Multimedia resource management in OS/390 LAN Server

by A. Dan
S. Dulin
S. Marcotte
D. Sitaram

In a multimedia server, resource reservation is critical for guaranteeing jitter-free delivery of video. In this paper, we describe the* resource manager *component of the OS/390™ LAN Server (video server) that implements resource reservation. The resource manager functions can be divided into three categories: (1) the* system management *functions that allow arbitrary multimedia resources to be dynamically defined, undefined, and calibrated, and their capacities monitored remotely via the Simple Network Management Protocol, (2) the* operational *functions that allow video streams to reserve and release resources for supporting playback without explicit specification of the needed resources, and (3) the* real-time data import *function that places videos on the storage devices so as to balance the load among the devices after making the necessary space and bandwidth reservations. We finally discuss research issues to exploit* economies of scale.

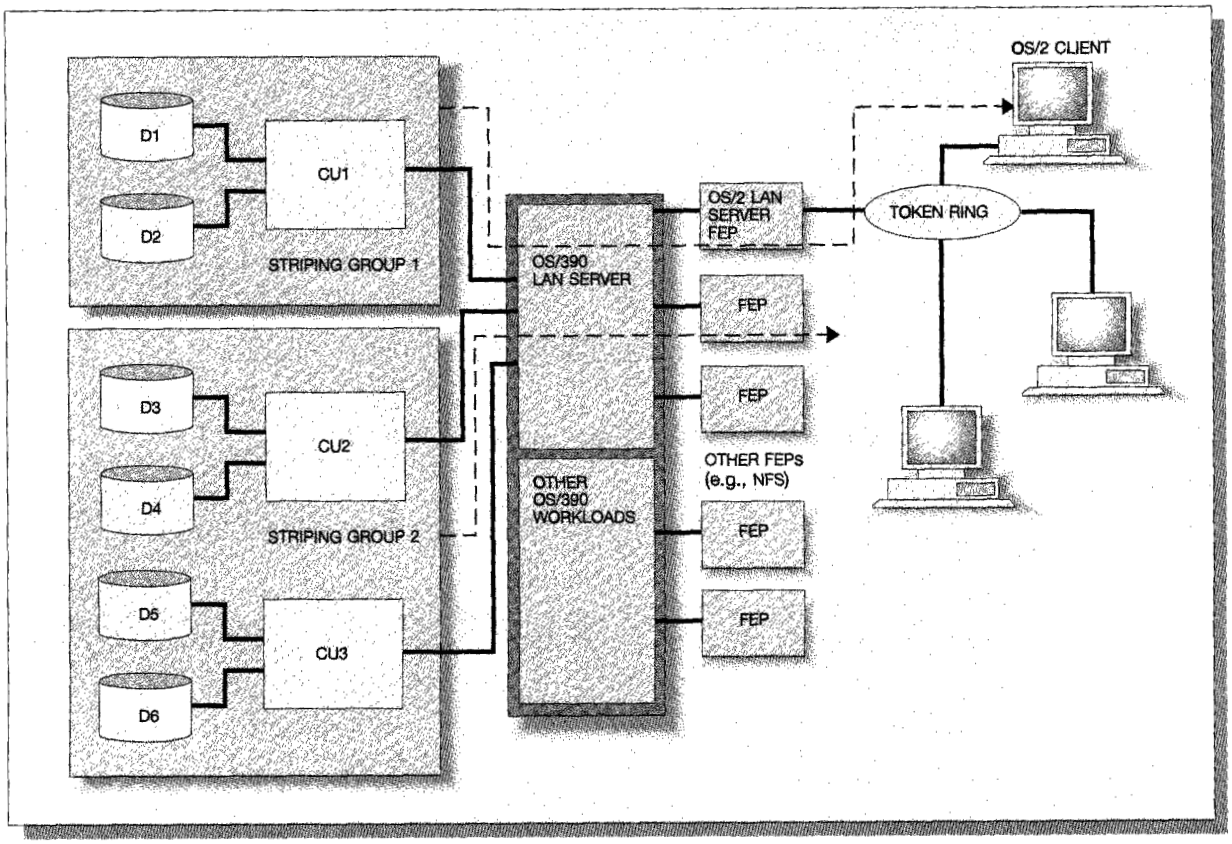
Rapid growth in computer, communication, and display technologies has led to the development of newer applications that embed continuous media objects such as video and audio, as well as traditional objects (e.g., image data). Examples of such emerging commercial applications are video-on-demand, distance learning, digital library containing various media types, and multimedia database.[1] These applications require development of multimedia servers capable of storing media data and delivering these data to remote clients.

The design of such servers poses new challenges in addition to traditional requirements such as reliability, high availability, resource optimization, and cost-effectiveness. Foremost among these new challenges not found in the design of traditional servers are the requirement for smooth continuous delivery of video and audio data to clients and efficient storage and retrieval of large media files.[2–4] Large variations in response times, such as those encountered in traditional server systems (e.g., in transaction and query processing, and in file servers) will lead to "hiccups" in video and audio delivery. To avoid variations in response time and guarantee continuous delivery, the server needs to set aside sufficient resources for each stream before starting its playback. It is the requirement for resource reservation that distinguishes multimedia servers from traditional servers such as file servers.

In this paper, we describe the design and implementation of the *resource manager* (RM) for OS/390* LAN Server, the component that implements this resource reservation. The OS/390 LAN Server evolved from an earlier version that was a traditional high-performance file server for workstation clients.[5] The addition of the resource manager component enabled this file server to support delivery of continuous media and to become a full-fledged multimedia server. The other enhancements that further improved the

0018-8670/97/$5.00 © 1997 IBM

Figure 1 OS/390 LAN Server environment



performance for such I/O-intensive applications are detailed in Reference 5.

In the next section of this paper, we detail our design objectives and provide an overview of the resource manager. In the section on implementation, the details of implementations and various performance issues (e.g., bandwidth and space utilizations, and concurrent data structures for high performance) are discussed. New algorithms (e.g., caching, batching, load balancing) that can exploit economies-of-scale for long video applications are discussed in the section, "Research issues." We then summarize and conclude this paper in the last section.

Design objectives

The primary objective of the resource manager is bandwidth (BW) management of various (physical and logical) system resources as well as management of space for storage devices. The BW (the data transfer rate of a device) and space management objectives include efficient use of the system resources, support of continuous delivery guarantee, and hiding of the complexities of the system from a higher-level application. Upon a request for playback of continuous media objects, the resource manager reserves appropriate resources necessary for playback without explicit enumeration from the application. The resource reservation is implicit, and the end user may not even be aware of this.

Figure 1 illustrates a typical OS/390 LAN Server (multimedia server) environment consisting of storage subsystems, server, and a set of front-end processors. The applications may range from playback of a long video to retrieving many small video clips interactively as in a shopping application. As mentioned earlier, prior to playback of any video material, a log-

ical channel (i.e., a BW reserved data delivery path) needs to be established either statically for the entire application session or dynamically as needed. The client requests for setting up an application session arrive at the resource manager over a two-way channel established for the purpose of exchanging control messages. The resource manager is responsible for admission control, and it keeps track of the available resources (e.g., disk BW, CPU[6,7]). In Figure 1, each data path emanates from a striping group and passes through a mainframe node and a front-end processor (FEP) toward a client node. (Striping is the use of multiple DASDs [direct access storage devices] as a single logical volume. The set of volumes is referred to as a striped set. An FEP is a LAN [local area network] server workstation running OS/390 LAN Server, through which LAN workstation users can access files stored on an OS/390 system. There may be other FEPs accessing other workloads or even accessing nonvideo files managed by the OS/390 LAN Server.) The resources that need to be managed by the resource manager are defined explicitly. (The resources on the LAN that connects clients to the FEP are managed by the Operating System/2* [OS/2*] LAN Server Ultimedia*.) The resource manager also needs to be aware of the capacity of each resource and, hence, can guarantee quality of service by controlling admission. The resources required to play back a video or audio file are enumerated by the *path-based reservation* algorithm that reserves necessary BW on all resources in the chosen path. Each path will capture all the resources that could potentially be a bottleneck, depending on the combination of workloads. There could be multiple paths for playing back a video. For example, if a video file has multiple replicas residing on different storage devices, each path may start from a different storage device. The resources that need to be kept track of (e.g., buffer space, adapter capacity, etc., in addition to disk BW and CPU MIPS) depend on the possible system configurations and on how well balanced the capacities of various resources are. In a more general environment with a heterogeneous set of components and access methods (e.g., different types of adapters, different transport protocols, multiple system nodes accessing the same set of disks, etc.), it is difficult to predict the bottleneck resources.[8,9] An important design objective is the ability to handle diversity of configurations.

The defined resource could either be logical or physical. For example, the resource manager needs to be aware of a striping group made of multiple devices, so that the striping group can be treated as a single logical device. Otherwise, appropriate resources need to be reserved on all devices over which a media object is striped. Resource management can also be hierarchical, where a single server can be partitioned into multiple virtual machines, and the resource manager for each component manages its share of resources.[10] For example, some capacities (of CPU and disk) can be set aside (by setting the capacities of the resources to a lower value) from the resource manager to make sure that some resources are always available for system management tasks. We will use a similar method to set aside some disk BW for staging of assets. Additionally, by providing a higher priority to multimedia and other real-time tasks, the available background BW can be used for other services such as traditional file services (also see earlier discussion in Note 6).[11,12]

Functional objectives. The resource manager needs to support three classes of functions: system management, operational functions, and real-time data import.

System management. As described above, the bottleneck resources that need to be managed also need to be defined explicitly. The capacities of these resources (also referred to as devices) need to be either measured or set explicitly so that admission can be controlled. By setting capacities explicitly, the administrator can sometimes adapt to an unplanned situation or unforeseen condition. For example, if the measured capacities are incorrect, or the current system state results in a high rate of jitter, the administrator can explicitly set the capacities of the resources at a lower level.

Upon failure of a device, the resources need to be undefined or marked inactive, so that no newly admitted stream would use this device. Similarly, the resource manager needs to be notified after bringing a device on line. The resource manager should also support querying the state of these devices (e.g., active or inactive, allocated, and maximum capacity). To integrate monitoring of the multimedia server with those of other system components in an end-to-end multimedia environment, various exceptional states (e.g., low disk space) should be sent to a remote agent (e.g., Simple Network Management Protocol, or SNMP).

Operational functions. For isolating an application from the complexities of the environment, it is desirable for the application not to be aware of the various resources that need to be reserved by the re-

source manager for playback of a stream. These reservations should be automatic when the playback of a media object is requested. The various resource capacities (e.g., device BW, CPU MIPS) required to play back a media object can be maintained in an associated file to make this reservation process transparent. The associated file is referred to as a *metafile*.

A metafile contains the BW information for a particular asset, the number of replicas of the asset, and their locations. The client OPENs the metafile for an asset that results in selection of a replica for playback and implicit reservation of required resources by the resource manager. An *asset* is a data set that normally contains multimedia data to be delivered at a guaranteed rate to the client. There may be multiple copies of the asset that are referred to as *replicas*. In order to guarantee that sufficient BW is available for a specified number of concurrent users, multiple copies of the asset may be required on different volumes. When the client OPENs a metafile entry for an asset and resource reservation has been successful, this instance of the delivery of the asset to the client is referred to as a *stream*.

For efficient use of system resources, the reservation procedure should also determine all available paths from the storage devices containing replicas of the requested multimedia object to the network interface. The network interfaces may include FEPs connected by ESCON* (Enterprise Systems Connection) channels as well as various types of adapters (e.g., ATM, or asynchronous transfer mode). It should then select the path that is least utilized to balance the load across all devices. The reservation of all required resources should be *atomic*; that is, all resources will be acquired at once, or no resources will be acquired at all. Similarly, the release of resources is to be done atomically, and implicitly when a stream is closed. The atomic acquiring and releasing of resources will also be helpful in supporting recovery of a stream upon failure of a device on the delivery path (e.g., controller, disk, network interface).

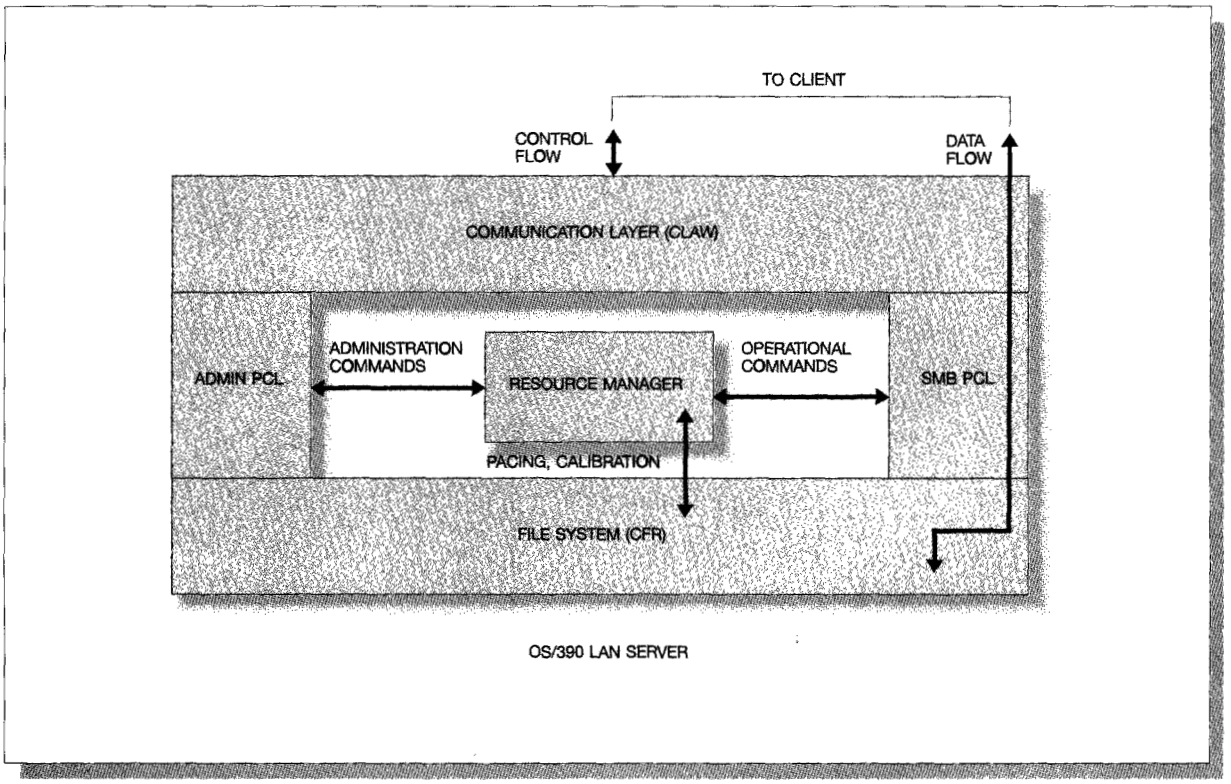
Real-time data import. Intelligent decisions are required on the part of the resource manager to place the media objects so that resources can be utilized efficiently. For example, hot (high-demand) videos should be placed on devices that have a higher BW. Also, hot and cold (low-demand) videos need to be mixed (these issues are detailed later in the section on implementation) so that both space and BW utilizations are matched. Before a media object (also referred to as an asset) is imported, appropriate space needs to be allocated, and appropriate BW needs to be reserved on other resources on the import path. The operations should be coordinated by the resource manager so that import applications do not have to keep track of various steps.

The placement decisions for new media objects would depend on the current placement of other existing media objects and their expected access patterns. Therefore, the resource manager should also keep track of the placed assets and their expected total access rate.

Other design objectives. In addition to the above functional design objectives, the following set of objectives were also taken into account:

- *Continuous media and traditional file services:* The ability to serve traditional services such as file services, transaction processing, etc., without affecting services for continuous media makes the OS/390 LAN Server environment a general-purpose platform. This can be achieved (1) by providing a higher priority to continuous media tasks so that their services are not affected[13] and (2) by controlling admission of continuous media tasks or by setting aside a fraction of the resource capacity through the resource manager.
- *Scalability:* To support a large number of clients and high rate of access requests, multiple resource manager threads need to access the same set of data structures (e.g., resource and stream tables). Therefore, the data structures need to be thread-safe, and locks and latches need to be used. Also, to avoid contention for access to data structures, the locks should be of fine granularity, and lock mode should be upgraded to exclusive only when it is needed. The locks need to be acquired in specific resource order to avoid deadlock. The details of these issues are discussed in the next section.
- *Transactional support for algorithms:* The various operational functions and algorithms access multiple data structures. To maintain consistency, the operations performed on behalf of a single client request need to be atomic. For example, multiple resources are acquired upon a single playback request. If some required resources are not available, or some other software failure occurs during this process, a cleanup action is started to release the resources acquired so far. This feature is very important for making the system robust and reliable. Certain algorithms (e.g., data placement) require a broad view of the available resources to determine what resources are to be acquired. A pro-

Figure 2 OS/390 LAN Server components



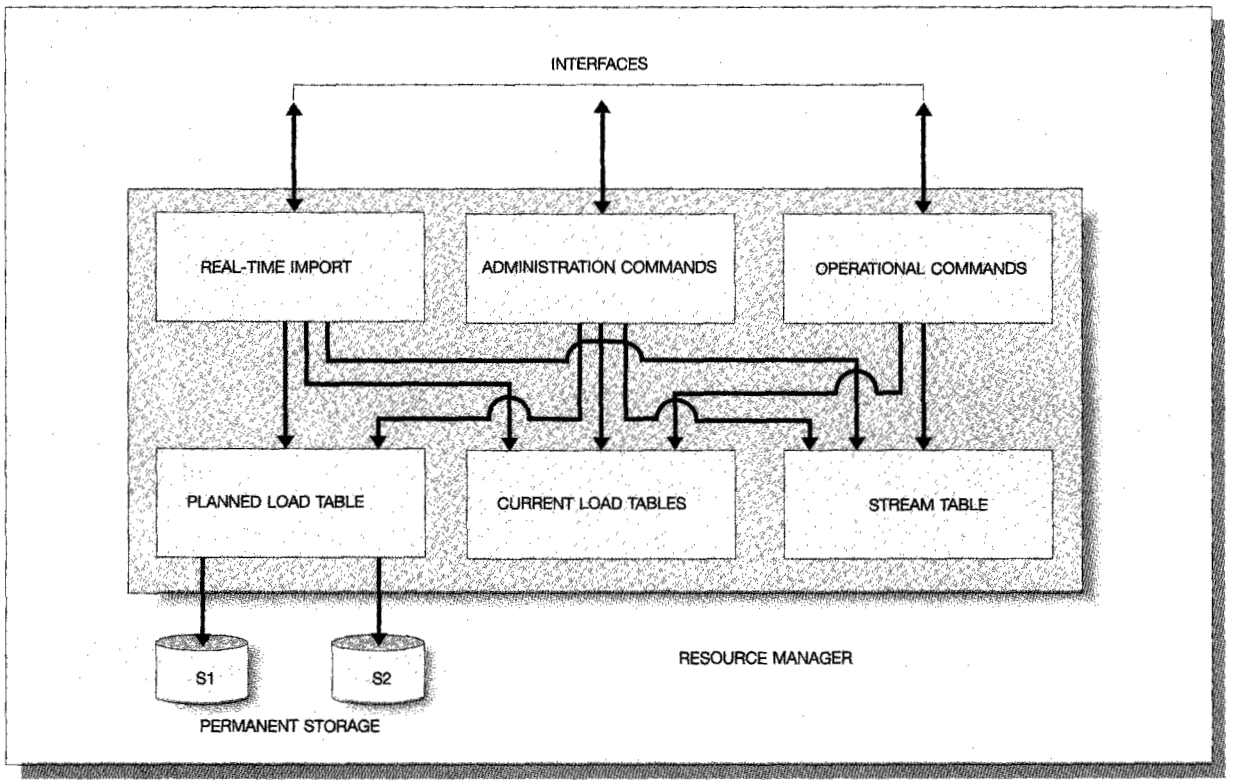
tocol similar to two-phase commit is used to follow resource acquisition order for deadlock avoidance, where in the first phase shared locks are used to determine the resources to be acquired, and then in the second phase exclusive locks are acquired to modify the resource entries.

- *Portability:* Significant attention was paid to this important software design principle by keeping the interfaces clean and using macros for system-dependent services (e.g., storage allocation, lock services, thread support, etc.). During the development process the code was mostly debugged on an AIX* (Advanced Interactive Executive*) platform and seamlessly integrated with the OS/390 LAN Server environment. After the code was "ported" to an RS/6000* SP* environment, it became the base code for the resource manager in the AIX Video Server.[14]
- *Platform for newer performance enhancement algorithms:* In the section on suggested future enhancements, we survey various performance enhancement algorithms (e.g., batching, caching) for a large-scale server that benefits from economies of scale. The requirements for implementing such algorithms in the resource manager were taken into account.

Implementation

The resource manager functions can be grouped into three categories (as described earlier): system management functions, real-time data import functions, and operational functions. These functions are invoked by various other components of the OS/390 LAN Server as described in more detail below (see Figure 2). The communication layer (CLAW) receives various commands (e.g., OPEN, READ, and WRITE file requests, administration commands) from clients and sends either appropriate responses or data, or both, to the client. The administration protocol conversion layer (Admin PCL) in turn sends the administration commands to the resource manager. The resource manager may call the file system (the Common File Repository, or CFR), for example, to calibrate the throughput of a striping group. Some operational commands (e.g., opening a media file)

Figure 3 Resource manager components



require reservation of resources, and hence, the appropriate file system protocol conversion layer (the server message block, or SMB PCL in this example) sends appropriate operational commands to the resource manager.

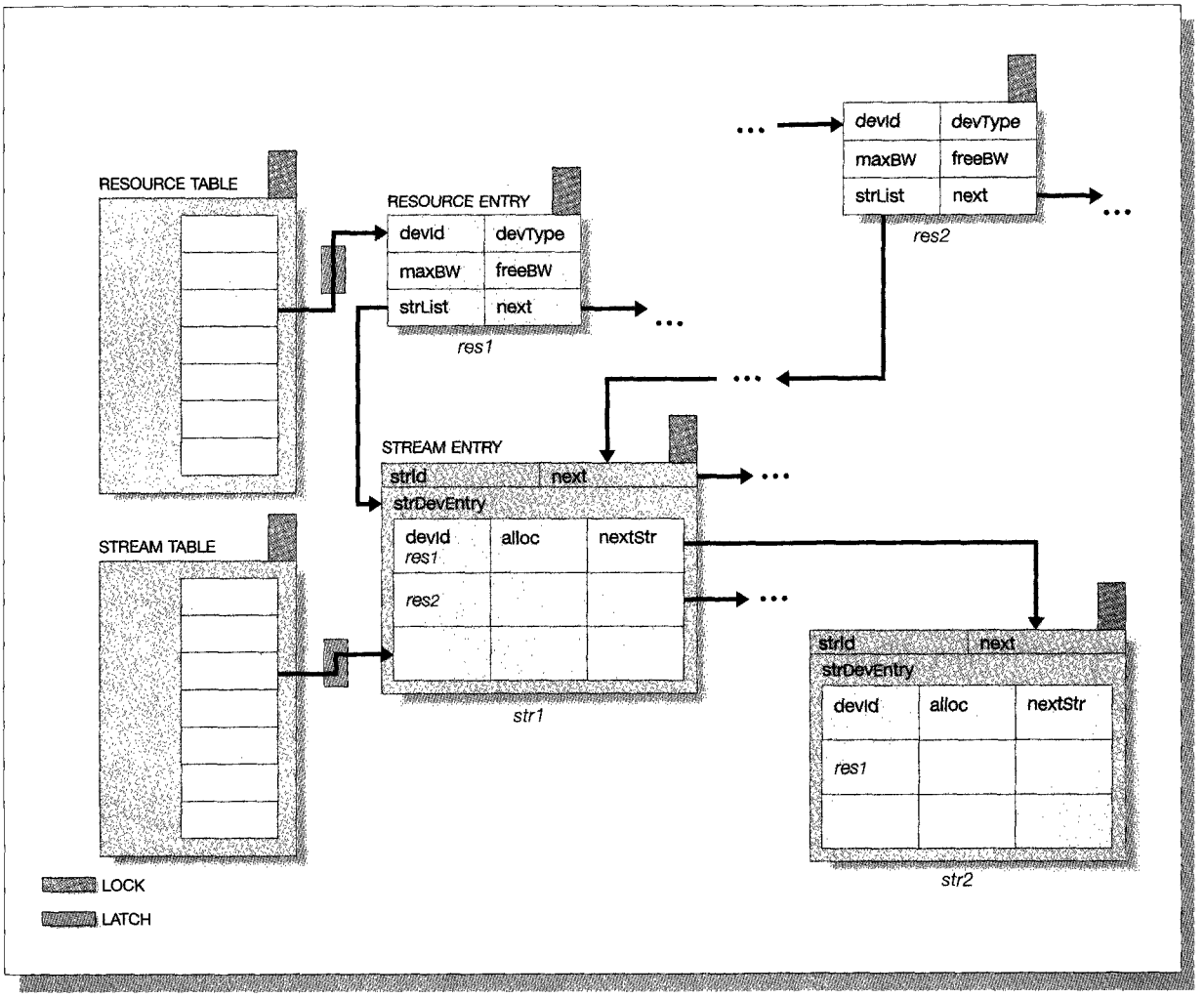
All the resource manager functions are nonblocking and are executed in the thread of the caller. Fine-grained locking allows resource manager functions to have concurrent and atomic access to the resource manager data structures. If execution of any of the functions is unsuccessful, cleanup is performed automatically. Figure 3 shows the components of the resource manager. The resource manager maintains three types of tables:

1. Planned load tables that are used for placement of media files on the storage devices. The placement policy attempts to match the BW and space of a device to the expected load and allocated space of the files placed on this device.
2. Current load tables that maintain the available capacities of various resources and the streams accessing these resources
3. Stream table that maintains the list of current streams and the set of resources reserved for these streams

In the following subsections, the resource manager data structures are described first, followed by a description of the resource manager functions.

Resource manager data structures. There are two main resource manager data structures—the resource tables and the stream table (see Figure 4). The resource tables contain resource entries that keep track of the state of the resources in the system, whereas the stream table entries store information about the various streams. Both *locks* and *latches* are used for ensuring concurrent access. Locks are used to obtain access to logical entities (such as tables or individual resource and stream entries). Latches are temporary locks that are held while traversing implementation-dependent data

Figure 4 Resource manager data structures



structures (e.g., hash row chains) and released once the desired entry is found. Deadlock is avoided since locks on various resources are obtained in a prespecified order and latches are released before obtaining any further locks. For the purpose of keeping track of the streams accessing a device, and the devices accessed by a stream, the various entries in the tables are cross-linked. These links have to be maintained in a consistent state while atomically updating the tables. In the following subsections, we first describe the resource tables, then the stream table and the cross-linking between the various tables.

Resource table. The resource manager maintains one resource table for each type of resource (e.g., striping group,[15] communication adapter). All the resource tables have the same structure, making it easy to extend the resource manager for handling new types of resources. Each resource table is a hash table of resource entries, indexed on the identifier of the resource. The resource ID is an arbitrary character string. Figure 4 also shows the main components of a resource entry:

- *devId*—the identifier of the resource. This component is used to index the resource table.
- *devType*—the hardware device type. As described below, this component can be used during initialization for determining the maximum BW of the device.

- *maxBW*—the maximum BW that can be allocated on this device. As described below, this component may be less than the actual maximum BW of the device.
- *freeBW*—the unallocated BW on the device
- *next*—the next resource entry on the hash chain
- *strList*—pointer to a list of stream entries that hold reservations on this device

Stream table. The stream table is a hash table of stream entries, indexed by the stream identifier. As shown in Figure 4, the main component of the stream entry, apart from the stream identifier strId and the hash chain pointer next, is a subtable called the strDevEntry table. This subtable contains a list of the devices on which the stream has a reservation. Each row of the subtable contains the resource ID of the resource that this row is for, the amount of reserved BW on this resource, and a pointer nextStr used to maintain the list of all streams that have reservations on this resource.

Cross-linking. Figure 4 illustrates the cross-linking between resource entries and stream entries that store the information about the resources held by various streams and the streams that are accessing various resources. By scanning the strDevEntry table, it is possible to find the resources on which a stream has reservations. For example, stream str1 has reservations on resources res1 and res2. The list of all stream entries that have reservations on a resource can be found by starting from the strList pointer in the resource entry and following the corresponding nextStr entries in the strDevEntry tables. In the example, the first stream that has a reservation on res1 is found by following the strList pointer for res1 and is str1. The next stream can be found by locating the strDevEntry row for res1 in str1 and following the nextStr pointer to str2. Since str1 has reservations on both res1 and res2, the stream entry for str1 will occur on both the stream entry lists for res1 as well as res2. The stream entry lists are NULL-terminated by convention. Note that while str1 and str2 have one resource (res1) in common, they need not have other resources in common and, hence, may be on different stream entry lists.

System management functions. The resource manager system management functions consist of configuration functions to dynamically add and delete various resources and define their BW (QOSM_AddDevice( ), QOSM_DeleteDevice). The BW also can be changed explicitly by (QOSM_SetBW( )). The state of the resources can be queried by various query functions (e.g., QOSM_QueryDevice( )). Additionally, during operation of the system, various alerts (high-priority events that warrant immediate attention) may be generated to signal the occurrence of exceptional events (e.g., denial of service for a request, low disk space, etc.). These alerts are sent to an SNMP agent for monitoring by the administrator.

The video server configuration is stored in a configuration file that is read in by the OS/390 LAN Server initialization routines. These routines then invoke the resource manager QOSM_AddDevice( ) function to define to the resource manager the resources (e.g., striping groups, network adapters) in the system. In case of failure to add a resource (e.g., label on a striping group proves to be unreadable) the QOSM_DeleteDevice( ) function may be invoked to delete the device. The system is ready for operation only after all the resources have been defined. Since the resources available may change dynamically (e.g., because of device failure or bringing a new device on line), the configuration functions can also be invoked during the operation of the system to add or delete resources or change their BW (using QOSM_SetBW( )).

An important part of defining resources to the resource manager is defining the BW of each resource. Defining the BW can be done by a number of methods as follows and is specified by one of the parameters to QOSM_AddDevice( ). The BW of a resource may be estimated by measurement, a process referred to as *calibration*. Calibration allows estimation of the BW of devices of unknown type, as well as accounting for the variation in BW between different units of a device. In the current version of the OS/390 LAN Server, calibration can be carried out only for striping groups.[16] The calibration process consists of starting multiple threads that read at random from the striping group and estimate the maximum read BW achievable. The number of threads to be started depends upon the number of disks.[11] Thus, the supported configurations for calibration are those where the bottleneck that limits the maximum achievable BW is in the striping group and not on another resource (e.g., the processor running the threads). To support more complex configurations in the future, it is possible to implement a more detailed measurement process that determines the bottleneck system components and measures the BW of each component.[9]

If calibration is not feasible, the administrator may explicitly specify the BW or direct the resource manager to estimate it based on the device type.[17] Ex-

plicit specification of the BW allows the administrator to reserve a portion of the resource BW for use by nonmultimedia applications, since the resource manager will not use more than the specified BW. Once the BW of a device is estimated by any of the methods, it is stored in permanent storage so as to allow quick restart of the system.

Operational commands. The resource manager operational functions reserve and release the resources needed for playback of a stream (RM_Select( ), QOSM_Release( )). Both foreground and background mode reservations are supported. In addition to reserving the required resources, RM_Select( ) also performs load balancing where multiple replicas of a video are available. Since the operational functions are the mainline paths through the resource manager, special attention has been given to their concurrency and scalability.

The resource manager reserves and releases resources for a stream when the stream starts or stops. These resources correspond to OPEN or CLOSE operations on a multimedia file. The operational functions of the resource manager are invoked during the OPEN or CLOSE operations on a file by the other components of the OS/390 LAN Server.[5] The OPEN will fail if the BW reservation is unsuccessful. It is not necessary for applications to understand the system configuration and explicitly reserve BW on the components needed to play back a stream. The *path-based reservation* algorithm in the resource manager atomically reserves all the needed resources; i.e., either all the required resources are reserved on behalf of the stream or no resources are reserved (if BW is not available).

The operation of the RM_Select function is as follows. The function finds all replicas of the requested video using the metafile. Subsequently, it considers in turn all the paths from the replica to the communications adapters that are reachable from the client. The function selects the path with the largest free BW, where the free BW of a path is determined by the BW on its bottleneck resource. The selection process actually occurs in two phases. In the first phase, only those replicas that have a "normal" status are considered. If the attempt to play from such a replica fails, the selection process enters an *emergency scan* phase where replicas not normally considered (e.g., replicas marked for deletion but not yet deleted) are considered as well.

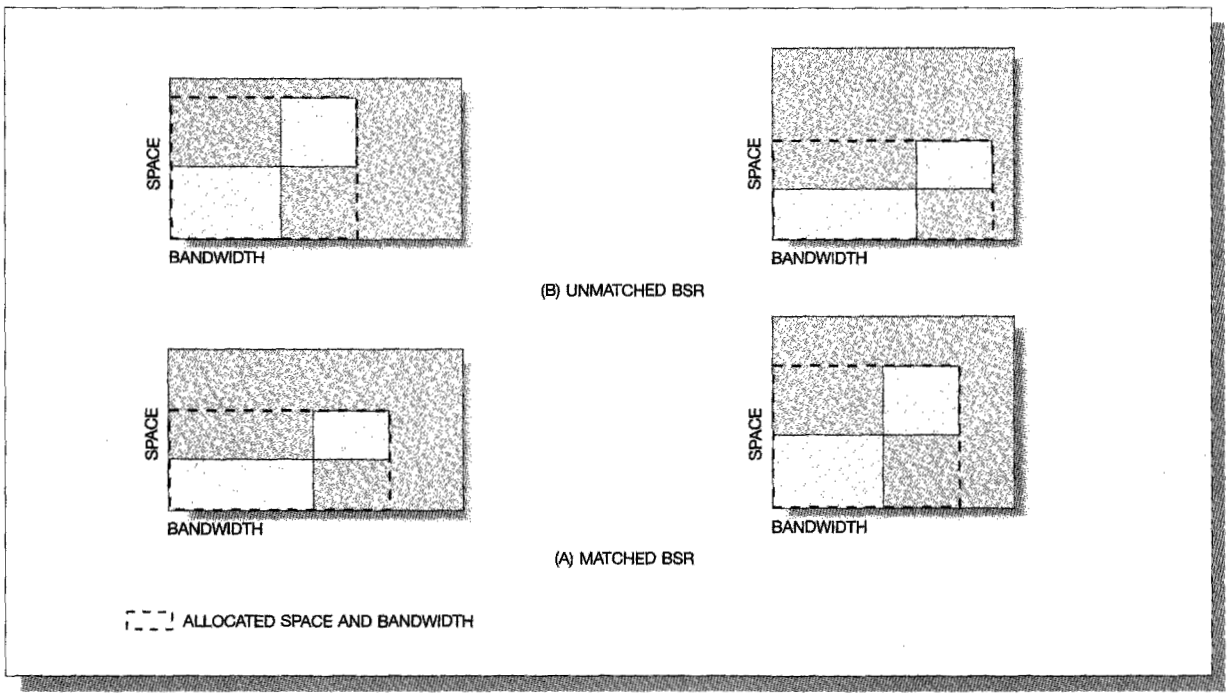
Since the operational functions are by far the most frequently executed resource manager functions, it is important that they be scalable and highly concurrent. This is achieved by locking only the minimum number of entries needed to ensure consistency and atomicity. The details are as follows. While finding the best path, the RM_Select function has to keep track of two paths: the best path found so far, and the path currently under consideration. These paths have to be locked, since their BW may be updated. To avoid deadlock, the function considers (and locks) the paths in sorted order. Because only two paths at most are locked at any given time, a high degree of concurrency and scalability is achieved.

The resource manager reserves and releases resources for a stream when the stream starts or stops.

The resource manager also supports the playback of real-time and non-real-time data from the same striping groups through the support of foreground and background I/O operations. A priority mechanism in the file system ensures that background I/O activity is performed only when there are no pending foreground I/Os. Background I/Os are used for non-real-time data and do not have any associated BW reservation. Because of the priority mechanism, they obtain any BW that is not reserved for foreground I/Os. As described earlier, to avoid starvation, BW can be set aside for background I/Os by setting the maximum BW of a resource to be less than its actual maximum BW.

Asset import. Importing assets into the video server environment consists of making placement decisions so as to efficiently utilize both the space and BW of the storage system in allocating necessary space and reserving appropriate BW for writing into the storage devices. These decisions cannot be made at the file system level since the file system does not have the necessary information regarding the resources and their capacities. Hence, it is important that placement be carried out in conjunction with a resource

Figure 5 Illustration of bandwidth-to-space ratio (BSR) policy



manager *placement policy*. In the OS/390 LAN Server resource manager, the placement policy is implemented by the RM_SetViewers function.

Since demand for a file changes with time and the storage device capacities and BWs may also change (because of changes in the configuration), placement decisions are also dynamic. Hence, it is necessary that the placement policy be an *on-line* policy, and that it can be invoked simply. In OS/390 LAN Server, there is a VIEWERS command that invokes the RM_SetViewers( ) to make any necessary changes in placement. For example, an on-line agent that monitors demand for various videos can use this command to dynamically change the placement to adapt to varying demand. In the following subsections, we first discuss the considerations behind the design of the placement policy, then give a description of the placement policy and describe some experimental results.

Placement policy considerations. A general multimedia environment may contain both large video objects (e.g., long movies) and short video clips (e.g., objects in interactive applications such as shopping, medical, etc.). The viewing frequency of these objects also varies, i.e., some movies as well as short clips will be viewed more frequently than others. The disk striping groups used for storing these videos are limited both by the number of concurrent streams they can support (i.e., *BW*) and by the number of video objects they can store (i.e., *space*). Without careful placement of video objects, maximum utilization of both space and BW of these striping groups will not be possible as illustrated in part B of Figure 5. It shows an undesirable placement of four video objects on two striping groups. The BW and space capacities of the groups are represented by rectangles with space on one axis and BW on the other axis. The allocated space and BW for each video object are shown by a solid rectangle, and each dotted rectangle shows the currently allocated total space and expected load on the placed files on this device. It can be seen that large, infrequently viewed objects are placed on the first group. The BW of that device may be underutilized during operation of the server. Hence (as shown in part A of Figure 5), 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.

A simplistic solution to the load imbalance problem is to stripe video files over all the storage devices, i.e., to include all storage devices in the system in a single striping group. However, as discussed below, there are many considerations that make support for multiple striping groups in a video server desirable. First, from practical considerations, the storage devices in the system may be of different types. If simple round-robin striping schemes are used, either the BW or space of some devices will be wasted. More complex schemes lead to much additional complexity in the file system, which is a mainline path in the video server. Hence, it is natural to put each device type in a separate striping group. The presence of multiple striping groups also limits the impact of the failure of a single disk. It also makes adding or removing disks simpler, since the entire video server is not affected. It is also shown in Reference 18 that with multiple striping groups and by replicating a small number of frequently viewed videos, the availability of the system (the number of lost viewers in case of disk failure) can be reduced.

Overview of the BSR policy. The placement policy used in the OS/390 LAN Server is referred to as the *bandwidth-to-space ratio* (BSR) policy.[19] The BSR policy takes three inputs: the identifier and new expected load of the video object to be placed and the required rate for creating the replica. The creation rate will depend on how quickly the video has to be placed (e.g., staging rate for retrieval from tape; data rate for import from real-time sources such as video cameras). The BSR policy makes two important decisions when placing a video object: estimating the number of replicas that are needed and deciding where to place them. The algorithm consists of four phases as stated below. Details of the algorithm can be found in Reference 19. In the first phase, the policy determines whether additional replicas are needed by examining the additional expected load that can be supported by the existing replicas and considering the resulting mismatch in the BSRs of the devices. If more replicas are required, the second phase of the BSR policy selects additional devices on which to place new replicas. With the devices selected, creating a new replica of the video will cause the BSR of the videos on the device to more closely match the BSR of the device. In the third phase, the BSR policy allocates the expected load to all the selected devices so as to match the BSRs of the devices. A replica consolidation phase is entered only if the policy is unable to place all of the expected load. In the consolidation phase, some of the existing replicas of the other videos are deleted in order to place the new load.[20] The above steps are followed when the expected load on a video object both increases and decreases. If the expected load decreases, the second phase (i.e., selecting additional devices) is skipped, but the rest of the sequence is executed.

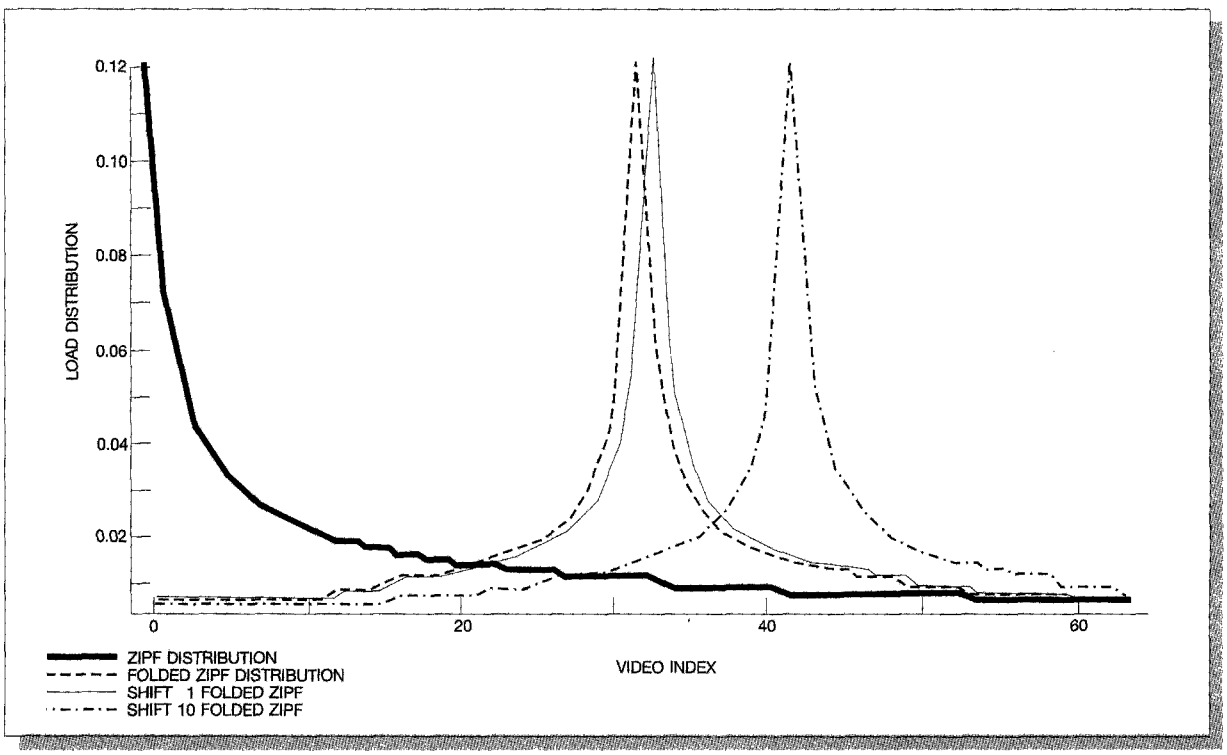
Selection of additional devices has to be made while keeping in mind the available current BW on these devices in order to satisfy the quickness criteria. A parameter to the BSR policy, called the availability parameter, specifies the maximum number of viewers that may be allocated to any replica of any video. This parameter specifies an upper bound on the number of viewers that are lost on failure of a storage device as it may be possible to move some of the viewers to another replica.

If more replicas are required, the second phase of the BSR policy selects additional devices on which to place new replicas.

The BSR policy in the OS/390 LAN Server resource manager is implemented by the RM_SetViewers function. In addition to the VIEWERS command, the INPUT command, which copies video files, may also invoke the RM_SetViewers function. The REMOVE command, which deletes a video file, also invokes the RM_SetViewers function with the expected load for the file set to zero for the purpose of deleting the file. The RM_SetViewers function returns an *action list*, which is a list of devices on which replicas are to be created or deleted. The actual creation and deletion of replicas is carried out by other components of OS/390 LAN Server.

Experimental study of BSR policy. The effectiveness of the BSR policy was studied by simulation in Reference 19. In the following, two of the experiments are described. The first experiment studies the effectiveness of the BSR policy in placing videos in a system that is initially empty. Since the BSR policy is intended to be an on-line policy that can change the placement of the videos in response to changing load, the second experiment reports on the ability of the BSR policy to adapt to changing load. The ef-

Figure 6 Video access distribution



fectiveness of the BSR policy was evaluated by considering the utilization of space and expected BW after placement. A high utilization would be indicative of good performance of the policy.

The simulated system for studying the BSR policy was assumed to have four striping groups with space capacity of 16 GB (gigabytes), 32 GB, 64 GB, and 128 GB. The striping groups had a BW capable of supporting 50, 100, 200, and 400 streams, respectively. Though the striping groups have differing storage capacities and BWs, their BSR is the same. Experiments using striping groups with different BSRs are reported in Reference 19. It was assumed that there were 64 videos, each requiring 3.6 GB, and the total expected load was assumed to be 750 streams. Since there was space for only a single replica of each video, and the total expected load was equal to the total BW of the system, this configuration resulted in a stress test for the BSR policy. The video access frequencies were assumed to be specified by an empirically derived Zipf distribution[21] as shown by the solid line in Figure 6. Figure 6 also shows the distribution used for simulating a shift in load. If the shift in load is simulated by rotating the Zipf distribution, there would be a large change in access frequency at the ends. To avoid such a change, the Zipf distribution was folded into a symmetric distribution (marked as the folded Zipf distribution), and the symmetric distribution was rotated. To simulate small and large changes in the expected load, the distribution was rotated by 1 and by 10 videos, respectively.

Figure 7 shows the results of using the BSR to place the videos into an initially empty system. Four groups of bars are shown, one for each striping group. The first bar in each set shows the actual capacity of the disk, while the second bar shows the expected load after placement by the BSR policy. It can be seen that for all the disks, the BW utilization is close to 100 percent, indicating the effectiveness of the placement by the BSR policy.

Figure 8 illustrates the ability of the BSR policy to adapt to changes in load. As before, there is one group of bars for each striping group, and the first

Figure 7 Allocated bandwidth

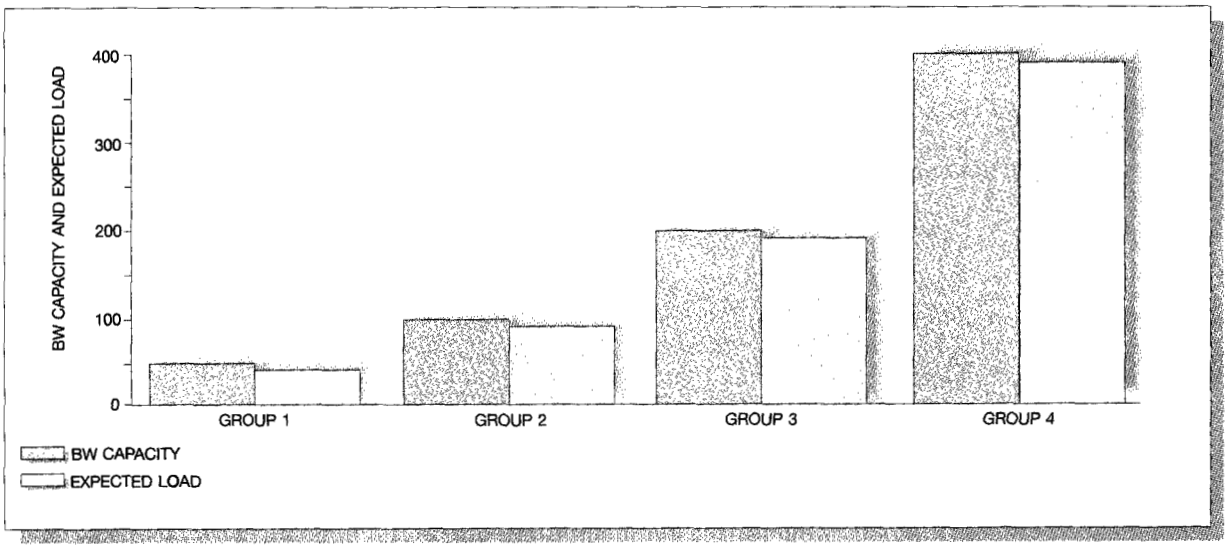
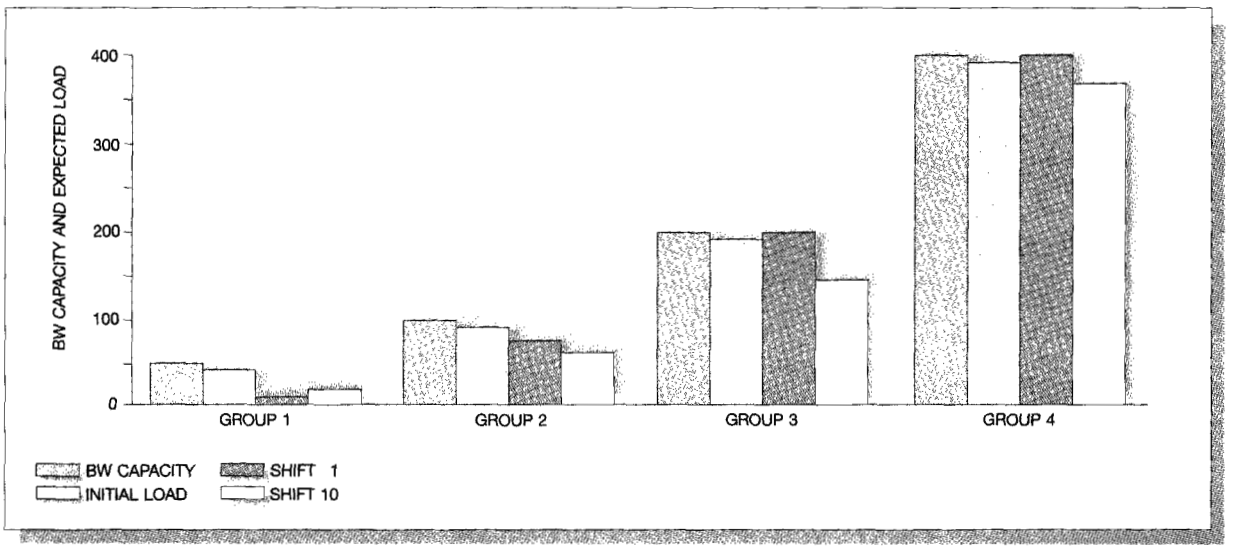


Figure 8 Allocated bandwidth after load change



two bars are the BW of the striping group and the expected load after the initial placement. The third and fourth bars show the expected load after a load shift when the BSR policy was used to change the placement of the videos. In these experiments, the striping groups were assumed to already contain the videos resulting from the initial placement. It can be seen that in both cases the BW utilization is still very high.

Research issues

Multimedia is a relatively new field. Various emerging applications when fully deployed will require

very-large-scale servers. Such large-scale servers will benefit from various performance optimization algorithms (described below, e.g., grouping of requests, multimedia caching) that exploit economies of scale. These optimizations can result in an order-of-magnitude increase in cost-effectiveness. For example, in a large-scale server, multiple concurrent requests for the same video can be served by a single data stream either through synchronizing playback of these requests (referred to as *batching*) or by using small running buffers to cache the intervals between successive streams.

Large-scale servers will be built by linking together multiple systems. This method will require coordination and efficient resource management across these systems. Generally speaking, partitioned systems fragment overall system resources (BW and space) and, hence, result in poor utilization.[22] In a shared system, resources belonging to various systems need to be managed either in a centralized manner or in a coordinated distributed manner, similar to management of accesses to data in a multisystem database system. The interactions across the resource manager components need to be managed carefully, and the number of messages exchanged across systems for resource reservation need to be minimized.

Performance enhancement exploiting economies of scale. Certain activities affect performance, and giving proper consideration to these activities can enhance performance.

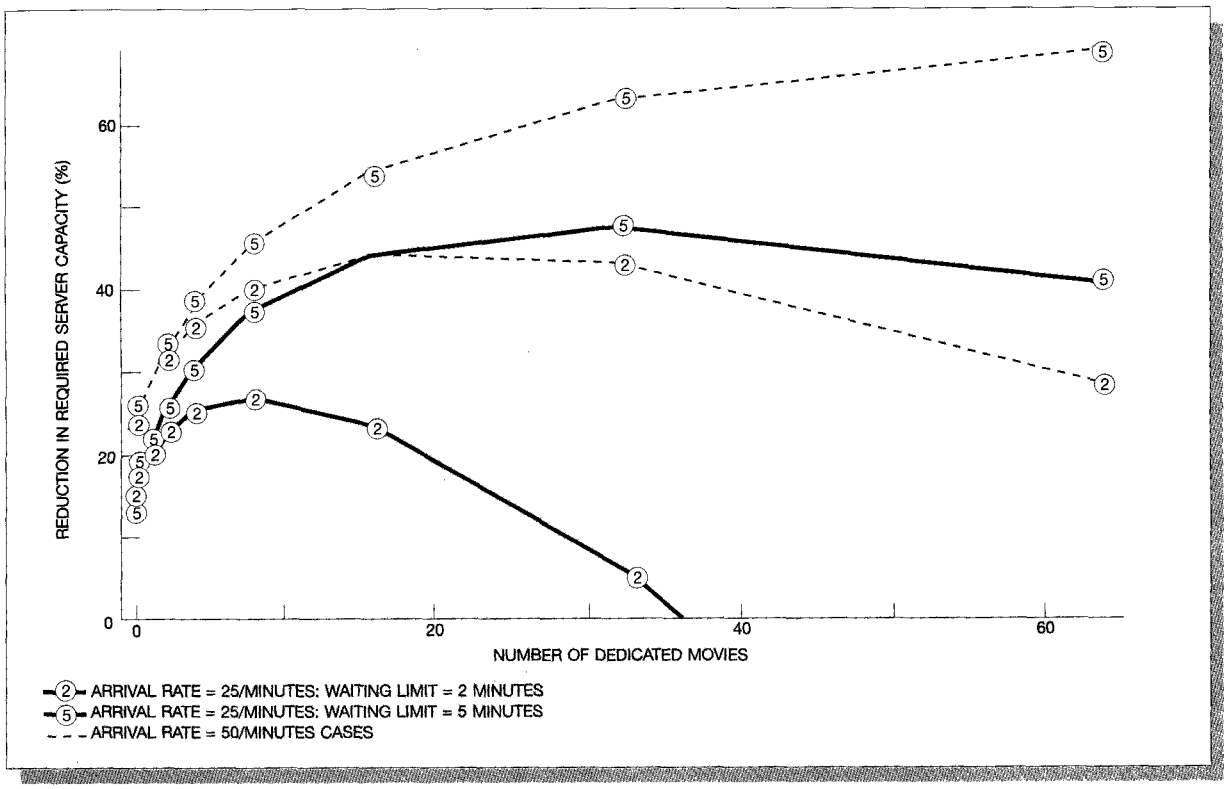
Batching and VCR control. In a large-scale server, multiple requests for the popular videos will arrive within a short time interval. The requests for the long-running videos can be batched together by delaying playback for some of the requests for a short time. The batched requests can be served by a single I/O stream by multicasting the data to the batched clients. Multicasting will result in substantial savings in required server capacity.[21,23] The startup delay can be masked by playing commercials or other short clips. The choices on how long to delay various clients and which video to play are made by the *channel scheduling policy*. The longer startup delay may cause the client to renege. The client behavior can also be influenced by the scheduling policy; for example, clients may wait longer if the startup delay is bounded.[21] Therefore, a scheduling policy should take into account client behavior. However, complex policies that depend upon accurate knowledge of client behavior are not useful in practice.[24] In Reference 21, we have proposed the FCFS-*n* policy that multicasts the *n* most popular videos in regular intervals and serves the remaining requests using FCFS (first-come-first-served) with batching. The policy satisfies all the above criteria and is shown to perform well under a wide variety of client behaviors. In Figure 9, the savings in required server capacity are shown to exceed 70 percent for a VOD (video-on-demand) workload in a moderate size server. See Reference 21 for the details of the workload.

While viewing a video, a client may stop at any point and resume viewing after a pause of random duration. By dedicating server resources to each stream even during a pause, the server can guarantee that the client can resume without delay. This situation will result in significant wastage of server resources particularly if, on average, the client pauses are long. A more efficient alternative is to release server resources upon a pause and reacquire them upon a resume. To assure client satisfaction, the server now needs to ensure that the delay in resuming playback is small.[21,23] A small delay can be achieved by setting aside a small fraction of server capacity (referred to as the *contingency capacity*) for these higher-priority requests.[21,23,25] In Reference 23, we have shown that substantial savings are possible even with frequent VCR (videocassette recorder) activities.

Caching. The need for a continuous delivery guarantee and the larger size of some video objects (e.g., movies) makes traditional caching policies such as LRU (least recently used) unsuitable. The sequential access pattern on these objects can be exploited by caching only the intervals between two successive streams on the same object, i.e., by retaining the pages brought in by a stream for reuse by a closely following stream and subsequently discarding them. In contrast, an interactive workload is composed of many short video clips (e.g., shopping) on which concurrent access will be rare. Hence, caching only the intervals will not be effective. A *generalized interval caching policy* that caches both short video objects as well as intervals or fractions of large video objects is shown to be effective in References 26 and 27. The cost-competitiveness is analyzed in Reference 28. The details of the interactions between the caching policies and the session scheduling are discussed in References 23 and 27.

Multisystem environment. In a multimedia environment with a large number of clients, large-scale servers will be built as clusters of smaller servers. Under random routing, many of the performance optimizations above may not be effective, since requests

Figure 9 Savings in server capacity



for the same video may be scattered over multiple servers. Hence, careful routing of requests to maximize performance is required.[8,22] Additionally, each individual server may have its own resource manager instance for reasons of scalability and performance. Coordination of the resource managers in a distributed environment also poses a challenge.

Affinity routing. In a distributed server environment, the resource manager selects a data server to serve each incoming request and routes the request to the selected server.[8,22,29–31] It also reserves enough resources on the data path from the data server to the client for playback of the stream. The application server is responsible for translating application-specific client requests into data server requests.[14,32] Hence, after establishment of an application session and data path, subsequent queries and access requests for video clips will go directly to the selected application server.

Under random routing, the performance optimizations above may not be effective because of resource fragmentation.[8,22] For example, if successive requests for the same video are routed to different servers and each server has its own cache, intervals between successive requests cannot be formed because of fragmentation of the cache. Affinity routing of requests for the same video will result in reusing the content of a cache and ensuring a good cache hit ratio, as well as for efficient batching of requests for the same videos.[8]

Quota system. Clustered servers are made up of groups of individual servers that are coupled together using fast communication links. They may also share server resources such as disks or communication links to the clients. A centralized resource manager is not suitable for such environments since it is not scalable. From the performance point of view, it may be desirable to have an individual resource manager on each server. For implementing resource reservation, each resource manager has to know the capacity reserved by the others on shared resources. However, it is not feasible for each resource manager to maintain the global state of the resources since it requires

a large amount of message exchange. A protocol is therefore required whereby the individual resource managers can cooperate to share the server resources.

A method, called the *quota system*, is proposed in Reference 33 for resource management in a cluster environment. At initialization, the resource managers partition the free capacity of the shared resources among themselves. When a resource manager receives a request for a new stream, it attempts to satisfy the request using the already allocated capacity for shared resources. If successful, it reserves the required capacity. Otherwise, it requests free resources from the other resource managers. When a stream ends, the released capacity on shared resources is retained by the local resource manager. The above method can be implemented by designating an owner resource manager for each shared resource. The owner partitions the capacity of the shared resource among the resource managers in the cluster and is responsible for responding to requests for additional capacity on the resource.

Conclusions

In multimedia servers, the requirement for continuous and "hiccup-free" delivery poses challenges in addition to the traditional server requirements. Continuous delivery can be guaranteed only by reserving the resources needed to deliver a stream. In this paper, we describe the design and implementation of the resource manager for OS/390 LAN Server, the component that implements this resource reservation. The OS/390 LAN Server evolved from an earlier version that was a traditional high-performance file server for workstation clients.[5] The addition of the resource manager component enabled this file server to support delivery of continuous media and to become a full-fledged multimedia server.

The resource manager for OS/390 LAN Server provides a rich set of functions necessary for administration and operation of a multimedia server. System management functions are provided for defining and dynamically changing the components of the server, as well as measuring and specifying the BW of individual components. Query functions and monitoring via SNMP agents are also supported. The operational functions reserve and release all resources needed to play a video without the need for explicit specification (and hence, knowledge) of the needed resources by the application. Reservation and release are atomic; i.e., either all or none of the resources are reserved or released. The real-time data import functions place data so as to efficiently utilize the storage devices. The import functions are on line, allowing the placement to be dynamically changed, and also support importing data from real-time sources such as digital cameras.

The implementation of the resource manager has been carried out keeping in mind its scalability to large-scale servers. This has been achieved by fine-grain locking that allows concurrent access to data structures. After the code had been ported to an RS/6000 SP environment, it became the base code for resource management in the AIX video server.[14] Emerging applications of the future will require large-scale servers. In the paper, we surveyed various optimizations (e.g., batching, caching) that will allow resource management for such servers to exploit economies of scale. We show that large improvements in cost-performance are possible through such optimizations. Such large-scale servers will consist of clusters of individual servers. Distributed resource management of such clusters poses many challenges. Affinity routing of requests in such an environment can lead to better use of server resources by increasing the effectiveness of some of the optimizations (such as caching). For managing such clusters, a quota system that partitions the resources between the individual node resource managers together with a protocol for renegotiation can be useful.

Acknowledgments

We would like to thank the OS/390 LAN Server development team, especially Pam Conti, Peggy Dixon, Steve Ferrante, John Harter, Tim Krein, Mike Morton, Bill Phillips, Vicki Pritko, and Fred Schwartz of Endicott, Don Jones of Raleigh, and Ray Mansell and Bob Berbec of Hawthorne. We would also like to thank Bill Tetzlaff, Martin Kienzle, Brent Hailpern, and Ambuj Goyal for their support of this work.

Cited references and notes

1. E. A. Fox, "The Coming Revolution in Interactive Digital Video," *Communications of the ACM* **7**, 794–801 (July 1989).
2. W. D. Sincoskie, "System Architecture for a Large Scale Video on Demand Service," *Computer Networks and ISDN Systems* **22**, 155–162 (1991).
3. P. V. Rangan, H. M. Vin, and S. Ramanathan, "Designing

an On-Demand Multimedia Service," *IEEE Communications Magazine* **30**, 56–65 (July 1992).
4. D. P. Anderson, "Metascheduling for Continuous Media," *ACM Transactions on Computer Systems* **11**, No. 3, 226–252 (August 1993).
5. M. Kienzle, R. Berbec, G. Bozman, C. Eilert, M. Eshel, and R. Mansell, *Multimedia File Serving with the OS/390 IBM LAN Server for MVS*, Research Report RC 20432, IBM Corporation, T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598 (April 1996).
6. CPU reservation is not enforced. This would only be meaningful if OS/390 LAN Server were the only workload running on the OS/390 system. OS/390 workload management (WLM) may be used to manage the entire workload on an OS/390 system. WLM allows the customer to specify a high enough velocity for OS/390 LAN Server so that it receives sufficient CPU resources to deliver video successfully.[7] Additionally there are four transaction groups defined for the OS/390 LAN Server that allow internal prioritization of its workload. The customer may assign the multimedia transaction group a higher velocity than the other groups to ensure successful delivery of video.
7. J. Aman, C. Eilert, D. Emmes, P. Yocom, and D. Dillenberger, "Adapative Algorithms for Managing a Distributed Data Processing Workload," *IBM Systems Journal* **36**, No. 2, 242–283 (1997).
8. A. Dan and D. Sitaram, "Multimedia Caching Strategies for Heterogeneous Application and Server Environments," for the special issue on "The State of the Art in Multimedia" of *Multimedia Tools and Applications* **4**, No. 3 (1997).
9. A. Dan, 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 U.S. patent).
10. A. Dan and D. Sitaram, "Resource Allocation for Multimedia Support in a Non-Realtime Environment," *IBM Technical Disclosure Bulletin* **38**, No. 11, 279–280 (1995).
11. *OS/390 IBM LAN Server for MVS Guide*, SC28-1731-00, IBM Corporation; available through IBM branch offices.
12. *OS/390 IBM LAN Server for MVS Configuration Files and Commands*, SC28-1732-00, IBM Corporation; available through IBM branch offices.
13. K. K. Ramakrishnan, L. Vaitzblit, C. Gray, U. Vahalia, D. Ting, P. Tzelnic, S. Glaser, and W. Duso, "Operating System Support for a Video-on-Demand File Service," *Multimedia Systems* **3**, No. 2, 53–65 (May 1995).
14. A. Dan, M. Eshel, J. Hollan, R. Kenneson, M. Kienzle, J. McAssey, R. Rose, D. Sitaram, and W. Tetzlaff, *The Research Server Complex Manager for Large-Scale Multimedia Servers*, Research Report RC 20705, IBM T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598 (1997).
15. We consider data sets consisting of a single disk to be a striping group of size one.
16. Data sets consisting of individual disks may also be calibrated. These are considered striping groups of size one by the resource manager.
17. Currently, this feature is not used.
18. A. Dan, M. Kienzle, and D. Sitaram, "Dynamic Policy of Segment Replication for Load-Balancing in Video-on-Demand Servers," *ACM Multimedia Systems* **3**, No. 3, 93–103 (July 1995).
19. A. Dan and D. Sitaram, "An Online Video Placement Policy Based on Bandwidth to Space Ratio (BSR)," *Proceedings of SIGMOD'95*, San Jose (May 1995), pp. 376–385.
20. In OS/390 LAN Server, the location of replicas of other videos is not available. Hence, the replica consolidation is achieved by deleting unneeded replicas during placement. Alternatively, the VIEWERS command can be invoked for the other videos.
21. A. Dan, D. Sitaram, and P. Shahabuddin, "Dynamic Batching Policies for an On-Demand Video Server" (invited paper), *ACM Multimedia Systems* **4**, No. 3, 112–121 (June 1996).
22. M. Kienzle, A. Dan, D. Sitaram, and W. Tetzlaff, "The Effect of Video Server Topology on Contingency Capacity Requirements," *Multimedia Computing and Networking* (January 1996).
23. A. Dan, P. Shahabuddin, D. Sitaram, and D. Towsley, "Channel Allocation under Batching and VCR Control in Video-on-Demand Servers," *Journal of Parallel and Distributed Computing* **30**, No. 2, 168–179 (November 1995).
24. K. Almeroth, A. Dan, D. Sitaram, and W. Tetzlaff, *Long-Term Channel Allocation Strategies for Video Applications*, Research Report RC 20249, IBM T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598 (1995). Also to appear in the proceedings of the INFOCOM conference in 1997.
25. J. Dey, J. Salehi, J. Kurose, and D. Towsley, "Providing VCR Capabilities in Large-Scale Video Servers," *Proceedings of ACM Multimedia Conference* (1994), pp. 25–32.
26. A. Dan and D. Sitaram, "A Generalized Interval Caching Policy for Mixed Interactive and Long Video Environments," *Multimedia Computing and Networking*, San Jose, CA (January 1996).
27. A. Dan and D. Sitaram, *Buffer Management Policy for an On-Demand Video Server*, Research Report RC 19347, IBM Corporation, T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY (1994).
28. A. Dan, D. Dias, R. Mukherjee, D. Sitaram, and R. Tewari, "Buffering and Caching in Large Scale Video Servers," *Proceedings of IEEE CompCon* (1995), pp. 217–224.
29. "Digital Equipment Corporation Enters Video-on-Demand Market," *Digital Video Server News*, Digital Equipment Corporation, Maynard, MA (October 19, 1993).
30. "nCube Breaks New Ground with Oracle Media Server," *Gartner Group Research Notes* (December 27, 1993).
31. Microsoft Corp., "Microsoft's Software Architecture for Interactive Broadband Networks," *DAVIC/CFP/034* (December 1994).
32. K. DuLac, "Video Pump Interface Recommendation to DAVIC," *DAVIC/CFP/004* (December 1994).
33. A. Dan and D. Sitaram, "Multimedia Resource Management for a Cluster Environment," *IBM Technical Disclosure Bulletin* **38**, No. 11, 501–502 (1995).

Accepted for publication January 21, 1997.

Asit Dan *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598 (electronic mail: asit@watson.ibm.com).* Dr. Dan received the B. Tech. degree in computer science and engineering from the Indian Institute of Technology, Kharagpur, India, and the M.S. and Ph.D. degrees from the University of Massachusetts, Amherst, in computer science and computer engineering, respectively. His doctoral dissertation on "Performance Analysis of Data Sharing Environments" received an Honorable Mention in the 1991 ACM Doctoral Dissertation Competition and was subsequently published by the MIT Press. Since 1990 he has been a research staff member at the Research Center. He has published extensively on the design and analysis of video server as well as transaction processing architectures, and participated jointly with Dr. Sitaram

on the design and development of video servers on various IBM platforms. He holds several top-rated patents in these areas and has received IBM Outstanding Innovation Awards for his work in both areas. His current research interests include Internet-based transaction processing, secure sandbox execution environment, and innovative multimedia applications. Dr. Dan has served on various program committees and is currently serving as a guest editor for the *IBM Journal of Research and Development* and is the tutorial chair for SIGMETRICS '97.

Stephen Dulin *IBM S/390 Division, 522 South Road, Poughkeepsie, New York 12601 (electronic mail: sdulin@vnet.ibm.com).* Mr. Dulin currently is an advisory programmer and has worked on OS/390 LAN Server since January 1995. He joined IBM in 1978 after receiving a B.S. degree from the State University of New York at Albany. His prior experience includes development of microcode for the IBM 3090™ and ES/9000® processors in Kingston and Poughkeepsie, respectively.

Scott Marcotte *IBM S/390 Division, Glendale Programming Laboratory, 1701 North Street, Endicott, New York 13760 (electronic mail: smarcott@vnet.ibm.com).* Mr. Marcotte is currently working on providing Parallel Sysplex support for the Distributed File System for MVS (DFS/MVS). He joined IBM in 1989 and worked in VM/ESA® performance before transferring to LAN File Services (LFS/ESA) development where he worked on the LFS/ESA multimedia projects and also on physical file system performance. He received his B.S. degree in computer science and applied mathematics from the State University of New York at Albany in 1987 and his M.S. degree in computer science from Syracuse University in 1989.

Dinkar Sitaram *Novell Software Development India Pvt. Ltd., 49/1 & 49/3 Garvebhavi Palya, 7th Mile, Hosur Road, Bangalore 560 068, India (electronic mail: sdinkar@Novell.com).* Dr. Sitaram was a research staff member at the IBM Thomas J. Watson Research Center. He received the B. Tech. degree in ECE from the Indian Institute of Technology, Kharagpur, India, and the M.S. and Ph.D. degrees in computer science from the University of Wisconsin-Madison. He has published extensively on multimedia servers, including a book chapter, an outstanding paper, and invited papers. He has worked jointly with Dr. Dan on the design and development of video servers on various IBM platforms. The work in these areas culminated in several top-rated patents. He received an IBM Outstanding Innovation Award and various Invention Achievement Awards for this work. His current research interests include Internet-based multimedia applications and transaction processing. Dr. Sitaram has been serving on the editorial board of the *Journal of High Speed Networks*.

Reprint Order No. G321-5650.