A New Grid Architecture using JMF for Video-on-Demand Applications

L.M.Nithya
Assistant Professor
SNS College of Technology
Coimbatore, Tamil Nadu, India

Dr.A.Shanmugam
Professor & Principal
Bannari Amman Institute of Technology,
Sathyamangalam, Tamil Nadu, India

ABSTRACT
Grid computing was used only in scientific laboratories for many years. Only during the last few years, it is evolving into a business-innovating technology that is driving commercial adoption. In this paper, a Grid-based multimedia application for the distributed tailoring and streaming of media files is proposed. An innovative multimedia data dissemination strategy, based on the Fibonacci series, is used. To respond to the quality-of-service (QoS) requirements, soft real-time applications, a new architecture is presented. Such architecture is able to manage the Grid resource allocation effectively. Technical and practical problems encountered during the development are discussed, and a thorough performance evaluation of the developed prototype is presented. The developed application is found to be better than the existing architectures.

Key words: Grid computing, distributed streaming, multimedia data dissemination.

1. INTRODUCTION
Grid technology [3] [4] is a new paradigm which has the potential to completely change the way of computing and data access. Computational Grids are widely regarded as the next logical step in computing infrastructure, following a path from standalone systems, to tightly linked clusters, to enterprise-wide clusters, to geographically dispersed computing environments. Generally speaking, we could consider the Grid as the new enabling technology to transparently access computing and storage resources anywhere, anytime and with guaranteed Quality of Service. Grid is already being successfully used in many scientific applications where huge amounts of data have to be $D_i = \mathrm{d} - n$ processed and/or stored. Some examples include the processing of data coming from physic-related applications, like the data generated in the nuclear accelerator of CERN [1], the data coming from radio telescopes, the data generated in complex simulations [8] and so on. Other fields, such as Media and Entertainment, can take advantage from Grid adoption in digital rendering, gaming applications, and multimedia data management. In fact, the increasing diffusion of Internet-based multimedia applications and the heterogeneity of client devices call for distributed techniques of multimedia data management which is able to face the enormous storage capacity that is required.

In order to adapt the requested multimedia service to the type of client used, video-on-demand (VOD) systems can have two types of tailoring techniques: the client-side adaptation and the server-side adaptation. Client-side adaptation suffers from resource overload restriction and unnecessary channel overload due to the transmission of high quality media. On the other hand, a scalability issue due to the large number of users affects the server-side adaptation. It is identified that grid computing represents a powerful way to overcome these problems: due to its distributed nature, it is possible to create robust and scalable systems without overloading user devices.

VOD system requires enormous storage capacity, and a centralized solution which has several disadvantages and implies high infrastructure costs [6]. For these reasons, in recent years, the researcher's efforts have been oriented to improve multimedia server performance by designing and developing radically different solutions. The Grid seems to be a good platform for the implementation of distributed VOD systems, ensuring high availability and reliability through redundancy mechanisms which are already implemented at middleware level.

A grid-based multimedia application for the distributed tailoring and streaming of media files is proposed in this paper. In the application, media files are stored across the Grid and split into a variable number of chunks. When a user requests for a streaming, all the chunks composing the media file are recovered and tailored, at provision time, in a parallel way in order to match the client device characteristics and the user preferences. The proposed application is developed on modification of Grid Video [2] to suit the growing requirements of multimedia applications.

The architecture proposed in the paper is totally job oriented and does not require software pre-installation or any modification of the adopted Grid middleware. In order to improve the performance of the proposed solution and to better exploit the Grid computing facilities, some innovative solutions are presented. A new intelligent data dissemination technique based on Fibonacci-based splitting strategy is used. Such technique allows to both reduce the provisioning delay experienced by users and to optimize the resource utilization by reducing the idle time of the computational nodes engaged in the distributed tailoring [2].

The paper aims to show a real experience on enabling a nonscientific application using Grid as the underlying

technology in order to promote such new computing paradigm to a business-oriented audience. The rest of this paper is organized as follows: In section 2, literature review is done, and in section 3, some basic concepts about grids in general and about the middleware are discussed. In section 4, the architecture of the proposed system is described. In section 5, implementation tools are discussed, performance analysis is done in section 6 and in section 7 results and conclusions are given.

2. RELATED LITERATURE

Many types of devices are used today for accessing video contents. As the processing capabilities of such devices are very much limited, a new framework is desired which also meets the huge requirement of server side resources. VOD streaming system allows user to select and watch video content over the internet adapting to the network changes and characteristics of user devices. VOD system is usually a client-server architecture where server stores compressed videos and is connected, through high-speed networks, to geographically distributed clients. The problems of such system are load distribution on the server and content management [6]. On increasing the number of servers the problem of server overload is solved. Beyond this a VOD system requires high storage capacity and a centralized solution. Peer-to-peer (P2P) approaches are analyzed in detail by other researchers [7]. Though P2P system provides high scalability in computation and storage requirements, the dynamic characteristics leads the system to instability [9].

Grid computing developed recently appears to be a suitable methodology for implementation of VOD system to achieve high scalability, availability and reliability. Grid based video systems as discussed in [2]. The existing VOD system consists of the following drawbacks,

- If the server is offline, the user can not view or download the video file.
- If the server load status is high, the user can access the video file only when the load status is minimized.
- The video streaming time is high and also quality of service is less.

The proposed application helps to overcome the drawbacks to a greater extent.

3. GRID ARCHITECTURE AND MIDDLEWARE

3.1 The Virtual Server

In a grid computing environment virtual servers are positioned at different locations. In the streaming phase a proxy, Virtual Server (composed by the VS and the Device Proxy) is able to switch, at the specific time, from one Streaming Server to another. If the user transits from Cell 1 to Cell 2 another VS close to Cell 2 is activated.

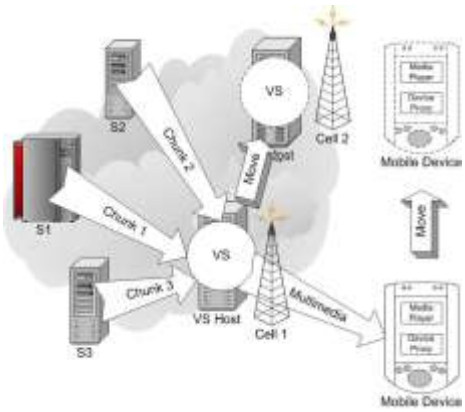

Figure 1 Virtual Server

3.2 The Virtual Organization

VO consisting of many wireless providers (with both wireless 802.x and 3GPP connectivity) use a shared SGMS to allow their users to access multimedia contents. This way, many media objects stored in high quality will be available to each user of the VO without specific device restrictions or mobile/network overload. As depicted in Figure 2, three Organizations (O1, O2, and O3), each representing a provider, share their resources in terms of computational and storage capacity in a VO.

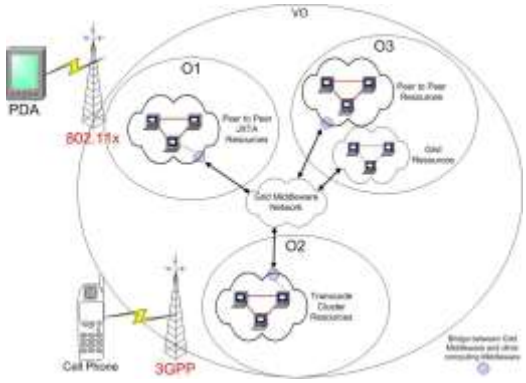

Figure 2 Virtual Organization (VO)

3.3 Storage Resources

Grid technologies are currently distinct from other major technology trends, such as Internet, enterprise, distributed, and peer-to-peer computing, these other trends can benefit significantly from growing into the problem space addressed by Grid technologies.

Computational resources: Mechanisms are required for starting programs and for monitoring and controlling the execution of the resulting processes.

Storage resources: Mechanisms are required for putting and getting files. Third-party and high-performance (e.g., striped) transfers are useful.

Network resources: Management mechanisms that provide control over the resources allocated to network transfers (e.g., prioritization, reservation) can be useful.

Code repositories: This specialized form of storage resource requires mechanisms for managing versioned source and object code.

Catalogs: This specialized form of storage resource requires mechanisms for implementing catalog query and update operations: for example, a relational database [5].

3.4 Grid Middleware

Client sends request to server for video. The request will first go to the middleware, the middleware designed in our system has a status manager which shows the loads of all the current working servers and it will forward the client's request by establishing a TCP connection to the server which has less load and the server will respond by providing the requested video. We can view or download the chunks.

4. ARCHITECTURE OF THE PROPOSED SYSTEM

The proposed application is a distributed video server that allows the provision of user-adapted multimedia contents. The key idea of such application is to use Grid resources both to store high-quality videos and to perform on-the-fly parallel tailoring to adapt the video streaming to user requirements. Omitting details such as registration, secure access, billing, and in order to better illustrate the architecture, it is suitable to restrict the description of the whole application to two main activities referred, respectively, as the multimedia content upload and streaming stages.

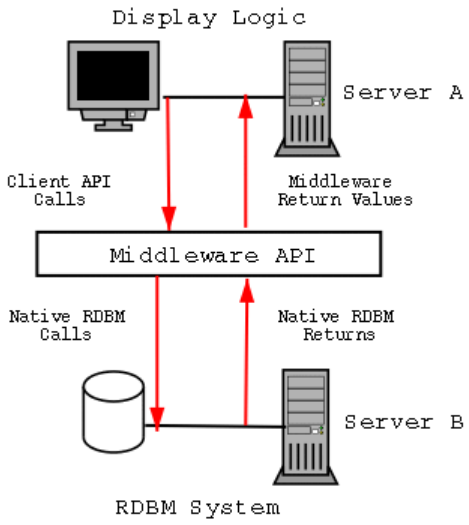

Figure 3 System Architecture

4.1 Multimedia Upload

The Multimedia Upload activity deals with the storage of multimedia contents on the Grid. First of all, the original media file is uploaded in a Grid node and then split (Splitting phase) into many chunks in order to allow the data dissemination and then the parallel tailoring at provision time. In fact, the number of chunks and the adopted splitting strategies influence the system performances in terms, for example, of delays during the streaming. It is then necessary to carefully carry out the splitting according to the available Grid resources and performing, if possible, different chunk splitting of the same media file, thus responding to different QoS requests. Once this phase is completed, each chunk is encoded with the maximum possible quality (Transcoding phase), in order to guarantee regularity between the stored files and to ensure the correct operation of the subsequent activity. Finally, during the Storage phase, each chunk is sent to the Grid storage system and registered on the Replica Catalog. Scalability could be improved by creating more replicas of each chunk and by disseminating them in a geographically distributed manner.

4.2 Multimedia Streaming

The Multimedia Streaming activity deals with end-user requests. Users request a content with specified characteristics (in terms of resolution, color depth, audio reproduction, and media format supported) using a GUI. The different chunks are then recovered and tailored, according to the requested characteristics, by a number of Grid nodes equal to the chunks number (Tailoring phase). Once the first chapter has been tailored, the Streaming phase allows users to watch the entire stream in a transparent way over a set of plain HTTP connections. The media player is not aware of the whole application architecture; in fact, the application automatically composes a playlist that contains the reference to each chunk in the correct order. The HTTP protocol has been chosen because; although the highly distributed nature of the application, the connections based on this protocol guarantee the best behavior with firewalls.

4.3 Tailoring and Profiling

Media tailoring is the phase where each chunk composing the media file is adapted to the user device specific characteristics. This is carried out by using three types of profiles: Device profile, User profile, and Network profile. Profiles are stored as XML files on the user devices. The Device profile holds information related to the screen (such as resolution and color depth), to the audio module (such as the reproduction bandwidth), to the computation such as the maximum frame rate supported), and to the playable formats (MP3, MPEG1, MPEG2, MPEG4, etc.). The User Profile holds the user preferences for receiving content (e.g., the user can choose to hear only the audio track rather than watch only the video track in the case of low link bandwidth). Finally, the Network profile holds information about the bandwidth available to the user (wireless or wired) calculated as the mean bandwidth between the user location and the Resource Broker and by assuming that this represents the bottleneck of the streaming channel. Tailoring actions are carried out by manipulating the media content, with respect to the operational conditions specified in the profiles, in order to adapt it to the user needs; typical tailoring operations are described in Table 1.

Table 1 Tailoring operations of Video files

Tailoring operations	
Transcoding	Video format transcoding
Scaling	Frame rate reduction Resolution reduction
Substitution	Video to image Video to audio Video to text
Removal	Video removal

4.4 Fibonacci-based data dissemination

In fact, the performance of the Grid- Video application is strictly influenced by the strategy adopted to split the media file into several chunks and to disseminate them among Grid nodes. First of all, the number of chunks used for the splitting improves the system performance, allowing the distributed and parallel execution of the tasks needed to the service provisioning. In the following, four different splitting strategies is presented. Let D be the size of the whole media file, n the number of chunks, and D^i the size of the i^{th} chunk, with i = 1; . . . ; n.

Uniform splitting

The size of each chunk is assumed to be the same that is

$$D^i = \frac{D}{n}$$

First-reduced splitting

The size of the first chunk is assumed to be k ways smaller than others that is

$$D^1 = \frac{D}{1+k(n-1)}$$

Linear splitting

The size of the i^{th} chunk is assumed to be i ways greater than the first that is

$$D^j = i * \frac{D}{\sum_{j=1}^{n} j}$$

Fibonacci-based splitting

The size of the i^{th} chunk is assumed to be proportional to the i^{th} Fibonacci number that is

$$D^j = F(i) * \frac{D}{\sum_{i=1}^{n} F(i)}$$

where F(i) indicates the i^{th} term of the Fibonacci series.

However, once the number of chunks is fixed, different performance can be observed concerned to the splitting strategies, which refer to the decisions about the size of each chunk. Splitting strategies influence different aspects of the system performance: the initial delay perceived by user, the service continuity, and the overhead due to the parallel execution, in terms of system efficiency[2]. As analysed and pointed in the paper [2] Fibonacci-based splitting strategy is used in our application for better results.

5. IMPLEMENTATION TOOLS

5.1 Java media framework

The proposed application is developed using Java classes and shell scripts. The Java Media Framework (JMF) is an exciting and versatile API that allows Java developers to process media in many different ways. Almost any type of media manipulation or processing is possible through JMF.

5.2 JMF and data processing model

JMF installed to the proposed machine. Most installers have an option to install native libraries in a system directory; for example, the Windows installer will present an option to "Move DLLs to Windows/System directory." The JMF component architecture is very flexible, and its components can generally be classified in three groups:

- **Input** describes some sort of media that is used as an input to the rest of the process.
- A **process** performs some sort of action on the input. A process has a distinct input and output. A large number of processes are available, and can be applied to an input or a group of inputs. These processes can be chained together so that the output from one process is used as an input to another process. In this manner multiple processes may be applied to an input.
- **Output** describes some sort of destination for media.

Figure 4 illustrates the JMF data processing model and gives some examples of each type.

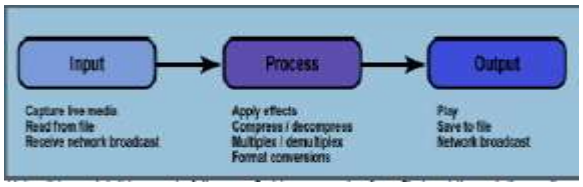

Figure 4 JMF Data Processing model

5.3 Implementing the application

Middleware will act as a intermediate between client and server. All the servers will update his load process to middleware. To upload the video in to the server the middleware splits the video in to several chunks based on the Fibonacci series then upload in to the server using the IP address. In this paper the concept of grid computing is used to split the videos in to several parts and store it in single server whereas in the existing system the entire video is stored in a single server.

This Module maintains the storage of the multimedia contents on the Grid. The original media file is uploaded in a grid node and then split into many chunks in order to allow the data dissemination such that Fibonacci-series and then parallel tailoring at provision time. It is then necessary to carefully carry out the splitting according to the available Grid resources. Each chunk stores on server as mpeg files. Each chunk is sent to the Grid storage system and registered on the Replica Catalog under the logical path. Scalability could be improved by creating more replicas of each chunk. The positioning of chunk is the main part of the application which gives better results than [2].

Client requests a video to middleware then middleware send load status of the servers. According to the servers load status client request video to that particular server and view that video. To view the video file, the client needs to access the middleware and that is done through the IP address of the middleware. The middleware keeps the track of a load status of the several servers connected to it. It identifies the server with lowest load status and provides the access to the client. If the load status of the server connected to the client is increased and crashed due to system failure, then the middleware disconnects the server from the client and give the access to other server having load status.

6. PERFORMANCE ANALYSIS

The performance of the system is analyzed by the study of the average CPU time needed by each Grid node to process its chunk. The CPU time gives an estimation of the computational power needed for the service provisioning. Figure 5 shows the average CPU time per chunk, with respect to the file size and to the number of users and varying the number of chunks. In order to carry out a comparative analysis, we also show the CPU time referred to the client-server streaming architecture, as shown in [2]. In order to study the system scalability, in figure 6, the average CPU time per user is plotted with respect to the number of users for both the client-server and the proposed architectures. Due to load balancing techniques exploited by the Grid system better results are obtained.

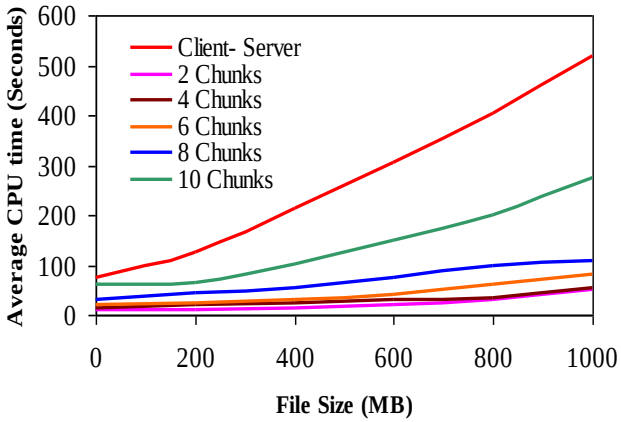

Figure 5 Comparison of CPU time with different number of chunks

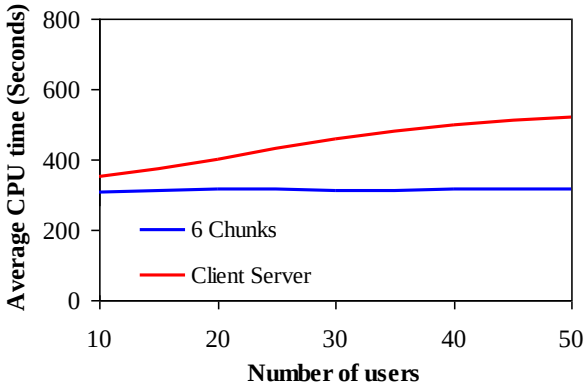

Figure 6 Comparison of CPU time with different number of users for 6 chunks

7. CONCLUSION

A new architecture is developed for effective multimedia storage, adaptation, and delivery, which enables users belonging to a VO to access multimedia services tailored to the real characteristic of the client device in a totally transparent way. In order to improve the system performance, a Fibonacci-based data dissemination strategy and a reservation-based QOS architecture have been proposed. The adoption of such innovative solutions allowed us to both improve the system efficiency and reduce the provisioning delay. Performance evaluations proved the feasibility of the application.

8. REFERENCES

[1] CERN open lab for data grid applications. http://proj-openlab-datagridpublic.web.cern.ch/proj-openlab-datagrid-public/.

[2] Dario Bruneo and Antonio Puliafito, 2009, GridVideo: A Practical Example of Nonscientific application on the Grid, IEEE Transactions on Knowledge and Data Engineering, Vol. 21, No. 5.

[3] Erwin, D.W., and Snelling, D. F, 2001, UNICORE: A Grid computing environment.Lecture Notes in Computer Science, 2150.

[4] Foster, I., 2001, The anatomy of the Grid: Enabling scalable virtual organizations. Lecture Notes in Computer Science, 2150.

[5] Foster, I., Kesselman, C., and Tuecke, S., 2001, The Anatomy of the Grid: Enabling Scalable Virtual Organizations, Int'l J. High Performance Computing Applications, vol. 15, no. 3, pp. 200-222.

[6] Grid Computing Focus Group Software Engineering and Technology Labs,, 2006 Distributed Video-on-Demand—A Grid Based VoD Solution, white paper, http://www.infosys.com/industries/mediaentertainment/ studios-network/white-papers/video-ondemand. pdf.

[7] Liao, X., Jin, H., Ni, L.M., and Deng, D., 2006, Anysee: Peer-to-Peer Live Streaming, Proc. IEEE INFOCOM '06, pp. 1-10.

[8] The datagrid project http://www.datagrid.cnr.it/.

[9] Yiu, W.-P.K. ,Jin, X., and Chan, S.-H.G. , 2007, Challenges and Approaches in Large-Scale P2P Media Streaming, IEEE Multimedia, pp. 50-59.