Using Grid Computing for Rendering to Support 3D Animation Training Courses

Wichian Premchaiswadi[1], Anucha Tungkasthan[2] and Nipat Jongsawat[3]
Graduate School of Information Technology in Business, Siam University, Thailand
wichian@siam.edu[1], aimdala@hotmail.com[2], nipatj@yahoo.com[3]

Abstract: Rendering of an animated scene is considered to be one of the most important steps in 3D animation construction. Rendering basically converts 3D geometric models into graphic images. In 3D animation training courses, rendering complex 3D models is a very time consuming task since thousands of frames are needed to create an animation. It is considered one of the major limitations for creating professional 3D animation. This paper presents the use of grid computing for 3D rendering. It can reduce the rendering time and still maintain the quality of the final animation. Software and system architecture solutions are proposed and developed. A graphical user interface (GUI) plug-in and web portal were developed in order to access grid computing facilities. Animators are able to render highly complex 3D models in order to create their animation sequences by using high performance grid computer technologies, monitor rendered scenes, and download the finished images from the server to their own computer. These applications can be used as tools to assist animators in developing their animations and motivate them to produce a better quality animation within optimum cost and time parameters.

I. Introduction

Rendering is considered to be one of the most important steps in 3D animation construction. Rendering converts 3D geometric models into graphical images. Rendering of an animated scene is a very time consuming task since thousands of frames are needed to create an animation. Typically, rendering computer animation frames takes about several hours when the source scene is complex or when photo-realistic images are required. Despite the development of new techniques and algorithms, this process is computationally intensive and requires a large amount of time in order to obtain a professional quality computer animation.

In 3D animation training courses, it is unreasonable to require students or animators to spend thirty minutes to create a 3D animation model and then have to wait several hours to render animation frames on their computers. Normal computers such as personal computers (PCs) would be insufficient because rendering on a PC takes an extremely long time. This paper focuses on using grid computing for rendering 3D animation frames in parallel. Using this powerful method, it can reduce rendering time and still maintain the quality of the final animation. Students or animators are able to render highly complex 3D models for creating their animation sequences by using high performance grid computer technologies. They can complete the overall process of 3D animation construction within the specific time frame defined by an instructor in a class.

In this paper, the software architecture and system architecture of grid-based computer animation rendering and two types of animation rendering applications are also presented. All of them are described in detail in the remainder of this paper.

This paper is structured as follows: Section 2 addresses related work. Section 3 addresses the details of the software architecture and the system architecture for grid-based computer animation rendering. Section 4 describes the design and implementation of a GUI plug-in and web portal. Section 5 presents conclusions and additional topics for future research.

II. Related Work

There are a number of papers concerning grid-based computer animation rendering, but this paper only discusses the most relevant subset of them. The related work concerned with using a grid for rendering, developing the rendering programs/plug-ins and portals are described in the following.

Anthony Chong, et al [1] presents a grid-based rendering framework for on-demand animation sequence rendering. Their proposed framework provides two methods for accessing the grid, namely direct rendering from 3D Maya and rendering through a portal access. Direct rendering from 3D Maya allows users to submit jobs to the grid through a GUI plug-in. The batch rendering process is designed to appear transparent to the animators. They can only click on the render icon to send the animation sequences to the grid for rendering. The rendered images are stored and can be downloaded from a database through the portal. The animators can submit rendering jobs, render jobs, and download the rendered images through an access portal. An open source plug-in (LiquidMaya) for Maya 3D is used to create a GUI for animators. Their plug-in converts the Maya scene to RenderMan RIB format [11], [12] and uses Pixie for rendering. A ruby script is used to distribute the rendering tasks to the grid. The Sun Grid Engine (SGE) meta-scheduler is used in each grid cluster to distribute the entire animation job to its available rendering computers in each cluster. Chong's work is the work most similar to this paper except for two factors. First, they developed the plug-in for Maya 3D but this project has developed the plug-in for the Blender software. Even through the LiquidMaya is open source software and allows the user to fully develop an application, the Maya software is commercial software and a software license is necessary for installation. Second, the software and system architecture they present are different than the one presented in this paper.

Bader Aljaber, et al [2] present the Gridbus POV-Ray distributed rendering application which was developed as a Java program. POV-Ray is short for Persistence of Vision Ray tracing [17]. It is a tool for producing high-quality computer graphics. The POV-Ray program creates three-dimensional, photo-realistic images using a rendering technique called ray-tracing. Ray-tracing is a very compute intensive method for generating 3D images, but it produces very high quality images with realistic reflections, shading, perspective and other graphical effects [3]. POV-Ray in its normal form uses only one computer to render an image. Because of this, some images may take many hours or even days to fully render on the most current PCs. Animation, as a natural extension can take an intractable amount of time [4]. Basic clustering, through useful, might not be sufficient to render a very complex image or animation in a reasonably short amount of time. One way to achieve a more efficient rendering process is to use more than one cluster: that is, applying grid computing technology [13]. Thus POV-Ray executing on a global grid was introduced [15]. The distributed POV-Ray rendering application was developed as a Java program, and uses the Gridbus Grid Resource Broker [5] Application Programming Interface (API) [6], [18]. The broker allows transparent and seamless access to heterogeneous resources, providing services such as resource discovery, secure access, scheduling, monitoring, and job submission [6]. It supports both computational and data grid applications. Naturally, obtaining and managing access to grid resources is not a simple task. Use of a resource broker aims to simplify this process by providing an abstraction layer for users who just want to get their work done. Resource brokers are software components that let user access heterogeneous resources transparently, without having to worry about availability, access methods, security issues and other policies. The broker takes care of job submission and scheduling. It dispatches tasks to available resources, collects the results and returns them to the users. It isolates the complexity of the underlying system architecture of the grid and provides a uniform access method to different core middleware grid components. Bader Aljaber et al [2] implemented a rendering program that creates tasks in the Gridbus broker by specifying the start row and end row for each slide of an image. The broker executes the tasks by sending them to its dispatcher module, which is responsible for submitting and monitoring the tasks on the resources. The broker then polls the resources periodically, and raises events when the status of tasks changes. The program updates the display accordingly when it receives these events. Once all the slices of one image have been collected after rendering, the Java program they created displays all these slides as one image. It then generates the final PNG file from the slices, and writes it to the working directory of the program. When the image has been rendered, the slides are sent to the resources in the correct order, but will be returned back from the resources possibly out-of-order, depending on the various speeds and load levels of the resources. The user can choose the image that needs to be rendered. All the files in the directory containing the '.pov' file are copied to each grid node. The user must make sure that all the files required for rendering a particular image, such as texture files and '.inc' files are included within the same directory. The user is allowed to specify the size of the image to render as well as the name of final resulting image. The final image is placed in the working directory of the program. In this application, the user can also choose the distributed resources from a pre-defined list, or specify an XML resource listing file, which contains a list of grid resources to use in the rendering. When all options have been selected, the user can initiate the rendering process. Once all of the slices have been rendered, they will form the final image on-screen, and the final '.png' file will be generated and written to the working directory of the program. When all nodes finish their jobs, the complete output is displayed.

In our literature review, no researchers have developed an overall grid-based computer animation rendering system and used it in a specific training course like this project for several reasons. First, both software and system architectures are very complicated and difficult to implement. Second, building, implementing, and managing

computationally intensive grid computers requires a lot of time and investment. Third, the professional software and hardware developers are required to complete the design and implementation of the system. Last, building a GUI rendering plug-in for a specific 3D animation program requires not only technical programming skill but also highly specific knowledge in 3D computer animation. All functions of the plug-in must be simple to use while the underlying software and system architectures are very complicated tasks to develop. All of these limitations are very challenging. This research presents the techniques and methods used to overcome such limitations.

III. Proposed Software and System Architecture

Theoretically, there are two types of rendering methods: network rendering and distributed (split-frame) rendering [7], [8], [9]. In network rendering, images can be rendered in parallel [14], as each frame can be processed independently of the others. In this case, the main communication between processors is used for uploading the initial models and textures and downloading the finished images. In distributed (split-frame) rendering, each frame is divided into tiles which are rendered in parallel. In this paper, the researchers have chosen the network rendering method for implementation.

This following section describes the software architecture and system architecture for 3D animation rendering on grid computing. The details of each component in the architecture are illustrated in the following section.

A. Software Architecture

Fig. 1 presents a simplified scheme of the software architecture for 3D animation rendering using grid computing. The software architecture describes what software is needed in each layer. The software's functionality is also described in detail. This section also describes how grid computing can be used for rendering computer animation and how applications can be used for accessing the grid computers by using several types of software. There are three main parts presented in the software architecture below. The details of each part are described as follows:

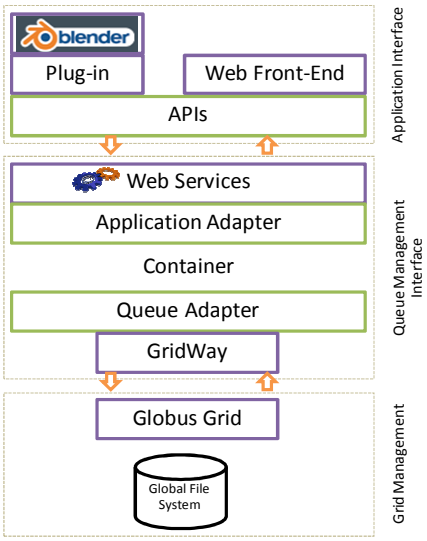

Fig. 1. Software architecture for 3D animation rendering on grid computing.

Part 1: **Application Interface**, this part consists of two main components or applications. One is the Blender plug-in and the other one is the rendering web portal or simply portal.

Blender Plug-in: Blender [10] is a free open source 3D computer graphics program. It is a fully functional 3D modeling and rendering animation package for several different operating systems such as Linux, Unix, and Windows. It is a powerful tool for 3D modeling, animation, rendering, post-production, interactive creation and playback. In 3D animation training courses at Siam University, the Blender program was chosen as the main 3D

animation development program for several reasons. First, the Blender is free, open source software. Open source means that it will continue to be improved. It is suitable for use in training courses and educational environments because no software license is required. Second, it provides a complete set of functions for 3D animation development. Third, the Blender community today is large and contains a number of useful services such as news, forums, galleries, downloads, links, tutorials, polls, and a chat room. Finally, it allows the animator or programmer to completely develop a rendering plug-in for the program. In our work on this project, the last one is the most important reason for selecting Blender as a program for the training courses. A Blender python script is used to create the GUI plug-in. This plug-in provides two main functions for animators. First, it contains a 3D configuration function which allows animators to easily set up the necessary pre-rendering parameters for their jobs before submitting them to the grid for rendering. Second, the 3D rendering function provides animators with the capability to render jobs on the grid. All of these parameters are sent to web services through application programming interfaces (APIs).

A Rendering web front-end or portal is specifically designed and implemented to facilitate the animator's ability to upload their rendering jobs on-demand, and download their rendered images from the remote server at a later time. In addition to job submission and rendering, by using the portal they can also manage files using a file manager, monitor the progress of rendered scenes, and download the finished images. The portal was developed using open source software, namely, PHP and MySQL. This portal is actually an entry point for accessing web services. It calls web services by sending a HTTP request containing a Simple Object Access Protocol (SOAP) message.

Part 2: **The Queue Management Interface** is the middle layer. The queue management interface helps establish the connection between an application interface layer and grid management layer. This part consists of two main components. One is web services and the other one is the GridWay meta-scheduler.

Web services: The web services presented in this paper are APIs that can be accessed over the Internet and executed on a remote system hosting the requested services. It allows several parts such as the plug-in/portal and the meta-scheduler to communicate with each other using the HTTP protocol. There are two rendering scripts (XML messages) used for web services that are defined for the communication. The first one is a job submission script and the second one is a job statistics script. Some parameters defined in the job submission script consist of the type of application, job name, comment, scene name, directory name, external name, output directory, etc. Some parameters defined in the job statistics script consist of job name, account id, owner, scene, frame count, state, etc.

GridWay: GridWay is the grid meta-scheduler. It was specifically designed to work on top of Globus services. It is a workload manager that performs job execution management and resource brokering on a grid consisting of distinct computing platforms managed by Globus services. The GridWay meta-scheduler used for this paper performs four important functions. First, it receives a job and its details from the user. Second, it selects a resource. Third, it submits the job to the resource. Fourth, it monitors and reports job status to the user. The grid meta-scheduler also provides central job and resource management, enabling users to submit jobs without requiring knowledge of the grid configuration and status, and makes scheduling decisions when job submissions are received.

Part 3: **Grid Management** is the bottom layer of the architecture and contains the grid software toolkit called Globus [16]. The Globus Grid or Globus toolkit is an open source software toolkit used for building and managing computational grids. It is being developed by the Globus Alliance and many others around the world. Globus provides software tools that makes it easier to build computational grids and grid-based applications. In the Grid management implementation for this paper, all computers or nodes in the grid were installed with Globus toolkits [1]. The Globus toolkits used in this paper consists of GRAM (Globus Resource Allocation Manager) and GSI (Grid Security Infrastructure). GRAM is responsible for managing resources and GSI is a core service for managing access and authorization using a public key infrastructure based on X.509.v3 and Secure Socket Layer (SSL) Protocol/Transport Layer Security (TLS) protocols for authentications.

B. System Architecture

The system architecture for 3D animation rendering on grid computing is constructed as illustrated in Fig. 2.

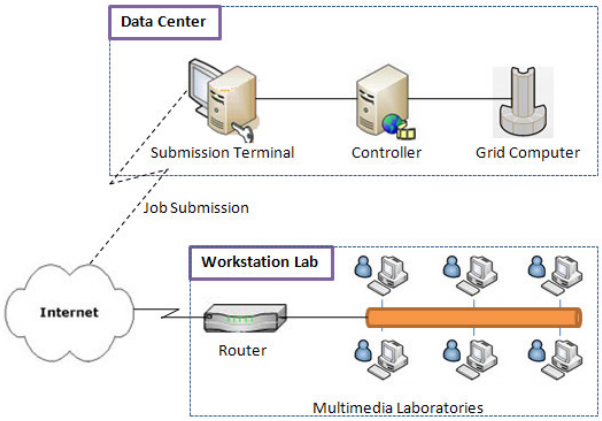

Fig. 2. System architecture for 3D animation rendering on grid computing.

Fig.2 shows two main physical components or infrastructures: the data center and the workstation lab (multimedia laboratories). The data center consists of submission terminal, controller, and grid computers.

A submission terminal is the front end of the data center. It receives rendering jobs from the animators who submit jobs through the web portal. The web portal is located and run on a submission terminal. The submission terminal actually runs as a web server. A client is able to upload rendering jobs to the submission terminal by using the portal. The submission terminal primarily provides a job queue for the controller.

The GridWay meta-scheduler is installed as part of the controller. The controller is responsible for assigning batch jobs to a given computer or node in the grid to be executed/rendered when it receives the rendering jobs or animation frame sequences from a submission terminal. It automatically finds the most appropriate node on which to run any given job that is waiting to be executed. The controller reacts to currently available resources on the grid. It uses measurement information about the current utilization of nodes to determine which ones are not busy before submitting a job. It also monitors the progress of scheduled jobs thereby managing the overall work-flow. The controller utilizes a job priority system. As grid computers become available to execute jobs, the jobs are taken from the highest priority queues first. Policies of various kinds are also implemented using the meta-scheduler installed in the controller.

The grid or rendering grid used for this paper is a collection of computers that provides a rendering service or function for a series of jobs. It means that the Blender software needs to be installed on all computers or nodes. The rendering time for each job depends primarily on the numbers of computers running on the grid. The rendering time can be further reduced by having more computers in the grid. However, keeping hundreds of computers in the same location requires a significant amount of physical space. This limitation can be removed by linking up interconnected computers across different geographical locations. The grid system used for rendering jobs for this project is under control of the Thai National Grid Center (TNGC).

In multimedia laboratories, all computers are simply connected into a local area network (LAN). The animators create their 3D models on their own computer and upload their rendering jobs to the submission terminal. They can use both the plug-in and a portal to render their jobs without worrying about the technical details of animation rendering on grid computing such as the grid configuration.

IV. Implementation

In this section, two main 3D animation rendering applications on a grid computing facility will be described, namely, the plug-in and web portal.

A. Blender Plug-in

Direct rendering on a grid computing facility using the plug-in was designed and implemented to facilitate the ability for animators to render their jobs (job rendering execution) without using any additional programs and tools. The animators can create 3D animations on their own computers during a training class (Fig. 3), save and upload jobs to the submission terminal at the data center, and then submit the job execution command for rendering

by using the plug-in. Fig. 4 shows the two plug-in submenus installed in the main menu, namely, 3DGrid configuration and 3DGrid rendering. An animator selects the menu item 3DGrid configuration to set up some of the pre-rendering parameters such as the output directory, add or delete computing resources for rendering (Fig.5) and 3DGrid rendering to set up the application, project, and scene information such as priority, frame/CPU, start frame, end frame, types of image files and confirmation for job rendering execution on the grid. Fig. 5 and Fig. 6 show the popup windows when the animator clicks on the 3DGrid configuration or 3DGrid rendering submenus. In the 3DGrid rendering window, the submission job ID will be returned to the animator immediately if the submission terminal receives the job successfully (Fig. 7). In the 3DGrid configuration window, the animator clicks on the View Progress after clicking the submission button in the option menu to monitor the progress of rendered scenes. An animator can download the rendered images to their own program for the next animation process(es), interactive creation and playback (Fig. 8).

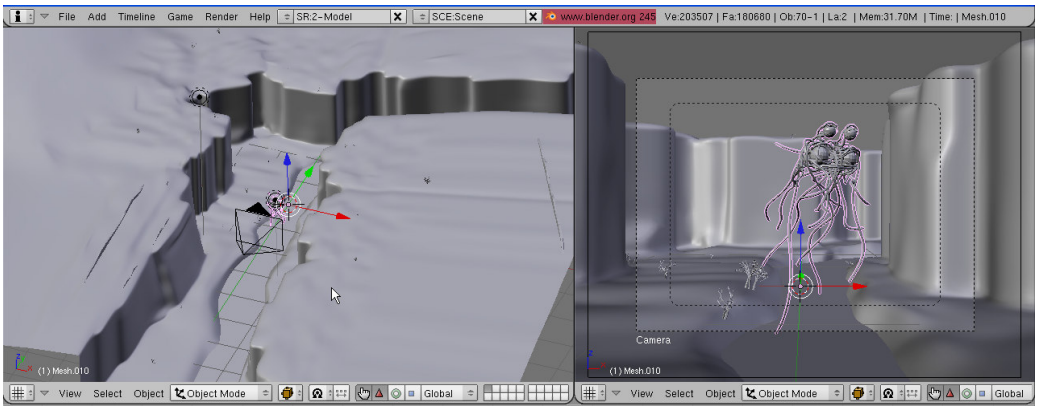

Fig. 3. 3D animation development in the Blender program.

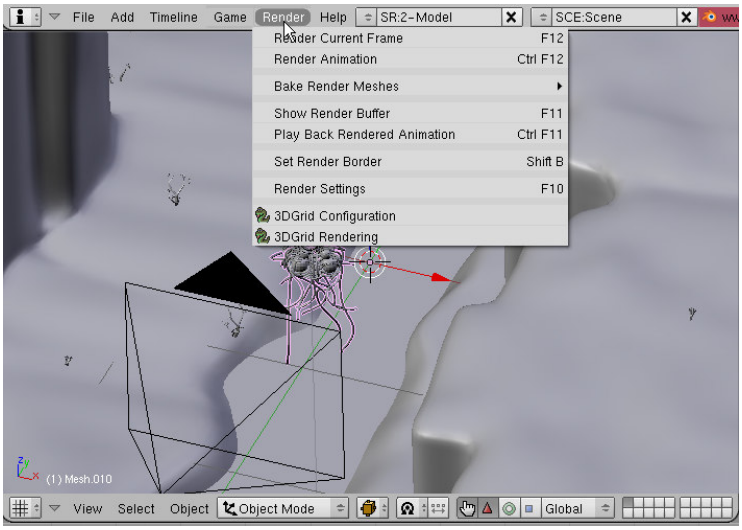

Fig. 4. Plug-in (3DGrid configuration and 3DGrid rendering)

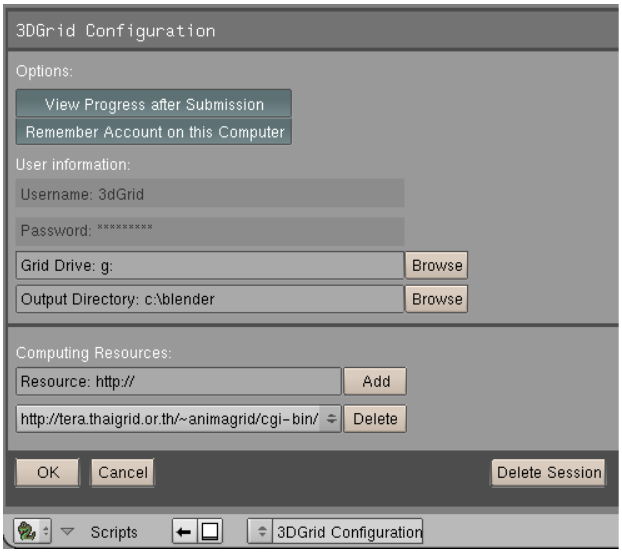

Fig. 5. 3DGrid configuration window.

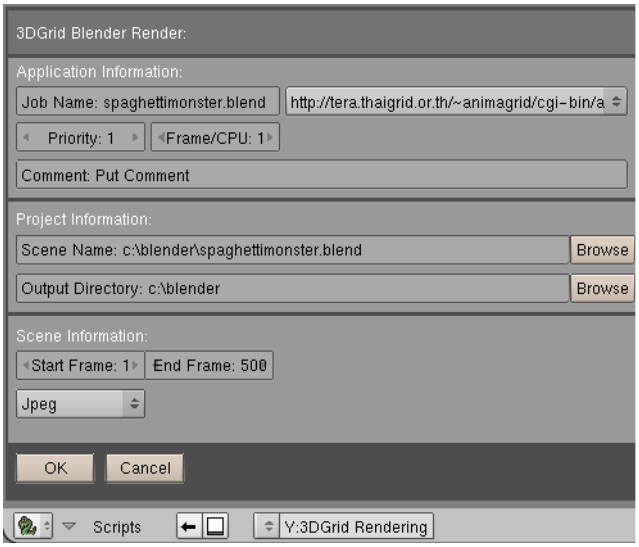

Fig. 6. 3DGrid rendering window.

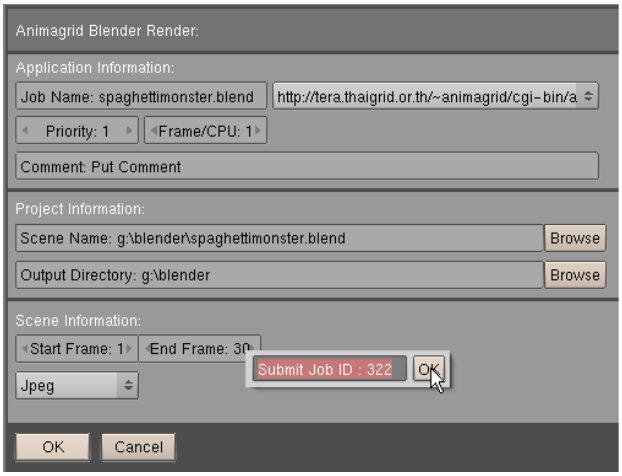

Fig. 7. Return submit job ID to the user.

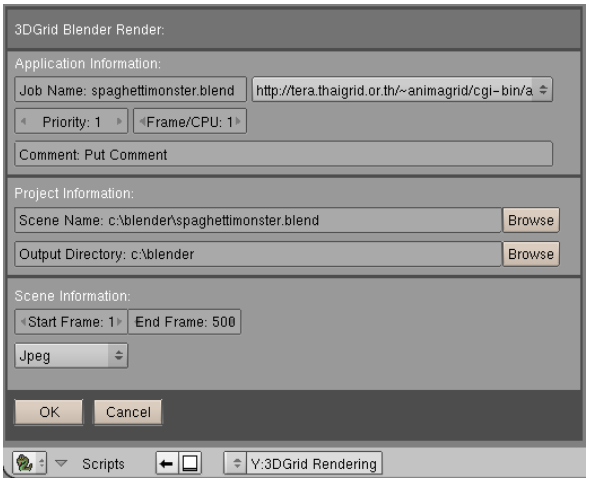

Fig. 8. Playback in Blender after downloading all rendered frames.

B. Web Portal

Fig. 9 shows a web portal. An animator is required to login to the portal in order to use several of the functions provided such as job submission, file management and rendering progress monitoring. The portal provides three main functions for animators. First, job submission allows an animator to upload jobs or animation sequences and submit job rendering execution commands to the grid through the portal (Fig. 10). The portal works exactly the same as direct rendering from the plug-in except that the portal allows the animators to submit their animation scene representation files from any computer with or without the Blender software installed. In case all image files are already uploaded to the submission terminal, animators can easily use the portal to render jobs on-demand and download the rendered images from the server at a later time. Second, file management allows animators to access the database server (submission terminal) and manage rendered image files stored in a database accessible through

the portal (Fig. 11). They can create a folder to store rendered image files, move, cut, copy, and delete files from the server. They can also use the file manager for uploading jobs to the server and downloading the finished images to their own computer (Fig. 12). A Zip file format is provided for downloading. They can also view the finished images before downloading (Fig. 13). Lastly, the rendering animation monitoring function allows animators to check the progress of the rendering on the grid. The progress is shown using a percentage format and job/sub-job status is immediately presented when they refresh the web browser (Fig. 14).

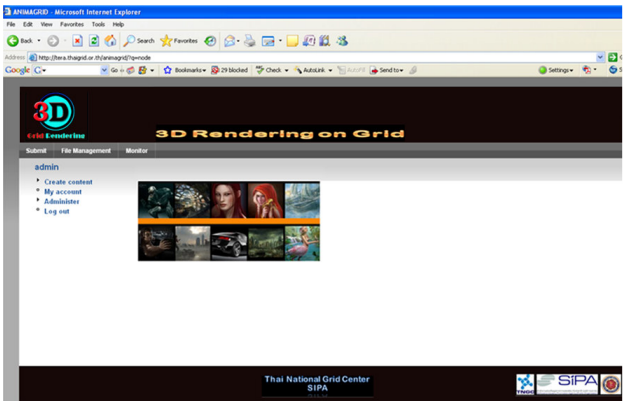

Fig. 9. Screenshot of a web portal.

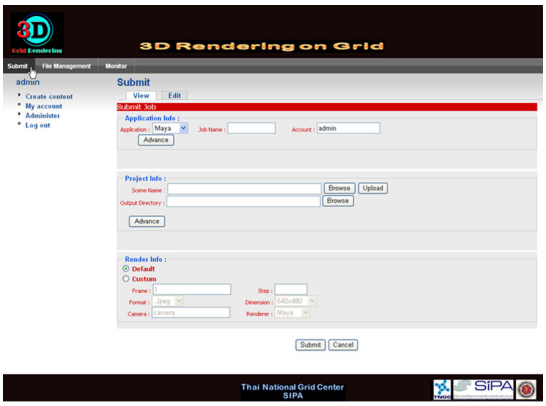

Fig. 10. Screenshot of job submission web page.

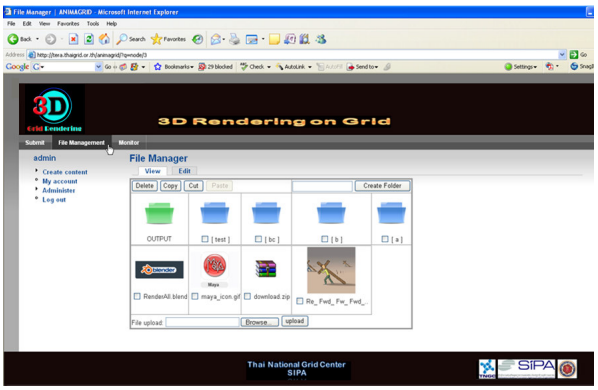

Fig. 11. Screenshot of the file management web page.

Fig. 12. Download the finished images

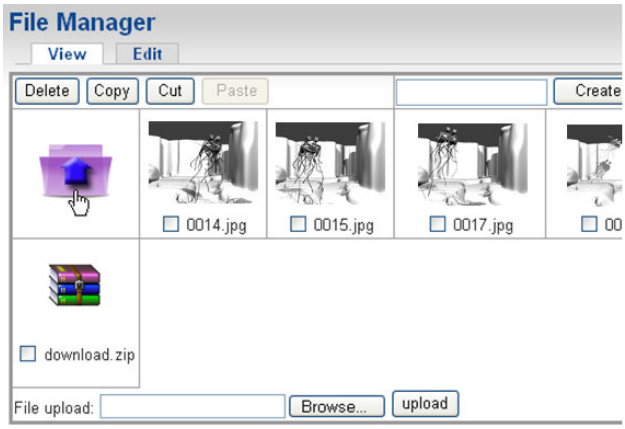

Fig. 13. Viewing the rendered images.

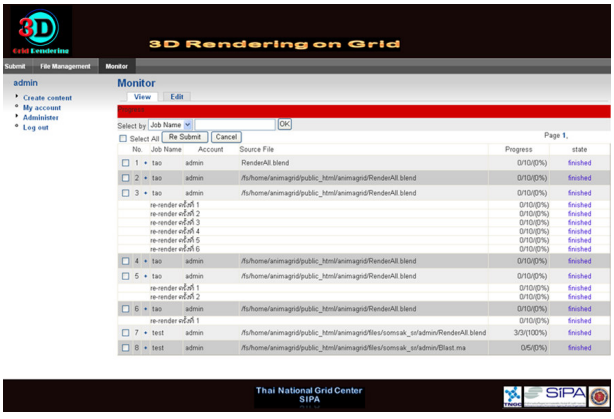

Fig. 14. Screenshot of the job monitoring web page .

V. Conclusion

Based on the experience of teaching 3D animation training courses, rendering 3D animation images is a very time consuming task. The rendering step can take several hours if the source scene is complex or photo-realistic images are required. It also requires a powerful computer for rendering animation scenes. The excessive amount of time and the need for powerful computers are considered serious limitations for professional 3D animation creation. This paper presents a method for overcoming such limitations by proposing and developing both a software and system architecture for 3D animation rendering on grid computing facilities and developing applications that help animators to accomplish their tasks within a reasonable time frame. The two applications developed for the 3D animation training courses consist of the plug-in and the web portal. Both of them allow animators to submit jobs to the grid, send the rendering execution command, check the progress of the rendered scenes, and manage image files. The standard interface (Application Program Interface-APIs) was designed for enabling interoperability among systems as shown in the architecture. The goal of this project was to provide the tools for accelerating the speed at which animators could produce a high quality 3D animation within optimum cost and time parameters.

Future work will concentrate on improving the GUI for both the plug-in and the web portal and to make them more flexible for users. The plug-in for other 3D animation programs will be considered and developed based on the existing software and system architecture solutions proposed in this paper

Acknowledgement

The authors would like to thank the Thai National Grid Center (TNGC), Software Industry Promotion Agency (Public Organization), Thailand for providing the TeraGrid used for 3D animation rendering and for the great technical support.

References

[1] Anthony Chong, Alexei Sourin, and Konstantin Levinski, "Grid-Based Computer Animation Rendering", GRAPHITE 2006, Kuala Lumpur, Malaysia, November 29 – December 02, 2006.

[2] Bader Aljaber, Thomas Jacobs, Krishna Nadiminti, Rajkumar Buyya, "Multimedia on Global Grids: A Case Study in Distributed Ray Tracing", Malaysian Journal of Computer Science, Vol. 20(1), 2007.

[3] POVray project description- http://www.povray.org/documentation/view/3.6.1/3/ (accessed in June, 2006).

[4] Davis, T., "Generating Computer Animation with Frame Coherence in a Distributed Computing Environment", in Proceedings of the 36th Annual Southeast Regional Conference, April 1998.

[5] S. Venugopal, R. Buyya, and L. Winton, "A Grid Service Broker for Scheduling e-Science Applications on Global Data Grid", Concurrency and Computation: Practice and Experience, Vol. 18 No. 6, pp. 685 – 699, Wiley Press, New York, USA, May 2006.

[6] K. Nadiminti, S. Venugopal, H. Gibbins, and R. Buyya, "The Gridbus Grid Service Broker and Scheduler (2.0) User Guide", Technical Report, GRIDS-TR-2005-4, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, April 22, 2005.

[7] Foster, I. and Kesselman, C. 1998. The Globus Project: A Status Report. In Proceedings of Heterogeneous Computing Workshop, IEEE Press, 4-18.

[8] Carlos Gonzalez - Morcillo, Gerhard Weiss, David Vallejo, Luis Jim?enez1y, and Jose A. Fdez-Sorribes. "3D Distributed Rendering and Optimization using Free Software". Free software : Research and Development. UPGRADE Vol. VIII, No. 6, December 2007.

[9] Rudrajit Samanta, Thomas Funkhouser, and Kai Li. "Parallel Rendering with K-Way Replication". Proceedings of the IEEE Symposium on Parallel and Large-Data Visualization and Graphics, p.75-84.

[10] http://www.blender.org/(accessed in November, 2008)..

[11] Apodaja, T. 1995. Using RenderMan in Animation Production. ACM SIGGRAPH 1995 Course 4, www.renderman.org/RMR/Publication/sig95.course04.pdf.

[12] Apodaja, T., Gritz, L., Pharr, M., Hery, H.,Bjorke, K. and Treweek, L. 2001. Advanced RenderMan3;Render Harder ACM SIGGRAPH 2001 course 48, www.renderman.org/RMR/Publication/sig01.course48.pdf.gz.

[13] Brodlie, K., Duce, D., Gallop, J., Sagar, M., Walton, J. and Wood, J. 2004. Visualization in Grid Computing Environments. In Proceedings of IEEE Visualization, 155-162.

[14] Crockett, T. W. 1997. An Introduction to Parallel Rendering. Parallel Computing 23, 7, 819-843.

[15] P. Asadzadeh, R. Buyya, C.Ling Kei, D.Nayar, and S.Venugopal, “Global Grids and Software Toolkits: A Study of Four Grid Middleware Technologies”, High Performance Computing: Paradigm and Infrastructure, Laurence Yang and Minyi Guo (editors), Wiley Press, New Jersey, USA, June 2005.
[16] I. Foster I. and C. Kesselman, “Globus: A Metacomputing Infrastructure Toolkit”, International Journal of Supercomputer Applications, Vol 11 No.2: pp. 115-128, 1997.
[17] B. Freisleben, D. Hartmann, T. Kielmann, “Parallel Raytracing: A Case Study on Partitioning and Scheduling on Workstation Clusters system Sciences”, in Proceedings of the Thirtieth Hawaii International Conference on System sciences, Vol. 1, 7-10 Jan. 1997, pp. 596 – 605.
[18] R. Buyya and S. Venugopal, “The Gridbus Toolkit for Service Oriented Grid and Utility Computing: An Overview and Status Report”, in Proceedings of the First IEEE International Workshop on grid Economics and Business Models (GECON 2004, April 23, 2004, Seoul, Korea), pp. 19-36, IEEE Press, New Jersey, USA.