Clustered Multimedia Servers: Architectures and Storage Systems

Hai Jin, Guang Tan and Song Wu

Internet and Cluster Computing Center
Huazhong University of Science and Technology, Wuhan, 430074, China
Email: hjin@hust.edu.cn

Abstract

This paper presents an overview of the recent advances on the design of clustered multimedia servers. It is focused on the following aspects: server architecture, media data organization, real-time stream scheduling and admission control algorithms. For the architecture, we give a taxonomy regarding the various structures of cluster components, and survey over 20 existing server systems, either from the laboratories or the industry. We then examine the data organization methods employed in multimedia servers, which shape the system behavior in many aspects. To reveal the essentials of multimedia service, we take a closer look at the real-time scheduling techniques for streaming data. The admission control issue is finally discussed as an indispensable component for both efficient resource utilization and guaranteed QoS for clients.

1. Introduction

The recent past years have witnessed surprisingly rapid advances in optical communication, high-speed packet switching, data compression, and processor and memory design technologies, which have made it feasible and economically viable to provide a variety of on-line media services for network clients. Multimedia mail, orchestrated presentations, high quality video-on-demand, and virtual reality environments are a few examples of such applications. An application in this environment will typically use one or more media streams, such as audio, graphics, video, images and text. Due to the data intensive nature of the component media streams, these applications impose stringent demands on the multimedia server system.

The primary requirements of a large-scale multimedia server are:

1. Large storage system and network bandwidth: The data intensive and periodic nature of multimedia streams demands large amounts of network and storage system throughput. For example, a video-on-demand server that supports one thousand users with HDTV quality movies will require network and storage bandwidth in excess of 20 Gbps. Storage throughput of this level is two orders of magnitude more than that observed today with the state-of-the-art storage technologies such as Redundant Arrays of Inexpensive Disks (RAID).

2. Real-time service: The periodic nature of the multimedia data necessitates QoS guarantees in the form of guaranteed throughput and bounded latency for all active streams. This requirement presents challenges for the operating system and application software design. A general-purpose operating system may not efficiently exploit the available resource without special consideration for the characteristics of multimedia services. Moreover, a multimedia server should take into account mixed workload in service as the consumers' media services requirement become more and more diversified. For example, an interactive electronic meeting application may simultaneously generate video, audio requests which exhibit periodic characteristic, text and image requests which require a large throughput service, and interactive requests which require a best-effort service for as prompt responses as possible. These different requirements demand different resource scheduling, which must be considered in a uniform framework.

3. Large storage capacity: Given the storage intensive nature of multimedia data, the collective storage requirements for thousands of multimedia documents may exceed tens of terabytes. For example, a movie server with two hundred HDTV quality (20 Mbps, 2 hour long) movies will require roughly 3.6 terabytes of storage. Similarly, a multimedia storage server that stores a large number of multimedia documents each composed of multiple media streams will require a comparable storage capacity.

The problems of designing large capacity, scalable multimedia servers and providing guaranteed bandwidth with desired QoS have been researched widely and the outcomes of this research are being commercialized steadily. Currently, there are two major types server system architectures:

1. **Shared memory multiprocessors** In a multiprocessor system, a set of storage nodes and a set of computing nodes are connected to a shared memory. The data to be retrieved is sent to memory buffers through a high-speed network or bus, and then to clients. A mass storage system has presented the capacity of supporting hundreds of media streams.
2. **Distributed memory clustered architectures** A clustered server consists of a set of server nodes interconnected by a high-speed network. Each node has independent storage system and separate physical memory address. These nodes are also divided into several classes, for example, a set of storage nodes responsible for data storage and retrieval and a set of delivery nodes responsible for data collecting and transmitting.

Compared with the first solution, the cluster-based architecture is relatively more scalable due to the loosely coupled architecture. In practice, it's easy to scale up such a system to hundreds or thousands of server nodes. What's more, the off-the-shelf commodity components present great price advantage over the shared memory architecture. Hence the cluster-based architecture is more economically viable. For these reasons a lot of researchers have contributed their efforts on the study of cluster-based multimedia server over the past years. These researches focus on different aspects of the server design including server architecture, storage sub-system, network sub-system, streaming protocols, etc. In this paper, we'll concentrate on several most important issues of the server design: server architecture, storage system, disk scheduling algorithms and admission control algorithms.

The server architecture determines how different components of the cluster are assigned with different roles, and hence how the data is flowing through the server; the storage system is a major concern when considering the sub-systems of the server. It involves several design issues such as load balancing, fault tolerance, etc., which together shape the system behavior in many aspects. The disk scheduling algorithms and admission control algorithms determine how data requests are performed over the disks and how to accommodating as more clients as possible with limited resource, while not violating the real-time requirements of admitted clients. So far as we know, there have been quite a few literature documenting the development of storage system design [36][29][71], either for a single machine or a distributed server, but few comprehensive discussion of clustered multimedia server architecture can be found. This paper survey over twenty server systems and develop a taxonomy for the server architecture. For the storage system issues, we intend to give an overview over the most recent advances in this area.

The remainder of this paper is organized as follows: section 2 gives a detailed overview on the server architectures; section 3 discusses various techniques involved in storage system design; to address the real-time requirements of multimedia service, we describe some disk scheduling algorithms in section 4, specially we present some recent research on the mixed media workload scheduling. In section 5 we examine a set of admission control algorithms. Finally we conclude in section 6.

2. Clustered Multimedia Server Architecture

In a clustered multimedia server consisting of multiple server nodes, there are two basic kinds of nodes. One is called access servers, which serve as the contact points for the clients. Any client intending to open a media stream should always firstly contact with the access server, which will handle the admission control and provide the client with necessary information for successive streaming operations. The other kind of nodes composes the storage sub-system, the data container of media data. They can be PCs or workstations equipped with multiple disks, or storage devices directly attached to the internal network, for example, FC disks connecting to the FC switch using SCSI protocol. The requested media data is retrieved from the storage sub-system and sent out over the network. They may or may not be returned through the access server.

Whether the requested media data route via the access servers or not plays an important role in the data flow of the server system, and hence have significant influence upon the system behavior. From this prospective, a clustered multimedia server can have two data flow modes: (1) Proxy-based Mode and (2) Direct Access Mode (See Fig.2.1 and Fig. 2.2, respectively). In proxy-based mode, the access servers work like proxies located between the clients and the media data, taking responsibility of fetching data from the storage sub-system and then forwarding them to the clients. The storage sub-system is hidden behind the access servers. By contrast, in direct access mode the requested data is directly sent to the clients from the storage sub-system without an intermediate forwarding by the access servers. However, the clients may need to contact with individual storage node to retrieve data by themselves.

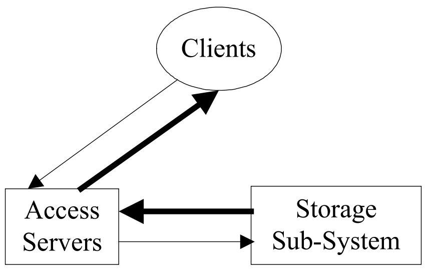


Fig. 2.1 Proxy-based mode

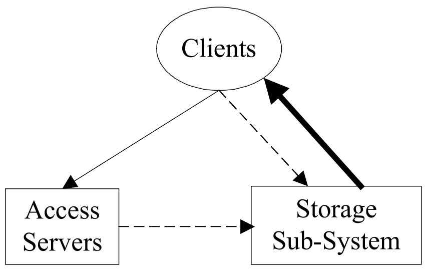


Fig.2.2 Direct Access mode

For the Proxy-base working mode, there are two possible organizations of the server components, and they result in two server architectures: *two-tier* architecture and *flat* architecture. Also, server systems with Direct Access mode can be further divided into two categories: *multiple-access-point (MAP)* architecture and *single-access-point (SAP)* architecture. This classification can be described in a hierarchy structure as shown in Fig. 2.3.

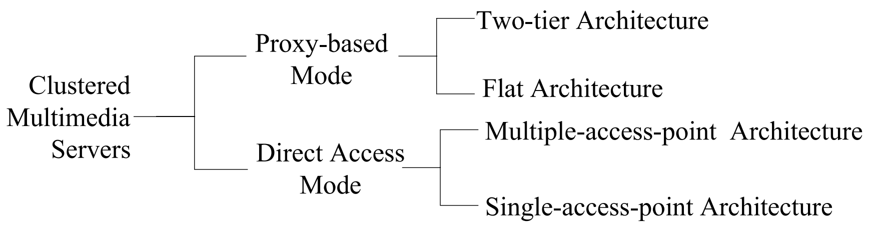

Clustered Multimedia Servers
Proxy-based Mode
Direct Access Mode
Two-tier Architecture
Flat Architecture
Multiple-access-point Architecture
Single-access-point Architecture

Fig. 2.3 Classification of Clustered Multimedia Server Architectures

2.1 Proxy-base Mode

In a proxy-based system, there exists a Proxy Server (PS) in addition to the Access Server (AS) and Storage Server (SS). The Proxy Server serves as stream combiner in a server with distributed storage system. In such a system, the data is striped across multiple storage nodes, and a client stream may be divided into multiple concurrent sub-streams from several storage nodes. These sub-streams should be synchronized so as to form a coherent stream playable for the clients. Depending on the storage policy, the Proxy Server can be a software module coexisting with other system components, an independently running server or a special-purpose device. Proxy Server is logically an interface between the storage sub-system and the applications-specific software.

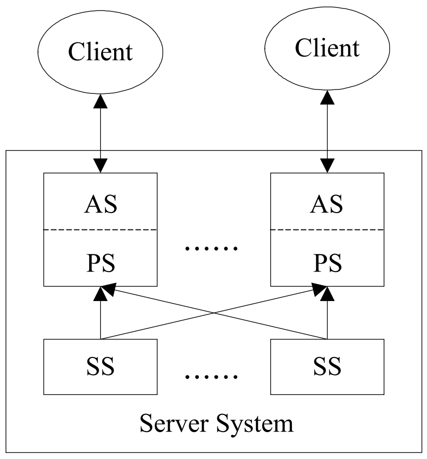


Fig. 2.4 Two-tier Architecture

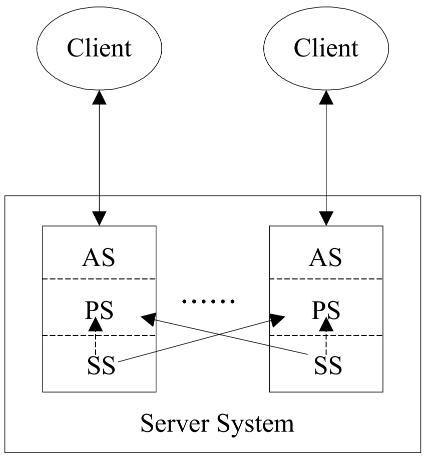


Fig.2.5 Flat Architecture

Fig. 2.4 presents the Two-tier Architecture. In this architecture, the Access Server and Proxy Server reside on the same host, which is separated from the Storage Server. The multiple storage nodes are hidden behind the access servers and only take care of data storage and retrieval. With this organization, the logic functionalities of various nodes are clear and the implementation can therefore be simplified. For example, the parallel file systems such as Tiger Shark File System [43], Symphony [71], etc. can be easily imported to the server to obtain specially optimized real-time storage and retrieval. By virtue of the Proxy Server, the Access Server needs not be aware of the storage system's distributed nature, and can focus on the interaction with clients according to the standardized protocols such as RTSP [70], RTP/RTCP [69], etc.

One characteristic of this architecture is the requested data generally make two trips on their way to clients: from the storage nodes to the proxy servers, and then from the proxy servers to the clients. So the actual transmission for every N Bytes is 2N bytes inside the server system.

In Flat Architecture, as shown in Fig.2.5, the Access Server, Proxy Server and Storage Server all exist on each server node, which result in a symmetric structure. Having essentially the same data flow, this architecture shares many characteristics with the two-tier architecture. One exception is that the actual transmission for N bytes of requested data. Assuming there are M nodes in the system and they evenly service the data requests, then (2M-1)N/M bytes of data transmission is needed for every N bytes of data. As in the two-tier architecture, there exists heavy extra data traffic for a requested stream, and thus forms a potential bottleneck for the internal network and server processing capability.

In practice, both architectures need some mechanism to direct client requests to different Access Servers in order to obtain load balancing, this can be realized by a special front-end machine [56], or by using Round Robin DNS [13] or some Layer 4 switching techniques [23].

2.2 Direct Access Mode

The Multiple-access-point (MAP) Architecture and Single-access-point (SAP) Architecture are described in Fig.2.6 and Fig.2.7 respectively. In MAP architecture, the clients can access the Storage Server directly. The Access Server here is often a meta-data server providing the clients with the data distribution information.

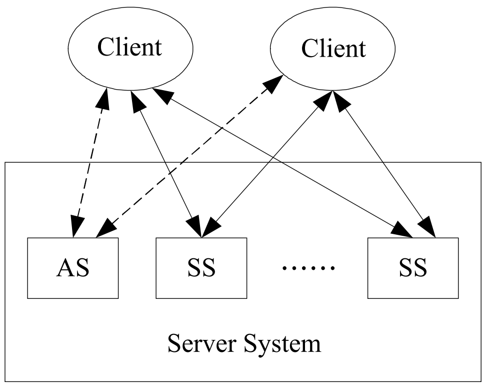


Fig.2.6 Multiple-access-point

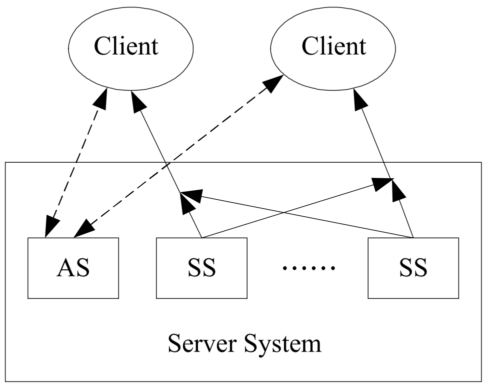


Fig.2.7 Single-access-point

The exposal of the Storage Servers brings some differences as compared to the previous two architectures. On the one hand, this architecture requires only half the amount of data transfer, since the requested data does not require an extra collect-and-forward process at the server side. On the other hand, as it need to contact with individual Storage Server separately, the client can't get a single system image of the distributed server system, which may affect the interoperability of the service, that is, a client can't request the data without knowing the server internals, even though it is conforming to widely accepted standard protocols.

The proposal of SAP architecture overcomes the drawback of MAP's lacking client transparency. As shown in Fig. 2.7, this architecture also hides all the Storage Servers and gives the clients a single contact point, the Access Server. The client connects to the Access Server to request a stream service, and then receive data from the same address, just like interacting with a single machine. This client transparency is achieved through the strict control of concurrent data streams by the access server. In a push-based servicing mode, the data for a client stream can be pushed concurrently from multiple Storage Servers, so the synchronization problem presents a challenge for the implementation. When a pull-based servicing mode is adopted, the Access Server can help to relay the clients' separate data requests to appropriate storage nodes, which will accordingly retrieve the requested data and transmit them to the clients directly.

One weakness of this architecture is that a part of application software is required to be bounded with the Storage Servers in addition to their responsibility of data storage and retrieval. For some storage architectures like SAN (Storage Area Network) [29]or NAS (Network Attached Storage) [38], in which the storage nodes are composed of a variety of heterogeneous storage devices such as disk arrays, tape libraries and optical storage arrays, this is sometimes unfeasible due to the limited software supports by these devices.

It should be noted that the high scalability achieved by the SAP architecture is often limited to its outbound data throughput. As for the inbound throughput, the scalability is restricted by the Access Server, the single entry point of the system. As in the Proxy-based mode, this problem can be solved by a Round Robin DNS [13]or some other load balancing mechanisms [23][25].

2.3 Examples of Existing Systems and Projects

2.3.1 Two-tier Architecture

The Multimedia Servers based on IBM's Tiger Shark file system [43] are typical examples of two-tier architecture. As shown in Fig. 2.8, the server nodes are divided into file system nodes and storage nodes. The file system nodes here represent the Access Servers of the Proxy-based mode. The whole system is constructed based on the Tiger Shark, a parallel file system designed to support interactive multimedia on IBM's AIX operating system. Note that the file system nodes and storage nodes are not necessary distinct nodes since it's very simple to combine the two roles on one node, and with that design the server is turned to a flat architecture.

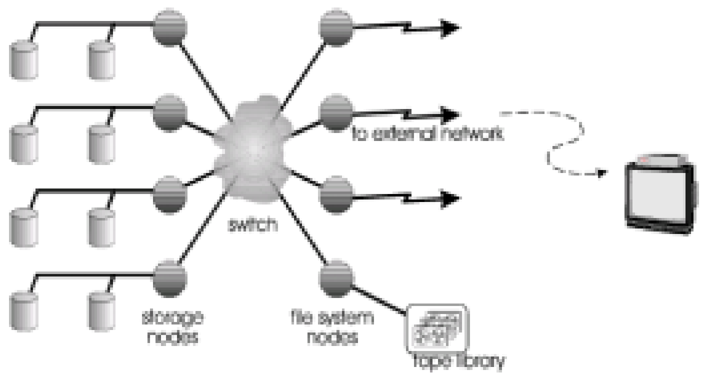


Fig.2.8 Multimedia Server based on Tiger Shark file system

Some other multimedia servers, for instance, the NASD-based VoD system [8], adopt similar architecture. Instead of using workstation or PC, the system employs NASD (Network Attached Secure Disk) as its storage elements in an attempt to reduce the cost. In the proposed system, the function of the Access Servers is accomplished by the so-called Merging Servers.

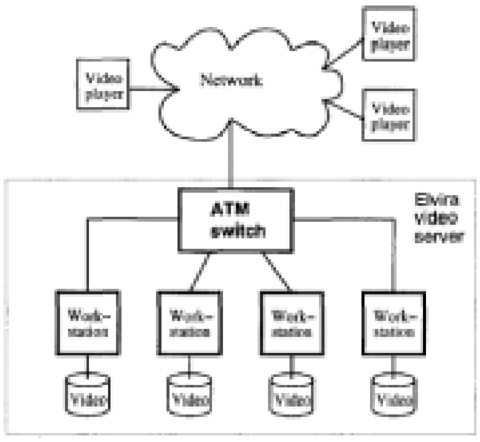
[22] proposed a FC-based cluster media server. It differs from the above systems in that its FC-based system, rather than connecting storage devices to the storage server (often a full-fledged machine), directly connects storage devices to the switch, thus eliminating the storage nodes all together. This is because Fiber Channel devices, including FC disks, FC switches, and FC host interface cards, can communicate directly using the SCSI protocol, which enables the "true" direct attachment of storage devices to the interconnect. The

elimination of storage nodes reduces the system cost. However, the FC switch-based system has several weaknesses: first, the per port cost of an FC switch is much higher than the general purpose interconnect such as Ethernet; second, the FC-based scheme can not support heterogeneous server nodes, which is easy for the traditional interconnect based system, and the scalability of the such systems is severely restricted by the FC-switch.

VoDKA project [89] aims to build a Linux cluster-based Video-on-Demand server using a hierarchical structure. The three-level hierarchical structure is composed of: (1) the storage level consisting of mass storage devices; (2) the cache level responsible for data retrieval and scheduling; (3) the streaming level in charge of buffering and protocol adaptation. The cache level consists of a set of cluster nodes with local storage; and the streaming level is composed of a set of nodes called cluster heads. In fact, the cache level is corresponding to the storage nodes and the streaming level corresponds to the access nodes.

2.3.2 Flat Architecture

The Elvira video server [68] is built on a cluster of standard UNIX workstations interconnected by an ATM switch, as Fig 2.9 shows. There is an integrator process on each workstation responsible for stream integration. When the Integrator process gets a stream request, it contacts the Video Pump processes on all the machines which have to participate to fulfill the request, and instructs them to set up connections for video delivery[1].



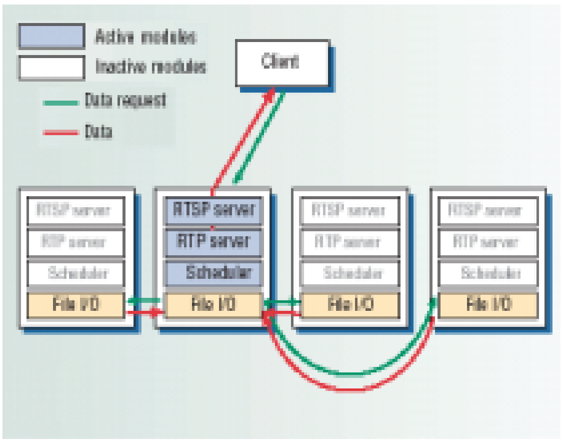


Fig.2.9 Elvira Architecture **Fig. 2.10 Yima-1 Architecture**

Yima-1 [74] has adopted nearly the same architecture as Elvira. As shown in Fig. 2.10, the Yima-1 software consists of two components: Yima-1 distributed file system and Yima-1 media-streaming server. The distributed file system consists of multiple file I/O modules located on each node, and provides a complete view of all the data on every node. The media-streaming server itself is composed of a scheduler, a real-time streaming protocol (RTSP) module, and a real-time protocol (RTP) module. The multiple media-streaming

[1] In Elvira, the video can also be stored in a non-striping style. This storage policy obviates the need of data assembling but leads to another data flow mode, as will be discussed latter in section 2.3.3.

servers share the clients' requests with a RR-DNS load balancing mechanism. For a specific client session, only one node runs the media-streaming server while all nodes run the file I/O module, forming a master-slave structure. With this design, an application running on a specific node operates on all local and remote files.

With the Yima-1 architecture, several major performance problems offset the ease of using clustered storage, such as a single point of failure at the master node and heavy inter-node traffic.

The SESAME-KB [12] parallel video sever shares many characteristics with the Elvira system. The responsibilities of the Access Server and Storage Server are assumed by GOM (Global Object Manager) and LOM (Local Object Manager) respectively. The GOM receives the queries and performs the admission control. If a query is admitted, the GOM will choose one of the nodes involved to execute the query. Also this node's function is similar to the Integrator in Elsvira.

Some researches particularly address the storage system design. In [11], multiple PCs with local storage units are interconnected with Myrinet network. Each cluster node can at the same time serve clients and handle part of the storage. The cluster file system, which relies on a Cluster Communication Library, provides the upper layer RTSP daemons with a global view of the distributed file system. In contrast to its kernel-level implementation, the CrownFS [63] realizes the similar file-related functions at user-level. Both file system fall into the category of flat architecture.

2.3.3 Multiple-access-point (MAP) Architecture

The MAP architecture is first proposed as Server Array architecture in 1995 [6]. In the proposed system, as shown in Fig. 2.11, a set of symmetric servers is servicing the clients' request in parallel, just like a disk array in a traditional machine. A single stream is distributed over several server nodes of the server array. Each server node only stores a sub-stream of the original stream. The clients are responsible to split a stream into sub-streams for storage and to re-combine the sub-streams during the retrieval of a stream. When retrieving media data, the client must set-up a data connection to each of the server nodes and synchronizes the streams using a specific protocol.

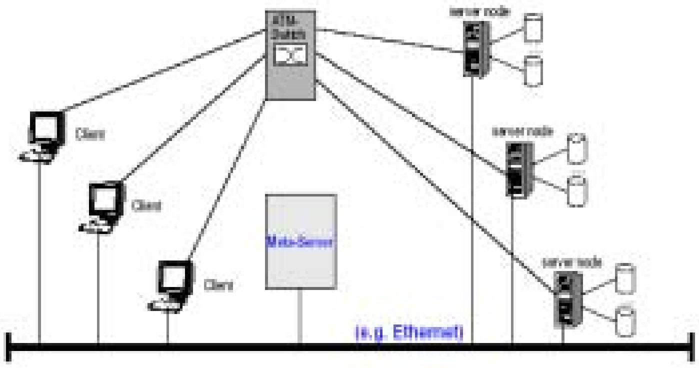


Fig. 2.11. Server Array Configuration

Calliope [44] is a MAP system with non-striping storage policy. It uses the *Coordinator* as the access server and *Multimedia Storage Unit* (MSU) as the storage server. The Coordinator is in charge of request authentication, resource allocation, etc. Upon receiving a read request, the Coordinator finds a MSU with enough resource to serve the request. The request may be queued if not enough resource is available. The authors adopt the non-striping storage instead of more prevalent striping policy because they believe the striped environment will introduces undesirable latency and complexity of management.

Autonomous Network Attached Disks [1][2] is designed to be building blocks for a multimedia file systems. Like CMU-NASD [37], it also achieves direct transfer between client and storage elements in a networked environment. The storage nodes have specific modules called AD-DFS in their operating system kernels, and the storage nodes send the requested data via their own network interfaces. The client needs particular module in its OS kernel to understand the block-based object interface exported by AD-DFS. This mechanism is exactly like the idea adopted by MAP architecture.

The same working mechanism of MAP architecture can also be found in some implementations like SPIFFI [30] and CANDID [78], etc. Other systems like [34], Viola [90] and the researches by Jack Lee *et al.* [53][54] consider the same architecture, too. In [34], the issues of scheduling of a large number of video objects as well as the reliability aspects are addressed. Jack Lee *et al.* particularly analyze the synchronization and scheduling problems of the MAP architecture in both client-pull and server-push modes.

2.3.4 Single-access-point (SAP) Architecture

The single-access-point architecture differs from the MAP architecture in its possible single system image (SSI) provided for the clients. While the single system image seems to be just a functional extension from the MAP architecture, we believe it is an important and promising feature for an Internet-oriented service. With this feature, client software conforming to the open standard protocols interacts with the multimedia servers anywhere on any platform, despite of the various server implementations. This is like the relationship between the Web browser and the Web servers. Moreover, this functional requirement does present several challenges for the server design and implementation, such as inter-server synchronization, fault masking and QoS control.

The design of Yima-2 [74] (See Fig. 2.12) is directly motivated by the drawbacks of Yima-1, which has unsatisfactory scalability due to the heavy inter-node traffic. The Yima-2 system eliminates the overhead by making the RTP servers send their local data through their own network interfaces. From the viewpoint of a client, there is only one RTSP server, and the data flow from multiple RTP servers is controlled by a PAUSE/RESUME mechanism. As quality control functionality, selective packets retransmission is realized by extending the standard RTP protocol. But the retransmission mechanism requires the client to contact with the individual node separately, which may make it invisible by those clients without RTP extension.

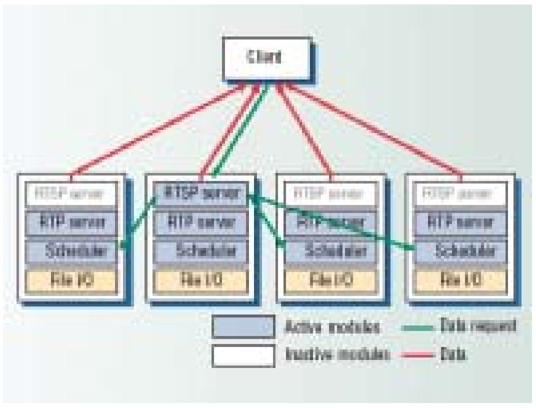


Fig.2.12 Yima-2 architecture

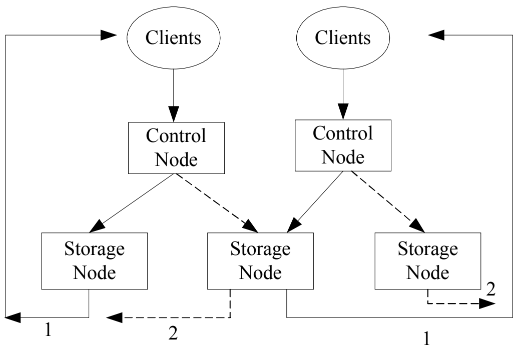


Fig. 2.13 WanLan architecture

WanLan video server [80] is another example of this architecture, as illustrated by Fig.2.13. In WanLan system, a movie is partitioned into multiple segments across the storage nodes, each segment being a playable section. When playing a movie for a client, the server instructs the stream to move from node to node according to the playing order. Here a client stream is not divided into multiple parallel sub-streams as in the Yima-2 server. Instead it is transferred as an integral stream. The system load balancing is achieved by overlapping many such streams and with a dynamic streaming scheduling algorithm. Though the system adopts a coarse-grain parallelism, the performance is proved to be fairly good, and the synchronization problem is greatly simplified. In WanLan, the media packets encapsulated in RTP protocol are sent out from the storage nodes, while the incoming request and feedback (in RTCP protocol) are taken over by the control node. There can be multiple control nodes sharing the client connections under the scheduling of Linux Virtual Server 1.1.1[56].

DAVID [15] achieves a fine-grain parallelism. A client's data is concurrently pushed out from multiple storage nodes. The synchronization is controlled by the access server, which sends a command message to one storage node for each data block to be transmitted. Though the load balancing effect is good, the message traffic inside the cluster system brings non-trivial overhead for the internal network, therefore restricting the scalability of the system.

MARS [14] is a system more tightly coupled compared to WanLan and DAVID server. One key component in MARS is an ASIC (Application Specific Integrated Circuit) called APIC (ATM Port Interconnect Controller), which provides a direct interface for the host (workstations as well as servers) and a variety of I/O devices. The storage nodes put the requested data in its dual ported RAM, and the APICs fetch the data and transfer it to clients in a synchronized manner. The connection maintenance and flow control are also performed by the APICs.

Microsoft's Tiger [10] is a special-purpose file system for video servers distributing data over ATM networks. It consists of a number of nodes (called Cubs) acting under a central-controller node's direction. Files are striped across all disks of all the nodes. To play a video stream, Tiger establishes a multipoint-to-point ATM switched virtual circuit between every node and the user. The data synchronization is realized by a distributed scheduling mechanism. One major disadvantage of Tiger is its reliance on ATM's multipoint-to-point characteristic. In a non-ATM environment, extra processors may be needed to combine the ATM packets into stream.

2.4 Comparison

In this section we present a comprehensive comparison of the various architectures (See Fig.2.14). We compare them from several aspects: scalability, adaptability and client transparency. For each architecture, we list the corresponding projects/systems introduced before.

Note the scalability we discuss here is in term of the ultimate number of supported clients by the server, but not the storage system bandwidth alone, as done by most existing literatures. In fact, how the data is delivered from the server to a large number of clients or vice versa over the Internet in a scalable manner must be considered, in addition to the scale-up of storage system. To do this, all the architectures except MAP architecture need some mechanism to balance the network load among multiple front-end machines (the Access Servers). Some systems simply assume a RR-DNS used to accomplish this task [74], while others may balance the load inside the server through dynamic scheduling. However, the load balancing capability of RR-DNS is proved to be quite limited [28][61], and the server-internal scheduling usually cannot work with the inbound traffic, because the server cannot determine for which node the data should be destined. Therefore, we take into account the load balancing effect on the AS's when discussing the system scalability. For all the architectures we list the potential factors that are most likely to affecting the system performance and hence the system scalability.

Categories \ Features		Network transfer per byte[1]	Factors affecting system scalability[2]		Suitable Environment	Client Trans-parency	Example Projects/ Systems
			Outbound bandwidth	Inbound bandwidth			
Proxy-based Mode	Two-tier	2	Internal net bandwidth AS load balancing SS load-balancing	Internal net bandwidth AS load balancing SS load-balancing	LAN WAN Internet	Yes	Tiger Shark based system[43] NASD-based system[8] FC-based system[22] VoDKA[89]
Proxy-based Mode	Flat	$\frac{2M-1}{M}$	Internal net bandwidth AS load balancing SS load-balancing	Internal net bandwidth AS load balancing SS load-balancing	LAN WAN Internet	Yes	Elvira[66]; Yima-1[74] SESAME-KB [12]; [11] CrownFS [63]
Direct-Access Mode	MAP	1	SS load-balancing	SS load-balancing	LAN	No	Server Array[6]; Calliope[44] Autonomous NAS [1][2] CMU-NASD[36]; SPIFFI[29] CANDID[78];[34]; Viola[89]
Direct-Access Mode	SAP	1	SS load-balancing	AS load balancing SS load-balancing	LAN WAN Internet	Yes	Yima-2[74]; WanLan [80] DAVID [15]; MARS [14] Microsoft's Tiger [10]

Fig.2.14 Comparison of Clustered Multimedia Server Architectures

[1] *M* refers to the number of storage nodes.
[2] AS: Access Server; SS: Storage Server

3. Storage System and Media Data Organization

In multimedia servers, the storage subsystem is one of the most important components since it provides the content and I/O bandwidth for data retrieval [36]. To support hundreds or thousands of simultaneous sessions or streams, the storage subsystem often consists of a large number of storage nodes and disks, which together meet the requests for great amount of data. Data layout in the storage subsystem has significant influence on system availability and load balancing because it determines how the storage nodes share the load and how to response to node failures. In this section we fist discuss several designing issues of the storage subsystem and then focus on media data placement schemes of the clustered multimedia server.

3.1 Considerations in Storage System Design

High throughput, large capacity, fault-tolerance and load-balancing are four major challenges for the storage systems of clustered multimedia servers. In what follows, we will discuss the storage system configuration from these aspects.

3.1.1 High Throughput

Storing an entire media file on one node limits the number of concurrent accesses to that object. The throughput of that node dictates the number of clients retrieving the same file. To overcome this limitation, data striping techniques was proposed. A media file, under data striping schemes, is divided into many segments and scattered the set of nodes in a clustered multimedia server. For example, in Fig. 3.1, 12 segments of a continuous media file are placed on the 6 nodes (labeled as S[i]) in a round robin manner and Segment 0-5 of this file can be accessed in parallel. Besides increasing throughput, an important issue in design of a data-striping scheme is to balance the load of most heavily loaded nodes while keeping latency small. Data striping evenly distributes the service load to all the nodes in the clustered multimedia server to avoid overload situations. The load-balancing issues will be discussed in detail in Section 3.1.4.

With data striping techniques, there are two possible methods of data retrieval. One method is, for each stream, to access each node in a service round. This retrieval method expects to ensure a perfectly balanced load for the nodes. However, it requires more buffer space per stream and has to face the synchronization problem among the nodes. In the other retrieval method, for a given stream, in each round, data is extracted from one of the nodes. Hence, the data retrieval for the stream cycles through the set of nodes. In order to maximize the system throughput, it is necessary to ensure that in each round the retrieval load is balanced across the nodes. Given that each stream cycles through the node set, this load balancing can be achieved by staggering the streams. With staggering, each steam considers the round to begin at a different time.

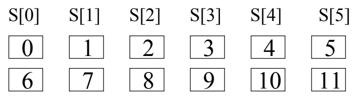


Fig. 3.1. Data Striping and Layout in a Round Robin Manner

3.1.2 Large Storage Capacity

The cost for large media files is prohibitively high if a large number of disks are used for storage. To keep the storage cut down, tertiary storage must be added such as automated tape libraries and optical jukebox. Hierarchical storage architecture can be used to reduce the overall cost. Under this architecture, only a fraction of the total storage is kept on disks while the major remaining portion is kept on a tertiary system. Frequently requested media files are kept on disks for quick access and the remainder resides in the tape library.

To deploy multimedia services at a very large scale, a storage-area-network (SAN) architecture was proposed [29][42]. An SAN can provide high-speed data pipes between storage devices and hosts at far greater distances than conventional host-attached small-computer-systems-interface (SCSI). The connections in an SAN can be direct links between specific storage devices and individual hosts, through fiber-channel arbitrated loop (FC-AL) connections; or the connections in an SAN can form a matrix through a fiber channel switch. With these high-speed connections, an SAN is able to provide a many-to-many relationship between heterogeneous storage devices (e.g., disk arrays, tape libraries, and optical storage arrays), and multiple nodes in the clustered multimedia server.

Another approach to deploy large-scale storage is network-attached storage (NAS) [38]. Different from SAN, NAS equipment can attach to a local area network (LAN) or a wide area network (WAN) directly. This is because NAS equipment includes a file system such as network file system (NFS) and can run on Ethernet, asynchronous transfer mode (ATM), and fiber distributed data interface (FDDI). The protocols that NAS uses include hypertext transfer protocol (HTTP), NFS, TCP, UDP, and IP. On the other hand, both NAS and SAN achieve data separation from the application server so that storage management can be simplified. Specifically, both NAS and SAN can obtain high scalability.

3.1.3 Fault Tolerance

To build a scalable multimedia server, fault tolerance issues must be considered because with the increase of components, the probability of node failures may increase beyond an unacceptable level. In order to ensure uninterrupted service even in the presence of node failures, a server must be able to reconstruct lost information. This can be achieved by using redundant information. The redundant information could be either parity data generated by error-correcting codes like FEC or duplicate data on separate nodes. That is, there are two kinds of fault-tolerant techniques: parity-based and replica-based schemes. Parity-based schemes add a small storage overhead but require synchronization of reads and additional processing time to decode lost information. In contrast, replica-based schemes do not require synchronization of reads or additional processing time to decode lost information, which significantly simplifies design and implementation of multimedia servers. However, replicating incurs at least twice as much storage volume as in the non-fault-tolerant case. As a result, there is a tradeoff between availability and complexity. A study in [81] shows that, for the same degree of availability, replica-based schemes always outperform parity-based schemes in terms of per-stream cost, as well as restart latency after node failure.

Though replica-based schemes have a higher storage overhead than parity-based ones, they can offer better performance in term of throughput, response time and availability, especially in the fault mode. Moreover, the trend of disk improvement is that the storage space capacity improves much faster than the effective I/O bandwidth of disks due to the advancement of recording density. Note that many multimedia applications such as video service may be I/O bandwidth bounded instead of storage capacity bounded because there are a large number of concurrent accesses to a few popular object. Therefore, it is feasible to trade in more data redundancy in order to achieve more availability and better performance. In this research we only focus on the replica-based schemes rather than the parity-based ones.

3.1.4 Load Balancing

The load balancing issue stems from different object popularities among the movie library, which is a nontrivial issue when modeling and designing large-scale multimedia servers. Because some media objects are more popular than others, the retrieval of media objects is highly skewed in many multimedia applications such as video-on-demand [16][41][57][62]. A multimedia server with large amount of video objects has to take video popularity into account. There are two kinds of video popularity, inter- and intra- movie skewness [92]. The former is the popularity among different video objects and the latter describes the popularity among different parts in the video object.

3.1.4.1 Inter-movie skewness

In a multimedia server, the demand for the video objects is usually skewed. For example, newly released movies are likely to attract most of the viewers while older movies receive very few requests. It has been shown that this inter-movie skewness can be characterized by a Zipf distribution. The Zipf [5] distribution is frequently used to express the probability of selection of a particular object from a fixed number of objects where there is a skewness toward some of the objects. It is defined as

$$Z(i) = \frac{C}{i^{1-\alpha}},\quad C = 1 \Big/ \sum_{i=1}^{m} \frac{1}{i^{1-\alpha}} \tag{1}$$

In this formula, m is the number of available video objects and i is the index of a video object in the list of m objects that are sorted in the order of decreasing popularity. α is the parameter specifying the skewness. A uniform distribution corresponds to its value of 1, and a value of 0 represents a highly skewed distribution. The value of α for 92 video objects is about 0.271 [27] and in this case the most popular 10 video objects receive about 50% of the total viewing requests.

The large discrepancy among the retrieval rates of video objects caused by the inter-movie skewness probably leads to load imbalance and hot spot problem. It is obvious that a node with too many hot video objects is a potential bottleneck of the whole server system. There are two common ways to reduce the load imbalance caused by this skewness. One is data replicating and the other is data striping. Replication strategies [9][33][85] replicate the popular video objects and balance the workload by distributing user requests to several replicas. However, replication algorithms likely cost lots of system resources such as storage capacity, disk and network bandwidth. With data striping techniques as shown in Section 3.1.1, the video objects are striped into segments and stored across multiple nodes, so the service load can be evenly distributed and the utilization of system servicing capacity can be maximized. Data striping and data replicating techniques can be combined in a large-scale clustered multimedia server. Although striping of media data can balance the utilization among the nodes, such technique across a large number of nodes may exhibit additional complexity, for instance, in data management. Therefore, it is more practical to limit the extent of data striping. The nodes can be arranged in groups and only intra-group data striping is allowed, while popular media objects are replicated among the groups [55].

3.1.4.2 Intra-movie skewness

When viewing a video object, not all of clients view it all the way to the end. It is possible that some clients stop watching halfway and have different viewing time. A familiar example is that the beginning segment of a video object is more popular than others because of sequential access pattern of video applications. We call this instance the intra-movie skewness. In the clustered multimedia servers, because video objects are striped into many segments stored on multiple nodes, the intra-movie skewness may have considerable influence on the media data placement.

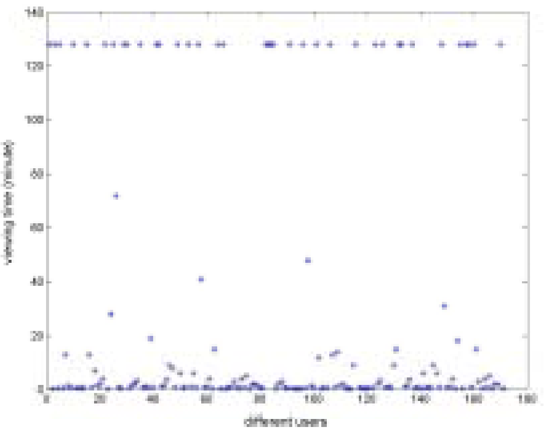


Fig.3.2. Viewing Time of Different Users

Fig. 3.2 shows the users' viewing time of a video object. The statistic data in the Fig. is extracted from the log files of a multimedia server [92]. From the Fig. we can see that only some of users completed viewing the whole video object, and others stopped halfway. Moreover, most of those who had canceled watching stopped at the beginning of the video object. Hence, in Fig. 3.1, the 12 segments of the video object (hence the 6 nodes) probably have different opportunities to service the users' requests. Segment 0 likely has higher retrieval rate than Segment 11, so S[0] may take on more workload than S[5]. The segments have uniform retrieval rate only if the users have the same stop probabilities on the segments, and in the extreme case, all the users have viewed the whole video object.

For instance, Fig. 3.3 and 3.4 show the segment retrieval rate and users' stop probability of several video objects [92]. In Fig. 3.3 we can see that segment retrieval frequency decreases rapidly and nearly linearly at the head of movies. At other parts it declines tenderly and continuously. Fig. 3.4 indicates lots of users stop watching before the end and most of them halt at the movie's beginning. These two Fig.s show us the existence of the intra-movie skewness that leads to large gap between the retrieval rates of different video segments.

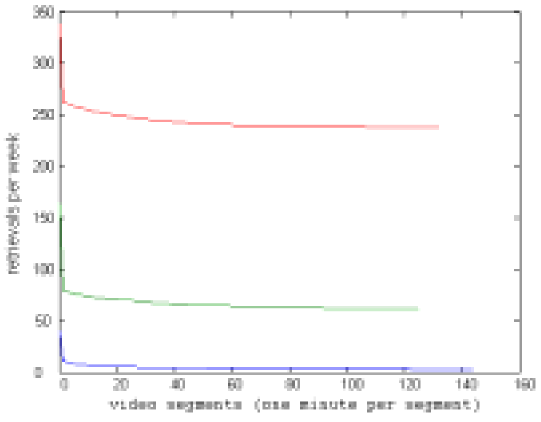


Fig. 3.3. Retrieval Rate of Video Segments

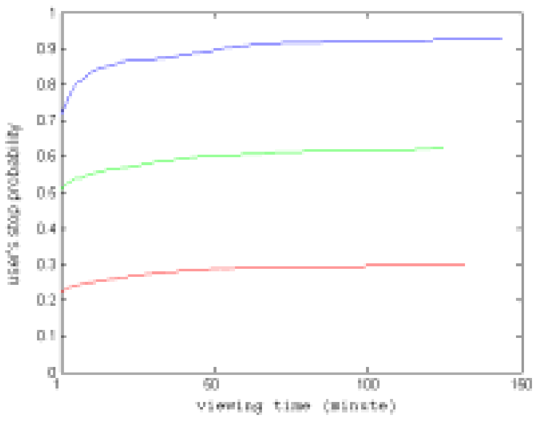


Fig. 3.4. Users' Stop Probability

The intra-movie skewness has two main reasons. Firstly, before a client view a video object, it is hard to estimate whether he or she will watch it throughout or stop halfway. Secondly, users have to wait during buffering time before viewing a video object. If the buffering time is prolonged because of some reasons such as network congestion, users will probably lose their patience. It is likely that most of multimedia applications with sequential access pattern have similar skewness.

Because video segments have different retrieval frequency, the intra-movie skewness probably causes the nonuniform distribution of workload on the nodes of clustered servers with data striping techniques. That is to say, while data striping reduces the negative impact of the inter-movie skewness, it may introduce new problems caused by the intra-movie skewness if data is not suitably laid out. An analysis in [93] proves that data layout in the traditional round robin way will result in obvious load imbalance among the cluster nodes and hence lead to system performance degradation.

There are two common ways to eliminate the load imbalance caused by the intra-movie skewness. One is to replicate the hottest parts (usually the beginning segments) of video objects and store them on all the nodes. The other is to buffer the hottest parts of video objects. The first method distributes the user retrieval requests for the most popular segments to their replicas, which balances the workload effectively. The second one can be combined with batching techniques. These two methods solve the load imbalance problem caused by the intra-movie skewness and improve the system throughput. However, they may cost lots of system resources. For example, assuming that node number equals to N and average playing time of video objects equals to T minutes, storing the first minute of video objects on all the nodes will result in a redundant ratio of (N-1)/(N+T-1). If there are many nodes and short video objects such as MTV and news in the clustered multimedia server, this redundant ratio may be unacceptable.

3.2 Data Striping and Placement Techniques

In the clustered multimedia server, data placement strategy determines how the storage nodes share the load and how to response to node failures. Especially, in presence of the sequential access pattern that has similar characteristics with the intra-movie skewness, data layout has important effect on system performance. In this part, we analyze the data placement scheme from several aspects. They are fault tolerance, load balancing in both fault free and fault modes, and suitability to different access pattern. Data striping across the nodes provides the potential node-level fault tolerance for the clustered multimedia server. An ideal data placement strategy should provide protection against the data unavailability due to node failures, implement load balancing across the nodes in both fault free and fault modes, and still function gracefully in the event of failures. As mentioned in Section 3.1.3, we only focus on the replica-based schemes.

3.2.1 Mirrored Declustering

Mirrored Declustering (MD), or RAID-10, is a widely used technique for duplication in many applications including multimedia service to provide high data availability. In MD, the data is interleaved across a set of nodes and is mirrored over another set of nodes, as shown in Fig. 3.5, where i' is a backup copy of i. When a node fails, say S[1], the corresponding node S[4] will take over all the load of the failed node while other nodes experience no load increase. For multimedia streaming service, if the load is simply shifted to the node that contains a backup copy during a node failure, the load on the backup node will be doubled. Clearly, it would be better if all surviving nodes experienced a 20 % increase in load. Therefore, while MD offers high level of fault tolerance, it does a poor job of distributing the load of a failure node. With the increase of nodes involved in the clustered multimedia server, the possibility of two failures rendering data unavailable increases because of the imbalance in workloads among the operational nodes in the event of a failure.

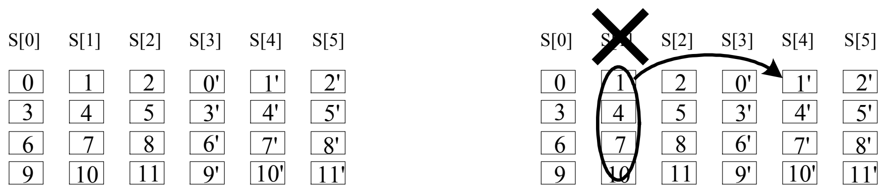


Fig.3.5 Mirrored Declustering

3.2.2 Chained Declustering

Another widely used scheme is called Chained Declustering (CD) [39], as shown in Fig. 3.6. With CD scheme, data will be unavailable only if two logically adjacent nodes fail. Note that S[0]'s copy of Segment 0 is on S[1], S[1]'s copy of Segment 1 is on S[2], and so on. If S[1] fails, S[0] and S[2] share S[1]'s load but other nodes experiences no load increase without dynamic load balancing. Therefore, CD probably still suffer load imbalance because the load of the failed node is completely shifted to its adjacent two nodes, though it is much better than having a single node bear the entire load of the failed node.

By performing dynamic load balancing, CD can obtain better performance. For example, since S[3] and S[5] have copies of some data from S[2] and S[0] respectively, S[2] and S[0] can offload some of their normal load on S[3] and S[5]. Similarly, S[4] has copies of some data from S[3] and S[5], so S[3] and S[5] can also offload some of their normal load on S[4]. In this way the system achieves uniform load balancing. Chaining the data placement allows each node to offload some of its load to either the node immediately following or preceding the given node. By cascading the offloading across multiple nodes, a uniform load can be maintained across all surviving nodes.

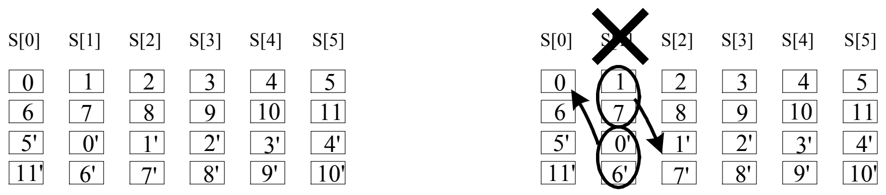


Fig.3.6. Chained Declustering

3.2.3 Interleaved Declustering

Interleaved Declustering (ID), or Segmented Information Dispersal (SID) [24] is a method to achieve load balancing in the fault mode. With ID, a backup copy is subdivided into sub-segments each of which is stored on a different node except the one containing the primary copy, as shown in Fig. 3.7. For example, the backup copy of Segment 1 is striped into 5 sub-segments denoted as 1.0, 1.1, 1.2, 1.3 and 1.4 where i.j represents the j-th sub-segment of the backup copy of Segment i. When a node failure occurs, ID is able to do a better job of balancing the load than MD and CD, since the workload of the failed node will be distributed among operational nodes. Fig. 3.7 shows the sub-segments of Segment 1 share the load of the primary copy when it is unavailable due to node failure. However, ID only can tolerate one node failure and may suffer performance degradation due to small block size of sub-segment when the number of nodes is large. For this scheme, a tradeoff exists between load balancing in face of failure and the data availability of the system.

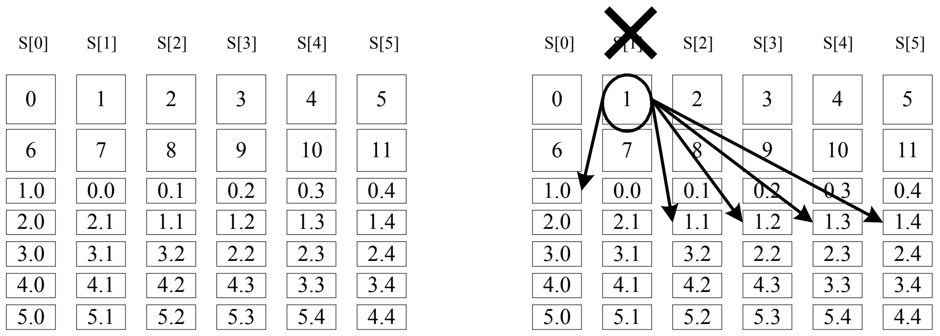


Fig.3.7. Interleaved Declustering

3.2.4 Rotational Mirrored Declustering

Rotational Mirrored Declustering (RMD) [21] combines the merits of CD and MD, so it can remedy some of the drawbacks in both schemes. In RMD, the nodes are divided into some node sets and replicas are stored in different node sets, which is similar to MD scheme. RMD is different from MD in that the replica placements in different node sets are rotated to increase the load balancing performance in the event of node failure. Fig. 3.8 shows the basic idea of the RMD scheme with two node sets. If node failure occurs in one node set, the load of failed node is uniformly shifted to all the nodes in other node sets as shown in Fig. 3.8, which is much better than MD. Moreover, because RMD chains the data placement like CD scheme, with dynamic load balancing, it can distribute the load of the failed node to the rest of surviving nodes.

In the clustered multimedia server with abundant storage capacity, RMD can be used to support high data availability and increase the supportable throughput in video applications.

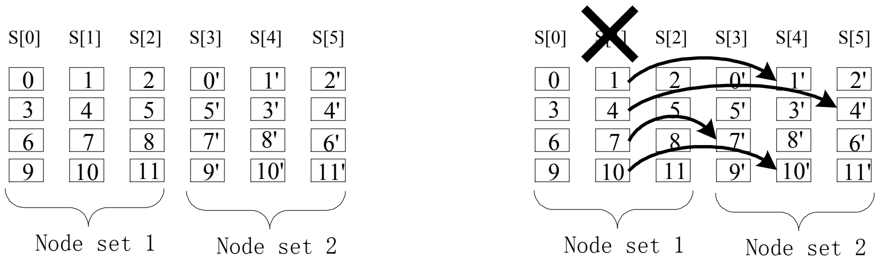


Fig. 3.8. Rotational Mirrored Declustering

3.2.5 Multi-Chained Declustering

Fig. 3.9 illustrates a variation on the CD data placement called multi-chained declustering (MCD) [51]. Instead of having a single chain of stride one, multiple chains of varying strides are used. The system illustrated in Fig. 3.9 uses chains of both strides one and strides two. If dynamic load balancing is not used, this scheme provides better load balancing than CD scheme. For example, if S[1] fails, other nodes each experience a 20 % increase in the load compared to 50 % with CD. In large configurations, MCD has an additional benefit over CD. With CD, multiple failures can prevent uniform load balancing by breaking the chain in more than one place. This effect is most pronounced if the two failures occur close together in the chain. With MCD, there is no problem since the chain of stride two can be used to skip over the failed servers. As expected, however, MCD is less reliable than CD. If, for example, S[1] fails the failure of any other node results in data unavailability. As this example illustrates, in the absence of dynamic load balancing, MCD trades data availability of the system for load balancing in the face of node failure.

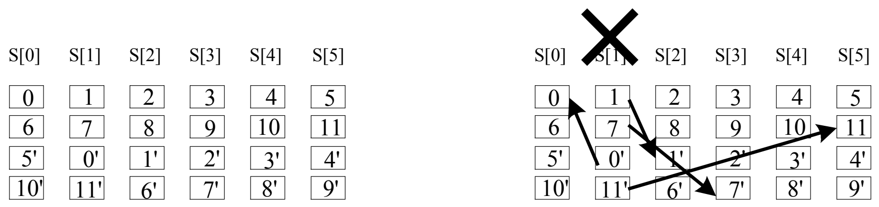


Fig.3.9. Multi-Chained Declustering

3.2.6 Orthogonal Striping and Mirroring

Orthogonal Striping and Mirroring (OSM), or RAID-x [45], is a special data placement for the purpose of improving write performance. The orthogonal placement of data segments and their replicas in OSM is illustrated in Fig. 3.10. The primary copies of data segments are striped across the nodes horizontally and their corresponding backup copies are stored on a single node vertically. For example, Segment 0, 1, 2, 3, 4 are stored on S[0], S[1], S[2], S[3], S[4] respectively, which is called horizontal striping. Their copies are stored on S[5], which is called vertical mirroring. Note that no segment and its copy are placed on the same node. This horizontal striping and vertical mirroring constitute the orthogonal property.

In the event of node failure, OSM distributes the load of the failed node to the other operational ones as shown in Fig. 3.10. However it is not able to tolerate any multiple-node failures. One of the major advantages of OSM is its superior write performance. It can be applied to the write-intensive multimedia applications such as video recording, interactive video editing, etc., which generally generate large numbers of write requests.

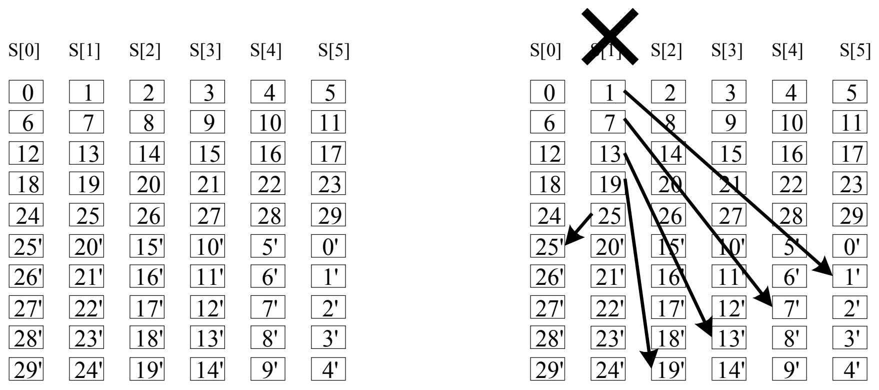


Fig.3.10. Orthogonal Striping and Mirroring

3.2.7 Random Declustering

Random Declustering (RD) store the data blocks randomly, while most of data placement schemes store the segments and their copies in a regular way. There are two kinds of Random Declustering (RD). Semi-Random Declustering (SRD) [84][81] randomly distributes backup copies onto the nodes except the one containing the corresponding primary copies, while primary copies are still striped in a round robin style. Fig. 3.11 is an example of SRD placement. In face of node failure, the load of the failed node can be evenly distributed to all active nodes due to the random assignment of backup copies. Full-Random Declustering (FRD) [76] randomly distributes both primary and backup copies to the nodes, as shown in Fig. 3.12. FRD has better adaptability to different user access patterns and can support more generic workloads than above schemes. However, RD only provide probabilistic guarantee of load balancing in the fault mode and cannot tolerate multiple node failures. And, it has the drawback of maintaining a huge video index of the striping data blocks in practice.

S[0]	S[1]	S[2]	S[3]	S[4]	S[5]
0	1	2	3	4	5
6	7	8	9	10	11
1'	3'	0'	4'	5'	2'
8'	11'	10'	7'	6'	9'

Fig.3.11. Semi-Random Declustering

S[0]	S[1]	S[2]	S[3]	S[4]	S[5]
3	2	1	0	4	5
6	7	11	9	8	10
1'	3'	0'	4'	5'	2'
8'	11'	10'	7'	6'	9'

Fig.3.12. Full-Random Declustering

3.2.8 Symmetrical Declustering

Clustered multimedia servers provide many services with sequential access pattern such as VoD. As discussed in Section 3.1.4.2, there is probably skew reference pattern (the intra-movie skewness) in the sequential access application. A uniform division of the data does not correspond to a uniform division of the load because of this access skewness. Therefore, the load is likely not uniformly divided among the nodes if data placement scheme is not suitable. However, most of the data placement strategies, including MD, CD, ID, RMD, MCD, OSM and SRD, are primarily employed to support random access applications and not suitable for sequential access applications. The intra-movie skewness may cause load imbalance and performance degradation in the clustered multimedia server with these schemes. Because of the random assignment of data segment, FRD is not sensitive to access patterns and supports more generic workloads. But, due to its poor fault tolerance ability and huge data index, FRD is less desirable for clustered multimedia server with high availability.

Symmetrical Declustering (SD) is a data placement style suitable for both random and sequential access pattern. The basic idea of SD is to organize primary and backup copy of data segments in a symmetrical way, which can balance the workload of the nodes in the presence of the intra-movie skewness. There are three kinds of SD placement. Mirrored Symmetrical Declustering (MSD) and Shifted Symmetrical Declustering (SSD) respectively emphasize high availability and load balancing in the event of a failure. Chained Symmetrical Declustering (CSD) is a compromise between them.

Fig. 3.13 illustrates the data placement layout of MSD that has high level of availability.

MSD has the similar characteristics as MD except that this method has good load balancing in the normal mode for both sequential and random access pattern. It is suitable for the clustered multimedia server demanding high availability.

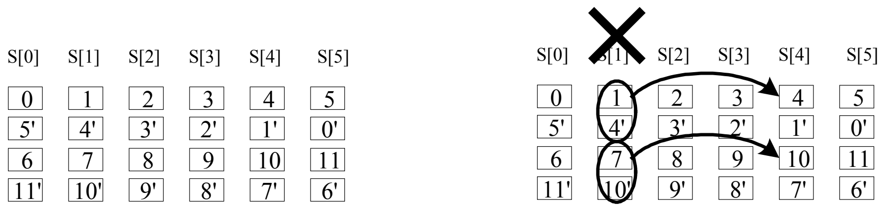


Fig.3.13. Mirrored Symmetrical Declustering

SSD is shown in Fig. 3.14. In order to obtain high performance in the fault mode, SSD scheme trades availability for load balancing. With SSD, the load of the failure node can be distributed over other active nodes without dynamic load balancing. As illustrated in Fig. 3.14, if S[1] fails, its load can be reassigned to other five nodes.

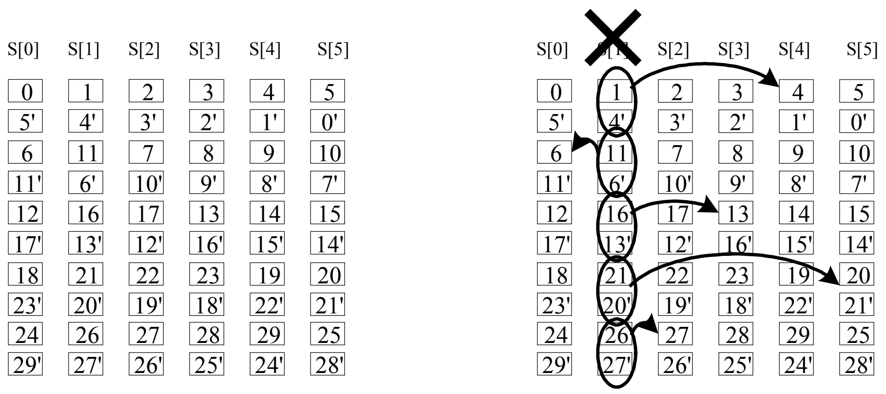


Fig.3.14. Shifted Symmetrical Declustering

Fig. 3.15 is the data placement of CSD scheme. It has the similar characteristics as CD except that this scheme is suitable for not only random access pattern but also sequential access pattern. The figure shows load distribution of CSD in the event of node failure without dynamic load balancing. If S[1] fails, its load are distributed to it two logically adjacent nodes, S[0] and S[4]. With dynamic load balancing, the load of the failed node can be uniformly distributed to other operational nodes.

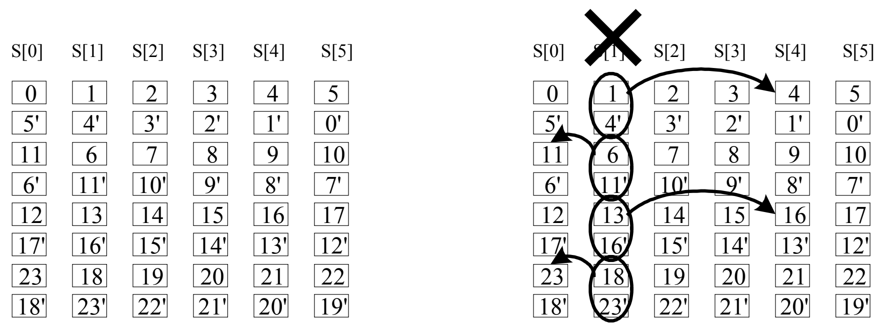


Fig.3.15. Chained Symmetrical Declustering

3.3 Summary

	suitability		fault tolerance		load redistribution in fault mode		
	for random access pattern	for sequential access pattern	tolerate single node failure	tolerate multiple node failure	to single active node	to some active nodes	to all active nodes
MD	√		√		√		
CD	√			√		√	
CD*	√			√			√
ID	√		√				√
RMD	√			√		√	
RMD*	√			√			√
MCD	√		√				√
OSM	√		√				√
SRD	√		√				√
FRD	√	√	√				√
MSD	√	√		√	√		
SSD	√	√	√				√
CSD	√	√		√		√	
CSD*	√	√		√			√

√ means that a scheme has the corresponding characteristic shown at the top of the table.
* means a scheme with dynamic load balancing in the fault mode.

4. Real-time Stream Scheduling

The Overall goal of stream scheduling in multimedia systems is to meet the deadlines of all time-critical tasks. Closely related is the goal of keeping the necessary buffer space requirements low. At the same time, the server should ensure that aperiodic requests be scheduled without delaying an infinite amount of time. The scheduling algorithm must find a balance between time constraints and efficiency.

4.1 Real-time Disk Scheduling

Traditionally, disk scheduling algorithms (e.g., first come first served (FCFS), shortest seek time first (SSTF), SCAN, etc.) have been employed by servers to reduce the seek time and rotational latency, to achieve a high throughput, and to provide fair access to each client. The addition of real-time constraints, however, makes direct application of traditional disk scheduling algorithms inappropriate for multimedia servers.

Techniques for scheduling real-time tasks have also been extensively studied in the literature. The best known algorithm for real-time scheduling of tasks with deadlines is the Earliest Deadline First (EDF) [58] algorithm. In this algorithm, after accessing a media block from disk, the media block with the earliest deadline is scheduled for retrieval. Scheduling of the disk head based solely on the EDF policy, however, may yield excessive seek time and rotational latency, and hence, may lead to poor utilization of the server resources.

In the context of multimedia applications, SCAN-EDF [65], a hybrid of SCAN and EDF algorithm, is proposed to address the real time and efficiency requirements. In this algorithm, the requests are served in EDF order, but when several requests have the same deadline, they are served using SCAN. Since the optimization only applies for requests with the same deadline, the following technique [79] can be used to improve the efficiency: all requests have release times that are multiples of the period p, so that all the deadlines are also multiples of the period p. Therefore, the requests can be grouped together and be served accordingly.

4.2 Real-time Data Scheduling

The disk scheduling algorithms generally deal with the optimization of disk heads movement among the disk cylinders for a given set of requests with deadlines. But these algorithms themselves don’t specify which data needs to be retrieved by when, nor the order in which units of data are read into the memory. These tasks are accomplished by the data scheduling algorithms, which determine how the data for a continuous stream is driven off the disk and apart from the memory. It is responsible for generating requests deadlines and submits them to the disk driver, guaranteeing there will be no consumer starvation or buffer overflow. In an environment of multiple streams, it should also strive to maximize the resource utilization to support as more streams as possible. In the scheduling, the tradeoff between different types of resource like disk bandwidth, memory should always be considered.

There are two classes of data scheduling policies: *round-based scheduling* [7][36] and *deadline-based scheduling* [75][94].

4.2.1 Round-based Data Scheduling

In round-based scheduling, time is divided into round of length p. While users are consuming data over the network during round i, the storage server is inserting data into memory with the data needed for round i +1. During each round, all media streams are serviced.

The round length represents an upper bound on the time in which the storage server must retrieve from disk the next figments for all active continuous displays, or some displays will suffer a glitch. Within a round, it is possible to employ either a round-robin or a SCAN algorithm. The latter performs seek optimization, resulting in better disk throughput. However, this is achieved at the expense of higher start-up latency; the display cannot be started immediately after the retrieval of its block but only after the end of the round. This is done to avoid glitches since the service order differs from round to round. This situation is not present when using round-robin scheduling which also has lower RAM buffer requirements since it does not require the double-buffering scheme required by SCAN between successive rounds. A compromise was achieved with the Group Sweeping Scheduling (GSS) algorithm [20]. GSS groups streams and employs round-robin scheduling for the different groups and SCAN scheduling for the streams' block in a group. Thus, when there is only one group GSS reduces to SCAN and when each stream is in its own group GSS degenerates to round-robin.

The round length is a crucial parameter in the data scheduling. To service all clients without a hiccup, sufficient data must be put in the buffer to be sent in any period. As the service period becomes longer, each stream must read more data from the disk into the buffer. However, due to the limited size of the buffer, the period can't be longer than an upper bound, say, T_{max}. On the other hand, reading all the necessary data in a short period requires relative high disk bandwidth. Due to the limit of disk bandwidth, the period cannot be shorter than some minimum value T_{min}. The upper/lower bound can be calculated according to the allowable overflow/starvation probability which is given as the QoS parameter. A lot of researches address this problem. For example, Chen et al. [19] proposed a scheme to find lower and upper bound of the period. [67] studied the smoothing effect on the data retrieval brought out by varying the round length. [65] studied the effect of extended deadline, which is essentially equal to the increased round length regarding the buffer requirement, on the disk performance.

4.2.1 Deadline-driven Data Scheduling

The deadline-driven data scheduling policy is usually combined with a random data placement strategy. Compared with the round-based approach which retrieves blocks in advance by employing optimized disk scheduling, the deadline-driven scheduling allows fewer optimizations to be applied, potentially resulting in more wasted bandwidth and less throughput. However, the startup latency is generally shorter compared to round-based scheduling, because with round-based scheduling the initial startup latency for an object might be large under heavy load. This makes it more suitable for interactive applications.

Sahu Sambit et al. [67] examined the round-based scheduling combined with a CTL data access policy (termed CTL-Round) and the deadline-based scheduling combined with a CDL data access policy (termed CDL-EDF) in the context of VBR media streaming. From the simulation, it is observed that CDL-EDF is able support the retrieval of the same number of streams with lower resource (disk rate, buffer size) requirements. The reason is attributed to the data rate variability within and across the CTL rounds. Although smoothing is achieved within a CTL round, round-based retrieval does not help reduce the variability that is present across the CTL rounds. In contrast, CDL-EDF reduces the inter-round variability by workahead, i.e., prefetching data ahead in time. Based on this observation, the authors further developed an extension to CTL-Round policy that integrates the advantage of workahead of a deadline-based retrieval policy into a periodic data access policy.

4.3 Disk Scheduling for Mixed-Media Workloads

A digital library application may consist of both continuous media and traditional data (termed non-real-time) such as text, still images and ASCII text. Non-real-time object requests do not have deadlines and are given a lower priority than video and audio objects. However, these requests must be served in a manner that prevents starvation.

Not all real-time object requests have identical priorities. Some are more important than others. For example, with MPEG encoded video, a video clip is represented as a sequence of I, P, and B frames. Losing an I frame is more disruptive than losing either P or B frames. A disk scheduling algorithm that gives a higher priority to the retrieval of I frames results in a better Quality of Service (QoS). As another example, it is well known that humans are more sensitive to faults in audio than in video. If it is inevitable to discard data packets, it might be better to discard video packets in favor of delivering audio packets.

As a final example from video on demand applications, the service providers may offer a range of services with higher quality service provided at higher prices. For example, there might be two levels of services: high-quality and best-effort. Customers ordering high-quality services are expected to be charged more than those ordering best-effort streams. One way to honor the high-quality services is to assign different priorities to object requests and implement scheduling techniques that service them in a manner that maximizes profits.

In all examples, scheduling of real time requests alone or real time requests with the same priority will not satisfy the application requirements. The desired disk scheduling algorithms should meet these goals:

- The requests with the highest priority should be scheduled without deadline violations.
- The non-real-time requests should have small average response times and small response time variance.
- If it is inevitable to discard some of the data requests at peak system loads, it desirable to discard those with lowest priority.

The work in 1.1.1[65] studied the scheduling of aperiodic requests mixed in real-time stream requests with some classic disk scheduling algorithms, including EDF, CSCAN, and SCAN-EDF. The affect of aperiodic requests on the real-time service capacity and the response time of the aperiodic requests are analyzed through simulation.

[66] overviewed the performance goals for mixed workload scheduling in the context of continuous media stream and discrete requests. The disk scheduling algorithms for mixed-media workloads are classified by: (1) *Number of separate scheduling phases per round*. One-phase algorithms produce mixed schedules, containing both discrete and continuous data requests. Two-phase algorithms have two, not timely overlapping, scheduling phases serving discrete and continuous data requests isolated in the corresponding phases. (2) *Number of scheduling levels*. Hierarchical scheduling algorithms for discrete data requests are

based on defining *clusters*. The higher levels of the algorithms are concerned with the efficient scheduling of clusters of discrete requests. The lower levels are efficiently scheduling the requests within a cluster. The most important task to solve in this context is how to schedule discrete data requests within the rounds of continuous data requests, which are mostly served by SCAN variations.

Cello [72] uses a two level disk scheduling architecture. It combines a class independent scheduler with a set of class-specific schedulers. Two time scales are considered in the two levels of the framework to allocate disk bandwidth: (1) coarse-grain allocation of bandwidth to application classes is performed by the class-independent scheduler; and (2) the fine-grain interleaving of requests is managed by the class-specific schedulers. This separation enables the co-existence of multiple disk scheduling mechanisms at a time depending on the application requirements. Cello defines 3 service classes: (1) interactive best-effort applications; (2) throughput-intensive applications and (3) real-time applications with periodic consumption. The cello mechanism has been implemented in the project Qlinux [40].

The similar service classes were adopted in Prism [90]. It realized an resource allocation and scheduling mode that differentiates among 3 types of streams: periodic streams, aperiodic streams and interactive streams, as shown in Fig. 4.1. The disk I/O scheduler schedules I/O requests in two time frames: namely main period and sub period. There are several sub periods per main period. In every main period the list of periodic requests and available aperiodic requests are combined in SCAN order. But in the sub periods this order can be adjusted to consider the interactive requests, which are maintained in FIFO queue. This mechanism allows for faster response times for interactive requests at the cost of increased seek times. An interactive request is scheduled only if there is sufficient slack to schedule periodic requests for the current main period, and an aperiodic request is scheduled only if there is sufficient slack time for both periodic and interactive requests. Service class specific admission controllers limit the resource consumption of specific class to assigned levels.

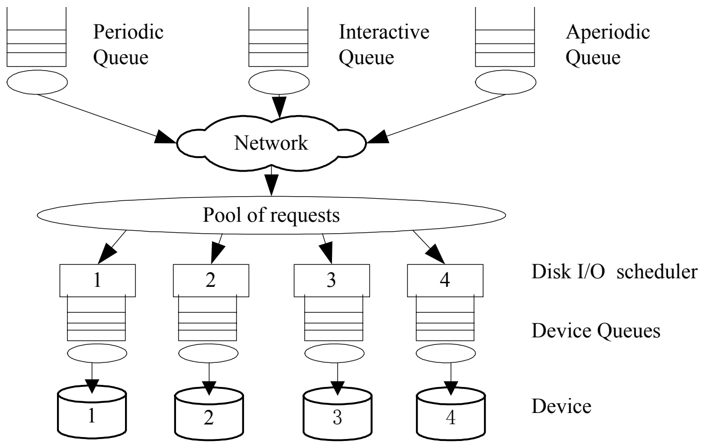


Fig. 4.1 Prism resource allocation and scheduling framework

The quick response requirement of interactive requests in multimedia service also motivated BubbleUp [17], a disk scheduling algorithm proposed to minimize initial latency for interactive requests. With this technique, all periodic requests are served in SCAN order, however, when the first request of a new stream arrives, it is inserted at the head of the queue for immediate service.

[46] proposed a deadline driven disk scheduling algorithm in support of real time requests with multiple priorities, e.g., those for different object classes in digital library applications. Instead of maintaining separate queue for each priority level as in traditional schemes, it maintains one queue for all requests to enhance utilization of disk bandwidth. By this way the disk optimization can be performed globally. The simulation experiments exhibits significant increase in the number of serviced low priority requests with the proposed scheme when compared with the multi-queue scheme. But under some conditions the proposed algorithm might violate the deadline of a few high priority requests.

While a lot of literatures have concentrated on data retrievals in support of a continuous display, scant attention has been paid to the write scheduling. In [3], a method is proposed to address this problem. It tries to minimize the occurrences where the writing requests violate the deadlines of a few block reads during peak system load. The scheduler achieves it in two ways. First, it schedules the reading and writing of blocks to maximize the utilization of disk bandwidth (by minimizing the impact of disk seek and latency using algorithms such as SCAN-EDF). Second, it delays the writing of blocks when the amount of available buffers is abundant. This is achieved by developing a methodology for associating an artificial deadline with each write request to the disk. This deadline is a function of the arrival rate of block write requests and the amount of available buffers. Simulation studies demonstrates that the proposed scheme is superior to an alternative that maintains different queues for the read and write requests.

5. Admission Control

Given the real-time performance requirements of each client, a multimedia server must employ admission control algorithms to determine whether a new client can be admitted without violating the performance requirements of the already admitted clients. In the simplest case, the scheduler can make decision based on the worst assumptions regarding various resource requirements of the new request. This scheme can give deterministic Qos guarantee to the new client. However, it is overly conservative and may result in very poor resource utilization because the resource consumption may be much less than the worst case in normal state. A lot of researchers have exploited the variability of different resource consumption to improve the number of concurrently supported clients with available resources.

The variable factors in a multimedia service are manifold. For example, the seek time and rotational latency of disks, the encoded bit rate of the media file, the congestion condition of communication network, etc., all brings some unpredictable affects upon the servicing. Some researchers proposed schemes to predict the possibility of accommodating a new client by observing the system behavior in a past time window; while other researchers employ statistics models to characterize the behavior of disks and bit rate of existing media files, so as to conduct probabilistic analysis regarding the admission of the new client.

Typically, admission control policies are divided into three classes:

- *Deterministic policy* guarantees specified QoS requirements of existing clients and admits a new client only if its service demand does not affect the present clients.
- *Statistical-based policy* handles admission control based on the probabilistic distributions of various factors, such as the bit rate and frame size of stored media files, the file block access time, etc. The admission control results are given in a probabilistic form.
- *Measurement-based policy* monitors the resource utilization over a time window, and estimate the new client's resource requirement based on this observation. The observed utilization will determine if the server can meet the requirements of the new client.

The deterministic policy ensures an uninterruptible service for a client once it is admitted, while a statistical-based policy and measurement-based policy only provide probabilistic guarantee for the requests' real-time requirement. The latter two policies are justifiable because of the fact that some applications may be able to tolerate some missed deadlines. For example, a few lost video frames, or the occasional pop in the audio may be tolerable in some cases - especially if such tolerance is rewarded with a reduced cost of service.

In what follows, we will discuss the admission control algorithms focusing on the disk aspect. Since the CBR (Const Bit Rate) stream can be regarded as a special example of VBR (Variable Bit Rate) streams, we restrict our discussion in the VBR admission control, and for simplicity, we will refer to the disk admission control algorithms for VBR streams as admission control algorithms.

5.1 Deterministic Policy

The simplest method to provide deterministic QoS for clients is to employ the maximum data rate of a stream and the worst-case data rate of the disk as the decision parameters [35][64]. With this scheme, the admission decision can be performed very simply. However, this is an extremely conservative scheme that significantly under-utilizes the disk resource. Often it is used as baseline for the system performance optimization.

An intuitive improvement for the above algorithms is to use a vector that records the maximum data rates in successive rounds, rather than a single maximum value of the whole stream, to describe the requirement of the stream [59][18]. The server maintains a scheduling table recording the amounts of data to be retrieved in each time round, and when a steam request arrives, it adds the data rate vector to the scheduling table, checking if there exists any round in which the aggregate data rate exceeds the lower bound of disk rate (termed *minRead*). If so, the request is rejected, otherwise the stream is admitted. Though superior to the first algorithm, this algorithm is also overly conservative. One major advantage of both algorithms is that neither of them require any additional buffer space for read ahead at the server, since all the disk requirements for every round can be met by the disk reads performed during that round.

The deterministic policy does not necessarily leads to poor disk utilization. VbrSim [60] solves the problem of being too conservative in admissions, but still gives deterministic guarantees to the clients. This algorithm builds on the second algorithm mentioned above by making better use of the scheduling table. It enforces the server read *minRead* blocks in each round, hoping that the bandwidth peaks can be smoothed by the data read ahead in rounds with lower data rate. This rate of reading, of course, is only possible when there are enough buffers to hold the data read early. To cope with the possible buffer overflow, the server also maintains a buffer allocation table to manage the buffer utilization. The buffer space storing data of a round will be reclaimed at the start of next round, and in some cases the allocated buffer needed furthest in the future can be “stolen” for current buffer allocation.

5.2 Statistical-based Policy

In Vin et al.[87], a statistical admission control algorithm is presented, which considers not only average bit rates, but the distributions of frame sizes, and probability distributions of the number of disk blocks needed during any particular service round. In rounds (referred to as overflow rounds) the disk is over subscribed (with a certain probability which can be controlled by the algorithm), the system attempts to judiciously distribute the effective frame loss among the subscribed clients. A greedy disk algorithm attempts to reduce the actual occurrence of overflow rounds. This algorithm requires some knowledge of the syntax of the data stream, such as where display unit (i.e. video frame) boundaries exist and which display units are more important than others (i.e. MPEG I Frames vs. MPEG B Frames).

When caching is employed in multimedia servers, the disk workload may exhibit different characteristics. Kang et al. [47] studied this problem on the basis of *interval caching* [26]. They take into account the effect of caching by expressing it as load reduction imposed on the disk. Hence, the ultimate disk load can be calculated through subtracting the load reduction by caching from the disk load without caching. The calculating also adopts a probability-based method and the service guarantee is given in a probabilistic fashion.

The RSAC (Reliable Statistical Admission Control) [48] method also exploits cache when performing admission control. It attempts to optimize the server utilization and minimize jitter by reserving a certain amount of disk bandwidth for those streams which are evicted from the cache, and are likely to cause jitter. The amount of reserved bandwidth is estimated as a function of the average number of cached streams, which in turn indicates improvement in server capacity with interval caching.

5.3 Measurement-based Policy

Measurement-based policy is based on the assumption that the amount of time spent in servicing each of the already admitted clients will continue to exhibit the same behavior, even after a new client is added into the system. A multimedia server that employs such an observation-based approach is referred to as providing predictive service guarantees to clients.

In Vin's work [88], such an algorithm is presented. It uses the predicted extrapolation from the status quo measurements of the disk utilization to decide if the new client can be added without violating the service requirements of all existing clients. A greedy disk scheduling algorithm is used in conjunction with it to achieve high disk utilization. The technique for minimizing and distributing the discarded media blocks are also presented to improve the QoS for playback.

The new client's requirement can be estimated using either a Worst-Case or Average-Case extrapolation [86]. The Worst-Case algorithm uses the maximum service time in the recent past to estimate the new client's requirement, while the Average-Case algorithm estimates the new requirement based on the average service time of existing streams. The experiment results show minor different between these two algorithms regarding the maximum supported streams, but the Worst-Case algorithm is more reliable in meeting the real-time requirements of requests.

6. Conclusions

This paper has sought to bring together the ideas and work in the area of clustered multimedia servers generated in the past decade. While the involved researches issues are many, we focus on two aspects, namely architecture and storage system, which plays central roles in a multimedia server and have received a great deal of attention from the academia. For the architecture, we survey over 20 projects/systems and give taxonomy for their system structures. Then we examine the media data organization schemes and the stream scheduling issues, including the techniques of real-time stream scheduling and admission control algorithms. The intention of this paper has been to present a suitable framework for comparing past work on these topics, and, if possible, provide some guidance for the design and strategy choosing in building a clustered multimedia server.

As in the case in any survey, there are many pieces of work to be considered. It is hoped that the examples presented fairly represent the main efforts made on the practice of architecture and storage system design of clustered multimedia servers. Most of these systems are from the laboratories since few published materials about the detailed design of commercial products can be found. The exclusion of any particular results has not been intentional. Decisions as to which papers to use as examples were made purely on the basis of their applicability to the context of the discussion in which they appear.

References

[1] C. Akinlar and S. Mukherjee. "A Scalable Distributed Multimedia File System Using Network Attached Autonomous Disks". *Proc. of 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pp.

180-187. 2000.

[2] C. Akinlar and S. Mukherjee. “Bandwidth Guarantee in a Distributed Multimedia File System Using Network Attached Autonomous Disks”. *Proceedings of Sixth IEEE Real-Time Technology and Application 2000 Symposium*. RTAS 2000. Page(s): 237 –246

[3] W. G. Aref, I. Kamel and S. Ghandeharizadeh, “Disk Scheduling in Video Editing Systems,” In *IEEE Transactions on Knowledge and Data Engineering*, Volume 13, Number 6, 933-950, December 2001

[4] K. Argy, “Scalable multimedia servers,” *IEEE Concurrency*, Volume 6，Issue 4 Oct.-Dec. 1998. Page(s) 8 –10

[5] R. L. Axtell, “Zipf Distribution of U.S. Firm Sizes,” *Science*. Sept. 7，2001，Vol. 293. pp. 1818-1820

[6] Christoph Bernhardt and Ernst Biersack, “A Scalable Video Server: Architecture, Design, and Implementation,” *Real-time Systems Conference*, Paris, France, January 1995, pp.63-72

[7] S. Berson, S. Ghandeharizadeh, R.R. Muntz and X.Ju, “Staggered Striping in Multimedia Information Systems”, *Proc. of the International Conference on Management of Data (SIGMOD)*, Minneapolis, Minnesota, pp.79-90, 1994.

[8] Miklos Berzsenyi，Istvan Vajk，Hui Zhang. “Design and implementation of a video-on-demand system,” *Computer Networks and ISDN Systems* 30 (1998) 1467-1473

[9] C. C. Bisdikian and B. V. Patel, “Issues on Movie Allocation in Distributed Video-on-Demand Systems,” *Proc. ICC95*, IEEE Press, Piscataway, N.J., 1995, pp.250-255.

[10] William J. Bolosky, Robert P. Fitzgerald, and John R. Douceur. “Distributed Schedule Management in the Tiger Video Fileserver,” In *Proceedings of SOSP-16*, St.-Malo, France, Oct. 1997. Pages 212-223

[11] Alice Bonhomme and Loic Prylli, “A Distributed Storage System for a Video-on-Demand Server,” *Lecture Notes in Computer Science*, vol.1900, pp.1100-1110, 2001

[12] Klaus Breidler, “Communication Infrastructure for the Parallel Video Server SESAME,” Master Thesis, Institute of Information Technology at University Klagenfurt, Austria. February 2000.

[13] T. Brisco，“DNS Support for Load Balancing,” RFC 1794，[Online] http://www.ietf.org/rfc/rfc1794.html.

[14] Milind Buddhikot and Gurudatta, Parulkar, “Efficient Data Layout, Scheduling and Playout Control in MARS,” in *Proceedings of the 5th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, April 1995

[15] A. Calvagna，A. Puliafito and L. Vita. “Design and implementation of a low-cost/

high-performance Video on Demand server," *Microprocessors and Microsystems*. 24(2000) 299-305

[16] C. K. Chang，C. C. Shih，T. T. Nguyen and P. Mongkolwat，"A Popularity-based Data Allocation Scheme for a Cluster-based VOD Server," *Proc. of COMPSAC'96*，Seoul，Korea，August 1996， pp. 62-67

[17] E. Chang and H. Carcia-Molina, "BubbleUp: Low Latency Fast-Scan for Media Servers," *Proc. of the 5th ACM Conference on Multimedia*, June 1997

[18] E. Chang and A. Zakhor, "Cost Analyses for VBR Video Servers," in *IST/SPIE Multimedia Computing and Networking*, (San Jose, CA), pp. 381--397, Jan. 1996

[19] Huagn-Jen Chen and T.D.C. Little, "Storage allocation policies for time-dependent multimedia data," *IEEE Transaction On Knowledge and Data Engineering*. 8(5):855-864, October, 1996.

[20] M. S. Chen, D.D. Kandlur, and P.S. Yu, "Optimization of the Grouped Sweeping Scheduling (GSS) with Heterogenous Multimedia Streams", *Proc. ACM Multimedia 93*, Anaheim, CA, Aug. 1993, pp. 235—242

[21] M. S. Chen et al., "Using Rotational Mirrored Declustering for Replica Placement in a Disk-Array-Based Video Server," *Multimedia System*, Vol. 5, pp. 371-379, December 1997.

[22] Shenze Chen and Manu Thapar, "A Fibre Channel-based Architecture for Internet Multimedia server Clusters". Proc. of *3rd International Conference on Algorithms and Architectures for Parallel Processing*, 1997. pp. 437-450

[23] Cisco Local Director，Cisco Systems，Inc.，http://www.cisco.com/univercd/cc/td /doc/pcat/ld.htm

[24] Ariel Cohen and Walter A. Burkhard，"Segmented Information Dispersal (SID) Data Layouts for Digital Video Servers," *IEEE Transactions on knowledge and data engineering*，Vol. 13， No.4，August 2001. pp. 593-606

[25] Damani et al. "ONE-IP: Techniques for Hosting a Service on a Cluster of Machines", in Proc. of *Sixth International WWW Conference*, April 1997.

[26] A. Dan and D. Sitaram, "Buffer management policy for an on-demand video server," IBM Research Report RC 19347.

[27] A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling Policies for an On-Demand Video Server with Batching," in *Proc. of ACM Multimedia 94 Conference*, pp. 15-21, October, 1994.

[28] Daniel M. Dias, William Kish, Rajat Mukherjee, and Renu Tewari, "A Scalable and Highly Available Web Server," in *Proceedings of IEEE COMPCON'96*.

[29] D. H. C. Du and Y. J. Lee, "Scalable server and storage architectures for video streaming," *Proc. IEEE Int. Conf. Multimedia Computing and Systems*, pp. 62–67, June 1999.

[30] C. S. Freedman and D. J. DeWitt, "The SPIFFI scalable video-on-demand system," in

Proc. ACM SIGMOD'95, San Jose, CA, May 1995, pp. 352-363

[31] J. Friedrich，T. Grun，and J. Keller, “Video-on-Demand on the SB-PRAM,” In *Proc. of the 6th International Workshop on Network and Operating System Support for Digital Audio and Video*，NOSSDAV’96，Zushi，Japan. April 1996，pages 105-111

[32] J. Gafsi， E. Biersack， “Data striping and reliability aspects in distributed video servers,” *Cluster Computing* 2(1), 1999. pp. 75-91

[33] J. Gafsi and E. Biersack, “A Novel Replication Placement Strategy for Video Servers,” *6th International Workshop on Interactive and Distributed Multimedia Systems*, Toulouse, France, October 12-15 1999.

[34] J. Gafsi, Ulrich Walther, Ernst W.Biersack. “Design and Implementation of a Scalable, Reliable, and Distributed VOD-Server”. In Proceedings of *the 5th joint IFIP and ICCC Conference on Computer Communications*, AFRICOM-CCDC, Tunis, 1998

[35] J. Gemmell and Stavros Christodoulakis, “Principles of Delay-Sensitive Multimedia Storage and Retrieval,” *ACM Transactions on Information Systems*, 10(1): 51-90, 1992

[36] J. Gemmell, H.M. Vin, D.D. Kandlur, P.V. Rangan, and L.A. Rowe, “Multimedia Storage Servers: A Tutorial,” *IEEE Computer*, Vol.28, no.5, May 1995, pp.40-49

[37] G. A. Gibson，D. F. Nagle，K. Amiri，and et al. “File Server Scaling With network-attached secure disks”. *Proceedings of the ACM International conference on Measurement and Modeling of Computer Systems (Sigmetrics’97)*，Seattle, WA. June 1997. pages 272-284

[38] G. A. Gibson and R. V. Meter, “Network attached storage architecture,” *Commun. ACM*, vol. 43, no. 11, pp. 37–45, Nov. 2000.

[39] L. Golubchik，J. C. Lui，and R. R. Muntz，“Chained Declustering: Load balancing and robustness to skew and failures，” in *Proceedings of the Second International Workshop on Research Issues in Data Engineering: Transaction and Query Processing*，February 1992. pp. 88-95

[40] P. Goyal, J. K. Sahni, P. Shenoy, R. Srinivasan, H. Vin, and T.R.Vishwanath. QLinux 2.4.x: A QoS enhanced Linux Kernel for Multimedia Computing. Qlinux home page, 2002. http://lass.cs.umass.edu/software/qlinux/

[41] C. Griwodz, M. Bar, and L. C. Wolf, “Long-term Movie Popularity Models in Video-on-Demand Systems or the Life of an On-Demand Movie,” *Proc. Multimedia 97*, ACM Press, New York, 1997, pp.349-357.

[42] A. Guha, “The evolution to network storage architectures for multimedia applications,” Proc. *IEEE Int. Conf. Multimedia Computing and Systems*, pp. 68–73, June 1999.

[43] Roger L. Haskin and Frank B. Schmuck. “The Tiger Shark File System,” *Proc. of COMPCON’96.* 1996. Page(s): 226-231

[44] A. Heybey, M. Sullivan, and P. England, “Calliope: A distributed, scalable multimedia server,” *Proc. of USENIX 1996 Annual Technical Conference*, 1996.

[45] Kai Hwang, Hai Jin, R. Ho, “Orthogonal Striping and Mirroring in Distributed RAID for

I/O-Centric Cluster Computing," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 13, NO. 1, pp. 26-44, January 2002.

[46] I. Kamel, T. Niranjan, and S. Ghandeharizadeh, "A Novel Deadline Driven Disk Scheduling Algorithm for Multi-Priority Multimedia Object," In *Proceedings of the IEEE Data Engineering Conference*, February 2000

[47] Sooyong Kang and Heon Y. Yeom, "Statistical admission control for soft real-time VOD servers," *Proc. of ACM Symposium on Applied Computing 2000*, pp 579-584

[48] S. Kim and C. R. Das, "A Reliable Statistical Admission Control Strategy for Interactive Video-On-Demand Servers with Interval Caching," *Proceedings of the 2000 International Conference on Parallel Processing (ICPP)*, pp. 135-142

[49] Argy Krikelis, "Scalable multimedia servers", IEEE Concurrency, Volume 6, Issue 4, pp. 8-10, Dec. 1998.

[50] J. B. Kwon, H. Y. Yeom. "An Admission Control Scheme for Continuous Media Servers using Caching," in *Proceeding of IEEE International Performance, Computing,and Communications Conference, 2000*. IPCCC'00. pp. 456-462

[51] E. K. Lee, "Highly-Available, Scalable Network Storage," in *Proc. of CompCon*, March 1995.

[52] J. Y. B. Lee, "Parallel video servers: a tutorial," *IEEE Multimedia*, Volume 5 Issue 2， April-June 1998，pp. 20 –28

[53] J. Y. B. Lee. "Concurrent Push – A Scheduling Algorithm for Push-Based parallel Video Servers," *IEEE Transactions on Circuits and Systems for Video Technology*. VOL. 9， No. 3， April 1999. pp. 467-477

[54] J. Y. B. Lee. "Supporting Server-Level Fault Tolerance in Concurrent-Push-Based Parallel Video Servers". *IEEE Transactions on Circuits and Systems for Video Technology*. VOL.11，No.1， January 2001. pp. 25-39

[55] Peter W. K. Lie, John C. S. Lui, Leana Golubchik, "Threshold-Based Dynamic Replication in Large-Scale Video-on-Demand Systems," *Multimedia Tools and Applications* 11(1): 35-62 (2000)

[56] Linux Virtual Server Project，http://www.LinuxVirtualServer.org/

[57] T. Little and D. Venkatesh, "Popularity-based Assignment of Movies to Storage Devices in a Video-on-Demand System," *IEEE Multimedia*, vol.2, pp.280-287, Spring 1995.

[58] C. L. Liu and J.W.Layland, "Scheduling algorithms for multiprogramming in a hard real time environment," *Journal of ACM*, Jan 1973, vol.20, no.1, pp.46-61

[59] D. Makaroff, G. Neufeld and N. Hutchinson, "An Evaluation of VBR Admission Algorithms for Continuous Media File Servers," *ACM Multimedia'97*, Seattle, WA, November, 1997, pp 143-154

[60] Dwight J. Makaroff, Gerald W. Neufeld, Norman C. Hutchinson, "Design and Implementation of a VBR Continuous Media File Server," *IEEE Transactions on Software Engineering* 27(1): 13-28 (2001)

[61] Jeffrey Mogul. “Network behavior of a busy Web server and its clients”. Research Report 95/5, DEC Western Research Laboratory, October 1995.

[62] Miyazaki, Klara Nahrstedt, “Dynamic Coordination of Movies According to Popularity Index and Resource Availability within a Hierarchical VoD System,” in *Proc. of IEEE Region 10 Annual Conference, Speech and Image Technologies for Computing and Telecommunications*, pp. 199-203, Queensland, Australia, December, 1997

[63] Chang-Soon Park, Mann-Ho Lee, Young-Sung Son and Oh-Young Kwon, “Design and implementation of VoD server by using clustered file system,” *IEEE International Conference on Multimedia and Expo 2000*, Volume 3. Page(s): 1465 –1468

[64] P.V. Rangan and H. M. Vin, “Efficient Storage Techniques for Digital Continuous Multimedia,” *IEEE Transactions on Knowledge and Data Engineering Special Issue on Multimedia Information Systems*. August 1993

[65] A.L.N. Reddy and J.C. Wyllie, “I/O Issues in a Multimedia System,” *IEEE Computer*, March 1994, pp.17-28

[66] Y. Rompogiannakis, G. Nerjes, P. Muth, M. Paterakis, P. Triantafillou, G. Weikum, “Disk scheduling for mixed-media workloads in a multimedia server,” *Proceedings of the sixth ACM international conference on Multimedia*, p.297-302, September 13-16, 1998.

[67] Samit Sahu, Zhi-Li Zhang, Jim Kurose, and Don Towsley, “On the Efficient Retrieval of VBR Video in a Multimedia Server,” in *Proceedings of IEEE Conference on Multimedia Computing and Systems*, Ottawa, Ontario, Canada, June 1997

[68] Olav Sandsta, Stein Langorgen, and Roger Midtstraum. “Video Server on an ATM Connected Cluster of Workstations,” in *Proceedings of the 17th International Conference of the Chilean Computer Science Society (SCCC '97).* pp. 207-217

[69] H. Schulzrinne，S. Casner，R. Frederick，and V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications,” RFC 1889，http://www.ietf.org/rfc/rfc1889.html, Jan. 1996

[70] H. Schulzrinne，A. Rao，and R. Lanphier, “Real Time Streaming Protocol (RTSP),” RFC 2326，http:// www.ietf.org /rfcs/rfc2326.html, Apr. 1998

[71] P. J. Shenoy， P. Goyal， S. S. Rao，and H. M. Vin, “Symphony: An Integrated Multimedia File System,” *Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking*，1998. pp. 124-138

[72] P. Shenoy and H. Vin, “Cello: A disk scheduling framework for next generation operating systems, ” in *Proc. ACM SIGMETRICS 1998*. pp. 44-55.

[73] P. Shenoy and H. Vin, “Efficient Striping Techniques for Variable Bit Rate Continuous Media File Servers,” *Performance Evaluation*, vol. 38, pp.175-199, 1999.

[74] Cyrus Shahabi, Roger Zimmermann, Kun Fu, and Shu-Yuen Didi Yao. “Yima: a second-generation continuous media server,” *Computer,* Volume: 35 Issue: 6 , June 2002 Page(s): 56 –62

[75] J. R. Snatos and R. Muntz, “Performance Analysis of the RIO Multimedia Storage

System with Heterogeneous Disk Configurations," Proc.of *6th ACM International Multimedia Conference*, 1998

[76] J. Santos, R. Muntz, B. Ribeiro-Neto, "Comparing random data allocation and data striping in multimedia servers," *Proc. of ACM SIGMETRICS, 2000*, pp.44-55.

[77] Dinkar Sitaram and Asit Dan, "Multimedia Servers – Applications, Environments and Design," Morgan Kaufman Publishers, 2000.

[78] Valery Soloviev and Alex Rousskov, "The CANDID Video-on-Demand Server," Technique Report NDSU-CSOR-TR-95-07.

[79] R. Steinmetz and K. Nahrstedt, *Multimedia: Computing, Communications and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1995

[80] Guang Tan，Hai Jin，and Liping Pang，"A Scalable Video Server Using Intelligent Network Attached Storage"，to appear in *Proceedings of IFIP/IEEE International Conference on Management of Multimedia Networks and Services 2002*

[81] R. Tewari, D. M. Dias, W. Kish, and H. Vin, "High Availability for Clustered Multimedia Servers," in *Proc. of International Conference on Data Engineering*, Feb. 1996.

[82] Renu Tewari，Rajat Mukherjee，Daniel M. Dias，and et al., "Design and Performance Tradeoffs in Clustered Video Servers," *Proc. of MULTIMEDIA'96*. pp. 144-150

[83] C. A. Thekkath, T. Mann and E. K. Lee, "Frangipani: A Scalable Distributed File System," *ACM Symposium on Operating System Principles*, 1997. pp. 224-237

[84] W. Tetzlaff and R. Flynn, "Disk Striping and Block Replication Algorithm for Video File Servers," in *Proc. of ICMCS'96*, pp. 590-597, June 1996.

[85] N. Venkatasubramanian and S. Ramanthan, "Load Management in Distributed Video Servers," *Proc. 17th Int'I Conf. On Distributed Computing Systems*, IEEE Computer Society Press, Los Alamitos, Calif., 1997, pp.528-535.

[86] M. Vernick. *The Design, Implementation, and Analysis of the Stony Brook Video Server*. Doctoral Dissertation. Computer Science Dept. at State University of New York at Stony Brook. 1996. pp. 77-98

[87] Harrick M. Vin, Pawan Goyal, Alok Goyal, and Anshuman Goyal, "A Statistical Admission Control Algorithm for Multimedia Servers," In *ACM Multimedia*. pp.33-40, October 15-20, 1994, San Francisco.

[88] Harrick M. Vin, Alok Goyal, Anshuman Goyal, and Pawan Goyal, "An observation-based admission control algorithm for multimedia servers," In *Proc. of the First IEEE International Conference on Multimedia Computing and Systems (ICMCS)*, pages 234--243, May 1994.

[89] VoDKA Projects. http:// http://vodka.lfcia.org/

[90] VIOLA (VIdeo-On-LAns) Project, http://www.cintec.cuhk.edu.hk/mns/viola.html

[91] R. Wijayaratne and A. L. Narasimha Reddy, "System support for providing integrated

services from networked multimedia storage servers," in *Proc. of ACM Multimedia Conf.*, Sept. 2001.

[92] Song Wu, Hai Jin, "Symmetrical Pair Scheme: a Load Balancing Strategy to Solve Intra-Movie Skewness for Parallel Video Servers," in *Proc. of the International Parallel and Distributed Processing Symposium*, Marriott Marina, Fort Lauderdale, Florida, April, 2002.

[93] Song Wu, Hai Jin, Guang Tan, "Analysis of Load Balancing Issues Caused by Intra-Movie Skewness for Parallel Video Servers," to appear in *Parallel and Distributed Computing Practices, 2002.*

[94] Roger. Zimmermann, Kun Fu, Cyrus Shahabi, Didi Yao and Hong Zhu, ``Yima: Design and Evaluation of a Streaming Media System for Residential Broadband Services," *Proc. VLDB 2001 Workshop Databases in Telecommunications*, Springer-Verlag, Berlin, 2001, pp. 116-125.