest in one. Therefore, it is commonly agreed that a universal, software-based, end-to-end TRE is crucial in today's pervasive environment.

On the receiver side, we propose a new computationally lightweight chunking (fingerprinting) scheme termed PACK chunking. PACK chunking is a new alternative for Rabin fingerprinting traditionally used by RE applications. Experiments show that our approach can reach data processing speeds over 3 Gb/s, at least 20% faster than Rabin fingerprinting. Offloading the computational effort from the cloud to a large group of clients forms a load distribution action, as each client processes only its TRE part. The receiver-based TRE solution addresses mobility problems common to quasi-mobile desktop/laptops computational environments. One of them is cloud elasticity due to which the servers are dynamically relocated around the federated cloud, thus causing clients to interact with multiple changing servers.

Another property is IP dynamics, which compel roaming users to frequently change IP addresses. In addition to the receiver-based operation, we also suggest a hybrid approach, which allows a battery-powered mobile device to shift the TRE computation overhead back to the cloud by triggering a sender-based end-to-end TRE. To validate the receiver-based TRE concept, we implemented, tested, and performed realistic experiments with PACK within a cloud environment. Our experiments demonstrate a cloud cost reduction achieved at a reasonable client effort while gaining additional bandwidth savings at the client side. The implementation code, over 25 000 lines of C and Java. Our implementation utilizes the TCP Options field, supporting all TCP-based applications such as Web, video streaming, P2P, e-mail, etc.

II. Existing System

Traffic redundancy arises from common end-users' activities, such as repeatedly accessing, downloading, uploading, distributing, and modifying the same or similar information items. TRE is used to eliminate the transmission of redundant content and, therefore, to significantly reduce the network cost. Cloud providers cannot benefit from a technology whose goal is to reduce customer bandwidth bills.

A. *Limitations of Existing System:*

The problems present in this system are,

1) Cloud providers cannot benefit from a technology whose goal is to reduce customer bandwidth bills, and thus are not likely to invest in one.
2) The rise of "on-demand" work spaces, meeting rooms, and work-from-home solutions detaches the workers from their offices. In such a dynamic work environment, fixed-point solutions that require a

client-side and a server-side middle-box pair become ineffective.

3) Cloud load balancing and power optimizations may lead to a server-side process and data migration environment, in which TRE solutions that require full synchronization between the server and the client are hard to accomplish or may lose efficiency due to lost synchronization.
4) Current end-to-end solutions also suffer from the requirement to maintain end-to-end synchronization that may result in degraded TRE efficiency.

III. PROPOSED SYSTEM

we present a novel receiver-based end-to-end TRE solution that relies on the power of predictions to eliminate redundant traffic between the cloud and its end-users. In this solution, each receiver observes the incoming stream and tries to match its chunks with a previously received chunk chain or a chunk chain of a local file. Using the long-term chunks' metadata information kept locally, the receiver sends to the server predictions that include chunks' signatures and easy-to-verify hints of the sender's future data. On the receiver side, we propose a new computationally lightweight chunking (fingerprinting) scheme termed PACK chunking. PACK chunking is a new alternative for Rabin fingerprinting traditionally used by RE applications.

A. *Advantages of Proposed System:*

1) Our approach can reach data processing speeds over3 Gb/s, at least 20% faster than Rabin fingerprinting.
2) The receiver-based TRE solution addresses mobility problems common to quasi-mobile desktop/ laptops computational environments.
3) One of them is cloud elasticity due to which the servers are dynamically relocated around the federated cloud, thus causing clients to interact with multiple changing servers.
4) We implemented, tested, and performed realistic experiments with PACK within a cloud environment. Our experiments demonstrate a cloud cost reduction achieved at a reasonable client effort while gaining additional bandwidth savings at the client side.
5) Our implementation utilizes the TCP Options field, supporting all TCP-based applications such as Web, video streaming, P2P, e-mail, etc.
6) We demonstrate that our solution achieves 30% redundancy elimination without significantly affecting the computational effort of the sender, resulting in a 20% reduction of the overall cost to the cloud customer.

IV. SYSTEM MODEL

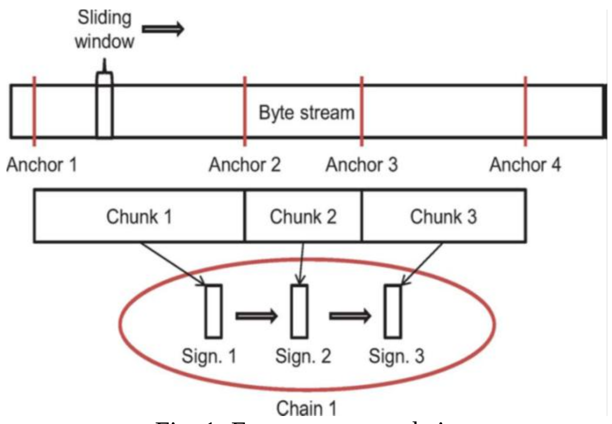

Fig. 1: From stream to chain

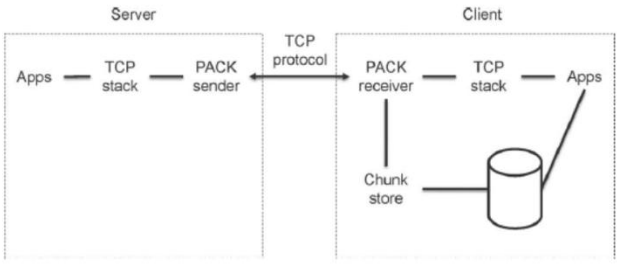

Fig. 2: Overview of the PACK implementation

V. MODULES

– Server Operational Cost
– Chunking Scheme
– Adaptive Receiver Virtual Window
– Cloud server as a Receiver
– Receiver Based TRE Solution
– Pack Messages Format

A. *Modules Description:*

1) *Server Operational Cost:*

We measured the server performance and cost as a function of the data redundancy level in order to capture the effect of the TRE mechanisms in real environment. To isolate the TRE operational cost, we measured the server's traffic volume and CPU utilization at maximal throughput without operating a TRE. We then used these numbers as a reference cost, based on present Amazon EC2 pricing. The server operational cost is composed of both the network traffic volume and the CPU utilization, as derived from the EC2 pricing.

	No TRE	PACK	Server-based
Traffic volume	9.1 TB	6.4 TB	6.2 TB
Traffic cost reduction (Figure 4)		30%	32%
Server-hours cost increase (Figure 9)		6.1%	19.0%
Total operational cost	100%	80.6%	83.0%

Table 1: Cloud Operational Cost Comparison

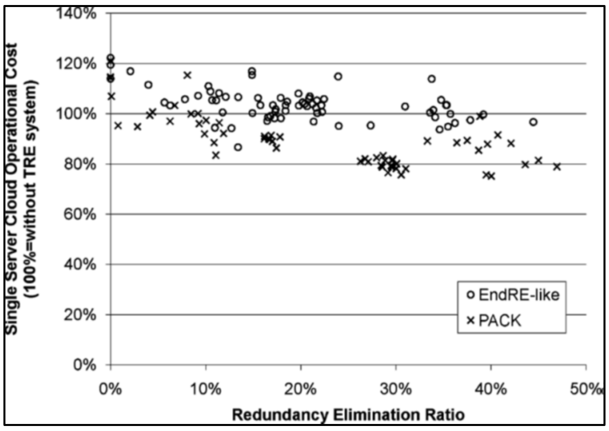

Fig. 3: Cloud Operational Graph

2) Chunking Scheme:

Our implementation employs a novel computationally lightweight chunking (fingerprinting) scheme, termed *PACK chunking*. The scheme, presented in Proc. 8 and illustrated in Fig. 13, is an XOR-based rolling hash function, tailored for fast TRE chunking. Anchors are detected by the mask in line 1 that provides on average 8-kB chunks. The mask, was chosen to consider all the 48 B in the sliding window.

```
1) mask<=0x00008A3110583080{48 bytes window; 8 KB chunks}
2) longval<={has to be 64 bits}
3) for all byte stream do
4) shift left longval by 1 bit {lsb←0 ; drop msb}
5) longval<=longval bitwise-xor byte
6) if processed at least 48 bytes and (longval bitwise-and mask) then
7) found an anchor
8) end if
9) end for
```

Above Table summarizes the processing speed of the different chunking schemes. As a baseline figure, we measured the speed of SHA-1 signing and found that it reached 946 Mb/s

Scheme	Window	Chunks	Speed
SampleByte 8 mark-ers	1 byte	32 bytes	1,913 Mbps
Rabin fingerprint	48 bytes	8 KB	2,686 Mbps
PACK chunking	48 bytes	8 KB	3,259 Mbps
SampleByte 1 marker	1 byte	256 bytes	5,176 Mbps

Table 2: Chunking Schemes Processing Speed Tested With 10-Mb Random File over A Client's Laptop, Without Either Minimal or Maximal Limit on the Chunk Size random binary file.

3) Adaptive Receiver Virtual Window:

PACK enables the receiver to locally obtain the sender's data when a local copy is available, thus eliminating the need to send this data through the network. We term the receiver's fetching of such local data as the reception of *virtual data*. When the sender transmits a high volume of virtual data, the connection rate may be, to a certain extent, limited by the number of predictions sent by the receiver.

4) Cloud server as a Receiver:

In a growing trend, cloud storage is becoming a dominant player—from back up and sharing services to the American National Library, and e-mail services. In many of these services, the cloud is often the receiver of the data. If the sending client has no power limitations, PACK can work to save bandwidth on the upstream to the cloud. In these cases, the end-user acts as a sender, and the cloud server is the receiver. The PACK algorithm need not change. It does require, however, that the cloud server—like any PACK receiver—maintain a *chunk store*.

5) Receiver Based TRE Solution:

To further understand how TRE would work for a cloud-based Web service with returning end-users, we obtained a traffic log from a social network site for a period of 33 days at the end of 2010. The data log enables a reliable long-term detection of returning users, as users identify themselves using a login to enter the site. We identified the sessions of 7000 registered users over this period. We then measured the amount of TRE that can be obtained with different cache sizes at the receiver (a synchronized sender-based TRE keeps a mirror of the last period cache size).

6) Pack Messages Format:

In our implementation, we use two currently unused TCP option codes, similar to the ones defined in SACK [32]. The first one is an enabling option *PACK permitted* sent in a SYN segment to indicate that the PACK option can be used after the connection is established. The other one is a *PACK message* that may be sent over an established connection once permission has been granted by both parties. A single PACK message, piggybacked on a single TCP packet, is designed to wrap and carry multiple PACK commands. This not only saves message overhead, but also copes with security network devices (e.g., firewall) that tend to change TCP options order. Note that most TCP options are only used at the TCP initialization period, with several exceptions such as SACK and timestamps. Due to the lack of space, additional implementation details are left out and are available.

VI. CONCLUSION

Cloud computing is expected to trigger high demand for TRE solutions as the amount of data exchanged between the cloud and its users is expected to dramatically increase. The cloud environment redefines the TRE system requirements, making proprietary middle-box solutions inadequate. Consequently, there is a rising need for a TRE solution that reduces the cloud's operational cost while accounting for application latencies, user mobility, and cloud elasticity. In this paper, we have presented PACK, a receiver-based, cloud-friendly, end-to-end TRE that is based on novel speculative principles that reduce latency and cloud operational cost. PACK does not require the server to continuously maintain clients' status, thus enabling cloud elasticity and user mobility while preserving long-term redundancy. Moreover, PACK is capable of eliminating redundancy based on content arriving to the client from multiple servers without applying a three-way handshake.

Our evaluation using a wide collection of content types shows that PACK meets the expected design goals and has clear advantages over sender-based TRE, especially when the cloud computation cost and buffering requirements are important. Moreover, PACK imposes additional effort on the sender only when redundancy is exploited, thus reducing the cloud overall cost.

REFERENCES

[1] E. Zohar, I. Cidon, and O. Mokryn, "The power of prediction: Cloud bandwidth and cost reduction," in Proc. SIGCOMM, 2011, pp. 86–97.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph,R.Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," Commun. ACM, vol. 53, no. 4, pp. 50–58, 2010

[3] U. Manber, "Finding similar files in a large file system," in Proc. USENIX Winter Tech. Conf., 1994, pp. 1–10.

[4] N. T. Spring and D. Wetherall, "A protocol-independent technique for eliminating redundant network traffic," in Proc. SIGCOMM, 2000, vol. 30, pp. 87–95.

[5] A. Muthitacharoen, B. Chen, and D. Mazières, "A low-bandwidth network file system," in Proc. SOSP, 2001, pp. 174–187.

[6] E. Lev-Ran, I. Cidon, and I. Z. Ben-Shaul, "Method and apparatus for reducing network traffic over low bandwidth links," US Patent 7636767, Nov. 2009.

[7] S.Mccanne and M. Demmer, "Content-based segmentation scheme for data compression in storage and transmission including hierarchical segment representation," US Patent 6828925, Dec. 2004.

[8] R. Williams, "Method for partitioning a block of data into sub blocks and for storing and communicating such sub blocks," US Patent 5990810, Nov. 1999.

[9] Juniper Networks, Sunnyvale, CA, USA, "Application acceleration," 1996 [Online]. Available: http://www.juniper.net/us/en/products-services/application-acceleration/

[10] Blue Coat Systems, Sunnyvale, CA, USA, "MACH5," 1996 [Online]. Available: http://www.bluecoat.com/products/mach5

[11] Expand Networks, Riverbed Technology, San Francisco, CA, USA, "Application acceleration and WAN optimization," 1998 [Online]. Available: http://www.expand.com/technology/application-acceleration.aspx

[12] F5, Seattle, WA, USA, "WAN optimization," 1996 [Online]. Available: http://www.f5.com/solutions/acceleration/wan-optimization/

[13] A. Flint, "The next workplace revolution," Nov. 2012 [Online]. Available: http://m.theatlanticcities.com/jobs-and-economy/2012/11/
Next work place-revolution/3904/

[14] A. Anand, C. Muthukrishnan, A. Akella, and R. Ramjee, "Redundancy in network traffic: Findings and implications," in Proc. SIGMETRICS, 2009, pp. 37–48.