Management of Media Assets in Large Scale Server Networks

L.-O. Burchard and R. Lüling
Paderborn Center for Parallel Computing
University of Paderborn, Germany

Abstract. In this paper, a management system for the implementation of large scale distributed media streaming applications is presented. The architecture makes use of distributed media servers, each holding a subset of the overall content. Since content management is a key issue, the management software "Distributed Server Management System" (DSMS) is presented, designed to deal with an arbitrarily large number of clients and able to handle a network of distributed servers located at arbitrary nodes in a network. The DSMS automatically replicates media streams, thus achieving that each client connected to the server network can transparently access media streams stored on any of the servers on the network, independent from its physical location. In order to efficiently utilize the network and storage capacities of the available servers, planning algorithms based on simulated annealing heuristics are used to place assets as close to the clients as possible. These algorithms make extensive use of media replication and transcoding of media streams with different bit rates. This paper describes the implementation of the DSMS software as well as an integration into the distributed video information system of a German company.

1 Introduction

In order to implement large scale streaming media applications, e.g. business TV systems in company-wide intranet, a central server for the storage and delivery of broadband media streams keeps the administrative effort low but has the obvious disadvantage of a limited scalability, concerning both network and storage resources. The use of high-quality streams, such as MPEG-2 encoded videos, poses severe constraints on the hardware infrastructure, especially network and storage capacities. Compared to a central server system, a network consisting of several media servers, some of which act as caches, has the advantage that it can be scaled to a large extent and allows to support a large number of widely distributed clients, but has the obvious disadvantage of requiring additional effort for the content management.

The approach described in this paper is to place cache servers next to the clients. These cache servers are connected with each other using common network connections such as the Internet or satellite links. A media server network of this kind can be organized such that one or more media servers are used as cache
servers for a single island (i.e. an environment which provides the necessary network infrastructure for streaming high-quality media assets, e.g. a branch office of a company or a campus network of a university) which store the media assets most likely being accessed from within such an island. Thus, a client has easy and cheap access to any media asset stored on a media server that is located within the same island as the client. In addition to that, clients inside islands have access to the assets stored on the servers located in remote islands or somewhere in the backbone network (see Figure 1). Usually this access imposes heavy load onto the backbone and an access to a remote server is therefore more expensive than a local access.

![Structure of a Media Server Network](image)

*Fig. 1. Structure of a Media Server Network*

Streaming in real-time from a remote server (i.e. a server outside the local island) might not be possible due to insufficient network bandwidth. Therefore, the management software of the server network must provide means for an efficient and reliable distribution of the media assets such that assets which are interesting for a particular client are mapped onto a media server near that client. Otherwise clients which request media assets that cannot be streamed in real-time have to wait until the stream is replicated to the client’s local media server. Such a functionality is also supported by the system presented in this paper, however latencies caused by migration processes to local cache servers must be avoided.
In this paper, we present the mechanisms used by the Distributed Server Management System (DSMS), a server management software designed to handle the administration of large scale media server networks [2]. Besides providing solutions for the previously described mapping problem, the software allows the deletion of media streams on each server of the network, dynamic integration and deletion of media servers. In [2], the architecture of the DSMS was described, mainly focusing on the mechanisms for using differentiated services in order to allow advance reservations of network resources for streaming and file transfers. In this paper, we present extensions of the DSMS that allow to map the media assets onto the server network in a way that efficiently uses the available network and storage resources.

The mapping problem studied here and especially the algorithms presented for the allocation of media assets onto the server network have some relations to the classical 'File Allocation Problem (FAP)'. The FAP is a classical optimization problem but always considers the files that have to be placed onto a network as being of a given and fixed size without allowing to adapt their size as it is the case for media streams handled here. Therefore the problem studied here extends the work on the FAP. Similar mapping problems as studied for the FAP have also been studied for the problem of assigning global variables to a network of processors in an MPP system. As the data items placed in this problem are much smaller as the files in the FAP or the media assets to be mapped in the problem studied in this paper completely other strategies are applied there.

The problem studied in this paper has also been described in [3] and [4]. In these two papers presented by Venkatasubramanian and Ramanathan policies for replication of media assets on distributed server are described. The results presented in this paper extend the work of Venkatasubramanian and Ramanathan.

The DSMS as described in this paper is an integrated system for the provision of broadband media information to a large number of widely distributed clients. This system is currently developed within the framework of the project HiQoS (High Performance Multimedia Services with Quality of Service Guarantees) together with industrial partners.

In the following sections, we will present the structure and functionality of the DSMS, the basic principles for the mapping of media streams onto a network of distributed servers, and the application of the management software in the corporate network of Pixelpark AG, Germany.

2 The Distributed Server Management System, DSMS

3 The DSMS

In this Section, the functionality and architecture of the DSMS is described.

3.1 Functionality

The DSMS organizes a server network in a decentralized way. Media assets can be installed and removed from each of the servers in the network. Once a new
asset is installed on a single server, the DSMS 'publishes' this asset to the rest of the network thus providing access for each client. After that, the new media asset can be distributed in the server network.

Besides the opportunity of dealing with multiple copies of the same media asset introduced due to the requirement for distributing media assets, the DSMS allows a single media asset to be present on the servers of the network with different bit rates or encoding formats. In order to distinguish between the content and the physical copies, each media asset is addressed using a logical and a physical address. The logical address describes a collection of assets with the same content but different properties. The physical address uniquely defines a physical media stream located on one of the servers. The DSMS automatically creates the physical addresses during the storage process and adapts them during the migration. Each client only uses the logical address to request a media asset without having to know the physical locations of the different copies.

An important functionality used to improve the overall QoS for users of the DSMS is to optimize the placement of media assets on the network. A key factor for the perceived QoS of the system is the minimization of asset transmissions, i.e. the aim is to place assets of interest for users on the server located these users, which can be achieved using the algorithms and methods described in Section 4. In order to minimize the requirement for dynamically remapping the assets onto the server network during run-time, the DSMS provides the functionality for clients to subscribe to content categories, which are stored together with each media asset. These categories describe collections of assets with similar content, e.g. news broadcasts. Users can subscribe to content categories which means the DSMS automatically copies newly installed media assets to the local servers where the corresponding categories were subscribed. In case, a user requests assets of a category not subscribed by one of the users in this user’s island, the DSMS copies these assets to the client’s local server.

3.2 System Architecture

The DSMS groups a number of islands of the server network as administrative domains being controlled by modules which are central instances for a single domain. The number of servers within such a domain is generally not restricted and may vary, i.e. during run-time additional server can enter the system and others may be switched off. In addition to that, DSMS modules running on each server node are responsible for the administration of media assets and the communication between different servers (see Figure 2).

The modules running once per domain are the resource management (RM) and a database component. The RM component is usually running on the same node as the database. It is responsible for scheduling the migration and transmission of media files and allows the reservation of bandwidth for streaming media assets in advance as described in [2]. The RM manages the available bandwidth resources and grants or refuses access to the network for streaming traffic. The transmission of media files is also scheduled and initiated by this component. The logical and physical addresses are kept in the database component of the
DSMS, also running as a central service for a single domain. In order to build a
large-scale server network, the central components of different domains can be
connected.

The purpose of the modules running on each server node is the administration
of media assets on the respective node and the physical transmission of media
assets among different servers. The transmission is scheduled and started the
domain central RM module which contacts both transmission modules on the
server nodes involved in order to initiate the transfer of an asset.

The open architecture of the DSMS allows to use the software not only for
managing media servers but also other network services which require to transmit
large amounts of data using different types of networks.

4 Mapping of Media Assets onto Server Network

In this Section, the algorithms used for the mapping of media assets within a sin-
gle domain are presented. In the current implementation, mapping across domain
borders is not supported, however such an extension is rather straightforward
and can be easily implemented.

4.1 Model and Problem Definition

A network of media servers can be modeled as a graph $N = (H, E)$, $|H| = n$,
of $n$ servers connected by a number of communication lines. Each server (host)
has a given capacity that can be used to store media assets. This is represented
by a capacity function $c : H \rightarrow \mathbb{N}$ that assigns an integer number to each host,
representing the MByte of storage capacity of the media server. Each edge $e \in E$
represents a communication line connecting two servers in the server network.
The function $w_e : E \rightarrow \mathbb{N}$ assigns the communication capacity in MBit/sec to
each communication line.

The network of servers stores a set of media assets $M = \{a_1, \ldots, a_m\}$. The
access of user clients (being connected to the media servers) to the media assets
is modeled as a graph $A = (V_A, E_A)$ and $V_A = H \cup M$, $E_A \subseteq \{(v, a) \mid v \in V, a \in M\}$.
Thus, an edge \((v, a) \in E_A\) indicates that a client connected to server \(v\) aims at receiving asset \(a\) from the network of media servers.

Media assets can now be provided in different qualities by the server network to the users. This is possible by encoding the media assets in different bitrates (MBit/sec). The set of suitable bitrates is given by \(B \subseteq N\).

For a given network of servers \(N\), an access pattern \(A\) and a set of bandwidths \(B\), the question is now weather it is feasible to map the media assets in such a way onto the media servers that each access request can be fulfilled. An access request is fulfilled if the host \(u\) requesting a media asset stores this asset directly or a path in the communication network to a host \(v\) hosting the media asset provides sufficient bandwidth to stream the asset to the requesting host \(u\).

The optimization problem is now formalized as follows:

given: A server network \(N = (H, E)\), access structure \(A = (V_A, E_A)\) and a set of possible encoding bitrates \(B\)

Optimization: Find a mapping \(\pi_R : A \rightarrow P(E)\) and \(\pi_b : A \rightarrow B\) with

\[
\pi_R((v, a)) = \{ (v, v_1), (v_1, v_2), \ldots, (v_{l-1}, v_l) \} \text{ for an } l \in \mathbb{N}_0, \pi((v, a)) = v_l
\]

and \(\pi_b((v, a)) = h\)

\[
\sum_{(v, a) \in E_A, \pi((v, a)) = h} \pi_b((v, a)) \leq c(h) \forall h \in H \text{ and}
\]

\[
\sum_{(v, a) \in A, (u_1, u_2) \in \pi_h((v, a))} \pi_b((v, a)) \leq w_*(e) \forall e \in E, e = (u_1, u_2)
\]

and \(\text{QoS} := \sum_{(v, a) \in E_A} \pi_b((v, a)) \rightarrow \max\)

In order to fulfill the access requests, and to optimize the overall QoS, we have a number of tools available. We allow media assets to be mapped redundantly onto different hosts, the encoding bandwidth of media assets can be scaled and therefore the size of a media asset is scalable according to the given bandwidths in \(B\) and the communication lines can be used to stream media assets via the network.

### 4.2 Algorithms

Local search algorithms, e.g. Simulated Annealing, have been shown to be very efficient for the solution of combinatorial optimization problems. A local search algorithm defines a neighborhood relation (graph structure) on the set of all feasible solutions of a given optimization problem. Starting from a randomly chosen feasible solution, a search process is performed on the network of feasible solutions, identifying one node of this graph as the current solution in each iteration. For a closer introduction to local search algorithms see [1].

To construct a simulated annealing algorithm for the solution of the mapping problem, we study a number of basic algorithms. These algorithms define the neighborhood structure of the local search algorithm. All algorithms work in two phases.
Neighborhood structure - Phase 1: For a given configuration \( s \) being a feasible solution of the mapping problem the computation of a neighboring configuration \( s' \) is done in two phases. In a first phase a node in the network of servers is identified and the encoding bitrate of one or more media assets mapped onto this server is increased respectively decreased. This perturbation of the current solution also affects other hosts which access the media asset from the selected server. Thus, in the second phase of the computation of a local neighbor the connection of non-local media assets is assigned.

We present three different algorithms for the perturbation of media assets being stored on a single server. The first algorithm (Figure 3) is rather simple as it selects a random host and media asset and increases the bitrate of the asset to the next available bitrate. After this, random assets on this host are chosen to decrease their bitrates until the overall capacity restriction is met again.

\begin{verbatim}
1 a node \( h \in \{1, \ldots, n\} \) and asset \( a \in M \) are chosen at random (uniform distribution)
2 if \( a \) is mapped onto \( h \) the bitrate is increased to the next possible bitrate
3 if \( a \) is not mapped onto \( h \) then map it with the smallest possible bitrate onto \( h \)
4 while (current load of host \( h \) exceeds \( c[h] \))
5 an asset \( a \) is randomly with uniform distribution
6 if \( a \) is mapped onto \( h \) and \( a \) encoded with the lowest bitrate then delete \( a \) from \( h \)
7 if \( a \) is mapped onto \( h \) decrease the bitrate of \( a \) to the next possible bitrate
\end{verbatim}

**Fig. 3. Algorithm I.1**

The second algorithm presented in Figure 4 also chooses a random host first. An asset is chosen at random from those media assets that are accessed by the chosen host. The bitrate of this asset is increased in a larger step compared to the algorithm I.1. To meet the capacity restrictions assets stored on the server are chosen randomly and their bitrates are decreased. In this algorithm the assets that decrease their bitrates are chosen with exponential distribution giving a higher priority to those assets who are mapped with small bitrates only onto the server. In this way redundant copies which only have smaller bitrates are deleted with higher probability.

\begin{verbatim}
1 a node \( h \in \{1, \ldots, n\} \) is chosen at random with uniform distribution
2 an asset \( a \in M \) and \( (h, a) \in A \) is chosen randomly (uniform distribution)
3 if \( a \) is mapped onto \( h \) with bitrate \( b \), choose \( b' \) random from \([b, \text{max bitrate for } a]\]
4 if \( a \) is not mapped onto \( h \) choose a random bitrate \( b' \) available for \( a \)
5 map \( a \) with bitrate \( b' \) onto node \( h \)
6 while (current load of host \( h \) exceeds \( c[h] \))
7 choose \( a \) at random with exponential distribution according to the bitrate
8 if \( a \) is mapped onto \( h \) and \( a \) encoded with the lowest bitrate then delete \( a \) from \( h \)
9 if \( a \) is mapped onto \( h \) decrease the bitrate of \( a \) to the next possible bitrate
\end{verbatim}

**Fig. 4. Algorithm I.2**

7
The third algorithm (algorithm I.3) used for the first phase follows the same scheme as algorithm I.2 in Figure 4, but takes in line 7 the number of hosts into account that access a media asset stored on the server and not the bitrate of the media asset. In this way, media assets that are accessed by a large number of other remote servers are receiving a decrease of their bitrate or a total deletion with a very small probability only.

\textit{Neighborhood structure - Phase II:} In the second phase of the algorithm to compute a neighboring configuration all edges $(v, a)$ of the access pattern for a host $v \in H$ and media asset $a \in M$ where $a$ is not available on host $v$ are considered. These edges have to be mapped onto the network, i.e. the media stream has to be routed from a host storing $a$ to $v$. We define the following algorithm presented in Figure 5 for this:

\begin{verbatim}
1 for h := 1 to n do
2   for j := 1 to n do bw\_in(j) := 0
3   for all a ∈ M with (h, a) ∈ A and a is not mapped onto h
4     let K be the set of all hosts that store a copy of asset a
5     while not successful and K ≠ ∅ do
6       choose a host h' ∈ K with bitrate of asset a on h' is maximal
7         K := K \ {h'}
8       if bw\_in(h') + bitrate of a on host h' ≤ w_s((h', h)) then
9         successful := true; asset a is streamed from host h' to h
10        if not successful then ‘access to asset a from host h not possible’
\end{verbatim}

\textbf{Fig. 5. Algorithm II.1}

\subsection{4.3 Computing an initial solution}

For a given instance of the mapping problem already the decision problem is NP-complete. Thus, to find an initial solution for a given instance is already an NP-complete problem. We assume therefore in the following, that the storage capacity as well as the network capacity is sufficient to allow for a random mapping of each media asset onto the network of servers. A very simple algorithm places each media asset using the smallest possible bitrate on exactly one host. Then it uses the algorithms for the first phase to map each edge of the access graph onto the server network.

The second algorithm uses the same scheme but increases the bitrate of each asset that is assigned to its maximum and maps additional copies of media assets on hosts that request this asset and still have available storage capacity.

\subsection{4.4 Performance evaluation of algorithms}

In this section the performance of the algorithms presented in the previous section will be investigated in detail. To do this, we define a set of benchmarks in-
stances that reflect the implementation of large scale distributed video archives and their typical access pattern. But before we do this an upper bound for the solution quality is given that is used later on to evaluate the quality of the solutions found by the algorithms.

The achievable quality that can be reached by one host in the network is bounded by the hardware provided to map a sets of media assets on, i.e. the capacity of the hosts and that of its associated communication lines.

If the provided communication bandwidth of the network or the capacity of the hosts cannot be completely used by the access pattern, the quality that can be reached by one host is bounded by the number of requested media assets of this host.

All over all, the following bound is valid for a given network, access pattern and bandwidth:

\[ QoS \leq \sum_{i=1}^{n} \min\{c_i + \sum_{j=1, j\neq i}^{n} w_r((j, i)), \max_{b \in B} b \cdot |\{(i, j) \in E_A\}|\} \]

**Definition of benchmark set** The benchmark instances assumes a clique network for the connection of the host as well as a constant server capacity and bandwidth of the communication edges. Thus, a benchmark instance is defined by the number \( n \) of hosts, the capacity \( c \) of each host and the communication bandwidth \( w_r \) of each communication line, the number of media assets \( m \) and the access pattern. We assume that each asset can be encoded using a number of bandwidths given by the set \( B \).

Table 1 presents the details for the different benchmark classes. All access patterns were randomly generated. The first set of benchmark instances (marked R) are built using access patterns where a host chooses an asset with a given uniform distribution. The second class of access patterns (marked E) use an exponential distribution for the access of media assets. These patterns reflects the case for the implementation of a video library that contains newer and older movies. These kind of access patterns are typical for popular movies and reflect also the access pattern for the information service presented in the next section.

<table>
<thead>
<tr>
<th>Benchmark Class</th>
<th>( n )</th>
<th>( m )</th>
<th>( c )</th>
<th>( w_r )</th>
<th>( B )</th>
<th>access pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>R,32,256,750,100,0.5</td>
<td>32</td>
<td>256</td>
<td>750</td>
<td>100</td>
<td>random</td>
<td>( p = \frac{5}{6} )</td>
</tr>
<tr>
<td>R,32,256,1000,100,0.5</td>
<td>32</td>
<td>256</td>
<td>1000</td>
<td></td>
<td>exponential</td>
<td></td>
</tr>
<tr>
<td>R,32,256,1500,100,0.5</td>
<td>32</td>
<td>256</td>
<td>1500</td>
<td></td>
<td>exponential</td>
<td></td>
</tr>
</tbody>
</table>

**Table 1.** Definition of benchmark instances using a clique network

**Performance results and discussion** Table 2 shows the resulting QoS computed by the simulated annealing algorithms as well as the upper bound for the
benchmark instances defined above. In detail the table shows the results for the
6 different benchmark classes and 6 algorithms and compare the performance to
the upper bound (difference in percent). For each benchmark class 5 instances
were generated that were run 10 times using the (randomized) simulated annealing
algorithms. The values presented in the table are the mean values of the
solutions (QoS) found by the 10 runs of the algorithms.

The measurement results show, that the difference between the upper bound
and the results gained by the simulated annealing algorithm are very small. They
range from about 10 percent down to less than 3 percent. As this is the compari-
tion to an upper bound and not to the optimal solution, it can be concluded
that the algorithms are able to find very good heuristic solutions. Comparing the
different methods presented in the previous section, it can be concluded that the
more intelligent neighborhoods lead to better results than the simple algorithm
1.1 for the first phase. A comparison of the algorithms for the computation of the
initial solution shows that the second algorithm provides better solution quality.

<table>
<thead>
<tr>
<th>Initialization</th>
<th>SA Algorithms</th>
<th>Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 1 1 2 2</td>
<td>1 2 3</td>
</tr>
<tr>
<td>Phase I</td>
<td>1 2 3 1 2</td>
<td>1 2 3</td>
</tr>
</tbody>
</table>
| B.2,256_750_100,0.5 | 108690 112553 113425 109772 116980 118224 | 121200
|               | 13.34 % 9.45 % 8.61 % 12.23 % 5.31 % 4.20 % | |
| B.2,256_1000_100,0.5 | 118265 122540 123348 119329 124992 124875 | 131200
|               | 10.93 % 7.06 % 6.36 % 9.95 % 4.97 % 5.07 % | |
| B.2,256_1500_100,0.5 | 132883 140195 140856 134331 141552 140565 | 147200
|               | 10.77 % 4.99 % 4.50 % 9.58 % 3.99 % 4.72 % | |
| E.2,256_250,100,0.5 | 12390 12563 12433 12690 12930 12754 | 13300
|               | 7.34 % 5.86 % 6.97 % 4.81 % 2.86 % 4.28 % | |

Table 2: Performance Results

5 Application Scenario

The current version of the DSMS is implemented in the corporate network of
Pixelpark AG, Germany. The server network consists of 4 media servers, capable
of streaming MPEG-1 and MPEG-2 encoded videos. These servers are located in
the Pixelpark branch offices in Berlin, Hamburg, Cologne, and Stuttgart. They
are connected with each other by the companies intranet, bandwidths between
2 and 4 MBit/s.

The DSMS is integrated in a web-based information system that provides
access to video streams for both employees and customers or visitors. The infor-
mation system covers events related to the company such as presentations
held in the company or TV broadcasts about the company. The content consists
not only of media stream, but integrates streaming and non-streaming, such as
images or texts etc., content in a single application. The non-streaming content
objects are stored on a HTTP server and displayed synchronously to the media stream. The information system can be accessed both by information terminals spread over the branch offices and common PCs at the desks of the employees.

In order to administrate the DSMS a web-based service has been implemented, providing a simple and easy to use interface to the media streams stored on the network. For users, access to media assets is nearly transparent, only in case media streams

6 Conclusion and Future Work

In this document, we presented the DSMS, its functionality and an application of the DSMS in a corporate network. The DSMS allows to deal with server networks of arbitrary size in a hierarchically structure. We described how the DSMS deals with media streams and the way the mapping of media assets onto the network is handled.

In the current implementation of the DSMS, additional server nodes and the properties (i.e. bandwidth, time of availability etc.) of the underlying network is statically configured in a database. The reservation of bandwidths for the replication of media streams is an important feature, a next version of the DSMS will provide.

In the future, the DSMS system will be extended by an mechanism for advanced reservation of communication lines. This is necessary to make reservations for a secure delivery of a media asset from one server to another. Brokerage mechanisms will be integrated that allow to allocate dynamic prices to the communication lines and all in this way to incorporate market processes in the provision of broadband media to a large number of widely distributed clients.

References