SCALABILITY OF CLOSED-LOOP VIDEO DELIVERY SERVICE

Wei Shu and Min-You Wu
The University of New Mexico

ABSTRACT

Scalability is a key issue of on-line video service. Different methods have been proposed for the closed-loop video service, such as batching and patching. However, scalability itself has not been formally defined and the condition for a scalable system is not well studied yet. This paper studies scalability and the condition for scalable systems. It is found that when the arrival rate is across the threshold the system becomes scalable. This threshold value depends only on the number of videos, video length, and request distribution. We present an analysis for batching and patching. Comparison of batching, patching, and a new method, scheduled video delivery (SVD), is presented in this paper.

1. INTRODUCTION

Video-on-Demand (VoD) is gaining popularity in recent years with the proliferation of broadband networks. True VoD is still expensive since the server and the network deliver videos for individual requests. It is not scalable because of its high bandwidth requirement.

It is critical to investigate new methods for scalable video delivery. Batching [1, 2] and patching [3, 4] have been proposed to combine more requests to minimize the number of broadcast or multicast streams. Patching has been claimed as a scalable method. However, this is not always true since many requests cannot be combined. In fact, when the arrival rate of requests is less than a certain level, patching is not scalable. It is also true for the batching method. The scalability issue has not been well studied. Furthermore, the condition of a scalable video system is unknown. The relation between scalability and system parameters such as the number of videos, the video length, request distribution, and request arrival rate is not known either. The topic of this paper is to study effects of these parameters. The major finding is that there is a minimal request arrival rate for each method. Analysis shows that only when the arrival rate exceeds the threshold the system becomes scalable where the number of streams required does not increase or slowly increases with the number of requests. In this paper, the analysis for batching and patching is presented. The analysis shows that a system with a large number of videos is less scalable. A video repository of more than 1,000 video objects results in a non-scalable system in practical situation.

A new method, scheduled video delivery (SVD), is able to improve scalability. The performance comparison to SVD is also presented in this paper.

2. PARAMETERS IN A VIDEO DELIVERY SYSTEM

There are two components in a video delivery system, the stored videos and the requests to videos. Two parameters describe the property of videos: (i) the number of videos in the video repository (N); and (ii) the length of videos (L)

We assume all videos have the same length in this paper. Another parameter of videos is the constant-bit-rate (CBR) or variable-bit-rate (VBR) videos which has little influence to scalability and therefore is not considered in this study.

Two parameters describe the property of video requests: (i) request distribution; and (ii) request arrival rate (a) The probability that a request accesses a video j is $p_j = C/j^\alpha$, where

$$ C = \frac{1}{\sum_{j=1}^{N} j^\alpha} $$

This is called the Zipf distribution [5]. Video 1 is the most popular and video N is the least popular. The Zipf distribution can be generalized as

$$ p_j = \frac{C}{j^\alpha}, \text{ where } C = \frac{1}{\sum_{j=1}^{N} j^\alpha}. $$

Assume the request arrival rate is a, the arrival rate for video i can be computed by

$$ \lambda_j = ap_j = \frac{Ca}{j} \quad (1) $$

Most of this study assumes $\alpha = 1$. A sensitivity analysis of $\alpha$ will be given at the end of this paper. For simplicity, we assume that the request arrival times are evenly distributed. Using these four parameters to model a video delivery system, analysis can be conducted for various delivery methods.

3. SCALABILITY OF ON-DEMAND SERVICE

In this section, we analyze On-Demand Service. The analysis of batching [1, 2] and patching [3, 4] will be presented in the following sections.

The simple VoD serves each request individually. That is, a video stream is issued for each request. Thus, there are $aL$ requests arrived during the time period of a video length $L$ and the number of video streams issued is $aL$. Simple VoD is not scalable since the number of video streams required is proportional to the arrival rate.
With broadcast or multicast, the requests arrived at the same time can be combined and served together. This broadcast/multicast VoD is a special case of batching. Thus, when the batching time is very small, it can be considered as an immediate VoD service. For example, assume the video length is $L$ seconds and all requests arrived in one second are served by a single stream, the maximum number of simultaneous streams is $NL$ for $N$ videos. Once the arrival rate of each video is larger than 1 per second (0.0166 per minute), the VoD system becomes scalable. However, according to the Zipf distribution, the arrival rate of less popular video objects is very small. For example, assume $N = 1,000$ and $\alpha = 1$. Then $C = 0.1336$ and $\lambda_N = ap_N = Ca/N = 0.1336a/1000$. When $\lambda_N \geq 1$, at most one stream is issued per second for each video and the system becomes scalable:

$$a \geq \frac{1000}{0.1336} = 7,485.$$  

Thus, the system needs no more than $aL$ channels when the arrival rate reaches 7,485 per second, that is, 449,100 per minute or 26.9 millions per hour. Since the arrival rate can hardly reach this level, VoD is not considered as scalable.

Scalability of a closed-loop video service depends on the arrival rate $a$. When $a$ exceeds the threshold $\Theta$, the system is scalable. In this paper, the threshold values for batching and patching are derived, respectively.

**4. SCALABILITY OF BATCHING**

Batching combines the requests arrived in some time period $T$, say, 600 seconds, and serves them together by one stream. It is a simple method, but the maximum waiting time is $T$ and the average waiting time is $T/2$. At most $NL/T$ streams are required, independent of the number of requests.

In general, assume the batching time is $T_j$ for video object $O_j$, then $\lambda_jT_j$ requests can be combined. When $\lambda_jT_j < 1$, no request can be combined for video object $O_j$.

Assume that $\lambda_jT_j$ is monotonously decreased with $j$. It is true when all $T_j = T$ since $\lambda_j$ is monotonously decreased with $j$. Then the entire video repository can be divided into two sets by a value $v$. The first set, named as $S_1$, is from video object $O_1$ to video object $O_v$ where requests can be combined. The other set, named as $S_2$, is from video object $O_{v+1}$ to video object $O_N$ where requests cannot be combined. The value of $v$ can be obtained by

$$v = \text{Max}(j)|\lambda_jT_j > 1.$$  

Without losing generality, assume $\lambda_vT_v = 1$. Thus, from Equations (1),

$$\lambda_vT_v = \frac{C}{v}aT_v = 1 \quad \text{and} \quad v = CaT_v.$$  

Since $v \leq N$, $v = \text{Min}(CaT_j, N)$.

In order to understand importance of $v$, we define a parameter $\rho$ to measure what percentage of requests belongs to the first set of videos, $S_1$:

$$\rho = \frac{\sum_{j=1}^{v} \lambda_j}{\sum_{j=1}^{N} \lambda_j} = \frac{\frac{1}{\lambda_1} + \frac{1}{\lambda_2} + \ldots + \frac{1}{\lambda_v}}{\sum_{j=1}^{N} \frac{1}{\lambda_j}} = \sum_{j=1}^{v} \frac{C}{\lambda_j}.$$  

(2)

Let us find out requirements on the number of channels for the two sets of videos:

- **S_1.** For every video object $O_j \in S_1$ during batching time $T_j$, we can combine $\lambda_jT_j$ requests since $\lambda_jT_j > 1$. Therefore, for $O_j$, at most $L/T_j$ channels are required. The total number of channels required to serve video set $S_1$ is:

$$m_1 = \sum_{j=1}^{v} \frac{L}{T_j}.$$  

(3)

Notice that $m_1$ is independent of arrival rate.

- **S_2.** For every video object $O_j \in S_2$, since $\lambda_jT_j \leq 1$, during batching time $T_j$, no request can be combined. Therefore, the total number of channels required to serve video set $S_2$ is equal to the number of requests which belong to $S_2$

$$m_2 = (1 - \rho)aL.$$  

(4)

From Equations (3) and (4), the number of channels for request arrival rate $a$ is

$$m = m_1 + m_2 = \sum_{j=1}^{v} \frac{L}{T_j} + (1 - \rho)aL.$$  

(5)

When all $T_j = T$, Equation (5) can be simplified to

$$m = \sum_{j=1}^{v} \frac{L}{T} + (1 - \rho)aL = \frac{L}{T} + (1 - \rho)aL.$$  

(6)

The scalability of batching can be addressed based on the value of $\rho$.

When $CaT < N, \rho < 1$. From Equation (6)

$$m = CaT\frac{L}{T} + (1 - \rho)aL = (C + 1 - \rho)aL \geq CaL.$$  

The number of channels for the simple VoD is known as $aL$ and $C$ is a constant for a given value of $N$. Thus, the number of channels is proportional to the number of requests. Since $C$ is a constant which is usually larger than 0.1, improvement over the simple VoD is limited. The system is not scalable under this condition.

When $CaT \geq N, \rho = 1$. From Equation (6) $m = m_1 = NL/T$. In this case, the number of channels is independent of the number of requests. Therefore, when the arrival rate is higher than $\Theta_b$, the system is scalable to the number of requests.

The value of batching time $T$ is critical for the system performance since when $a$ is larger than or equal to $\frac{N}{\theta}$, the batching system becomes scalable. The longer the $T$, the early the system becomes scalable when the arrival rate $a$ increases. Note that $T$ can be larger then $L$ and the number of streams required can be smaller than $N$. On the other hand, the longer the $T$, the longer the user must wait.
5. SCALABILITY OF PATCHING

Patching combines requests with the patching stream. When a request misses the first part of a previous stream, it shares the rest of the stream and the server issues a patching stream to make up the request.

Patching is better than batching since it satisfies requests immediately. However, it is more complex and many versions of patching have been invented [6, 3, 4, 7]. The most efficient patching method is the recursive patching where the patching streams are merged recursively by “patching the patching stream” approach to minimize the bandwidth requirement [8, 9]. A common assumption for patching is the receive-two model where a client can receive at most two streams at any time. Although different versions of patching require different number of streams, the bandwidth requirement can be roughly modeled as \((\ln(\lambda_j L) + 1)\) streams for video object \(O_j\) [8, 9].

With this model, the number of streams required can be obtained as follows. When \(\lambda_j L < 1\), no request can be combined. Although many algorithms do not attempt merging with an existing stream that is already at least half over, it is possible that some requests can be combined partially when \(\lambda_j L > 1\). Thus, we divide the entire video repository into two sets according to \(\lambda_j L < 1\) and \(\lambda_j L \geq 1\). Thus, the value of \(v\) can be obtained as:

\[
\lambda_v = \frac{\sum_{j=1}^{v} a_j L}{v} = 1 \quad \text{and} \quad v = CaL.
\]

Also, because the upper limit of \(v\) is \(N\), we have

\[
v = \min(CaL, N).
\]

The number of streams required can be expressed as follow:

\[
m = \sum_{j=1}^{v} (\ln(\lambda_j L) + 1) + (1 - \rho) a_j L
\]

\[
= v + \ln((CaL)^v) \cdot \prod_{j=1}^{v} \frac{1}{j} + (1 - \rho) a_j L
\]

\[
= v + \ln((CaL)^v! + (1 - \rho) a_j L,
\]

where, \(\rho\) is from Equation (2).

When \(CaL < N\),

\[
m = CaL + \ln((CaL)^{CaL}) \cdot (CaL)! + (1 - \rho) a_j L \geq CaL.
\]

Thus, patching is not scalable when \(CaL \leq N\).

When \(CaL \geq N\),

\[
m = N + \ln((CaL)^N)
\]

The number of streams required increases with \(N \ln a\).

When \(CaL \geq N\), the entire video repository is in \(S_1\). The arrival rate \(a = \frac{N}{T}\) is marked as the threshold \(\Theta_p\). The system becomes scalable when arrival rate \(a\) exceeds the threshold \(\Theta_p\). Patching provides immediate response while the value of \(\Theta_p\) only depends on \(N\), \(C\), and \(L\). Longer \(L\) provides more chance to combine. However, when \(CaL\) is larger than \(N\), the number of channels required still slowly increases with \(a\).

6. COMPARISON STUDY

In this section, we provide the analysis results generated from the formulas derived above. Most results show the number of streams required when the arrival rate increases. Unless mentioned otherwise, the number of videos \(N = 1,000\). The video length \(L = 120\) minutes and \(T_d = 1,440\) minutes. The \(\alpha\) value is set to be 1.

A video delivery paradigm, Scheduled Video Delivery (SVD), has been proposed in [10]. In the SVD paradigm, users submit requests with specification of start time. A pricing scheme ensures that the user-specified start time reflects user’s real needs. The SVD system combines requests to form multicast groups and schedules these groups to meet the deadline. With this paradigm, requests can be combined to reduce the server load and network traffic. Furthermore, the traffic can be smoothed by shifting the peak-time traffic to non-peak time. SVD performs similar to batching, but provides a longer equivalent batching time \(T\) when \(a\) is small. Thus, the system becomes scalable early when \(a\) increases. Better than batching, the system can provide immediate response while it encourages users planning ahead. Its drawback is the same as patching, that is, when the system is scalable, the number of streams required still slowly increases. It can be improved by combining patching and SVD. SVD can combine with patching for a better performance. Due to space limitation, the analysis of SVD is not shown in this paper. A complete analysis can be found in [11].

Figure 1 shows the relation between the threshold of \(\Theta\) and the number of videos for batching, patching, and SVD. The threshold \(\Theta_p\) for patching is smaller than that of batching \(\Theta_b\). SVD becomes scalable much earlier than patching.

Figure 2 shows the performance of batching. Obviously, the longer the batching time, the better the performance. For very short batching time such as one second \((T = 0.0166)\), there is virtually no waiting time. Therefore, it is equivalent to the true VoD. However, its scalability is poor since the arrival rate cannot reach 450,000 per minute in prac-
Tuesday. Longer batching time results in good performance. For example, when $T = 20$ minutes, the batching system becomes scalable when the arrival rate reaches 375 per minute or 22,500 per hour.

We now compare the performance of different methods in Figure 3. Here, the batching time $T$ is set to be 20 minutes where batching is comparable to other methods. Batching becomes scalable when $a = 375$. After that the number of streams remains 6,000. The value of $\Theta_p$ of patching is $a = 63$ and that of SVD is only $a = 7$. However, after these points the number of streams slowly increases. SVD with patching shows a much better performance. Its value of $\Theta$ is the same as SVD without patching but the number of streams increases much slower. For patching, when the arrival rate reaches 63 per minute or 3,780 per hour, 480 streams are required. When arrive rate reaches 1000 per minute or 60,000 per hour, 4,766 streams are required. SVD with patching shows good scalability. When arrive rate is 7 per minute or 420 per hour, only 157 streams are required. And when arrive rate reaches 1000 per minute or 60,000 per hour, 956 streams are required. Thus, SVD with patching is a more scalable method.

7. CONCLUSION

Scalability of closed-loop on-demand video service has been studied in this paper. Performance models of batching and patching are presented. Analysis based on these models shows when a system can be scalable. The value of $\Theta$ depends on the size of video repository, video length or batching time, and request distribution. The major conclusion is that the current methods are not scalable for a large repository of videos. They are scalable for only a small repository of videos. SVD with patching performs the best among these methods. It scales to a quite large number of videos.

8. REFERENCES