Available online at www.sciencedirect.com
ScienceDirect
Journal of Systems Architecture 53 (2007) 39–52
JOURNAL OF SYSTEMS ARCHITECTURE
www.elsevier.com/locate/sysarc

Resource consumption-aware QoS in cluster-based VOD servers ☆

Dongmahn Seo, Joahyoung Lee, Yoon Kim, Chang Yeol Choi, Manbae Kim, Inbum Jung *

Department of Computer Science and Engineering, Kangwon National University, 192-1 Hyoja 2-Dong, Chuncheon, Kangwon Do 200-701, Republic of Korea

Received 13 August 2005; received in revised form 21 April 2006; accepted 3 July 2006
Available online 13 October 2006

Abstract

For Video-On-Demand (VOD) systems, it is important to provide Quality of Service (QoS) to more clients under limited resources. In this paper, the performance scalability in cluster-based VOD servers is studied with several grouping configurations of cluster nodes. To find performance bottlenecks, the monitoring functions are employed and the maximum QoS streams are measured under the various requests including VCR functions. To support more user friendly interface, an embedded set-top model is suggested for the QoS of TV clients. From our detailed experiment results, a new admission control method is proposed that is based on available system resources and the actual amount of resource consumed for QoS streams. The proposed method provides not only more scalable QoS in cluster-based VOD servers but also the enhancement of resource utilization by guaranteeing the maximum number of QoS streams.

Keywords: VOD; Cluster system; Streaming media; QoS; Resource aware; Parallel processing

1. Introduction

VOD is a representative streaming media service technology and has been researched in various areas. However, it is hard to expect good performance if the VOD system is implemented without considering the interrelation between the server and client [1]. The performance of the VOD system is represented with the number of concurrent clients giving a guarantee against a stable QoS. The QoS is closely related to the server architecture model, storage systems, network capability and client terminal types.

☆ This research was supported by the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Assessment) (IITA-2005-(C1090-0502-0022)). This work was partially supported by the Kangwon Institute of Telecommunications and Basic Research Program of the Korea Science Engineering Foundation (R05-2003-000-12146-0).
* Corresponding author. Tel.: +82 33 250 6396; fax: +82 33 252 6390.
E-mail address: ibjung@snslab.kangwon.ac.kr (I. Jung).

1383-7621/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.sysarc.2006.07.003

In this paper, the cluster-based VOD server denoted as the VODCA (Video-On-Demand Cluster Architecture) is presented. Our devised system is composed of a general PC cluster platform and performs parallel processing for MPEG 1, 2 media. Not only does the VODCA provide the scalable performance within the pre-defined QoS ranges, but also it supports a various requests from clients including VCR functions.

For the parallel processing of MPEG 1, 2 media, the granularity for parallelized materials should be determined. In our research, the Group of Pictures (GOP) layer in MPEG is regarded as a granularity unit. The MPEG movie data is striped as GOP units and evenly distributed to backend nodes. The GOP units extracted from MPEG media are independently processed in backend nodes. Based on parallel storage, parallel retrieval and parallel transmission into a network, the load balancing is maintained among backend nodes. From our approaches, the VODCA can supply the scalable streams to clients within the pre-defined QoS metrics.

Due to intrinsic MPEG characteristics, it is hard to implement VCR functions such as fast rewind, fast forward and resume commands in VOD servers [2,3]. If these functions are implemented by means of sending and playing the movie data at a faster speed, the network is easily saturated. In our study, we propose VCR functions by extracting I frames from MPEG data and by managing these frames independently. Our experiment shows that the proposed method for VCR functions reduces the network traffic by sending only I frames.

The PC is usually considered as the standard equipment of VOD clients, but we employ TV to provide non-specialists with a friendlier interface. To satisfy the interactive requirements of clients on the TV side, we implement a set-top box that is composed of an embedded board and an infrared sensor device controlled by a remote controller. This TV client provides a more comfortable human interface by selecting desired movies within the range of the remote controller.

In the VOD system, admission control is required to guarantee the quality of streaming media while all clients are being serviced. In our research, we devise system monitoring functions to analyze resource consumption for serviced streaming media. Based on these functions, we measure the quantity of memory, disk bandwidth and network bandwidth consumed for satisfying various client requests with QoS streaming movies. From these detailed measurements, the bottlenecks of scalable performance in cluster-based VOD servers are investigated. Based on these analyses of resource consumption, a RCAAC (Resource Consumption Aware Admission Control) method is proposed. It is based on available system resources and the amount of resources consumed for QoS streams. In our experiments, we confirm that the proposed method increases the utilization of the system resource by guaranteeing the maximum number of QoS streams.

The rest of this paper is organized as follows. Section 2 discusses the parallel processing of MPEG data and the implementation of VCR functions. Section 3 provides the details for our VODCA system. In Section 4, the performance of our VOD system is measured and the results are analyzed. Section 5 makes suggestions for the resource consumption-aware admission control. Section 6 describes related work and Section 7 concludes the paper.

2. Parallel processing of MPEG media

2.1. Characteristics of MPEG media

MPEG-1, 2 media consists of a video sequence layer, a Group of Pictures (GOP) layer and a picture layer. The video sequence layer is a group of sequential pictures that have the same frame size and frame rate. The GOP layer is a minimal unit for playing movies and is usually exploited at the random access unit. The picture layer is the single image displayed on the screen. This picture layer is composed of 4 kinds of frames known as I, B, P, D frames. Each frame has its own different function for its decoding procedure. Each GOP is a random access unit and includes at least one or more I frames. Fig. 1 shows the architecture of GOP. An I frame uses only transform coding and provides a random access point into the compressed video data. I frames can be used for predicting P and B pictures. A P frame is coded using motion compensated prediction from a previous I or P frame. This technique is called forward prediction from I/P to P as shown in Fig. 1. P frames can accumulate coding errors. P frames go through the feedback loop and can be used for predicting P and B pictures. A B frame is coded using both a past and/or future picture as a reference. Thus it is called bidirectional prediction, as shown in Fig. 1. The errors are not accumulated, since the B picture is never used as a reference. B frames can be coded using forward or backward (or both) motion compensation. A D frame is a special case of intra in

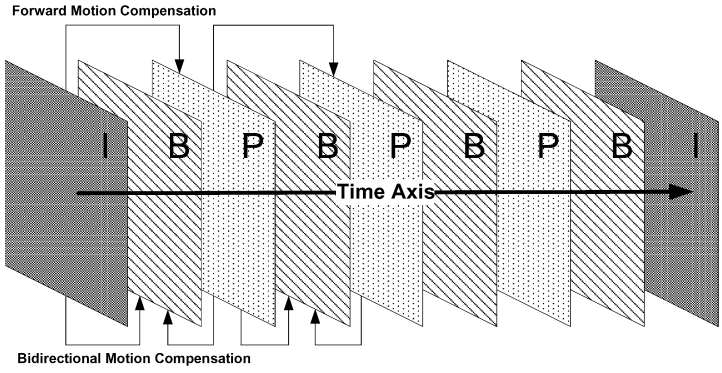

Fig. 1. Architecture of GOP.

which only the DC coefficient of each (8×8) block is coded. D frames provide simple and fast forward mode but yield limited image quality. Currently D frames are not used [4].

Each layer in MPEG-1, 2 has its own header and is distinguished from specific byte streams. The byte stream of headers begins with $[0 \times 000001 +$ start bits for each layer]. The start bits of the video sequence layer are $0 \times B3$. The GOP layers have $0 \times B5$ and the start bits of picture layers are 0×00. From these headers, it is possible to extract GOPs and specific frames from MPEG media.

2.2. Striping and distribution for MPEG media

To apply parallel processing for MPEG media, the movie files are striped based on the defined granularity policy. After that, the movie file is partitioned into many fragments. These fragments are distributed into backend nodes with their header information.

The choice of striping methods is related to the load balancing, clients' preference and VCR functions. If each backend node has different sizes of movie fragments, it is hard to keep the load balance among backend nodes. Two approaches are usually suggested in striping policies. One method is to partition the MPEG movie into fragments with same size. This way is easy to implement in the striping step and evenly uses the disk space of each node. However, it is difficult to implement the VCR functions such as fast forward, rewind and resume requests. Another approach is that a MPEG movie is split into fragments with equal running time. This approach provides relatively good functionality for VOD clients but it is not easy to create equal amounts of playing time for all fragments. In our research, the second method is exploited for supporting various functions in our VOD system. To exploit MPEG media characteristics, we use one GOP size as a striping unit. Since each GOP has approximately equal running time in MPEG streams, the MPEG movies are split into GOPs and distributed into each node with their sequence number and size. For example, if a movie decodes 30 frames per second and one GOP consists of 15 frames, each backend node takes charge of the streaming service for 0.5 s.

There are several methods for striping among backend nodes [5,6]. In round-robin method, GOPs are distributed into each node based on the sequence of nodes such as $1 \rightarrow 2 \rightarrow \cdots \rightarrow N \rightarrow 1 \cdots$, whereas the SCAN method distributes them as $1 \rightarrow 2 \rightarrow \cdots \rightarrow N \rightarrow N-1 \cdots \rightarrow 1$. The selection of striping method depends on the construction of the disk storage, the effects of data prefetching and the weight of clients' preference movies. In our research, when the movies are deployed in our system, the system administrator can choose the striping method by considering the organized cluster's properties.

2.3. VCR functions

Several approaches are proposed to supply the VCR functions. For example, there are the employment of separated movie files, transmitting movie data more quickly and SCAN method [1]. While first method has overhead for making additional data files and requiring extra disk storage for VCR functions, there is no runtime overhead when the VCR functions are being serviced. The second method has no overhead in the initial stages. However, this method severely exhausts network bandwidth, because it requires more network bandwidth to transmit the amount of MPEG media in as much quantity at fast rates. The last SCAN method is that

specific frames are extracted from MPEG movies for running time and they are transmitted to the clients requesting VCR functions. In this method, servers are easily overloaded due to the gathering of the specific frames. In addition to the overhead, it is difficult for other clients to reuse the frames extracted from the same movie.

In our research, by using the separated movie files together with the SCAN method, the advantages of the two methods are exploited. When a new MPEG movie is enrolled into our VOD system, I frames are extracted from the GOPs of the new movie file and I frame files are created for VCR functions. Each I frame file is composed of all of I frames and header information. Since I frames are decoded and played independently without referencing other frames, each I frame file is dispatched onto the node that the corresponding GOP is located on. By exploiting the sequence number in the header, there is no synchronization overhead between backend nodes while the streaming service is proceeding. Since the size of I frames is smaller than that of GOPs, the transmitting of I frames instead of GOPs reduces the amount of network bandwidth consumed.

In our approach, when new movies are enrolled, the overhead to create I frame files exists. However, there are no additional costs for supporting VCR functions in running time. Since no runtime overheads enhance the utilization of system resources, a greater number of QoS streams are available.

3. Architecture of the VODCA system

To examine out performance limitations in large scale VOD services, we implemented the VODCA (Video-On-Demand on Clustering Architecture) system. The VODCA system includes not only server sides but also client sides. Servers in the VODCA consist of a HS (Head-end Server) node and several MMS (Media Management Server) nodes known as backend nodes. The client system in the VODCA is working together with HS and MMS nodes. Fig. 2 shows the architecture of our VODCA system.

3.1. Servers in the VODCA system

3.1.1. Head-end Server (HS) node

The HS node not only receives clients' requests but also manages MMS nodes to support QoS. When new MPEG movies are enrolled, the HS splits them and distributes them into each MMS node. To perform these administrative functions, the HS

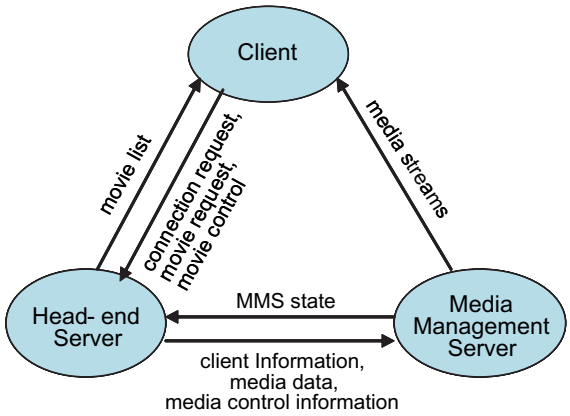

Fig. 2. Architecture of the VODCA system.

consists of a striping module, monitoring module, service_control module and a main_daemon module as shown in Fig. 3.

The striping_module reads the header from MPEG movie files and splits the movie data into GOP units and also extracts I frames to support VCR functions. To keep the order of split movie data during the movie runtime, the sequence number and header data are attached in front of both the GOPs and I frames. With the identities for GOP and I frame files like this, these files are delivered to each MMS node according to the striping policy. In the VODCA system, both round-robin policy and SCAN policy are provided for a system administrator.

The monitoring module provides functions to manage the VODCA system. It displays usages of CPU, memory and network of the HS node and MMS nodes. A system administrator can insert, delete, and modify MMS nodes and s/he can measure the quantity of memory, disk and network bandwidth used for each stream.

To monitor the working state of MMS nodes, a heartbeat protocol has been devised between the HS node and the MMS nodes. Since the monitoring period of the heartbeat protocol is 2 s, it does not cause the performance degradation of MMS nodes.

The connection request from a VOD client is managed by a thread of service_control module. This module provides the information of enrolled movies to clients. When clients demand movie, this module establishes the connection between MMS nodes and clients. After connecting, the demanded movie begins to transmit from MMS nodes. During the movies, all control requests including VCR functions are sent to MMS nodes via this module.

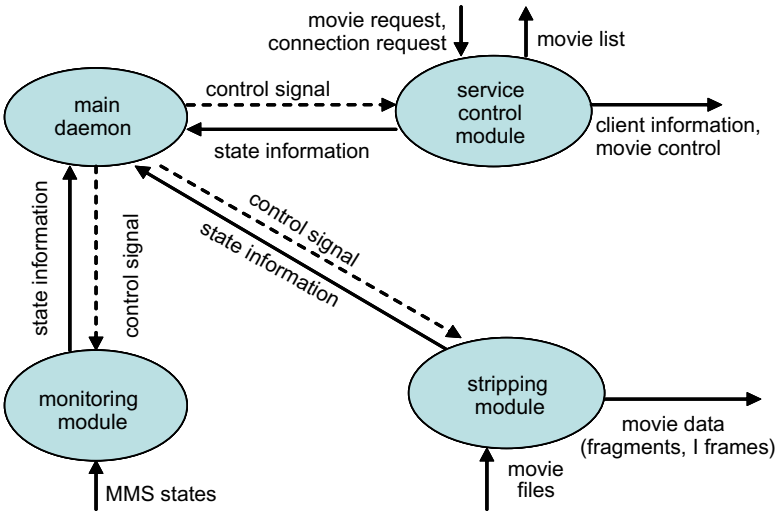

Fig. 3. Architecture of HS node.

The main_daemon module supervises all modules within the HS node and also provides the administrator with general interfaces to manage nodes.

3.1.2. Media management server (MMS) node

The MMS nodes transmit their stored movie fragments to clients under the supervision of the HS node. Each MMS node sends the present working status to the HS node periodically. This message operates as a heartbeat protocol between MMS nodes and the HS node. Each MMS node consists of a media_management module, media_service module, resource_management module and a main_daemon module. Fig. 4 shows the architecture of MMS nodes.

The media_management module handles the various requests from both the striping module and service module working in the HS node. In addition to storing the movie fragments into the disk storage of each MMS node, it also provides removal and modification functions.

The resource_management module collects the information about the MMS node's internal state

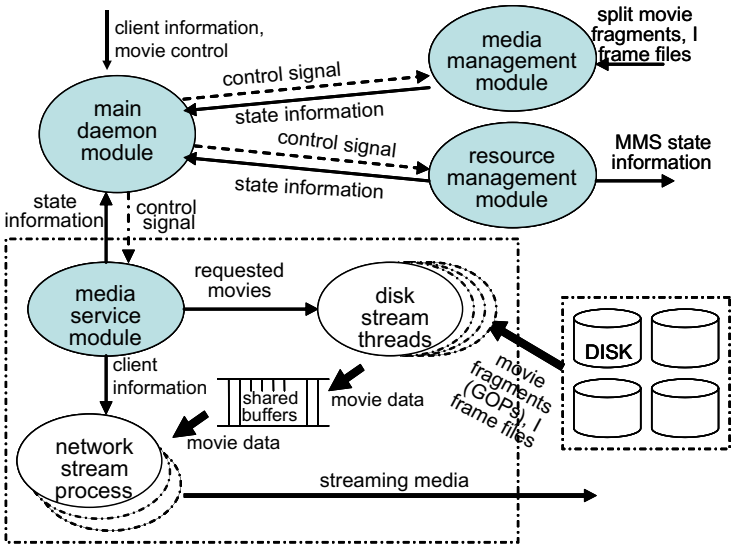

Fig. 4. Architecture of MMS node.

such as the usage of CPU, memory, network and disks. The */proc* file system in Linux is exploited for this purpose. This information is plugged into message packets and sent to the HS node every 2 s as a heartbeat message. Since the size of this heartbeat message is 32 bytes, it dose 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 new client's requests.

As shown in Fig. 4, the media_service module consists of disk stream threads and network stream processes. In this module, the stored movie fragments are retrieved from disks and transmitted into the network. For one client, one disk stream thread and one network stream process is allocated in all MMS nodes. When a new client arrival is registered from the HS node, a network process is created and the connection to client is established. After that, a disk stream thread is created and begins to retrieve the requested movie fragments. These fragments are loaded on the shared buffers as shown in Fig. 4. The corresponding network stream process reads movie fragments from the shared buffers and sends them via the network path connected to the client. To guarantee the integration of data on shared buffers, synchronization primitive such as mutex is applied.

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

3.2. Clients in the VODCA system

3.2.1. Software architecture for VOD clients

The client side in VOD system should provide for the various client interfaces. It sends client requests to the VOD server and decodes the MPEG movies and plays them on the screen. As shown in Fig. 5, the client architecture in the VODCA system consists of a client_interface module, network_receive module, data_reordering module and a media_playback module.

The client_interface module delivers 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 users.

The network_receive module receives movie data packets from MMS nodes. This module communicates with all MMS nodes and receives movie data. The received packets are merged into individual GOP fragments and stored on the shared memory areas in the client side. Each fragment involves a GOP data and its 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 sequence order of these fragments based on their sequence number within header information. According to the sequential ordering of fragments, they are passed to the media_playback module via the pipe mechanism of Linux. This module decodes the movie

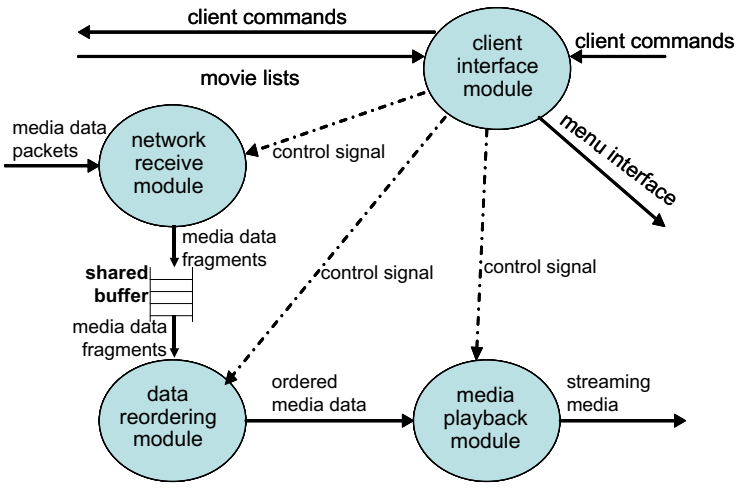

Fig. 5. Software architecture of VOD client.

data passed from the pipe and displays them on the screen. In particular, while the VCR functions are serviced, only video data without the audio data are decoded and displayed.

3.2.2. VOD client with TV equipment

To provide non-specialists with a friendlier interface, we employ TV as the equipment of VOD clients. To provide interactive functions for TV clients, a set-top box composed of an embedded board and an infrared sensor device controlled by the remote controller is deployed. TV clients provide a comfortable human interface by selecting desired movies within the range of the remote controller.

The basic structure of our set-top device includes 4 internal software modules described in Fig. 5. In addition to these modules, the set-top equipment consists of the re-compiled embedded Linux kernel, the design of client GUI to exploit frame buffers and the movie player working in limited resources. Our set-top operates based on Linux operating system and embedded Qt libraries [7–9].

To install the Linux on our embedded boards, the general PC is exploited as a test-bed. For a target disk, a specific directory is created and is formatted as the ext3 type [7]. This disk is organized as the independent booting environment with minimum files and libraries. When the Linux kernel compiles, we have the option to activate the frame buffer mechanism. This re-compiled kernel is also stored in the target disk area.

The embedded Qt 3.0 is used to make the client GUI environment. It supports window development tools based on Linux frame buffers. While the VOD servers are based on C libraries, our embedded client is implemented on Qt libraries. Due to the different working environment, there are some interoperability problems between servers and clients. To solve these problems, the client's network module is designed with C libraries.

The MPlayer 0.18 version is utilized for our movie player. It is supported on open source project committee [10]. Since the embedded environment has limited resources, the MPlayer 0.18 was compiled to fit into our environment. Furthermore, since the MPlayer does not support the VCR functions such as fast rewind and fast forward, we modified the signal handler parts of MPlayer to handle the external signals from the client remote controller. These additional handlers operate their independent roles according to each VCR function demand.

We also modify MPlayer to cooperate with the data_reordering module taking charge of the reordering mechanism of movie fragments. As a result, all preserved materials stored in the target disk are downloaded to the embedded board being used as a set-top and it is working for a TV client.

4. Performance evaluation

4.1. Experiment environment

The VODCA server for our experiments consists of a HS node and 6 MMS nodes. Each node operates on the Linux operating system. The MMS nodes, HS node and clients are connected via a 100 Mbps Ethernet switch. All applications included the system administrative tools of the HS node are developed on Qt, C and C++ libraries. Table 1 shows the hardware components for each MMS node in the VODCA system.

We use the yardstick program to measure the performance of our cluster-based VOD servers [11]. The yardstick program consists of the virtual load generator and the virtual client daemon. The virtual load generator is located in the HS node and generates client requests based on the Poisson distribution with $lambda = 0.25$ [2,6]. These requests are sent to each MMS nodes. After that, all MMS nodes concurrently begin streaming media services to satisfy the clients' demand.

The virtual client daemon locates in test-bed PCs for clients. It plays the role of receiving movie data from MMS nodes. Based on MPEG-1, 2 specifications, we assume that a QoS stream requires 1.5 Mbps of network bandwidth. To support this QoS metric, the virtual client daemon measures the time elapsed for receiving 1.5 Mbits of data. If the elapsed time is below 1 s, the virtual client daemon remains in an idle state until a 1 s period passed. After exhausting this remaining time, the daemon wakes up again and begins to receive the

Table 1
Specification of MMS nodes

CPU	Intel Pentium 4, 1.6 GHz
Memory	256 MB DDR
Disk	Segate Baracuda ATA IV 40 GB 7200 RPM × 2
Operating system	RedHat 7.3 (Kernel 2.4.18)
Network	100 Mbps Fast Ethernet, 100 Mbps Ethernet Switch with 24 ports

next media data. This waiting process makes the virtual client daemon act as a real client. However, if our virtual client daemon receives movie data below the 1.5 Mbps rate, the MMS nodes are regarded as they are in an overloaded working state. This situation means that the QoS for streaming media is not supported. The virtual client is implemented based on our test-bed PCs. In our experiments, it can be found that a PC plays enough roles for 30 virtual clients.

VCR functions are a kind of image search request. Since clients are satisfied with the transmission of I frame, the QoS metrics for VCR functions are not considered in our experiments.

Table 2 shows the detail specification of movies used in our experiments. They are MPEG-2 movies and have enough running time to evaluate their performance in our VODCA system. As shown in Table 2, we also measured the sizes of GOPs and I frame in these movies. This information is used practically to find out the sources of performance bottlenecks described in the next Sections.

Table 3 shows the internal components of the embedded kit for developing set-top equipment. This kit is used to implement a set-top device for a TV client. However, while the total number of clients is measured in our VODCA system, general PCs loading the yardstick program are used.

Table 2
Specification for experimental movies

Movie name	John Q	Ice Age
Frame size (width × height)	352 × 288	352 × 288
Frame rates (frames/s)	25	25
Running time (min)	110	85
GOP size (Max-Avg-Min, KB)	252.9–124.1–7.4	281.7–120.8–20.8
I frame size (Max-Avg-Min, KB)	43.6–25.8–6.9	50.2–22.8–6.6
Header size (Bytes)	4910	2558

Table 3
Set-top developing environment

Board	IB790
CPU	Pentium III 600 MHz
Memory	64 MB SDRAM
Flash memory	128 MB Flash Disk
Input device	RS-232 C serial infrared sensor
Operating system	Linux Kernel 2.4.18
GUI	Embedded Qt 3.0.6
Movie player	MPlayer 0.18
Test bed	Redhat 7.3, P4 1.6 GHz, 256 MB

4.2. Server performance

4.2.1. Performance under normal play requests

Fig. 6 shows the performance scalability of the VODCA according to the number of MMS nodes. The *Y*-axis represents the maximum number of clients giving a guarantee against the QoS metric. As assumed in the above Section, the QoS metric in our research is that the transmission rate of 1.5 Mbps is guaranteed. The results in this figure are the average of 5 times measurements under the same experimental environment.

As illustrated in Fig. 6, the maximum number of clients linearly increases until the number of MMS reaches 4. However, when over 4 MMS nodes are participated, the linear scalability suffers from the limitation of MMS nodes' internal resources. To find the probable causes of performance bottleneck, the amount of CPU, network, disk, and memory used are measured when the scalability curve reaches the saturation point.

From our investigation, the usage of CPU in each MMS node did not exceed 10% at maximum. Thus, the CPU performance did not relate to the nonlinear scalability over 4 MMS nodes.

In the aspect of network bandwidth, since all MMS nodes and clients are connected to 100 Mbps Ethernet switch, one MMS node could support about 66 clients with 1.5 Mbps network bandwidth. When 6 MMS nodes are employed, the total number of clients is theoretically 396 streams. However, using 6 MMS nodes in Fig. 6, only 230 clients were measured with the Ice Age movie. To study the limitation of network bandwidth, we divide MMS

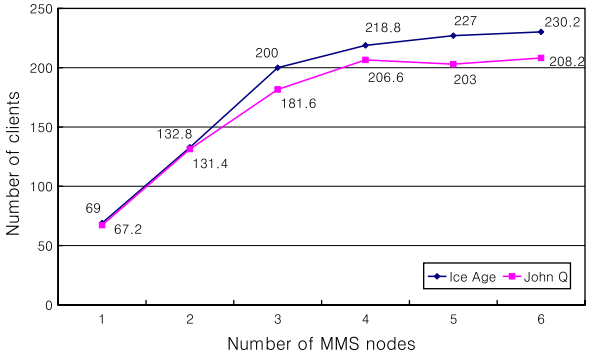

Fig. 6. Performance scalability with increment of MMS nodes.

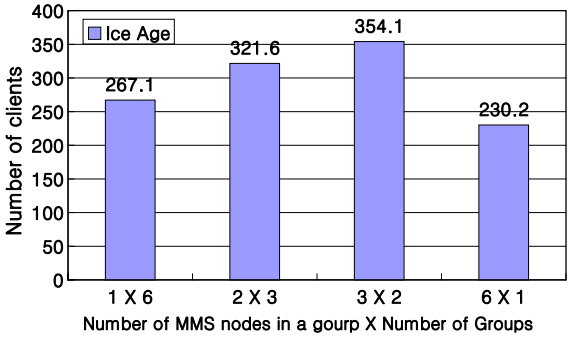

Fig. 7. Performance under the sub-grouping configurations of MMS node.

nodes into sub-groups loaded with the same movies. Based on these sub-grouping configurations, we measure the maximum performance of each group. Fig. 7 shows the maximum performance under the several sub-grouped configurations of MMS nodes. The best performance is acquired on the 3×2 model. The former is the number of MMS nodes in one group and the latter is total number of group. As shown in Fig. 7, the 3×2 configuration model supports about 354 clients. The value is approximately close to the maximum number of clients in our network capacity. From these experiments, the network capacity in our implemented VODCA system does not work as a bottleneck to the linear performance scalability.

Each hard disk in the MMS node provides data in 100 MB/s at maximum and could support 22–41 MB/s continuously [12]. Since the hard disk speed is superior to the network bandwidth, the disk bandwidth is also not also the bottleneck in performance scalability.

Our final consideration to find performance bottlenecks is the memory capacity in our system. The media_service module in MMS nodes uses a buffer for each client stream. The buffer size is varied according to size of two GOPs. The GOP size for our movies is represented in Table 2. The buffer in each process takes the major portion of memory usage. From the experiment, we found that one streaming service consumed about 1300–1500 KB of memory in MMS nodes. Based on these observations, only 187–200 streams are supported in each MMS node with 256 MB memory. From additional experiments, we also found that the number of memory swapping highly increased after the number of streams is over 200. Therefore, we confirm that the memory usage of media_service module could work as a reason for the nonlinear scalability over 4 MMS nodes.

Table 4
Performance of the VODCA in the mixture requests

VCR function rate	6 nodes × 1 group (number of streams)	3 nodes × 2 groups (number of streams)
0%	230.2	354.1
10%	239.0	378.2
20%	275.4	389.0
30%	279.2	319.0

4.2.2. Performance with VCR functions

Table 4 shows the maximum number of QoS streams in the VODCA when the normal play and VCR functions are mixed. The % item shows the mixed rate of VCR function requests among total streams. From these experiments, the higher mixed rates, the better performances occur.

When handling the VCR functions, MMS nodes transmit I frame of each GOP. The size of I frame is smaller than the size of GOP. From Table 2, while the average GOP size of the 2nd movie Ice Age is 120 Kbytes, the average I frame size is just 22.8 Kbytes. The size of I frame is smaller than GOP as much as 6 times. Since the media_service module uses two buffers for supporting one stream and the buffer size is equal to the size of GOP, a normal play stream requires 240 Kbytes buffer size but one VCR function stream needs 45 Kbytes buffer size. In conclusion, the processes served for VCR functions exhaust less buffer memory than other processes supporting normal plays. Due to the reduced memory usage, it may be possible to increase the number of total streams according as the mixed rates are increased.

4.3. Client performance

In the client side, the memory usage is small but the most CPU resources are exhausted due to decoding the MPEG media in MPlayer. There is about 1 s of delay when the player state is changed from the normal play to VCR functions. The extreme delay time comes from fast forward state to fast rewind state, it takes about 3–4 s maximum. The reason is due to both the delay of client request transfers and the buffer refilling mechanism in MMS nodes. The former is influenced by the polling period to find client's request in the interface module and the network delay for bypassing client requests from HS node to the MMS nodes.

The buffer refilling problem is the time for flushing the buffers and refilling into new movie data in

MMS nodes. In addition to these reasons, the last movie data remained in the client side aggravates the delay time. To enhance the responsibility for client requests, it is useful for avoiding the buffer mechanism when treating the VCR requests as well as the client's network_receive module should perform immediate buffer flushing as soon as the VCR function requests are issued.

5. Resource consumption-aware admission control

From the last section, the performance of the VODCA system is studied on the limited internal resources. To analyze the causes of performance degradation, we measured the amount of resources consumed for each QoS stream by exploiting the monitoring functions. Within the implemented VODCA system, it was clear that the main memory caused the major bottleneck and the network bandwidth did not affect the scalable performance greatly. However, to get the scalable performance, there were no burdens in the aspects of disk retrieving and CPU computation ability. The result is from the parallel disk retrieving and very low CPU consumption in MMS nodes.

In VOD system, admission control is required to guarantee the quality of media streaming while clients are being serviced. Based on the information about the amount of resource consumptions, we propose a new admission control called as RCAAC (Resource Consumption-Aware Admission Control) in the cluster-based VOD servers. The RCAAC is driven from the amount of available resources in MMS nodes in addition to the actual amount of resource consumed for guaranteeing one QoS stream. Since the proposed method drags up the utilization of system resources, the more QoS streams can be provided. Table 5 shows the notations for our resource consumption-aware admission control.

Table 5
Notations for resource consumption-aware admission control

M_i	Total memory of ith MMS node
m_{ij}	Memory usage of j movie in ith MMS node
N_i	Total network bandwidth of ith MMS
n_{ij}	Network bandwidth usage of j movie in ith MMS node
S_i	Total system memory usage in ith MMS node
AM_i	Available memory of ith MMS node
AN_i	Available network bandwidth of ith MMS node
CM_i	Consumed memory for supporting a new movie stream in ith MMS node
CN_i	Consumed network bandwidth for supporting a new movie stream in ith MMS node

When a new movie is enrolled in the HS node, the internal information for each movie is created for parallel processing of MPEG movie. They include a sequence number for each GOP, GOP size and I frame size for each movie. Based on their information, the actual amounts of resources consumed per 1 s are computed to the whole running time. In particular, since both the GOP size and I frame size influence building the streaming buffers in media_service module, they have an impact upon the total memory consumption. The pre-computed data are used to aware the resource consumption for each QoS stream. The data items are composed of $m1_{ij}$ (memory usage for normal play state), $m2_{ij}$ (memory usage for fast forward state), $m3_{ij}$ (memory usage for fast reverse state), $n1_{ij}$ (network usage for normal play state), $n2_{ij}$ (network usage for fast forward state), $n3_{ij}$ (network usage for fast reverse state). These 6 items signify the actual amount of consumed resources for the QoS stream of 1 s. They are stored in the HS node for our RCAAC algorithm.

For example, if the running time of one movie is an hour, 3600×6 items are computed and stored into the HS node. When a client request any arbitrary playing point of a movie A, the HS node determines the admission control for this new client based on both the amount of available resources in MMS nodes and the amount of resources consumed for servicing the corresponding point of the movie A. If the amounts of available resources are bigger than those of resources exhausted by the new client, the admission control algorithm in the HS node accepts the new client. For example, if the 1st MMS node runs the normal play to the movie A, the memory usage is notated as $m1_{1A}$. After this new client is accepted, the item m_{1A} of the Table 5 involves the $m1_{1A}$ to signify the amount of memory consumption by the new normal play. The next equations show the basic equations used in the RCAAC

$$AM_i = \left\{ M_i - S_i - \sum m_{ij} - CM_i \right\}, \tag{1}$$

$$AN_i = \left\{ N_i - \sum n_{ij} - CN_i \right\}, \tag{2}$$

where j denotes all movies running in ith MMS node.

Based on the above Eqs. (1) and (2), the HS node computes the recent value of AM_i and AN_i for each MMS node per 1 s. The update for these values is

performed whenever a new client is accessed and a serviced movie is finished. The M_i and N_i is actual memory capacities for each MMS node and S_i is easily obtained from resource monitoring tools in the HS node.

The computation overheads for these equations are small because the values of $M_i - \sum m_{ij}$ and $N_i - \sum n_{ij}$ for the current client are re-used for the next new client. When a new client is arrived, the results accumulated by the last client are exploited. Thus, the substantial computation in two equations is just the subtraction between CM_i and CN_i per every client entrance. As a result, the overhead for computing the two equations does not greatly affect the performance of the HS node.

In addition to small CPU computation overhead, the RCAAC does not require large memory space for its data items. For example, when a movie has two hours running time and 4 bytes integer is used, the array structure has 7200 elements. The array size is about 170 Kbytes (7200 × 4 bytes × 6 items). Even though 100 movies are running simultaneously, only 17 Mbytes memory is needed for loading the array structures. Furthermore, if the preference movies depend on the Zipf distribution rule that follows the Zipf's Law, only the specific movies are intensively demanded. In that case, since the array for the same movies is re-used after the first time loading, the retrieving overhead between disks and memory is reduced.

```
//  input: the number of request movie
//  output: 1 or 0, 1 means admission,
but 0 means rejection
int Resource_Consumption_Aware_Admis-
sion_Control
                        (int movie_no) {
  for (i = 1; i <= number of MMS nodes;
  i++)
  {
    CM[i]= the amount of memory
    request for a new client;
    CN[i]= the amount of memory
    request for a new client;
  }
  if(the number of current
  streams==0) // it is first access
  {
    for(i = 1; i <= Number of MMS nodes;
    i++)
      {
        AM[i] = M[i] - S[i] - CM[i];
        AN[i] = N[i] - CN[i];
        Store AM[i], AN[i] into files;
      }
      return 1; // admission
    }
    read AM[i], AN[i] from files;
  for(i = 1; i <= Number of MMS nodes;
  i++)
    {
      AM[i] = AM[i] - S[i] - CM[i];
      AN[i] = AN[i] - CN[i];
      If(AM[i] < 0||AN[i] < 0)
        return 0; // rejection
    }
    Store AM[i], AN[i] into files;
    return 1; // admission
  }
```

The above source code shows the pseudo code for internal algorithm of the RCAAC. The input parameter is the number of request movie and the return value is 1 or 0 (1 means the admission for a new client and 0 means the rejection). This algorithm is implemented into the VODCA system and tested under the yardstick program.

Fig. 8 shows the total number of QoS streams supported by the VODCA under RCAAC algorithm and also points out the failed result of QoS under the no admission control. As shown in this figure, when the admission control does not consider, the number of the QoS streams abruptly dropped after that 440 s. The reason is that the new clients added from this point destroy the QoS of all clients currently being serviced. However, under the VODCA with the RCAAC, if the new clients have an negative impact on the QoS for all clients, the RCAAC rejects the request of new clients so that the VODCA system

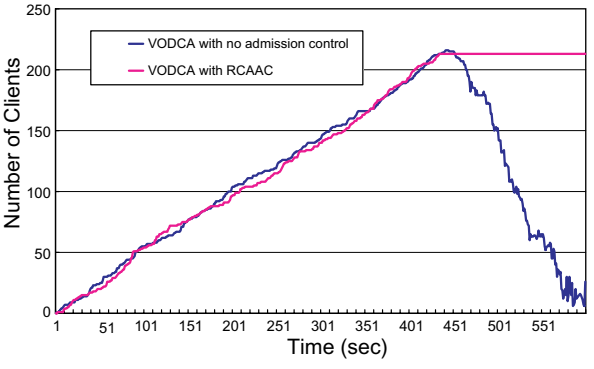

Fig. 8. The number of QoS streams and the admission control.

can guarantee the QoS of the currently serviced streams. As shown in this figure, even if new clients continuously arrive after that 440 s, the VODCA with RCAAC sustains 213 QoS streams constantly. Furthermore the CPU usage of HS that runs the RCAAC is under 10%, because the its algorithm complexity is simpler than others as shown following equation.

$$f(nm) \in O(m), \tag{3}$$

where n denotes the number of concurrent users and m denotes the number of nodes.

6. Relate work

Many research were undertaken for VOD systems to provide a stable service to more users under the various user requirement features and limited resources [13,14,16,19–21,23]. Many VOD systems were implemented and studied for research purposes or for commercial purposes [13,16–18,23]. There were several problems in the previous VOD services. They did not support VCR like functions and actual admission control mechanism in implemented levels because the characteristics of MPEG are passed over. In the actual VOD working environment, the anticipated performance suffers from unexpected internally implemented skill problems.

Much of previous research is focused on topics related to a single video server design such as disk striping, video block placement, and admission control at the level of disks and disk groups [26]. Research in real-time networks, analyzing network conditions for guaranteed services has been discussed [26]. Research in distributed VOD systems focuses on load balancing schemes based on replication and placement techniques [26].

The support of stable QoS in clustered VOD system was studied in many areas. One of them focused into real time schedules based on EDF but this research did not consider the internal resource variation for running the VOD service [22]. There was a study into the way that the server assigns priorities to each picture in MPEG data based on the characteristics of MPEG [15]. In this paper, VOD servers transmit ahead the picture frames with higher priority according to service environments. However, this approach incurred many overheads for measuring the total information about running movies and involved communication overheads among multi-servers.

In addition, research were carried out into QoS based on the effective memory management techniques [27–29]. Since they have re-computed the available memory amount every period for system running, these overheads caused the performance degradation accordingly as the number of servers increased.

For control user admission, many research have been discussed. Many statistical admission controls have been researched. HRM policy that cannot accept arbitrary sets of isochronous and guaranteed-service tasks for execution has been proposed [1]. Three Random variable Admission Control (TRAC) that models a comprehensive set of features of real time storage and retrieval has been studied [24]. Call Admission Control (CAC) Schemes have been adopted to VOD server systems [1,25]. However these admission control schemes not only cause a overhead of computing, but also do not consider VCR-like functions, available resources or striping media between nodes.

From previous research for VOD system, it is clear that further study is needed for reducing the internal overheads to support scalable performance in cluster-based VOD servers. In addition to decreasing overheads for measuring available resources in internal servers, the various streaming modes like to VCR functions should be supported in order commercial VOD services to succeed. However, these additional functions require the changing of MPEG layers and incur the overheads for managing the specific picture frames.

Based on these research requirements for actual VOD services, our research was focused on both reducing overheads to get the internal resource information and supporting the VCR functions to clients.

7. Conclusion

For successful VOD systems, it is important to provide QoS streams to more clients and a more user friendly interface. In this paper, we studied the performance issues for supporting QoS streams in cluster-based VOD servers. Based on our implemented VODCA system, the performance scalability is investigated according to the number of MMS nodes and various client demands including VCR functions are supported. Experiments have shown that a nonlinear scalability phenomenon occurred over

the threshold points. To find out the causes of performance bottleneck, the quantities of internal resources consumed are measured on performance saturation points. From the experiments, it was clear that our limited memory capacity incurred the major bottleneck and the network bandwidth had the possibility to have a negatively impact on the scalable performance. However, there were no burdens in the aspects of disk retrieval and CPU computation ability to get the scalable performance. The reason lies in the parallel disk retrieval and very low CPU consumption in our MMS programs.

In VOD systems, a new client was not allowed that affect the QoS of the currently serviced clients. To guarantee stable QoS streams, we proposed a new admission control called as RCAAC for the cluster-based VOD servers. The RCAAC was based on the amount of available resources in MMS nodes in addition to the actual amount of resource consumed for guaranteeing one QoS stream. The RCAAC increased the utilization of system resources to provide more QoS streams. From the experiment, the RCAAC rejected the new client request and guaranteed the QoS of the currently serviced streams, if there is possibility of this request ruining QoS for every serviced stream.

In our future work, we plan to evaluate the effectiveness of RCAAC in more various streaming media types. The different kinds of MPEG media require the various amounts of resources consumed in VOD services. We will also investigate the method to detect the media type of the required movies and automatically apply the RCAAC to the different streaming media service.

References

[1] Dinkar Sitaram, Asit Dan, Multimedia Servers: Applications, Environments, and Design, Morgan Kaufman Publishers, 2000.

[2] W.C. Feng, M. Lie, Critical bandwidth allocation techniques for stored video delivery across best-effort networks, in: The 20th International Conference on Distributed Computing Systems, April 2000, 201–207.

[3] D.H.C. Du, Y.J. Lee, Scalable server and storage architectures for video streaming, in: IEEE International Conference on Multimedia Computing and Systems, June 1999, pp. 191–206.

[4] K.R. Rao, J.J. Hwang, Techniques and Standards for Image, Video, and Audio Coding, Prentice-Hall PTR, 1996.

[5] http://www.mpeg.org.

[6] Jung-Min Choi, Seung-Won Lee, Ki-Dong Chung, A muticast delivery scheme for VCR operations in a large VOD system, in: The 8th IEEE International Conference on Parallel and Distributed Systems, June 26–29, 2001, pp. 555–561.

[7] Gerard Beekmans, Linux From Scratch Version 3.3. Available from: <http://www.linuxfromscratch.org>.

[8] Tome Fawcett, The Linux Bootdisk HOWTO. Available from: <http://www.tldp.org>.

[9] Qt/Embedded Whitepaper. Available from: <http://trolltech.com/products/embedded/>.

[10] http://mplayerhq.hu.

[11] Brian K. Schmidt, Monica S. Lam, J. Duane Northcutt, The interactive performance of SLIM: a stateless, thin-client architecture, in: ACM SOSP'99, 1999, pp. 31–47.

[12] http://www.seagate-asia.com/.

[13] Jim Gemmell, Harrick M. Vin, Dilip D. Kandlur, P. Venkat Rangan, Lawrence A. Rowe, Multimedia storage servers: a tutorial, IEEE computer 28 (5) (1995) 40–49.

[14] Florin Lahan, Irek Defee, Marius Vlad, Aurelian Pop, Prakash Sastry, Integrated system for multimedia delivery over broadband ip networks, IEEE Transactions on Consumer Electronics 48 (3) (2002) 564–565.

[15] Joseph Kee-Tin Ng, Calvin Kin-Cheung Hui, Wai Wong, A multi-server design for a distributed MPEG video system with streaming support and QoS control, in: IEEE RTCSA, 2000.

[16] Calvin K. Hui, Joseph K. Ng, Wai Wong, Karl R.P.H. Leung, The implementation of a multi-server distributed MPEG video system, in: IEEE RTAS, 2001.

[17] SuperNAVA. Available from: <http://archive.dstc.edu.au/Supernova/>.

[18] VODKA. Available from: <http://vodka.lfcia.org/>.

[19] Sooyong Kang, Heon Y. Yeom, Modeling the caching effect in continuous media servers, ACM Multimedia Tools and Applications 21 (5) (2003) 203–224.

[20] Jack Y.B. Lee, Parallel video servers: a tutorial, IEEE Multimedia (1998) 20–28.

[21] Prashant J. Shenoy, Pawan Goyal, Harrick M. Vin, Issue in multimedia server design, ACM Computing Surveys 27 (4) (1996) 636–639.

[22] Wanghong Yuan, Klara Nahrstedy, Kihun Jim, R-EDF: a reservation-based EDF scheduling algorithm for multiple multimedia task classes, in: IEEE RTAS, 2001.

[23] Craig S. Freedman, David J. DeWitt, The SPIFFI scalable video-on-demand system, in: Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, 1995, pp. 352–363.

[24] Roger Zimmerman, Kun Fu, Comprehensive statistical admission control for streaming media servers, in: ACM International Conference on Multimedia, 2003, pp. 75–85.

[25] Harry G. Perros, Khaled M. Elsayed, Call Admission control schemes: a review, IEEE Communications Magazine 34 (11) (1996) 82–91.

[26] P. Mundur, R. Simon, A. Sood, Integrated admission control in hierarchical video-on-demand systems, IEEE Multimedia Systems (1999).

[27] Sang-Ho Lee, Kyu-Young Whang, Yang-Sae Moon, Wook-Shin Han, Dynamic buffer allocation in video-on-demand systems, IEEE Transactions on Knowledge and Data Engineering 15 (6) (2003) 1535–1551.

[28] Nabil J. Sarhan, Chita R. Das, Caching and scheduling in NAD-based multimedia servers, IEEE Transactions on Parallel and Distributed Systems 15 (10) (2004) 921–933.

[29] Senthi Sengodan, Victor O.K. Li, A shared buffer Architecture for Interactive VOD servers, in: INFOCOM'97, Sixteenth Annual Joint Conference of the IEEE Computer and Communication Societies, April 09–11, 1997, pp. 1341–1348.

Dongmahn Seo received his B.E. and M.E. degrees in Computer Engineering and Computer Information and Telecommunication Engineering from Kangwon National University, in 2002 and 2004, 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.

Joahyoung Lee received his B.E. and M.E. degrees in Information and Telecommunication Engineering and Computer 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.

Yoon Kim received his B.S., M.S., and Ph.D. degrees in Electronic Engineering from the Department of Electronic Engineering, at Korea University, in 1993, 1995, and 2003, respectively. From 1995 to 1999, he was with the LG-Philips LCD Co. where he was involved in research and development on digital image equipments. In 2004, he joined the Department of Electrical and Computer Engineering at Kangwon National University where he is currently an Assistant Professor. His research interests are in the areas of video signal processing, multimedia communications, and sensor network.

Chang Yeol Choi received his B.E. and M.E. degrees from Kyungpook National University, and the Ph.D. degree in computer engineering from Seoul National University. He was with ETRI as a principal engineer responsible for a computer system development from 1984 to 1996. His major interests are computer system architecture, multimedia systems, and mobile computing.

Manbae Kim received his B.S. degree from Hanyang University, in 1983 and the M.S. and Ph.D. degrees in Electrical Engineering from University of Washington, Seattle in 1986 and 1991, respectively. From 1991 to 1998, he was with Samsung Advanced Institute of Technology (SAIT), Korea. He is currently associate professor at Kangwon National University. His research interests include MPEG-21, realistic broadcasting system design and multi-view video processing.

Inbum Jung received his B.S. degree from Korea University, in 1985 and the M.S. 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.