ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 10, October 2015

Improvement of Network Optimization and Cost Reduction in End To End Process Implementing in Clouds

A. Sree Valli[1], R. Chandrasekhar[2]

PG Scholar, Department of C.S.E, KIET College, JNTUK A.P[1]

Assistant Professor, Department of C.S.E, KIET College, JNTUK A.P[2]

Abstract: Many past systems have explored with the problem is how to eliminate a novel end-to-end traffic redundancy by which improves network efficiency. Several of these systems operate at the application layer while the more recent systems operate on individual packets. In this paper, we present PACK (Predictive ACKs), a novel designed for cloud computing customers end-to-end traffic redundancy elimination (TRE) system. PACK's main advantage is its capability of increase the offloading the cloud-server TRE effort very end client, thus minimizing the processing costs activate by the TRE algorithm. Like other techniques PACK does not require any external support to maintain client's status. By combing cloud computing client the efficiency per physical hosted server is improved greatly, data and resources are hotfoot provisioned provided as standardized offerings to users with a flexible price. But it is important to provide the convenient pricing model for the users of cloud. Hence we design a new traffic redundancy and elimination scheme for reducing the cloud bandwidth and costs. PACK is based on a novel TRE technique, which allows the client to use newly received block to identify previously received block chains, which in turn can be used as reliable, with respective predictors to future transmitted blocks. 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.

Index Terms: cloud computing, traffic redundancy elimination, quality of service**,** caching, network optimization.

1. INTRODUCTION

Cloud Computing provides computer resources as a service and it is a technology revolution offering flexible IT usage in a cost efficient and pay-per-use way. 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. 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. 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 while proprietary middle-boxes are popular point solutions within enterprises; they are not as attractive in a cloud environment. 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. 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. Clearly a TRE solution that puts most of its computational effort on the cloud side may turn to be less cost-effective than the one that leverages the combined client-side capabilities. The sender-based end-to-end TRE solutions add a considerable load to the servers, which may eradicate the cloud cost saving addressed by the TRE and it require to maintain end-to-end synchronization that may degraded TRE efficiency. Here make use of 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.

2. RELATED WORK

2.1 A low-bandwidth network files system (LBFS)

The LBFS is a network file system designed for low bandwidth networks. LBFS exploits similarities between files or versions of the same file to save bandwidth. It avoids sending data over the network when the same data can already be found in the server's file system or the client's cache. Using this technique in conjunction with conventional Compression and caching, LBFS consumes over an order of magnitude less bandwidth than traditional network file systems on common workloads.

2.2 SmartRE: architecture for coordinated network wide redundancy elimination

SmartRE is a practical and efficient architecture for network wide RE. The SmartRE can enable more effective utilization of the available resources at network devices by reducing the wide-area footprint, and by improve end-to-end application performance. Therefore SmartRE can magnify the overall benefits of network-wide RE. SmartRE is naturally suited to handle heterogeneous resource constraints and traffic patterns and for incremental deployment.

2.3 Redundancy in network traffic: Findings and implications

The protocol-independent redundancy elimination, are used
to improve network link performance by removing duplicate strings from within arbitrary network flows, has emerged as a powerful technique to improve the efficiency of network links in the face of repeated data. Many vendors offers such redundancy elimination middle boxes to improve the effective bandwidth of enterprise, data center and ISP link

2.4 EndRE: An end-system redundancy elimination service for enterprises

EndRE is an alternative approach where redundancy elimination (RE) is provided as an end system service. Unlike middle boxes, such an approach benefits both end-to-end encrypted traffic as well as traffic on last-hop wireless links to mobile devices. EndRE uses a new fingerprinting scheme called Sample Byte is much faster than Rabin fingerprinting while delivering similar compression gains. Unlike Rabin fingerprinting, Sample Byte can also adapt its CPU usage depending on server load. End-to-end latencies by up to 30%, and translates bandwidth savings into equivalent energy savings on mobile smart phones. successful solutions that allow nodes to communicate with each other in these extreme networking environments [1]–[3]. Typically, when there is no end-to-end connection between a source and a destination pair, the messages from the source node may need to wait in the intermediate nodes for a substantial amount of time until the connection would be eventually established.

3. EXISTING SYSTEM

Traffic redundancy systems from common end users activities, such as repeatedly accessing, downloading uploading, distributing and modifying same or similar information items (documents, data, web and video). TRE is used to eliminate the transmission of redundant content .To significant reduces 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. Commercial TRE Solutions are popular 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 goal is to reduce customer Bandwidth bill.
 The rise of ―on-demand‖ work spaces, meeting rooms, and work-from-home solutions dethatches 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 synchronisation.
 Current end-to-end solutions also suffer from the requirement to maintain end-to-end synchronisations that may result in degraded TRE efficiency.

4. 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 user's data.

ADVANTAGES OF PROPOSED SYSTEMS

 Our approach can reach the data processing speed over 3Gbs at least 20% faster than Rabin fingerprinting
 The receiver-based TRE solution addresses mobility problems common to quasi-mobile desktop.
 One of them 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.

5. SYSTEM ARCHITECTURE

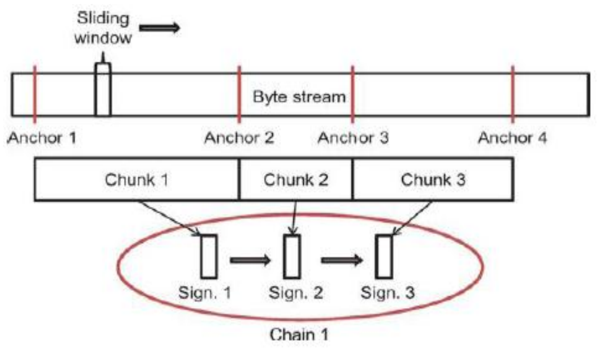


Fig1. System Architecture

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 10, October 2015

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. 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.

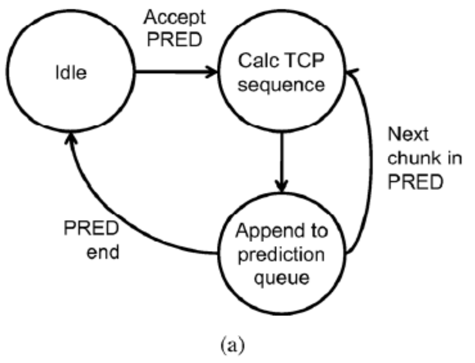
```
Proc. 1: Receiver Segment Processing 1. if segment carries payload data then
2. calculate chunk
3. if reached chunk boundary then
4. activate predAttempt()
5. end if
6. else if PRED-ACK segment then
7. processPredAck()
8. activate predAttempt()
9. end if
Proc. 2: predAttempt()
1. if received chunk 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 not match, 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 to the 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 and includes the prediction in the packet's Options field. Last, the sender sends confirmation message(PRED-ACK) replacing actual data Sender algorithms. (a) Filling the prediction queue. (b) Processing the prediction queue sending PRED-ACK



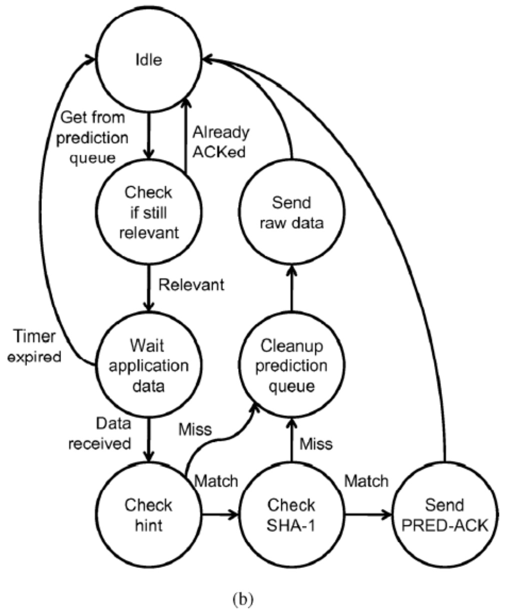


Fig.2. Sender algorithms. (a) Filling the prediction queue. (b) Processing the prediction queue and sending PRED-ACK or raw data.

PACK wire protocol in a nutshell:

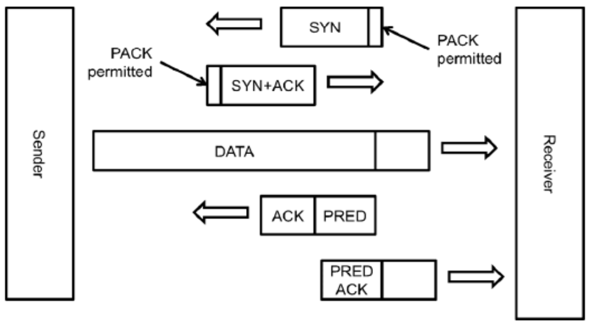


Fig 3 PACK wire protocol

6. 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 from backup 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.

C. Hybrid Approach PACK's receiver-basedmode is less efficient if changes in the data are scattered. In this case, he 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] and [7].

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
```

7. CONCLUSION

The proposed PACK is 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 redundncy based on content arriving to the client from multiple servers without applying a three way handshake.

8. FUTURE WORK

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. Two interesting future extensions can provide additional benefits to the PACK concept. First, our implementation maintains chains by keeping for any chunk only the last observed subsequent chunk in an LRU fashion. An interesting extension to this work is the statistical study of chains of chunks that would enable multiple possibilities in both the chunk order and the corresponding predictions. The system may also allow making more than one prediction at a time, and it is enough that one of them will be correct for successful traffic elimination. A second promising direction is the mode of operation optimization of the hybrid sender–receiver approach based on shared decisions derived from receiver's power or server's cost changes

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]

BIOGRAPHIES

A. Sree Valli pursuing Master of Computer Technology in Software Engineering in KIET Kakinda, East Godavari, A.P, India.

R. Chandra sekhar working as an Assistant Professor in KIET College of Engg, Kakinada. Presently he is pursuing Ph.D in Big Data. His area of interests are cloud computing and Big data.