British Journal of Mathematics & Computer Science
3(4): 811-821, 2013

SCIENCEDOMAIN *international*
www.sciencedomain.org

Estimation of the Concurrent Capacity of a Streaming Media Server Based on User Behavior Analysis

Qiang Ling[1*], Lixiang Xu[1], Yicheng Zhang[1] and Jinfeng Yan[1]

[1]Department of Automation, University of Science and Technology of China, Hefei 230027, China.

Research Article

Received: 05th June 2013
Accepted: 21st August 2013
Published: 28th August 2013

Abstract

With the fast development of computer and network technology, streaming media has attracted more and more attention. The concurrent capacity is a major performance index, especially for media service providers. In the current literature, the concurrent capacity of a server is usually determined through experiments, which can only be done after building a server and are time-consuming. This paper proposes a method to estimate the concurrent capacity just with the configuration parameters of a server. Due to the fast CPU and high-speed network cards, the bottleneck of the concurrent capacity is the I/O speed, which is determined by both the fast memory and low-speed hard disks in a server. By analyzing the behavior of users, we estimate an upper bound on the percentage of data supplied by the memory, named byte-hit-ratio, under any realistic scheduling policy between the memory and disk for a given memory capacity. Based on the byte-hit-ratio bound, we can obtain an upper bound on the average I/O speed of the server, which is proportional to its concurrent capacity. Our method does not require any actual tests and can guide the design of streaming media servers.

Keywords: Streaming media, concurrent capacity, behavior analysis, byte-hit-ratio.

1 Introduction

With the fast development of computer and network technology, the Internet has found more and more applications. As a synthetic combination of text, sound and image, multimedia has attracted more and more attention. Streaming media is a popular pattern of online multimedia service. The concurrent capacity of a streaming media server is a major performance index, especially for media service providers. Actually commercial servers are distinguished by the maximum number of concurrent streams supported by the server without degrading the quality of streams.

A server's concurrent capacity depends on its CPU, network cards and the I/O speed [1]. Modern servers generally have powerful processing ability by employing multi core processor and CPU may not possibly be the bottleneck resource. Gigabit network cards make high bandwidth

**Corresponding author: qling@ustc.edu.cn;*

available for all the users and may not limit the concurrent capacity. The I/O speed of a server is determined by both memory and hard disks. The modern memory, like DDR3 memory, exhibits powerful ability in its accessing speed while the disk still suffer the low speed and limits the server's concurrent capacity. The experiments in [1] also show that the server performance is disk-bound when users request for different files.

To improve the overall I/O speed, modern computer systems usually employ a special storage architecture, called memory hierarchy [2]. The most requested objects are stored in the memory to serve most of the user requests. In order to estimate the overall I/O speed, we have to figure out which parts of all the requested objects should stay in memory. Lots of scheduling policies have been proposed to choose the best objects for the memory [3]. Then another question comes out, which policy is the best? Furthermore, what is the byte-hit-ratio of the best policy, which is defined as the percentage of the data supplied by the memory? Of course, different scheduling policies may yield different byte-hit-ratio.

Here we consider a *"genie"* scheduling policy, which knows the user behavior ahead. The "genie" scheduling policy can surely achieve the highest byte-hit-ratio, which is actually an upper bound on the byte-hit-ratios of all realistic scheduling policies. This upper bound gives us the highest percentage of data supplied by the memory and yields an upper bound on the overall average I/O speed. Then by dividing the overall average I/O speed bound with the users' average encoding bit rate, we can get an upper bound on the number of concurrent users which a server can support without degrading the quality of streams. The advantage of our method is that it needs only the hardware configuration of a server and does not require building the server ahead and do any real experiments. To the best of our knowledge, all the methods to evaluate the concurrent capacity of a server are based on tests and there is no method being similar to ours in the literature.

The rest of this paper is organized as follows. Section 2 introduces the related work. Section 3 presents the details of our method of estimating the concurrent capacity of a streaming media server based on the user behavior analysis. Some experimental results are shown in Section 4. Finally Section 5 summarizes this paper.

2 Related Work

In the current literature, the concurrent capacity of a server is usually measured by running a sequence of tests with an increasing number of clients requesting files from the server at certain encoding bit rates [1]. In [1], a set of *capacity metrics* and *basic benchmarks* are introduced to determine the performance limits of media servers and analyze the main performance bottleneck. Two sets of benchmarks, namely *Single File Benchmark* and *Unique Files Benchmark*, are tested to see how many concurrent streams of the same bit rate can be supported by a media server without degrading the quality of any stream. The *Single File Benchmark* measures a media server's concurrent capacity when all the clients in the test are accessing the same file from the memory while the *Unique Files Benchmark* measures that when each client in the test is accessing a unique file from the hard disk.

Fig. 1 shows the results of the tests. From that figure we can see that the *Single File Benchmark* yields about 2.5-3 times more concurrent streams than the *Unique Files Benchmark*. It was pointed out that the bottleneck resource of the server is CPU under the *Single File Benchmark* and the I/O speed of the hard disk under the *Unique Files Benchmark*. It shows the server's concurrent

capacity under two extreme accessing patterns, either all requests are served from memory or all requests are served from the hard disk. In reality, some popular objects are stored in the memory and can be supplied at a high I/O speed while the other ones in the hard disk and at a low I/O speed. So the real number of concurrent streams the server can support lies between the two curves in Fig. 1. As we see in Fig. 1, the bounds of the two curves are quite loose to precisely predict the concurrent capacity of a server.

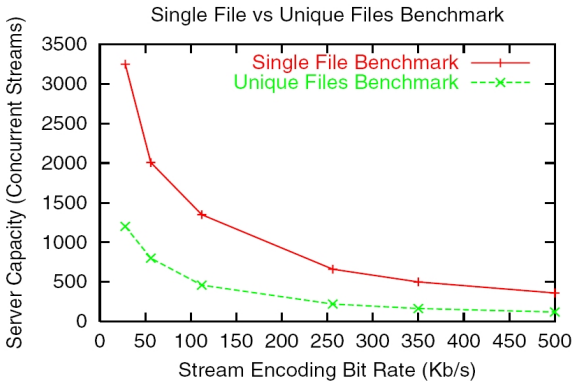

Fig. 1. Performance of single file benchmark and unique files benchmark [1]

An improvement to the loose bounds in [1] was done in [4]. Instead of the two extreme cases in [1], the accessed files could stay either in the memory or in the hard disk as shown in [4]. A LRU (Least Recently Used) scheduling policy is adopted to distribute files between the memory and the hard disk. Due to LRU, some popular files are stored in the memory while the other less popular ones stay in the hard disk. Most of the requested bytes can be streamed from the memory and yields high I/O speed. Compared with the case when all requests are served from the hard disk, the server's concurrent capacity increases significantly by about 2 times as shown by the testing results in Fig. 2. These results mean that the more bytes are streamed from the memory, the more concurrent users can the server support. Although this new method gives us more practical evaluation regarding the concurrent capacity of a server, it raises two problems,

1. LRU is a very simple scheduling policy. When a better scheduling policy is taken, we expect an improvement in the server's concurrent capacity. What is the “highest” concurrent capacity under the “best” policy?
2. The method in [4] still relies upon a long test, which is time-consuming.

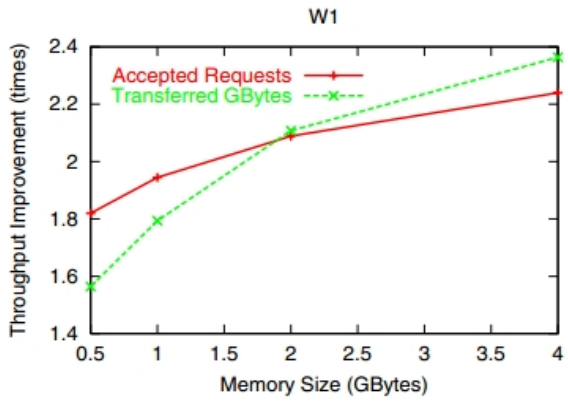

Fig. 2. Throughout improvements under the LRU scheduling policy compared to Unique files benchmark [4]

This paper finds out that the concurrent capacity of a server is determined by not only the scheduling policy but also the user behavior. It assumes a "genie" scheduling policy, which knows the user behavior ahead. The concurrent capacity under that "genie" policy actually places an upper bound on the ones of all policies. Moreover, our method does not require any real test and can provide guideline to the server configuration in advance.

3 Estimation of the Concurrent Capacity of a Streaming Media Server

Despite the diverse types of streaming media users, some general properties are found in the user behavior. This guides researchers to model the user behavior with appropriate mathematical techniques. Various models are proposed to describe the general properties of users and accessed files. Based on these models, we find a theoretical way to evaluate the server's concurrent capacity.

3.1 User Behavior

Studies on web workload have identified strong locality in user accesses. In [5], *static locality* is observed that 10% of the files accessed on the server typically account for 90% of the server's requests and 90% of the bytes transferred. Cherkasova obtained a similar conclusion for a media server system [6] where 90% of the media sessions target 14% of the files. Lots of models have been proposed to describe the locality. Glassman used *Zipf's* law to model the distribution of web page requests where the probability of a request for the $i-th$ most popular page is proportional to $\frac{1}{i}$ [7]. In later study [8], researchers, however, found that the distribution of page requests generally follows a *Zipf-like* distribution where the probability of a request for the *i-th* most popular page is proportional to $\frac{1}{i^{\alpha}}$. In [9], authors approximated the educational media server daily workloads using the concatenation of two *Zipf-like* distributions. These models may provide the required visiting probabilities of videos for our subsequent analysis.

To study the locality, we consider a working set with N different videos whose access probability is $p_1, p_2, \cdots, p_N$. All videos are divided into M equal small blocks and are assumed to share the same fixed internal popularity, i.e., the $j-th$ blocks of all videos have the same internal probability, which is denoted as $q_j (j = 1, \cdots, M)$. This assumption is based on the analysis of videos' internal popularity in [10] and is confirmed in Section 4. Thus the access probability of block j inside video i, p_{ij}, is determined by two factors, p_i and q_j, as shown in eq. (1).

$$p_{ij} = p_i * q_j \tag{1}$$

The access probabilities of all the NM blocks are shown in Table 1. Rearrange p_{ij} in Table 1 in an descending order as $t_1, t_2, \cdots, t_{NM}$, i.e., t_k stands for the $k-th$ largest one among p_{ij}.

Table 1. Access probability of different blocks

	Block 1	**Block 2**	⋯	**Block M-1**	**Block M**
Video 1	$p_1 * q_1$	$p_1 * q_2$	⋯	$p_1 * q_{M-1}$	$p_1 * q_M$
Video 2	$p_2 * q_1$	$p_2 * q_2$	⋯	$p_2 * q_{M-1}$	$p_2 * q_M$
⋮	⋮	⋮	⋱	⋮	⋮
Video N-1	$p_{N-1} * q_1$	$p_{N-1} * q_2$	⋯	$p_{N-1} * q_{M-1}$	$p_{N-1} * q_M$
Video N	$p_N * q_1$	$p_N * q_2$	⋯	$p_N * q_{M-1}$	$p_N * q_M$

3.2 An Upper Bound on the Byte-Hit-Ratio of Realistic Scheduling Policies

A streaming media server usually has both memory and hard disks. The requested files may be stored in the memory or in the hard disks. Due to the significant I/O speed difference between the memory and the hard disks, the more bytes are streamed from the memory, the higher average I/O speed and the higher concurrent capacity can be expected. We measure the percentage of data supplied by the memory with *byte-hit-ratio* H_m', which is defined as follows.

$$H_m' = \frac{D_{Memory}}{D_{Total}}, \tag{2}$$

where D_{Memory} is the bytes streamed from the memory and D_{Tolal} is the total requested bytes of all users. So a higher H_m' makes the server perform better.

A scheduling policy will always attempt to select the most popular video blocks to stay in the memory. In reality, the scheduling policy can only observe users' requests, from which it tries to

estimate the probabilities $t_k (k = 1, 2, \cdots, NM)$ of video blocks. Due to randomness of user requests, the estimation of t_k cannot be perfect. Here we assume a *genie* policy, which can exactly know all t_k. The *genie* policy has more information than other policies and its byte-hit-ratio is actually an upper bound on the byte-hit-ratio of other realistic polices. In order to achieve the highest $H_m^{'}$, the *genie* policy surely fills the memory with the video blocks with the highest probabilities. Suppose the memory can hold at most K video blocks, then the byte-hit-ratio of the *genie* policy, or the upper bound on the average byte-hit-ratio of all realistic policies, is

$$H_m = \sum_{i=1}^{K} t_i \tag{3}$$

Note that $t_1, t_{2,\cdots,} t_K$ is obtained by rearranging p_{ij} in Table 1 in the descending order.

3.3 Evaluation of the Concurrent Capacity of a Streaming Media Server

The upper bound on the *byte-hit-ratio* of the memory with a realistic scheduling policy determines the percentage of the data streamed from memory. Based on that, we can obtain an upper bound on the overall average I/O speed, r_u, of the server like eq. (4).

$$r_u = r_M * H_m + r_D * (1 - H_m) \tag{4}$$

where r_M is the I/O speed of the memory and r_D is the I/O speed of the hard disk. Now we assume that all streams have the same average bit rate r_0. Then we can get an upper bound N_u on the number of concurrent streams of a server without degrading the quality of streams. Note that when we say a server can support a certain number of concurrent streams without degrading the quality, we mean that the server can serve all the concurrent users with satisfied quality for a long period of time.

$$N_u = \frac{r_u}{r_0} \tag{5}$$

4 Experimental Results and Analysis

We get a batch of data with 2 million requests from a campus media server in University of Science and Technology of China. There are 1000 different videos ($N = 1000$), whose probabilities are $p_1, p_2, \cdots, p_N$ and shown in Fig. 3. Note that p_i is obtained by counting the number of visits of the $i-th$ most popular video. In Fig. 3, the curve of the visiting probability of videos fits a straight line well except for the beginning part. We can describe the curve with a Zipf-like distribution model and the value of parameter α is 1.03.

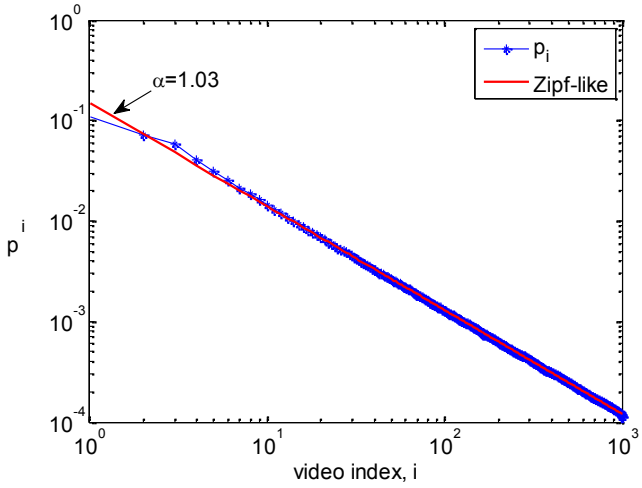

Fig. 3. The visiting probabilities of videos

Here all videos are assumed to have the same length of 2GB and the same fixed internal popularity. To check the validity of the assumption about videos' internal popularity, we divide the videos into two parts, including the most popular 20% videos and the less popular 80% videos, and compute their visiting probabilities, respectively. Furthermore all videos are divided into 2000 ($M = 2000$) equal small blocks, whose visiting probabilities, $q_1, q_2, \cdots, q_M$, are shown in Fig. 4. q_j is similarly obtained by counting the number of visits of the $j-th$ block. We can see from Fig. 4 that after normalization, the curves of the most popular 20% videos and the less popular 80% videos almost have the same shape. These conclusions provide a support for our assumption that all videos share a fixed internal popularity.

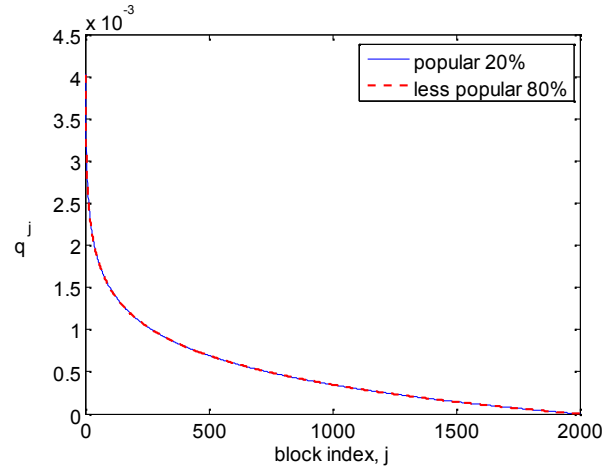

Fig. 4. The visiting probabilities of blocks

Thus the access probability of block j inside video i, p_{ij}, can be computed according to eq. (1) and Table 1 can be formed. As mentioned in Subsection 3.2, we rearrange p_{ij} as $t_1, t_2, \cdots, t_{NM}$ in a descending order.

Suppose the memory size is n GB. The number of blocks inside the memory is

$$N_m(n) = \frac{n * M}{L} \tag{6}$$

where L represents the video size. Here $L = 2GB$. The "genie" scheduling policy will store the most popular $N_m(n)$ blocks in the memory. According to the analysis in Subsection 3.2, we can compute the upper bounds H_m on the byte-hit-ratios under different memory sizes, which are shown in Fig. 5. Considering that the memory size of a server is usually not too big, we only give the results when memory size increases from 0 to 20 GB. Note that only the most popular 0.5% blocks of all the video blocks are stored when $n = 20GB$. The curve in Fig. 5 seems to be linear as the memory size is small relative to the whole videos.

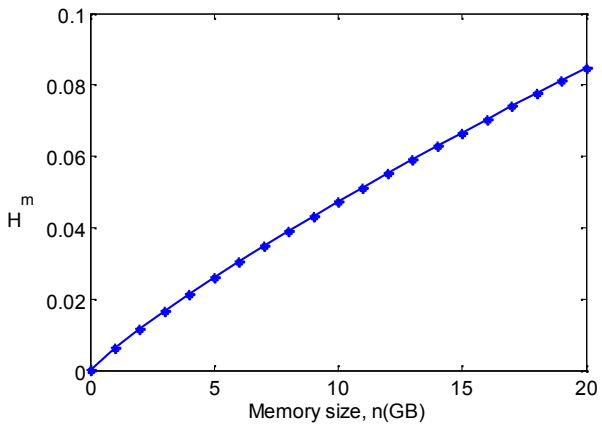

Fig. 5. The upper bound on the byte-hit-ratio of the memory

The I/O speeds of the memory and the hard disks are set at $r_M = 8GB/s$ and $r_D = 67MB/s$. The average bit rate is set at $r_0 = 250KB/s$. According to the method in Subsection 3.3, we can compute the upper bounds N_u on the number of concurrent streams under different memory size n, which are shown in Fig. 6. When the memory is large enough to store all the requested videos, it turns into the case similar to the *Single File Benchmark* in Section 2. Because memory is always more expensive than hard disks, media service producers would like to use as least memory as possible to satisfy certain users' needs. Our method will help to decide the minimum memory size required to support a certain amount of concurrent users.

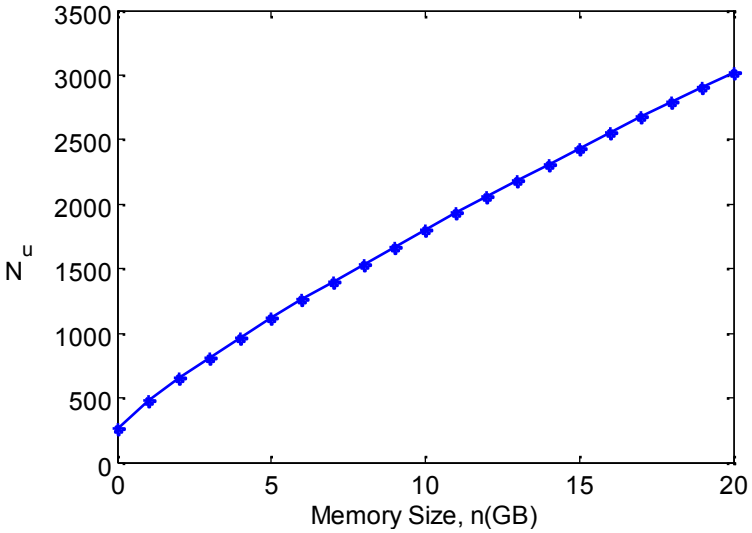

Fig. 6. The upper bound on the number of concurrent streams of a server

Additionally, we find that the increasing speed of the number of concurrent users fall behind that of the memory size. To be more precise, we compute the increase of N_u, ΔN_u, when n increases by 1GB. As shown in Fig. 7, the concurrent performance benefit is slowing down as the memory size increases. The reason is that the "genie" policy always fills the memory with the most popular blocks. When the memory size increases, the newly added blocks are less popular than the ones previously placed in the memory, which explains that the curve in Fig. 5 is not strictly linear.

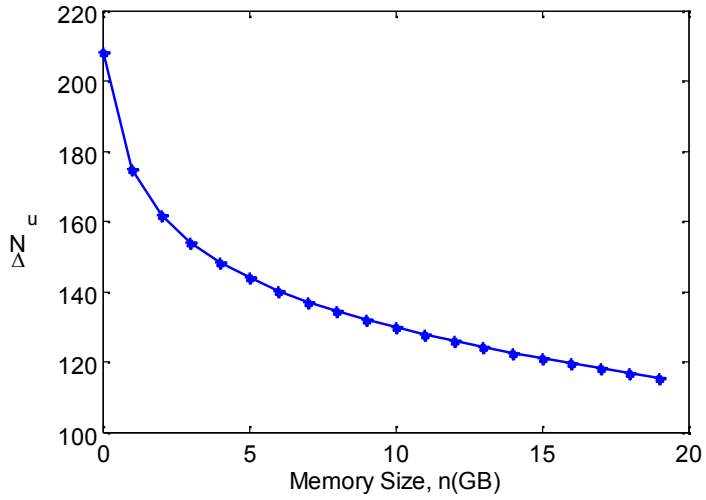

Fig. 7. Concurrent performance benefits under different memory sizes

5 Conclusion

The concurrent capacity of modern media servers is limited by the I/O speed, particularly the low I/O speed of hard disks. This paper provides a method to place an upper bound on the number of concurrent streams of a server under given user behavior. It takes a "genie" policy to schedule data between the memory and the hard disks, which knows the user behavior in advance, makes the best scheduling decision under the given user requests and can place an upper bound on the byte-hit-ratio of the memory for all scheduling policies. This byte-hit-ratio bound is transformed into an upper bound on the overall average I/O speed, and an upper bound on the number of concurrent streams. The major advantage of our method is that no real test is needed. In summary, our method can theoretically estimate the concurrent capacity of a streaming media server only with its configuration, even before it is really built.

Acknowledgment

This work is partially supported by National Natural Science Foundation of China under Grant 61273112 and the fund from Young Innovation Promotion Association of CAS.

Competing Interests

Authors have declared that no competing interests exist.

References

[1] Cherkasova L, Staley. Building a performance model of streaming media applications in utility data center environment. In: Proceedings of the ACM/IEEE Symposium on Cluster Computing and the Grid (CCGrid), Tokyo, Japan. 2003;52-59.

[2] Mellor-Crummey J, Whalley D, Kennedy, K. Improving memory hierarchy performance for irregular applications. In: Proceedings of the 13th International Conference on Supercomputing, New York, U.S.A. 1999;425-433.

[3] Liu J, Xu J. Proxy Caching for Media Streaming Over the Internet. IEEE Communication Magazine. 2004;42(8):88-94.

[4] Cherkasova L, Tang W, Vahdat A. A unified benchmarking and model-based framework for building QoS-aware streaming media services. Multimedia Systems. 2006;11(6):532-549.

[5] Arlitt M, Williamson C. Web server workload characterization: the search for invariants. In: Proceedings of ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, Philadelphia, U.S.A. 1996;126-137.

[6] Cherkasova L, Gupta M. Analysis of enterprise media server workloads: access patterns, locality, content evolution and rates of change. IEEE/ACM Transactions on Networking. 2004;12(5):781-794.

[7] Glassman S. A caching relay for the World Wide Web. Computer Networks and ISDN Systems. 1994;27(2):165--173.

[8] Breslau L, Cao P, Fan L, Phillips G, Shenker S. Web Caching and Zipf-like Distributions: Evidence and Implications. In: Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies, New York, U.S.A. 1999;126-134.

[9] Almeida JM, Krueger J, Eager DL, Vernon MK. Analysis of educational media server workloads. In: Proceedings of the 11th Int. Workshop Network Operating System Support for Digital Audio Video (NOSSDAV 2001), New York, U.S.A. 2001;21-30.

[10] Jiang Yu, Chun Tung Chou, Xu Du, Tai Wang. Internal popularity of streaming video and its implication on caching. In: Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA'06); 2006.

Peer-review history:
The peer review history for this paper can be accessed here:
http://www.sciencedomain.org/review-history.php?iid=240&id=6&aid=1954