Using Network of Workstations to Support a Web-Based Visualization Service

Wilfrid Lefer and Jean-Marc Pierson

Laboratoire d'Informatique du Littoral
LIL, B.P. 719
62228 Calais, France
{lefer,pierson}@lil.univ-littoral.fr

Abstract. Nowadays, huge amount of data can be retrieved thanks to the World Wide Web, raising to the need for new services for online data exploration and analysis. In the other hand, scientific visualization offers varying techniques to represent any kind of data visually, taking advantage of the natural skills of the human brain to analyze complex phenomena through visual representations. These techniques are computationally high demanding and can not be handled by a single machine. A Web-based visualization service would rapidly overload the machine and slow down the Web service dramatically. This paper describes *viscWeb*, a distributed visualization architecture, which allows us to use a pool of workstations connected through Internet as a computational resource for a Web-based visualization service. This general architecture could be used to support any kind of Web-based computation service.
Key Words: Distributed Architecture, Web Based Visualization Services, NOW.

Introduction

Nowadays, the World Wide Web is the most powerful facility to access to the largest information space and raises the need for effective visualization systems to speed up the process of searching and analyzing information. The most commonly used way to visualize information gathered on the Web consists in downloading the data onto a local workstation and then applying visualization algorithms to compute images of the data. But this process raises a number of problems. The memory necessary to store all the data could easily exceed the local disk capacity and it could take a long time to download the whole data set. Indeed, the user more likely would like to be able to browse the data for identification of a particular area of interest before deciding to download part of the data for further exploration and analysis.

At LIL we developed *viscWeb* [5], a distributed visualization architecture which allows us to consider a pool of machines connected through Internet as a networked resource to support a Web-based visualization service. This article presents the main changes in the architecture, our scheme of automatic parallelization as well as new results. The outline of the article follows : the first part presents some related works, the second gives an overview of the service, the third describes the system architecture while part four details some parts of the implementation. Finally, we purpose response times for the service and we analyze the benefit of the parallelism from the user point of view.

1 Previous Work

Previously published works include distributed architectures for computational steering and Web-based visualization systems. The European project PAGEIN [3] was initiated to set up and evaluate environments for distributed Computer Supported Collaborative Work in a high performance computing and networking context. The project led to the COVISE [14] environment for supporting distributed visualization. Recent extensions to COVISE include a Web-based interface and a distributed VR environment. Shastra [1] is a distributed visualization environment supporting collaborative work. There are other systems especially tuned for computational steering [12] [4].

The concept of a Web-based visualization service has been studied by several authors. The objective is to put visualization on every desk. Typically the user has access to a visualization service using a Web browser, and sends visualization requests by means of HTML (Hyper Text Markup Language) forms. Several scenarios have been investigated, including *thin client*, where the server computes an image of the data and sends it back to the user, and *intelligent client*, where a Java-based visualization program is downloaded on the client site to make the visualization [15][8][13].

2 Overview of viscWeb

With *viscWeb* the user can submit parameterizable requests through HTML forms and receives images or VRML descriptions (Virtual Reality Modeling Language), which are visualized by the browser itself or an external application. *viscWeb* is not handled by a single machine but rather takes advantage of a pool of Unix machines connected by Internet. The features currently supported by *viscWeb* are:

- user sessions: each user is assigned his own user session and at least one visualization server is entirely dedicated to that user,
- dynamic code placement: when a new visualization server has to be started, a machine is elected among the pool of machines available for the service as a function of various parameters, including machine charge and number of currently logged users.
- dynamic code migration: servers can be moved from one machine to another at any time, for instance if a charge imbalance occurs or a regular user logs in the machine.

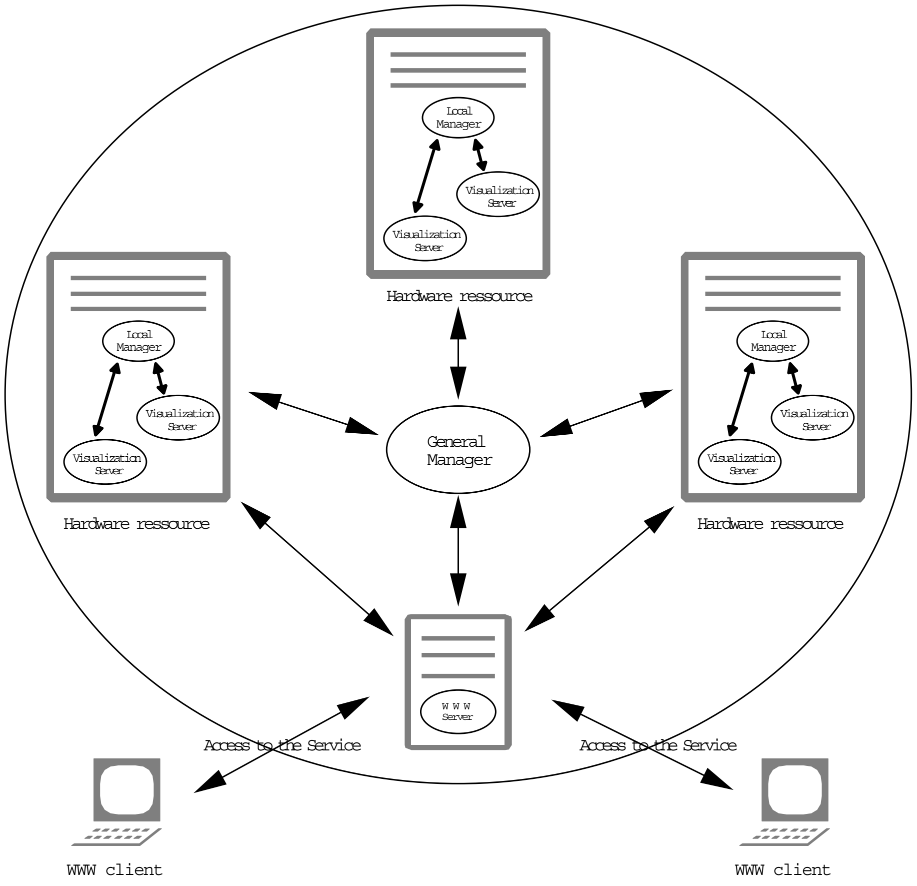

Fig. 1. *viscWeb* architecture.

– automatic parallelization: depending on the number of machines available, several of them can be used to process a visualization request in a parallel scheme.

3 System Architecture

The architecture of the system is shown on figure 1. A number of visualization servers are running to process requests from currently connected users. The distribution of the servers among the machines is managed by a special program called *General Manager*. In addition, on every machine a *Local Manager* is running. Each Local Manager provides the service with the same basic functionalities, including launch a new server, get information about all servers currently running on the machine, stop a server, ... Communications occur directly between CGI (Common Gateway Interface) programs executed on the machine supporting the Web service and visualization servers executed on remote machines. The Local Manager communicates only with local visualization servers and the Global Manager. Local Manager communications are only service messages, for instance to launch a new visualization server or to inform the Global Manager about the situation on the local machine. All communications in *viscWeb* are based on sockets and TCP (Transport Control Protocol) to ensure reliable transmissions. The visualization part of *viscWeb* has been written with VTK [10] (The Visualization ToolKit) because it is freely available on all major systems as source code.

4 System Features

4.1 User Sessions and Server Placement

When a user connects to the service, a machine is elected and a visualization server dedicated to that user is launched on this machine. This is achieved by a RPC call (Remote Procedure Call). This implies that the visualization service should have been declared on each machine or a least a registered daemon should run on each machine. Data transmissions in *viscWeb* rely on the XDR format (eXternal Data Representation), which allows transparent data exchange between binary incompatible systems. Thus *viscWeb* can handle any machine that runs a Unix system.

The machine chosen to launch a new visualization server is elected as a function of a number of parameters. In the current version we check whether a regular user session is currently running on the machine and we evaluate the average workload of the machine. In the future, additional parameters will be taken into account, such as the bandwidth between the machine and the Web server, the amount of available memory, ... This visualization server is attached to the user. Because parameter values are subject to changes over time, we have developed a mechanism to dynamically move a server when necessary (see section 4.2). The Global Manager maintains a table which contains, for each visualization server, all necessary information concerning the link between this server (identity, IP address, socket connection) and the related user.

In *viscWeb* a new visualization server is not started each time a visualization request is received by the Web server but rather a new user session is initiated when a user connects to the service and this session keeps running until the user disconnects. The advantage of this solution is that it avoids recomputing all the visualization pipeline each time a parameter is changed, which is the way visualization pipelines are maintained by traditional softwares such as VTK or IRIS Explorer. The drawback is that if the connected user does not disconnect, its environment on the machine running its visualization server is not properly cleared, and a mechanism to detect zombies servers must be installed.

4.2 Dynamic Code Migration

The parameters used to choose a machine when a new user session is started change over time. For instance a regular user of the machine may log in. In this case it is not possible to complete the task on the same machine. The Local Manager detects the user login and asks the Global Manager for a new location where to move its visualization servers. The algorithm used by the Global Manager is the same as for a new user connection. When new machines have been found to handle the work, the Local Manager stops visualization servers (by means of Unix signals) and gives them their identities. When a new server is launched, it receives from the current server all data and information necessary to complete the task, then the old server terminates.

4.3 Automatic Parallelization

When a visualization request has to be processed, several machines could be available for the task. The final goal being to ensure response times as low as possible, *viscWeb* includes automatic parallelization. The Local Manager chooses a subset of machines that can be used to process the data among the whole pool of available machines, using the same parameters as for a new user connection. In the current version, a data parallel scheme is used to distribute computation and data over available machines. This mechanism is used for surface extraction with the Marching Cubes and volumetric ray-tracing (both available as sequential codes in VTK). These algorithms are easily parallelizable but additional investigations are needed for a general extension to automatic parallelization. Nevertheless, for a good parallelization, the algorithm might take into account such parameters as the network bandwidth, the individual power of the machines, . . . in order to deal with heterogeneous load balancing. In the current version, the Local Manager chooses all machines that are not used at the instant of the request. If a second visualization request occurs in the service, a Local Manager is likely to select the same machines since the parameters won't have change much between two consecutive requests.

This problem also will be addressed in future version of viscWeb, where a limited number of machines will be chosen, sparing some of them for a following request. Moreover, some algorithms become inefficient when being too much parallelized, if the data is too partitioned and too many machines used. Many known works point out this issue in purely parallel scopes. Indeed a balance has to be found between increasing the number of machines and the computational power, and increasing the control, hence the number of messages in the network.

4.4 Portability to Other Systems

Our software is fully written in C++, it is based on sockets, TCP, XDR and RPC calls for communications, and VTK for the graphic stuff. Because all these technologies are Unix standards, the portability of viscWeb to any Unix system is straightforward. Currently we have machines running under SGI IRIX and

Linux. The portability toward non-Unix worlds, such as Windows, more likely involves using Java. In fact, since C++ and VTK are available on Windows, only the communication part should be rewritten in Java, for which we plan to use Java RMI (Remote Method Invocation), and the native code in C++ be reused thanks to the JNI (Java Native Interface). This is part of the on-going work concerning this project.

The client part of the software has been tested successfully on IE4.x and Netscape4.x, what is an expected result since it is web pages, forms, and CGI communication based on standard HTTP (Hyper Text Transfer Protocol). We plan to rewrite the client part with Java to make te interface more interactive, which would allow us for instance to validate the parameters before the form be send to the server.

5 Results - Timings

One of the main issues of Web-based visualization concerns the time necessary for the user to get the results. Important factors which affect significantly the response time are the network bandwidth, the server charge and the computational requirements of the visualization. In order to decrease the response time, it is necessary to consider all these three aspects. It is not in the scope of this article to deal with algorithmic issues of scientific visualization and indeed in *viscWeb* we did not develop any specific visualization method. We fully rely on the VTK toolkit, using well known algorithmic approach to parallelize our code.

The network problem is mainly a hardware issue. Today high bandwidth networks such as the 622Mbps MBone offer a suitable solution for Web-based visualization services, assuming no real-time interaction is required. VRML viewers provides us with some kind of real-time interaction but interaction is limited in the navigation through pre-computed scenes, that is only the rendering phase is interactive, the visualization phase remaining static. A suitable approach to enhance interactivity is progressive downloading of hierarchical structures [2] (either images or 3D descriptions), although VRML 2.0 does not support progressive downloading and visualization. One of the goals of this work is to propose a distributed architecture which allows us to take advantage of a pool of machines to decrease the response time of the system.

In order to evaluate the effectiveness of our system, we had made two series of measures for two different requests. A SPECT (Single Positron Emission Computed Tomography) volume of the brain of resolution $64 \times 64 \times 10$ is visualized. This technique is used to detect disfunctionalities in the brain by imaging the blood flow.

The Marching Cubes [7] algorithm is used to extract an iso-surface from the volume which produces 14,000 triangles. Two kind of results had been produced : an image of the volume of resolution 500×500 (image of size 10 kbytes), and a VRML description of size 1 mega-bytes. Timings had shown, for a sequential scheme, two main properties : the time to compute the image is highly prominent as compared to the times to send the HTML form to the Web server, to send

	T1[a]	T2[b]	T3[c]	T4[d]	T5[e]	T6[f]
Req 1 (1 host)	1,213.8	432.9	102,341.4	139.98	36.26	104,164.47
Std deviation	7.21%	5.86%	6.45%	61.52%	16.79%	6.41%
% of total	1.16%	0.41%	98.24%	1.34%	0.03%	100%
Req 2 (2 hosts)	1,322.08	574.79	81,087.19	157.32	40.30	83,181.71
Std deviation	22.76%	43.21%	13.58%	66.65%	16.83%	13.18%
% of total	1.58%	0.69%	97.48%	0.18%	0.04%	100%
Req 3 (4 hosts)	1,120.08	480.79	73,491.76	76.80	28.19	75,197.64
Std deviation	7.82%	9.88%	3.11%	67.28%	83.12%	2.92%
% of total	1.48%	0.63%	97.73%	0.1%	0.03%	100%

[a] Time spent by the CGI program to analyze the request.
[b] Event transmission from the CGI program to the visualization server.
[c] Visualization time.
[d] Result transmission from the visualization server to the CGI program.
[e] Result transmission from the CGI program to the Web server.
[f] Total time necessary to process the visualization request.

Table 1. Impact of parallelism in *viscWeb*. Times are in milliseconds.

the data to the visualization server or to send back the result to the client (from 86% for the VRML description up to 99 % for the GIF image. Please refer to [6] for further details.

We decided obviously to figure out the benefit obtained using a parallel scheme compared to the previous sequential one. We used a bigger data set of resolution $64 \times 64 \times 64$, which produces up to 70,000 triangles. Three requests have been executed with 1, 2 and 4 machines to compute these triangles. In this scheme, only the Marching Cube part of the algorithm is parallelized, where each machine handles the same number of voxels (this will have to be improved since it might lead to an unbalanced system where each machine compute different number of resulting triangles).

Table 1 details some of the results obtained for each request. We used the workstations of our lab to test the system: SGI R4600PC with 32Mbytes for the Web server and SGI R5000SC with 256Mbytes for the visualization server with the sequential scheme, and up to four SGI R4600PC with 32Mbytes for the parallel one. Since the charge of our LAN and the workstations change continuously each request has been executed five times and the given values are the average values. Requests 1, 2 and 3 outline timings using respectively 1, 2 and 4 machines, associated to compute the final result.

The values in rows 2, 5, 8, 11 and 14 give the standard deviation and rows 3, 6, 9, 12, 15 give the percentage of the total time spent in each stage. A first remark concerns the high standard deviation which is due to the fact that machine and network charges change over time. The network also is shared between the members of the laboratory : we did the tests during the day, when others were using the network, leading to high unbalance in the network related times of the result table. We did not want to have especially artificial good results, which could have been achieved using the visualization server during the night, since we wanted to test its behavior over a realistically loaded network.

We can easily see that most of the time is spent in the visualization time, with 1, 2 or 4 machines. The speedup we obtain for 2 processors is about 1.25 and reach 1.4 with 4 processors. The total time to compute the result is lowered by about 30 seconds, which is a great improvement for the user connected to the service. The results could be even better if the final stage of the pipeline visualization (rendering of the produced triangles) was also parallelized.

6 Conclusion and Future Work

viscWeb allows us to enhance the quality of a Web-based visualization service by increasing the computational power used to support the service. Statistical studies have been made about the usage of workstations in a LAN, which has shown that the CPU usage is generally far below average [9][11]. The goal of this version of *viscWeb* is to use the available power in a LAN to support a Web-based visualization service. The next step is to propose a flexible Internet-wise environment to provide visualization services to end-users with a basic equipment. Depending on the hardware and software requirements of a user request, the visualization pipeline will have to be distributed over the global Internet. The placement of the visualization servers will depend on the location of the data, on the availability of the software components necessary to process the request, on the location of the computational resources, on the network bandwidth, ...and we should include the possibility for the user to handle part of the pipeline on its own machine. Moreover such a flexible environment should be perfectly interfaced with DBMS (Data Bases Management System). This involves taking into account all recent standards and technologies for data and code exchange over heterogeneous platforms at the Internet level, such as Java and Corba.

References

[1] V. Anupam and C. Bajaj. Shastra: an architecture for development of collaborative applications. *International Journal of Intelligent and Cooperative Information Systems*, pages 155–172, jul 1994.

[2] Andrew Certain, Jovan Popoviĉ, Tony DeRose, Tom Duchamp, David Salesin, and Werner Stuetzle. Interactive multiresolution surface viewing. *Computer Graphics*, pages 91–98, aug 1996. Proc. of SIGGRAPH '96.

[3] Michel Grave. PAGEIN: Pilot application in a gigabit european integrated network. Technical Report R2031/ONE/DI/034/b1, ONERA, Châtillon, France, feb 1996.

[4] J.J. Hare, J.A. Clarke, and C.E. Schmitt. The distributed interactive computing environment. Proc. of the Workshop on Distributed Visualization Systems, Research Triangle Park, NC, oct 1998.

[5] Wilfrid Lefer. A distributed architecture for a web-based visualization service, apr 1998. Proc. of Nineth Eurographics Workshop on Visualization in Scientific Computing, Blaubeuren, Germany.

[6] Wilfrid Lefer and Jean-Marc Pierson. A thin client architecture for data visualization on the world wide web. In *International Conference on Visual Computing (ICVC99), Goa, India*, feb 1999.

[7] William E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(3):163–169, jul 1987. Proc. of SIGGRAPH '87.

[8] Cherilyn Michaels and Michael Bailey. Vizwiz: A java applet for interactive 3D scientific visualization on the web. In Roni Yagel and Hans Hagen, editors, *Proc. of IEEE Visualization '97*, pages 261–267. IEEE Press, Los Alamitos, CA, oct 1997.

[9] D. A. Nichols. Using idle workstations in a shared computing environment. *ACM Operating System Review*, 21(5):5–12, nov 1987.

[10] William Schroeder, Kenneth Martin, and Bill Lorenzen. *The Visualization Toolkit - An Object-Oriented Approach to 3D Graphics*. Prentice Hall, 1996.

[11] M. M. Theimer and K. A. Lantz. Finding idle machines in a workstation-based distributed environment system. *IEEE Transactions on Software Engineering*, 15(11):1444–1458, nov 1989.

[12] Robert van Liere, J.A. Harkes, and W. C. de Leeuv. A distributed blackboard architecture for interactive data visualization. In *Proc. of IEEE Visualization '98, Research Triangle Park, North Carolina, Oct 19-24, 1998*. IEEE Press, Los Alamitos, CA, oct 1998.

[13] Kiril Vidimce, Viktor Miladinov, and David C. Banks. Simulation and visualization in a browser. Proc. of the Workshop on Distributed Visualization Systems, Research Triangle Park, NC, oct 1998.

[14] A. Wierse, U. Lang, and R. Rühle. A system architeture for data-oriented visualization. In John P. Lee and Georges G. Grinstein, editors, *Database Issues for Data Visualization*, number 871 in Lecture Notes in Computer Science, pages 148–159. Springer-Wien-New-York, 1993. Proc. of IEEE Visualization '93 Workshop, San Jose, California.

[15] Jason Wood, Ken Brodlie, and Helen Wright. Visualization over the world wide web and its application to environmental data. In Roni Yagel and Gregory M. Nielson, editors, *Proc. of Visualization '96*, pages 81–86. IEEE Press, Los Alamitos, CA, oct 1996.