Int J Parallel Prog (2009) 37:175–194
DOI 10.1007/s10766-008-0092-3

Recovery Strategies for Streaming Media Service in a Cluster-Based VOD Server with a Fault Node

Joahyoung Lee · Inbum Jung

Received: 24 October 2006 / Accepted: 4 December 2008 / Published online: 1 January 2009

Abstract Due to the economic cost and good scalability, cluster-based server architecture is used for VOD services. This server consists of a front-end node and multiple backend nodes. In this server architecture, backend nodes are added simply to support large-scale on-demand clients. However, as the number of backend nodes increases, the possibility of backend node failure also increases. A backend node fault not only degrades the quality of serviced streaming media but also decreases the number of streams supported in the VOD server. For successful VOD service, even if a backend node enters a fault state, the streaming service in progress should be re-continued after a short recovery time. As the recovery strategy, when legacy RAID methods are applied to cluster-based VOD servers, the excessive internal network traffic between the backend nodes causes performance degradation. In addition, the backend nodes demonstrate inefficient CPU utilization for the recovery process. In this paper, to address these problems, a new fault recovery strategy is proposed based on the pipeline computing concept. The proposed method not only distributes the network traffic generated from the recovery operations but also makes efficient use of the CPU time available in the backend nodes. Based on these advantages, even if the cluster-based server has a backend node that fails, the proposed method provides more QoS streams compared to the existing recovery method. In addition, since the proposed method needs a very short recovery time, the streaming services in progress are sustained without degradation of media quality.

Keywords Cluster server · Recovery · Pipeline computing · Streaming service

J. Lee · I. Jung (✉)
Department of Computer Science and Engineering, Kangwon National University,
Chuncheon 200-701, South Korea
e-mail: ibjung@kangwon.ac.kr

J. Lee
e-mail: jinnie4u@kangwon.ac.kr

1 Introduction

Based on recent advanced computer and communication technologies, commercial multimedia services such as Video-On-Demand (VOD), digital library and e-learning have been provided for online clients. In these multimedia services, the VOD service is the most highlighted multimedia application. It provides online clients with streaming media stored in VOD servers (http://www.mpeg.org) [1]. Contrary to traditional file servers, VOD servers are subject to real-time constraints for retrieving and delivering movie data to clients. For successful commercial VOD services, the streaming media should satisfy the designated QoS criteria such as the no-jittering phenomenon and the right bit rates for clients. In particular, even if a fault event occurs in servers, the streaming service should continue within acceptable human Mean Time to Repair (MTTR) values [2,3].

The cluster system architecture has been exploited in the areas of web, database, game and VOD servers [7]. It has an advantage of the ratio of performance to cost and is easily implemented with PC equipment. The cluster system consists of a front-end node and multiple backend nodes. All the nodes have the same hardware specifications or not. In order to use this system as the server architecture for VOD services, MPEG video data should be distributed to multiple backend nodes. To support client requests, the backend nodes concurrently transmit stored video data to clients. For large-scale VOD services, the backend nodes not only store varied video content but also provide sufficient QoS streams. To satisfy these requirements, the addition of backend nodes could be considered. However, as the number of backend nodes increases, the fault rate of the backend nodes also increases. A backend node can fail due to various reasons, such as hard disk failure, a network malfunction, OS failures and so on. A failed backend node not only degrades the quality of the serviced streaming media but also decreases the number of QoS streams supported in the VOD servers. Since a failed node causes low video quality such as jittering phenomena and low bit rates below the expected level, such events keep VOD services from being commercially successful. To deal with a realistic VOD service, the recovery mechanism for treating the fault event should be designed in the VOD servers.

To study the recovery methods of failed backend nodes, we implement a cluster-based VOD server composed of general PCs. Based on the cluster-based architecture, we adopt parallel processing for MPEG media to support a large number of clients. From the implemented VOD server, we evaluate a traditional recovery system composed of the advantages of RAID-3 and RAID-4 algorithms. Since these RAID algorithms are known for providing a very high-speed data transfer rate, they are suitable for VOD servers treating video streaming. In this paper, the basic concept of these legacy algorithms is introduced as the recovery mechanism of the cluster-based VOD server. However, this recovery system causes an input network bottleneck in the recovery node and demonstrates inefficient CPU usage in backend nodes. To address these issues, we propose a new recovery strategy based on the pipeline computing concept. Since the proposed method distributes the recovery workload composed of exclusive-OR operations, the utilization of CPU resources in backend nodes could be perfectly balanced. This effect alleviates the bottleneck problem invoked in the input network port of a recovery node. In our implemented cluster-based VOD system, even if a

backend node fails, the new recovery method provides more QoS streams compared to the existing recovery method. In addition, since the proposed method has a very short MTTR value, the streaming services in progress are sustained without degradation of media quality. Due to these advantages, the proposed recovery strategy could be applied in VOD service for large-scale on-demand clients.

The rest of this paper is organized as follows. Section 2 describes related work. Section 3 explains our cluster-based VOD servers and the recovery methods based on RAID-3 and RAID-4. In Sect. 4, a new recovery strategy based on pipeline computing is proposed. Section 5 describes the experimental environments. In Sect. 6, the performance evaluation is shown. Section 7 concludes the paper.

2 Related Work

Many studies have been undertaken on VOD systems to provide a stable service for more clients under various client requirements and limited resources [4–6]. For a commercially successful VOD service, even if the partial failure state occurs, media streaming should be sustained to clients within the limited MTTR values. The human acceptable MTTR value is an important QoS metric for outstanding VOD service. There has been much research in the area of fault tolerance for normal files, databases and web servers. However, streaming media need real-time processing in the data retrieving and transmission procedure. Until now, there have not been enough studies to sustain the QoS streams in the partial failure condition of VOD servers.

Based on the mirror concept, several studies have been performed regarding recovering failed storage systems [7,8]. The Tiger video server was implemented in the mirror-based storage system for VOD service [9]. Rotational Mirrored Declustering (RMD) techniques were suggested for recovering failed disks or individual nodes [10]. However, these mirror-based approaches use disk storage inefficiently and incur the heavy burden of the recovery node.

Redundant Array of Inexpensive Disks (RAID) mechanisms are usually exploited to recover the failed disks or parallel nodes in clustered servers. In particular, the RAID-3, 4 and 5 mechanisms are based on the parity-based recovery algorithm [8, 11,12]. In RAID-3, the data blocks are striped and written on the disks. Due to the fine striped unit, numerous disk accesses are necessary to retrieve large video blocks. This characteristic degrades the data transfer rate from the disks. In RAID-4, the entire blocks are stored in a disk. Since this method provides a coarse striped unit, disk performance can be improved by retrieving larger blocks during every disk access. Both RAID-3 and RAID-4 have their own parity disks used in the recovery procedure. However, RAID-5 spreads out the parity blocks of the same rank across all disks. When a backend node fails, all remaining backend nodes participate individually in the complete recovery operations. However, since the parity blocks are distributed to all disks, to exchange these blocks, the network traffic between backend nodes greatly increases. Furthermore, since the actual load of the backend nodes cannot be estimated in real time, steady QoS streams cannot be sustained during the failure of a backend node.

Recently, cluster server architecture has been used for various areas due to its low cost and high performance http://www.ieeetfcc.org. In particular, for the QoS

of streaming media, a real-time requirement for the data retrieving and transmission process is necessary. Even if a backend node fails, the irregular ceasing and jittering phenomena of streaming media should be solved within the human acceptable MTTR period [2,3]. Therefore, to obtain commercially successful VOD services, the recovery system should be devised based on the characteristics of streaming media. For these reasons, we focus on both improving the performance of the recovery system and achieving a better MTTR value.

3 Cluster-Based VOD Server

3.1 Architecture of VODCA

For large-scale VOD services, we implemented a cluster-based VOD server called Video On Demand on Clustering Architecture (VODCA) [13]. VODCA consists of a front-end node named the Head-end Server (HS) and several backend nodes known as the Media Management Server (MMS). Figure 1 shows the architecture of our VODCA server and various VOD clients. To provide streaming services, the HS node works together with the MMS nodes. Throughout the internal network path between an HS node and the MMS nodes, they exchange working states and internal commands.

3.1.1 Head-End Server (HS) node

The HS node not only receives clients' requests but also manages the MMS nodes to support QoS. Figure 2 shows the architecture of an HS node. When new MPEG movies are enrolled, the HS splits them and distributes them into each MMS node.

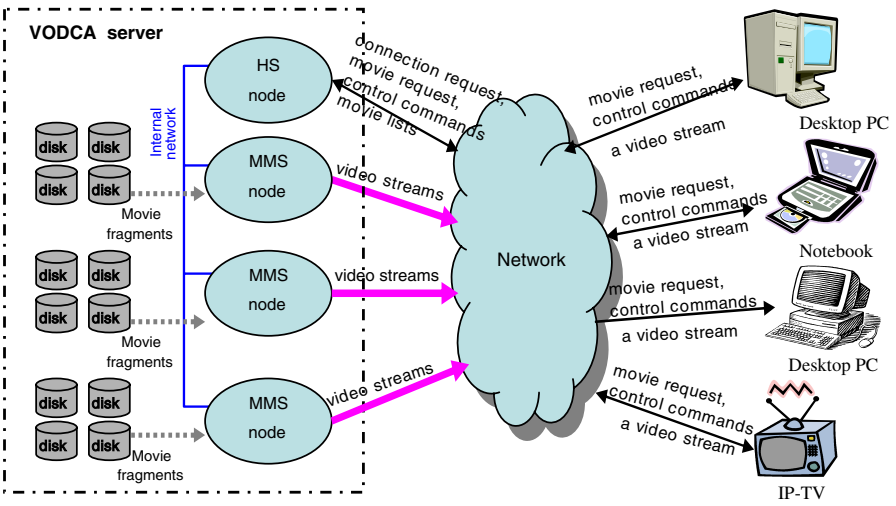

Fig. 1 Architecture of VOD service

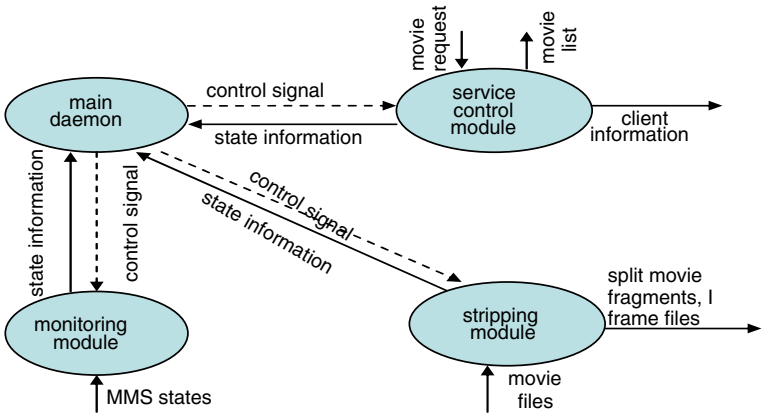

Fig. 2 Architecture of an HS node

To perform these administrative functions, the HS consists of a striping module, a monitoring module, a service_control module and a main_daemon module.

The striping module reads the header of the MPEG movie files and splits the movie data into GOP units. To maintain the order of the split movie data, the sequence number and header data are attached in front of both the GOP files and the frame files. These files are delivered to each MMS node according to the striping policy. As striping policies in the VODCA system, the round-robin policy or the SCAN policy can be selected.

The monitoring module provides functions for managing the VODCA system. The module displays CPU usage, memory and network in the HS node and MMS nodes. The system administrator can insert, delete and modify the MMS nodes, and s/he can measure the quantity of memory, disk and network bandwidth currently used to provide the streaming services. To monitor the working state for the MMS nodes, a heartbeat protocol is used between the HS node and the MMS nodes. Since the monitoring period in the heartbeat protocol is 2 s, it does not cause performance degradation of the MMS nodes.

The service_control module manages the network connection between the VOD servers and the clients. In this module, a thread is created to support a VOD client, and information about the enrolled movies is transmitted to the clients. For the normal play mode, this module establishes a UDP connection between the MMS nodes and the clients. After that, this module lets the MMS nodes transmit the demanded movie data to the client. In addition, this module treats the control commands such as fast forward, fast rewind, pause and resume. These commands are sent to the MMS nodes via this service_control module. To guarantee accurate data transmission for control commands, we use TCP connections. The main_daemon module supervises all modules within the HS node and provides the administrator with general GUI interfaces.

3.1.2 Media Management Server (MMS) node

The MMS nodes transmit the movie fragments stored as GOP units to the clients. Each MMS node periodically sends its present working status to the HS node. This message

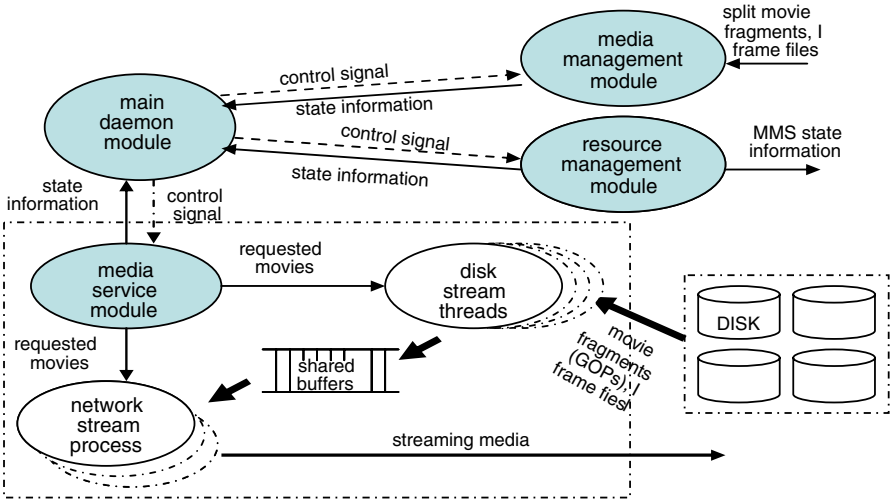

Fig. 3 Architecture of MMS nodes

operates as a heartbeat protocol between the MMS nodes and the HS node. Figure 3 shows the architecture of an MMS node. Each MMS node consists of a media_management module, a media_service module, a resource_management module and a main_daemon module.

The media_management module treats the various requests from both the striping module and the service_control module located in the HS node. In addition to storing movie fragments, the module also provides the removal and modification functions for the movie data.

The resource_management module collects information about the state of the MMS node such as CPU usage, memory, network and disks. The/proc file system in Linux is exploited for this purpose. This information is plugged into the heartbeat message packets and sent to the HS node. Since the size of this heartbeat message is 32 bytes, it does not affect the total performance of the VODCA system. Based on this information, the HS node supervises all MMS nodes and determines the admission control for the new client's requests.

In the figure, we can see that the media_service module consists of disk stream threads and network stream processes. For one client, a disk stream thread and a network stream process are allocated. When the HS node admits a new client, a network stream process is created, and a network connection for the client is established. After that, a disk stream thread is created and begins to retrieve the requested movie data. These movie data are loaded on shared buffers. The network stream process reads the movie data from the shared buffers and transmits them to the client.

The main_daemon module receives the control information from the HS node. According to the information, this module makes the MMS's internal control signals and sends them to the related modules.

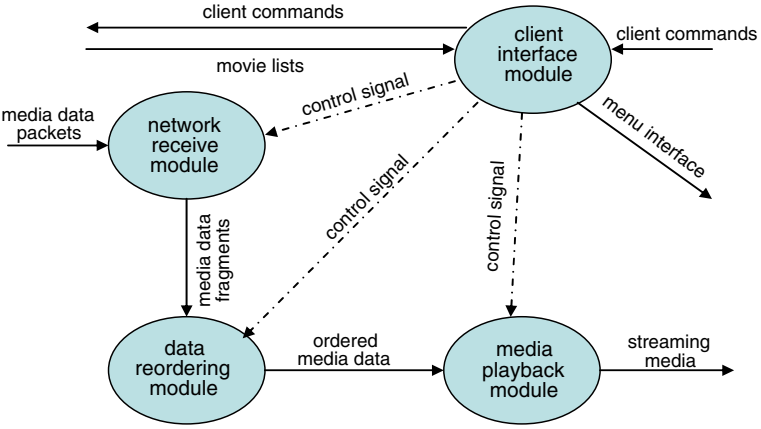

Fig. 4 Architecture of the client

3.1.3 Clients in the VODCA system

The client side in the VOD system provides for various client interfaces and notifies the VOD server of client requests. Figure 4 shows the client architecture in the VODCA system. It consists of a client_interface module, a network_receive module, a data_reordering module and a media_playback module.

The client_interface module delivers the client commands to the HS node and displays the movie lists and their information on the screen. This module also provides the graphic user interface and menu icons for clients. The network_receive module receives the movie data packets from the MMS nodes, and the received packets are merged into a movie fragment. The regenerated fragments are stored in the shared memory space. A fragment involves the GOP data and header information.

In the VODCA servers, each MMS node concurrently transmits the movie fragments to clients. Since the movie fragments are received in random order, the data_reordering module adjusts the order of these fragments according to the order of playback time. For this reordering, the sequence number within the header information is exploited. The arranged movie fragments are passed to the media_playback module via Linux's pipe mechanism. This module decodes the MPEG movie data and displays them on the screen.

3.2 Striping of Video Blocks

The cluster server easily provides a parallel computing environment based on the independent working spaces of the backend nodes and the high-speed network between them. To apply parallel processing for MPEG movies, we stripe the movie files according to the defined granularity policy. After striping, the movie file is partitioned into many fragments, and they are distributed to the backend nodes with their header information. As the granularity unit, we use the GOP size of MPEG movies as a striping

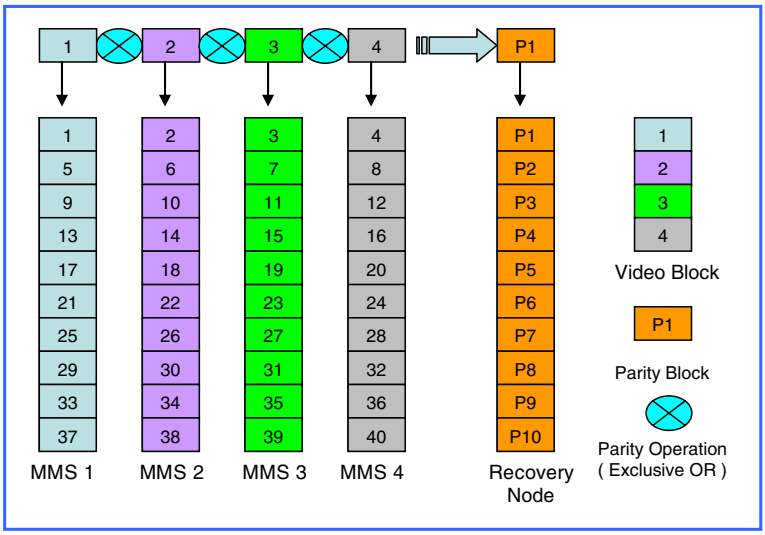

Fig. 5 Striped video blocks and parity blocks

unit. Since each GOP has approximately equal running time in MPEG streams, the MPEG movies are split into GOPs and distributed into each MMS node with their sequence number.

The RAID mechanism is usually used to recover the inaccessible video blocks stored in the failed MMS node. Various kinds of RAID levels exist in commercial areas. In our research, RAID-3 and 4 levels are used for the basic recovery system in our VODCA server. These levels are suitable for the video streaming service by supporting high-speed data transfer rates [11,12].

In the RAID-4 algorithm, the parity block for the same rank blocks should be generated on writes and recorded on the parity disk. Figure 5 shows the distributed video blocks across the MMS nodes and the parity blocks stored in a recovery node. For example, the parity block P1 stored in the recovery node is generated by the exclusive-OR operation to video blocks 1, 2, 3 and 4. RAID-4 reduces the number of disk assesses but suffers from the burden of aggregating the video blocks from the MMS nodes at once. To address this problem, when the video blocks of the MMS nodes are transferred to the recovery node, we use the RAID-3 algorithm that is based on the fine-grained unit. During the recovery process, the large video blocks stored in the MMS nodes are partitioned into small data units. After that, these data units are transferred to the recovery node, and they participate in the exclusive-OR operation to rebuild the failed video blocks.

3.3 Recovery Flow on RAID-3 and 4

Figure 6 shows the recovery flow operating on basic RAID-3 and 4 mechanisms [11,12]. We denote this recovery model as the Recovery System based on Basic RAID

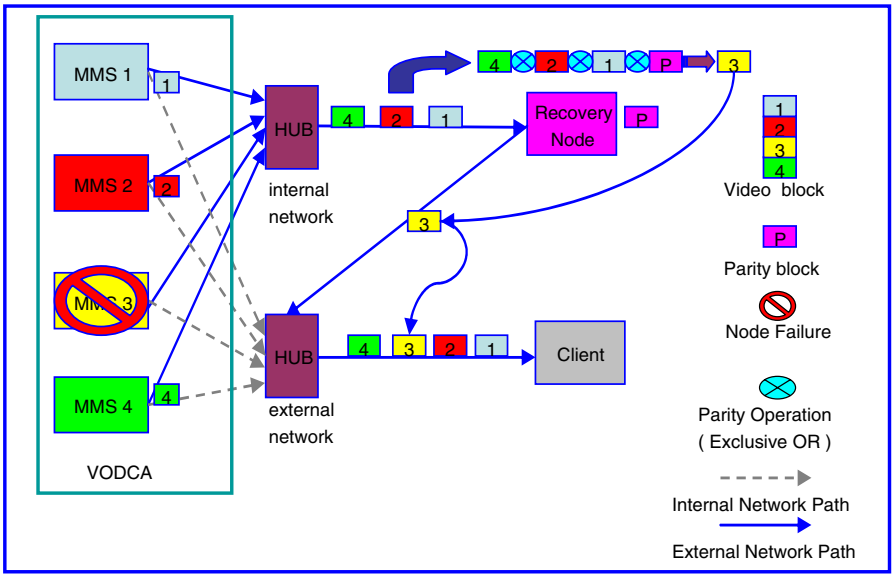

Fig. 6 Architecture of RS-BRM

Mechanisms (RS-BRM) and adopt it in the VODCA server as a basic recovery system. As shown in this figure, two network paths are used. One is an external network for connecting 4 MMS nodes and clients. Another network is an internal network path deployed between 4 MMS nodes and a recovery node.

When all the MMS nodes work normally, they transmit the stored video blocks to clients through the external network path. However, if an MMS node fails to work, the remaining MMS nodes begin to send their video blocks to the recovery node. To regenerate the video blocks stored in the failed MMS node, the recovery node uses the video blocks transmitted from the remaining MMS nodes and the parity blocks stored in its own disks. The internal network path is used for these recovery operations. The external network just takes charge of streaming services to the clients.

For example, as shown in Fig. 6, when MMS node 3 fails, the recovery node regenerates video block 3 by executing the exclusive-OR operation with received video blocks 1, 2 and 4 and its own parity block P. Regenerated video block 3 is transmitted to the client via the external network path. Therefore, the streaming media service could be supported uninterrupted as if nothing had happened. However, for recovery operations, all video blocks stored in the remaining MMS nodes should be transmitted to the recovery node at the same time. Due to this operation characteristic, the input network port of the recovery node suffers from traffic congestion. In addition, during the recovery period, since the remaining MMS nodes do not participate in the exclusive-OR operations to regenerate the failed video blocks, their CPU utilization is low.

4 Pipeline Computing to Recover Failed Video Blocks

4.1 System Architecture

To address the problems in the RS-BRM, we propose a new recovery method based on the pipeline computing concept [14]. It is denoted as Recovery System based on Pipeline Computing Mechanism (RS-PCM). The proposed method not only distributes the network traffic generated for the recovery operations to the remaining MMS nodes but also uses the available CPU time of the MMS nodes in the regenerating process of the video blocks.

In the parity-based RAID algorithm, exclusive-OR operations are necessary to regenerate the failed blocks. They bring out high computational burden in the recovery node. In addition, to rebuild the video blocks, exclusive-OR operations in several stages should be performed. Each stage of exclusive-OR operation need two video blocks stored in each MMS node at a time. However, due to the characteristics of exclusive-OR, the final result is independent of the processing order of stages. Therefore, even if the exclusive-OR operations are performed out of order, the same result is achieved. Based on this characteristic, the RS-PCM distributes the necessary exclusive-OR stages to both the remaining MMS nodes and the recovery node. Due to the advantage of load distribution, the RS-PCM can solve the network congestion problem on the input network port of the recovery node. In addition, the available CPU time of the MMS nodes can be fully used.

Figure 7 shows the flow of video blocks in the system architecture of the RS-PCM. It spreads the exclusive-OR workloads over all the MMS nodes except the failed MMS 3 node. The result blocks performing the exclusive-OR operations are sent to the neighboring MMS node successively like a chained list. In an MMS node, the video blocks received from the neighboring MMS node are either the raw video blocks stored in the

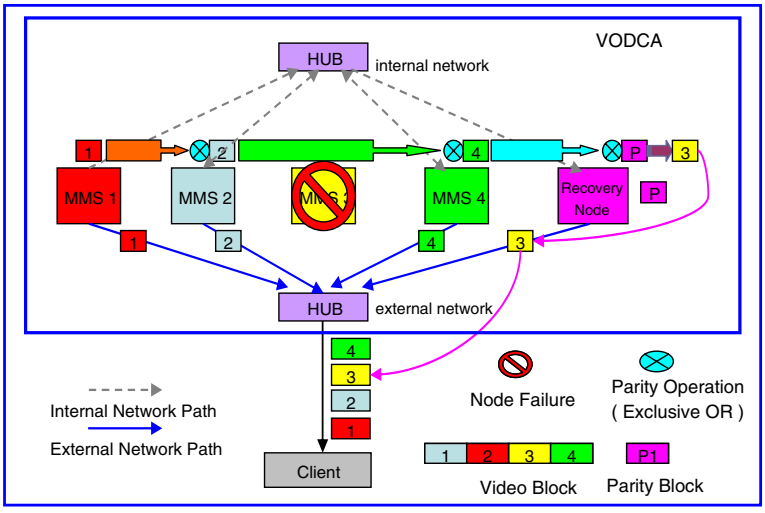

Fig. 7 System architecture of the RS-PCM

disks or the result block performs the pre-stage exclusive-OR operations. The recovery node receives the aggregate result and performs the last exclusive-OR operation with its parity block. Finally, the video block stored in the failed MMS node is rebuilt. It transmits to clients through the external network path.

For example, in Fig. 7, when MMS node 3 fails, MMS 1 node sends video block 1 to the MMS 2 node. The MMS 2 node performs the 1st exclusive-OR operation with both video blocks 1 and block 2 stored in its own disk. After that, the result is sent to the MMS 4 node to perform the exclusive-OR operation with video block 4. The MMS 4 node performs the 2nd exclusive-OR operation on the result block received from the MMS 2 node and its own video block 4. The result of the 2nd exclusive-OR operation is sent to the recovery node. Finally, the recovery node regenerates video block 3 by performing the exclusive-OR operation with the corresponding parity block.

4.2 Pipeline Computing

Figure 8 shows the recovery operations like the pipeline concept. As shown in this figure, every cycle marked in the bottom area, the RS-PCM simultaneously performs the loading of video blocks and the exclusive-OR operation and the transmission of the computed results. It is similar to the pipeline concept of instructions [14]. In this figure, the MMS 3 node enters a fault state, and it has video blocks 3, 7, 11, 15, 19 and 23. These video blocks are regenerated in the recovery node every cycle. The recovery node executes the exclusive-OR operation with a received block and a parity block each cycle.

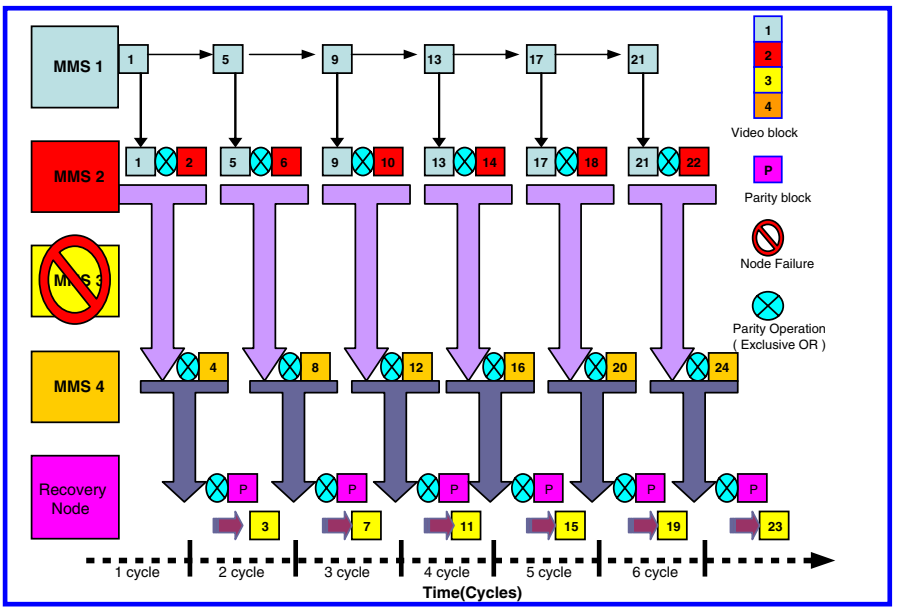

Fig. 8 Recovery steps in every Cycle

As applied to the pipeline computing concept, the RS-PCM distributes not only the internal network traffic but also the computational load for exclusive-OR operations into all MMS nodes. The input network traffic of the recovery node is equal to the output traffic of one MMS node. Each MMS node also has the same amount of internal network traffic. The recovery node and MMS nodes could utilize the full capacity of the internal network path. If the $n - 1$ MMS nodes are working and the output traffic of an MMS node is m, only m network traffic exists on the input port of the recovery node. Since the input network traffic of the recovery node is limited to the amount of output traffic from a neighboring MMS node, network congestion on the input port of the recovery node could be avoided.

4.3 Algorithm Behaviors

Figure 9 shows the algorithm behaviors of the MMS nodes when the RS-PCM is applied. For a VOD client, a disk stream thread and a network stream process are needed. The disk stream thread retrieves the video blocks from the disks and loads them into the disk buffers. After the disk buffers are full, these video blocks are moved into the streaming buffers for transmission to the client. In this figure, the diagrams shown in white on the left side indicate the situation when all the MMS nodes work in the normal state. On the other hand, the diagrams on the right side shown in gray represent the recovery behaviors in the RS-PCM. In the recovery behaviors, if an MMS node fails in the fast-forward and fast-rewind play modes, the recovery steps are bypassed because these VCR-like functions can be supported with the video blocks stored in the remaining MMS nodes.

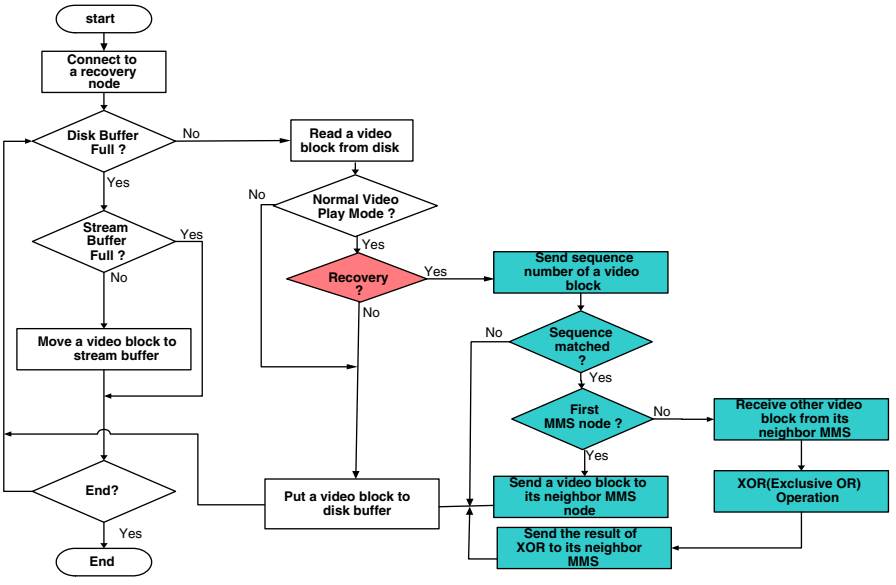

Fig. 9 Operations of MMS nodes

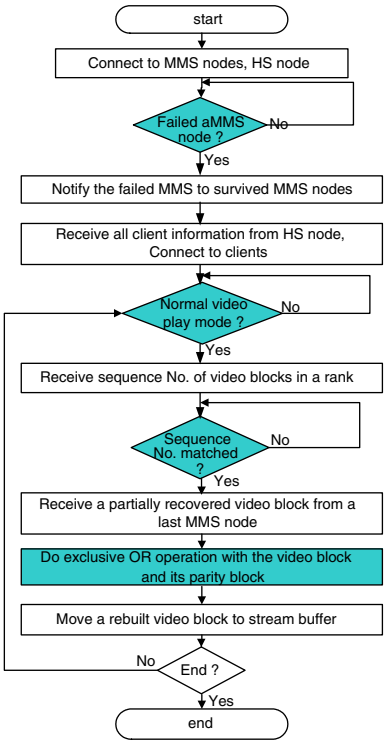

Fig. 10 Recovery node operations

The diagram including the "sequence-matched ?" checks that a video block belongs to the same rank for the exclusive-OR operation. The sequence number of a video block represents the position of the current MMS node among all the MMS nodes. In the gray diagrams at the bottom right of the figure, if the current MMS node is the first MMS node, this MMS node just sends its own video block to the neighboring MMS node without the exclusive-OR operation. On the other hand, if this is not the first MMS node, they perform the exclusive-OR operations with the video blocks received from the neighboring MMS node and their own video blocks.

Figure 10 shows the algorithm behaviors of the recovery node. The recovery node waits until an MMS node enters the fault state. When the recovery node detects the failure of an MMS node, the recovery node sends information about the failed MMS node to the remaining MMS nodes. And the recovery node acquires client information from the HS node, such as the IP address, the playing position of the serviced movies and the play modes (normal, fast forward, fast rewind). Based on the information, the recovery node can connect to the clients and maintain uninterrupted streaming media service.

As shown in the diagram that includes the "normal video play mode?," the RS-PCM advances the recovery steps when the normal play mode is serviced. The recovery node checks that the sequence number of the video blocks belongs to the same rank for the exclusive-OR operation. After that, the recovery node receives

the video blocks from the last neighboring MMS node. These video blocks include the results of exclusive-OR operations computed by the remaining MMS nodes. Finally, by performing the exclusive-OR operations with the parity block, the recovery node rebuilds the video blocks stored in the failed MMS node. The rebuilt video blocks are moved to the stream buffers.

5 Implementation for Experiment

5.1 System Configuration

The VODCA server for our experiments consists of an HS node, 4 MMS nodes and a recovery node. Each node operates on the Linux operating system. The MMS nodes, HS node and clients are connected via a 100 Mbps Ethernet switch. All MMS nodes and the recovery node are also connected via the internal network path constructed by a 100 Mbps Ethernet switch.

All applications including the system administrative tools of the HS node were developed on Qt, C and C++ libraries. Table 1 shows the hardware components for each MMS node in the VODCA system. Table 2 shows the detailed specifications of the movies used in our experiments. They are MPEG-2 movies and have enough running time to evaluate the performance of our system.

5.2 Load Generator and Virtual Clients

We use the yardstick program to measure the performance of our cluster-based VOD server [15]. The yardstick program consists of the virtual load generator and the virtual client daemon. The virtual load generator works on the HS node and generates client requests based on the Poisson distribution with $\lambda = 0.25$ [16]. These requests are sent to the MMS nodes.

Table 1 Specifications of MMS nodes and a recovery node

CPU	Intel pentium 4, 3.0 GHz
Memory	1 GByte DDR
Disk	Seagate barracuda ATA IV 40 GB 7200 RPM × 2
OS	RedHat 7.3 (Kernel 2.4.18)
Network	100 Mbps fast Ethernet, 100 Mbps Ethernet switch with 24 ports

Table 2 Specifications of experimental movies

Movie name	John Q	Ice age
Frame size (H × V)	352 × 288	352 × 288
Frame rates (number/s)	25	25
Bit rates (bps)	1,437.6	1,437.6
Running time (min)	110	85
GOP size (Kbytes)	124.1	120.8

The virtual client daemon is located in test-bed PCs for clients. It plays the role of receiving movie data from the MMS nodes. Based on MPEG-1, 2 specifications, we assume that a QoS stream requires 1.5 Mbps network bandwidth. To support this QoS criterion, the virtual client daemon measures the time elapsed for receiving 1.5 Mbits. If the elapsed time is below 1 s, the virtual client daemon stays in an idle state until a 1-s period elapses. After exhausting the remaining period, the daemon wakes up and begins to receive the next data. Due to this waiting mechanism to satisfy the designated bit rate, a virtual client daemon can work as a real client. However, if our virtual client daemon receives the movie data below the 1.5 Mbps rate, then it is likely that the MMS nodes suffer from the overloaded state due to too many clients. This result means that the streaming services in progress cannot satisfy the QoS requirement for clients. Therefore, since MMS nodes do not provide the QoS streams for clients, the streaming services may be automatically interrupted. We implemented the virtual client on our test-bed PCs. In our experiments, we found that a PC had a role for 30 virtual clients.

5.3 Performance Metrics

We use the variation of network traffic driven from MMS nodes as well as that of network traffic in the input port of a recovery node as the performance metrics. In addition, after an MMS node fails, the average Mean Time to Recovery (MTTR) is chosen. From the implemented environment, when an MMS node fails, the reactions of the remaining MMS nodes are observed, and the total number of QoS streams is evaluated based on the performance metrics.

6 Performance Evaluation

6.1 Network Traffic

Figure 11 shows the network traffic of an MMS node and the recovery node when the RS-BRM is applied. The output network traffic from an MMS node is 12 MB/s. This means that 4 MMS nodes provide 256 streams (12 M * 8 bit * 4 MMS nodes/1.5 Mbps = 256). The failure of an MMS node takes place at the 120 s on the time line. After an MMS node fails, the remaining MMS nodes begin to transmit the video blocks to the recovery node. The amount of total network traffic from the remaining 3 MMS nodes is 36 MB/s. However, as mentioned in Table 1, the maximum input network bandwidth of the recovery node is 12.5 MB/s (100 Mbps). Therefore, due to the excessive input data, the input network port of the recovery node suffers from the bottleneck phenomenon.

To address this problem, the remaining MMS nodes automatically decrease the number of video blocks transferred to the recovery node. As shown in this figure, after an MMS node enters a fault state, the output traffic from the remaining MMS nodes to the recovery node drops to 4 MB/s rates. In this figure, they are represented as rectangles. According to the regulation of output traffic in MMS nodes, the input traffic of the recovery node is measured at 12 MB/s rates. It is marked as the x shapes

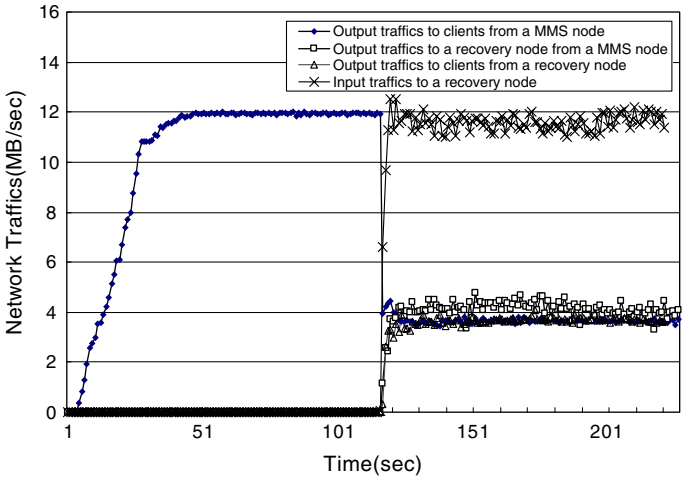

Fig. 11 Network traffic in the RS-BRM

in this figure. Therefore, the recovery node receives the video data of 12 MB/s from the remaining 3 MMS nodes and rebuilds the video blocks stored in the failed MMS node.

From the triangles, we can find that the recovery node transfers the rebuilding video blocks to the clients at 4 MB/s rates. Depending on the amount of the output traffic from the recovery node to the clients, after an MMS node fails, the output traffic from the remaining MMS nodes to the clients is also sustained at 4 MB/s rates. The 4 MB/s traffic means that only 85 clients can be supported (4 M *8 bit * 4 MMS nodes/1.5 Mbps = 85). As a result, after an MMS node fails, the RS-BRM method can support only 33.2% of the original 256 streams.

Figure 12 shows the network traffic in an MMS node and the recovery node when the RS-PCM is applied. Before an MMS node fails, 256 streams are supported in this method. An MMS node fails at 120 s on the time line. After an MMS node fails, the first MMS node transmits its own video blocks to a neighboring MMS node. The other MMS nodes execute exclusive-OR operations with the video blocks transmitted from a neighboring MMS node. The results are transmitted to the neighboring MMS node successively. From the circles and rectangles in this figure, we can find that the amount of input traffic from the neighboring MMS node is almost equal to that of its own output traffic transmitted to the neighboring MMS node.

As shown in this figure, the input traffic from the last MMS node to the recovery node reaches 12 MB/s rates. According to the amount of input network traffic, the recovery node can also rebuild the video blocks and transmit the regenerated video blocks to clients. The triangles show that the output traffic from the recovery node to the clients uses 12 MB/s rates. These transfer rates of the rebuilt video blocks are equal to the output traffic rates from the remaining MMS nodes to the clients. As a result, even if an MMS node fails, since the video block transfer rates from the recovery node and the remaining MMS node are 12 MB/s, streaming media service for 256 clients can be continued (12 M * 8 bit * 4 MMS nodes/1.5 Mbps = 256).

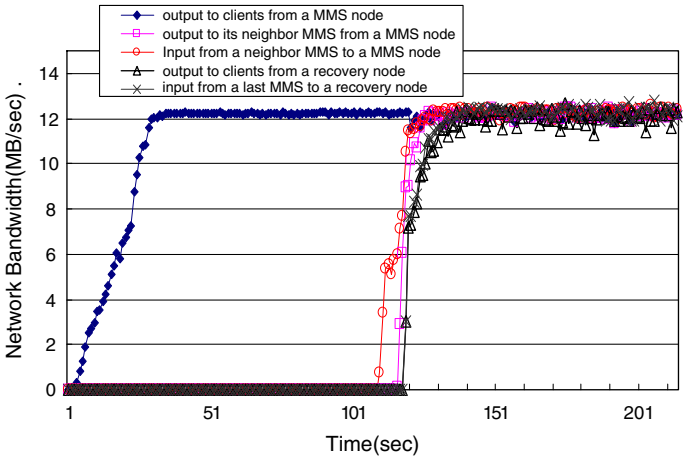

Fig. 12 Network traffic in the RS-PCM

The experimental results showed that the RS-PCM not only distributed the network traffic generated in the recovery operations but also utilized the available CPU resources of the MMS nodes. Due to these advantages, the RS-PCM resulted in a better performance than the RS-BRM. In the experimental results, after an MMS node failed, the RS-PCM sustained 256 clients continuously, but the RS-BRM supported only 85 clients. When compared to the RS-BRM, the RS-PCM provides about 3 times unceasing QoS streams in the same working environment.

6.2 Mean Time to Recovery (MTTR)

Figure 13 shows the reading times of 1 GOP in the client side while the streaming service is in progress under the RS-BRM. The experiments are performed between the 7 MB/s and the 12 MB/s loads. Although an MMS node fails at 120 s on the time line, the fluctuation of the GOP reading times starts at the 156-s position. The time delay is due to the network delay and the buffering time in the client side. As shown in this figure, when all MMS nodes work normally, the average reading time is about 0.65 s, and it maintains a steady state. However, after an MMS node fails, the reading times vary. These variations are due to the loss of packet data, the initial setup time of the recovery algorithm and the traffic congestion in the recovery node. In particular, the fluctuation rates are high at the 11 MB/s and the 12 MB/s load. Under these workloads, unstable oscillations of the GOP reading times emerged between 156 and 266 s. The difference between the maximum GOP reading time and the minimum GOP reading time is about 1.18 s. After the fluctuation period passes through, the recovery node is working normally, and the GOP reading times converge to the 0.65 s level again. The MTTR value is 110 s. It can be regarded as an impatient period to VOD clients [2,3].

Figure 14 represents the GOP reading times in the RS-PCM. The failure of an MMS node takes place at 120 s on the time line. Since the network delay has emerged between the server and clients, the fluctuation of the GOP reading time on the client

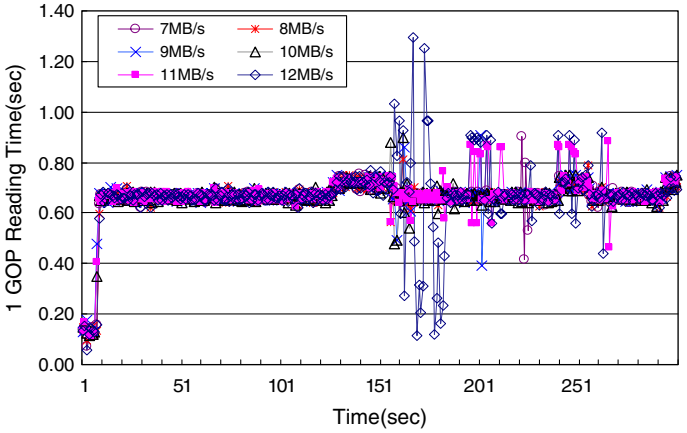

Fig. 13 GOP reading time in the RS-BRM

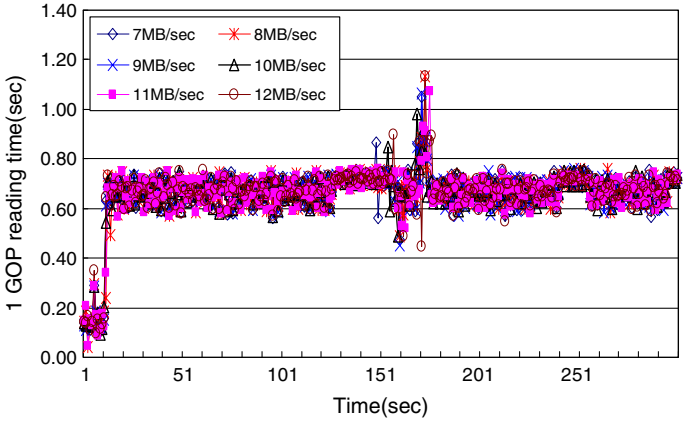

Fig. 14 GOP reading time in the RS-PCM

side begins at 148 s and finishes at 176 s. After the agitation state, the reading times promptly converge to the steady state of 0.65 s. The fluctuation period is just 28 s. When compared to the RS-BRM, the period of the fluctuation is very short. Since the recovery operations are distributed into all MMS nodes, the recovery node can transmit the rebuilt video blocks within a relatively short time. The fluctuation period of RS-PCM is 4 times shorter than the RS-BRM. The difference between the maximum GOP reading time and the minimum GOP reading time is 0.68 s. This result is the half of the difference issued by the RS-BRM. In the RS-PCM, since the period of fluctuations and the vibration amplitudes are very short, we confirm that the RS-PCM results in a much better MTTR value than the RS-BRM.

7 Conclusion

In this paper, recovery mechanisms in cluster-based VOD servers were studied to support QoS streams when a backend node fails. To study the recovery methods in the actual VOD system, we implemented a cluster-based VOD server called the VODCA. Based on the VODCA system, the RS-BRM was designed with the advantage of RAID-4 in disk retrieving speed and the advantage of RAID-3 in effective memory usage. In the RS-BRM, we found that the input network port of a recovery node was easily saturated with the video blocks transmitted from the remaining MMS nodes. The reduction in the number of video blocks transmitted to the recovery node caused a reduction in the number of serviced clients and the degradation of video stream quality. In addition, the RS-BRM showed the inefficient CPU usage of the MMS nodes. To address these issues, we proposed the RS-PCM based on the pipeline computing concept. In the RS-PCM, the recovery node rebuilt the video blocks stored in the failed MMS node every cycle. This mechanism is similar to the pipeline process of instructions. The RS-PCM made efficient use of the available CPU time of the MMS nodes so that all MMS nodes participated in the recovery procedures to rebuild the video blocks stored in the failed MMS node. Based on this pipeline computing, the RS-PCM distributed not only the network traffic needed for the recovery process but also balanced the computation loads driven by exclusive-OR operations. From our experiment, we observed that the RS-PCM supported 3 times more QoS streams than the RS-BRM.

For commercial VOD services to be successful, the fluctuation period of unstable streaming service due to failure events should be short. In the GOP reading time of clients, the RS-PCM showed a 4-time shorter fluctuation period than the RS-BRM. Since the RS-PCM quickly recovered the failure state of the VOD servers, the streaming media services in progress could be sustained without degradation of media quality. The experiments showed that the RS-PCM had an appropriate MTTR value to deal explicitly with failure events in cluster-based streaming media servers.

Acknowledgements This research was financially supported by the Ministry of Education, Science Technology (MEST) and Korea Industrial Technology Foundation (KOTEF) through the Human Resource Training Project for Regional Innovation.

References

1. Sitaram, D., Dan, A.: Multimedia Servers: Applications, Environments, and Design. Morgan Kaufmann Publishers, San Francisco (2000)
2. Fox, A., Patterson, D.: Approaches to recovery oriented computing. IEEE Internet Comput. **9**(2), 14–16 (2005). doi:10.1109/MIC.2005.39
3. Tang, D., Zhu, J., Andrada, R.: Automatic generation of availability models in RAScard. In: IEEE International Conference of Dependable Systems and Networks, June 23–26, pp. 488–494 (2002)
4. Sarhan, N.J., Das, C.R.: Caching and scheduling in NAD-based multimedia servers. IEEE Trans. Parallel Distrib. Syst. **15**(10), 921–933 (2004). doi:10.1109/TPDS.2004.49
5. Kang, S., Yeom, H.Y.: Modeling the caching effect in continuous media servers. Multimedia Tools Appl. **23**(3), 203–224 (2003). doi:10.1023/A:1025702332314
6. Shenoy, P.J., Goyal, P., Vin, H.M.: Data storage and retrieval for video-on-demand servers. In: IEEE Fourth International Symposium on Multimedia Software Engineering, pp. 240–245, December 2002

7. Gafsi, J., Biersack, E.W.: Modeling and performance comparison of reliability strategies for distributed video servers. IEEE Trans. Parallel Distrib. Syst. **11**(4), 412–430 (2000). doi:10.1109/71.850836
8. Gafsi, J., Biersack, E.W.: Data striping and reliability aspects in distributed video servers. Cluster Comput.: Netw. Softw. Tools Appl. **2**(1), 75–91 (1999)
9. Bolosky, W.J., Pitzgerald, R.P., Draves, J.H.: Distributed schedule management in the tiger video fileserver. In: Proceedings of the Sixteenth ACM Symposium on Operating Systems Principles, Saint Malo France, pp. 212–223, 05–08 October 1997
10. Chang, T., Shim, S., Du, D.: The designs of RAID with XOR engines on disks for mass storage systems. In: IEEE Mass Storage Conference, pp. 181–186, 23–26 March 1998
11. Merchant, A., Yu, P.S.: Analytic modeling and comparisons of striping strategies for replicated disk arrays. IEEE Trans. Comput. **44**, 419–433 (1995). doi:10.1109/12.372034
12. Holland, M., Gibson, G., Siewiorek, D.: Architectures and algorithms for on-line failure recovery in redundant disk arrays. J. Distrib. Parallel Databases **2**, 295–335 (1994). doi:10.1007/BF01266332
13. Seo, D., Lee, J., Jung, I.: Resource consumption-aware QoS in cluster-based VOD servers. J. Syst. Archit. **53**(1), 39–52 (2007)
14. Patterson, D.A., Hennessy, J.L.: Computer Organization & Design, pp. 392–490, Kaufmann (1998)
15. Schmidt, B.K., Lam, M.S., Northcutt, J.D.: The interactive performance of SLIM: a stateless, thin-client architecture. ACM Symposium on Operating Systems Principles, pp. 31–47 (1999)
16. Choi, J.-M., Lee, S.-W., Chung, K.-D.: A multicast delivery scheme for VCR operations in a large VOD system. In: 8th IEEE International Conference on Parallel and Distributed Systems, pp. 555–561, 26–29 June 2001