2009 International Conference on Communication Software and Networks

# A Scalable and Cooperative Caching Scheme in a Distributed VOD System

LI Jing YANG Jian XI Hong-Sheng
Joint Lab of NCSC, Key Lab of Anhui NCSC
University of Science and Technology of China
Hefei, Anhui, 230027, P.R.China
jingl3@mail.ustc.edu.cn jianyang@ustc.edu.cn xihs@ustc.edu.cn

***Abstract*—In distributed streaming media server clusters, by adopting cooperative caching technique, the free cache memory of all the servers in the cluster can be used more efficiently. It will help raise the hit rate of the cache and reduce disk accesses, resulting in the improvement of the overall throughput of cluster. In this paper, a novel distributed caching model called Scalable and Cooperative Caching (SCC) is presented. SCC makes use of each server to perform the caching task and redirects the request to the proper server in the cluster to make the cache scheme more efficient. This way, SCC can provide high scalability and throughput in clustered servers with a reasonably small amount of communication among hosts. Simulation results show that SCC is an effective caching strategy providing significant performance gains in clustered video servers.**

## I. INTRODUCTION

In recent years, Video-On-Demand service has become more and more popular and many VOD service applications are emerging. Designing the video servers capable of supporting these steeply growing demands is, however, still considered challenging. The challenge is mainly due to the high bandwidth requirement to maintain concurrent video streams at a specified playback rate. Some multicast schemes, such as batching [1], periodic broadcast [2], and patching [3] are developed to improve performance. These schemes serve many requests simultaneously through multicast or broadcast transmission, but large buffer space in client devices limits their implementation on various devices such as handsets and set-top boxes.

Caching has been proved to be an important technique for improving the performance in VOD system. By adopting caching scheme, the number of concurrent streams of the VOD server can be increased, thereby serving more requests. Many caching schemes have been proposed [4] [5]. These schemes consider only one single media server site where clients interact with the server site directly. When a VOD system's client population grows, single streaming server can easily become a performance bottleneck, affecting the system's scalability. Therefore, a better solution is to use distributed streaming server systems, which are multi-server clusters. In order to make the server clusters more effective, some cooperative caching (CC) techniques [6] [7] [8] in streaming server clusters are proposed. The idea behind these techniques is to make the free memory cache of all servers to form a big shared cooperative cache. These techniques all need a manage server (can be a router or a load balancer) to store the global information, schedule the client requests and run the cache replacement algorithms, making the cluster a tightly-coupled one. With the growth of the client population, this manage server becomes the bottleneck, affecting the performance of the whole system.

In this paper, we investigate the efficient use of distributed caches in a loosely-coupled clustered VOD system which doesn't have such a manage server, and propose a novel caching algorithm called Scalable and Cooperative Caching (SCC). The SCC algorithm uses interval caching to determine which part of a video object needs to be cached for a given request, and also describes an intelligent way of having hosts cooperate in cache management.

The rest of this paper is organized as follows. Section 2 describes the system model of the clustered server. Details of SCC are explained in Section 3. Section 4 delivers the results of performance evaluation of SCC. Finally, Section5 concludes this paper.

## II. SYSTEM DESCRIPTION

Fig.1 shows the system architecture of the loosely-coupled clustered video server consisting of 4 streaming servers, connected by fast connections such as fast Ethernet or Gigabit Ethernet. There is no central entrance server in this cluster, each streaming server is independent and has its own storage system. Servers in the cluster can have different OS or hardware, forming a heterogeneous environment. Each server is placed according to the actual geography environment, for example, one server for an area and serves requests from its own area first. If a streaming server finds itself unable to serve a request, it can redirect the user request to a remote server through the network.

978-0-7695-3522-7/09 $25.00 © 2009 IEEE
DOI 10.1109/ICCSN.2009.98

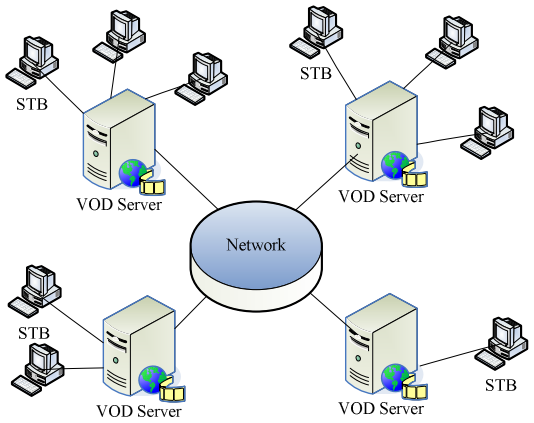


Figure 1. Clustered server architecture

## III. SCALABLE AND COOPERATIVE CACHING

This section is dedicated to describe the SCC algorithm. We first define some terminologies used in the algorithm as follows:

- Local request: The request from a server's own domain.
- Remote request: The request redirected from other servers.
- Registry: A server maintains information of its current states in a data structure called registry. It contains information about a request including request type, movie id, arrival time, interval size between the request and its preceding stream; it also contains information about current load and free cache size of the server. When a request is finished or terminated, the corresponding item in the registry is removed and information of the relevant items is modified.

Because clients' requests in different area may be biased toward different movies, some popular movies in one area may be unpopular in another. If each server only servers its local requests, then the Interval Cache scheme on each server will be inefficient. The main idea of SCC scheme is to reassign different requests to proper servers and balance the load on each server, thus the cache scheme in each server can be more efficient and the capability of the whole system can be improved.

Figure 2 illustrates the procedure of the SCC algorithm. The key issue of SCC is to decide when to redirect a local request and whether to accept a remote request. If servers redirect too many local requests, the network connecting the servers may be overloaded; if a server accepts too many redirected requests, some of its local requests may be affected, which is unbearable (as mentioned above, each server is greedy, it only cares for requests from its own domain, it will not cooperate with others if it can't get any benefit).

We assume there's some revenue when a request is served and some rules can be made:

- All servers aim to get more revenue.
- If a server serves one local request, it will get all the revenue.
- If a redirected request is served by some remote server, the revenue will be divided into two parts, one for the local server who redirects the request, the other for the remote server who serves the request.
- The accept message sent by remote server will contain a parameter which denote the expense of accepting the redirected request. The revenue will be divided according to the expense, the local server will get more revenue if the expense is lower.

According to the first three rules, a server will try to get more revenue, so it won't redirect too many local requests, and a server won't accept too many redirected requests which may lower its capacity for its local requests either, it only accept a redirected request when it's lightly loaded. The fourth rule makes sure that the local server will redirect the request to the remote server with the smallest expense, so the capacity of the whole cluster can be increased.

Let $l_i$ be the current load on server $i$ in terms of the number of concurrent streams served from disk I/O, $L_i$ be the disk bandwidth of server $i$. SCC uses two thresholds $T_l$ and $T_h$ ( $T_l < T_h$ ) to measure the load status on each server. When the load $l_i$ is less than $T_l$, there is no need to redirect any requests since only a small part of the service capacity is used and the server can still accommodate new requests without the risk of reaching its service capacity. When the load $l_i$ is between $T_l$ and $T_h$, there is a possibility the load $l_i$ may increase to the service capacity, so if the local request can't form an interval then request redirection is performed with a certain probability $P_i$ based on the current status, which can be represented as

$$P_i = P_{\max} \frac{l_i - T_l}{T_h - T_l} \tag{1}$$

When the current load is above $T_h$, there is a great potential that the load $l_i$ will increase to the service capacity. If the incoming request can't form an interval cache, request redirection is performed with a certain probability $P_{\max}$.

The second core step is for the server to decide whether to accept a redirected request or not. Assume server $i$ receives a redirected request message, if the movie that this redirected request ordered is a hot movie and can be served with only a little cache expense, or

current load $l_i$ is less than $T_l$, then an accept message will be sent back with a parameter $s$ ,which can be represented as

$$s = \begin{cases} \dfrac{b_{req}}{L_i - l_i}, serve\ the\ remote\ request\ from\ disk \\ \dfrac{C_{req}}{C_{free}}, serve\ the\ remote\ request\ from\ cache \end{cases} \quad (2)$$

where $b_{req}$ denotes the disk bandwidth required, $C_{req}$ denotes the cache size required and $C_{free}$ denotes the current free cache size in the server.

## IV. PERFORMANCE EVALUATION

In this section, we present simulation results and our analysis. For cooperative caching algorithms, the most significant metric is the number of cached streams. It is a good indicator of possible improvement in server capacity, so we use average number of cached streams as a main performance metric of our performance study. Then, cache hit ratio is calculated to depict the ratio of number of cached streams to total number of streams in the clustered server.

In the experiments, the following default values of system and workload parameters are used. The number of servers in the cluster is 8. Total cache size is chosen to be 1GB per server. The bias of client requests for different movie files conforms to the Zipf distribution. The probability function of the Zipf distribution for the $i$-th popular movie is given by $zipf(i) = C / i^{1-\theta}$ with parameter $\theta$ and normalized constant $C$. We choose the usually assumed $\theta = 0.271$ to represent a normal skewness of request pattern in VOD servers.

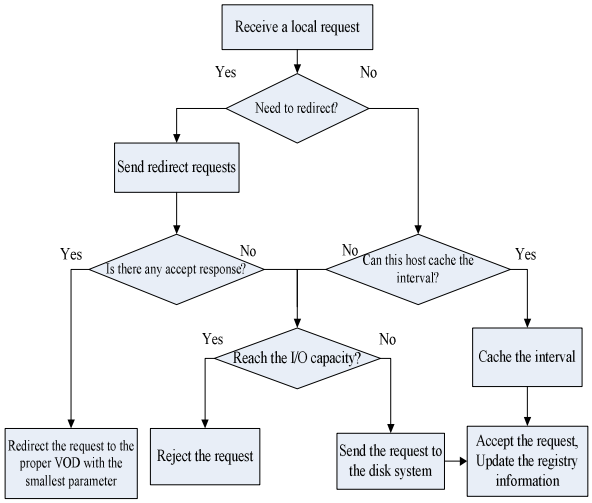


Figure 2. SCC Algorithm

### A. Impact of cache Cooperation

In this experiment, the performance gain obtained by cache cooperation is investigated. For each server, arrival rate of requests is assumed to follow a Poisson distribution with mean arrival rate 0.05. With given movie duration of 90 minutes, it makes the average number of concurrent streams in each server 2160 according to Little's law. The playback rate of movies is assumed to be 1.5Mbps. Without cache cooperation, when a server can't cache a local request using interval caching, it immediately sends it to the I/O system. If the I/O system has already reached its capacity, then the request will be rejected. With cache cooperation, such a request may be redirected to other servers and finally served from other caches. Figure 3 shows the number of cached streams in the cluster with and without cache cooperation. The performance result suggests that if different servers cooperate in a proper way, the number of cached streams can be significantly improved and more requests can be served.

### B. Impact of Movie Popularity

SCC uses interval caching for individual cache management. The performance of interval caching relies on the temporal locality of movie requests, which is determined by a function of arrival rate and the movie access pattern. With a higher arrival rate, the cache hit ratio of Interval Caching algorithm becomes larger because frequent arrivals produce smaller interval sizes, thus more intervals get accepted with a given cache size. The movie popularity of different servers also affects the performance of SCC algorithm. When movie popularity of different servers is not same, the cache hit ratio of the whole cluster is larger than the situation when different servers have the same movie popularity. Figure 4 shows the performance of SCC with the same and different movie popularity in different servers.

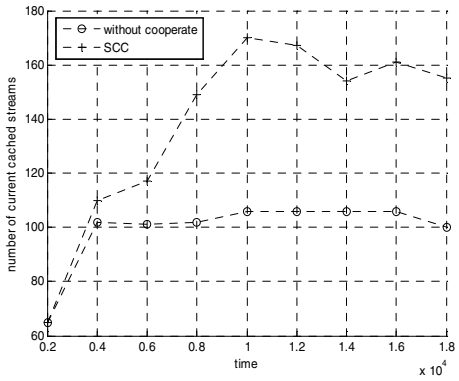


Figure 3. Number of current cached streams

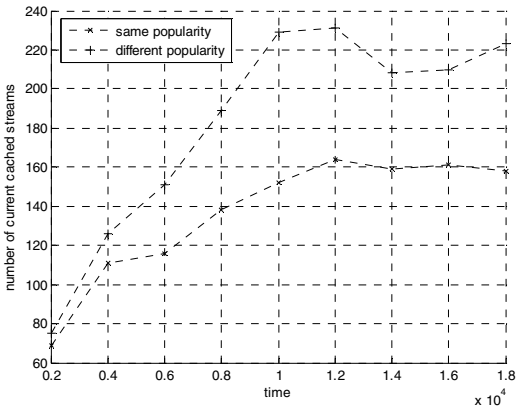


Figure 4. Impact of movie popularity

## V. CONCLUSIONS

In distributed streaming server cluster systems, by adopting cooperative caching technique, the disk accesses can be effectively reduced, and the disk bottleneck can be alleviated. As a result, the performance and scalability of the server clusters improve, making the clusters more effective. Therefore, the designing of cooperative caching algorithms is very significant. This paper presents a new cooperative caching strategy SCC for distributed streaming server clusters. SCC is based on server level cooperation and employs interval caching to maintain individual caches in the server of the cluster. When receiving a client request, it takes into account existing streams in the local server and remote servers, and a redirection may be made, so that the overall cache hit rate can be improved. Compared with traditional cooperative caching algorithms and cache algorithms, our algorithm don't need to be performed on a central server of the cluster which stores all the information of the cluster and is more suitable to the distributed streaming server cluster systems.

## ACKNOWLEDGEMENTS

The authors would like to thank the anonymous referee whose insightful comments helped us to improve the presentation of the paper. This work was supported by National "863" Project of China (Grant No. 2006AA01Z114)

## REFERENCES

[1] A Dan, D Sitaram, and P Shahabuddin, "Dynamic batch- ing policies for an on-demand video server", Multimedia Sys-tems, Springer-Verlag New York, Secaucus NJ USA, June 1996, pp. 112-121.

[2] Kien A. Hua and Simon Sheu, "An efficient periodic bro-adcast technique for digital video libraries", Multiamedia Tools and Applications, Springer Netherlands, October 2004, pp. 157-177.

[3] Kien A. Hua, Ying Cai, and Simon Sheu, "Patching: a multicast technique for true video-on-demand services", In Proceedings of the sixth ACM international conference on Multimedia, ACM New York, NY USA, 1998, pp. 191-200

[4] A. Dan and D. Sitaram. "A generalized interval caching policy for mixed interactive and long video workloads", In Proceedings of ACM/SPIE Multimedia Computing and Net-working (MMCN'96), January 1996, pp. 344-351

[5] Taeseok Kim, Hyokyung Bahn and Kern Koh. "Consider-ing user behavior and multiple QOS supports in multimedia streaming caching", The Journal of VLSI Signal Processing, Springer Netherlands, February 2007, pp. 113-122

[6] Suneuy Kim and Kim Hang Thi Tran. "Supporting scalable and cooperative interval caching in a clustered video server", In Sixth IEEE International Symposium on Network Computing and Applications (NCA 2007), IEEE Computer Society Technical Committee on Distributed Processing, July 2007, pp.283-286

[7] Tiecheng-Gu, Baoliu Ye, Minyi Guo and Daoxu Chen. "Implementing cooperative caching in distributed streaming media server clusters",Embedded and Ubiquitous Computing, Springer Berlin, July 2004, pp. 807-817