International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622
NATIONAL CONFERENCE on Developments, Advances & Trends in Engineering Sciences
(NCDATES- 09th & 10th January 2015)

RESEARCH ARTICLE OPEN ACCESS

PACK: Prediction-Based Cloud Bandwidth and Cost Reduction System

A.padmaja, Mr.Rajshekar
M.Tech Scholar, CMR Engineering College
Asst.Professor, Dept. of CSE, CMR Engineering College,Hydearbad,
E-Mail: ruthikareddy23@gmail.com

Abstract:
In this paper, we present PACK (Predictive ACKs),a novel end-to-end traffic redundancy elimination (TRE) system,designed for cloud computing customers. 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 endclients.PACK does not require the server to continuously maintain clients' status. 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.
Index Terms—Caching, cloud computing, network optimization, traffic redundancy elimination.

Introduction

CLOUD computing offers its customers an economical and convenient *pay-as-you-go* service model, known also as *usage-based pricing* [2]. Cloud customers1 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 [2]. 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, parsed according to the data content, prior to their transmission.When redundant chunks are detected, the sender replaces the transmission of each redundant chunk with its strong signature [3]–[5]. Commercial TRE solutions are popular at enterprise networks, and involve the deployment of two or more proprietary-protocol, state synchronized middle-boxes at both the intranet entry points of data centers and branch offices, eliminating repetitive traffic between them(e.g., Cisco [6], Riverbed [7], Quantum [8], Juniper [9], BlueCoat [10], Expand Networks [11], and F5 [12]).

While proprietary middle-boxes are popular point solutions within enterprises, they are not as attractive in a cloud environment.Cloud providers cannot benefit from a technology whose goal is to reduce customer bandwidth bills, and thus are not likely to invest in one. The rise of "on-demand" work spaces, meeting rooms, and work-from-home solutions [13] detaches the workers from their offices. In such dynamic work environment,fixed-point solutions that require a client-side and a server-side middle-box pair become ineffective. On the other hand, cloud-side elasticity motivates work distribution among servers and migration among data centers. Therefore, it is commonly agreed that a universal, software-based, end-to-end TRE is crucial in today's pervasive environment [14], [15]. This enables the use of a standard protocol stack and makes a TRE within end-to-end secured traffic (e.g., SSL) possible. Current end-to-end TRE solutions are sender-based. In the case where the cloud server is the sender, these solutions require that the server continuously maintain clients' status. We show here that *cloud elasticity* calls for a new TRE solution.First, 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. Second, the popularity of rich media that consume high bandwidth motivates content distribution network

(CDN) solutions, in which the service point for fixed and mobile users may change dynamically accordingto the relative service point locations and loads.

EXISTING SYSTEM

Traffic redumdamcy systems from common end users activities,such as repeatedly accessing,downloading ,uploading,distributing and modifying same or similar information items(docu--ments,data,web and video). TRE is used to eliminate the transmission of redundant conten t and, therefore To significant reduce the network cost. In most com--mon TRE solutions, both the sender and the receiver Examine and compare signatures of data chunks, parsed according to the data content,prior to their tra--nsmission.when redundant chunks are detected,the Sender replaces the transmission of each redundant Chunk with its strong signature.Commercial TRE Solutions are poipular enterprise networks,and involve the development of two or more proprietary-Protocol,state synchronise middle-boxes at both the intranet entry points of data centers.

DISADVANTAGES OF EXISTING SYSTEMS:

- Cloud providers cannot benefit from a technology whose gole is to reduce customer Bandwidth bills,and thus are not likely to invest is one.
- The rise of "on-demand" work spaces, meeting rooms,and work-from-home solutions detatches the workers from their offices.In such a dynamic work environment,fixed-point solutions that require client-side and a server- side middle-box pair become in effective.
- 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 loss efficiency due to lost synchornisation.
- Current end-to-end solutions also suffer from the requirement to maintain end-to-end synchornisation that may result in degraded TRE effeiciency.

PROPOSED SYSTEM

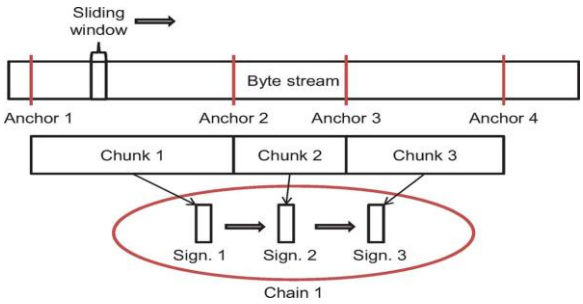
In this paper,we present a novel receiver-based end-to-end TRE solution that relies on the power of prediction to eliminate redundant traffic between the cloud and its end-users.In this solution , each receiver observes the incoming stream and tries to match its chunks with a previously received chunk 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 users data.

ADVANTAGES OF PROPOSED SYSTEMS

- Our approach can reach the data processing speed over3 Gbs,atleast 20% faster than rabin fingerprinting
- The receiver-based TRE solution addresses mobility problems common to quasi-mobile desktop.
- One of then is cloudy elasticity due to which servers dynamically relocated around the federate cloud,thus causing clients to interact with multiple changing servers.
- 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 band width savings at the client side.

SYSTEM ARCHITECTURE



PACK ALGORITHM

The stream of data received at the PACK receiver is parsed to a sequence of variable-size, content-based signed chunks similar to [3] and [5]. The chunks are then compared to the receiver local storage, termed *chunk store*. If a matching chunk is found in the local chunk store, the receiver retrieves the sequence of subsequent chunks, referred to as a *chain*, by traversing the sequence of LRU chunk pointers that are included in the chunks' metadata.

A. Receiver Chunk Store

PACK uses a new *chains* scheme, described in Fig. 1, in which chunks are linked to other chunks according to their last received order. The PACK receiver maintains a *chunk store*, which is a large size cache of chunks and their associated metadata. Chunk's metadata includes the chunk's signature and a (single) pointer to the successive chunk in the last received stream containing this chunk. Caching and indexing techniques are employed to efficiently

maintain and retrieve the stored chunks, their signatures, and the chains formed by traversing
the chunk pointers When the new data are received and parsed to chunks, the receiver computes each chunk's signature using SHA-1. At this point, the chunk and its signature are added to the chunk store.
In addition, the metadata of the previously received chunk in the same stream is updated to point to the current chunk. The unsynchronized nature of PACK allows the receiver to map each existing file in the local file system to a chain of chunks, saving in the chunk store only the metadata associated
with the chunks.3 Using the latter observation, the receiver can also share chunks with peer clients within the same local network utilizing a simple map of network drives. The utilization of a small chunk size presents better redundancy elimination when data modifications are fine-grained, such as sporadic changes in an HTML page. On the other hand, the use of smaller chunks increases the storage index size, memory usage, and magnetic disk seeks. It also increases the transmission overhead of the virtual data exchanged between the client and the server. Unlike IP-level TRE solutions that are limited by the IP packet size (B), PACK operates on TCP streams and can therefore handle large chunks and entire chains. Although our design permits each PACK client to use any chunk size, we recommend an average chunk size of 8 kB.

B. Receiver Algorithm

Upon the arrival of new data, the receiver computes the respective signature for each chunk and looks for a match in its local chunk store. If the chunk's signature is found, the receiver determines whether it is a part of a formerly received chain, using the chunks' metadata. If affirmative, the receiver sends a prediction to the sender for several next expected chain chunks. The prediction carries a starting point in the byte stream (i.e., offset) and the identity of several subsequent chunks (PRED command).

Upon a successful prediction, the sender responds with a PRED-ACK confirmation message. Once the PRED-ACK message is received and processed, the receiver copies the corresponding data from the chunk store to its TCP input buffers, placing it according to the corresponding sequence numbers.
At this point, the receiver sends a normal TCP ACK with the next expected TCP sequence number. In case the prediction is false, or one or more predicted chunks are already sent, the sender continues with normal operation, e.g., sending the raw
data, without sending a PRED-ACK message.

Proc. 1: Receiver Segment Processing
1. **if** segment carries payload *data* **then**
2. calculate chunk
3. **if** reached chunk boundary **then**
4. activate predAttempt()
5. **end if**
6. **else if** PRED-ACK segment **then**
7. processPredAck()
8. activate predAttempt()
9. **end if**

Proc. 2: predAttempt()
1. **if** received *chunk* matches one in chunk store **then**
2. **if** foundChain(*chunk*) **then**
3. prepare PREDs
4. send single TCP ACK with PREDs according to Options free space
5. exit
6. **end if**
7. **else**
8. store *chunk*
9. link *chunk* to current chain
10. **end if**
11. send TCP ACK only

Proc. 3: processPredAck()
1. **for all** offset PRED-ACK **do**
2. read data from chunk store
3. put data in TCP input buffer
4. **end for**

C. Sender Algorithm

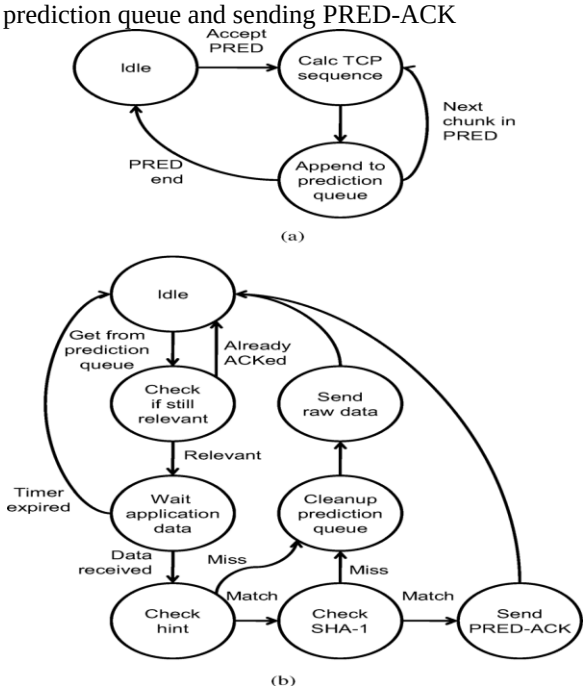
When a sender receives a PRED message from the receiver, it tries to match the received predictions to its buffered (yet to be sent) data. For each prediction, the sender determines the corresponding TCP sequence range and verifies the hint. Upon a hint match, the sender calculates the more computationally intensive SHA-1 signature for the predicted data range and compares the result to the signature received in the PRED message. Note that in case the hint does notmatch, a computationally expansive operation is saved. If the two SHA-1 signatures match, the sender can safely assume that the receiver's prediction is correct. In this case, it replaces the corresponding outgoing buffered data with a PRED-ACK message.

D. Wire Protocol

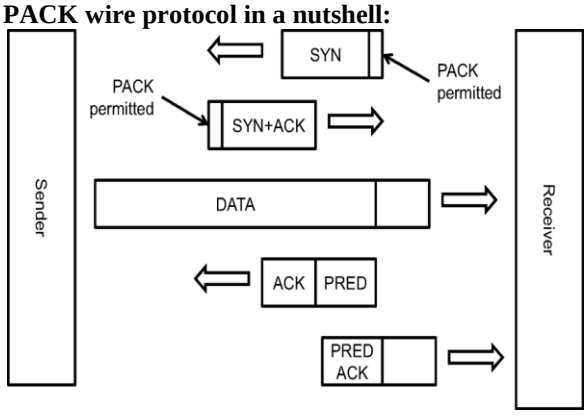
In order to conform with existing firewalls and minimize overheads, we use the TCP Options field to carry the PACK wire protocol. It is clear that PACK can also be implemented above the TCP level while using similar message types and control fields. Fig. 3 illustrates the way the PACK wire protocol operates under the assumption that the data is redundant. First, both sides enable the PACK option

during the initial TCP handshake by adding a *PACK permitted* flag (denoted by a bold line) to the TCP Options field. Then, the sender sends the (redundant) data in one or more TCP segments, and the receiver identifies that a currently received chunk is identical to a chunk in its chunk store. The receiver, in turn, triggers a TCP ACK message andincludes the prediction in the packet's Options field. Last, the sender sends a confirmation message (PRED-ACK) replacing the actual data.

Sender algorithms. (a) Filling the prediction queue. (b) Processing the
prediction queue and sending PRED-ACK



PACK wire protocol in a nutshell:



OPTIMIZATIONS

For the sake of clarity, Section III presents the most basic version of the PACK protocol. In this section, we describe additional options and optimizations.

A. 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. This, in turn, means that the receiver predictions and the sender confirmations should be expedited in order to reach high virtual data rate. For example, in case of a repetitive success in predictions, the receiver's side algorithm may become optimistic and gradually increase the ranges of its predictions, similarly to the TCP rate adjust

Proc. 4: predAttemptAdaptive()—obsoletes Proc. 2
1. {new code for Adaptive}
2. **if** received *chunk* overlaps recently sent prediction **then**
3. **if** received *chunk* matches the prediction **then**
4. *predSizeExponent()*
5. **else**
6. *predSizeReset()*
7. **end if**
8. **end if**
9. **if** received *chunk* matches one in signature cache **then**
10. **if** foundChain(*chunk*) **then**
11. {new code for Adaptive}
12. prepare PREDs according to *predSize*
13. send TCP ACKs with all PREDs
14. exit
15. **end if**
16. **else**
17. store *chunk*
18. append *chunk* to current chain
19. **end if**
20. send TCP ACK onlyment procedures.

Proc. 5: processPredAckAdaptive()—obsoletes Proc. 3
1. **for all** offset PRED-ACK **do**
2. read data from disk
3. put data in TCP input buffer
4. **end for**
5. {new code for Adaptive}
6. *predSizeExponent()*

B. Cloud Server as a Receiver

In a growing trend, cloud storage is becoming a dominant player [23], [24]—from backup and sharing services [25] to the American National Library [26], and e-mail services [27], [28]. 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*.

C.Hybrid Approach

PACK's receiver-basedmode is less efficient if changes in the data are scattered. In this case, the prediction sequences are frequently interrupted, which, in turn, forces the sender to revert to raw data transmission until a new match is found at the receiver and reported back to the sender. To that end, we present the PACK hybrid mode of operation, described in Proc. 6 and Proc. 7.When PACK recognizes a pattern of dispersed changes,
it may select to trigger a sender-driven approach in the spirit of [4], [6], [7], and [18].

Proc. 6: Receiver Segment Processing Hybrid—obsoletes
Proc. 1
1. **if** segment carries payload *data* **then**
2. calculate chunk
3. **if** reached chunk boundary **then**
4. activate predAttempt()
5. {new code for Hybrid}
6. **if** detected broken chain **then**
7. calcDispersion(255)
8. **else**
9. calcDispersion(0)
10. **end if**
11. **end if**
12. **else if** PRED-ACK segment **then**
13. processPredAck()
14. activate predAttempt()
15. **end if**

Proc. 7: processPredAckHybrid()—obsoletes Proc. 3
1. **for all** offset PRED-ACK **do**
2. read data from disk
3. put data in TCP input buffer
4. {new code for Hybrid}
5. **for all** chunk offset **do**
6. calcDispersion(0)
7. **end for**
8. **end for**

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.

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 andM. 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 subblocks and for storing and communicating such subblocks," 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/ nextworkplace-revolution/3904/