Research Journal of Applied Sciences, Engineering and Technology 7(21): 4559-4564, 2014
DOI:10.19026/rjaset.7.834
ISSN: 2040-7459; e-ISSN: 2040-7467
© 2014 Maxwell Scientific Publication Corp.
Submitted: January 10, 2014 Accepted: February 15, 2014 Published: June 05, 2014

Research Article

Outliers Based Caching of Data Segment with Synchronization over Video-on Demand using P2P Computing

[1]M. Narayanan and [2]C. Arun
[1]Depertment of Computer Science and Engineering, Saveetha School of Engineering, Saveetha University,
[2]Depertment of Electronic and Communication Engineering, R.M.K. College of Engineering and Technology, Chennai, Tamilnadu, India

Abstract: Nowadays live streaming plays an important role in various field of real time processing like education, research etc. The videos are maintained at the server then the clients may access the videos from the server may leads the performance problem. So the videos are downloaded by the clients and send it to the requesting clients. The mesh network is suitable for the live streaming because there is no master/slave relationship among the clients and in P2P (Peer-to-Peer) live streaming, each peer holds the video segment for satisfying the needs of the requesting peer. If the cache is full the data segments are replaced by using various replacement algorithms. These algorithms are mainly used in the data segment at the head part of the cache. The tail part of the data segments are never used to satisfy the peers even though it is a relevant data segments. The proposed work mainly focuses on the data segment of the tail part for live streaming is called outliers. The tail part of the cache is synchronized with the other neighboring peers with the help of the segment table. This segment table has to maintain each peer to overcome the unavailability of the data segment at the peers. There is various tag formats are proposed for representing the tail part of the cache. In future the performance will be improved in maximum level.

Keywords: Caching, distributed systems, P2P computing, VoD

INTRODUCTION

There are many ways of sharing the information across the systems by using the internet. The files are available at the server and the clients can access those files by generating the request. The response of the server is send back to the client which is not synchronized with the response. The client requested the video from the video server is different from the ordinary file processing. If the number of request is increased it will lead the server gets overloaded. This problem is overcome by using the replicated video server for satisfying the client requests. The master/slave relationship between the clients and video servers are also an important issue in the live streaming. The performance of the present live streaming has been lacking in higher rate when compared to the traditional way of handling the video. P2P computing has been introduced for handling the live streaming over the internet. Each peer can hold the data segments in cache in order to reduce the workload of the video server. The requesting peer can get the data segment from the other peers or from the video server. The data segments are always downloaded by the peer will store it into the cache. Suppose the cache is full then the data segments are replaced with the newly arrived data segments. There is various replacement algorithms has been introduced such as First In First Out (FIFO), Most Copied Cache (MCC), Least Recently Used (LRU), Least Frequently Used (LFU) etc., (Luo *et al*., 2009). These algorithms hold the data segments based on the popularity. The most popular data segments are stored in the head part of the cache where as the least popular data segments are maintained at the tail part of the cache. The cache servers can be deployed at the access points with various regions for uploading the data segments. Then P2P downloading request redirects the data segments with associated regional network. The peer (server) will fetch the data from other peers when the required data is not available at the cache. The old data segments are removed from the cache space when the cache is full. The channel popularity is important in the live streaming for designing a cache algorithm. A novel caching algorithm SLW (Sliding Window) in P2P live streaming has been introduced for improving the best performance among the other replacement algorithms (Xu *et al*., 2010). The mesh based P2P attack is the malicious nodes which become neighbor of

Corresponding Author: M. Narayanan, Department of Computer Science and Engineering, Saveetha School of Engineering, Saveetha University, Chennai, Tamilnadu, India

the other peers which leads the security problem. This malicious node will delay the sending packets to the receiving peers. The novel reputation scheme is combined with the feedback which comes from the overlay network. The evaluation has been placed over the real world applications by using various mechanisms of the P2P live streaming (Seibert *et al*., 2011). The buffer based request/reply has been implemented using the RQRP pull delivery model to accept the audio and video streams in VoD systems. There are two variations of VoD systems namely half duplex and full duplex modes are also evaluated (Poon *et al*., 2000). There are two algorithms of multi-video-source for streaming media systems are proposed namely serial and parallel. These algorithms are properly switched according to the given situation and the mobile features of the peers are also validated in order to get a better video quality by the client. These algorithms suffer only in the online system because of the shorter running time (Xie *et al*., 2012). A central P2P content distribution problem can be identified by achieving the maximum throughput by negotiating the neighboring peers in content exchange with each other. There are several versions of problem has been studied such as request allocation, neighbor selection and server load minimization. The Discrete, distributed and adaptive practical algorithms are evaluated in order to eliminate the load balance problem (Wang *et al*., 2011). The channel surfing involves in switching overlay which introduces a delay and problem in users experience when compared to the multicast-based IPTV. OAZE (Overlay Augmentation for Zapping Experience) is a distributed system which reduces the cross domain traffic and also speed up the switching process. Each peer will maintain the connections of other neighboring peers to focus on the channel assignment problem. This system improves the connection between the peers and to it reduces the switching delay and network cost (Chen *et al*., 2012).

There are various model has been proposed in P2P live streaming but all methods are mainly concentrated on the data segments which are available at the cache. If the cache is full the data segments are replaced with the newly arrived data segment which is placed based on the popularity of the channel and user experience. There is large number of replacement algorithms it mainly focuses on the head part of the cache. The old data are moved to the tail part of the cache which is not at all considered for further process. The proposed work focuses on the data segments which are reside on the tail part of the cache because these data segments will not be used for further processing. The assessment of popularity is based on the time in which the data segments are available on the cache. If the cache is updated frequently then the data are replaced with newly arrived data and also it will move into the tail part of the cache. The time taken for the particular data segment at the head part is measured and evaluated. Each peer has its own cache part in order to satisfy the neighboring peer's request. The main objective of the proposed work is to achieve the synchronization among the tail part (i.e., outliers) of the cache in order to keep the data segment available for further processing. The peer will select and update the data segments based on the novel selection and the updating algorithm.

METHODOLOGY

Conceptual diagram of the proposed model: Figure 1 shows that the overall concept of the proposed work. The peers are connected in a mesh based topology. Each peer will maintain the segment table which has the details about the tail part of the cache for accessing to/from the neighboring peers. There are various fields are available in the segment table which will be discussed in the following sections.

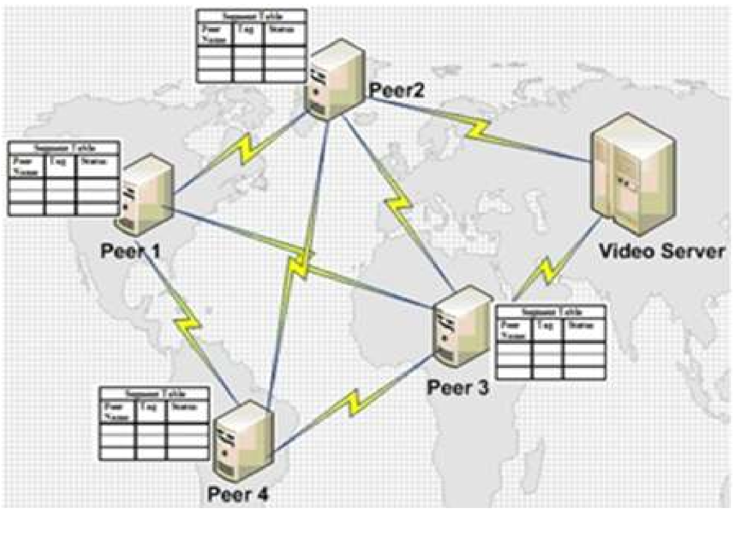

Fig. 1: Conceptual diagram of the proposed model

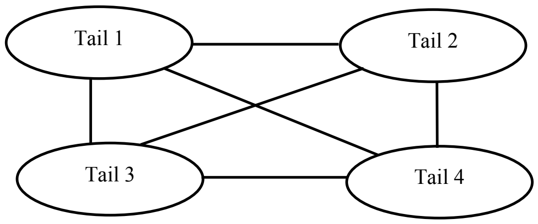

Fig. 2: Tag synchronization of the proposed model

Tag 1	Data piece 1	Tag 2	Data piece 2	…	…	Tag n	Data piece n

Fig. 3: Tail data format of the proposed model

ID	Time	Size	Data piece n

Fig. 4: Tag format of the proposed model

Peer Name	Tag	Status
Peer 1	Tag ID	1
Peer 2	Tag ID	0
Peer 3	Tag ID	0
…	…	…
Peer n	Tag ID	1

Fig. 5: Structure of the segment table

Tail synchronization: Figure 2 proposed that the logical view of tag synchronization on various peers of P2P computing. Each peer will maintain the cache in their own location which consists of two part namely head part, tail part, respectively. The head part contains the most popular data segment whereas the tail part holds the data segments which are less popular based on the accessibility. These data segments are also popular which are called outliers can be used for handling the peer's request. Suppose the requested data segment is not available in the tail part of any one peer it automatically finds it in the other peer which overcomes the unavailable problem. The tail part of the cache in the peers are synchronized each other in order to improve the performance of the live streaming.

Tail data format: Figure 3 shows that the tail data format of the peer's cache. The format consists of various tags along with the data pieces. The purpose of the tag is to identify the data segments at the tail part of the cache which will be discussed in the next section. The data pieces have the actual data segment of the live video. These data are moved from the head part of the cache to the tail part whenever the cache gets full. The tail part of the data is considered as least popular but practically the outliers have the most popular data segments. The reason is that the data segments (or data pieces) are maintained at the head part of the cache in some duration, so these data are again used for P2P live streaming. There are plenty of tags and data pieces are available in the tail data format which is used to eliminate the data unavailable issue in the P2P live streaming.

Tag format: Figure 4 Shows that the tag format of the cache at the peers in P2P computing. The tag consists of various parameters such as ID, time, size and data pieces of the live streaming. ID refers the identification of the data segment at the tail part of the cache. Time refers to the time duration of the movement from head part to the tail part, so the popularity of the data segments are based on the time taken for residing in the head part of the cache. Size refers to the actual size of the data segments for estimating the tail part of the cache.

Structure of the segment table: Figure 5 shows that the structure of the segment table. The purpose of the segment table is to maintain the details about the neighboring peers. The peer can download the data segments from other peer instead of using the actual video server. The segment table consists of peer name, tag ID and status of the data segments. The peer name refers to the name of the neighboring peers. The tag ID is used to identify the data segment at the neighboring peers. The status will indicate whether the data segments are available at the cache or not. If the status is 1 then the data segments are available at that peers so other peers may download the video segments form that intended peers. If the status is 0 then no data segments are available at the tail part of the peer's cache. This segment table will periodically updated for further reference. The data piece is maintained at the last portion of the tag format.

Algorithm of the proposed model:
Searching algorithm:

```
Algorithm Searching ()
Begin
        Number of peers in the networks P;
        Tail part of the cache $C_{Tail;}$
        Data piece in the tail part $D_{Tail;}$
        Segment table tag of data piece $S_{Tag;}$
        Tag for each data piece $T_{Tag}$;
        Level of cache L;
L1: for each peer € P do
      for each tailpart € CTail do
          for each tag € TTag do
            if (tag = = STag) then
              get the availability of the data pieces
              with the intended peer;
                if (Status = = 1) then
                  download the data piece from the
                  intended peer;
                else
                  Select the next available peer for
                  data piece;
                  goto L1;
```

```
                    end: if
                end: if
            end: for
        end: for
end: for
end: Searching
```

Updation algorithm:

```
Update_data_segment()
begin
head, tail, cache, cachelevel;
if (cache = = full) then
        head--;
        tail++;
        if (tail<tailsize) then
        Append the data segment into the cahce;
Add the tag, peer name, status in the segment table;
      else
        Increase the cache level i.e., cache level++;
        Append the data segment into the cache;
        Add the tag;
      end: if
    Update the tag at the segment table of the
                        highbouring peers;
    Update the time, size and status of the another peers;
else
      no need for updation;
end: if
end: Update_data_segment
```

ASSESSMENT OF THE TAIL BASED CACHING

Definition:
Let,

T is the tail part of the cache
P is the peer
NP is the neighboring peers
C is the cache at the peer
L is the level of the cache
DT is the data tag
DP is the data piece
S is the cache status
ST is the segment table
NP = {P1, P2, P3….Pn}
DP = {DP1, DP2, DP3….DPn}
S = {available, unavailable}

Lemma 1: Tail of the cache in one peer is synchronized with other peers:

$$\{T \cup C\} \rightarrow \{T \cup C \cup NP\} \quad (1)$$

Lemma 2: If the cache is full then the data part will be placed at tail part:

$$\{P \cup C\} \rightarrow \{DP \cup C \cup T\} \quad (2)$$

Lemma 3: If the tail is full, then increase the level of the tail:

$$\{P \cup C \cup T\} \rightarrow \{NP \cup C \cup T\} \quad (3)$$

Lemma 4: If the data segment in not available at the tail then the data is accessed from another peer:

$$\{P \cup C \cup T \cup \text{not}(DP)\} \rightarrow \{NP \cup C \cup T \cup Dp\} \quad (4)$$

Lemma 5: All neighboring peers have to update the availability in the segment table of other peers:

$$\{NP \cup DT \cup S\} \rightarrow \{NP \cup DT \cup S \cup ST\} \quad (5)$$

Time based analysis: It is used to specify the actual time of the data segment available at the cache. We can take the time as an important factor to analyze the popularity of the data segment. If the cache is full, then the data segment of the cache is replaced with the newly arrived data segments. The old data are placed at the tail part of the cache. Suppose the cache has been updating frequently then the data segments in the cache will also be replaced rapidly. The data segments which are replaced can take small amount of time in the cache. Suddenly it will move to the tail part, but these data segments also have some popularity. We believe that this analysis will help to improve the performance of the cache replacement.
Let,

T_{reside}	=	How long the data piece available in the head part.
T_{move}	=	The time taken for moving the data segment from head part of the cache to the tail part of the cache.
$T_{replace}$	=	The time taken to replace the new data segments with old segments.
δ	=	The maximum threshold of the time at the cache [i.e., head part]
τ	=	The popularity assessment:

$$\tau = \begin{cases} \text{Treside} < \delta, \text{ high channel popularity} \\ \text{Treside} > \delta, \text{ less channel popularity} \end{cases}$$

τ gets the value of the assessment it is inversely propositional to the time threshold δ. The time of the removed data segments is less, so it is always considered as more popular because the data reside in the cache is less:

$$\tau \propto \delta$$

ANALYSIS OF THE PROPOSED SYSTEM

Analysis of the cache with replacement: Figure 6 shows that the replacement of the newly arrived data segments in the cache. The data segment at the cache

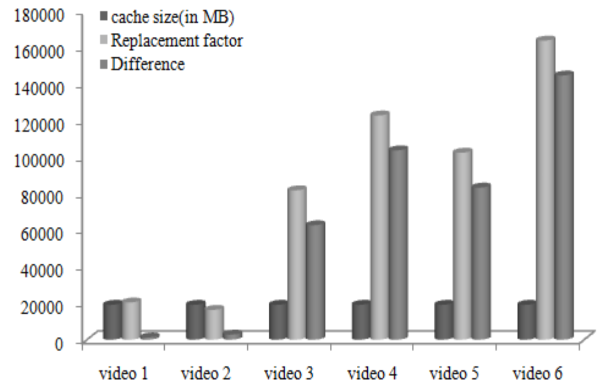

Fig. 6: Replacement of data segment in cache

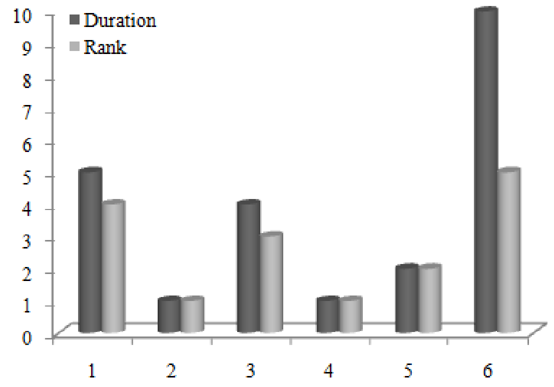

Fig. 7: Data segment with rank calculation

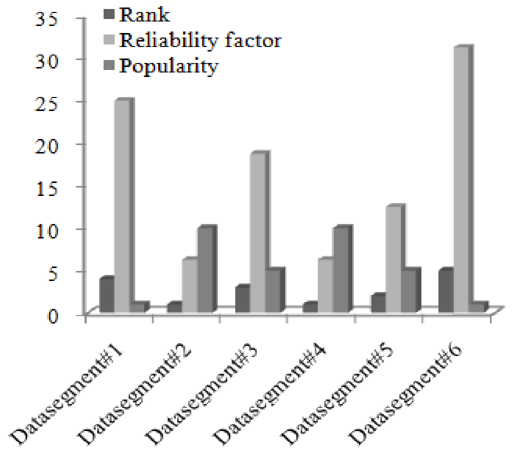

Fig. 8: Reliability factor of the data segments

has been replaced frequently, at the same time the removal of the data segment also done frequently. According to the analysis the free space of the cache is smaller than the newly arrived data segment; as a result a high difference will occur in the replacement factor. If the difference is high then the need for the replacement of the cache will also high i.e., the time taken for the data segment at the head part is less. These data segments are moved to the tail for further usage.

Analysis of the data segment with rank: Figure 7 shows that the rank based data replacement at the cache. The calculation of the rank is based on the duration i.e., the time difference between the start time and the end time of the data segment at the cache. If the duration is high then the rank value is also high. The highest rank value is assigned to low priority whereas the lower value will be assigned to higher priority. The following calculation is used to find the rank of the data segment in which it is replaced:

$$\text{Duration} = (\text{Start\_Time} - \text{End\_Time})$$

Analysis of the data segment with reliability: Figure 8 shows that the reliability of the data segment at the tail part of the cache. The reliability factor is calculated by using the following formula. The popularity of the data segment is measured based on the reliability factor and rank. The high priority of the rank gets higher popularity. These most popular data segments are used by the other neighboring peers. The data segment which got high rank with reliability factor provides more popularity. The scope of the proposed system is to use the data segment of the tail part which satisfies the peers request instead of removing the data segments. The popularity constant is used to identify the popularity of the data segment at the tail part of the cache:

$$\text{Popularity} = (\text{Rank}/\text{Popularity Constant}) * 100$$

CONCLUSION

Video on Demand is the main application in most area like education, business, entertainment etc. In live streaming the data are accessed from the video server by any client. The huge number of clients is accessing the server and suffers the overload problem. This problem is overcome by introducing the mesh based peer to peer network in which the peer can send the data piece to the other neighboring peers. The client first sends the request to the neighboring peers to download the available data segments otherwise it can download from the video server. The peer receives the data pieces from the server and places it into its own cache. Suppose the cache is full the cache segments are replaced using the replacement algorithm. During replacement the head part of the data pieces are replaced with newly arrived data pieces. The old data pieces are placed in to the tail part of the cache. The proposed work focuses on the popularity of the data pieces which are available in the tail part by using the time duration of each data piece. The analysis also taken place for improving the performance i.e., instead of removing the data pieces of the tail part we can keep it for further processing. In future the performance can be improved by implementing most efficient techniques.

REFERENCES

Chen, Y., E.L. Merrer, Z. Li, Y. Liu and G. Simon, 2012. OAZE: A network-friendly distributed zapping system for peer-to-peer IPTV. Comput. Netw., 56(1): 365-377.

Luo, J.G., Q. Zhang, Y. Tang and S.Q. Yang, 2009. A trace-driven approach to evaluate the scalability of P2P-based video-on-demand service. IEEE T. Parall. Distr., 20(1): 59-70.

Poon, S.M., S. Jie, B.S. Lee and C.K. Yeo, 2000. Performance of bu€er-based request-reply scheme for VoD streams over IP networks. Comput. Netw., 34(2): 229-240.

Seibert, J., X. Sun, C. Nita-Rotaru and S. Rao, 2011. A design for securing data delivery in mesh-based peer-to-peer streaming. Comput. Netw., 55: 2730-2745.

Wang, Y., T.Z.J. Fu and D.M. Chiu, 2011. Design and evaluation of load balancing algorithms in P2Pstreaming protocols. Comput. Netw., 55(18): 4043-4054.

Xie, M., G. Wei, Y. Ge and Y. Ling, 2012. Receiving-peer-driven multi-video-source scheduling algorithms in mobile P2P overlay networks. Comput. Electr. Eng., 38(1): 116-127.

Xu, K., M. Zhang, J. Liuc, Z. Qin and M. Ye, 2010. Proxy caching for peer-to-peer live streaming. Comput. Netw., 54(7): 1229-1241.