J Supercomput
DOI 10.1007/s11227-009-0305-6

Parallel failure recovery techniques in cluster-based media servers

Joahyung Lee · Inbum Jung

Abstract For large-scale video-on-demand (VOD) service, cluster servers are highlighted due to their high performance and low cost. A cluster server consists of a front-end node and multiple backend nodes. Though the increase in backend nodes provides more quality of service (QoS) streams, the possibility of backend node failure is proportionally increased. The failure causes not only the cessation of streaming services but also the loss of current playing positions. In this paper, when a backend node fails, recovery mechanisms are studied to support the streaming service continuously. Without considering the characteristics of cluster-based servers and MPEG media, the basic redundant array of independent disks (RAID) techniques cause a network bottleneck in the internal network path and demonstrate inefficient CPU usage in backend nodes. To address these problems, a new failure recovery mechanism is proposed based on the pipeline computing concept. The proposed method not only distributes the internal network traffic generated from the recovery operations but also utilizes the CPU time available in the backend nodes. In the experiments, even if a backend node fails, the proposed method provides continuous streaming media services within a short MTTR value as well as more QoS streams than the existing method.

Keywords Streaming media · Parallel failure recovery · Cluster servers · QoS stream

1 Introduction

Recent advanced computer and communication technologies have provided economically feasible multimedia services such as VOD (video on demand), digital libraries,

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

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

Published online: 27 May 2009

and e-learning. In these multimedia services, since the video data are given a great deal of weight due to the preference of clients, the VOD service is the most prominent multimedia application [9, 17]. Contrary to traditional file servers, VOD servers are subject to real-time constraints while storing, retrieving, and delivering movie data to the network. Since ceasing and jittering streaming videos are meaningless for VOD clients, the streaming media should satisfy the designated quality of service (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 MTTR (*m*ean *t*ime *to repair*) values [4, 18].

The cluster server architecture has been used in the areas of web, database, gaming, and VOD servers [6]. This architecture has an advantage of the ratio of performance to cost and is easily extended from general PC equipment. The cluster server architecture usually consists of a front-end node and multiple backend nodes. When this server architecture model is used as a VOD server, the video data are distributed to several backend nodes. The performance of the storage devices could be achieved accordingly as the number of backend nodes increases. However, even if the cluster server can be scaled by just adding new backend nodes, the fault rate of the backend nodes also increases in proportion to the number of backend nodes. Faulty backend nodes cause not only the cessation of streaming services but also the loss of the current serviced positions of all playing movies.

To sustain QoS streams despite the failure of a backend node, recovery mechanisms should be prepared for dealing with realistic VOD services. In this paper, the recovery mechanisms in cluster-based VOD servers are studied to support QoS streams while a backend node fails. 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. In experiments, even if a backend node fails, the proposed method provides continuous streaming media services within a short MTTR value as well as more QoS streams than the existing method.

The rest of this paper is organized as follows. Section 2 describes related work. Section 3 explains our cluster-based VOD servers and the management of video blocks. 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 [10, 13, 16]. 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 on sustaining 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 [5, 6]. The Tiger video server was implemented in the mirror-based storage system for VOD service [1]. RMD (rotational mirrored declustering) techniques were suggested for recovering failed disks or individual nodes [2]. However, these mirror-based approaches use disk storage inefficiently and incur the heavy burden of the recovery node.

RAID (redundant array of inexpensive (or independent) disks) 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 a parity-based recovery algorithm [5, 7, 11]. 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, 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 operation. 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 [8]. 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 [4, 18]. 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.

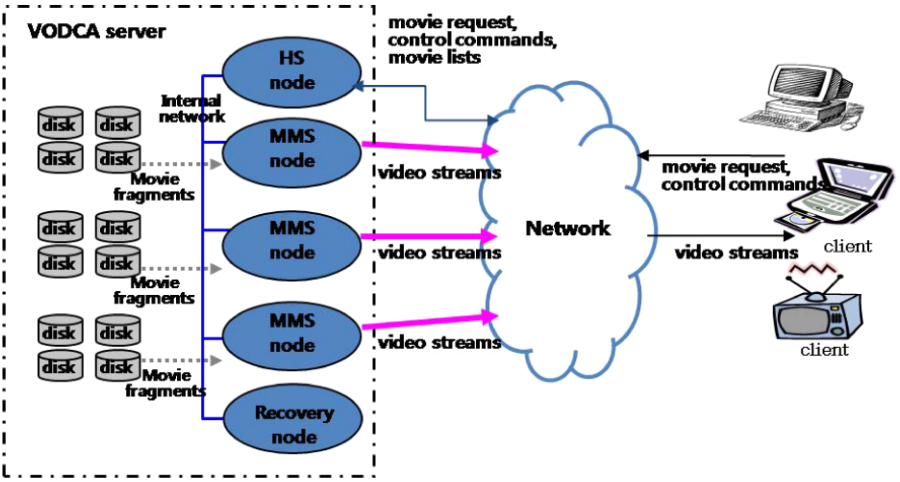

Fig. 1 Architecture of the proposed VOD service

3 Cluster-based VOD server

3.1 Architecture of VODCA

For large-scale VOD services, we implemented a cluster-based VOD server called VODCA (Video On Demand on Clustering Architecture) [15]. The VODCA consists of a front-end node called the HS (Head-end Server) and several backend nodes known as MMSs (Media Management Servers). 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 MMS nodes, they exchange working states and internal commands with each other.

The HS node not only receives clients' requests but also manages MMS nodes to support QoS with the admission control policy. When new MPEG movies are enrolled, the HS splits them and distributes them into each MMS node. The MMS nodes transmit stored movie fragments to clients under the supervision of the HS node. Each MMS node periodically sends its present working status to the HS node. This message operates as a heartbeat protocol between the MMS nodes and the HS node. When an MMS node fails, a recovery node operates to rebuild the video blocks stored in the failed MMS node. In our architecture, if the independent recovery node does not work, the HS node could operate as the recovery node.

3.2 Striping of video blocks and RAID algorithms

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, the group of pictures (GOP) size of MPEG movies is used as a striping unit. Since each GOP has approximately equal running time in MPEG streams, the MPEG movies are split into GOPs and distributed to each MMS node with their sequence number.

In the VOD service, admission control is required to guarantee the quality of streaming media for the service period. For example, when a new client arrives, if the streaming media services in progress cannot provide the required bit rates, the admission control rejects the entrance of a new client. Admission control is one of the necessary requirements for the commercial VOD service. To support admission control, the maximum number of QoS streams should be estimated based on the available resources in the MMS nodes. In our VOD system, the HS node determines the admission control for this new client.

The RAID algorithms are usually used to recover the inaccessible video blocks stored in a failed MMS node. Various kinds of RAID levels exist in commercial areas. RAID-5 has been used in many fields such as the database server, web server, game server, and so on. However, since RAID-5 spreads out the parity blocks of the same rank across all MMS nodes, an accurate number of QoS streams cannot be estimated during the recovery process. When an MMS node fails, all remaining MMS nodes participate individually in the complete recovery operations. The remaining MMS nodes receive video blocks of the same rank from other MMS nodes and perform the exclusive-OR operations to regenerate the failed video blocks. Since the receipt of video blocks occurs simultaneously, the input network port of each MMS node suffers from the bottleneck phenomenon. Due to the delay of receiving video blocks, the actual load of the MMS nodes cannot be estimated in the recovery period. Without the accurate admission control metric, the quality of streaming services in progress cannot be guaranteed in the remaining service time. Therefore, RAID-5 cannot sustain steady QoS streams for the recovery period.

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 [7, 11]. These methods use a recovery node to regenerate the failed video blocks. Based on the independent recovery node, the estimation of the available resources in the MMS nodes can be accurately calculated. From the estimation value, we can design the admission control method so that steady QoS streams can be provided even in the faulty state.

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 2 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

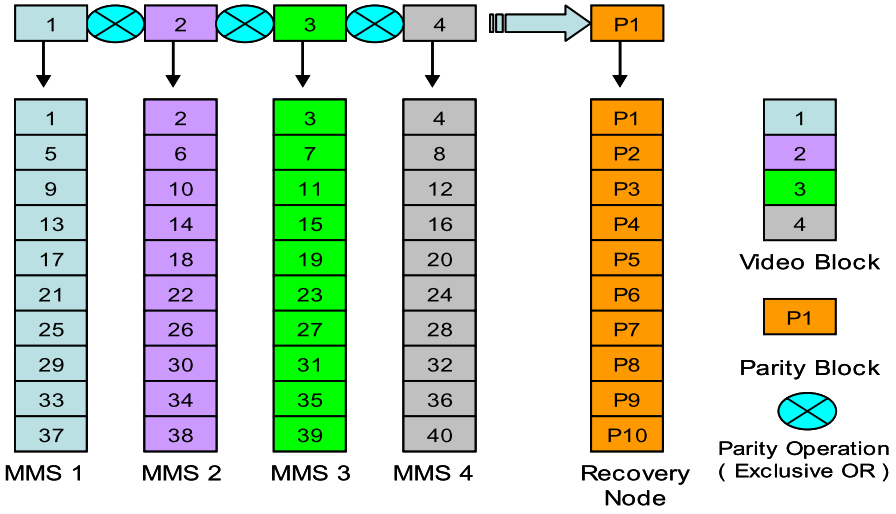

Fig. 2 Striped video blocks and parity blocks on RAID-4

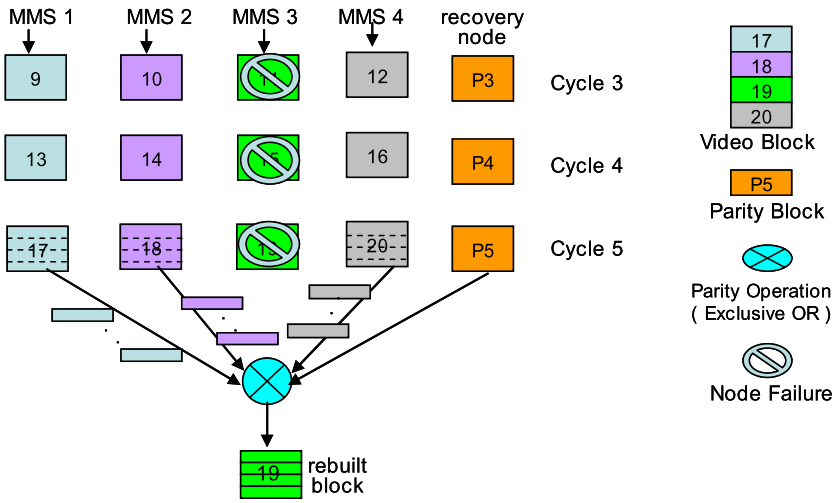

Fig. 3 Video blocks aggregation on RAID-3

video blocks. Figure 3 shows that the recovery node regenerates the failed blocks by performing exclusive-OR operations to the transferred small striped data. In Fig. 3, when the MMS 3 node fails, video block 19 stored in the failed MMS 3 node is rebuilt by the exclusive-OR operation with a parity block P5 and video blocks 17, 18, and 20. As shown in Fig. 3, these video blocks are striped across the MMS nodes as small data units. During the recovery operations, these data units aggregate in the recovery node to rebuild the failed video blocks.

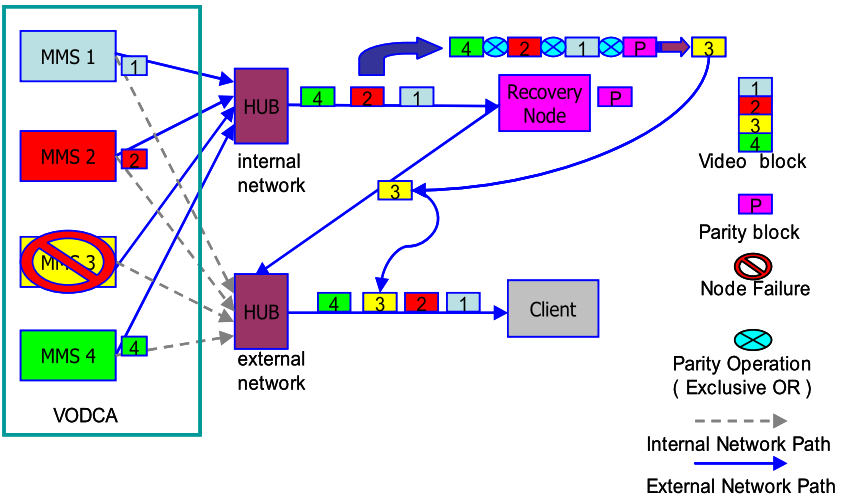

Fig. 4 RS-BRM architecture

3.3 Recovery flow on RAID-3 and 4

Figure 4 shows the recovery flow operating on basic RAID-3 and 4 mechanisms [7, 11]. We denote this recovery model as the RS-BRM (Recovery System based on Basic RAID Mechanisms) and adopt it in the VODCA server as a basic recovery system. As shown in Fig. 4, 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. 4, 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, the MMS nodes' CPU utilization is low.

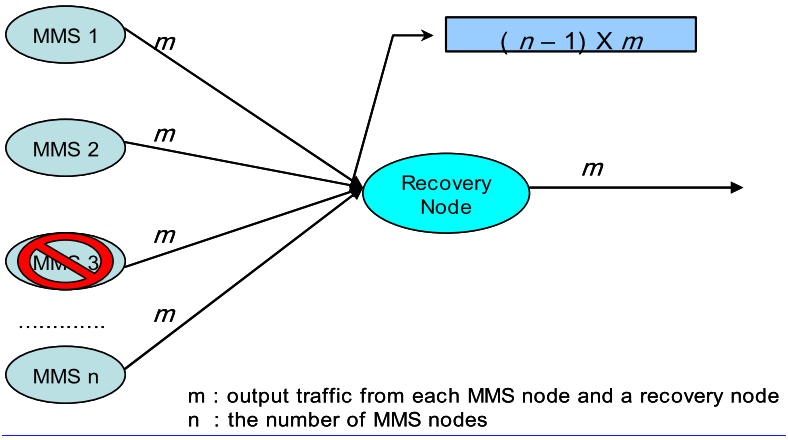

Fig. 5 Network traffic in RS-BRM

3.4 RS-BRM traffic analysis

Figure 5 shows the network traffic to perform the recovery operation when the MMS 3 node fails. If the $n-1$ MMS nodes remain and the output traffic of each MMS node is m, the network traffic of $(n-1) \times m$ is transmitted to the input network port of the recovery node. If the input network capacity of the recovery node is B, the m value is driven as $m = B/(n-1)$. For example, if there are 5 MMS nodes and the maximum capacity of the input network port is 100 Mbps, then m is 25 Mbps $=$ 100 Mbps$/(5-1)$. To avoid the bottleneck phenomenon in the network input port of the recovery node, each MMS node should transmit the video data below the 25-Mbps rate. If the data transfer rates of the MMS nodes are over 25 Mbps, the input port of the recovery node has the bottleneck phenomenon. Due to this effect, the recovery node cannot treat all the video data transmitted from the MMS nodes, so the MMS nodes should decrease the data transfer rates transmitted to the recovery node. As a result, the MMS nodes decrease the data transmitting rates to the clients for synchronizing with the data transfer rates to the recovery node. As shown in this equation, the input network traffic of the recovery node is dependent on the number of MMS nodes. This limitation should be considered in the design of the recovery system in the cluster-based VOD server.

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 [12]. It is denoted as RS-PCM (***r***ecovery ***s***ystem based on ***p***ipeline ***c***omputing ***m***echanism). The proposed method not only distributes the network traffic generated for the recovery operations to the remaining MMS nodes

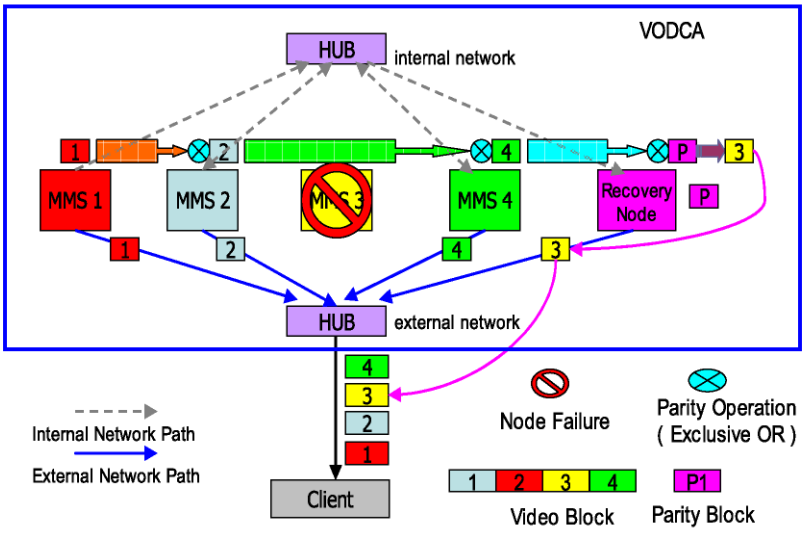

Fig. 6 RS-PCM system architecture

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 a 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 the exclusive-OR operation needs 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 the 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 6 shows the flow of video blocks in the RS-PCM system architecture. It spreads the exclusive-OR workloads over all the MMS nodes except for 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 disks or the result block performing 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. 6, when the MMS node 3 fails, the 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

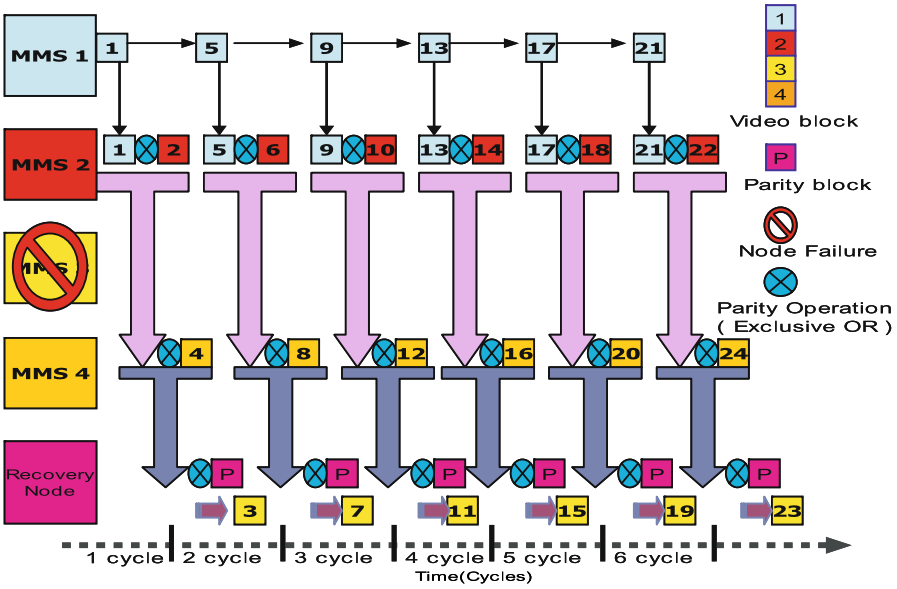

Fig. 7 Recovery steps in every cycle

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 7 shows recovery operations such as the pipeline concept. As shown in Fig. 7, 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 instructions concept [12]. In Fig. 7, 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.

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 to 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 the 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

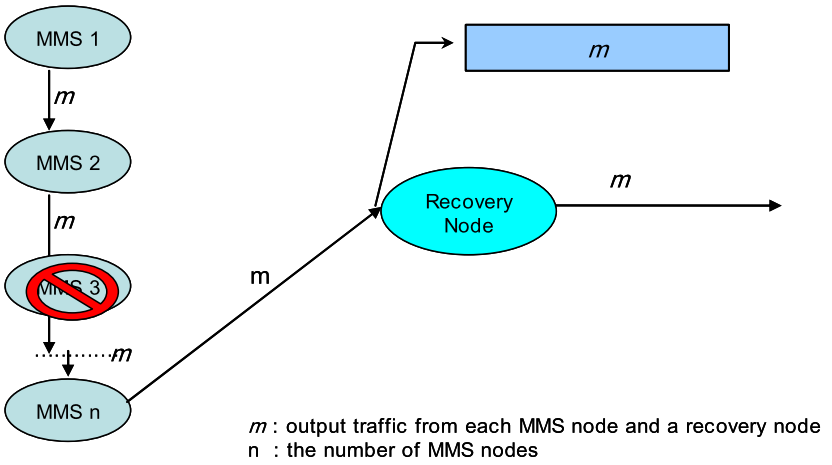

Fig. 8 Network traffic in RS-PCM

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 RS-PCM traffic analysis

The recovery node in the RS-PCM executes an exclusive-OR operation just one time for each cycle and sends the result to the client. The proposed RS-PCM distributes not only the computation load for exclusive-OR operation but also the network traffic to all MMS nodes. The input network traffic to the recovery node is equal to the output traffic of one MMS node. Figure 8 shows the network traffic in the RS-PCM. Each MMS node has the same amount of network traffic as its output m. The recovery node and MMS nodes could utilize the network bandwidth capacity fully. As shown in Fig. 8, even if the $n-1$ MMS nodes remain and each output traffic is m, only the m network traffic exists on the input port of the recovery node. Unlike the RS-BRM, the input network traffic of the recovery node does not depend on the number of MMS nodes. Therefore, the input network port of the recovery node does not suffer from the bottleneck phenomenon. As a result, since both the MMS nodes and the recovery node can completely use their computing resources, their maximum number of QoS streams can be achieved.

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

Table 1 Specifications of the 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 the movies used in the experiments

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

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 [14]. 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 $\lambda = 0.25$ [3]. These requests are sent to the MMS nodes.

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 and 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 less than 1 second, the virtual client daemon stays in an idle state until a 1-second 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 MTTR (mean time to recovery) 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 9 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/sec. This means that 4 MMS nodes provide 256 streams ($12\ \mathrm{M} * 8\ \mathrm{bit} * 4$ MMS nodes$/1.5$ Mbps $= 256$). The failure of an MMS node takes place at 120 seconds 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/sec. However, as listed in Table 1, the maximum input network bandwidth of the recovery node is 12.5 MB/sec (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 Fig. 9, after an MMS node enters a fault state, the output traffic from the remaining MMS nodes

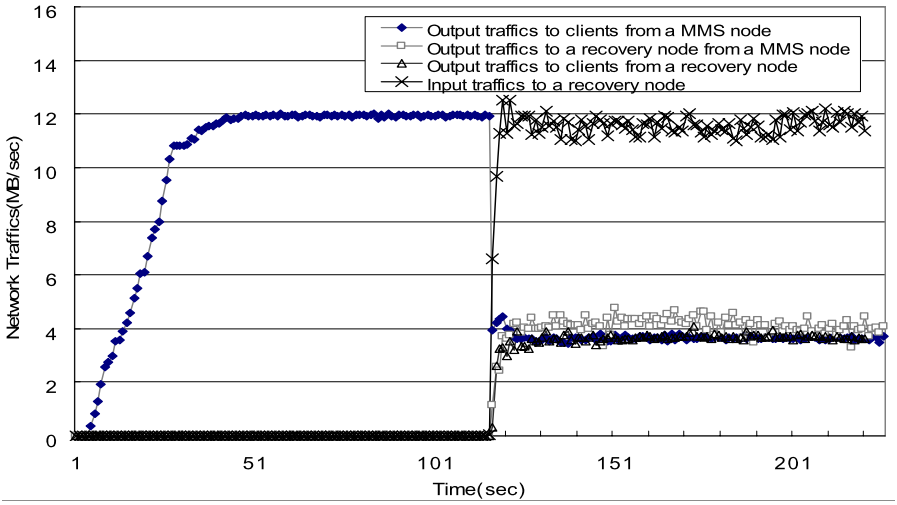

Fig. 9 Network traffic in the RS-BRM

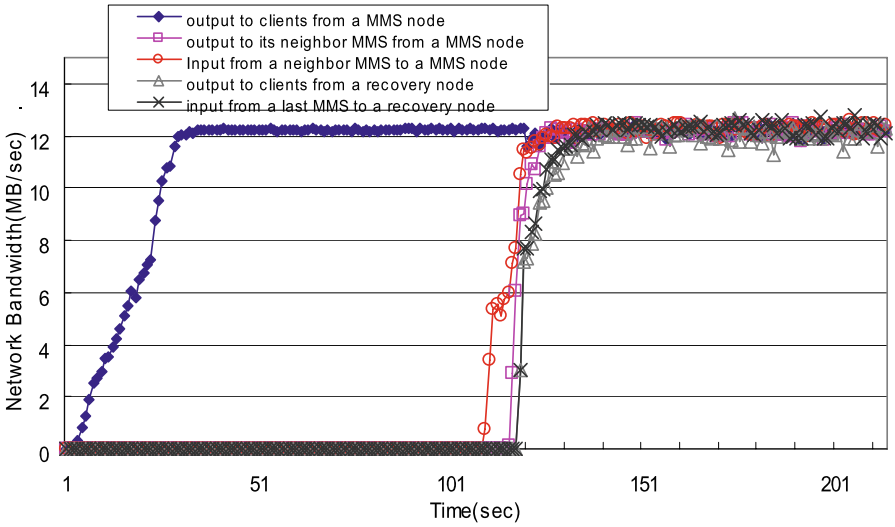

Fig. 10 Network traffic in the RS-PCM

to the recovery node drops to 4-MB/sec rates. In Fig. 9, 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/sec rates. The input traffic is marked as the $\times$ shapes in Fig. 9. Therefore, the recovery node receives the video data of 12 MB/sec 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/sec rates. Depending on the amount of 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/sec rates. The 4-MB/sec traffic means that only 85 clients can be supported ($4\ \mathrm{M} * 8\ \mathrm{bit} * 4\ \mathrm{MMS\ nodes}/1.5\ \mathrm{Mbps} = 85$). As a result, after an MMS node fails, the RS-BRM method can support only 33.2% of the original 256 streams.

Figure 10 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 seconds 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 Fig. 10, we 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 Fig. 10, the input traffic from the last MMS node to the recovery node reaches 12 MB/sec 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

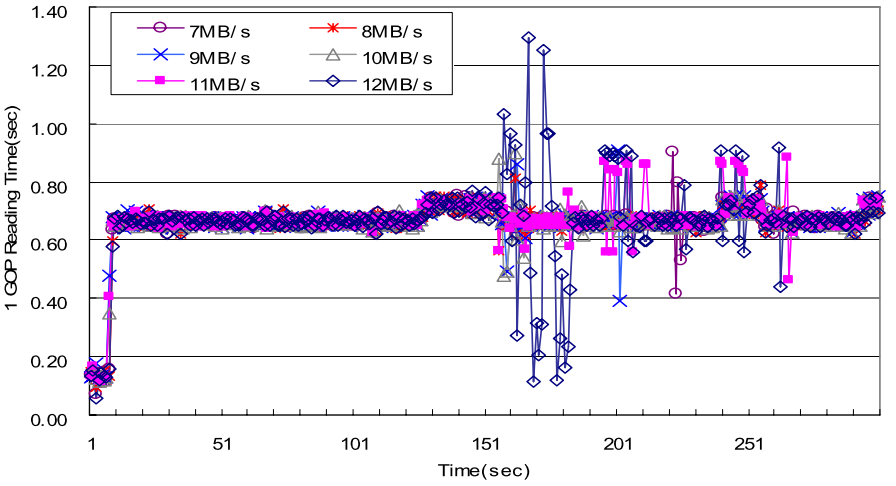

Fig. 11 GOP reading time in the RS-BRM

to the clients uses 12-MB/sec 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/sec, streaming media service for 256 clients can be continued ($12\ \mathrm{M} * 8\ \mathrm{bit} * 4\ \mathrm{MMS\ nodes}/1.5\ \mathrm{Mbps} = 256$).

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 MTTR (mean time to recovery)

Figure 11 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/sec and the 12-MB/sec loads. Although an MMS node fails at 120 seconds on the time line, the fluctuation of the GOP reading times starts at the 156-second position. The time delay is due to the network delay and the buffering time in the client side. As shown in Fig. 11, when all MMS nodes work normally, the average reading time is about 0.65 second, 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/sec and the 12-MB/sec load. Under these workloads, unstable oscillations of the GOP reading times emerged between 156 seconds and 266 seconds. The difference between the

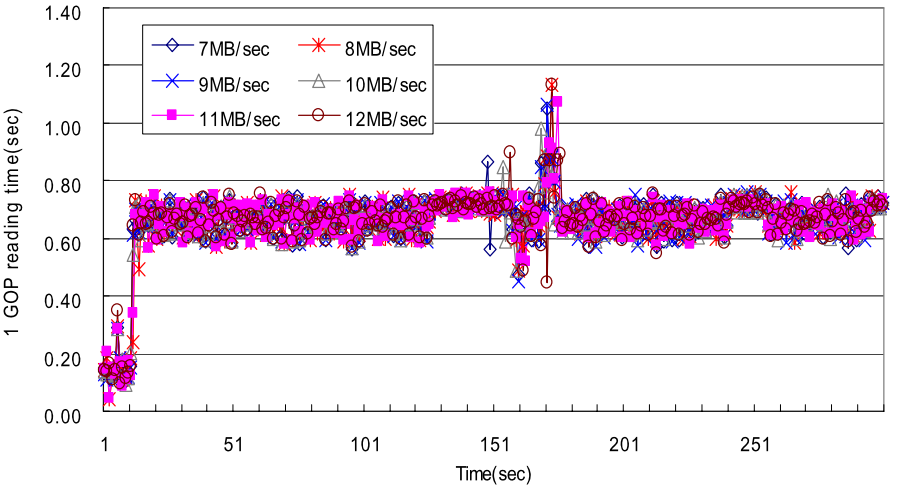

Fig. 12 GOP reading time in the RS-PCM

maximum GOP reading time and the minimum GOP reading time is about 1.18 seconds. After the fluctuation period passes through, the recovery node works normally, and the GOP reading times converge to the 0.65-second level again. The MTTR value is 110 seconds. It can be regarded as an impatient period to VOD clients [4, 18].

Figure 12 represents the GOP reading times in the RS-PCM. The failure of an MMS node takes place at 120 seconds 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 side begins at 148 seconds and ends at 176 seconds. After the agitation state, the reading times promptly converge to the steady state of 0.65 second. The fluctuation period is just 28 seconds. When compared to the RS-BRM, the fluctuation period is very short. Since the recovery operations are distributed to 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 that of the RS-BRM. The difference between the maximum GOP reading time and the minimum GOP reading time is 0.68 second. This result is half of the difference issued by the RS-BRM. In the RS-PCM, since the fluctuation period and the vibration amplitudes are very short, we confirm that the RS-PCM results in a much better MTTR value than the RS-BRM.

6.3 Performance scalability

In the VOD service, admission control is applied to support the quality of the streaming services in progress. If an additional client gives rise to the negative impact on the quality of streaming services, the admitting request should be rejected by the admission control. In our VOD system, the HS node performs admission control for new client requests. For admission control, the HS node has the maximum number of QoS streams supported by the MMS nodes. Even if an MMS node fails, the HS node performs admission control to support the streaming services in progress. To evaluate the performance scalability in two recovery methods, when an MMS node fails, we

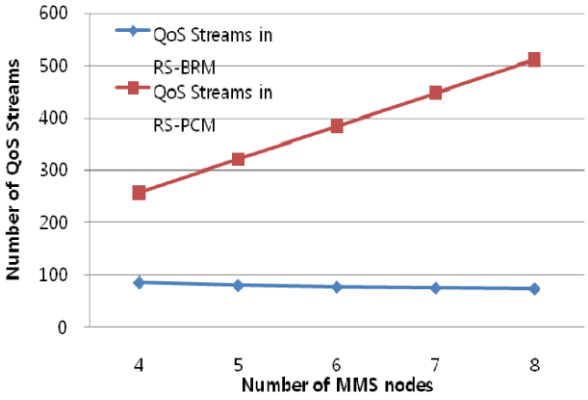

Fig. 13 Total QoS streams

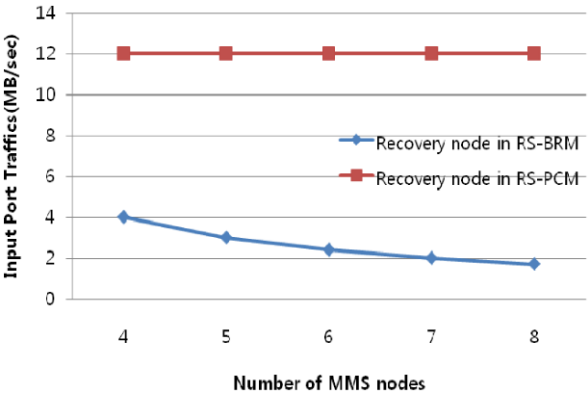

Fig. 14 Input network traffic in the recovery node

measure the total number of QoS streams supported by the remaining MMS nodes and a recovery node. These numbers mean the maximum client requests decided by the admission control of the HS node.

After an MMS node fails, Fig. 13 shows the maximum number of QoS streams provided by the remaining MMS nodes and a recovery node. These results are measured on the yardstick program referred to in Sect. 5.2. This program applies the admission control to the given MMS nodes, and it can measure the maximum number of QoS streams in the given environment.

As shown in Fig. 13, as the number of MMS nodes is increased, the RS-PCM represents the linearly scalable performance. However, in the RS-BRM, the number of QoS streams does not increase even if the number of MMS nodes increases. These results are explained by the data traffic of the input port of the recovery node. From the RS-BRM results shown in Fig. 14, the input port traffic of the recovery node decreases as the MMS nodes are added. The reason is that the simultaneous transmission of video blocks from the remaining MMS nodes causes the bottleneck phenomenon in the input port of the recovery node. As the input port traffic of the recovery node decreases, the output traffic from MMS nodes to the clients also decreases. As a result, the total number of QoS streams supported by the RS-BRM decreases, as shown in Fig. 13. However, in the RS-PCM, the recovery node receives

only the results of exclusive-OR operations transmitted from an MMS node, not all remaining MMS nodes. Therefore, even if MMS nodes are added, since the bottleneck phenomenon does not happen, the input port traffic of the recovery node can be sustained at 12 MB/sec. The recovery node regenerates the video blocks and transmits to the clients at the 12-MB/sec rate. The remaining MMS nodes also transmit their video blocks to the clients at the same rate. As a result, as shown in Fig. 13, the total QoS streams supported by the RS-PCM increase according to the number of MMS nodes.

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 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 not only distributed 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. Bolosky WJ, Pitzgerald RP, Draves JH (1997) Distributed schedule management in the tiger video fileserver. In: Proceedings of the sixteenth ACM symposium on operating systems principles, Saint Malo, France, October 5–8, 1997, pp 212–223
2. Chang T, Shim S, Du D (1998) The designs of RAID with XOR engines on disks for mass storage systems. In: IEEE mass storage conference, March 23–26, 1998, pp 181–186
3. Choi J-M, Lee S-W, Chung K-D (2001) A multicast delivery scheme for VCR operations in a large VOD system. In: IEEE international conference on parallel and distributed systems, June 26–29, 2001, pp 555–561
4. Fox A, Patterson D (2005) Approaches to recovery oriented computing. IEEE Internet Comput 9(2):14–16. doi:10.1109/MIC.2005.39
5. Gafsi J, Biersack EW (1999) Data striping and reliability aspects in distributed video servers. Cluster Comput Netw Softw Tools Appl 2(1):75–91
6. Gafsi J, Biersack EW (2000) Modeling and performance comparison of reliability strategies for distributed video servers. IEEE Trans Parallel Distrib Syst 11(4):412–430. doi:10.1109/71.850836
7. Holland M, Gibson G, Siewiorek D (1994) Architectures and algorithms for on-line failure recovery in redundant disk arrays. J Distrib Parallel Databases 2:295–335. doi:10.1007/BF01266332
8. http://www.ieeetfcc.org (2009)
9. http://www.mpeg.org (2009)
10. Kang S, Yeom HY (2003) Modeling the caching effect in continuous media servers. Multimedia Tools Appl 23(3):203–224. doi:10.1023/A:1025702332314
11. Merchant A, Yu PS (1995) Analytic modeling and comparisons of striping strategies for replicated disk arrays. IEEE Trans Comput 44:419–433. doi:10.1109/12.372034
12. Patterson DA, Hennessy JL (1998) Computer organization & design. Morgan Kaufmann, San Mateo, pp 392–490
13. Sarhan NJ, Das CR (2004) Caching and scheduling in NAD-based multimedia servers. IEEE Trans Parallel Distrib Syst 15(10):921–933. doi:10.1109/TPDS.2004.49
14. Schmidt BK, Lam MS, Northcutt JD (1999) The interactive performance of SLIM: a stateless, thin-client architecture. In: ACM symposium on operating systems principles, 1999, pp 31–47
15. Seo D, Lee J, Jung I (2007) Resource consumption-aware QoS in cluster-based VOD servers. J Syst Archit 53(1):39–52
16. Shenoy PJ, Goyal P, Vin HM (2002) Data storage and retrieval for video-on-demand servers. In: IEEE fourth international symposium on multimedia software engineering, December 2002, pp 240–245
17. Sitaram D, Dan A (2000) Multimedia servers: applications, environments, and design. Morgan Kaufmann, San Mateo
18. Tang D, Zhu J, Andrada R (2002) Automatic generation of availability models in RAScard. In: IEEE international conference of dependable systems and networks, June 23–26, 2002, pp 488–494

Joahyung Lee received his B.E. and M.Sc. degrees in Information and Telecommunication Engineering from Kangwon National University in 2003 and 2005, respectively. He is currently a Ph.D. candidate in Computer Engineering at Kangwon National University. His research interests include multimedia system, parallel processing, embedded system and wireless sensor network.

Inbum Jung received his B.Sc. degree from Korea University, in 1985 and the M.Sc. and Ph.D. degrees in Computer Science from KAIST, in 1994 and 2000, respectively. From 1984 to 1995, he was with Samsung Electronics Co. Ltd., Korea. He is currently a faculty member at Kangwon National University. His research interests include operating system, parallel processing, streaming media and wireless sensor network.