RC 20705 (1/28/97)
Computer Science

IBM Research Report

The Research Server Complex Manager for Large-Scale Multimedia Servers

A. Dan, M. Eshel, J. Hollan, R. Kenneson, M. Kienzle,
J. McAssey, R. Rose, D. Sitaram, W. Tetzlaff

IBM Research Division
T.J. Watson Research Center
Yorktown Heights, New York

LIMITED DISTRIBUTION NOTICE

This report has been submitted for publication outside of IBM and will probably be copyrighted if accepted for publication. It has been issued as a Research Report for early dissemination of its contents. In view of the transfer of copyright to the outside publisher, its distribution outside of IBM prior to publication should be limited to peer communications and specific requests. After outside publication, requests should be filled only by reprints or legally obtained copies of the article (e.g., payment of royalties).

Research Division
Almaden · T.J. Watson · Tokyo · Zurich

The Research Server Complex Manager for Large-Scale Multimedia Servers

A. Dan, M. Eshel, J. Hollan, R. Kenneson, M. Kienzle,
J. McAssey, R. Rose, D. Sitaram, W. Tetzlaff

IBM Research Division
T. J. Watson Research Center
Hawthorne, NY 10532

Abstract: The Research Multimedia Server Complex is a scalable collection of heterogeneous multimedia servers which uses a variety of communications protocols and network types to deliver multimedia objects to clients. The multimedia objects, which vary greatly in size, may be a mixture of data types including video, audio, and image. Providing a uniform interface that can accommodate a large variety of servers and communication interfaces is a significant challenge. The major functions of the Research Server Complex Manager (RSCM) are as follows. The RSCM provides a uniform external interface to applications hiding the heterogeneity and making the server complex appear as a single multimedia server. QOS for streams is guaranteed by the RSCM by reserving resources for individual requests. The RSCM provides interfaces for entering multimedia objects into the server complex as well as for recording information about the objects needed for efficient management of the server complex. Finally, a single operational interface for managing the complex is provided by the RSCM.

The Research Server Complex Manager is designed and implemented in order to validate various design decisions and to experiment with alternatives. It has been used in trials and forms the basis for the Media Streamer product from IBM. This paper justifies the particular functions of the RSCM, explains the design decisions and tradeoffs, and summarizes the lessons learned during implementation and use. The overall Server Complex Architecture and other components are also discussed.

1 Introduction

Multimedia servers deliver digital data into a network destined for a display or sound generation device. A user hears and views the material through use of hardware, known as a client system. The client system might be a PC, workstation, video game, network computer, or set top box, and might contain application programs that create part of the visual and auditory interface. The client hardware may also have applications programs that choose the next multimedia object to be sent to the client machine. In addition to client based applications, server based applications may also determine what multimedia objects are to be delivered. These applications may be aimed at education, training, sales, and information services [1]. The applications may interact with existing computer systems and data bases, and hence, in many cases the application will be a logical extension of an existing information system, augmented with multimedia data.

The networking hardware that may be used to distribute digital multimedia objects to the client systems are of many different types. These include satellite broadcast, terrestrial broadcast, local wireless systems, cable television systems, telephone Asymmetric Digital Subscriber Lines (ADSL), fiber optic networks. Similarly, the transport protocols used over these networks are also numerous, and include TCP/IP, UDP, NFS, FTP, ATM.

Multimedia files are voluminous. To ensure delivery of multimedia data to the client system in a jitter-free manner, the response time of the system needs to be guaranteed. There are many possible ways to construct multimedia storage servers that store, retrieve and deliver large volume of multimedia data upon a client request. In some cases the servers are inherently scalable and in other cases scalability is achieved through multiple instances of the same multimedia storage server. High throughput to particular multimedia objects may be achieved by replication of multimedia objects, striping across multiple disk devices (i.e., successive sequential blocks of a file are placed on different devices), or storage in solid state memory.

The role of the Research Server Complex Manager (RSCM) is to allow applications, in either the application server or the client machine, to use a standard interface to request multimedia objects. All of the complexity of the multimedia server complex will be hidden by the RSCM. This will allow the complex to grow, change, and allow insertion of new technology without disrupting application code. The RSCM also takes on the responsibility for determining the capacity of the system and guaranteeing that once deliver of an object begins that it will continue to meet the stringent time demands of audio or video. When video objects are entered into the system they come through an RSCM interface. During this process metadata about the object is stored. The metadata is essential for knowing the quality of service requirements at playback. Finally, the RSCM has the responsibility for the operational interfaces to the multimedia server complex. It provides a consistent set of interfaces and then uses whatever internal commands are necessary to direct the elements of the multimedia server complex.

The remainder of the paper is as follows. In Section 2, we describe the end-to-end environment and the relationship of RSCM with other components. Section 3 describes some of the key concepts employed in the design of RSCM and the major functional objectives. An overview of the design of the RSCM and its components are described in Section 4. Section 5 describes how the RSCM is used in the Tokyo Metropolitan Government trial. Finally, Section 6 concludes the paper.

2 Environment

Figure 1 shows the end-to-end data path and environment for accessing stored multimedia data. The display clients (e.g., PC or TV) establishes connection to an application server for accessing stored data as in traditional client-server systems. However, applications accessing multimedia data are distinguished from traditional applications by the way they store, retrieve and deliver voluminous multimedia data. In order to ensure jitter-free (isochronous) delivery of data to the display devices both the server and network have to

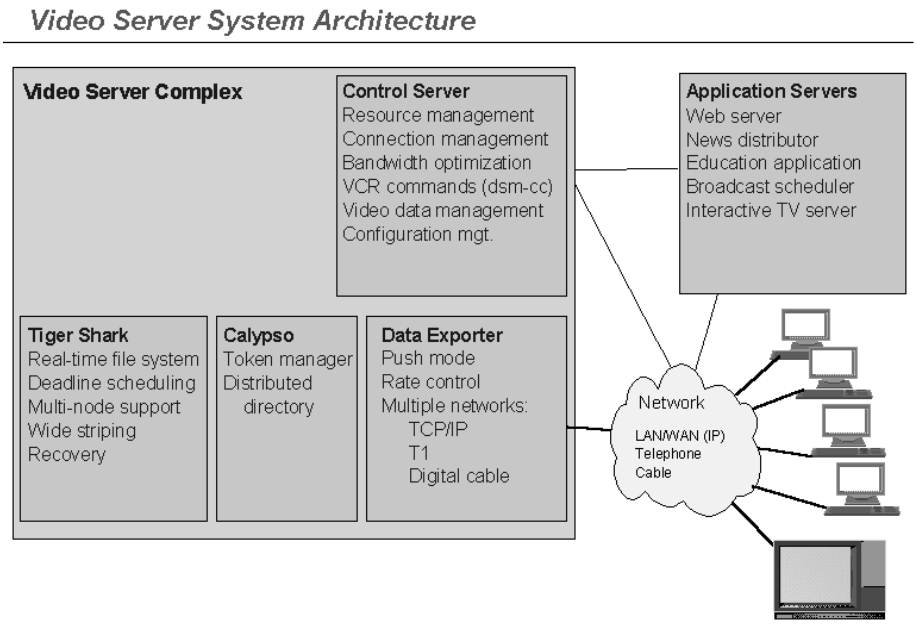

Figure 1: End-to-End system architecture for multimedia data access

reserve sufficient resources for this task [4, 13]. For the network this is achieved by setting up either a dedicated connection or providing some other admission control mechanisms. Similarly, the server has to set aside sufficient resources upon receiving each data delivery request. The data server also has to organize its data so that its performance is predictable for such isochronous delivery. Therefore, in many environments the data server(s) that are responsible for delivering multimedia data (e.g., Video Server Complex in Figure 1) may be separate systems from the application server systems. Additionally, many traditional applications may be enhanced to deliver multimedia data for better audio and visual user interfaces. This may also result in a similar logical structure of the server systems.

The access to video data needs to be controlled to guarantee strict response time bound (i.e., Quality of Service or QoS). There is a hard limit on the number of simultaneous streams that can be supported for a given server configuration. In response to a play request, the RSCM sets up a bandwidth (BW) guaranteed selected path from a chosen data source (a particular disk replica, cache, tape, etc.). The various applications may compete for the same set of resources, and hence, a single point of management is needed. In Figure 1, the Control Server module plays this gate-keeping role. It manages the resources (e.g., CPU MIPS, disk space and BW, etc.) of the video server complex. The Control Server module responsible for run time operations and the Administrative Server module (not shown in the Figure) that provides various administrative interfaces for system configuration together form the RSCM.

Once a session is established on behalf of an application, (i.e., a bandwidth guaranteed path is selected and appropriate resources on this path are allocated by the RSCM),

subsequent operations on video or audio files (e.g., OPEN, PLAY, PAUSE, CLOSE) can be performed. Upon receiving such requests, the RSCM translates them into file access operations. It informs the Data Exporter module the file position from which the playback should begin for the current stream, the rate at which the data needs to be delivered to the client. The Data Exporter reads subsequent blocks from the file system at the given pace (until the end of file or instructed to pause) and delivers data to the client. The data exporter may allocate several buffers for a stream and prefetch data for on-time delivery. The real-time file system (Tiger Shark file system in an AIX or RS/6000 SP environment [10, 9]) is performance enhanced for such operations. It stores video files by striping over a large number of storage devices to achieve load balancing, and schedules data retrieval operations according to their delivery deadlines. The Calypso Token manager provides locking and distributed directory services [2].

2.1 Design Objectives

A general purpose video server complex consists of heterogeneous sets of devices and/or storage servers, file systems, communication adapters and communication protocols (e.g., T1, TCP/IP, ATM, HFC). For example, as the system evolves with time new types of storage devices (e.g., faster, with larger storage capacities) may be added to the system. Addition or deletion of new devices should not be disruptive, i.e., no major reorganization effort should be required. The system should also be able to support from a small number of streams (under a single node smaller configuration) to a very large number of streams (i.e., of the order of thousands) with a complex configuration.

The RSCM presents a single, uniform image of the video server to its applications, hiding from them the configuration and internal interfaces of the data servers. The RSCM can support different kinds of data servers. At one point during the development the RSCM supported, in addition to the AIX based data server described in this paper, a special purpose data streamer of different architecture, and with different interfaces. Other data servers can be accommodated as needed. Where configuration details cannot be hidden completely, they are exposed through logical concepts. The conceptual elements map to physical components in a way that is transparent to the applications.

An additional objective of the RSCM is to manage the resources of the server complex and to provide QoS to multimedia applications. It hides the complexity of the server environment by providing a clean and well defined interface to the outside world. Such a "black box" approach allows new functionalities and performance optimizations to be introduced in the server easily without affecting the overall design of the end-to-end system. The module responsible for setting up an end-to-end reserved path from the server complex to an individual user does not have to keep track of the complexities of all possible server environments. Note that reservation of resources outside the video server are assumed to be carried out by an external agent.

In summary, the RSCM is the focal point through which applications and viewers interact with the video server complex. The RSCM server provides three types of functions.

- The stream delivery functions implement retrieval and playing of data streams. They provide an application with session management to establish a control server context for subsequent operations. Connection management manages the connections from the data servers to the viewers. Resource management allocates and keeps track of real-time resources in the data servers, such as the processing power and the I/O bandwidth, and performs admission control. The stream control operations allow the applications to the control video and audio streams.
- Asset management supports content management applications. It performs data import and data management, including data placement across multiple data servers. A simple catalog function supports the tracking of assets on the server.
- The server management functions support administering and monitoring of the video server by external agents.

3 Key Concepts and Functional Objectives

In this Section, we define some of the key concepts that make managing such complex configurations and providing QoS to applications easier. Various administrative and operational commands are built on these simple abstractions.

3.1 Grouping of Multimedia Contents: Asset, Asset Group and Bandwidth-to-Space Ratio (BSR)

3.1.1 Asset

An asset is a digitized multimedia resource that is stored for later retrieval as requested by an application. An example of such a resource is a digitized video, such as a movie. Typically a video is stored as a file in a file system supported by the data servers. An asset consists of one or more replicas of such a file (multiple replicas may be necessary to support multiple concurrent viewers), along with the system structures that describe the characteristics of the asset. Various information associated with an asset are stored in an auxiliary file (referred to as an *Metafile*). The asset attributes may include size, playback BW and location of each replicas and the popularity of the video (used in placement decision).

A *stream id* is a unique character string that identifies an asset. A stream id value is unique within an asset group (defined shortly). The stream id is assigned by the Research Server Complex Manager when an asset is first defined to the server. The server maintains a catalog of all known assets. This catalog is subdivided by asset groups, and contains a cross-reference of asset long names and stream id's. The Control Server provides functions to add and delete catalog entries, as well as to look up an entry by either the asset long name or the stream id.

3.1.2 Asset Group

The asset group is a logical grouping of assets stored across heterogeneous set of servers and/or striping groups (which will be defined in the next subsection). It is used for purposes of asset placement and management of assets with different owners. An asset group is mapped to one or more storage devices. The asset group definition satisfies two primary objectives:

- Logical fencing of assets owned by various applications. This is equivalent to grouping of files into directories under any file system. Each application is unaffected by the presence of assets in other asset groups, and hence, management of assets becomes easier. The asset groups can also be used for controlling access to assets by various applications.
- Grouping of physical resources using a symbolic name that can be viewed by an application as a single resource with various attributes (e.g., device BW, space, connectivity). Asset group can be used to co-locate a large set of related video files on the same data server. Hence, by establishing a static broadband channel during the opening of a session for accessing a particular asset group, it can be guaranteed that all assets under that asset group can be played over the same channel.

Each application can define its own asset group, using the command `csiDefineAssetGroup (assetGroupName, MaxExpectedViewers, avgExpectedViewers, space)`. The parameters `MaxExpectedViewers, avgExpectedViewers` represent the maximum and average number of concurrent viewers expected for this asset group. The resource manager uses this expected load information and information on space requirement and selects the appropriate data server for allocating space and bandwidth to this asset group [3]. The allocated space and bandwidth of an asset group can subsequently be changed by the command `csiChangeAssetGroup (assetGroupName, newMaxExpectedViewers, newavgExpectedViewers, newSpace)`.

A new asset is created in an already defined asset group through the command, `csiImportDefine`. An asset can be moved from an asset group to another group using the command `csiMoveAsset (strId, newAssetGroup)`. Note that `strId` (i.e., unique name of an asset) is defined as the string `assetGroup/assetId`. Moving an asset from an asset group to another may or may not result in physical movement of the asset from one data server to another. An asset group can be deleted using the command `csiDeleteAssetGroup (assetGroupName)`. Deletion of a non-empty asset group is not allowed.

In addition to the user defined asset groups, the following default asset groups are predefined:

- The default asset group `ALL` includes all the data servers in the system.

- Each data server is associated with a default asset group that includes all the striping groups of that data server and is referenced by the same name as that of the data server. Initially the bandwidth and space of the asset group associated with a data server is set to those of the data server. User defined asset groups reserve space and bandwidth on data servers. This results in appropriate reduction in space and bandwidth of the default asset groups. Similar adjustment in space and bandwidth of the default asset groups are made after invocation of the commands `csiChangeAssetGroup` and `csiDeleteAssetGroup`.

3.1.3 Bandwidth-to-Space Ratio (BSR)

To avoid load imbalance across storage devices, a simplistic solution would be to stripe video files over all the storage devices, i.e., to place subsequent blocks of a file in different storage devices according to a block allocation scheme. This results in a logical grouping of the storage devices and the group is referred to as a striping group. However, striping all storage devices into a single group is undesirable from several considerations [12]. In a general system the storage devices may be of different types. Under a simple round-robin striping scheme, either the Bandwidth (BW) or space of some of the devices may be under-utilized. Alternatively, more complex schemes will lead to additional complexity in the file system, which is a mainline path in the video server. Therefore, it is desirable to put different device types in separate striping groups. Multiple striping groups also limits the the impact of a failure of a single disk. By replicating a small number of frequently viewed video files across striping groups, the availability of the system (the number of lost viewers in case of disk failure) can be reduced [12]. Finally, it makes addition or removal of storage devices simpler, since only a single striping group is affected by such operations.

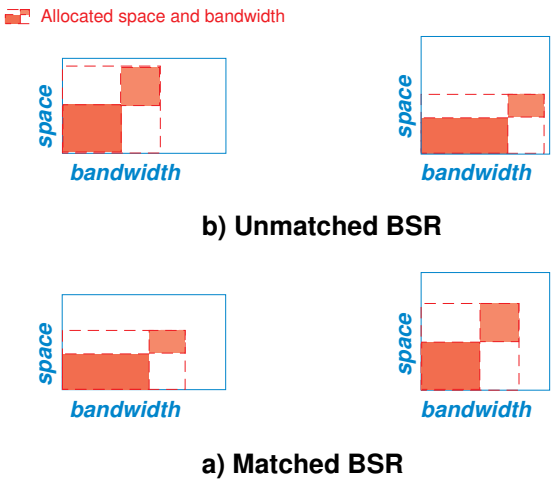

Figure 2: Illustration of BSR policy

A striping group is characterized by two quantities: Bandwidth and Space capacities. In Figure 2, these are represented by two axes of large rectangles. Similarly, an asset or asset

group can be characterized by its storage space requirement and the expected load (i.e., number of concurrent viewers). These are represented by smaller solid rectangles. Without careful placement of video objects, maximum utilization of both space and BW of these striping groups will not be possible. Figure 2 illustrates both an undesirable and desirable placements of four video objects on two striping groups. The dotted rectangles show the currently allocated total space and expected load on the placed files on each striping group. It can be seen that in the unmatched case, where large infrequently viewed objects are placed on the first group, the BW of that group may be underutilized during operation of the server. Therefore, as shown in the matched case, large and short, frequently and infrequently viewed objects have to be mixed properly on each striping group to utilize both the space and BW of the groups.

3.2 Network Connectivity: Port and Subnet

The video server sends data to clients over the external network via its interfaces (i.e., adapters). These adapters are connected to physical nodes of a server complex. Depending on external switching capability and network configuration, all clients may not be reachable from an adapter. However, a client may be reached via one or more paths from a video server complex. Before selecting an internal data path (i.e., network node, adapter, etc.) the RSCM needs to map all external addresses via *Port* and *Subnet*.

3.2.1 Port

A port is a hardware device or a portion of reservable bandwidth on a hardware device that can be allocated for data transmission in real-time. A port may also be referred to as an adapter. Each port belongs to a specific Data Server and connects to a specific subnet.

3.2.2 Subnet

A subnet is a collection of ports of a given type capable of connecting to a remote destination. Subnets are used at connection time to determine which ports are eligible for use in a given session.

3.3 Other Server Resources: Data Servers and Storage Devices

The various resources of a server complex that need to monitored are explicitly defined to the Control Server. .

3.3.1 Data Server

A Data Server is a combined hardware and software entity capable of storing and delivering in real-time, multimedia data. It may reside within an RS/6000, a discrete "black box" (such

as an Media Streamer), an RS/6000 Scalable Parallel (SP), or various other platforms. Its control interface must be accessible to the Control Server.

3.3.2 Storage Device

A storage device is a physical device or a set of devices that can be addressed as a single unit for purposes of determining capacity and placing data. A storage device is usually a physical disk or a striping group. In some Data Servers the entire contents of the Data Server may be recognized as a single storage device. Each storage device is controlled by a single Data Server.

3.4 Application Contexts: Sessions, Streams and Playlists

3.4.1 Session

An application establishes a data access session with the control server that relates all subsequent interactions from the same client until the session is closed. Depending on the nature of the session (e.g., static or dynamic), a data path may or may not be established. A static session is preferred when the maximum BW requirement of data streams accessed via this session are known a priori, and the overhead or latency of dynamic path establishment is to be avoided. However, if the resource requirement changes drastically for different assets or if there is a significant idle period (e.g., PAUSE) during a session resources are allocated on demand. This is referred to as a dynamic session. A data path may be terminated at the closing of a data stream or at the end of a session.

3.4.2 Stream

After establishment of a session, a client may playback one or more multimedia assets. A stream or data access stream is established for data delivery of each asset. The Control Server selects a data exporter (on the physical node of the selected data path, and hence, the adapter) and informs the data exporter of the asset replica or list of assets to be played back, the client address, and the rate at which the asset would be played back. The data exporter sets up the pipeline by establishing contact with the client, allocating buffer to this stream, and opening the asset replica. The subsequent application commands (e.g., PLAY, PAUSE, JUMP) starts/stops streaming of data.

3.4.3 Playlist

A *playlist* is a stream consisting of a list of assets that are to be played in sequence. By pre-defining playlists to the Research Server Complex Manager , the latencies associated with opening and closing the individual asset files are hidden by overlapping them with the

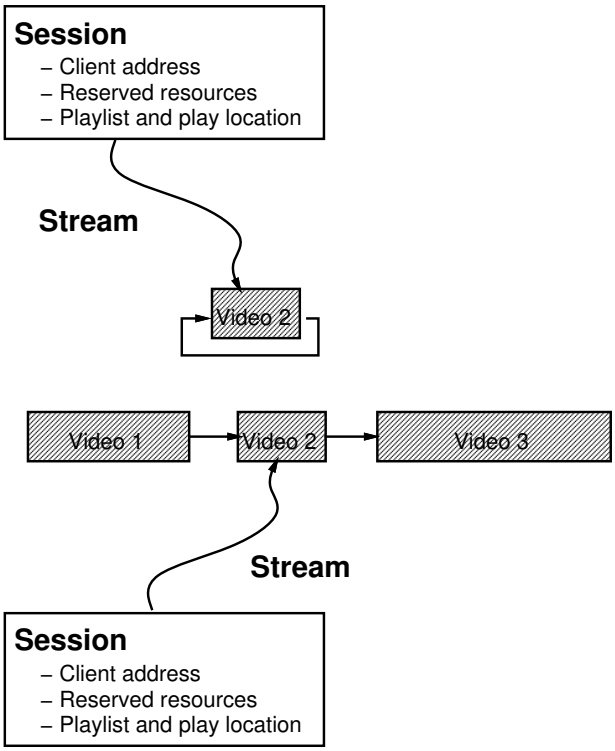

Figure 3: Illustration of Session, Stream and Playlist

reading of the prior and subsequent assets. Hence, the transitions between one video and the next seen by the client will be as seamless as possible.

The following are the Resource Manager interfaces for defining playlists:

`csiRMDefinePlayL` defines a playlist consisting of one asset and returns handle `streamHandle` for the playlist. A flag `loopFlag` specifies whether the playlist is to loop continuously and `strPos` specifies the position in the file from which play is to start. The function must be invoked after `csiRMOpenSession`. The playlist then becomes part of the session.

`csiRMAddToStream` inserts an asset `metaPath` into a previously defined playlist (stream) `streamHandle`. The new asset is inserted immediately after the n^{th} element of the playlist. If n is 0, the asset is inserted at the head of the playlist. The variable `strPos` indicates the position from which playback is to start. The parameters are similar to those for `csiRMOpenStream`.

`csiRMDelFromStream` Deletes the nth element of the previously defined playlist `streamHandle`.

The following functions are available for using playlists. If the client executes a VCR control action during playback of an element of a playlist, the corresponding Resource

Manager functions in support of VCR control actions (e.g. `csiRMJump`) should be invoked.

`csiRMStartPlayL` Notify Resource Manager that the `n`th element of the playlist is to be primed for immediate play. Returns the stream handle of the stream (`streamHandle`), data path of the selected asset replica (`dataPath`) and pointer to stream data structure (`strData`).

`csiRMAdvancePlayL` Notify Resource Manager that the previous element of the playlist has terminated and that the next element is to be primed for play. Returns the data path, stream handle and stream data as above. If the previous element was the last element of a looping playlist, the first element of the playlist will be primed. If the previous element was the last element of a non-looping playlist, `NULL` pointers will be returned.

4 Design Overview

The functions of the Research Server Complex Manager can be be grouped into three classes.

- *Asset management:* The assets are imported to the video server complex via these interfaces. For ease of import, various internal steps (i.e., selection of placement location(s), resource reservation on the import path(s), etc.) are combined into a single interface. In addition to the names of the asset and the asset group to which it is added, the importing application provides the parameters on the expected load and space requirement. The resource manager makes judicious decisions about the number of replicas needed for serving this expected load and determines the placement location for each of the replicas. An asset can be imported in real-time, i.e., from a live source, since appropriate resources are reserved in advance. As detailed in Section 3, the assets can also be moved across asset groups or deleted from data servers.

- *Session and stream control:* As detailed in Section 3, an session is established before starting any playback operation. Additionally, a stream is defined for each playback stream. As part of the above operations, a data source is selected based on the current utilizations of the various server resources and network connectivity, and a data path is established from the selected source to the client. Various playback and VCR control operations (e.g., PLAY, PAUSE, END) fall in this class.

- *System management:* The status of various resources and system components, streams, and sessions can be queried via these interfaces. These interfaces also include start up and termination of the server, and managing system configuration.

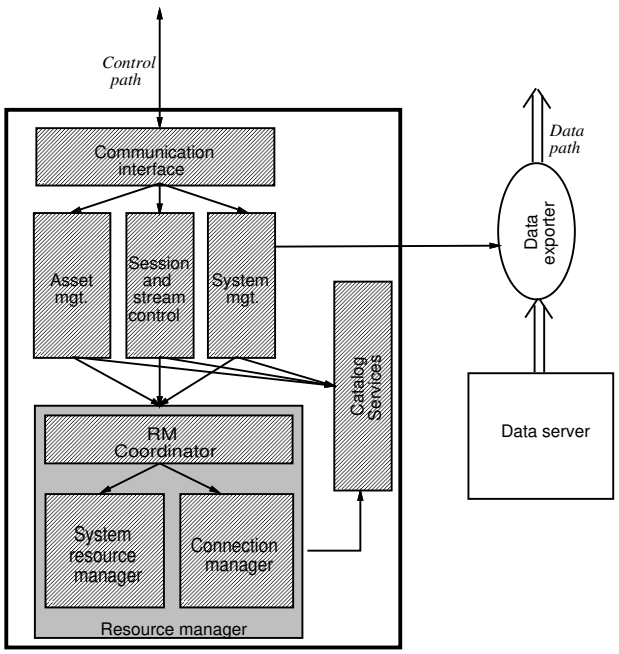

Figure 4: Structure of the Research Server Complex Manger

4.1 Structure of the Research Server Complex Manager

Figure 4 shows the components of the Research Server Complex Manager. An application communicates via Distributed Computing Environment (DCE) Remote Procedure Call (RPC) using the *Communication interface*. Concurrent requests are serviced on separate server process threads. Depending on an application request, appropriate services (i.e., *Asset management, Session and stream control* and *System management* functions) are invoked. The execution of these requests may in turn invoke catalog services that map external names to internal unique names and maintain metafiles. The maintenance of resource state (e.g., Maximum and available BW and space, defined resources, etc.), placement of assets, and selection of data paths are performed by the *Resource manager* (RM) functions. The RM module consists of three subcomponents: *RM coordinator, Connection manager* and *System resource manager*. Recall a data path emanates from a source (i.e., storage device) passes through other system resources[1] before before being connected to an external network via an adapter. The connectivity of the client and subnet configurations are maintained by the Connection manager (CM), while the system resources are maintained by the System resource manager (SRM). The information about sessions and streams are maintained by the RM coordinator. During the path selection, the RM coordinator invokes both the CM and SRM for enumerating all available paths before selecting a path.

[1] In a RS/6000 SP configuration a data path includes the node to which the storage device is attached, the switch, the network node to which the selected adapter is attached as well as the buffers, and CPU mips of these nodes.

4.2 Session and Stream Control Operations

For data access, the Control Server performs two types of mapping functions. It maps the DSM-CC[2] stream-ids [7] supplied by the applications into file names on the data servers, based on the DSM-CC operation and on the Data Server on which the file resides. For "trick mode" operations (e.g., fast forward), the Control Server selects the proper trick file. The Control Server also maps the DSM-CC operations into Data Server calls. In addition, the Control Server maintains mapping files needed for trick modes, such as the indices relating the different trick mode files of a multimedia asset to each other. All files relating to a particular multimedia asset are accessed through a single *metafile* which, in addition to these pointers, may also contain other information about the asset such as the multimedia format (e.g., MPEG-I, MPEG-II, Indeo, Wave files) data rates of the compressed data, compression modes (i.e., fixed bit rate or variable bit rate), ATM connection characteristics required, and buffering requirements. Any data related to the contents of the multimedia assets, or application dependent data, are assumed to be maintained by the applications using system software such as relational data bases, object oriented data bases.

When a new stream is created invoking the `OpenStream` routine, it first calls the Resource Manager to select the specific copy to be used for this stream. The RM also selects the port to be used for data delivery to the network. Using the above two pieces of information, it will instruct the Push Exporter to set up the pipe between Shark and the communication adapter and prime it. The `Play` routine instructs the Push Exporter to start the flow of data from the communication adapter buffers out into the network. Similarly, the `Pause` routine instructs the Push Exporter to stop the flow of data from the communication adapter. Calling the `Play` routine again will resume the flow of data. The `Jump` routine seeks to the specified offset and resumes play form that location of the stream. Finally, the `CloseStream` routine first calls the Push Exporter to stop the flow of data and then to takedown the pipe. It then calls the Resource Manager to release resources acquired at `OpenStrem`.

4.3 Resource Manager

We first describe the key data structures maintained by the RM, and provide an overview of the bandwidth reservation algorithm. The following data structures are maintained by the resource manager.

Logical Device Table: This is a table with entries for all the logical devices in the system. It is indexed by the *logical device id*. The fields are as below:

- *Logical device id:* This identifies the device. It consists of
  - *Device Type:* An enumerated type representing the type of the device (e.g., disk, CPU, striping group, Media Streamer, T1 adapter, etc.).

[2] Digital Storage Media Command & Control [8]

  - *Device Name:* Character string

- *Maximum Capacity:* Maximum capacity of device
- *Allocated Capacity:* Allocated capacity of device
- *Status:* Status of device
- *Device Map:* If a logical device made up of subcomponents, a pointer to a list of device map entries for the subcomponents.

Path Table: This is a table indexed by (source, destination) pairs. Each entry in the table represents a distinct path between the source and the destination. The table is created during the calibration process that measures the capacities of each observable system component [13]. During start up or system initialization, I/O streams are created in a controlled fashion so as to saturate various system resources (e.g., CPU of a node, I/O BW of a device or storage group). The number of simultaneous streams that can be sustained provides a measure of the capacity of the saturated system component. The capacities of the system components can be be stored in a permanent file and be reread during subsequent start ups (if desired). In the current version, the capacities are read from a file created off-line.

The source in a data path represents a source of video data such as striping groups or Media Streamers. Destinations are communication interfaces such as front-end nodes. Each entry in the path table (representing a path) is a linked list of the logical devices on which bandwidth has to be reserved if the path is used for delivering video. In the case of the Media Streamer, there will be only one list entry in the path (representing the server).

Bandwidth reservation Algorithm: The bandwidth reservation algorithm also does *path selection* in the case where multiple paths are available for playing a video (e.g. through multiple front-end nodes).

The procedure is invoked as `RM_Select (videoMetafile, clientStructPtr)`. `clientStructPtr` is an structure opaque to the Resource Manager that can be used by the Connection Manager to establish connections to the client.

The data path selection algorithm is as follows:

- Call Connection Manager with `clientStructPtr` to get list of front-end nodes for playing video.
- From `videoMetafile` get list of data sources.
- Apply filtering criteria to list of data sources (e.g. assetgroup).
- Lookup path table to find list of paths. If no paths, return failure.
- Sort paths based on desirability. If affinity routing is being used, select front-ends where video is being played. If not, use available capacity of the bottleneck node as the criterion.

- Loop over paths
  - Invoke Connection Manager to make connection to client using `clientStructPtr`.
  - If connection failure, try next path.
  - If connection made, reserve b/w on other components of path; exit.

Path establishment: The detailed path establishment flow depends upon the characteristics of the communication subnetwork connecting the client to the server, the adapter(s) available on the server and the type of connection desired by the application. The following connection types are supported:

T1 lines A T1 port identifier is returned to the L2 gateway (via which an external client request is accepted in a public network) from the `csiOpenSession` call. The L2 gateway has to dial in to this port and this port is used for the entire session.

ATM network ATM ports are dynamically assigned in response to `csiOpenStream` calls. The server dials out through the selected port to the client.

Mirage Mirage nodes are dynamically selected in response to `csiOpenStream` calls. The Mirage node establishes connections to the client.

4.4 Startup/Initialization/Termination

The Startup/Initialization/Termination component is activated when the Research Server Complex Manager is started. This component then initiates the other components of the system, monitors their execution, and coordinates an orderly termination of the system when requested. To make this possible, execution of other components and transient tasks is controlled by Execution Objects.

Startup The Startup subcomponent is broken up into two phases. First the Startup *process* is created, either as part of the operating system startup or by operator command. A separate instance of this process is started in each participating node if running in an SP2. This process exists only for the purpose of forking a second process which runs the Research Server Complex Manager . The Research Server Complex Manager process then begins execution with the Startup *thread*, which performs the initialization work to begin running the other threads that make up the Research Server Complex Manager . The Startup process remains active, periodically monitoring the status of the other process.

Initialization The Startup thread reads from ODM a list of definitions for Execution Objects. These describe tasks to be run in separate threads, the conditions for running them, and the state changes to effect upon termination of the tasks. An Execution Object with no execution conditions can run immediately. These initial Execution Objects perform the necessary initialization work.

Termination Termination consists of a prepackaged set of Execution Objects that send the appropriate signals, terminate other Execution Objects, and cleanup the system for an orderly (or other) shutdown of the Research Server Complex Manager .

Execution Objects The administrative process schedules, monitors, and controls execution of threads with Execution Objects. These are definitions of entities to be executed, when to execute them, and state changes that result from their completion. Other components of the system may still create and manage threads of their own or, if they wish, they may use the Administrative facilities to control thread execution with Execution Objects.

4.5 System Status Monitor / SNMP

The System Status Monitor interacts with an SNMP agent to keep an up-to-date record of the current status of critical components of the system, and to ensure that all components of the Research Server Complex Manager are functioning normally. These monitored resources may include Disk space utilization, Network ports, CPU utilization, Active paths, Opened streams, Available bandwidth on server/striping groups and Diagnostic information.

5 Tokyo Metropolitan Government Trial

The RSCM is being used in the Tokyo Metropolitan Government trial. In this section we describe in brief the role of RSCM, and demonstrate how the design has been validated by such a large scale trial.

5.1 Purpose of the System

The TMG Video On Demand project is intended to serve as a prototype for future "VOD" systems in Japan and other countries and to determine the feasibility of such systems in general. The requirements established for this trial included:

scalability in terms of overall bandwidth, number of customers served, number of simultaneous streams supported, and total content

true video *on demand* - no waiting for scheduled start times

broadcast quality video - i.e. 6Mbps MPEG2 streams for video

support for multiple stream formats

5.2 System Overview

At the heart of the TMG VOD system is the Video Server Complex, which manages resources, accepts commands from an application program, and produces digital output streams for playback to the user. An IBM SP/2 with 12 nodes was chosen for this purpose. The SP/2 is uniquely suited to provide the required scalability, and offers an excellent platform to integrate the RS/6000-based products that were already under development within IBM Research.

The structure of the TMG system is essentially as depicted in Figure 1 on page 3. The software components are distributed among multiple nodes of the SP/2. The complex contains a cluster of 160 disk drives containing 750 gigabytes of storage. It provides service to 500 subscribers, of which 100 simultaneous viewers can be supported. Output streams are delivered by the MPEG Stream Connection (MSC) card, which was also developed at IBM Research.

5.3 Role of the RSCM

The TMG implementation is a classic example of the need for the RSCM. The entire complex comprises arrays of each hardware component, which provides the necessary scalability, but must be managed in an intelligent and efficient manner. Each component of the system has its own external interface. Even components that serve the same purpose, such as communications adapters, may have completely different interfaces. The RSCM hides these differences from applications, making it possible for each component to perform its own function, without having to adapt to other components' unique characteristics.

6 Conclusions

The Research Server Complex Manager (RSCM) provides the external interfaces for a scalable collection of video storage servers. These interfaces include ones for inserting content, initiating playout of video, and managing the complex. An innovative video placement algorithm, called Bandwidth Space Ratio, is used to make decisions on allocation of disk space. The RSCM also makes automatic decisions to make multiple replicas of video files to increase the available throughput. In addition the RSCM innovative path based algorithms to determine server capacity, which is needed because RSCM manages admission control to assure on time delivery of video. Because the RSCM provides all the external interfaces it becomes possible to significantly change and reconfigure the video server without external impact. This was exemplified when the TMG server was grown from a single RS/6000 to a multiple node Scalable Parallel RS/6000. The RSCM is the precursor to the Control Server in the IBM product video servers called the Media Streamer and the Internet Video Server.

References

[1] Fox, E. A., "The Coming Revolution in Interactive Digital Video," *Communication of the ACM*, Vol. 7, July 1989, pp. 794–801.

[2] A. Mohindra, M. Devarakonda, "Distributed Token Management in the Calypso File System", Proceedings of the *IEEE Symposium on Parallel and Distributed Processing,* IEEE, New York, 1994.

[3] Dan, A., and D. Sitaram, "An Online Video Placement Policy based on Bandwidth to Space Ratio (BSR)", *Proc. SIGMOD'95*, San Jose, May 1995, pp. 376–385.

[4] Dan, A., D. Sitaram, and P. Shahabuddin, "Dynamic Batching Policies for an On-Demand Video Server," *ACM Multimedia Systems*, Vol 4, No. 3, pp. 112–121, June 1996.

[5] Kienzle, M., A. Dan, D. Sitaram, and W. Tetzlaff, "The Effect of Video Server Topology on Contingency Capacity Requirements" *Multimedia Computing and Networking*, Jan 1996.

[6] "OS/390 IBM LAN Server for MVS Configuration Files and Commands," *Document Number SC28-1732-00*.

[7] Microsoft Corp., "Microsoft's Software Architecture for Interactive Broadband Networks", *DAVIC/CFP/034*, Dec. 1994.

[8] "Digital Storage Media Command & Control" *ISO/IEC 13818-6*, Committee Draft, June 1995.

[9] R. L. Haskin, "The Shark Continuous Media File Server", Proceedings of the *IEEE 1993 Spring COMPCON,* San Francisco, CA, 1995, pp. 209-216.

[10] R. L. Haskin, "Tiger Shark - A Scalable File System for Multimedia" Submitted to *IBM Journal of Research and Development*.

[11] T. Sanuki, Y. Asakawa, "Design of Video Server Complex for Interactive Television", Submitted to *IBM Journal of Research and Development*.

[12] Dan, A., M. Kienzle and D. Sitaram, "Dynamic Policy of Segment Replication for Load-Balancing in Video-on-Demand Servers", *ACM Multimedia Systems*, Vol. 3, No. 3, July 1995, pp. 93–103.

[13] Dan, A., D. Sitaram, M. Kienzle and W. Tetzlaff, "A Path Based Approach to Bandwidth Reservation and Empirical Capacity Estimation of a Video Server Environment ", *IBM corporation*, (Pending US Patent).