DYNAMIC MEMORY ALLOCATION AND DATA SHARING SCHEDULE IN MEDIA SERVER

Kaihui Li[1,2], Yuanhai Zhang[1,2], Jin Xu[1,2], Changqiao Xu[1,2]
[1] Institute of Software, The Chinese Academy of Sciences, Beijing, P.R. CHINA 100080
[2] Graduate University of the Chinese Academy of Sciences, Beijing, P.R. CHINA 100039
kaihui01@ios.cn, yuanhai02@ios.cn, xujin03@ios.cn, changqiao@ios.cn

Abstract—Most of the existing buffer allocation and sharing schemes can not make full use of the system resources while being implemented independently. Here, based on analysis of the existing schemes, we propose a new algorithm, Balanced Buffer Sharing of Limited Resource (BBSLR), which allocates resources for each request according to the available cache and disk bandwidth, and adjusts buffer size according to the current request distribution and available resources dynamically. It does a good job of managing resources to maximize the number of simultaneous clients and enhance start-up delay. With BBSLR, the resources consumption will be balanced by rational allocation of the available resources, the average start-up delay will be reduced by caching the data at start and more clients will be served consequently. These conclusions are proved by comparing with several existing methods experimentally.

1. INTRODUCTION

The growing popularity of streaming media places an increasing strain on both network and server resources. Streaming media requires large amounts of network and disk bandwidth to successfully transmit streams from a server or proxy to the client. The most common solution is to simply purchase more resources. However, it doesn't make sense always for technical or economic reasons. So some algorithms proposed can make full use of the server resources by substituting one set of resources for another. These algorithms include static buffer allocation scheme [1], dynamic buffer allocation scheme [2], buffer sharing, prefetching [3], batching [4], etc. Each of them is explicitly designed for a particular problem and is not generalized to generic resources types. On the other hand, these algorithms never take the distribution of the requests into consideration while allocating resources.

In this paper, we propose a dynamic buffer allocation and sharing schedule based on these existing solutions. For the server configured with nowadays probability resources, this algorithm allocates resources for a new request according to available cache and disk bandwidth and adjusts buffer at run-time according to current request distributing and available resources dynamically. With this algorithm, the resources consumption will be more balanced and the video server will simultaneously serve more clients without any additional resources.

2. RELATED WORK

The static buffer allocation scheme determines the minimum buffer size based on the assumption that the system is in the fully loaded state. It can not use memory efficiently by allocating a larger buffer than necessary when the system is not in the fully loaded state [2]. The dynamic buffer allocation scheme, that mends it by computing the buffer size at run-time, is more adaptive to these applications similar to VOD.

Buffer sharing uses a single disk stream to serve multiple clients by caching data in memory. It can reduce disk stream requirement, enhance system performance and reduce start-up delay because of the faster read speed of the memory also [5]. Buffer sharing methods include frequency caching [6], fixed-sized caching [7] and interval caching (IC) [8] and so on. IC method is relatively effective [9]. It works as follows: if two client requests for the same file arrive close together in time, the data delivered to the first client can be retained in memory until it is delivered to the second client. In this method, we need to determine the size of interval buffer and the times of allocating buffer and remaining data. Typical algorithms include Generalized Interval Caching (GIC) [8], Non Preemptive Interval Caching(NIC), Controlled Buffer Sharing (CBS) [10], and Multi Policy Integrated Cache （MUPIC） [11] and so on. This section mainly introduces CBS and MUPIC.

CBS [10] presents a concept of distance threshold (denoted d_t), and defines $I_\$/M_\$$ (when $I_\$/M_\$$ is an integer), $\lfloor I_\$/M_\$ \rfloor$ or $\lceil I_\$/M_\$ \rceil$ (when $I_\$/M_\$$ is not an integer) as the optimal d_t. $M_\$$ and $I_\$$ separately denote the price of memory and disk bandwidth needed by a independent request. Distance threshold d_t is the number of buffer blocks which can be used to cache the sharing data. If the needed buffer blocks between two adjacent displays referencing the same video exceed d_t, they are allowed to neither pin their intermediate data pages nor share one disk stream using memory. This algorithm computes the amount of required buffer and disk bandwidth according to the input parameters of request arrival rate, access distribution, d_t and so on at system design time, then configures the system with

the computed results. At system run time, the data will not be cached in the buffer until the new request arrives, whose distance does not exceed d_t with the former request. So, there will be a transition state where the new request would need to be served with the disk while the corresponding interval is being cached.

If we apply CBS to the practice, it may result in that the requirements of the resources computed can not be satisfied for technical or other reasons. On the other hand, since it does not cache the data used by the former request in the buffer before the new request which can share the data with the former one arrives, the new request needs to be served by a new disk stream for an interval time. Relative to cache the data when serving the former request, this way needs to expend more resource (disk bandwidth) and can not also lessen start-up delay.

MUPIC [11] mainly depends on comparison of the ratio of cache usage (M_i/M) and the ratio of disk bandwidth usage （B_i/B） to determine whether or not cache data in the buffer. Here, M denotes the size of the whole memory. M_i denotes the size of the needed buffer when a new request is served with the buffer. B and B_i separately denote the bandwidth of the whole disk and the required bandwidth of each stream served by a disk stream. When the ratio of cache (disk bandwidth) usage is lower than the ratio of disk bandwidth (cache) usage, it will use the buffer (disk bandwidth) to serve requests. When the two ratios are equal, it does not matter if the stream uses disk bandwidth or buffer. Similar to CBS, MUPIC does not cache the data in the buffer until new request needs to be served with the buffer.

Although MUPIC can decrease the consumption rate of single resource, but it is possible that one resource has reached bottleneck while another has many available since the rate of the whole resources usage is not considered. If the latter request needs to be served by the bottleneck resource, MUPIC needs to replace resource first. At this time, MUPIC needs to reallocate the resources before system steps forward and will lead to more start-up delay.

3. BBSLR ALGORITHM

Balanced Buffer Sharing of Limited Resources (BBSLR) algorithm allocates the size d_m of buffer for the initial request. The value of d_m is dynamically determined based on available cache, disk bandwidth and bandwidth required by the requested stream. The size of the allocated buffer is dynamically adjusted at run-time according to the current request distributing and available resources.

For the sake of obtaining the mathematical representation of the problem, we give some assumptions:

1) the stream will have a constant bit rate.
2) the requirements of disk bandwidth and cache of each request are fixed.
3) the stream served by using disk stream will demand some buffer space. The stream served by using buffer will only demand relevant buffer space. The streams will be only served by using disk and/or buffer.

Table 1 gives the variables used in BBSLR algorithm.

Table 1 The variables used in BBSLR

Variable	Description
M	Size of the whole memory (byte)
B	Size of the whole disk bandwidth (bps)
M'	Size of the currently available memory (byte)
B'	Size of the currently available disk bandwidth (bps)
M_i	Size of the cache used by the i_{th} request (byte)
B_i	Size of the bandwidth used by the i_{th} request (bps)
M_i'	Size of the cache required by the i_{th} request served by disk stream (byte)
B_i'	Size of the Bandwidth required by the i_{th} request served by disk stream (bps)
n	Number of user requests in service at some time
d_m	Limit of interval cache size between two adjacent displays referencing the same video (byte)
T	Size of whole interval cache of a certain video(byte)

After media server accepts a request, each individually served display needs to consume disk bandwidth of one stream and some memory space. For instance, if the i_{th} request is served individually by a disk stream, M_i and B_i used by it will be separately M_i' and B_i'. According to table 1, the resources used by n users can not exceed the total resources of server [11]. These are shown in (1) and (2).

$$\sum_{i=1}^{n} M_i \le M \tag{1}$$

$$\sum_{i=1}^{n} B_i \le B \tag{2}$$

Here, d_m is a distance threshold [10]. It limits the size of the needed sharing buffer between two adjacent displays referencing the same stream. If the needed buffer size between two adjacent displays exceeds d_m, they are allowed to neither cache their intermediate data nor share one disk stream by using memory. The requirements of the buffer and disk bandwidth are affected by the value of d_m. In the practical system, choosing optimal d_m can make the best of the existing resources. The value of T denotes size of the interval cache finally formed by these requests sharing one disk stream. The value of d_m is calculated by using (3) in this algorithm:

$$d_m = \begin{cases} \left\lceil \frac{B_i'}{B'} M' \right\rceil (B' \neq 0 and B_i' \le B') \\ Min(M', \left\lceil 2\frac{M-M'}{n} \right\rceil)(B' = 0 or B_i' > B') \end{cases} \quad (1 \le i \le n) \tag{3}$$

The computation of the buffer size to be allocated considers both available resources and disk bandwidth required by the requested stream in (3), so that it can keep balance of cache and disk bandwidth consumptions.

Request management process of BBSLR is described in the following.

1) If the video A is requested by the i_{th} request and the requested data is not in the buffer, the bandwidth B_i' and the buffer d_m (d_m, equal to the bigger of M_i' and the result calculated by (3)) (T=d_m) are allocated for the request i. Then the video A is read from disk by B_i', sent to the user and simultaneously cached in d_m.
2) If the video A is requested by the i_{th} request and the requested data is in the buffer, this request will be served directly from the buffer. Then perform 3).
3) If the i_{th} request can be served from the buffer, assuming that t is the interval time between this request and the former request for video A. A new d_m' is calculated by (3) and compared with (d_m-t*B_i'). If dm' is bigger, the size T of the buffer is grown to the length of T', where T'=T+d_m'-(d_m-t*B_i'). Otherwise, keeping T fixed and above procedure repeats until no new request arrives.
4) If no new request for the video A arrives in the period of the buffer remained by T or T', superfluous buffer space is reclaimed at the end of T or T' and only part of the buffer is left to provide service for these requests arriving in this period.
5) When new request arrives, if there are not enough available resources and enough resources can not be replaced (i.e., (1) or (2) is not satisfied), the new request is rejected.
6) If the service terminates, the resources of buffer and disk bandwidth are reclaimed.

3.1. The Buffer Allocation and Reclamation Algorithms

Buffer allocation: when a new request arrives, d_m is calculated by (3). If this request is served by disk stream, the buffer $Max(d_m,M_i')$ is allocated to this request. If the request is served from the buffer, we compare the value of d_m with the size of the buffer, which is remaining after we have allocated and some of them have formed interval, then set the bigger as the size of new available buffer. If available cache is not enough, the buffer reclamation algorithm is called.

Buffer reclamation:

1) when the service is over, the idle buffer resource is reclaimed.
2) if no new request for the video A arrives in the period of the buffer remained by T or T', superfluous buffer space is reclaimed at the end of T or T' and only part of the buffer is left to provide service for these requests arriving in this period.
3) if available cache is not enough when new request arrives, buffer reclamation operations as follows: (a) If there is the buffer that has been allocated and interval cache is not formed in it yet, we directly reclaim it. (b) Otherwise, if interval cache has been formed in M_i, there is no adjacent request after M_i and the rate of buffer M_i to bandwidth B_i' is the largest, then the buffer M_i is reclaimed, and these requests served by M_i will be served by disk bandwidth.

3.2. The Disk Bandwidth Allocation and Reclamation Algorithms

Disk bandwidth allocation and reclamation:

1) when available disk bandwidth is enough, the required bandwidth is allocated to the request.
2) when available disk resource is not enough, the disk bandwidth, which is serving the request j which has the least interval time from the former request and is served by disk stream, will be reclaimed and request j will be served from the buffer.
3) when the service is over, the corresponding disk bandwidth is reclaimed.

4. EVALUATION OF BBSLR

These experiments are based on network architecture that includes a video server and many users, and that the users can directly communicate with the server by enough network bandwidth. The server contains 100 constant bit rate videos, B_i'=2Mbps($1 \le i \le n$), M_i'=1MB($1 \le i \le n$). The length of each video is 90 minutes. For experimental purposes, we assume that the request arrival follows an Average process with rate r. The frequency of access to each object is based on a Zipf distribution with parameter 0.271, and the videos in the disk are organized according to the access frequency of each video. We configure experimental server with B=1Gbps and M=1/2/3/4GB. We compare the performance of different buffer sharing techniques (including BBSLR, CBS, MUPIC and FCFS) by experimental results. Here, we use a round-based technique similar to the one described in [10].

First, M is fixed 2GB. Experiments are executed by changing the value (20, 40, 60, 80, 100) of arrival rates r. Fig. 1 shows simultaneous clients sustained by each scheme. The result shows that the BBSLR algorithm achieves good performance and the result accords with that of MUPIC.

Secondly, r is fixed 50 requests per minute. Experiments are executed by changing the value (1, 2, 3, 4) of M. Because system configuration will be adjusted with current market in reality, we compare adaptive abilities of these schemes for two resources different rate by changing the size of one resource. Fig. 2 shows simultaneous clients sustained by each scheme for different M. The result shows that adaptive ability of the BBSLR algorithm is relatively better.

At last, we test response time of user requests, because one of the purposes of buffer sharing is to reduce start-up delay of users properly by quick response of memory. Assume that response time is T_a seconds directly served by disk stream and T_a*10^{-3} seconds served by buffer. Fig. 3 and Fig. 4 show respectively average start-up delay of the

requests for different *r* and different *M*. Due to caching the data in the buffer at the start and providing service for the latter requests directly, BBSLR algorithm can improve the average start-up delay.

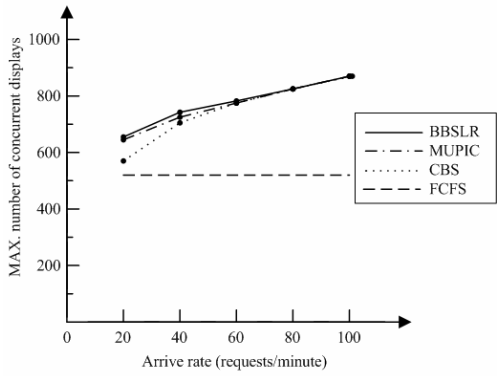

Fig. 1 Comparison of simultaneous clients for different *r*

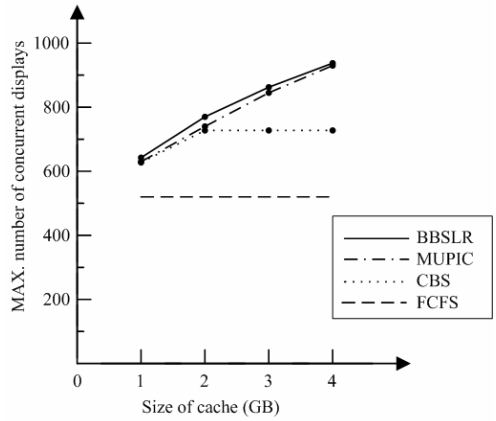

Fig. 2 Comparison of simultaneous clients for different *M*

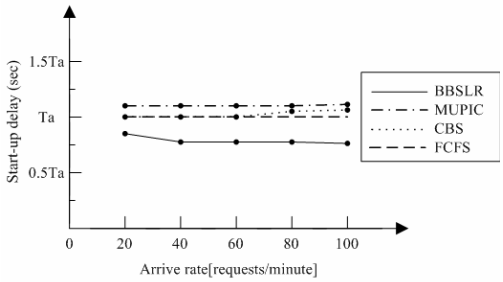

Fig. 3 Comparison of average start-up delay for different *r*

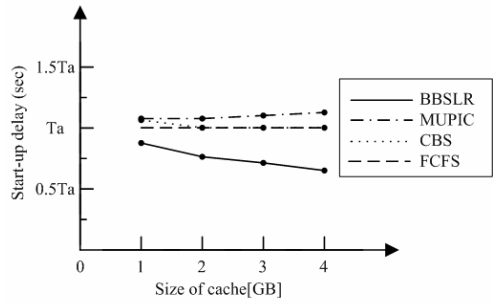

Fig. 4 Comparison of average start-up delay for different *M*

5. CONCLUSIONS

In this paper, based on the existing buffer allocation and sharing schemes, we propose BBSLR algorithm. Being compared with existing algorithms, the advantages of BBSLR algorithm lie in: (1) the resources can be utilized rationally because this algorithm dynamically allocates resources based on available cache and disk bandwidth and dynamically adjusts buffer according to current request distributing and available resources; (2) it makes the two resources to consume evenly by allocating the resources according to the ratio of available cache to disk bandwidth; (3) caching the data in the buffer at the start can not only reduce the average start-up delay, but also lessen the requirement of disk bandwidth.

In our next research, we will add support of VCR functions to BBSLR algorithm, and extend BBSLR to variable bit rate (VBR) media.

6. REFERENCES

[1] T.-PJ To and B. Hamidzadeh. "Dynamic real-time scheduling strategies for interactive continuous media servers", Multimedia Systems, ACM, 7(2):91-106, 1999.

[2]Sang Ho Lee, Kyu-Young Whang, Yang-Sae Moon, Il-Yeol Song: "Dynamic Buffer Allocation in Video-on-Demand Systems", SIGMOD Conference 2001.

[3]R. Shah, P. J. Varman, J. S. Vitter. "Online algorithms for prefetching and caching on parallel disks", SPAA 2004: 255-264.

[4]Ma H, Shin K G. "Multicast Video-on-Demand Services", ACM Computer Communication[J]. Review, ACM Press, Volume 32 , Issue 1, pp.31-34, January 2002.

[5]M. Bradshaw, J. Kurose, P. Shenoy, D. Towsley. "Online scheduling in modular multimedia systems with stream reuse", NOSSDAV 2005: 195-200.

[6]H. Vin. "Video compresion. Course CS384M: Multimedia Systems", University of Texas, 1996.

[7]E. Bommaiah, K. Guo, M. Hofmann, and S. Paul. "Design and implementation of a caching system for streaming media over the internet", In Proceedings of IEEE Real Time Technology and Applications Symposium, May 2000.

[8]A. Dan and D. Sitaram. "A Generalized Interval Caching Policy for Mixed Interactive and Long Video Workloads", In Proceedings of Multimedia Computing and Networking Conference (MMCN), pages 344-351, January 1996.

[9] Nabil J. Sarhan and Chita R. Das. "Analysis of Caching Performance in Multimedia Servers", In the Proceedings of the 8th International Conference on Internet and Multimedia Systems and Applications, pages 288-293, August 2004.

[10]W. Shi, S. Ghandeharizadeh. "Controlled Buffer Sharing in Continuous Media Servers", Multimedia Tools Appl. 23(2): 131-159 (2004).

[11]J. Fernández, J. Carretero, F. Garcia, J.. Pérez, A. Calderón. "Enhancing Multimedia Caching Algorithm Performance Through New Interval Definition Strategies", 36th Annual Simulation Symposium 2003.