Multimedia Tools and Applications 7, 83–101 (1998)
© 1998 Kluwer Academic Publishers. Manufactured in The Netherlands.

BRAHMA: Browsing and Retrieval Architecture for Hierarchical Multimedia Annotation

ASIT DAN asit@watson.ibm.com
DINKAR SITARAM sitaram@watson.ibm.com
IBM Research Division, T.J. Watson Research Center, Hawthorne, NY 10532

JUNEHWA SONG junesong@cs.umd.edu
University of Maryland, College Park, MD 20742

Abstract. Traditional browsing of large multimedia documents (e.g., video, audio) is primarily sequential. In the absence of an index structure browsing and searching for relevant information in a long video, audio or other multimedia document becomes difficult. Manual annotation can be used to mark various segments of such documents. Different segments can be combined to create new annotated segments, thus creating hierarchical annotation structures. Given the lack of structure in media data, it is natural for different users to have different views on the same media data. Therefore, different users can create different annotation structures. Users may also share some or all of each other's annotation structures. The annotation structure can be browsed or used to playback as a composed video consisting of different segments. Finally, the annotation structures can be manipulated dynamically by different users to alter views on a document. *BRAHMA* is a multimedia environment for browsing and retrieval of multimedia documents based on such hierarchical annotation structures.

Keywords: media annotation, retrieval architecture, hierarchically annotated media, video browsing and viewgroup management

1. Introduction

A picture is worth a thousand words.

Recent advances in computer and communication technology have made it feasible to support complex multimedia applications on each desktop [7–9, 14–16]. Users can compose and share multimedia documents consisting of fragments of media data stored in a large multimedia database [7, 15, 16] (see figure 1). In traditional multimedia applications, large video or audio files are played back sequentially. However, newer applications may require searching and retrieval of relevant fragments from these files. Pure content based browsing which analyzes the files online is limited by the computational overhead of such tasks [5, 11, 13, 17, 18]. Most of the proposed techniques for automatic analysis of video content are based on exhaustive comparisons of several physical features such as color and texture from sequences of video frames, and currently provide a limited set of content information such as shot changes, key frames, similarity of frames, or the existence of certain shapes. These methods require processing of huge amounts of data (half an hour of video data is 45 Mb even in 1.5 Mb/s MPEG 1 compressed format). Extracting more general

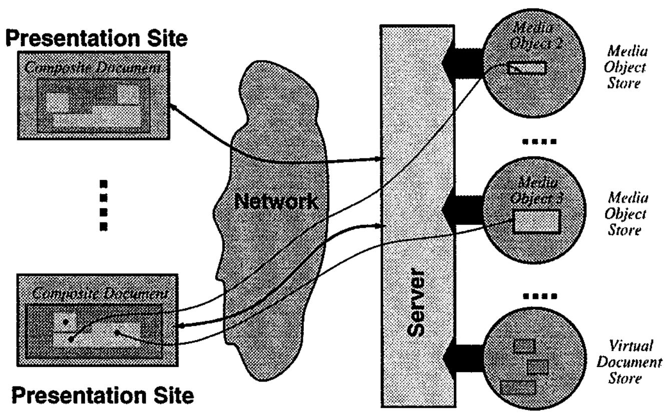


Figure 1. Distributed multimedia environment.

content information that matches high level user perceptions (e.g., appearance or location of a specific person such as Clinton or Gore) is even more computationally intensive. Also, it is difficult to express complex abstractions or semantic information that may be present in a video segment (i.e., across multiple frames). Examples of such useful but complex abstractions are: training videos on a surgical operation or operation of a particular machine, audio signature of Vivaldi's *The Four Seasons*, etc.

Without structural information a large volume of data needs to be retrieved as well as searched based on the content. Manual annotation can be used to mark relevant segments for later retrieval [14, 16]. However, annotation can never capture all the details present in a media file. Therefore, different users may annotate the same or overlapping segments differently with what they see as important. Note that an important property of media data is *semantic continuity of data* which makes possible chunking of an arbitrary portion of a file in a meaningful way. This is unlike text based or other structured information (e.g., contrast a video segment to a set of records in a database).

Multiplicity of views. As the previously mentioned proverb suggests, even a single picture carries a lot of information which cannot be expressed with a few words. Different users may be interested in different abstractions of the same picture frame. Even from an ordinary portrait of a person, one could see many different interpretations of his/her facial expression, physical attributes, fashion consciousness, and the surroundings in which the portrait was composed. The possible number of abstractions across multiple frames is theoretically unbounded. Therefore, it is natural to provide the means for diversity of expressions over the same media content.

Users may also share their views of relevant fragments with others. A user can further annotate existing segments, or create new virtual objects from many existing segments. Consider the following video conferencing example where people from various divisions of a corporation participated in a task-force to bring a new product to the market. Figure 2 shows a part of an example annotation hierarchy. The product to be marketed is composed of two components: Component A and Component B. Each component is again comprised of two

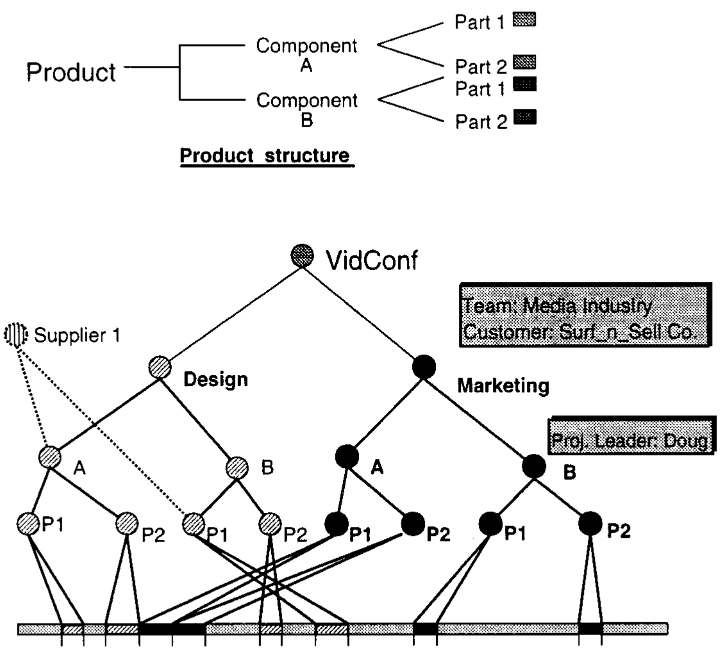


Figure 2. Example annotation hierarchy: video conferencing.

parts, part 1 (P1) and part 2 (P2). The *design* team is more concerned with the schedules of various parts, while the *marketing* team participated in discussing the useful features. Each node in the hierarchy represents a view of a portion of the video stream. For later reviewing, the various teams annotated different segments of the unstructured conference, and created structured views. The view of the *marketing* team is represented by the portions with the thick lines in the graph. The product *design* team has annotated additional relevant segments discussing the schedules. Some of the product components have a dependency on *supplier* 1. Hence, a different view of the video conference is created for sharing with *supplier* 1.

In this paper, we propose the BRAHMA platform for supporting the applications of the type described above. The primary features supported in BRAHMA are dynamic creation, browsing, and sharing across multiple users of annotation hierarchies and playback of composed videos based on such hierarchies. Annotations in BRAHMA associated with a node consist of an *Attribute-Value* pair. Hence, the BRAHMA annotation structure is richer than a flat annotation structure as used in many current systems (e.g., digital library). Both the name of the attribute and the value are arbitrary strings. The hierarchical annotation structure facilitates sharing of views among multiple users and enriches the semantics of the video data. In the above example, *supplier* 1 and *design* teams share only a part of the annotation structure.

The rest of the paper is as follows. Section 2 describes the BRAHMA environment for creation and browsing of annotation structures and playback of composite media objects. Section 3 describes how BRAHMA facilitates sharing of multimedia information across users and maintains annotation structures internally. Section 4 discusses issues related to retrieval of composite media objects and delivery of such objects over the networks. Section 5 provides a summary and conclusions of this paper.

1.1. Contributions and relationship to other work

Video algebra for composition of videos using manually annotated segments is proposed in [16]. In BRAHMA, we extend this framework in several ways. First, BRAHMA accommodates sharing of views and different annotation structures for different users. The users can update these structures dynamically. Second, the attributes are dynamically chosen by users and hence, annotated segments can be identified by other users via *attribute synonym matching*. In contrast, attributes are predefined in currently proposed systems that maintain a database for storing attributes of video segments (e.g., Video Database Browser [14]). Once the database schemas are defined, it is difficult to add new attributes to the database. Additionally, relationships across segments are also stored in the annotation hierarchy. As mentioned above, the UC Berkeley Video Database Browser [14] is implemented using a meta-data database along with query interfaces for searching video segments. Therefore, unlike BRAHMA, it does not explicitly deal with hierarchical annotations and uses only predefined categories of annotations. Hence, their model can be implemented on standard relational database systems (e.g., POSTGRES). Purely text-based annotations that lacks *attribute-value pair* representation and hierarchical structure, limits the capability to query stored annotations. The contrast in the capabilities between the BRAHMA and the pure text based annotation systems is analogous to the contrast between the database systems and the file systems in terms of their query capabilities on stored textual data (e.g., *SQL* vs. *grep*).

A new data model for supporting video data types is proposed in [6]. The data model includes annotation for thematic indexing of video contents. However, the model accommodates annotations only for physical video segments (i.e., a set of frame sequences), but does not support hierarchical annotation structures. The annotation is also based on a predefined set of attributes that are specific to an application. Therefore, as in the case of video meta-data [14], it is hard to define attributes on the fly, and to accommodate the needs of diverse individual users. Using such a data modeling methodology (or the meta-data database) by creating different customized data models for different users will also result in replication of common information across users. More sophisticated schemes that factors out common information across a set of users (assuming such information can be identified a priori) will require (as in BRAHMA) an integrated access control mechanism.

A mathematical framework for multimedia database systems has been proposed in [10]. In their framework, the authors define a media-abstraction scheme on top of physical media representations (audio, video, etc.). The paper further defines a general purpose logical query language and indexing structure which is independent of the media instances. In this framework, a media presentation can be generated from a sequence of queries.

Finally, the nodes in BRAHMA are also different from structured objects with pre-defined attributes and classes, since the node attributes are defined dynamically by users to incorporate the diversity of possible semantics. Creation and maintenance of annotation structure in BRAHMA is similar in many ways to *schema evolution* in object-oriented databases. However, the annotation structure in BRAHMA is not predefined at any given time, whereas in OODB the focus is on adapting to the changing views via schema evolution of once defined object schemas. Other contrasting points are the usage of this structure (i.e., *attribute synonym matching*), frequent changes in annotation structures in BRAHMA and sharing of partial views.

2. BRAHMA user environments

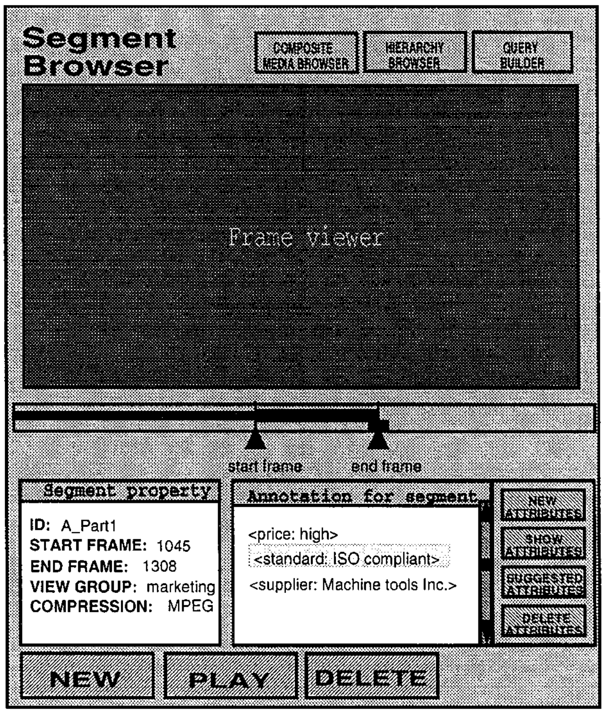
As explained earlier, the applications being considered here will access a large volume of data. Automatic building of relationships across video segments is not suitable due to the high overhead and computational complexity in automatic extraction of features. Therefore, manual annotation of video segments is necessary. Additionally, different viewers may have different perspectives on the same video, and may want to create different manual annotations of the same video. Tools are needed for providing flexibility in creating such annotation structures. Note that manual annotation can be integrated with feature extraction as the lowest level of the structure built on a video.

BRAHMA provides various user interfaces (UI) to annotate a physical segment, to browse and/or update hierarchical annotation structure, to playback composite media objects and to build queries for composition and/or playback of new composite media object views. Multiple windows can be created to access different UIs simultaneously. Each UI window can also toggle between various UI modes.

2.1. Segment browser

The *Segment Browser* can be used to create or playback a physical segment node (figure 3). It also supports creation, modifications or browsing of the associated annotations. The Segment Browser is composed of a *frame viewer*, a *video scroll bar*, a *Segment property* pane and an *Annotation for segment* pane. The video scroll bar provides direct access to any video frame. A user browses a video sequence frame-by-frame using the scroll bar. A new segment is defined by clicking the NEW button and then by marking the starting and ending frames. The various properties (ID, VIEWGROUP, etc.) of this segment are then set on the “Segment property” pane. The annotations associated with this segment are defined in the “Annotation for segment” pane as $\langle$*attribute*, *value*$\rangle$ pairs. When a user selects an existing segment, it shows the currently defined $\langle$*attribute*, *value*$\rangle$ pairs. The properties of the segment as well as its associated annotations can be modified at this time. To aid users in this annotation process, a list of (possibly meaningful) suggested attributes can be loaded into the segment browser environment. Finally, by clicking one of the mode buttons (HIERARCHY BROWSER, COMPOSITE MEDIA BROWSER, QUERY BUILDER) the corresponding windows can be instantiated.

Random access to a MPEG compressed video frame. One of the problems with MPEG compressed video sequences is the difficulty of direct access to a frame in the sequence. This is due to the inter-dependence of frames and predictive coding provided in MPEG. To



ID: A_Part1
START FRAME: 1045
END FRAME: 1308
VIEW GROUP: marketing
COMPRESSION: MPEG

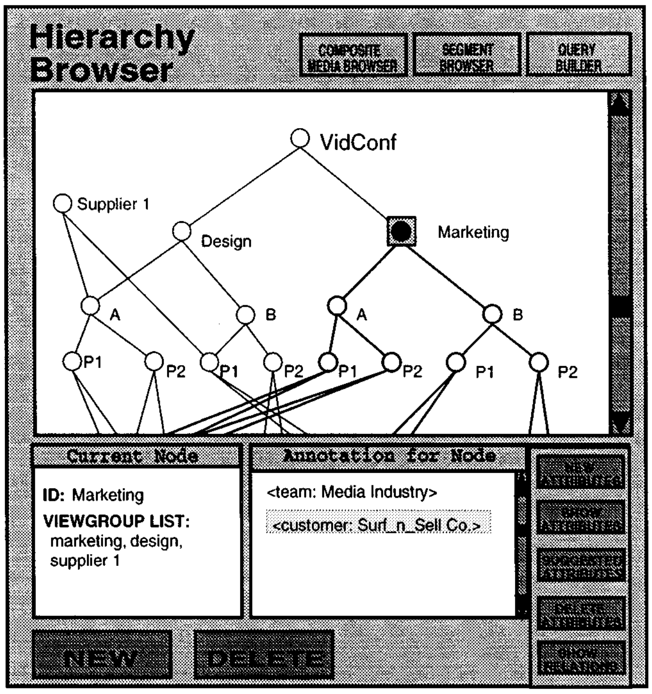
<price: high>
<standard: ISO compliant>
<supplier: Machine tools Inc.>

Figure 3. Creation and browsing of a video segment.

alleviate this problem, we use an associated non-MPEG auxiliary file that can be efficiently browsed frame-by-frame. The auxiliary file contains a reduced image sequence generated from the original MPEG compressed video sequence [17]. The reduced image sequence typically is 1/8 or 1/4 of the original size in both horizontal and vertical dimensions and hence, incurs a low storage overhead. Although it does not generate full resolution video frames, the reduced image sequence provides sufficient resolution to browse and annotate video segments. The generation of a reduced image sequence is very efficient as it is done entirely in the compressed domain.

2.2. Hierarchy browser

The hierarchical annotation structure in BRAHMA can be used to display the annotation hierarchy structure and to select a node (i.e., composite media object) for browsing its annotations. Users can also select a node for playback that results in playback of all associated segments in a pre-defined manner (i.e., according to pre-defined temporal and spatial relationships). A user may also browse the annotations associated with a selected node or navigate the hierarchy structure. Finally, a user can display/update the relationships across the component media objects of the current node.



ID: Marketing
VIEWGROUP LIST:
marketing, design,
supplier 1

<team: Media Industry>
<customer: Surf_n_Sell Co.>

Figure 4. Creation and browsing of an annotation hierarchy.

The creation and browsing of a composite object and its annotations are done using the *Hierarchy Browser* (figure 4). It is composed of panes for a *Hierarchy Navigator*, properties of the selected node (*Current Node*) and its associated annotations (*Annotation for Node*). The scroll bar as well as mouse clicks can be used to navigate the annotation hierarchy. A new node or composite media object is created by clicking the NEW button, and results in the creation of a new node in the "Hierarchy Navigator" pane. Various existing nodes can be linked (via the mouse) and made components of this new node. The properties of this new node are set in the "Current Node" pane. Similarly, properties of a selected existing node are displayed (and can be updated) in the "Current Node" pane. Annotations of the new node are created in the pane named "Annotation for Node". As before, annotations of a selected existing node can be updated in this pane. The "Annotation for Node" pane also allows creation of relationships across component nodes (see below). The various other buttons associated with this pane are self-explanatory.

2.3. *Structural relationships*

Consider an example of three subsequent video segments which are annotated as $\langle$*location* : *Bosnia*$\rangle$ and $\langle$*speaker* : *Clinton*$\rangle$ as shown below. Using structural analysis of the video and using simple reasoning (that can be expressed via video algebra), it can be inferred that Clinton is speaking about Bosnia in the broader video segment.

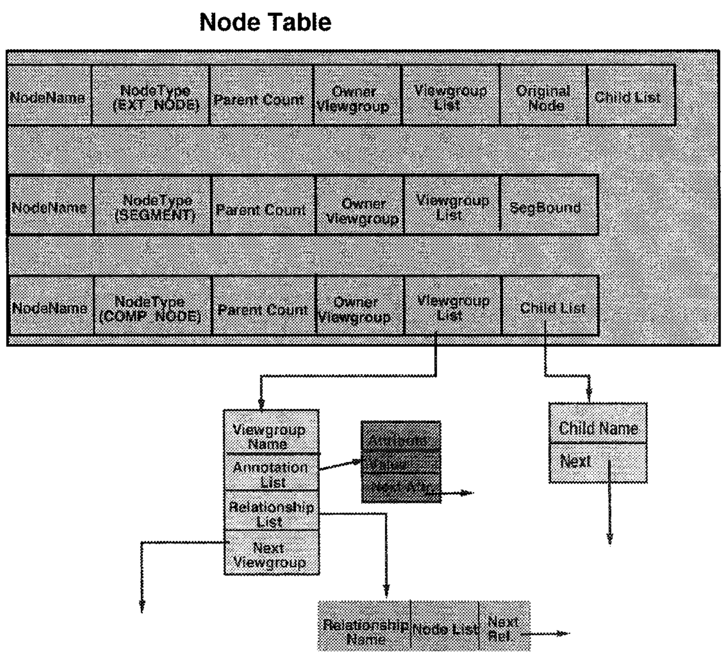


Figure 5. Data structures for maintenance of annotations.

```
        <location:  <speaker:   <location:
         Bosnia>    Clinton>    Bosnia>
-----|----------|----------|-----------
```

From the above example, it is clear that the relationships across various segments or even sub-nodes contain important information. If such relationships are not explicitly stored, they have to be inferred. To avoid high computational overhead, such relationships need to be maintained explicitly. Some relationships may also be impossible to infer (as reasoned earlier for storing explicit attribute annotation). Consider another example of a medical instructional video, where various segments may refer to various phases of an operation. Here, the phases may be semantically related (e.g., opening or closing of a wound). The relationships across sub-nodes are stored as $\langle$*RelationshipName*, *nodelist*$\rangle$ (see figure 5).

2.4. *Composite media browser*

Playback of a composite media object and creation of display relationships amongst its components are described below. In a more elaborated case, a user can also create or display media objects by using a query. For example, a user may wish to retrieve all video segments containing songs by a specific singer, and create a new album with a particular theme and/or order. Video algebra can be used for this purpose [16].

Structural composition. Associated with a node in the annotation hierarchy are display relationships among its component nodes (which themselves may be atomic or composite media objects). When a media object is selected for playback its components are displayed according to these stored relationships. The structural composition includes both temporal and spatial relationships. For temporal relationships, we use Allen's interval algebra [1] which is summarized below.

- *before*(X,Y): Media object X is played before Y.
- *equal*(X,Y): Media objects X and Y are played in parallel.
- *meets*(X,Y): Media object Y starts when object X finishes.
- *overlaps*(X,Y): Media object Y starts while X is being played.
- *during*(X,Y): Media object Y starts after X starts and ends before X ends.
- *starts*(X,Y): Media object X and Y starts at the same time and X ends before Y ends.
- *finishes*(X,Y): Media object Y starts before X starts; X and Y end at the same time.

The spatial relationships can also be represented among media objects as following:

- rightOf (X, Y, d): Media object X is right of Y by distance d.
- below (X, Y, d): Media object X is below Y by distance d.
- topAlign (X, Y), bottomAlign (X, Y), leftAlign (X, Y), rightAlign (X, Y): Media object X and Y are aligned by the top, bottom, left, and right, respectively.

The sizes of individual display windows for component objects can also be specified explicitly.

Interactive browsing. As multimedia data involves visual or audio information, an efficient browsing capability is essential in effective management of multimedia data. This browsing capability needs to provide sufficient interactive capability such as fast forward, fast backward, pause, direct access to a specific frame, etc. The volume of multimedia data, however, makes it difficult to provide all such capabilities with full resolution.

3. Information sharing across users

The hierarchical annotation structure supported by BRAHMA can facilitate sharing of multimedia data as well as collaborative management of multimedia data. It provides a shared view of annotation and composite media objects by defining *viewgroups* for the hierarchy structure.

Viewgroup definition. A user can create a new viewgroup by claiming himself/herself as the owner of the viewgroup. He/she can also include other users and other viewgroups as members of the newly created viewgroup. Each member is granted selective permissions, i.e., browse privilege or browse/modify privilege of the structure and annotations. As a special viewgroup, the owner can include *public* as a member, which gives the access privilege to all users in the system. Each group member has the specified privilege to the annotation hierarchy of BRAHMA. Members with a browse privilege in a viewgroup can query and browse annotations or video segments which belong to the viewgroup whereas members with browse/modify privilege can also create/modify annotations or nodes for the viewgroup. However, maintenance of the membership of a viewgroup, for example, inserting a new group member, is performed by the owner of the viewgroup.

Media object sharing. The owner of an media object can grant permissions to any set of viewgroups during or after the creation of this new node or structure. There are four levels of access permissions granted to viewgroups:

View: The lowest level of access permission. The members of the viewgroup with *View* permission can browse and query over the current node and its subtree. However, no modifications of the annotation or the structure are allowed.

Append/Update: The members are allowed to append annotations to the current node and its subtree or delete appended annotations. However, no structural changes are allowed.

Extend: The members are allowed to append annotations and delete appended annotations to the current node and its subtree. The members are also allowed to Extend a node in the subtree by defining new nodes as children (see figure 7).

Copy: In addition to *Extend* permission, the members are allowed to copy a node and thereby define a new node. The member's viewgroup becomes the owner of the new node.

Note that a created node can only be deleted by its owner viewgroup. The access permissions of a specific member are determined by the combination of the access permissions to the viewgroup and the pre-defined role of the member within the viewgroup. For example, even if a viewgroup is granted an *Extend* permission to a particular node, all the members of the viewgroup with lesser privileges cannot exercise this permission. The owner or the members of the viewgroup with Extend privileges can add children to the specified node. However, the members of the viewgroup with only *Browse* permission can not extend the specified node, but can browse the extended nodes by other members of the same viewgroup.

A single user may belong to multiple viewgroups with different permissions allowed to the user within each viewgroup. During a user session, the user can select a list of viewgroups (out of the viewgroups associated with the user) as the currently selected viewgroup list (referred to as the *Current Viewgroup List*). Hence, this *current viewgroup list* defines the scope of the access and the user's current view of the annotation hierarchy. The access permissions available to a user is the union of the access permissions granted to this user based on its memberships to the viewgroups in the *current viewgroup list*.

During navigation of a structure, only the nodes visible to a user (based on his/her viewgroup memberships) are displayed. Similarly, the query operations are performed only on the structures accessible to a user. With an addition or deletion of a viewgroup to/from the *current viewgroup list*, the permissions and the visibility of the annotation hierarchy change. The system first resolves access permissions granted to an user based on the *current viewgroup list*. It marks the nodes of which the structural views have changed (due to the change in the *current viewgroup list*). The changed views of the nodes in the hierarchy can be instantiated (reflected to the browser and reflected to the query) by clicking the nodes or by a query.

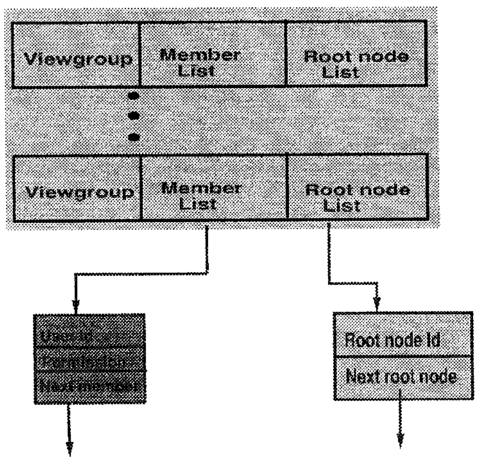


Figure 6. Viewgroup table.

3.1. Maintenance of annotation structure

The structures need to be manipulated dynamically: new nodes may be created, each segment or composed objects may be further annotated by new users, etc. Hence, the internal representation should provide support for efficient dynamic updates, look ups and video compositions. Below we describe the data structures used for maintenence of annotation structures and how these structures are used when creating a new node or modifying other aspects of the annotation structures. The system maintains two tables: the *Node Table* and the *Viewgroup Table* (see figure 6).

Figure 5 shows the Node Table, used to store the annotation structure. The Node Table contains an entry for each defined segment or composite object. There are three different types of nodes maintained in the Node Table:

Segment: The entry for a segment of a video consists of the Node Name, the Node Type (in this case, SEGMENT), a Parent Count which keeps track of the number of parents of a node, the Segment Boundaries, a Owner Viewgroup, and a Viewgroup List. The Segment Boundaries field contains the start and end of the segment together with the identifier of the video to which the segment belongs. The Owner Viewgroup is the viewgroup which created the current node and defines the access permissions to the node. The Viewgroup List contains, by viewgroup, a list of annotations on this segment and a list of relationships between this segment and other segments. Each viewgroup is allowed access to only those annotations and relationships, which are maintained in its Annotation and Relationship Lists. As described earlier, each annotation consists of an Attribute-Value pair. The relationships are stored as the Relationship name (e.g., NextSegment) followed by the list of nodes that have this relationship (e.g., ThisNode, Node2).

Composite node: The entry for a composite node is similar to that of a segment, except that it contains a Child List instead of a Segment Boundaries field. The Child List is a

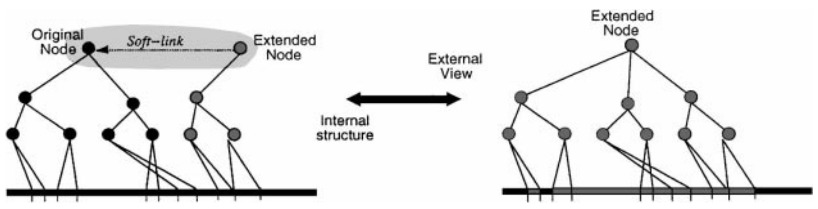


Figure 7. Extension of a node.

list of the nodes that are descendents of this composite node in the annotation subgraph structure.

Extended node: An extended node is a special type of a composite node. The structure of a node can only be changed by the owner viewgroup. However, other viewgroups, when given *extend* permission, can extend the structure of a node by using a soft-links. An extended node has a unique field called *original node* which points the original node from which the current node is extended (see figure 7).

The viewgroup table maintains one entry for each viewgroup that contains points to the list of its members and their permissions as well as the list of the root nodes accessible to the viewgroup. Any browsing and/or modification operations by the members of a viewgroup is started from this viewgroup table entry.

3.2. *Operations on hierarchy*

The operations on the nodes in the annotation hierarchy are:

- Creation/deletion of a node
- Extension of a node
- Insertion/deletion of annotations
- Query using the hierarchy

as well as

- Creation/deletion/maintenance of membership list of a viewgroup
- Insertion/deletion of viewgroups to current viewgroup list

Insertion/deletion of a node can be only done by the owner viewgroup. When a new segment or composite (or extended) node is created, it is inserted into the Node Table, with the various fields in the entry initialized to appropriate values. Once a node is created, the owner can define the viewgroups and their permissions to the node.

3.3. *Composition of objects via query*

A query can be used to prune a subtree(s) rooted at a selected node(s). The query specifies the component objects and sub-objects to be selected based on the annotations and structural relationships used to identify these components. The query language is based on Boolean combinations of simple predicates [16] that test for identifying annotation attributes or relations. As in video algebra, the query operates by recursively examining every sub-node in the selected tree. Selected nodes results in a pruned subtree where only the matched nodes and its ancestors (but not siblings) are retained. The selected and pruned subtree can be made persistent by including it in the hierarchy forest. At each stage, only those attributes and relationships are considered that are visible to the viewgroups to which the user belongs. As explained earlier, the user can further select the scope of a search in the query by specifying a list of viewgroups.

Attribute synonym matching. Without attribute synonym matching, a query searching for selected nodes with a particular attribute will not select nodes annotated with synonym for that attribute. Synonyms include not only a different word with the same meaning, but also different forms of the same word (singular vs. plural, noun vs. verb, etc.) and expressions (e.g., employer vs. employed by). *Attribute synonym matching* selects nodes not only with exact attribute matches, but also with synonym matches. For example, the marketing group may have an attribute named "team" in its annotation while the scheduling group (or a query) may have an attribute "department". Currently, BRAHMA uses a synonym dictionary (customizable by users) for attribute synonym matching.

4. Playback and media delivery

The playback of a composite media object or a simple physical segment will result in retrieval of its component media objects. The media objects are large and can not be easily transported from client to client. In most environments, the content (e.g., video) data would not be replicated, rather would be stored in centralized server sites (e.g., video server) as shown in figure 1. To avoid jitter the content of the entire media object can be downloaded to the display site. This would incur a substantial storage overhead and/or latency in start up. Alternatively, the data can be delivered isochronously over a bandwidth guaranteed channel [4]. Note that for presentation of composite media objects all the data streams for component objects need to be delivered isochronously [15]. The structural information maintained in BRAHMA can be used to intelligently prefetch and/or to cache content at the client for reuse [9, 15].

Multimedia presentation inherently requires huge amounts of data, and thus, large amounts of system resources along the data paths need to be reserved. Furthermore, several media objects could be presented in parallel over some period of time during the presentation of a composite multimedia object, requiring a lot more resources at the same time instant. Therefore, careful resource reservation and allocation schemes are necessary to improve the system throughput. Below we show how structural information can help increase the efficient retrieval and delivery of media data.

4.1. JINSIL delivery subsystem

The resource reservation and object delivery at each stage of data delivery path is handled by a separate layer called JINSIL [15]. At each stage, JINSIL considers the available resources (BW and buffer) and uses advance reservation, prefetching and reordering of presentation structure (when allowed) to minimize latency and/or improve resource utilization.

Figures 8 and 9 show the interactions of JINSIL with the client and the server component. Upon receiving a presentation request from an user, the *composite media player* invokes JINSIL to test the feasibility of the presentation under the client buffer size and the bandwidth limitation. If feasible, an *object delivery schedule* is constructed which will be used for playback. The objects may reside at remote sites and hence, resources need to be reserved in each component on the data path. The JINSIL layer in each stage (e.g., client, network, server) will pass an object delivery schedule to the next stage, that will in turn be used by that stage to reserve appropriate resources and to generate a prefetch schedule. The object delivery schedule of a stage is the prefetch schedule of the previous stage.

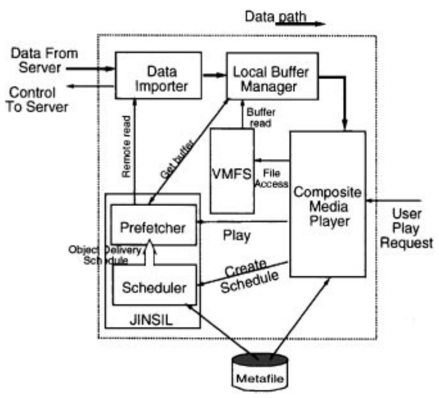


Figure 8. JINSIL interactions in the client system.

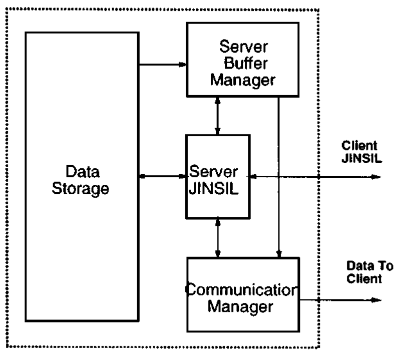


Figure 9. JINSIL interactions in the server system.

Once the process of reservation and creation of object delivery schedules are successful, the composite media player starts playing back the presentation. The composite media player merely informs JINSIL (to start prefetching if in pull mode[1]) and starts accessing the virtual media file system (VMFS). The data importer (in pull mode) prefetches data in the filesystem buffer.

4.1.1. Advance reservation. As mentioned above, the presentation of a composite multimedia object which involves both parallel and sequential playback of media objects results in a time variable data delivery request. In a shared system, the availability of resources such as bandwidth and buffer space also varies with time. One simple resource allocation policy for variable bandwidth requirement is to reserve the maximum bandwidth between the client and server in each component. However, there are several disadvantages to this simple approach. First, in many commercial environments (e.g., cable or phone connection to home) the bandwidth is limited at the final stage of the network. This is referred to as the "Last Mile Problem". Presentation of complex media objects may require multiple streams of video, audio, image or text data for a short duration. Higher up on the data delivery paths the bandwidth may be shared across multiple presentations. Allocating the peak bandwidth reduces the number of presentations that can be admitted.

For the efficient retrieval of media data and for maximizing the utilization of system resources, JINSIL takes into account the true bandwidth requirement of the presentation. In many cases, this true bandwidth requirement is time-dependent (fluctuates with time). Note that in a shared environment, the availability of the system resources in general could be time dependent. Therefore, the resource reservation in JINSIL also reflects this time dependent availability of the resources.

4.1.2. Prefetching. When some buffer space becomes available in a system component, the JINSIL subsystem reshapes the bandwidth requirement and generates a prefetching schedule. Prefetching is motivated by the following three objectives.

- *Presentation satisfiability:* Prefetching makes it possible to satisfy certain presentation requests that cannot be satisfied otherwise due to high instantaneous demands for resources exceeding their availability.
- *Minimize peak bandwidth:* The prefetching scheme tries not only to satisfy the unsatisfiable data request, but also to further minimize the peak value in the reshaped data requirement. This will stabilize the availability of the shared system resources and thereby increase the admissibility of later requests to the system. Note that prefetching with minimum peak bandwidth is done in all (shared or dedicated) system components on the end-to-end data path, resulting in successive smoothing of BW requirements over all the components. Even when enough resources are available (as in a dedicated system component, e.g., the client system), reshaping of data requests still leads to less fluctuations in available resources in the server.
- *Load balancing:* Multimedia objects are often retrieved from different sources. For example, the server storage may be composed of multiple independent devices with separate bandwidths. In such cases, the data path for a presentation of composite multimedia object is partitioned into multiple sub-paths at some stages in the end-to-end delivery.

JINSIL considers the loads on different data paths and tries to reduce possible load imbalance between the data paths by trading off bandwidth and buffer space.

4.1.3. Utilizing flexibilities in presentation request. In many cases, the specification of presentation request could be loosely structured. The temporal relationship across component media objects could be flexibly represented using minimum and maximum bounds. In this case, the schedule for object delivery can be adjusted within the allowed flexibility, depending on the availability of the system resources [2, 7]. In other cases, a user may request a set of media objects without specifying the presentation order among them. For example, a user may ask to retrieve all segments containing songs by a specific singer without specifying a presentation order. JINSIL can then arrange the presentation order and hence, the resulting retrieval order to maximize the system throughput [12].

4.2. Caching

The efficiency of data delivery and resource utilization are further improved by Generalized Interval Caching (GIC) policy [3]. Figure 10 illustrates the operation of the GIC policy. A presentation request from a user in the system may request retrieval of a long video sequence or it may be composed of requests for multiple short media objects, possibly chosen interactively. An interval is defined as the running segment between successive active requests for the same video object. In Movie 1 and Movie 2, several intervals are shown. For the requests to large video objects, GIC caches only small intervals between successive requests to a video sequence. The pages brought in by a preceding request are retained in the cache for reuse by a following stream. After reuse the pages are discarded immediately if the next following request is not served from the cache.

For interactive workloads consisting of small video objects, concurrent accesses are unlikely. A *predictive interval* is defined as the time between successive accesses to the same object (See, Clip 3 and Clip 4, in the figure). GIC still caches the shortest intervals whether the interval encompasses the entire video object or just a segment as in the large video objects. This policy has shown to be superior to the policy of caching the hottest

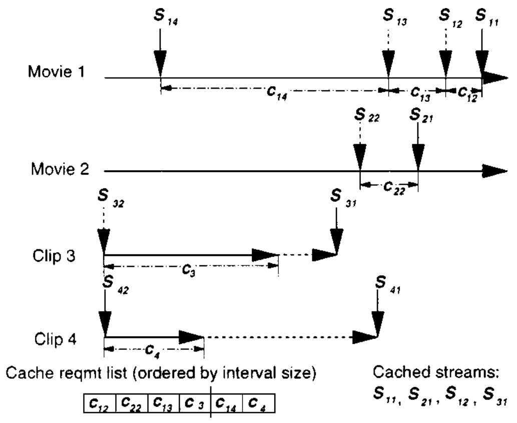


Figure 10. Illustration of Generalized Interval Caching.

movies. In addition, the policy dynamically adapts to the changing frequency of access to different movies unlike the static caching policy.

5. Conclusions

The large volume of data in multimedia databases makes it important to develop efficient methods for searching and retrieving information from such databases. Pure content-based browsing based on online analysis of the video content is limited by the high computational overhead incurred. Additionally, automatic extraction of high-level features (e.g., recognition of a specific person such as Clinton) is difficult. Manual annotation can be used to expand the scope of information associated with the video. However, even manual annotation cannot capture all the details associated with a video segment. Hence, it is necessary to allow each user to annotate the video with the information that is important to that user, and to allow this information to be shared with other users.

In this paper, we propose the BRAHMA system for efficient support of such applications. BRAHMA allows users to dynamically create and annotate a structured view of the videos, as well as share the structure and annotations with other users. For structuring the video, users may organize video segments into hierarchies that represent their view of the video. Each level of the hierarchy can be annotated with an *Attribute-value* pair, providing a richer scheme than a flat annotation structure. Note that in BRAHMA, the attributes that may be associated with the annotations are not pre-defined. The annotation hierarchy may be browsed or queried to locate particular video segments of interest. For efficient retrieval and delivery of the video BRAHMA uses pre-fetching, advance reservation and re-ordering of video segments.

Note

1. The data delivery model could be in pull or push mode. The difference between the two mode is that between any two stages, the object delivery schedule is used either by the source to deliver or by the destination to prefetch.

References

1. J. Allen, “Maintaining knowledge about temporal intervals,” CACM, Vol. 26, No. 11, 1983.
2. K.S. Candan, B. Prabhakaran, and V.S. Subrahmanian, “Retrieval schedules based on resource availability and flexible presentation specifications,” Technical Report, University of Maryland, 1996.
3. A. Dan and D. Sitaram, “Generalized interval caching policy for mixed interactive and long video environments,” Multimedia Computing and Networking, Jan. 1996.
4. A. Dan, D. Sitaram, and P. Shahabuddin, “Dynamic batching policies for an on-demand video server,” ACM Multimedia Systems, 1996.
5. A. Hampapur and H. Harashima, “Production model based digital video segmentation,” Multimedia Tools and Applications, Vol. 1, pp. 9–46, March 1995.
6. R. Hjelsvold and R. Midstraum, “Modelling and querying video data,” in Proceedings of VLDB Conference, 1994, pp. 686–694.
7. M. Kim and J. Song, “Multimedia documents with elastic time,” in ACM Multimedia Conference’95, 1995.
8. M. Kim and J. Song, “Hyperstories: Combining time, space, and asynchrony in multimedia documents,” Technical Report, IBM, RC 19277, 1995.
9. T.D.C. Little and A. Ghafoor, “Multimedia synchronization protocols for broadband integrated services,” IEEE JSAC, Vol. 9, No. 9, pp. 1368–1382, Dec. 1991.

10. S. Marcus and V.S. Subrahmanian, “Foundations of multimedia database systems,” Journal of ACM, Vol. 43, No. 3, pp. 474–523, May 1996.
11. A. Nagasaka and Y. Tanaka, “Automatic video indexing and full-motion search for object appearance,” in IFIP TC2/WG2.6 Second Working Conference on Visual Database Systems, Sept. 1991, pp. 113–127.
12. R.T. Ng and P. Shum, “Optimal clip ordering for news on-demand queries,” in 2nd International Workhop on Multimedia Information Systems, Sept. 1996, pp. 1–5.
13. K. Otsuji, Y. Tonomura, and Y. Ohba, “Video browsing using brightness data,” Visual Communications and Image Processing, pp. 980–989, 1991.
14. R.A. Rowe, J.S. Boreczky, and C.A. Eads, “Indexes for user access to large video databases,” in Storage and Retrieval for Image and Video Databases II, Symposium on Elec. Imaging Sci. and Tech., Feb. 1994.
15. J. Song, A. Dan, and D. Sitaram, “Jinsil: A system for presentation of composite multimedia objects in a distributed environment,” IBM T.J. Watson Research Center, Computer Science Research Report RC 20476, June 1996.
16. R. Weiss, A. Duda, and D.K. Gifford, “Composition and search with a video algebra,” IEEE Multimedia, pp. 12–25, Spring 1995.
17. B.L. Yeo, “Efficient processing of compressed images and video,” Ph.D. Thesis, Princeton University, 1996.
18. H. Zhang, A. Kankanhalli, and S.W. Smoliar, “Automatic partitioning of full motion video,” Multimedia Systems Journal, Vol. 1, pp. 10–28, July 1993.

Asit 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 IBM T.J. Watson Research Center, Yorktown Heights, NY. He has published extensively on the design and analysis of video servers as well as transaction-processing architectures, and is a major contributor to various IBM products in the above areas. He holds several top-rated patents, and has received two IBM Outstanding Innovation Awards, and five IBM Invention Achievement Awards. Currently he is leading a project on the development of COYOTE middleware for supporting long-running transactional applications across autonomous business organizations.

Dr. Dan has served on various program committees, served as the tutorial chair for SIGMETRICS’97 and is currently serving as a guest editor for the IBM Journal of Research and Development.

Dinkar Sitaram is currently a Research Staff Member at the IBM T.J. Watson Research Center, Yorktown Heights, NY. He received the B. Tech. degree in ECE from Indian Institute of Technology, Kharagpur, the M.S. and Ph.D. degrees in Computer Sc. from the University of Wisconsin-Madison.

Dr. Sitaram has published extensively on multimedia servers, including book chapter, Outstannding 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 has received IBM Outstanding Innovation Award and various Invention Achievement Award for this work.

His current research interests include Internet based multimedia applications and transaction processing. Dr. Sitaram is currently serving on the editorial board of the Journal of High Speed Networks.

Junehwa Song is a research staff member at IBM T.J. Watson Research Center. He received his Ph.D. from the Department of Computer Science, University of Maryland at College Park in 1997, his M.S. from State University of New York at Stony Brook 1990, and his B.S. from Seoul National University, Seoul Korea in 1988.

He has been working on various issues in the distributed multimedia systems and compressed video processing. He has been awarded an IBM Cooperative Fellowship from 1995 to 1997.