A grid computing-based architecture for on demand movie rendering

Y. Carpegna, M. Pissardo, B. Montrucchio[*], A. Sanna
{y.carpegna, mario.pissardo, montru, sanna}@polito.it
Politecnico di Torino - DAUIN, c.so Duca degli Abruzzi, 24, I-10129, Torino, Italy

Abstract

High performance distributed computing systems, also referred as computational grids, are becoming more and more important for time consuming applications. Distributed data analysis, as well as data mining and complex numerical simulations, take advantage from grid computing. However industrial grid applications are not of widespread use. In this paper we present an architecture that uses a computational grid for on demand movie rendering. From the user's point of view the grid can be seen as a web application, able to receive frames to be rendered and to give back the final movie. In fact our architecture provides a web interface, simplifying the access to the resources and enabling an industrial application. Moreover all used software is available as open source, showing how an industrial application can be obtained with a low cost of ownership. Finally the proposed architecture matches all security constraints that are required by industries and experimental results show that it can be really used for industrial purposes.

Keywords: Grid computing, Movie rendering, Web applications.

1 Introduction

Rendering high-quality computer animations requires intensive computation and therefore involves a large amount of time. On the other hand, rendering algorithms [1] are suitable candidates to take advantage from parallel and distributed architectures as pixels of a frame are independent and they can be concurrently processed. Moreover, other kinds of parallelism, denoted by means of the term coherence, can often be exploited in order to speed up the rendering process (temporal coherence, frame coherence, data coherence, and so on). For these reasons several works have been presented in order to deliver parallel/distributed rendering architectures.

[*]author for correspondence (montru@polito.it)

A classification of main parallel rendering techniques can be found in [2], and in [3]. Among important and recent parallel rendering projects, it is important to cite, as example, Chromium [4].

There exist also commercial products able to span rendering on a wide network, up to Internet, such as 3D Studio Max [http://www.3dmax.com], Maya [http://www.aliaswavefront.com], and others.
Entropia [http://www.entropia.com] can assemble a large number of idle PCs in people's home and office, in order to build a powerful computational resource. Another important example of possible use of Internet is Butterfly [http://www.butterfly.net]. Butterfly project is an end-to-end solution for on-line game, and it is the first commercial product grid for the video game industry. Butterfly grid (developed with the help of IBM) is able to run multiple games, with millions of gamers, using Globus Project open-source technologies to link servers.
Our prototype tries to address grid computing techniques (like Entropia), but using only open source projects (as partially done in Butterfly), in order to do parallel and distributed rendering, as in [5].

It is also important to define what a grid is. Unfortunately, a unique definition does not exist and, as recently explained by Ian Foster in [6] (see also [7], a grid should be evaluated in terms of applications, scientific results, and delivered business value rather than on its technical architecture. On the other hand, it is well known what a grid must provide. A grid has to be able to provide the user a transparent access to a set of geographically distributed resources by means of a well defined set of services. The term resource is generic and can refer to: computers, storage archives, on-line instruments, and so on. A grid is often tailored to manage a specific kind of resource and/or a specific kind of application; for instance, a collection of (heterogeneous) computers connected by a shared set of services to tackle computational intensive problems is labeled as computational grid [8]. In this paper we propose a solution for ray tracing frame sequences by means of a computational grid. Other kinds of grids can be designed for sharing large data archives (data grids) or for providing a set of services otherwise not available by any single machine such as certain multimedia applications (service grids).

A rigorous and exhaustive grid taxonomy (particularly tailored for computational grids) can be found in [9].

Only recently grid technology and high quality rendering have been considered together; in particular, Sun Microsystems [http://www.sun.com] and Side Effects Software [http://www.sidefx.com] presented a distributed version of the Houdini software during SIGGRAPH 2002 Conference[5].

In this paper we present a high performance distributed rendering system able to compute frame sequences on demand. The system presents an intuitive web interface to the grid that allows the user to submit the scenes to be rendered. Only open source software has been used in this project, and this is the main innovation, since, as previously depicted, there are commercial products for the same purpose, but software is usually patented. The Globus Toolkit [http://www.globus.org][10] has been chosen as grid architecture, while the Pov-Ray ray tracer [http://www.povray.org] is used to render animation frames. We propose a computational grid where a certain number of heterogeneous clusters is coordinated, in order to provide a high computational power. The user submits the sequence and receives back the movie; our architecture transparently manages the scheduling over the clusters where a parallel version of Pov-Ray is installed, the movie composition obtained assembling rendered frames coming back from the clusters, and security/authentication issues.

The paper is organized as follows. Section 2 describes basic ideas that are behind our architecture, and explains in details architectural design; Section 3 depicts implementation issues, and shows how the different components have been joined, while Section 4 gives experimental results and some remarks. Finally conclusions and future work are reported in Section 5.

2 Proposed architecture

A movie rendering architecture is usually implemented by means of proprietary software. An example is, as told in Section 1, 3D Studio MAX. In fact it is often used when the scenes to be rendered are not too complex or do not require particular tools. In such an environment, the rendering phase can be performed on several personal computers in the same time, and the main obstacle is computing time. However it is not possible to share resources that are not in the local area network (LAN), and several other limitations arise. For all these reasons the first basic idea of our architecture (as well as in other architectures, as Entropia) is to build a system that is able to share computational resources among LANs, also allowing to rent such resources to other persons. Security problems and total cost of ownership can be addressed using open source software. Also for rendering an open source solution can be used for several applications. Moreover, the choice we have done of using a grid also allows to create a web application. This web application can be considered local to the farm, when the computational power is provided from one department to another, or global, when different departments of the farm are located in different cities, or, even more important, when computational facilities are provided by other organizations. In this way, using a web application provides two different advantages:

- person that uses computational resources can access to these resources in an easy way;
- there is a complete separation between the computational resource provider and the user; it is also possible to create an organization that gives computational power on demand to its clients.

The possibility of giving computational power on demand to the clients makes very important the presence of the web application, and is the last, but not the least, basic idea. Our architecture shows an example of how grid computing and industrial services can be joined, and our web application is a first step toward a true web service for rendering purposes. You can view a high-level scheme of our architecture in Figure 1. From the user's point of view the rendering machine works on demand. User uploads his/her files on the remote server using a web client, and, when computation is finished, downloads the movie from the remote server, always using web client. As shown in Figure 1, communication is performed by means of a secure protocol (https), and all security issues are considered, since a complete grid infrastructure is used; the computational grid is, from user's point of view, hidden in Internet. Please note that also the payment could be entirely performed using the web client, if the organization providing computational power is not the same that uses resources.

In Figure 2 it is instead reported a view of the computational grid cloud, that in Figure 1 is a part of Internet.

The only input/output point for the grid is the link between the internal web server and web clients. For

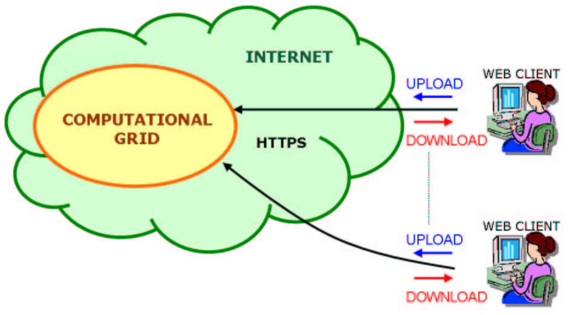


Figure 1: High level scheme of proposed architecture

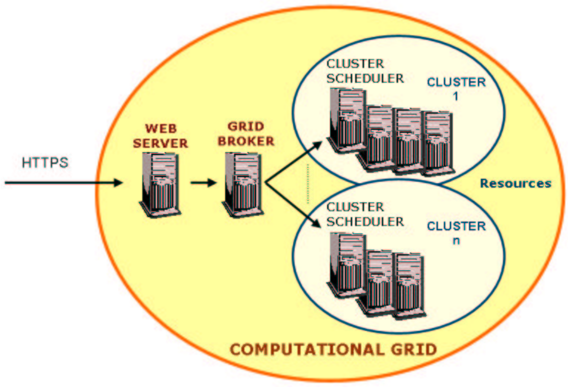


Figure 2: Computational grid internal architecture

sake of simplicity and without lack of generality, we can represent the grid as a unique cloud. In particular, in Figure 2, you can see the functional separation of the parts we have designed for our architecture. At first the web server, that replies to https queries, then the grid scheduler, that manages jobs, sending them to a part of the grid, and finally clusters (resources), each with a cluster scheduler, able to decide how to schedule the job in the cluster. Finally some words on the available resources. Since we want to render many frames of a movie and then generate the movie, and since it is not easy (even there exist techniques for this purpose) to know a priori the complexity of a frame, our architecture will send a certain number of frames for each cluster (at the grid broker level), and this number will depend on the computational power available on free clusters. In particular this means that this architecture requires that the single frames need a significant time to be rendered in order to avoid bottlenecks. However, as depicted in Section 4, this condition is usually satisfied.

3 Implementation

The architecture depicted in Section 3 has been implemented in order to test our choices. In Figure 3 the architecture is detailed.

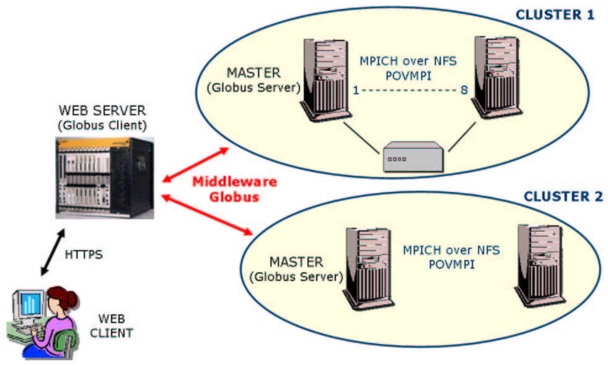
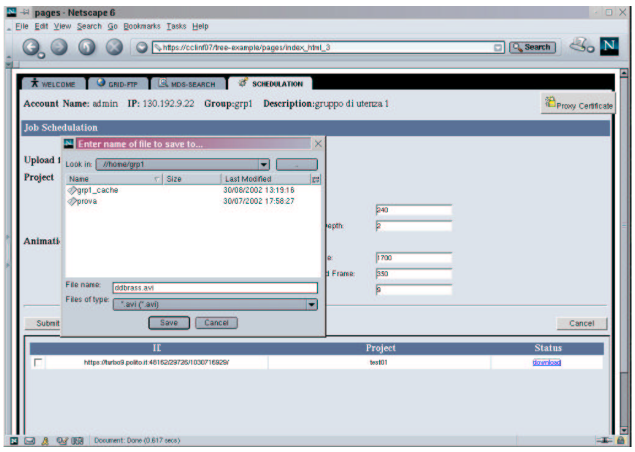


Figure 3: Implemented architecture

Web client (tested on Opera 6 under Windows and Netscape 6 under Linux), is used to upload the files on the web server. Web server (Apache, since it is the most widely spread and fast web server under Linux), as well as Globus client, runs on a high availability server. We have chosen a Flextel Webvision server, because of Web server and Globus client are in the center of the star that controls computing resources and receives inputs from web clients. Also web application database is allocated on Flextel server. This kind of server, produced by Flextel [http://www.flextel.it], is a high availability server. Our server runs on a dual processor (dual Pentium III 800 MHz) board with 1GB of RAM and 36 GB of disk (RAID 0). Operating system is Linux (RedHat 7.2); the choice was straightforward, because Globus client packages (Java CoG) for Windows were not fully compliant with Globus 2 version when our project started. We have chosen Globus (version 2, version 3 is going to be released) since it is the most widely spread, and since it has all security issues required in industrial environments. From the security point of view GSI (Globus Security Infrastructure), that is one of the core services of the grid middleware of Globus, manages accesses and authorizations using a public key infrastructure based on X.509.v3 and SSL(Secure Socket Layer Protocol)/TLS(Transport Layer Security) protocols for authentications. GSI also defines some new extensions for managing proxy certificates, that are very useful in grid computing scenarios. Another of core services of the grid middleware of Globus, GASS (Globus Access to Secondary Storage) is instead used to move remote files, executables, and input/output among remote nodes of the grid. We use GASS to move Pov-Ray files (organized in the unique uploaded compressed file) among grid components. In particular the Grid broker is performed distributing data across various available clusters; this is done depending on computational power available on clusters, since it is not easy to provide an “a priori” estimate of the complexity of the frames. The whole mechanism uses GSI to provide security. On each cluster data are distributed using NFS, and PovMPI (distributed version of Pov-Ray that uses MPI) executes computations. The cluster scheduler function is performed by PovMPI itself (by means of MPI), and MOSIX [http://www.mosix.org]. In fact MOSIX allows process migration and dynamic load balancing after scheduling, even if it has not a queue system. Finally computation is performed (Pov-Ray version is 3.1g, because it is the most recent version having MPI distributed computation patches) on two different clusters. The first cluster is composed, as reported in Figure 3, by eight personal computers, four dual AMD Athlon 1600+ with 512 MB RAM, and four single Athlon 1600+ with 512 MB of RAM; the eight computers are connected each other and to the LAN with a 3Com Gigabit 24 ports switch. The second cluster is composed by two single Pentium III 733 with 256 MB RAM, connected to a Fast Ethernet switch. The two clusters are connected each other, and with the Flextel server, by means of a switched Ethernet LAN (10Mbit/s). This asymmetry is useful to show that the architecture can achieve a good load balancing even when cluster machines are different in terms of computational power and network connection. When all frames have been rendered, our implementation generates a compressed file with all frames, and finally produces final movie using a MPEG-4 free encoder (animmaker 0.2.5 for generating an AVI file and mencoder to encode MPEG-4 file). At this point the user can download the movie, always using services provided by GSI.

As example, in Figure 4 it is reported the download interface (upload interface is specular), that is used at the end of the computation for downloading rendered movie.

We have done the interface using the Zope programming environment [http://www.zope.org]. Since Python is very well integrated with Globus libraries (with a Globus module, CoG (Commodity Grid)), we have chosen Zope because it is entirely written in Python, reducing in this way the computational overhead. Finally the Zope Corporation is one of the companies that is able to grow and make business producing open source software, in particular customizing the software for specific needs. In order to fully understand



Account Name: admin IP: 130.192.9.22 Group:grp1 Description:gruppo di utenza 1

Job Scheduation

Enter name of file to save to...

Look in: //home/grp1

Name	Size	Last Modified
grp1_cache		30/08/2002 13:19:16
prova		30/07/2002 17:58:27

File name: ddbrass.avi
Files of type: *.avi (*.avi)

Upload f
Project
Animati
epth: 2
240
e: 1700
d Frame: 350
9

	II	Project	Status
☐	https://turbo9.polito.it:48162/29726/1030716929/	test01	download

Document: Done (0.617 secs)

Figure 4: Screenshot of download phase

software implementation details, please see Figure 5.

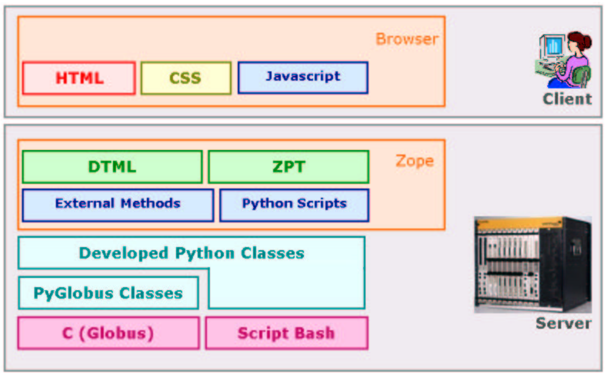


Figure 5: Programming languages used in the project

It is possible to notice how client and server sides have been implemented. The idea was to create an interface for our web application as simple as powerful. This interface makes the interaction easy for the user, hiding all details of the complex grid infrastructure. In the next Section it is possible to see how this simplicity does not prevent our architecture from getting optimal results from the computational point of view.

4 Experimental results and remarks

The implemented architecture has been tested using a quite complex scene. We have got the scene to be rendered from “The Internet Raytracing Competition” [http://www.irtc.org]. In particular we have chosen the scene that won the third place in the January–April 2002 competition, since it was done using Pov-Ray 3.1g (the same version we have used), and sources are freely available. It is called “Brass and Steel” (by Daniel

Dresser [http://www.irtc.org/anims/2002-04-15.html]). In particular we have rendered one hundred frames of this scene, that is quite complex (author says that the whole scene requires about 50 hours for rendering on an Athlon 1200). An example frame of the tested sequence is reported in Figure 6.

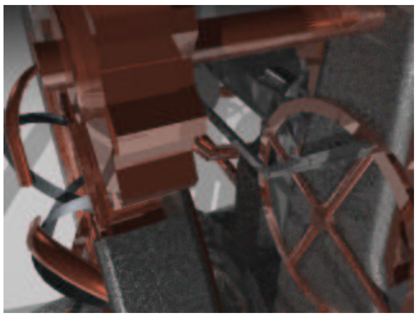


Figure 6: An example frame of the tested sequence

Hardware involved *(type and number of CPUs)*	**Rendering time** *(hh:mm:ss)*
All the local grid *(12xAthlon 1600+ and 2x PIII 733)*	0:16:49
One single Athlon 1600+	3:22:46

Table 1: Experimental results; 100 frames of 320×240; speed-up for the local grid is 12.05, while approximate ratio between the grid and the single PC is 13

The first tested configuration is represented by all available grid hardware. As reported in Section 3, it is formed by two clusters; one is composed from eight personal Athlon-based computer, four of which are dual processor, and the other is composed by two monoprocessor Pentium III 733MHz. We have introduced this lack of symmetry for testing the effects of different hardware in the used grid. Please note that, while in the first cluster connections are with a switched Gigabit Ethernet and in the second cluster the network is a switched Fast Ethernet, the server is connected to the clusters by means of an Ethernet switched network.
The second configuration is based on a single powerful personal computer, usually available to all people which make rendering. It is composed by one single processor Athlon 1600+ personal computer. The approximate ratio between the whole cluster and the single computer is about 13:1, because we can suppose that the two Pentium can substitute one of the Athlon 1600+. From the Table 1 it is possible to compute the speed-up, which is 12.05, and this is quite satisfactory. This means that a simple grid such as the one we have implemented can be very useful if complex rendering has to be done. In fact not only the speed-up factor gives an almost linear behavior, but total time is also reduced in a very significant manner.

The proposed architecture is, at the moment, a prototype. Future work will provide to make tests on a larger number of clusters, in order to address bottlenecks.
The first potential bottleneck is the structure master/slave. Even if in our tests this problem has not arisen, it is possible that with a very large number of frames the architecture will show problem. The solution is to use an approach similar to the Entropia's approach, with several server, and this can be easily done using our open source-based architecture. It is also worth noting that our implementation is based on a high throughput and high availability server, and this reduces this kind of problem. PovMPI is a good parallel renderer, and the main reason for which there is no network bottleneck is that, in ray-tracing, rendering time is usually much longer than transfer time. This shows that the proposed architecture can achieve good results in rendering.
Another problem is load balancing. Since frames are distributed over clusters without estimating "a priori" their complexity, load balancing problems can arise. Once again, in our tests we have not had such problems, but it is a possibility. Future tests will show how this problem is actual. However it is important to note that, in order to estimate frame complexity "a priori", some computational power has to be lost, and the problem is to find what is the trade-off between estimate and successive better load balancing; in particular when a complex architecture like the one depicted is used.
Another feature that has to be underlined is that the single frame must need a significant time to be rendered in order to avoid bottlenecks. However, this problem is not present in usual movie rendering, since the single frame rendering is always well longer than a few seconds. In particular, for tests, we have used a resolution of 320×240, with a rendering time of about two minutes for a single frame. It is also important to note that parameters like these are very common in rendering, and usually resolution and rendering times are much larger.
Last, but not the least, there is the problem of the movie compression, given the rendered frames. Also for this bottleneck, the time required is not significant, at least for rendering purposes like those proposed in this paper. In fact, for the movie under test, compressing time is of less than ten seconds, and this does not constitute a problem for rendering.

Finally, please note that the possibility of using this kind of software for industrial applications, maybe in little rendering farms, can help industry to get wise to the possibilities offered by the new grid technology.

5 Conclusions

High performance distributed computing systems, also referred as computational grids, are becoming more and more important for time consuming applications. We have presented a grid computing-based architecture for on demand movie rendering. It is configured as a web application, and the web interface allows an easy access to the resources and enables industrial applications. Our architecture is also built on open source software, and this is another advantage for an industrial application, since the total cost of ownership is substantially reduced. Anyway, our architecture can be completed using proprietary software, where required performances are greater. Until now this architecture is a prototype, and it has been tested on a little grid; but protocols and software used are able to span computations on wide grids.
Future work will then be aimed to test, and eventually adapt, the architecture on wider grids. Moreover future work will also aim to incorporate in the architecture other rendering engines. The proposed architecture tries to put together classical commercial master-slave architectures, like Entropia, and open source projects (used in other commercial products as Butterfly). This kind of join has not well analyzed until now, and our prototype tries to address problems, in order to show how open source projects can be used for industrial applications.
In conclusion our work is a first step toward the use of open source-based grid computing in actual industrial application, since it matches all security and performance constraints required.

Acknowledgments

This work was partially funded by the project of the Italian MURST (Ministry for University and Scientific Research), “Metodologie e strumenti per la progettazione, lo sviluppo e la gestione delle reti di telecomunicazione avanzate”, and by the CERCOM (Center for Multimedia Radio Communications) of Politecnico di Torino.

References

[1] J.D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes. *Computer Graphics, Principles and Practice.* 2nd ed. In C, Addison-Wesley, 1997.

[2] S. Molnar, M. Cox, D. Ellsworth, H. Fuchs, “A sorting classification of parallel rendering”. *IEEE CG & Appl.* July 1994 **14**(4), pp.23–32.

[3] T. W. Crockett, “An introduction to parallel rendering”. *Parallel Computing* July 1997 **23**(7), pp.819–843.

[4] G. Humphreys et al., “Chromium, a stream-processing framework for interactive rendering on clusters”. *ACM Trans. on Graphics* July 2002 **21**(3), pp.693–702.

[5] Systems/Enterprise. “Side Effects & Sun achieved rendering across a compute farm”. *Grid today* **1**(7), July 2002.

[6] Ian Foster, “What is the grid? A three point checklist”, *Grid today* **1**(6), July 2002.

[7] I. Foster, C. Kesselman, S. Tuecke, “The Anatomy of the Grid: Enabling Scalable Virtual Organizations”. *International Journal of High Performance Computing Applications* **15**(3).

[8] Geoffrey Fox and Dennis Gannon. “Computational grids” *Computing in Science & Engineering* **3**(4), 2001.

[9] K. Krauter, R. Buyya, and M. Maheswaran, “A taxonomy and survey of grid resource management systems for distributed computing”, *Software-Practice and Experience*, **32**(2) pp. 135–164, 2002.

[10] Ian Foster, “The Globus Toolkit for Grid Computing”. *Proceedings of the First IEEE/ACM International Symposium on Cluster Computing and the Grid.* 15–18 May 2001.