STUDY OF A SERVER-LESS ARCHITECTURE FOR VIDEO-ON-DEMAND APPLICATIONS

Jack Y. B. Lee and Raymond W. T. Leung

Department of Information Engineering
The Chinese University of Hong Kong

ABSTRACT

Video-on-demand (VoD) systems have traditionally been built around the client-server architecture, where a video server stores compressed video for delivery to clients connected by a network. As the system scales up, the server will need to be upgraded and this can become very expensive when the system scales beyond thousands of users. In this study, we investigate a radically different architecture where the bottleneck – video server, is eliminated altogether. Specifically, this server-less architecture relies on the client machines for distributed data storage and delivery. A client initiating a new streaming session will first locate other clients where the requested stream is stored, and then requests delivery of the stream directly from those clients instead of from a central server. This fully distributed architecture is inherently scalable as the storage and delivery capacity grows with the number of clients in the system.

1. INTRODUCTION

Current video-on-demand (VoD) systems are commonly designed around the client-server architecture. Under this architecture, a client sends a request to the video server for a video title and then the server transmits video data to the client for playback. As the number of user increases, the server will eventually reach its capacity limit. To further increase the system capacity, one can add more servers to share the load.

Nevertheless, the cost of video servers are still substantial, as video servers typically require high-end server hardware with high I/O bandwidth, large memory capacity, as well as storage capacity. Moreover, the distribution network will also need to be upgraded with more bandwidth to carry the vast amount of video traffic to the users. Given the high cost of long-distance backbone networks, it is no wonder why metropolitan-scale VoD services are still uncommon in practice.

In this study, we take a radically different approach to building scalable VoD systems. In particular, we turn our attention to an often-neglected element in a VoD system – the client-side device or commonly called the set-top box (STB).

Developments of STB have continued for many years and current STBs not only are low cost, but also are relatively powerful due to the rapid technological development and economy of scale achieved by the personal computer industry. This development opens a new way to build VoD systems.

Specifically, we take advantage of the increased storage and processing capability of STBs to build a completely distributed VoD systems that does not require dedicated server at all. We call this a server-less architecture for obvious reason. In this server-less architecture, all STBs, or called a node in this paper, in the system serve both as a client and as a mini-server. Video data are distributed among the nodes and multiple nodes work together to serve video streaming requests from other nodes. The beauty of this architecture is that the system is inherently scalable, i.e., when new users are added to the system, they add both streaming load and streaming capacity to the system. Moreover, network costs can also be reduced because the nodes are likely to be clustered together, reducing the need for costly long-distance backbone network.

However, building a server-less VoD system is not without challenges. First, one needs a placement policy to determine how to distribute the video data among the nodes. Second, one needs an algorithm to schedule the retrieval and transmission of video data for a video streaming session. Third, as video data are distributed among many nodes, the system will need to provide a directory service for a node to locate and request data for a new stream. Fourth, as nodes in the system are more loosely controlled than dedicated video servers, the system will need to adapt to changes in system configurations such as addition of new nodes, deletion of existing nodes, node heterogeneity, etc.

In this study, we address the first two of the above-mentioned challenges and leave the challenge of designing a directory service and system adaptation for future work. In the rest of the study, we assume that a node always knows the location of the data it needs and the system configuration never changes. The rest of the paper is organized as follows: Section 2 summarizes the contributions of this study; Section 3 presents the proposed server-less VoD architecture; Section 4 evaluates performance of the architecture using numerical results; Section 5 discusses the scalability issue and summarizes the paper.

2. CONTRIBUTIONS OF THIS STUDY

First, compared to the traditional client-server architecture, our approach decentralizes and distributes the server functions to the clients. This server-less architecture not only eliminates the primary bottleneck in the system, but also is inherently scalable.

0-7803-7304-9/02/$17.00 C2002 IEEE

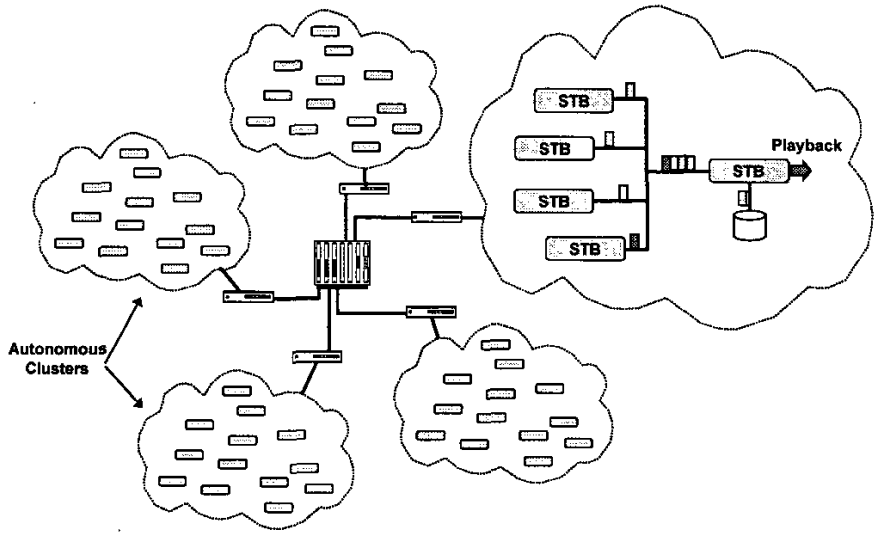

Figure 1: A server-less architecture for VoD systems.

Second, compared to current peer-to-peer (P2P) systems such as Naptser and Gnutella, the server-less architecture investigated in this study serves completely different applications, i.e. video-on-demand versus file sharing. In particular, VoD applications have stringent performance requirements that are essential to the correct operation of the system. Consequently, the server-less VoD architecture requires completely different data placement policy, retrieval scheduling, transmission scheduling, and fault tolerance mechanism compared to the current P2P systems.

To the best of the authors' knowledge, our study is the first to address issues in building server-less VoD systems with a goal to provide a service comparable to or even better than existing client-server-based VoD architectures. In particular, we address the issue of data placement policy and presented a striping-based algorithm for video data placement. New retrieval and transmission schedulers are then developed for this data placement policy. To address the issue of reliability, we incorporate the use of data and node redundancies into the system architecture and extend the scheduling algorithms to account for fault tolerant operation. Through numerical results, we show that building VoD systems using the presented server-less architecture is feasible and it can achieve performance comparable to client-server-based systems despite not having dedicated high-end video servers.

3. ARCHITECTURE

A server-less VoD system comprises a pool of homogeneous user nodes connected by a network as shown in Fig. 1. Each node has its own CPU, memory and disk storage. Inside each node there is a mini video server software that serves a portion of each video title to other nodes in the system. Unlike conventional video server however, this mini server serves a much lower aggregate bandwidth and therefore can readily be implemented in today's STBs and PCs. For large systems, the nodes can be further divided into clusters that each forms an autonomous system independent from other clusters.

3.1 Data Placement Policy

Unlike dedicated video servers where storage capacity is usually large, a node in the form of a STB or a PC will have relatively limited storage capacity. Therefore, instead of replication, we propose the use of striping as the data placement policy for the architecture.

Specifically, each video title is divided into fixed-size striping units (or called blocks) of Q bytes each. These blocks are then distributed to all nodes in the cluster in a round-robin manner. This node-level striping scheme avoids data replication while at the same time divides the storage requirement equally among all nodes in the cluster. To initiate a video streaming session, a client node will first locate the set of server nodes carrying blocks of the desired video title, the striping policy and other parameters (format, bitrate, etc.) through a directory service. These server nodes will then be notified to start transmitting the video blocks to the client node. The notification can be performed directly by the client node or indirectly by the directory service, of which the exact mechanism involved is beyond the scope of this study.

3.2 Retrieval and Transmission Scheduling

Let N be the number of nodes in the cluster and assume all video titles are constant-bit-rate (CBR) and share the same bitrate R_v. For a server node in a cluster, it may have to retrieve video data for up to N video streams, of which $N-1$ of them are transmitted while the remaining one played back locally. Note that as a video stream is served by N nodes concurrently, each node only needs to serve a bitrate of R_v/N for each video stream.

Many existing video server designs employ round-based schedulers such as SCAN and its variants [1]. In our design, we

employ the Grouped Sweeping Scheme (GSS) proposed by Yu, *et al.* [2] to schedule a node's disk retrieval and network transmission. Compared to the more common SCAN scheduler that maximizes throughput at the expense of buffer overhead, GSS allows one to control the tradeoff between disk efficiency and buffer requirement. This is a crucial feature as disk throughput may not be the bottleneck in a server-less VoD system (c.f. Section 4.3).

In GSS, streams are divided into g groups in which retrievals within a group are scheduled using SCAN. The groups are served in a round-robin manner. We call the period of serving a group a micro round and the period of serving all groups once a macro round. If one set g=1 and g=N then GSS reduces to SCAN and FCFS respectively. Intermediate values of g can be used to tradeoff disk efficiency with buffer requirement.

Both micro and macro rounds have fixed duration, denoted by T_g and T_f respectively. Given a data block size of Q bytes, up to N/g data blocks will be retrieved in a micro round using the SCAN scheduler. These retrieved data blocks will then be transmitted at a rate of R_v/N for a duration of T_f seconds, which precisely equals to g micro rounds. Therefore the g groups are effectively staggered in time and this reduces the combined buffer requirement [2].

3.3 Fault Tolerance

In a server-less VoD system, fault tolerance becomes an essential capability as reliability of STBs and PCs will be significantly lower than dedicated video servers located in a data centre run by professional operators around the clock. Moreover, given the relatively large number of nodes, the system needs to expect and prepare to recover not from a single failure, but from multiple simultaneously failures as well.

When a node fails, all data stored in that particular node becomes unavailable – called data erasure. To recover from data erasures, erasure-correcting codes such as the Reed-Solomon Erasure Correcting (RSE) Code [3,4] can be used. Specifically, a (n, h)-RSE codeword comprises n symbols of which (n–h) of them are message symbols (i.e. data) and the remaining h are redundant symbols. One can recover all (n–h) message symbols as long as *any* (n–h) out of the n symbols are correctly received.

By extending the striping-based placement policy in Section 3.1 with a (N, h)-RSE code, the system will have sufficient redundant data for a client node to recover all video data with up to h node failures in the cluster. To accommodate the RSE-code, we need to modify the placement policy and the schedulers. For the placement policy, an additional encoding step will be needed to compute the h redundant blocks for each group of (N–h) video data blocks. Moreover, as now only (N–h) of the stored data are playable data, we will need to increase the strip unit size from Q bytes to $Q_r = QN/(N-h)$ bytes to maintain the same data size of a striping group.

For the disk scheduler, the retrieval unit will be increased from Q bytes to Q_r bytes. Transmission rate will also increase from R_v to $R_r = R_v N/(N-h)$ to maintain the same video bitrate.

4. PERFORMANCE EVALUATION

In this section, we evaluate the system requirements and performance of the server-less VoD architecture presented in Section 3. Numerical results are computed using parameters listed in Table 1. Due to space limitation, derivations of the performance model are omitted.

4.1 Storage Capacity

Under the striping-based placement policy, the storage requirement is equally shared by all nodes in the cluster. Hence, the more nodes in a cluster, the less storage per node is required to store the same amount of video data. Therefore the storage requirement imposes a lower limit on the scale of a cluster.

For example, assuming a video bitrate of 150 KBps and a video length of 2 hours, the total storage for 100 videos is 102.9 GB. Given that today's harddisks have at least tens of gigabytes of storage capacity, even if each node only allocates 1 GB for video storage, the minimum cluster size needed is still only 108 nodes including redundancy of 5 nodes to achieve a system mean time to failure (MTTF) of 1000 hours.

4.2 Network Capacity

One way to connect nodes together to form a cluster is to use a switch-based network such as switched Ethernet. Today's medium-range Ethernet switches typically has switching capacity of 32 Gbps or more. Given a video bitrate of 150 KBps and ignoring protocol overhead, a node will consume 154.66 KBps both upstream and downstream including a redundancy of 7 nodes for a cluster size of 200 nodes. Using 154.66 KBps as the bandwidth requirement, a 32 Gbps switch running at 60% utilization can support a maximum cluster size of 8124 nodes.

On the other hand, the server-less architecture does require more bandwidth for the last-mile access network. In particular, the architecture requires more upstream bandwidth than traditional client-server-based VoD systems, which require little to no upstream bandwidth. This requirement will rule out some access network with asymmetric bandwidth such as ADSL or cable modem. Nonetheless, the emerging Ethernet-based access network running at 10Mbps or even 100Mbps full-duplex will provide more than sufficient bandwidth to accommodate a server-less VoD system in the future.

4.3 Disk Access Bandwidth

A node obviously will have finite resources, including memory and disk access bandwidth. We first investigate the disk access bandwidth issue as it determines the configuration of the disk scheduler (GSS), which in turn affects the buffer requirement.

The disk scheduler has two configurable parameters, namely block size Q and the number of groups g in GSS. Increasing the block size or decreasing the number of groups in GSS results in higher disk efficiency, at the expense of larger buffer requirement and longer system response time. Therefore in terms of buffer requirement and system response time, it is desirable to reduce the block size Q and to increase the number

of groups g in GSS as long as retrieval of requests in a group can be completed within a micro round.

Take an off-the-shelve harddisk – Quantum Atlas 10K [5] as an example. Using a block size of 4KB and setting $g=N$, the computed worst-case retrieval time for a group is 0.017 seconds while the micro-round time is 0.027 seconds. Hence the previous retrieval constraint is satisfied even if we reduce GSS to the least efficient special case of FCFS with $g=N$. This is in sharp contrast to conventional VoD systems where disk efficiency is one of the major performance bottleneck in the video server.

4.4 Buffer Requirement

Based on results from the previous section, we set $g=N$, effectively reducing the GSS scheduler to FCFS. With a block size of 4KB, the total buffer requirement is 2.376 MB at the cluster size of 200 nodes. Given the current cost of memory, this buffer requirement, while not insignificant, should be practical for STBs and ordinary PCs.

4.5 System Response Time

The system response time comprises scheduling delay and prefetch delay. Scheduling delay is the time to wait for admission to a group under the GSS scheduler, while prefetch delay is the time to receive one striping group of blocks to perform erasure correction. Fig. 2 plots the system response time, scheduling delay, and prefetch delay, versus the number of nodes in a cluster running at 90% system utilization. The prefetch delay increases with increase in the cluster size while the scheduling delay levels off for cluster size larger than 100 nodes. Clearly prefetch delay dominates and at a cluster size of 200 nodes, the system response time is 5.615 seconds.

5. SUMMARY AND CONCLUSIONS

Results from the previous section reveal several characteristics of a server-less VoD system. First, within the system parameters used, the system scalability is not limited by network or disk bandwidth. The disk in particular has more than enough bandwidth even when used with the least efficient FCFS scheduler. This property is in sharp contrast to conventional VoD systems built around the client-server architecture, where maximizing I/O efficiency is of primary importance.

Second, the primary limit on the scalability of a cluster is, surprisingly, the system response time. This is a result of the striping-based placement policy, where a striping group must be completely received before it can be erasure-corrected for playback. The size of the striping group, and consequently the prefetch delay, increases linearly with the number of nodes in the cluster. While one can further reduce the block size to reduce the response time, protocol overhead will become significant.

Nevertheless, one does not need to increase the cluster size unless storage capacity is insufficient. With results from Section 4.1 showing that the lower limit on the cluster size is 108 nodes, one can divide a large system into multiple clusters of say, 200 nodes each, to maintain the system response time to a reasonable 5.615 seconds. As these clusters are autonomous and independent of one another, one can continuously expand the scale of the system simply by forming new clusters.

ACKNOWLEDGEMENTS

This research is partially funded by research grants (Direct Grant, Earmarked Grants CUHK6095/99E) from the HKSAR Research Grant Council and the AoE-IT, a research grant from the HKSAR University Grants Council.

6. REFERENCES

[1] D. J. Gemmell, H. M. Vin, D. D. Kandlur, P. V. Rangan, L. A. Rowe, "Multimedia storage servers: a tutorial", *Computer*, vol.28(5), May 1995, pp.40–49.

[2] P. S. Yu, M. S. Chen, and D. D. Kandlur, "Grouped Sweeping Scheduling for DASD-based Multimedia Storage Management", *ACM Multimedia Systems*, vol.1(2), 1993, pp.99–109.

[3] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*, Englewood Cliffs, NJ: Prentice-Hall, 1995, pp.227–234.

[4] A. J. McAuley, "Reliable Broadband Communication Using a Burst Erasure Correcting Code", in *Proc. ACM SIGCOMM 90*, Philadelphia, PA, September 1990, pp. 287–306.

[5] G. Ganger and J. Schindler, "Database of Validated Disk Parameters for DiskSim", http://www.ece.cmu.edu/~ganger/disksim/diskspecs.html.

Table 1. Parameters used in numerical analysis in Section 4.

Parameters	Symbol	Value
Node mean time to failure	$1/\lambda$	1000 hours
Node mean time to repair	$1/\mu$	10 hours
Video block size	Q	4096 Bytes
Video bitrate	R_v	150 KB/s
Disk fixed overhead	α	0.176ms
Disk rotational latency	W^{-1}	5.99 ms
Minimum disk transfer rate	r_{min}	18.68 MB/s

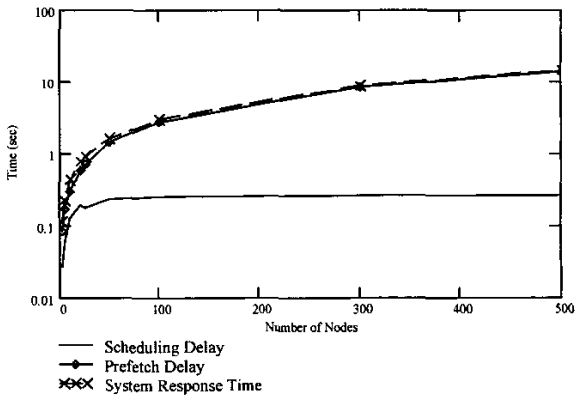

Figure 2: System response time, scheduling delay, and prefetch delay versus cluster size (Q=4KB, $g=N$, 90% utilization).