A Predictive Acknowledgement Model for Enhancing Traffic Redundancy Elimination in Cloud

R.Rathi, L.RamaParvathy

Abstract: ***Due to the huge increase in the utilization of cloud storage in recent days, it leads to a massive growth in data traffic from application based servers to applications like smart phones, which not only influences batteries and computational capacities but as well swamp down multi-hopping strategies during data transmission. To resolve this crisis, traffic redundancy elimination (TRE) is an effectual solution, where the chunks to be transmitted will be directly fetched out from the receivers' cache. Moreover, prevailing solutions cannot be directly applied or it is not appropriate for smart phones owing to its energy overhead ad high computation that is imposed on the applications. In order to overcome this problem, in this investigation, a novel a Predictive Acknowledgement for Eliminating Traffic (PACKET) is proposed which comprises of three significant elements. Initially, every application possess a clone in cloud that are responsible for calculating intensive tasks like detecting redundancy and parsing traffic. Secondly, consider that every cloud user has some specific applications like Facebook to be used in regular day to day life, every clones of cloud has to selectively determine the applications that are most frequently utilized and also reduce the high redundancy ratio. Thirdly, some cloud users always possess certain common applications; the proposed PACKET clusters those clones to co-operatively perform redundancy detection so as to diminish cache resource consumption in cloud. The simulation is carried out in MATLAB environment; the traces of applications are collected from online available data and are utilized for simulation purpose. Experimental outcomes demonstrates that PACKET can attain much higher hit ratio, reduced E2E delay, increased E2E throughput, energy efficiency and effectual bandwidth utilization in contrast to existing approaches. The proposed PACKET shows better and efficient trade-off than prevailing techniques.***

Index Terms: ***Traffic redundancy elimination; Predictive Acknowledgement for Eliminating Traffic; cloud applications; E2E delay, E2E throughput; redundancy; traffic elimination***

I. INTRODUCTION

In recent times, cloud computing offers customers with well-situated and an economical pay-as-you-go service model termed as utilization based pricing [1]. Usually, cloud customers only pay for actual use of resources utilized bandwidth and storage, based on the changing requirements, whereas to use the clouds' in a scalable and proficient manner based on computational capabilities. Specifically, transmission cost or data utilization (i.e. bandwidth) is an important task while tries to reduce costs [2]. Subsequently, cloud customers makes a judicious utilization of cloud resources which influences or motivates to use diverse traffic redundancy techniques, in specific traffic redundancy elimination (TRE) for diminishing bandwidth costs.

In general, Traffic Redundancy Elimination (TRE) grounded from end users' activities such as simultaneously accessing, downloading, uploading (for instance, application backup), deleting, modifying and distributing same information (files, data, videos, documents, audios and web). TRE is cast off to eradicate redundancy of transmission content, and drastically reduces network utilization cost. From the most common TRE solutions, both receiver and sender compare and examines data chunks signature, parsed data content, in accordance to prior transmission. While eliminating redundant chunks, sender modifies or replaces transmission of every redundant chunk with stronger signatures [3] [4] [5]. Traditional TRE solutions are more popular at enterprise network, and involves in deployment of two or more state synchronized middle-boxes, proprietary-protocol at both Cloud Service Providers' (CSP) office or intranet entry points like data centres, thereby eliminating redundant traffic amongst them (for instance, Riverbed [6], Cisco [7], Juniper [8], Quantum [9], Expand Networks [10], Blue coat [11] and F5 [12]).

Meanwhile proprietary middle-boxes are common point of solutions related to enterprises, but they are not much attractive in cloud computing environment. Cloud providers cannot utilize the advantages from technologies whose foremost objective is to diminish customer's utilization costs, and are not probable to invest in these needs. The increase in 'on-demand' work spaces, work from home and meeting rooms, conference halls solutions detaches workers from its corresponding offices [13]. In those sorts of dynamic environments, fixed point solutions are needed for cloud side and server side middle box pair which turns to be ineffective. Subsequently, cloud promotes work distribution among server migrations. Therefore, it is generally agreed that software based, universal, end-to-end TRE is vital in current pervasive environment [14]-[15]. This facilitates the utilization of standard protocol and utilizes a TRE within End-to-End secured traffic (for instance, SSL).

In current cloud scenario, End-to-End TRE solutions are sender based. In some cases, cloud server is the sender of customer queries, these solutions need that server has to monitor clients status constantly. This work focuses of projecting cloud elasticity calls for TRE solutions. Basically, power optimization and load balancing leads to a data migration environment and server side process, wherein TRE solutions that need full synchronization amongst client and server are

Revised Manuscript Received on July 10, 2019.

R.Rathi, Research Scholar, Department of CSE, Saveetha School of Engineering, Saveetha Institute of Medical and technical Sciences, Chennai, rrathi13@gmail.com

Dr.L.RamaParvathy, Professor , Department of CSE, Saveetha School of Engineering, Saveetha Institute of Medical and technical Sciences, Chennai, ramaparvathyl.sse@saveetha.com.

complex to acquire or lose efficacy owing to lost synchronization. Subsequently, popularity in media consumes higher bandwidth and motivates content distribution network (CDN) based solutions, where service points are permanent and mobile users may alters drastically in accordance to relative service point loads and locations as in fig 1. However, in if E2E solution is involved in computational cost and storage at cloud, it should be weighed over bandwidth saving gains.

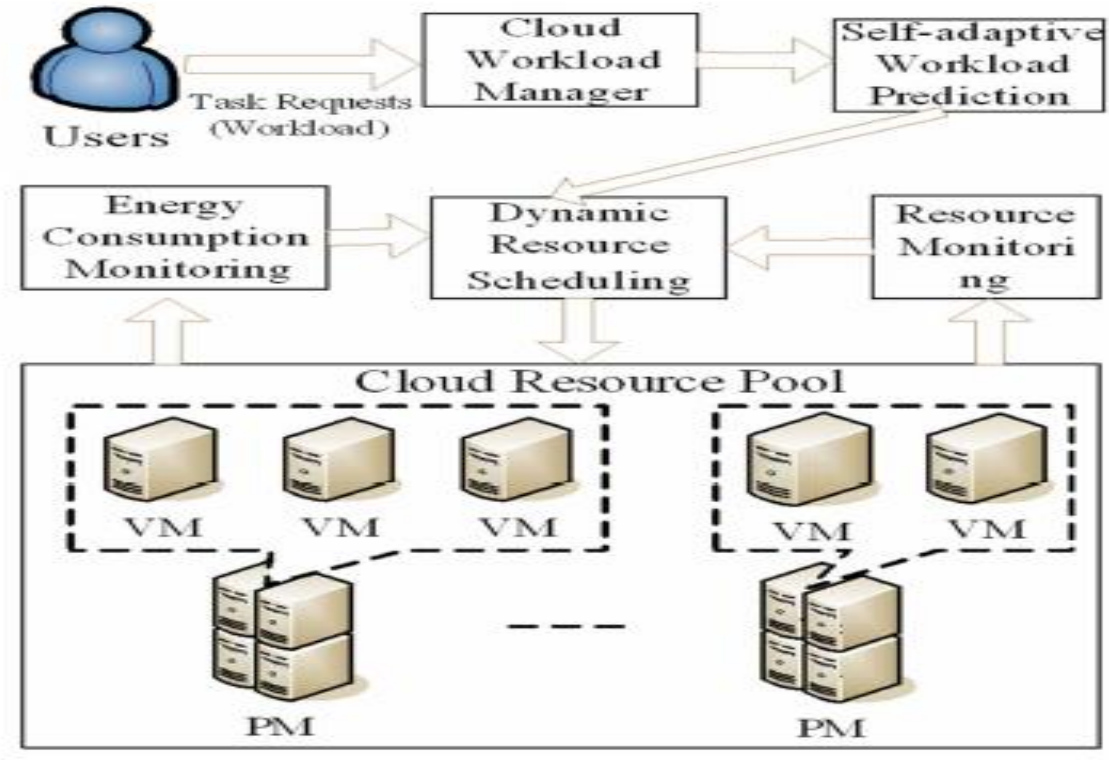

Fig 1: Cloud resource utilization

Obviously, TRE solutions has to put overall computational effort on cloud user side and may turns to be cost effectual than client side applications. End-to-end solution is found that this experimentation offers sender based E2E TRE solutions [14]-[15], which add a considerable load to servers, which may eliminate cloud cost saving addressed by TRE as a preliminary place. This investigation illustrates that present E2E solutions also lacks in monitoring of E2E synchronization that outcomes in TRE efficiency degradation. In this work, a novel predictive acknowledgement based eliminating traffic (PACKET) strategy is utilized for receiver side based E2E TRE solution that basically relies on predicting power to eradicate redundant traffic amongst end users and cloud. There are three essential elements in the proposed PACKET strategy, they are, Initially, every application possess a clone in cloud that are responsible for calculating intensive tasks like detecting redundancy and parsing traffic. Secondly, consider that every cloud user has some specific applications like Facebook to be used in regular day to day life, every clones of cloud has to selectively determine the applications that are most frequently utilized and also reduce the high redundancy ratio. Thirdly, some cloud users always possess certain common applications; the proposed PACKET clusters those clones to co-operatively perform redundancy detection so as to diminish cache resource consumption in cloud. Here, every receiver can monitor the incoming streams and attempts to match receiving chunks with the formerly attained chunks or chunk chains of locally utilized files. By maintaining the chunk meta-data locally for a long time, receiver transmits to server prediction which comprises of easy verification hints and chunks signatures of future incoming data. Sender usually examines hint and carry out PACKET operation only when the hint matches. The essential purpose is to eradicate TRE computation during traffic redundancy at sender side. When redundancy is identified, sender then transmits to receiver ACK for prediction, indeed of transmitting data.

Over the receiver side, this work anticipates a novel computational chunks strategy termed PACKET chunking. PACKET chunking is a novel alternative for fingerprinting which is used conventionally for cloud applications. Experimental outcomes shows that out model can attain higher data processing speeds over 3.5 Gb/s, i.e., 22% higher than conventional techniques.

Application based cloud offloading makes computational effort from cloud to larger client groups performs load distribution, as client processes TRE. TRE solution handles mobility issues for common quasi-model desktop/laptops computational environments. One amongst them is cloud elasticity owing to which servers can be reloaded dynamically around federated cloud, therefore leading clients to communicate with multiple servers. Another IP dynamic property deals with roaming users to change the IP address most frequently. Hybrid approach is suggested in receiver based operation, which facilitated mobile device to transform TRE computation overhead by influencing sender based TRE.

To validate TRE concept, this work performs realistic investigation using PACKET in cloud environment. This experiment illustrates cloud bandwidth utilization cost reduction that is attained at reasonable client while achieving bandwidth savings in client side. This cast off TCP options based applications such as web, video streaming, P2P, email and so on.

In addition, this work evaluates the solution and compares it with previous E2E solution using terabytes of data consumed by various cloud users, which is attained from the records of ISP and CSP. Traffic attained in media like social networking over a month. This solution attains approximately of 35% redundancy elimination without affecting computational effort significantly of sender that results in 25% reduction in overall cost of cloud customer.

This work is organized as follows: Section II shows the review of prevailing TRE solutions. Section III offers PACKET based TRE solutions and explains the prediction procedure and the TRE mechanism. Section IV demonstrates the numerical outcomes that are associated with PACKET. Section V depicts the conclusion and future extension of the work.

I. RELATED WORKS

In [16], TRE system is termed as WANAX that was specifically developed for rising needs of cloud storage and WAN bandwidth scarcity. For commercially expensive hardware, WANAT is software that is based on middle box replacement.

The sender of middle box transmits a signature that holds the backup of TCP streams. Receiver verifies whether data chunks are available in local cache. If chunks are not in local cache, data attained from sender or receiver is closer. Hence, these approaches acquire three handshaking strategies for non-cached data latency.

In [17], the authors explain about an end to end sender based TRE for commercially enterprise networks. It utilizes new chunking strategies that are quicker that the conventionally utilized Rabin fingerprint. Size of those chunks is much smaller in its size, i.e. 32-64B.

The server needs to maintain a reliable and a fully synchronized cache for every client. Size of chunk is maintained to be smaller enough, as system is not appropriate for medium to large content or in long term redundancy.

In [18], the author provides a PACK (Predictive ACKs), which is a novel End-to-End Traffic Redundancy Elimination system, which is designed for customer based cloud computing. Cloud based TRE has to be applied with judicious cloud resource utilization, therefore bandwidth based cost reduction is merged with additional TRE computation cost and storage will be optimized. PACK's significant benefit is its ability of offloading cloud server effort for TRE of end users, thereby reducing processing cost that induced by TRE algorithm.

In [19], the author depicts coding system, where input of system is encoded. It should embrace symbols sequence which repeats input by other input of system.

The encoding specifies the determination of target segment size, also it defines window size by recognizing fingerprint within window symbol at an offset in input data, it describes whether offset has to be designed as cut off point and segments the input data as specified by cut points set.

In [20], redundancy is initiated from users activities such as constantly accessing, uploading, downloading (i.e. backup), deleting, modifying and distributing same information items like files, web, document and video). TRE is utilized to eliminate redundant content transmission and henceforth to significantly diminish network cost.

The most general TRE solutions from both receiver and sender evaluates and compares data chunk signatures which is parsed in accordance to data content, that is prior to its transmission. While redundant chunks are identified, sender modifies transmission of every redundant chuck into storage signature.

As in [21], a commercial TRE solution are more famous at enterprise networks, and involves in deployment of two or more state synchronized middle-boxes, proprietary-protocol at both intranet entry points of huge data centers.

II. PROPOSED WORK

This section discusses in detail about the proposed PACKET model. Here, before analyzing user demands, initially, examine the users requirements such as data structure utilization [22], number of cloud resources used previously and cloud content. By evaluating the cloud data utilization, the users preferences can be sketched out based on demand description and soon. As well, short term and long term fluctuation period and flat period are considered separately. In specific, cloud fluctuation period (CFU_T) and cloud flat threshold (CFA_T) are determined to differentiate fluctuation period and cloud flat period. In various periods, diverse base predictions are adopted based on characters like auto regression model (ARM), exponential moving model (EMA), trend seasonality model (SMA), second moving average model (SMA). The output of those base predictors are transmitted to predictive acknowledgement for eliminating traffic. The transmitting data will be analyzed and trained to offer enhanced accuracy outcomes as an outcome. The outcome of PACKET is utilized to instruct allocated resources in cloud centres like IaaS. Prediction outcomes and actual resource demands are examined based on statistical analysis and diverse criteria. Evaluation outcomes are provided to historical database to enhance prediction performance. The resource demand prediction system model is provided in fig 2.

a. Receiver chunk Repository (RCR)

The proposed PACKET model utilizes a novel chain scheme as in fig 2, where chunks are connected to other chunks in accordance to lastly receiver order. PACKET receiver maintains a chunk repository, which stores huge amount of cache chunks and their corresponding Meta data. Chunks' repository based Meta data comprises of pointers to successive chunks that is received in chunk stream and chunks' signature. Indexing and caching approaches are utilized to effectually retrieve and maintain stored chunks, their corresponding chains and signatures constructed by traversing chunk pointers.

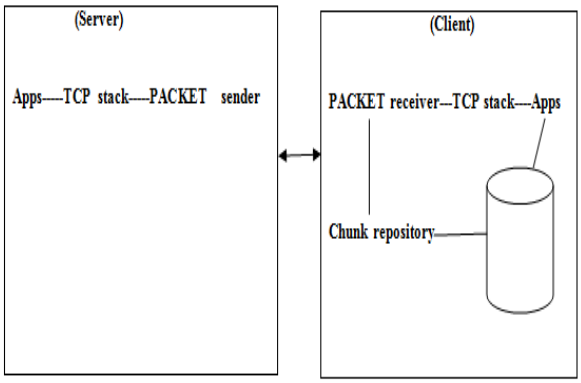

Fig 2: Server-client communication

When a new data are parsed and received at chunk repository, receiver evaluates every chunk signature using Homomorphic algorithm. Here, chunk and its signature are gathered in chunk store [23]. Metadata attained previously from chunks in stream will be updated to point the newly arrived chunk. Here, unsynchronized PACKET nature facilitates receiver to plot every existing file in local file system to chunk chain, which is saved in chunk store and the corresponding metadata is associated with chunk. Receiver share chunks with P2P clients in local network with drives map.

The usage of small chunk size offers better redundancy elimination, while data modifications are fine-grained like sporadic changes in HTML page. However, utilization of small chunk improves storage index size, magnetic disk size and memory usage. This as well increases transmission overhead of virtual data exchanged amongst server and client. Contrasting from IP level TRE solution, which is usually restricted in IP packet size, PACKET operates over TCP streams and therefore deals with larger chunks and complete chunk chains.

b. Receiver Algorithm

With the arrival of new data into the cloud, receiver computes corresponding signature for every chunk and searches for associative match of local chunk in the repository. If chunk signatures are identified from the repository, receiver determines whether it is attained from previously received chunk storage, by evaluating the corresponding chunks' metadata. Consequently, receiver transmits a prediction to sender for numerous further expected chunk chains. This chunk prediction determines the starting point in byte stream (that is, offset) and recognizes several corresponding chunks (PACKET commands). Based on prediction, sender broadcasts PACKET confirmation/ verification message [24]. As PACKET message is received, receiver copies chunk data to TCP input buffers, providing it to related sequence numbers.

Here, cloud receiver acquires normal TCP ACK by periodically expecting TCP sequence number. In these cases, predictions are false, wherein one or more chunks predicted are transmitted already, sender continues normal operation, for instance, sending raw data, without transmitting PACKET message.

Algorithm 1:

1. If cloud data packet and packet segment carries payload information
2. Then
3. Compute chunk
4. If chunk size attains its Max-LIMIT
5. Then
6. Activate PACKET verification message
7. End if
8. Else if generate PACKET_ACK message into data segment
9. Then
10. Process PACKET_ACK ( )
11. Execute PACKET verification based cloud chunk
12. End if

Algorithm 2: PACKET verification

1. If received cloud chunk matches with previous chunk in chunk repository
2. Then
3. If chunk chain determines (chunk)
4. Then
5. Transmit TCP-ACK with PACKET based on available free chunk space
6. Exit
7. End if
8. Else
9. Store chunk
10. Relate chunk with chunk repository
11. End if
12. Send TCP-ACK only

Algorithm 3: Process PACKET-ACK ( )

1. ∀ offset $\in$ PACKET-ACK do
2. Verify data in chunk repository
3. Store data in TCP input buffer
4. End for

c. PACKET-ACK phase

When the transmitter receives PACKET-ACK message tries to match PACKET ACK received with buffered data (to be transmitted). For every prediction, sender describes the associated TCP sequence value and verifies the metadata hint. Based on the match between hint, cloud transmitter evaluates more intensive computationally homomorphic signature for predicted data range and evaluates the signature with PACKET verification message. If the hint does not match with the data chunk in repository, computational operation is saved. If the homomorphic signature matches with sender information safely, consider that receiver PACKET-ACK message is appropriate. It substitutes buffered data with PACKET verification message. Fig 3 depicts operation of state machine. Initially, the sender determines the TCP sequence range to be sent already or not. If the range is already acknowledged, corresponding prediction will be discarded. Else, it attempts to match the prediction data to its TCP buffer data.

d. Predictive ACK receiver window

Here, PACKET facilitates receiver to attain senders' information locally while local copy is accessible, hence eradicates data to be transmitted. Cloud receiver fetches local data based on virtual data reception (VDR). While broadcasting huge virtual data, connection extends to some limit, which may be restricted by number of predictions by the receiver. Thus, in turns, determines that receiver prediction and sender verification will be expected to reach higher virtual data rate. For instance, if there is cyclic prediction success, receiver side algorithm gradually increases prediction range and TCP rate adjustment process.

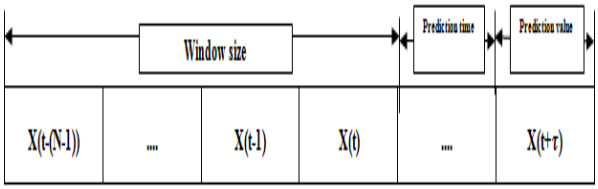

Fig 3: Window size

PACKET facilitates larger prediction size by either transmitting numerous successive PACKET verification messages or by enhancing PACKET range to deal with several chunks. PACKET facilitates receiver to merge numerous chunks into a nominal range, as sender is not known to anchor the receiver's data chunk algorithm. The combined range possesses new hint and new homomorphic signature with the concatenated content of chunks. Variable prediction size introduces receiver window notion of receivers' window [25]. Receiver's upper bound is virtual window for aggregating huge amount of data for all predictions that are still pending as in Fig 3. Virtual window is a primarily set a minimal value, that is identical to receivers flow control. Receiver enhances virtual window with every successful predictions, in accordance to above predictions.

With chunk match, receiver predictions are restricted. Before prediction at sender stage, data is transmitted previously from it. When original data arrives, receiver guarantees prediction and maximizes virtual window. PACKET ACK message increases virtual window size. This procedure sensibly resembles TCP rate control algorithm.

During the occurrence of mismatch, receiver switches back to initial virtual window. Homomorphic signature determines receivers' side performance. PACKET ACK behaviour during data segment for payload arrives after prediction was transmitted and virtual window is doubled. PACKET verification specified successful acknowledgement message reception (ACK message) from sender. Receiver observes data from chunk repository. It then alters subsequent byte of sequence number to last byte of redundant data that is counted as plus one and transmits TCP ACK that is piggybacked with further prediction. At last, virtual window size is doubled.

The virtual window size increases during the introduction in the trade off based prediction fails at some point. PACKET ACK message and PACKET verification message determines the receivers' behaviour while data arriving does not fit with recently transmitted predictions. Newly received chunks may start a new chain match. With the data reception, receivers transmits back to initial virtual window (guaranteeing normal TCP window size of receiver) until new match is identified in chunk repository. Note that, slide change in senders' data causes entire prediction range to be transmitted to receiver side as raw data. Therefore, with larger virtual window, a trade off is encountered amongst potential rate gain and recovery effort in missed prediction case.

e. Cloud as a Receiver (CaaR)

In the growing need of cloud, cloud storage turns to operate in a dominant role, i.e. sharing services and backup to E-mail services and huge cloud servers. In many of these cloud services, cloud is frequently acts as a data receiver. If sending client has no power constraint, PACKET functions to save bandwidth on uploading data to cloud storage. In these cases, end users' plays the role of sender and cloud server is the receiver. PACKET algorithm 1, PACKET algorithm 2 and PACKET algorithm 3 need not be changed. It can be achieved when the cloud server works like a cloud receiver, i.e. it has to maintain a chunk repository. Based on the PACKET algorithm, receiver side is less effectual during the changes in scattered data. Here, prediction strategies are constantly interrupted, which makes sender to relapse transmission till new match is provided and updates it to sender. Finally, PACKET algorithms' mode of operation determined in Algorithm 1 and Algorithm 2. When the algorithm identifies a pattern of dispersed changes, it selects a sender driven approach for further execution.

f. PACKET based resource prediction

As shown in figure 2, there are numerous opposite factors to be considered, like short term resource requirement demand and long term resource requirement demands along with cloud fluctuation and flat period. In general, Cloud fluctuation period is an abnormal violation of cloud resources for a period of time. Based on the user's long term resource requirements, there are certain factors to be considered:

1) Regularity is extremely obvious that short term resource failure. It is generally shown that long term users may offer certain repetition regularity.

2) Long term execution of cloud service leads to fluctuation and cloud flat period for certain duration of time. Based on short time requirements, regularity may not be much obvious, and fluctuation feature is highly noticeable.

Henceforth, short term or long term will not be conflicted with cloud flat of fluctuation periods. In case of long term data, it is determined with regularity. However, fluctuation and flat period should be differentiated. In case of short term data, regularity is not so easier to demonstrate and fluctuation may be induced. Prediction speed has to be guaranteed as short term resource availability which provides first place to faster response than other performance. Difference between long term and short term processing significantly concentrates on fluctuation processing period. Thereby, cloud resource based fluctuation and flat period was discussed effectually.

g. Threshold computation of flat period

With respect to the characteristics of flat period, data moving average (DMA) is utilized; it effectually diminishes lagging deviation amongst actual value and prediction value. In this technique, depict the sliding window concept, whose input value size will be 'N', i.e. $X = [x_t, x_{t-1} \dots, x_{t(N-1)}]$ over a time period of $[t-(N-1), t]$. Figure 3 shows sliding window model with window size 'N'.

The output value attained is $X_{t+\tau}$ with the prediction time interval of 'τ' that is a dependent variable set of 'X', where $X = x_1, x_2, \dots, X_N]$. The relationship between the above values is as follows in Eq.1:

$$X_{t+\tau} = f(X, \tau) \quad (1)$$

The i^{th} user's requirement for resource prediction at time $t + \tau$ is provided as follows in Eq. 2:

$$X_{t+\tau}(i) = a_t(i) + \tau b_t(i) \quad (2)$$

Based on the Equation above, $x_{t+\tau}(i)$ is prediction value in time interval $t + \tau, \tau$ is sequence number to be predicted at $a_t(i)$ and $b_t(i)$ that has to fulfill the following Eq. 3 & 4:

$$a_t(i) = 2M_t^1(i) - M_t^{(2)}(i) \quad (3)$$

$$b_t(i) = \frac{2}{N-1}[M_t^{(1)}(i) - M_t^{(2)}(i) \quad (4)$$

From the formula 4, $M_t^{(1)}(i)$ specifies first data moving average, similarly, $M_t^{(2)}(i)$ specifies second data moving average at t^{th} time of resource with i^{th} user request. 'N' is time period of data moving average period. As well, $M_t^{(1)}(i)$ and $M_t^{(2)}(i)$ is represented as below in Eq. 5 & 6:

$$M_t^{(1)}(i) = \frac{x_t(i) + x_{(t-1)}(i) + \dots + x_{t-(N-1)}(i)}{N} \quad (5)$$

$$M_t^{(2)}(i) = \frac{M_t^{(1)}(i) + M_{(t-1)}^{(1)}(i) + \dots + M_{(t-1)}^{(1)}(i)}{N} \quad (6)$$

Then, total amount of cloud resource requested by 'm' users are as in Eq. 7,

$$X_{t+\tau} = \sum_{i=1}^{m} x_{t+\tau}(i) \quad (7)$$

From the above analysis, the prediction time $t + \tau$ is determined with the values of 'N' period at time 't' for the total cloud users. Prediction result of cloud users at time 't' is analyzed.

h. Threshold computation of fluctuation period

Exponential data moving approach (EDMA) is an effectual technique for short term prediction and specifically appropriate for time series prediction of non-seasonal effect based on its faster responsiveness and weight reduces with time passes while cloud processing. Predicted values are evaluated based on smoothing constant. Exponential data moving approach is provided as in Eq. 8:

$$x_{t+1} = \alpha X(t, N) + (1 - \alpha) x_t \quad (8)$$

From Equation above, (t, N) is moving average value amongst past time $t-(N-1)$ and present time $'t'$. Therefore, time interval is $'N'$. α is smoothing constant evaluated using $\alpha = {}^{2}/_{(N+1)}$. It is identified that α is related to [0,1].

EDMA method offers higher weight to lower measure value and lower weight to upper measure value. Therefore, EDMA technique is capable to respond quicker to fluctuations in short time cloud resource demand and workload conditions. Moreover, there will be some delay introduced in gradual increase in window size. Based on the non-linear load prediction, polynomial orders are selected appropriately. If order is lesser $(degree \leq 2)$, then function will not react faster to load changes If order is higher $(degree \geq 3)$, then it is identified that the process is complex in resource allocation and certain undesirable fluctuation is introduced. Cost may be too expensive for run time context.

i. Recognizing cloud flat and fluctuation period

Even though there are diverse prediction techniques in accordance to resource availability levels, it is complex to identify the boundary limits of resources in various time intervals as in Fig 4. In this segment, cloud fluctuation threshold (CFU_T) and cloud flat threshold (CFA_T) is determined to analyze fluctuation and flat time interval. Fluctuation threshold value is determined as upper limit boundary value based on the demand of cloud resources, while flat threshold value is determined as the lower boundary limit for the need of cloud resources. In the last $'n'$ time period, if difference d_t of prediction value x_t and x_{t-1} in series with $\{x_t, x_{t-1}, \dots m\ x_t - (n-1)\}$ is higher than that of d_u, then CFU_T is reached. If difference of x_t and x_{t-1} is lesser than the d_l value, then CFA_T is reached. For certain resources like $'i'$, users' demands are fluctuating, as demand data in last 'k' times specifies that $Demand_{degree} \geq CFU_T)$, $Demand_{degree}$ specifies fluctuation degree of prediction time. CFU_T is upper limit value. Resource type like 'i' are experiencing flat period, when demand data in last 'k' fulfils $Demand_{degree} \leq CFA_T . Demand_{degree} \leq CFA_T \leq CFU_T$, resource demands are involved in fluctuations and flattening.

j. Approximation method

Let $'f'$ determines configuring the cloud demand and $'F'$ determines set of all feasible configuration for allocating cloud resources without traffic. Let u_f is a system objective provided for configuring cloud resources. Note that u_f is generally based on combinatorial optimization problems, in specific, no effectual computation solution is determined, specifically when number of variables increases, that causes exponential growth in solving problems. Luckily, certain investigators have offered Markov approximation framework to deal with this optimization problem, which is specifically handled for effectual computation. Therefore, it is essential to resolve approximation problem as in Eq. 9 & 10:

$$\min_{p\geq 0} \sum_{f\in F} p_f u_f + \frac{1}{\beta}\sum_{f\in F} p_f \log p_f \quad (9)$$

With respect to,

$$\sum_{f\in F} p_f = 1 \quad (10)$$

where p_f is probability for configuring f for adopting resources for cloud users. Positive constant $'b'$ is utilized to handle approximation accuracy. In accordance to Eq. 11, optimal solution for resource configuration as,

$$p_f^*(u) = \frac{\exp(-\beta u_f)}{\sum_{F'\in F}\exp(-\beta U_f'}} \quad (11)$$

Optimal system based objective value are provided as in Eq. 12:

$$\hat{u} = -\frac{1}{\beta}\log[\sum_{f\in F}\exp(-\beta u_f)] \quad (12)$$

Then, it is essential to determine the time-reversible Markov chain for allocation the resources to cloud users for certain time interval, in which the state space offers set of feasible configuration with stationary distributions provided by p_f as in Eq. 13. With this design, data hopping among various states is provided. As Markov chain converges, approximation solution is identified.

Algorithm 4:

1. For prediction value $x_t, x_{t-1}, \ldots, x_{t-(n-1)}$ do
2. Compute differences $d_t, d_{t-1}, \ldots, d_{t-(n-1)}$ to adjacent value
3. End
4. Set CFA_T and CFU_T as threshold value
5. Compute benchmark values d_u and d_l
6. If x_t and x_{t-1} is lesser than the d_l value, then CFA_T is reached
7. CFA_T is reached
8. Else if $Demand_{degree} \geq CFU_T$
9. CFU_T is upper limit value
10. End
11. For every resources 'i' do
12. Evaluate fluctuation degree
13. If it is greater than d_u
14. Resources experiences fluctuation period
15. Else if it is lesser than d_l
16. Resources experiences flat period
17. Else resource demand experiences both fluctuation and Flatness
18. End

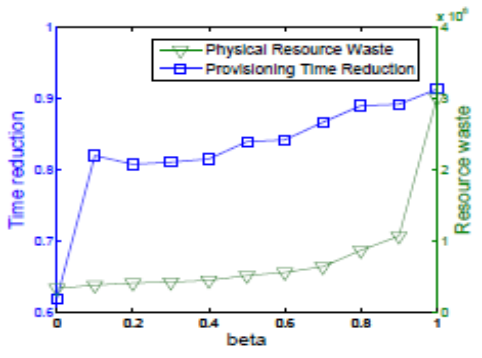

Fig 4: Resource utilization

Assume that, if a Virtual Machine (VM) placement request is generated from a system with independent request, and the time period is $1/m$. Therefore, it is identified that 'n' VM in the system follows M/M/1 queue model for stationary distribution where $p_n = me^{-r/n!}$, in which $r = 1/m$. While system recognizes VM which resides in it, F_n is feasible configuration, (n, f_n) is system state given by f_n. Ratio of time remains determine system objective as in Eq. 13:

$$p_n f_n = \frac{\exp(-\beta_n u_{f_n}^n)}{\sum_{f\in F}\exp(-\beta_n u_{f_n}^n)}\ \pi_n, for\ all\ f_n \epsilon\ F_n, n \quad (13)$$

Where β_n specifies positive constant based accuracy adjustment. In this design, system state multi-hopping from one to another VM specifies VM arrival or departure or VM migrations. To carry out Markov chain, targeted stationary distribution p_n, f_n causes VM migrations as possible, it only facilitates state transition in VM that departs or arrives from the system. Based on time reversible Markov chain, base equation has to be fulfilled: $p_n, f_{nq}(n, f_n) - (n+1, f_{n+1} = p_{n+1}, f_{n+1}, q(n+1, f_{n+1}$ where $q(n, f_n) - (n+1, f_{n+1})$ specifies system transient rate (n, f_n) to state $(n+1, f_{n+1})$. If current system state is (n, f_n). Consider 'c' which specifies available configuration for new VM placement request. Model the state transient rate as in Eq. 14:

$$q_{(n,f_n)} \to (n+1, f_{(n+1)} = \lambda \frac{\exp(-\beta_{n+1} u\_(f_{n+1}^{n+1}))}{\sum_{c'\epsilon C, g_{n+1}=f_n U)}\exp(-\beta_{n+1} u\_(f_{n+1}^{n+1}))} \quad (14)$$

Upon a VM departure, we need to update the system to achieve optimal configuration. Combine (5) and (7), we obtain Based on VM departure, update system to attain optimal configuration, by combining (13) and (14). It is determined from Eq. 15 below:

$$q_{(n,f_n)} \to (n+1, f_{(n+1)} = (n+1)\mu \frac{\exp\left(-\beta_{n+1} u_{f_{n+1}^{n+1}}\right)}{\sum_{c'\epsilon C, g_{n+1}=f_n U)}\exp\left(-\beta_{n+1} u_{f_{n+1}^{n+1}}\right)} * \frac{\exp\left(-\beta_{n+1} u_{f_{n+1}^{n+1}}\right)}{\sum_{c'\epsilon C, g_{n+1}=f_n U)}\exp\left(-\beta_{n+1} u_{f_{n+1}^{n+1}}\right)} \quad (15)$$

From the algorithm given below, a new VM placement request arrives; place it on candidate physical machine with transition probability. Based on VM departure, choose one VM from probability migration and network state to another host on various rack. In accordance to Eq. 15, candidate migrated VM is selected from where more traffic is generated into network outside data racks.

Algorithm 5:

1. Let VM_k placement request arrives
2. The host machine g_k accommodate VM_k
3. $F_k \to \{f_{(k-1)} * g_k\}$
4. Select $f \in F_k$ with transition probability

5. Perform VM placement based on system configuration
6. $f_k \rightarrow f$
7. VM departure:
8. VM_i → VM transmits most traffic to rack in data centers
9. $F_k \rightarrow$ Executes available host set outside the rack
10. For every physical machine
11. Let configuration starts from VM_i to j;
12. Calculate transition probability
13. End
14. Execute migration based on probability computation
15. $f_{k-1} \rightarrow f$

IV. NUMERICAL RESULTS AND DISCUSSIONS

This section explains about the numerical results of the proposed PACKET in detail. The simulation was carried out in MATLAB 2018a, it evaluates potential redundancy for abundant applications that resides in cloud to observe PACKET performance and bandwidth costs for redundancy elimination process. The computation is performed in 1) text samples captured by ISP 2) traffic acquired from social network service; and 3) benchmark dataset of workloads. In this segment, evaluation is performed in average chunk size of 8 KB, through the anticipated outcome that facilitates every client to utilize diverse chunk size.

a. E2E redundancy

It rises when self-similarity in traffic created by end-users. It is determined that end users usually download similar data like movies or other related items very often. These mentioned items are the significantly interesting redundancy developed by end users that skip backward and forward and produces several overlapping during downloading process. Such skip is performed approximately of 15% in this session and longer movie size of about (50 KB). Consider that in initial stage, cache is empty; it consumes certain time for chunk cache to fulfil and build steady state. In steady state process, approximately 35% traffic is acknowledged as redundancy and eliminated. Wrapping, time length for downloading cache movies, which outcomes in definite replays which does not causes download and all. Table I shows the parameter setup of proposed model.

Table I: Parameter setup

Parameters	Records
Traffic volume	1 TB
Maximum Speed	473 Mbps
Estimated packet TRE	30%
Session	146430
Videos	40,000
IPs address of end users	38,081

In this experimentation, to derive an effectual PACKET redundancy elimination, chunk-level redundancy requires to be applied in long chunk chains. To evaluate this phenomenon, the distribution of redundancy chain is determined in Email datasets. The resultant redundant data chain length distribution is shown below. 55% of chunks are described, Email about 90%. Moreover, chunks are extremely suitable to reside in data chunks as in Fig 5. These findings helps to determine redundancy in single chunk, it will be continued in next chunks. Moreover, this evaluation determines that videos and large files with small amount of redundant chunks and changes that are reside in chunk that are effectually handled by receiver side chunk TRE. In this segment, sender performance of PACKET and sender based E2E TRE is evaluated.

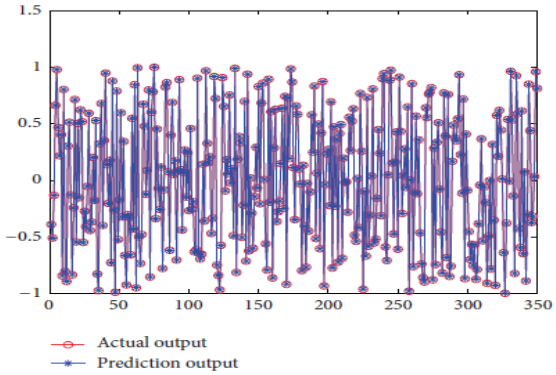

Fig 5: Random Prediction output

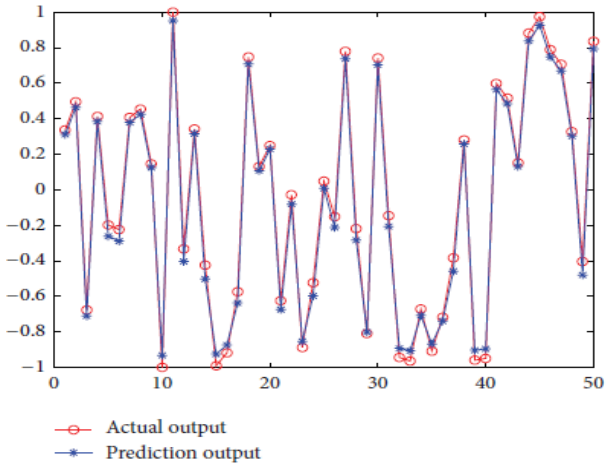

Fig 6: Actual prediction output

b. Server Computation

Initially, the computational effort of server is evaluated. Here, server is essential to perform Homomorphic functionality over diverse range in bytes (prediction specifies offset, starting point and prediction size) after validating hints; broadcasts prediction and describes match as in Fig 6. In sender based TRE, serve is essential to evaluate the stream slices into chunks, and compute homomorphic signature of every chunk to transmit it. Table II depicts server based computational effort of sender TRE in PACKET.

Table II: PACKET parameters

	No TRE	PACKET	Server based computation
Traffic volume	9 TB	6.5 TB	6.3 TB
Cost reduction		35%	32%
Server utilization cost		6.0%	19%
Total operation cost	100%	80.5%	85%

c. Node synchronization

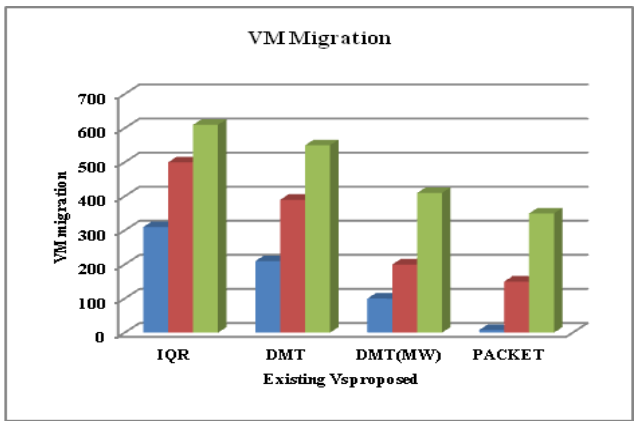

Fig 7: VM migration of PACKET

Numerous sender based E2E TRE method needs full synchronization amongst sender and receiver caches. When synchronization is performed, redundancy is identified and it is eliminated by the sender. During synchronization, it saves the three way handshake and eliminates redundancy chunks that arrive at receiver side by various senders. This crisis is eliminated in PACKET, however it does not accounts for efficiency computation in recent study.

d. Users Mobility

With the social media dataset, the effect of user mobility is determined with TRE as in Fig 8. It is obvious that, PACKET is not influenced to mobile devices, however users who uses the corresponding station from diverse locations. Moreover, in this experimentation, it is concentrated that user are connected through 3G cellular networks with numerous device types (smart phones, PCs and so on). Users have to complete the registration process, where the unique mobile number will be entered and acquires password through SMS messages.

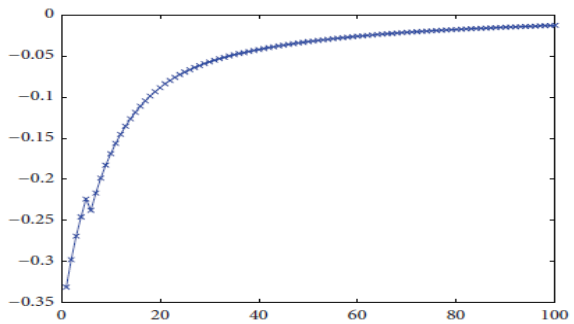

Fig 7: User convergence rate

e. Cost Estimation

Usually, TRE diminishes cloud traffic costs and increases server efforts for TRE computation based on increased server-hours cost. Cloud cost is evaluated based on serving social media videos as described and three components are analyzed: with PACKET, without TRE and sender-based TRE. Cost computation considers traffic throughput and server hours while eliminating storage costs that are determined to be very similar to examined setups.

Baseline of this comparison is measurement of single video server outputs 350 Mb/s to 600 clients concurrently. Cloud is provided with server array, set policy to have less than 0.25 computation powers. Server is eliminated from array, if 0.5 CPU power is left.

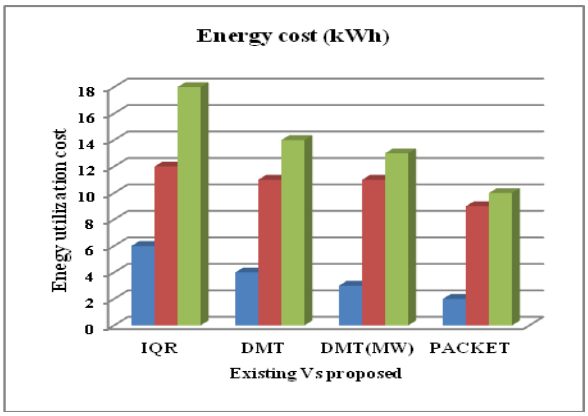

Fig 9: Energy cost of cloud utilization

Fig 9 shows the Energy cost of the proposed PACKET. This approach shows better trade off than IQR, DMT, DMT (MW).

V. CONCLUSION

Here, cloud computing is expected to trigger higher demand for TRE solutions as huge amount of data is exchanged amongst the cloud server and the end users as the need increases drastically. Cloud environment redefines a TRE system requirement that makes middle ware solution insufficient. Conversely, there is a huge rise for the need of TRE solution that diminishes clouds' operational cost during the computation of delay and throughput, user mobility and flexibility.

The evaluation of utilizing extensive collection of content types demonstrates that PACKET has to fulfil design goal expectation and sender based TRE, distinctively when buffering and computation cost is essential. However, PACKET possess additional effort on sender, when redundancy is eliminated, therefore reducing clouds' overall cost.

Initially, this implementation maintains chunk chain by maintaining the sequence of chunks in TRE fashion. Flat and fluctuation period is computed and evaluated along with MC based approximation method. This leads a promising solution for optimization process when a hybrid sender-receiver approach based sharing is retrieved from servers' cost change or receiver's power. For future extension, analysis the statistical evaluation of chuck chains that would facilitates multiple possibilities of ordering and prediction. This can also be evaluated in fog

computing as in edge computing.

REFERENCES

1. L. Qian, Z. Luo, Y. Du et al., "Cloud computing: an overview," in Cloud Computing, vol. 5931 of Lecture Notes in Computer Science, pp. 626–631, Springer, Berlin, Germany, 2009
2. C.-Y. Yeh, C.-Y. Kao, W.-S. Hung et al., "GPU virtualization support in cloud system," in Grid and Pervasive Computing, vol. 7861 of Lecture Notes in Computer Science, pp. 423–432, Springer, Berlin, Germany, 2013.
3. H.-S. Wu, C.-J. Wang, and J.-Y. Xie, "Terascaler ELB-an algorithm of prediction-based elastic load balancing resource management in cloud computing," in Proceedings of the 27th International Conference on Advanced Information Networking and Applications Workshops (WAINA '13), pp. 649–654,March 2013.
4. H. Zhang, P. Li, Z. Zhou, X. Du, andW. Zhang, "A performance prediction scheme for computation-intensive applications on cloud," in Proceedings of the IEEE International Conference on Communication (ICC '13), pp. 1957–1961, 2013.
5. V. Manish, G. R. Gangadharan, V. Ravi, and N. C. Narendra, "Resource demand prediction in multi-tenant service clouds," in Proceedings of the IEEE International Conference on Cloud Computing in Engineering Markets, pp. 1–8, IEEE, Bangalore, India, October 2013.
6. X.Kong, C. Lin, Y. Jiang,W. Yan, andX.Chu, "Efficient dynamic task scheduling in virtualized data centers with fuzzy prediction," Journal of Network and Computer Applications, vol. 34, no. 4, pp. 1068–1077, 2011.
7. Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 6, pp. 1107–1117, 2013.
8. Y.-C. Chang, R.-S. Chang, and F.-W. Chuang, "A predictive method for workload forecasting in the cloud environment," in Advanced Technologies, Embedded and Multimedia for Human-Centric Computing, vol. 260 of Lecture Notes in Electrical Engineering, pp. 577–585, Springer, Amsterdam, The Netherlands, 2014.
9. J. J. Prevost, K.M. Nagothu, B. Kelley et al., "Prediction of cloud data center networks loads using stochastic and neural models," in Proceedings of the 6th International Conference on System of Systems Engineering, pp. 276–281, 2011.
10. W. Fang, Z. Lu, J. Wu, and Z. Cao, "RPPS: a novel resource prediction and provisioning scheme in cloud data center," in Proceedings of the IEEE 9th International Conference on Services Computing (SCC '12), pp. 609–616, IEEE, Honolulu, Hawaii, USA, June 2012.
11. Miskhat, S. F., Rahman, R. M., & Amin, M. A. (2011). Neural network and regression based processor load prediction for efficient scaling of Grid and Cloud resources, Computer and Information Technology (ICCIT), Dhaka, Bangladesh, 22-24 Dec. 2011, 333-338.
12. Wu, Y. H., Cao, J., & Li, M. L. (2013). Energy Efficient Allocation of Virtual Machines in Cloud Computing Environments Based on Demand Forecast. Journal of Chinese Computer Systems, 34(4), 778–782.
13. Zhang, F. F., & Wu, J., Lü, & Z. H. (2013). Survey on prediction models in cloud resource management schemes. Computer Engineering & Design, 34(4), 778–782.
14. Zhou, W. J., & Cao, J. (2012). Cloud Computing Resource Scheduling Strategy Based on Prediction and ACO Algorithm. Computer Simulation, 29(9), 239–242.
15. Lin, W. W., Liang, C., Wang, J. Z., & Buyya, R. (2014). Bandwidth-aware divisible task scheduling for cloud computing. Software, Practice & Experience, 44(2), 163–174. doi:10.1002/spe.2163
16. Beloglazov, A.; Buyya, R. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurr. Comput. Pract. Exp. 2012, 24, 1397–1420
17. Clark, C.; Fraser, K.; Hand, S. Live migration of virtual machines. In Proceedings of the Symposium on Networked Systems Design and Implementation, Berkeley, CA, USA, 2–4 May 2005; DBLP: Trier, Germany, 2005; pp. 273–286.
18. Xu, M.; Tian, W.; Buyya, R. A survey on load balancing algorithms for virtual machines placement in cloud computing. Concurr. Comput. Pract. Exp. 2016, 29.
19. Khan, M.A.; Paplinski, A.; Khan, A.M.; Murshed, M.; Buyya, R. Dynamic Virtual Machine Consolidation Algorithms for Energy-Efficient Cloud Resource Management: A Review. In