Amsterdam CineGrid Exchange: A distributed high-quality digital media solution

Knopper, S.; Koning, R.; Roodhart, J.; Grosso, P.; de Laat, C.

Publication date
2009

Document Version
Final published version

Universiteit van Amsterdam

SNE

System and Network Engineering

Amsterdam CineGrid Exchange

A distributed high-quality digital media solution

Sander Knopper,
Ralph Koning, Jeroen Roodhart, Paola Grosso and Cees de Laat

July 30, 2009

Abstract

The SNE group is an active participant in the CineGrid initiative. We built the first integrated CineGrid portal. In this report we describe the rationale behind the architecture of our solution. We provide a component-wise description of design and functionality.

SNE technical report SNE-UVA-2009-01
`http://www.science.uva.nl/research/sne/reports/`

1 Introduction

CineGrid[1] is an organization focused on the distribution of very-high-quality digital media over photonic networks. It does this by stimulating research, development and demonstrations within its community. This community consists of researchers in the academic world but also some companies such as film studios.

The very-high-quality digital media we're currently working with is the 4k standard. 4K offers a resolution of 4096 by 2160 pixels. Table 1 provides an summary of the characteristics of various media formats.

Format	X	Y	Rate (fps)	Color (bits/px)	Frame (px)	Frame (MB)
720p HD	1280	720	60	24	921600	2.8
1080p HD	1920	1080	30	24	2073600	6.2
2k	2048	1080	24 / 48	36	2211840	10
SHD	3840	2160	30	24	8294400	25
4k	4096	2160	24	36	8847360	40

Table 1: Overview of current media formats

Compared to the 1080p HD standard, which is nowadays quite common for home cinema systems, the resolution of 4k is 4 times larger.

Format	Flow (MB/s)	Stream (Gbit/s)
720p HD	170	1.3
1080p HD	190	1.5
2k	240 / 480	1.9 / 3.8
SHD	750	6.0
4k	960	7.6

Table 2: Network requirements of the various media formats

Table 2 reports the network requirements for transport of each of these formats. 4K results in a bandwidth of 7.6 Gbit/s (compared to 1.5 Gbit/s for 1080p); this requires a quite advanced network infrastructure.

2 The CineGrid Exchange

The CineGrid Exchange is the central distribution point in the CineGrid collaboration. The world-wide version is under development. When com-

pleted, it will provide a distributed storage system for digital video content across the world.

The Exchange relies on the existence of metadata associated with the content. These metadata would include the author, title of the video for instance, but also some more technical information like resolution, color depth, etc. This information allows the system to automatically place the video into the storage system and make arrangements for the video to pop up in the user interface. It is possible to store the video in different formats (in terms of resolution and encoding for instance) and on different sites. In the end, a CineGrid user is not aware of the distributed nature of the storage. He would just be browsing on a website searching for content of his liking and - provided that he has the technical means to receive the stream- start displaying the content on his screen.

A major technical requirements for the Exchange is extensibility. For example, different encodings require different media players; mechanisms that allow to add new types of players related to new encodings seamlessly are a must. The playback functionality has to be fully pluggable. The same holds for the storage back-end, so also this part of the system is fully pluggable.

3 Architecture

There are 3 distinct subsystems in the prototype system we've developed:

- the web portal
- the metadata server
- the stream server

Fig.1 shows the overall architecture of our Exchange. A metadata and stream server form a pair within a single streaming server. We can have multiple sites where digital content is publicly available, but there's only a need for a single web portal. The web portal retrieves its data from the metadata servers at the different sites and when playback is requested, it notifies the corresponding streaming server.

The global workflow is as follow:

1. a user browses the portal. The portal contains information about all the available content. The portal has access to a database filled with information it gets from the metadata server.
2. the user selects a movie and decides he wants to display it.

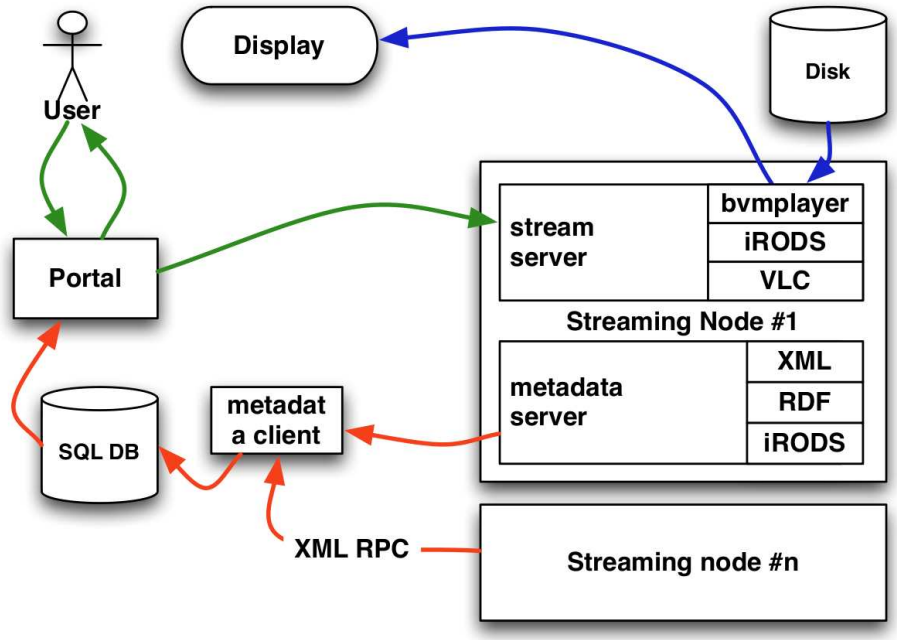

Figure 1: Architectural overview of the Amsterdam CineGrid Exchange

3. the portal communicates with the stream server which will then take care of streaming the content from disk to the display.

In the following sections we give a more detailed overview of the three components.

3.1 Streaming server

The streaming server manages the streaming of content to the display chosen by the user. It supports multiple media players and a queueing algorithm to make sure different media request do not interfere with each other.

Its functionalities are easily accessible by XML-RPC. It provides all the standard functions expected from a streaming server: it can start playing content, it allows to skip to another movie, it can stop playing the current movie and provide the status. This is accomplished by the use of threads.

The whole server is built around the scheduler, which schedules playback for the destinations (displays) available and holds queues. It periodically checks for movies that are finished playing and then frees that object and puts another movie in its place after which it will start that movie. The algorithm behind the scheduler is quite simple, but can easily be changed. It holds

an array of queues indexed by destination, so each destination (display) has its own queue. Within this queue are the player objects, these are objects already holding all the information needed to start playing the movie when the scheduler asks them to. When a new movie is being added, the scheduler checks the destination and looks if there's already a queue available for it. If there is, it simply puts the movie at the end of the queue, otherwise it creates a new queue for this destination and puts the movie in it while immediately giving it a signal to start playing. A periodic check loops through all the queues looking for movies that have stopped playing, while also cleaning up any empty queues.

When the scheduler starts playback for a specific player object, a new thread starts. This thread sets some initial variables, usually the parameters for the media player program, and after that it forks itself. In the forked process it starts the right media player with the correct parameters. At this point the media player takes care of actually displaying the content and the thread will exit eventually. The original process (before the fork) will simply wait for the new forked proces to finish, after which it will set the internal state to "finished playing" which will be noticed by the next periodic check of the scheduler.

Player objects are actually derived classes from the abstract player class. Each derived class is written to work with a single media player. Currently there is a class that works with SAGE[2] (a system used to stream content to tiled displays) and one that works with VLC. Adding support for new media players is relatively easy, simply create a new class derived from the abstract player class and fill in the required methods to properly call the media player executable.

3.2 Metadata server

The metadata server's main goal is to gather all the available information about the content and save it as if it were a database. A very simple one though, as you can't execute complex queries on it, but you can request all the available information.

It consists of 4 main components:

- XML-RPC server, which communicates with the outer world
- metadata database, which holds all data about the content
- metadata consumer, which converts available metadata to an internal format
- file system watcher, which keeps track of file changes

The XML-RPC server main functionality is to send the complete database to other parties. Beside that, it can also be used to retrieve the date of the last update, so another party can figure out whether they need to update or not.

The metadata database is just a thread-safe container for single metadata objects. Basically it's a large array with some added functionality to make sure operations from the XML-RPC server and the file system watcher don't cause data inconsistencies. This is important, since each of them runs in its own thread.

The interface of the metadata consumer is described in an abstract class, this allows for multiple methods of harvesting metadata. Currently there's only one implementation, which is a regular file system based one. In simply reads XML files of a mounted file system and converts the data to the internal representation that will be put into the database.

The file system watcher is an optional but quite useful component. It only works in combination with the file system metadata consumer described above. It watches the file system where the content is placed for modifications. This could be when a new file has been created, an existing file has been modified or a file has been deleted. If such a modification occurs, the file system watcher notifies the metadata consumer to re-read the metadata thus effectively updating the database. The file system watcher also has an abstract interface, currently only an inotify implementation exists for use under GNU/Linux, new implementations are relatively easy to write.

3.3 Web portal

The web portal is the only component the user sees and works with directly. It periodically checks for any updates the metadata servers might have, it then fetches all the metadata from that server and does this for all associated servers. It will then put all information into a local SQL database for fast lookup and support for complex queries, mostly required by the search functionality.

When a user enters the portal it sees the recently added movies along with a search field on top of the page. The user can search for specific keywords in the title of the movie for example, but also filter the output by author or video format.

After choosing a movie, a new page shows all the information about the movie and most importantly, all the formats the movie is available in. Formats are a combination of resolution and encoding, this lets low-bandwidth users also make use of the system for example. When the preferred format is

known, a page where the destination can be supplied shows. At the moment this is a combo box filled with pre-defined values by the administrator. In other words, filled with known displays.

The final step consists of the portal looking up the corresponding streaming server of the movie after which it will send an XML-RPC request to start playback or at least put it in a queue. The user sees a confirmation of what happened.

4 About languages, libraries and frameworks

The metadata and streaming server are completely written in C++, making extensive use of the C++ STL for strings, arrays, queues etc. Both programs come with an autotools-based build system, providing compile-time detection of available libraries. Such as inotify for example.

The SAGE[2] library is used by bvmplayer, a media player to display uncompressed or DXT encoded content.

VLC[3] is a media player, used to display content in all other encodings.

xmlrpc-c is an XML-RPC library for C/C++ which is heavily used in both the metadata and streaming server.

Django[5] is a Python-based web framework which we've used for the web portal.

5 Demonstrations

The preliminary version of this architecture has also been used at Nortel Networks Advanced Technology Summit (ATS) and SuperComputing 2008 (SC08)[6]. The preliminary version consisted of a very early implementation of the streaming server and no metadata server. The metadata had to be entered manually through the admin interface into the SQL database.

At ATS the focus of the demo was to demonstrate streaming very high quality video content using Provider Backbone Transport (PBT)[10] over a transatlantic link from our servers in Amsterdam to a display in Ottawa to test the Quality of Service properties of PBT. We used the Web portal to initiate the video streams and to select the proper display. We got some good results from this.

A week later at SuperComputing 2008 (SC08)[6] the same version of the portal was used as a part of a HPDMnet[12] demo. It involved streaming high quality content from our servers in Amsterdam to a display at the CANARIE[11] booth at SC08. The portal seemed to work very well, however there seemed to be networking problems resulting in major packet loss which could not be resolved during the event.

The architecture in its current form has been demonstrated at the Visualization Symposium in june 2009[13]. This included the metadata server, the new streaming server and an updated version of the web portal. During this demo people could select the video they wanted to see from the portal and stream it to the tiled panel display next to it. This worked very well and there were positive responses on the general look en feel, however when starting a stream people got confused selecting the proper streaming source and destination display. This is was because not all the streaming and display nodes in the list were functional at the time and you also had to be aware of the capabilities of the nodes, therefore this has to be improved or preferably automated in the future.

A poster has also been made about the architecture in its current form. This poster was demonstrated during the poster sessions of the Terena Networking Conference 2009[14].

6 Conclusion and future work

Although there's still quite a lot to be done, the overall foundations that this architecture provides lead to a usable digital content exchange. No matter what direction the CineGrid collaboration decides to go, the architecture

is relatively easy to extend and modify to adhere to new standards and procedures.

Some of the improvements that could be implemented in the future include:

- Persistent database. The metadata server holds the metadata database in-memory currently. This is fast, but could lead to difficulties with large databases.
- Incremental updates. It would be nice if a client can ask for updates in the metadata database since the last update or a specific date. Currently, the metadata server always sends the entire database.
- iRODS integration. iRODS is the Integrated Rule-Oriented Data System [7]. It will manage the storage in the distributed Exchange. The integration with iRODS regards metadata as well as playback.
- Use of NDL. NDL [8] the Network Description Language - is a series of RDF schema that allow better usage of the network,and in particular of circuit switched photonic networks, as the ones used in CineGrid. NDL can be used to determine available bandwith so the scheduler can schedule better.
- Security. XML-RPC connections aren't restricted in any way currently, implement some sort of security system possibly also using iRODS.

References

[1] *CineGrid*
http://www.cinegrid.org

[2] *Scalable Adaptive Graphics Environment*
http://www.evl.uic.edu/cavern/sage/index.php

[3] *VLC, the cross-platform open-source multimedia framework, player and server*
http://www.videolan.org/vlc/

[4] *XML-RPC for C and C++, a lightweight RPC library based on XML and HTTP*
http://xmlrpc-c.sourceforge.net/

[5] *Django, The Web framework for perfectionists with deadlines*
http://www.djangoproject.com/

[6] *SuperComputing 2008*
http://sc08.supercomputing.org/

[7] *iRODS, the Integrated Rule-Oriented Data System* http://www.irods.org

[8] *The Network Description Language* http://www.science.uva.nl/research/sne/ndl

[9] *Nortel Networks* http://www.nortel.com/

[10] *Provider Backbone Transport* http://www.nortel.com/pbt

[11] *CANARIE* http://www.canarie.ca/

[12] *HPDMnet* http://www.hpdmnet.net/

[13] *Mini-symposium: Networked Visualization for e-Science* http://staff.science.uva.nl/ delaat/symp-2009-06-19/

[14] *Terena Networking Conference 2009* http://tnc2009.terena.org/