# Creating Grid Resources for Use in Undergraduate Coursework

John N. Huffman
Brown University
Box 1824
Providence, RI
02912, USA
1-401-863-7291
John_Huffman@brown.edu

Richard Repasky
Indiana University
2711 E. 10th St.
Bloomington, IN
47408, USA
1-812-856-4404
rrepasky@indiana.edu

Joseph Rinkovsky
Indiana University Purdue University
535 West Michigan Street
Indianapolis, IN
46202, USA
1-317-278-6092
jrinkovs@indiana.edu

## ABSTRACT

Rendered animations are a significant part of many student and research projects being done at Indiana University. Typically the rendering process is very computationally intensive, but it is allocated minimal computing resources. By creating a Condor pool out of all student computing labs we have created a large scale, homogeneous grid resource dedicated to the single purpose of rendering.

By implementing a web portal based interface to submit and monitor rendering jobs the rendering system becomes independent of the end users architecture, OS, and even rendering package being used. This allows for a "learn once, run often" architecture despite the highly complex distributed computational backend. The web interface allows users to submit jobs with the needed parameters, monitor the progress of the job, and display finished frames of the animation. When the job finishes the render system notifies the user and will present a web based preview of the final movie, allowing the user to quickly assess the quality of the rendered results.

The modular nature of the render system allows for easy upgrade and additions to the rendering packages offered. New rendering packages can typically be added within 24 hrs, without any changes to the user interface or workflow.

## Categories and Subject Descriptors

H.3.4 [**Information Storage and Retrieval**]: Systems and Software – *Distributed Systems*

H.3.5 [**Information Storage and Retrieval**]: Online Information Services – *Web-based Services*

K.3.1 [**Computers and Education**]: Computer Use in Education – *Computer-assisted instruction (CAI)*

*John N. Huffman, Richard Repasky, Joseph Rinkovsky, Creating Grid Resources for Use in Undergraduate Coursework, International Workshop on Grid Computing Environments 2007, Nov.,11th, 2007, Reno, NV, USA. An electronic version of this document will be available at http://casci.rit.edu/proceedings/gce2007*

## General Terms

Management, Design.

## Keywords

Portal, animation, condor, education, grid, rendering.

## 1. INTRODUCTION

In many discipline areas, such as creative art, sciences, and business the production of animation movies is an integral part of the research and classroom curriculum [1]. Beyond the creative aspects of producing animations in undergraduate art classes, animations in the form of high definition stereo movies are also being used as a teaching tool, for outreach efforts, and for scientific visualization [2],[3]. At many institutions, adequate resources are not dedicated to the task of animation rendering, which holds true at Indiana University. There are usually a limited number of systems provided for these students to render on, typically a single computer lab of 20-30 systems is set up with the required software. Students find themselves having to limit the length of their movies, or decrease the complexity of their scenes. It is not uncommon to find students staying overnight in a computer lab to utilize multiple machines for rendering.

Animation rendering is an embarrassingly parallel problem, but each subtask can be quite complex. A single rendered image can take as little as a few minutes, to several hours, depending on the complexity of the scene. Since most animations consist of 24-30 frames per second, even creating a movie that spans a few minutes may require thousands of hours of computing time. Since many of the animations produced at Indiana University are stereo movies, the rendering process involves creating two full movies, one for each eye, doubling the computation time needed. Storage can also be an issue, with uncompressed HDTV movies easily reaching the 10s of Gigabytes of storage space. Students are typically only allocated a few GB of persistent disk storage, making it difficult or impossible for them to work with the original uncompress source. Finally, the software being used can also be a particular problem for the user. Since these packages are usually complex and expensive, only a single rendering suite is provided for students to use, and there can be version mismatch errors or compatibility problems with other applications.

To address these issues, we created the Render Portal, Figure 1. This system utilizes the spare cycles on the idle workstations located in the various student computer labs across campus. Jobs can be distributed and rendered without impacting their primary function as student workstations. We utilize the Condor [4] software system for integrating these workstations into a Condor pool. An important requirement for this project was to create an entire grid based rendering system that was easy to use, and can be integrated into the classroom curriculum. While many of the students are computer savvy, most of them are not familiar with the complexity of submitting jobs to a large scale grid based compute resource. Job submission, scheduling, and monitoring are fundamental parts of grid computing, but the typical methods to invoke these services are non-intuitive, and would only serve to discourage novice users such as undergraduate students, from using these resources. To achieve an easy to use interface to this grid based resource, we implemented a web based interface. This portal is based on the Gridsphere [5] portal frame work and utilizes custom portlets for users to submit rendering jobs, monitor/review the job process and when completed download their rendered images. The rendering system takes advantage of a modular package distribution that allows multiple animation packages to be available to the users, including multiple versions of the same application, ensuring greater compatibility and functionality to the user.

## 2. Cluster Back End

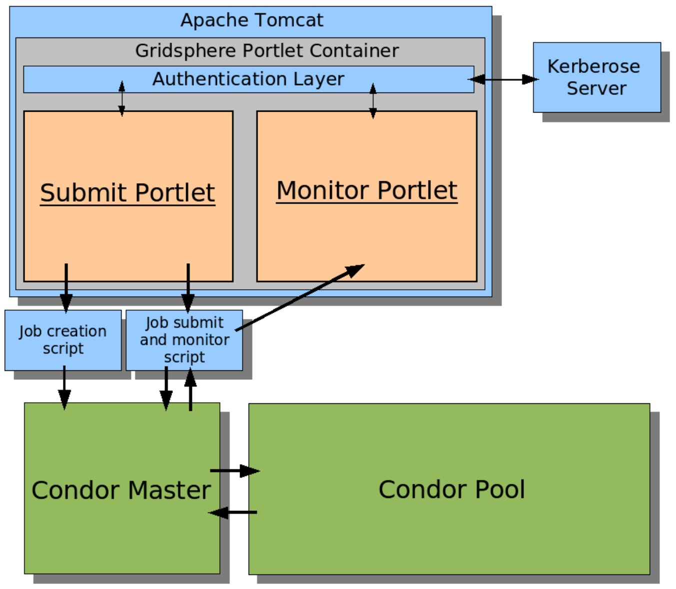


**Figure 1. An overview of the Render Portal framework.**

One of the most important considerations of this project was to have minimal or no impact on the systems being used to render. Since the primary purpose of these systems is student workstations, cycle scavenging was the only practical way to take advantage of these systems. We also decided early on to make minimal changes to these systems.

Condor allows us to work with mixed hardware, work with Windows XP, and to stage both the data and the application to the compute nodes.

Condor receives jobs, manages the queue, and handles application output. We are currently using Condor 2.6.3, which has shown greater scalability over previous versions on the Windows operating system.

Condor components can be broken down into two distinct parts: the Condor compute nodes, composed of the student workstations, and the Condor master node, a dedicated Linux server for the Render Portal.

### 2.1 The Compute Nodes

The Condor pool is composed of more than 2700 Windows XP workstations. These workstations are physically located in various computer labs across Indiana University's Bloomington campus, and are paid for almost entirely through Student Technology Centre (STC) fee, which is a mandatory fee that each student pays. The STC workstations are maintained with a pre-defined set of software, giving us a homogenous computing environment to work from. These systems are also part of a 3 year life cycle program, ensuring that most systems are reasonably fast, and well equipped with memory and disk.

Almost all systems are connected through a 100Mb switching system and are distributed across the entire campus, and therefore rely on local departmental networking.

### 2.2 Master Node

A single Condor Master system manages all jobs for all 2700+ Condor clients. This central point of administration and control lessens the chance of outside jobs being submitted to the condor pool. The master node is a dual 3.0Ghz Intel Xeon workstation with 4Gb of memory and 500 GB of local RAID storage. It is currently running Gentoo Linux 1.1.16. This system also hosts Apache Tomcat and Gridsphere that compose the web interface to this resource, and the MySQL server to store the condor usage statistics.

For storage, we use the local storage on the Condor master for temporary storage of smaller files. For the raw images and final animations we have mounted a large disk based storage system to the Linux box, called the Data Capacitor [6]. This utilizes the Lustre file system [7], and provides over 500TB of spinning disk storage. Files are kept active for 30 days, allowing enough time and disk space for user to complete there projects.

## 3. SYSTEM MODULARITY

To minimize the impact on the STC workstations, we have chosen not to install any of the rendering software packages permanently on these systems. The only software hosted on the STC systems is the Condor client software. The rendering packages and any other extra files are sent to the Condor client as part of the job. This includes a zip file of the rendering package, a decompression tool (unzip.exe), and tools for image post-processing.

The decision to send the rendering package as a zip file, instead of installing the application on every workstation was an important part of the Render Portal. There are several key reasons behind choosing this particular approach:

- Modular packaging frees up local disk space on the student systems when not in use as a rendering node. While system disks are getting larger, space can still be an issue on some of the older systems. Serving as student workstations is the most important task of these systems, so maintaining a small permanent footprint for the Render Portal was a concern.

- Modular packaging minimizes the risk that something can break on any given system. This ensures that all frames are rendered using the exact same software, that doesn't depend on any other installed software.
- Multiple revisions of the same render package can be hosted on the render portal. Often times a user needs a specific feature that is no longer in the newer software or they have an older version of the scene file that doesn't render correctly with the newer version of the rendering software. By sending the entire rendering software with the job, we can retain old versions of the rendering software
- Upgrading the render package can be done much faster, with no impact on the student systems. The task of adding a new rendering package becomes a simple task of adding the rendering software as a zip package, and customizing a few scripts. This typically takes 24-48 hours, depending on any specific dependencies that may arise. This allows for greater flexibility in the system, without having to make any changes on the STC systems.

This approach to render package management is one of the key differences between this system and similar systems, such as the DRE at Purdue University [8]. Despite the obvious overhead to the network, the added functionality greatly improves the scope of the system. We don't have to choose what rendering package to support; we can support any package that works within our framework. This also give a single interface to a wide variety of packages, allowing the student to concentrate more on modeling and animation, and less on the technical aspects of rendering.

## 4. THE WEB INTERFACE

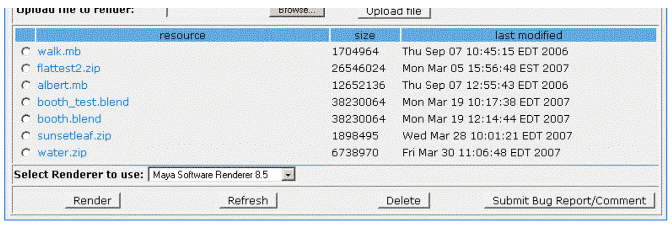
The Render Portal is based on Gridsphere 2.1.5 portal framework (that runs within a Apache Tomcat 5.5.17 servlet container [9]). Tomcat is configured with SSL support, so that all interactions with this web interface are secure. This is primarily to protect the password of the user during login. For authentication to this portal, we use Tomcat's JAAS authentication module and authenticate to Indiana University's central Kerberos server.

The portal users interact with the portal mainly through two JSR.168 [10] compliant portlets: the job submission portlet (Submit Portlet), and the queue monitoring portlet (Monitor Portlet). These two portlets handle the bulk of the tasks that the user performs on the Render Portal.

### 4.1 Submit Portlet

After a user logs in, the Submit Portlet is the initial interface into the Render Portal as seen in Figure 2. This is where the user can upload her/his scene file to their personal workspace, and see what other scene files are already in their workspace. This is also where the user can select which file they want to render, what options are needed for the rendering, what rendering software to use, and then submit the job to be rendered.

After the scene to be rendered has been selected, a page for other options allows the user to select what frames to render, what resolution the output images should be, and what camera within the scene file to use (Figure 3). A user is also asked to estimate


Upload file to render: Browse... Upload file

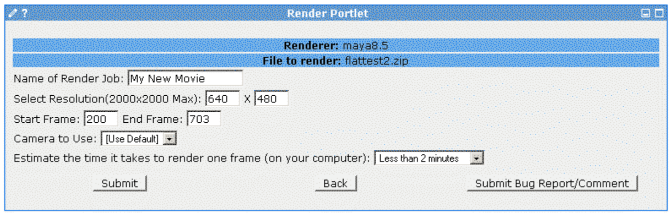
| | resource | size | last modified |
|---|---|---|---|
| ○ | walk.mb | 1704964 | Thu Sep 07 10:45:15 EDT 2006 |
| ○ | flattest2.zip | 26546024 | Mon Mar 05 15:56:48 EST 2007 |
| ○ | albert.mb | 12652136 | Thu Sep 07 12:55:43 EDT 2006 |
| ○ | booth_test.blend | 38230064 | Mon Mar 19 10:17:38 EDT 2007 |
| ○ | booth.blend | 38230064 | Mon Mar 19 12:14:44 EDT 2007 |
| ○ | sunsetleaf.zip | 1898495 | Wed Mar 28 10:01:21 EDT 2007 |
| ○ | water.zip | 6738970 | Fri Mar 30 11:06:48 EDT 2007 |

Select Renderer to use: Maya Software Renderer 8.5

Render Refresh Delete Submit Bug Report/Comment

**Figure 2. The initial Submit Portal screen. This is the interface used to submit scene files to the Render Portal.**

how long they expect for one frame to render on his/her workstation. This allows us to tune how the job is submitted to the Condor pool, and better utilize available resources. We have found that this is much more practical and reliable than using heuristics methods to try to guess a best fit.


Render Portlet

**Renderer:** maya8.5
**File to render:** flattest2.zip

Name of Render Job: My New Movie
Select Resolution(2000x2000 Max): 640 X 480
Start Frame: 200 End Frame: 703
Camera to Use: [Use Default]
Estimate the time it takes to render one frame (on your computer): Less than 2 minutes

Submit Back Submit Bug Report/Comment

**Figure 3. The render options portion of the Submit Portlet.**

### 4.2 Job Submission Scripts

When the user submits the job from the Submit Portlet, a script on the Condor Master system is invoked. This script generates two files, a condor .cmd file to submit to the Condor queue, and a .bat file to run the rendering software and other tasks on the compute nodes (Figure 4).

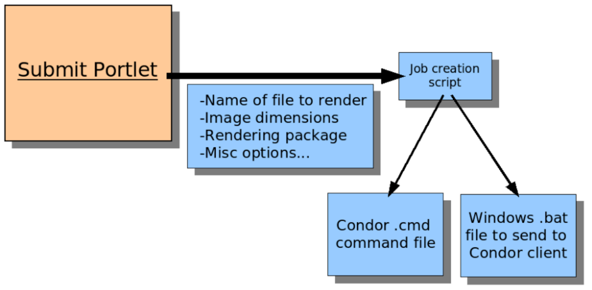
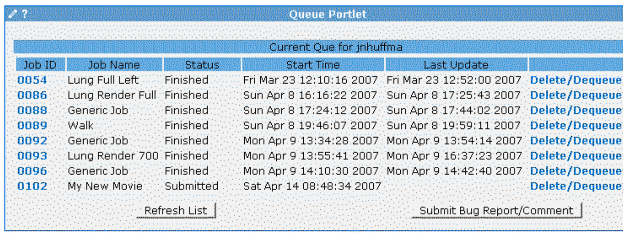


**Figure 4. The job creation script creates the files necessary to submit the render job to the Condor master.**

After the .cmd and .bat files are created, the Submit Portlet runs a second script (run_job) which in turn submits the job to Condor using the .cmd file. The .cmd file breaks the job into chunks of frames to be sent to the Condor clients. The size of each chunk is dependent on the estimated time to complete each frame, and can range from 1-8 frames being computed per Condor client. This .cmd file also specifies which scene file and rendering package to transfer to the client system.

The run_job script utilizes the Condor


Queue Portlet

Current Que for jnhuffma

| Job ID | Job Name | Status | Start Time | Last Update | |
|---|---|---|---|---|---|
| 0054 | Lung Full Left | Finished | Fri Mar 23 12:10:16 2007 | Fri Mar 23 12:52:00 2007 | Delete/Dequeue |
| 0086 | Lung Render Full | Finished | Sun Apr 8 16:16:22 2007 | Sun Apr 8 17:25:43 2007 | Delete/Dequeue |
| 0088 | Generic Job | Finished | Sun Apr 8 17:24:12 2007 | Sun Apr 8 17:44:02 2007 | Delete/Dequeue |
| 0089 | Walk | Finished | Sun Apr 8 19:46:07 2007 | Sun Apr 8 19:59:11 2007 | Delete/Dequeue |
| 0092 | Generic Job | Finished | Mon Apr 9 13:34:28 2007 | Mon Apr 9 13:54:14 2007 | Delete/Dequeue |
| 0093 | Lung Render 700 | Finished | Mon Apr 9 13:55:41 2007 | Mon Apr 9 16:37:23 2007 | Delete/Dequeue |
| 0096 | Generic Job | Finished | Mon Apr 9 14:10:30 2007 | Mon Apr 9 14:42:40 2007 | Delete/Dequeue |
| 0102 | My New Movie | Submitted | Sat Apr 14 08:48:34 2007 | | Delete/Dequeue |

Refresh List Submit Bug Report/Comment

perl library functions to monitor the progress of the submitted job and updates a status file for each job a user submits. This status file is used by the Monitor Portlet to relay to the user the current status of the job being run (Figure 5). When the job is completed, run_job will send a confirmation e-mail to the user. This script is also responsible for any post processing that needs to be done on the image collection.

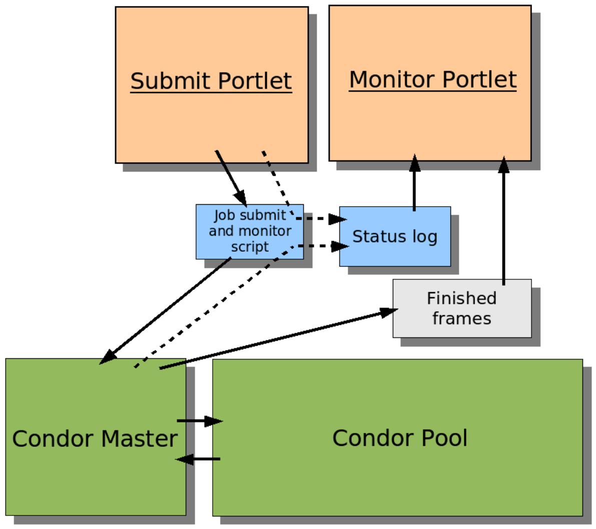


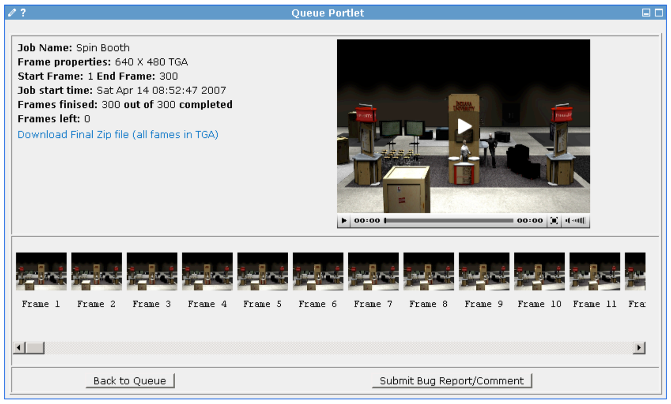
**Figure 5. The job submit script takes care of submitting the job to Condor, as well as creating a log of the progress of the job. This log is read by the Monitor Portlet.**

The .bat file that is sent to the client machines is responsible for initiating several processes: first it will uncompress the rendering software package, the scene file (if it is a zipped file) and some conversion utilities. Then it will set up various environment variables and directories, depending on the need of the rendering software. Once the rendering package is installed, it will then execute the specific commands and parameters to render the image(s). The rendering software will create the image(s) in a predefined format (TGA) and naming scheme. After the rendering is finished, the conversion utilities will create thumb nails of each of the images, and create a JPG image for each frame for preview. It will then clean up all extra files and directories, and only the finished images are sent back to the Condor master.

## 4.3 Monitor Portlet

After the user has submitted their scene to be rendered, he/she can monitor the progress of their job with the Monitor Portlet. This portlet give the user a list of all the jobs that have been submitted, the current status of each one, and time stamp of the last update (Figure 6). This summary screen gives an overview of all jobs that have been submitted by the user, including past jobs. In this way the user can keep track of all the work completed on the Render Portal, and have a repository to get to completed frames.

When a job is selected from the list, the user is presented with thumbnails of all of the finished frames as a filmstrip (Figure 7). Each image in the filmstrip links to the completed frame of the animation, so a user can select individual frames to inspect. If the user detects a problem with the images being rendered, he/she can abort and remove the job from Condor at this point and free up resources. When the job has completed, a flash movie is created on the Condor master system and embedded into the Monitor Portlet so the user can preview the final animation. A URL to a zip file containing all of the uncompressed frames is presented. The user also has the option to recompress the images into several other movie formats (various AVI and QT formats as well as MPEG2) that can be downloaded.



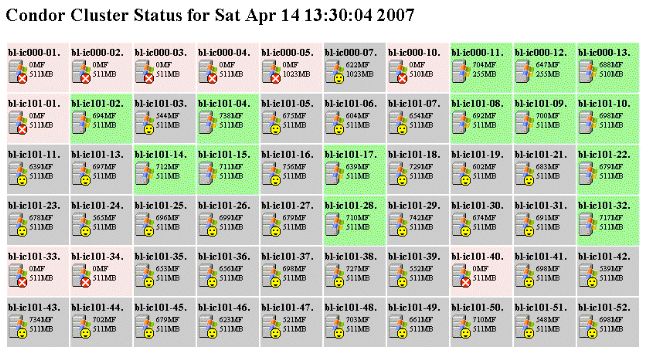
**Job Name:** Spin Booth
**Frame properties:** 640 X 480 TGA
**Start Frame:** 1 **End Frame:** 300
**Job start time:** Sat Apr 14 08:52:47 2007
**Frames finised:** 300 **out of** 300 **completed**
**Frames left:** 0
Download Final Zip file (all fames in TGA)

**Figure 7. When an individual job is selected from the Monitor Portlet, a summary screen of the job is presented. The summary shows all the frames that have been completed, as well as a preview flash movie of the final animation.**

## 5. SYSTEM MONITORING

As part of the Render Portal system we have also created a monitoring system for Condor that presents current and trend information on the status of the Condor pool, as well as the individual systems (Figure 8). Graphs are generated every 15 minutes to show the number of free Condor clients in the pool as well as the clients that are currently rendering. This can help a user determine the best time to submit large jobs, and to work around busy times of the day. This also is a useful tool for administrators to quickly detect and analyzed problems with the Condor pool, and identify problems with individual systems and labs within the Condor pool.

On every page of the Render Portal, a feedback/error report button is integrated into the interface. This button takes the user to a simple form that lets him/her to quickly write and submit a comment. The users are encouraged to use this anytime they encounter a problem, or have a suggestion. When the report is sent, all state information for the current portlet is kept and sent with that e-mail, making it easier for the administrators to diagnose and correct any errors that may have occurred.



**Condor Cluster Status for Sat Apr 14 13:30:04 2007**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| bl-ic000-01. 0MF 511MB | bl-ic000-02. 0MF 511MB | bl-ic000-03. 0MF 511MB | bl-ic000-04. 0MF 511MB | bl-ic000-05. 0MF 1023MB | bl-ic000-07. 622MF 1023MB | bl-ic000-10. 0MF 510MB | bl-ic000-11. 704MF 255MB | bl-ic000-12. 647MF 255MB | bl-ic000-13. 688MF 510MB |
| bl-ic101-01. 0MF 511MB | bl-ic101-02. 694MF 511MB | bl-ic101-03. 544MF 511MB | bl-ic101-04. 738MF 511MB | bl-ic101-05. 675MF 511MB | bl-ic101-06. 604MF 511MB | bl-ic101-07. 654MF 511MB | bl-ic101-08. 692MF 511MB | bl-ic101-09. 700MF 511MB | bl-ic101-10. 698MF 511MB |
| bl-ic101-11. 639MF 511MB | bl-ic101-13. 697MF 511MB | bl-ic101-14. 712MF 511MB | bl-ic101-15. 711MF 511MB | bl-ic101-16. 756MF 511MB | bl-ic101-17. 639MF 511MB | bl-ic101-18. 729MF 511MB | bl-ic101-19. 602MF 511MB | bl-ic101-21. 683MF 511MB | bl-ic101-22. 679MF 511MB |
| bl-ic101-23. 678MF 511MB | bl-ic101-24. 565MF 511MB | bl-ic101-25. 696MF 511MB | bl-ic101-26. 699MF 511MB | bl-ic101-27. 679MF 511MB | bl-ic101-28. 710MF 511MB | bl-ic101-29. 742MF 511MB | bl-ic101-30. 674MF 511MB | bl-ic101-31. 691MF 511MB | bl-ic101-32. 717MF 511MB |
| bl-ic101-33. 0MF 511MB | bl-ic101-34. 0MF 511MB | bl-ic101-35. 653MF 511MB | bl-ic101-36. 656MF 511MB | bl-ic101-37. 698MF 511MB | bl-ic101-38. 727MF 511MB | bl-ic101-39. 552MF 511MB | bl-ic101-40. 0MF 511MB | bl-ic101-41. 698MF 511MB | bl-ic101-42. 539MF 511MB |
| bl-ic101-43. 734MF 511MB | bl-ic101-44. 702MF 511MB | bl-ic101-45. 679MF 511MB | bl-ic101-46. 623MF 511MB | bl-ic101-47. 521MF 511MB | bl-ic101-48. 703MF 511MB | bl-ic101-49. 661MF 511MB | bl-ic101-50. 710MF 511MB | bl-ic101-51. 548MF 511MB | bl-ic101-52. 698MF 511MB |

## 6. RELATED AND FUTURE WORK

The Render Portal extends progress that has been made by previous work in which distributed, surplus computing cycles have been harvested to render animations. Madhavan et al. [8], [12] built a Condor-based rendering system in which users submitted jobs through a rendering client that was equipped with a plug-in that created job descriptions and passed them to Condor. Compute nodes in the rendering system were equipped with rendering clients. Our Render Portal implements a more general framework in several respects. Its modular design provides users with a choice of rendering packages, and it allows new rendering packages to be easily added in the future. Modularity of the Render Portal is enhanced because the system distributes both data and the application to compute nodes rather than distributing only data. Also, the Render portal frees users from requirements of particular operating systems or client software other than a web browser. This independence and, indeed, the modularity of the Render Portal come at the expense of a single interface in which users can both render scenes and combine them to form a complete movie. In the future we plan to add to the Render Portal tools that allow students to combine multiple frame sets and encode them into complete movies.

We are also exploring the idea of providing a venue in which students can share their work with their peers and rate each other's work. This, we believe, would cultivate interest in the Rendering Portal and build a community of users.

We plan to increase the portability of the Render Portal. Presently, the server that hosts the web interface (Tomcat/Gridsphere) must also host the Condor Master software, restricting access to Condor to a single server. Access can be expanded if the web server is decoupled from the Condor Master. For example, we could offer portlets in a variety of JSR.168 compliant containers such as Sakai [11] that are used to deliver course materials at Indiana University. We plan to decouple the portal server from the Condor Master by implementing web services within the Render Portal, and on the Condor Master.

Finally, the Render Portal is being introduced as a TeraGrid [12] resource, so researchers and educators anywhere in the US can use this rendering system. To make the Render Portal available on the TeraGrid, we are adding TeraGrid authentication to the portal. Grid credentials will be obtained from a TeraGrid Keberos server at the Pittsburgh Supercomputing Center. Instructors will be able to request accounts for themselves and their students.

## 7. CONCLUSION

The Render Portal has been a well received resource by the students. Before the portal was introduced, rendering time was an important consideration when a student animation project was started, but with the power and ease of use of this system, rendering has become an afterthought.

A 10 second benchmark movie takes 40 minutes per frame to render on a high end, dual CPU workstation at a resolution of 1280x768. There are a total of 300 frames, so the entire movie will take 12,000 minutes, or about 200 hours to render. On the Render Portal, it takes just over two hours to complete the entire rendering.

As a comparison, the rendering of the same movie was also done manually by the students within the computer labs using 30 computers. This took 9 hours to complete (this does not include the time to assemble all the images into one location).

The students have been very enthusiastic about the Render Portal, and within the first week of beta testing the entire animation class signed up to use the system. Currently, the Render Portal is generating over 10,000 frames per week.

## 8. ACKNOWLEGEMENTS

We thank Marlon Pierce at Indiana University Pervasive Technology Labs and Yu Ma at IU University Technology Services for the technical guidance they have given to the Render Portal project. We would like to thank Margaret Dolinsky IU H.R. Hope School of Fine Arts and her art students for providing critical feedback during the initial software deployment. We would like to give special thanks to Kianosh Huffman, IU Pervasive Technology Labs, for constructive criticism on all aspects of this project. This work was funded in part by NSF grant 0504075 to Craig Stewart, Indiana University.

## 9. REFERENCES

[1] W.P. Flanagan, R. Earnshaw, "Visual Learning for Science and Engineering", *IEEE Computer Graphics and Applications*, Dec. 2004

[2] D. Cox, "Visualization and Virtual Reality for Art and Science", *Computer Graphics International 1999*, p. 34, June 1999.

[3] (2006) An Animation Company takes Harvard University Students on a 3D Journey, [Online] Available: http://www.xvivo.net/-press/harvard_university.htm

[4] M. Litzkow, M. Livny, M.Mutka, "Condor – A Hunter of Idle Workstations" in *Proc. Of the 8th International Conference of Distributed Computing Systems*, pp 104-111, June 1998.

[5] J. Novotny, M. Russell, O. Wehrens, "GridSphere: an advanced portal framework" in *Proc. 30th EUROMICRO Conference*, pp 412-419, Aug. 2004.

[6] (2007) The Data Capacitor website. [Online]. Available: http://www.datacapcitor.org/

[7] (2006)The Luster Filesystem website. [Online]. Available: http://www.lustre.org/

[8] K.P.C. Madhaven, L.L. Arns, G.R. Bertoline, "A Distributed Rendering Environment for Teaching and Scientific Visualization" *IEEE Computer Graphics and Applications*, vol. 25, Issue 5, pp 32-38, Sept. 2005

[9] (2007) The Apache Tomcat website. [Online]. Available: http://tomcat.apache.org

[10] (2007) JSR-000168 Portlet Specification - Final Release. [Online], Available: http://www.jcp.org/aboutJava/communityprocess/final/jsr168/

[11] X. Yang, X.D. Wang, R. Allan, M. Dovey, M. Baker, R. Crouchley, A. Fish, M. Gonzalez, T. van Ark, "Integration of Existing Grid Tools in Sakai VRE", in *Proc. Fifth international Conference on Grid and Cooperative Computing Workshops, 2006*, pp 13-21, June 2006

[12] S.L. Gooding, L.L. Arns, P. Smith, J. Tillson "Implementation of a Distributed Rendering Environment for the TeraGrid", *IEEE Challenges of Large Applications in Distributed Environments*, 2006, pp 13-21, June 2006

[13] (2007) The TeraGrid website. [Online]. Available: http://www.teragrid.org