International Journal of Engineering Research-Online
A Peer Reviewed International Journal
Articles available online http://www.ijoer.in

Vol.3., Issue.2, 2015

RESEARCH ARTICLE

ISSN: 2321-7758

SCALABILITY AND BANDWIDTH EFFICIENCY OF VOD

J.MARY CATHERINE MONISHA[1], D.MEENA[2], M.SANGEETHA[3]

[1,2,] Students, [3] Assist.professor, Department of Computer Science and Engineering,
Panimalar Engineering College, Chennai.

Article Received: 16/03/2015 | Article Revised on:24/03/2015 | Article Accepted on:29/03/2015

J.MARY CATHERINE MONISHA

ABSTRACT

The video on demand systems are nowadays facing multiple numbers of challenges as it is having a tremendous growth with a large population of users. The main problems encountered are scalability and bandwidth efficiency. In this paper we propose a scalable bandwidth efficient and also a reliable VoD system which utilizes the patching and batching techniques along with frequent exchanging of state control messages among the servers. It is the idea where the servers are going to exchange control information about the bandwidth utilisation. It also covers the technique of prepopulation content placement where the contents of initial video segments (IVSs) are placed in a particular requesting peer on the basis of viewing pattern of previous peers. Using simulations we show that our proposed scheme effectively utilizes the server bandwidth and reduces the load on server. It also improves the response time and increasing the streaming rate. It mainly ensures the reliable transmission of video during failure of any server,

Key Words: batching, content distribution, load balancing, patching, prepopulation, gossiping, video on demand(VoD).

INTRODUCTION

The Video on Demand systems are now being used by large number of users and it is being estimated to be increasing in future. Thus the main issues to be considered are scalability and bandwidth efficiency. The very first used technique is unicasting where each user has to be allocated a separate independant channel from the start till the user continues watching the video. This method has a good user response but it is not efficient to handle large number of users. And also the server bandwidth resources are utilised to its maximum level when the number of users is increased linearly. Then came multicast methods where the clients are serviced as a group. The users who are requesting the same video are grouped together as a single batch and serviced.

There arrived two different multicasting techniques patching and batching. Batching is grouping together of set of clients having similar set of video requests. Patching is where latterly arriving peers will be joining the ongoing multicast stream and the initial video segments will be sent separately through a unicast stream. In the proposed paper we are going to join the two multicast techniques. The clients arriving earlier will be batched together to form a group and latterly arriving peers will join the ongoing batch. The contents of the initial video segments are placed in the clients based on the viewing pattern of the video

by previous peers. Once the peer has received the video segments it will be stored in the buffer and it will serve the requests of another peer. Thus peer by itself will act as server. By placing the initial video segment in the end users device the storage resources on the end user devices will be exploited well so as to reduce the startup delay. In this paper we ensure a reliable delivery of the video segments even when the server's bandwidth is totally occupied by the users by redirecting the requests to the nearby server. Here the servers will be exchanging the state control messages among themselves frequently. The state control messages will carry information about the idle and busy state of server. Here the server bandwidth is calculated the decision is made whether it can serve the request arriving from the nearby server. Hence our proposed work achieves a better performance by reducing the user delays.

EXISTING WORK

Most of the previous work mainly focuses on batching scheme where a batch of users will be serviced with a single multicast channel. The users arriving within some time with a same request will be batched. Hence the later arriving clients will have to wait until the start of next batch cycle. This time delay is the startup delay. To overcome this arrives another multicast technique called patching where the latterly arriving users within a particular threshold time will join the ongoing multicast scheme. Thus those peers will receive the video frames from the time they join the stream. Here the video segments are given to the late peers by the earlier peers. The video segments which are not received by the later peers will be sent by the server directly using a sophisticated unicast channel. Hence it helps in reducing the server stress. Thus it exploits the storage resource on the client side machines. When the server is too busy to serve a particular user request it will simply redirect the requests to the nearby server. If the nearby server is also busy then the redirection is failed it introduces additional delay in providing the service to the user and reliability of video delivery is breached when the video segments through the internet by servers which are geographically distributed from the peer.

PROPOSED WORK

In the proposed scheme both the multicast schemes batching and patching are combined to effectively utilize the server bandwidth. When a request for a video arrives and if there exists an ongoing multicast batch scheme it will be joined with that scheme. If there is no such scheme then it can be patched with already started batch. Not all the requests can be patched but the requests arriving within some threshold time only can be patched. Thus it reduces the servers load to a great extent. It also concentrates in placing the initial video segments in the client buffer earlier. The contents are placed based on the viewing pattern of video and the clients buffer capacity. In this paper the problem of redirection of server request to nearby server is considered. Here the servers will be exchanging the state control messages among themselves frequently and based on the information the redirection of the request will take place. Hence it reduces the delay and improves response time and ensures reliable video delivery.

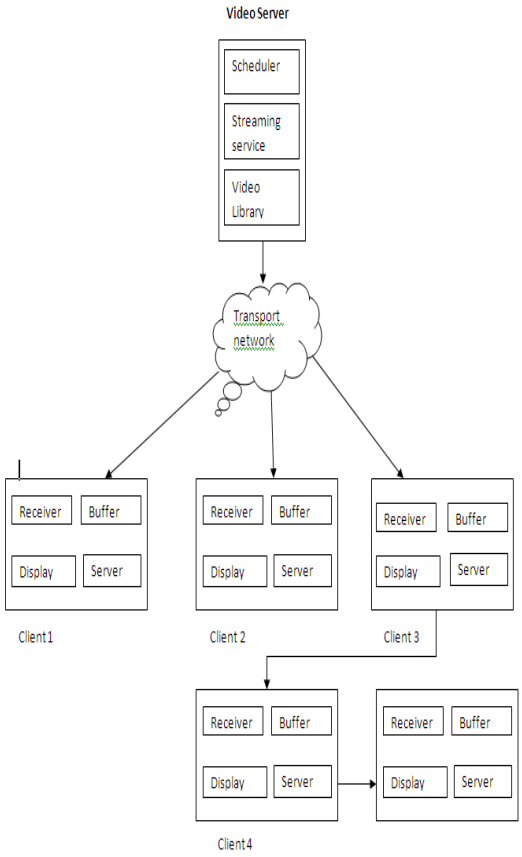
ARCHITECTURAL DESIGN

VOD,mainly focus on improving the client-server model. In the existing system video is divided into multiple segments and it is periodically broadcast to the clients through dedicated server channels. In order to provide uninterrupted streaming for all the users, the server keeps track of all the channels and ensures no interruptions exist between playing segments, resulting in heavy overhead. Further, as the requests randomly come to the server, the numbers of users in certain channels are limited.

In the proposed system, VOVO serves asynchronous peers with the combination of batching and patching. The server does not need to record any buffer information of peers. The VCR requests are resolved in a distributed manner. Moreover, VOVO employs a behavior-content-based scheme of prefetching.

METHODOLOGY DESCRIPTION

VOVO Server adopts the combination of prefetching and patching.Such a design is flexible with substantive concurrent and asynchronous clients.



PATCHING

Peers are clustered according to their arrival times and form sessions. Each session, together with the server, constructs an application multicast tree. When a peer joins in a session late and misses the initial part of the video, it picks up a few peers from the random list as patching sources and immediately starts to download the missing part from them. VOD divides the peers into generations according to their requests. Peers in VOD always cache the most recent content of a video. Only one stream from an early peer is needed to serve a late peer in VOD, while two streams, a patching, and a base stream are necessary for serving a late peer. Instead of deploying a patch server as failures and dynamics are handled locally in VOD, so that the server stress is further reduced.

Algorithm 1 GIP Algorithm

Input: T

Output: O1,O2, . . . ,OM

1: U ← {1, 2, . . . , M}

2: Ou = Ø, u = 1, 2, . . . , M // Set of all user nodes

3: Sort T in the decreasing order of ˜λi,ci,gi , grouping into videos where possible

4: for j = 1 to |T| do

5: Find node u ∈ U with minimum aggregated load,

which does not already contain a chunk identical to j.

6: Ou ← Ou + {chunk j}

7: if |Ou| = Su then

8: U = U − {u}

9: end if

10: end for.

The proposed GIP algorithm iteratively places chunks in local nodes. In each iteration, the remaining chunk with the largest arrival rate is placed in the user node which has minimum load, sufficient storage capacity and which does not contain the same chunk.

BATCHING AND PATCHING

The VOVO server uses batching to serve asynchronous peers. The server allocates a certain amount of dedicated outgoing bandwidth for each batching session. In each session, early joining peers directly become the children of the server. After the allocated bandwidth is fully occupied, late peers are redirected by the server and become the descendants of the early ones. Since the peers in a session transfer same video content currently broadcast by the server, the streaming mechanism inside a batching session is similar with P2P live streaming. Moreover, VOVO reinforces the batching scheme with patching. The server sends a list of randomly selected peers to each joining peer. When a peer joins in a session late and misses the initial part of the video, it picks up a few peers from the random list as patching sources and immediately starts to download. To support asynchronous accesses to the video content. When the server starts a new batching session, it need not wait any late peers. The early peers in a batching session obtain the video content and become the substitute video sources. Late peers can make up the missing content using patching from the early ones. Meanwhile, patching improves the system flexibility.

The following are the steps in multicast batching and patching:

Step-1: Start the process.

Step-2: Client sends video Request connection to the server.

Step-3: if there is a full multicast stream

Then server adds the request to the batch of requests available.

Step-4: if there is no full multicast stream but there is an ongoing stream

Then the client immediately joins the multicast stream.

Step-5: If there is a patch stream then the server joins the request of late client to batch of request served by patch scheme.

Step-6: If there is no patch scheme then server creates a patch and increases the length.

Step-7: the client then receives the video. **Step-8**: stop the process.

VOVO provides abundant backup stream sources for patching by adopting the hybrid caching strategy, where all the peers keep both the initial 5 minutes and the latest 5 minutes of the video played. Any late peer can instantly find patching sources and make up the missing part immediately after join, no matter if it starts from the beginning or any other offset of the video. Because the patching sources are selected randomly, load balance is kept among the peers.

GOSSIPING

Peers in VOVO conduct periodical gossips to exchange their state information. During each period, a peer generates a state message, including its latest state information. The format of the state message is {IP, Incremental playback record, Time stamp}, where IP is the peer's IP address, Incremental playback record refers to the string of segments the peer plays after it generates last state-message last time, Time stamp is the time since the peer joins in. On the other hand, each peer maintains a list of records. Each entry in the list corresponds to a peer and records its latest state. On receiving a state message, a peer performs relevant operations before it forwards the message to its neighbors:

If the time stamp of the state message is greater than that in the entry, the incremental playback record is inserted into the tail of the playback record in the entry, and the time stamp in the entry is updated.

Using the gossip-based state propagation, a peer is able to accumulate the information of playback history of all the peers. Furthermore, since the hybrid caching strategy is well known to all, through periodical gossips every peer can keep aware of the global distribution of video data on all the peers.

CONCLUSION

Our P2P-based retransmission scheme enables transferring the retransmission load, and some of the complexity with it, to the network edge where bandwidth is less scarce and costly. All receivers tuned in to the same video stream could organize themselves in overlay networks and retransmit lost packets to each other. Each receiver (retransmission peer) would classify other potential retransmission peers in the network on the basis of relevant performance metrics. Based on the performance we conclude that by the combination of batching and patching the server is able to serve many more concurrent clients than the original capacity of the source server. Based on the scheme of patching and the hybrid caching strategy, server provides abundant backup resource for asynchronous clients and frequent VCR requests. In this paper we proposed the new concept of using gossiping method which enables the server to respond the requests with short latencies.

REFERENCES

[1]. D.Eager, M. Vernon, and J. Zahorjan, "Minimizing bandwidth requirements for on-demand data delivery," IEEE Trans. Knowl. Data Eng.,vol. 13, no. 5, pp. 742–757, Sep. 2001.

[2]. Y.Cai, K. Hua, and K. Vu, "Optimizing patching performance," in Proc.Electron. Imag. Int. Soc. Optics Photon., 1998, pp. 204–215.

[3]. L. Gao and D. Towsley, "Threshold-based multicast for continuous media delivery," IEEE Trans. Multimedia, vol. 3, no. 4, pp. 405–414,Dec. 2001.

[4]. L. Gao and D. Towsley, "Supplying instantaneous video-on-demand services using controlled multicast," in Proc. IEEE ICMCS, vol. 2. 1999, pp. 117–121.

[5]. S. Ramesh, I. Rhee, and K. Guo,"Multicast with cache (Mcache):An adaptive zero-delay video-on-demand service," IEEE Trans. Circuits Syst. Video Technol., vol. 11, no. 3, pp. 440–456, Mar. 2001.

[6]. F. Liu, B. Li, and H. Jin, "Peer assisted on-demand streaming:Characterizing demands

and optimizing supplies," IEEE Trans. Comput., vol. 62,no. 99, p. 1, Feb. 2011.

[7]. W. Bing, S. Sen, M. Adler, and D. Towsley, "Optimal proxy cache allocation for efficient streaming media distribution," IEEE Trans. Multimedia, vol. 6, no. 2, pp. 366–374, Apr. 2004.

[8]. V. Gopalakrishnan, B. Bhattacharjee, K. K. Ramakrishnan, R. Jana, and D. Srivastava, "CPM: Adaptive video-on-demand with cooperative peer assists and multicast," in Proc. IEEE INFOCOM, 2009, pp. 91–99.

[9]. S. Kyoungwon, C. Diot, J. Kurose, L. Massoulie, C. Neumann,D. Towsley, et al., "Push-to-peer video-on-demand system: Design and evaluation," IEEE J. Sel. Areas Commun., vol. 25, no. 9, pp. 1706–1716,Dec. 2007.

[10]. J. Choi, A. Reaz, and B. Mukherjee, "A survey of user behavior in VoD service and bandwidth-saving multicast streaming schemes," Commun.Surveys Tuts., vol. 14, no. 1, pp. 156–169, Apr. 2012.

[11]. S. Ramesh, I. Rhee, and K. Guo, "Multicast with cache (Mcache):An adaptive zero-delay video-on-demand service," IEEE Trans. Circuits Syst. Video Technol., vol. 11, no. 3, pp. 440–456, Mar. 2001.

[12]. C.Jayasundara and V.Gopalakrishnan, "Facilitating multicast in VoD systems by content pre-placement and multistage batching," in Proc.COMSNETS'13. IEEE, Bangalore, India, 2013, pp. 1–10.

[13]. W. K. S. Tang, E. W. M. Wong, S. Chan, and K. T. Ko, "Optimal video placement scheme for batching VoD services," IEEE Trans. Broadcast.,vol. 50, no. 1, pp. 16–25, Mar. 2004.

[14]. J. Li-Shen and T. Li-Ming, "Harmonic broadcasting for video-ondemand service," IEEE Trans. Broadcast., vol. 43, no. 3, pp. 268–271,Sep. 1997.

[15]. F. Thouin and M. Coates, "Video-on-demand networks: Design approaches and future challenges," IEEE Netw., vol. 21, no. 2, pp. 42–48,Mar.–Apr. 2007.

[16]. L. Gao and D. Towsley, "Threshold-based multicast for continuous media delivery," IEEE Trans. Multimedia, vol. 3, no. 4, pp. 405–414,Dec. 2001.