

The PGPGrid Project: Wide Area Rendering Environment

Ali Anjomshoaa, Kostas Kavoussanakis, Gavin J. Pringle
EPCC, The University of Edinburgh, UK

All Hands 2004 Conference, Poster Paper

Abstract

The PGPGrid project aims to apply Grid technologies to the production of computer-generated animation. This involves undertaking the compute-intensive processes of modelling and rendering by employing Grids in a Virtual Organisation setting. The project will attempt to implement a *Wide-Area Rendering Environment* (WARE) that will allow the exploitation of remote rendering farms. This involves the design and implementation of a *Remote Rendering System* (RRS) based on Java and Web Services. This paper presents the high level designs of the WARE and RRS and the experience gained from the implementation of a prototype based on these designs.

Introduction

This paper describes the high level design of the Pepper's Ghost Productions Grid (PGPGrid) project's *Wide Area Rendering Environment* (WARE) and the experience gained from the prototype implementation of this design.

The PGPGrid Project

The PGPGrid project [1, 2] is a collaboration between:

- Pepper's Ghost Productions Ltd. (PGP) [3];
- 3D-Matic Labs of The University of Glasgow (3D-Matic) [4]; and
- EPCC of The University of Edinburgh [5].

The PGPGrid Virtual Organisation (VO) aims to apply Grid technologies to the production of computer-generated animation (CGA), where a unique method of motion capture is applied to the animation process. This will enable the VO to study the feasibility of this approach to CGA and the possible impact of Grid technologies on this industry. More specifically, the aims of the PGPGrid project are to determine:

- whether or not the formation of VOs, such as the PGPGrid VO, is a feasible method for future CGA production;
- whether the unique motion capture technology offered by 3D-Matic [6] can be used within an animation VO as part of the animation process; and

- whether it is possible to undertake the animation processes of modelling and rendering, which are required by 3D-Matic and PGP, within a Grid environment using existing Middleware technologies.

With respect to utilising a Grid environment, the intention is to determine whether this will enable the use of distributed computing resources for CGA. In addition, we aim to assess whether CGA processes may be controlled remotely when run on such a Grid. Finally, we look to identify the particular customisations that are required of current CGA infrastructures and workflows in order to take advantage of a Grid environment.

The PGPGrid Wide Area Rendering Environment

The design and implementation of PGPGrid's WARE will apply the ideas behind Grids to the rendering requirements of the PGPGrid project. These requirements include:

- the on-demand use of distributed and idle resources for the compute-intensive rendering work of PGPGrid;
- the use of a Grid interface for the submission of rendering jobs to a set of remote rendering farms over a network;
- the ease of use of this Grid interface from the animator's point of view at PGP; and

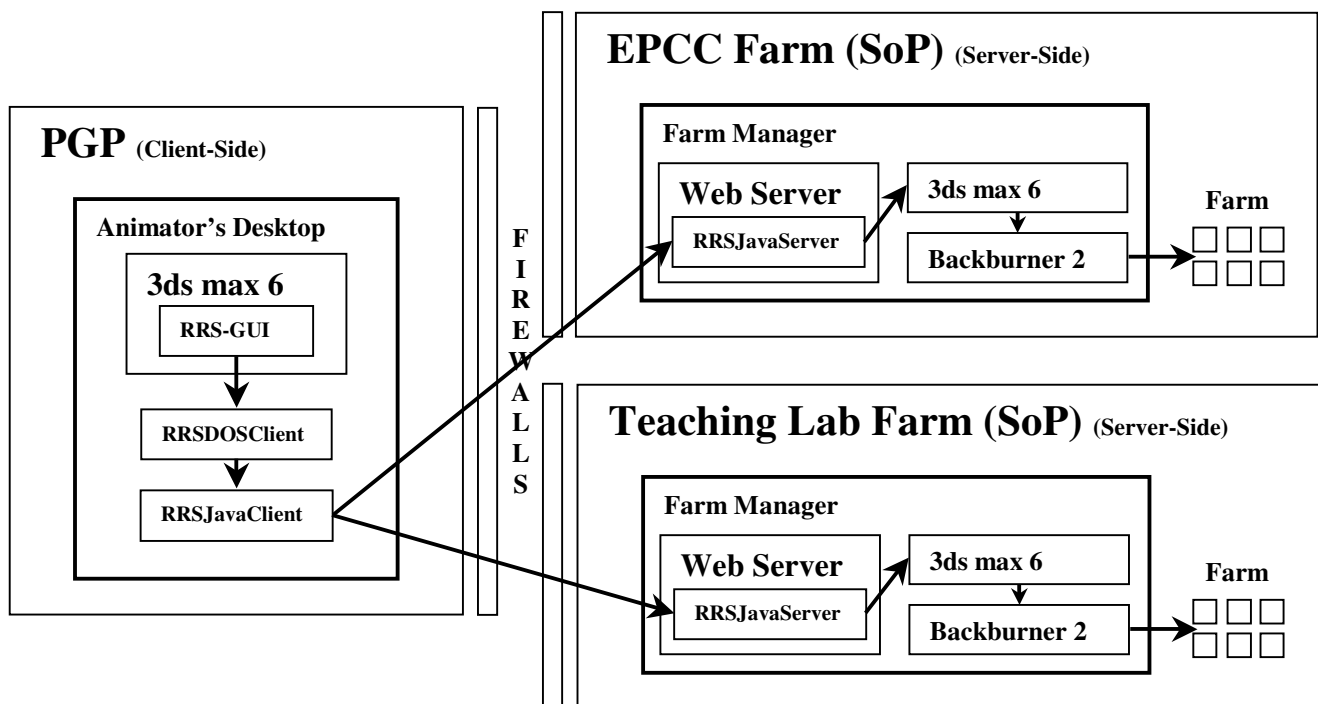


Figure 1: The deployment of the RRS client and server applications at the PGP domain (left) and the remote rendering farm domains (right), respectively. This figure shows the relationship between an animator's desktop machine at PGP and the deployed rendering farms at the SoP, the gateways for which are their farm manager machines.

- the transport of data files between the animator's domain and the remote rendering farms.

The rendering software that will be integrated into the WARE is that of the *3D Studio Max* software suite [7]: PGP's tool of choice. Existing 3D Studio Max features must be used for this integration. These features include:

- the command-line interface of the software as a means of hooking into 3D Studio Max from external application layers; and
- the use of the Backburner batch system [7], consisting of a manager, a monitor, and multiple server modules in a farm, which is provided with the 3D Studio Max software.

The WARE will be enabled by a *Remote Rendering System* (RRS) application layer and will encompass the PGP and the remote rendering farm domains. The RRS will be developed as a deliverable of the PGPGrid project.

The RRS will consist of a *client-side* application and a *server-side* application. The client-side application will be used by

animators to initiate and monitor rendering jobs, and will reside on the animator's desktop machine at PGP. The server-side application, on the other hand, will reside at the remote rendering farms, as shown in Figure 1, to which rendering jobs will be submitted from the RRS client.

The remote rendering farms will use the 3D Studio Max rendering module and the Backburner batch system for rendering job management. Current plans include the deployment of at least two independent farms at the School of Physics (SoP) of the University of Edinburgh. These will include EPCC's desktop PC cluster and one of the teaching laboratories at the SoP, as shown in Figure 1. The configuration of the latter will be a close copy of the EPCC render farm in terms of the deployment of the 3D Studio Max software and PGPGrid's render farm WARE components, including the RRS server.

The WARE and, therefore, the RRS, will be based on a Web Services infrastructure, with the aim of using GridFTP [8] for the transfer of associated data files. The reasons for choosing pure Web Services over Grid Services include:

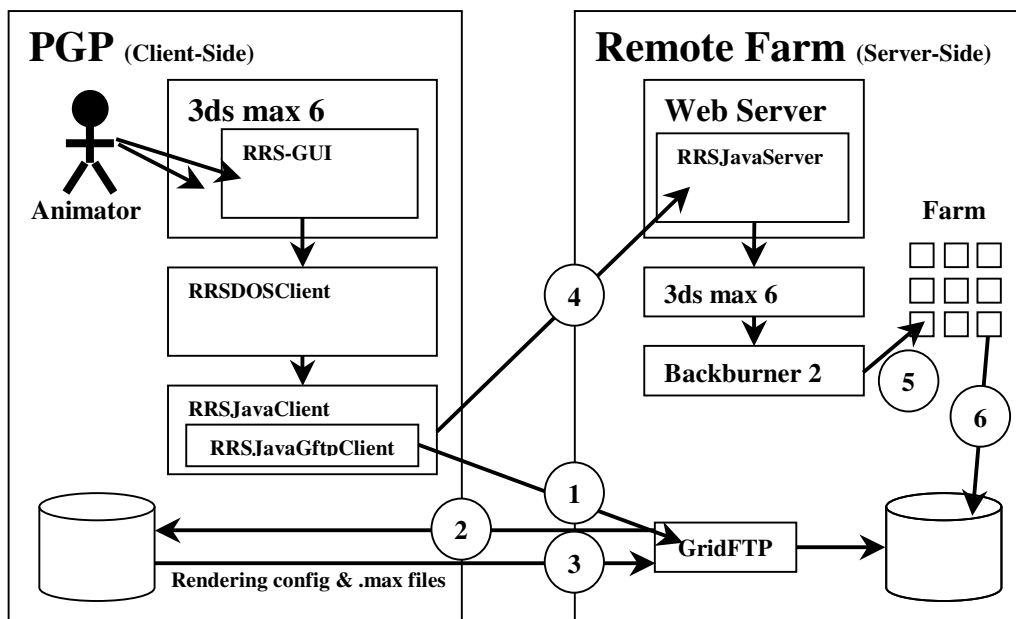


Figure 2: The WARE components on the client-side (left) and the server-side (right), showing the sequence of events for the submission, queuing, and execution of rendering jobs. The RRSJavaClient application invokes its embedded RRSJavaGftpClient module, which in turn invokes the GridFTP server at the remote rendering farm to copy over required files from the client-side to the server-side (1, 2 and 3). The RRSJavaClient then invokes the RRSJavaServer Web Service to initiate and monitor rendering jobs (4). Once a rendering job has been submitted to the Backburner manager, through the command-line interface of 3D Studio Max (3ds max 6), the job is queued before individual frames are submitted to the server machines on the farm for rendering (5). For each frame being rendered, the 3D Studio Max application running on that server will access the files required, which are stored locally at the farm (6).

- the amount and quality of support available for the design and building of Web Service based distributed systems and computing environments;
- the immaturity of current Grid Service specifications that are being drawn up in the specifications of the Web Services Resource Framework (WSRF) [9, 10] that are being developed in OASIS [11]; and
- the instability of Grid Service based Middleware such as the Globus Toolkit [12].

Java will be used for the development of the RRS due to the large amount of support and the number of application libraries that are available for the development of Web Services.

The WARE Components

The WARE will encompass the PGP and the remote rendering farm domains. Figure 2 shows the components of the WARE.

Client-Side

The client-side, hosted at PGP, will include a RRS-GUI. This GUI will be integral to the 3D Studio Max application and will be implemented using its plug-in scripting language, namely MAXScript. Hence, the RRS-GUI will provide animators with an interface to the RRS client in a familiar environment. Rendering job configuration, initiation and monitoring will be undertaken by the animator using the 3D Studio Max application and the RRS-GUI.

The RRS-GUI invokes the RRS client through an executable Windows batch file (RRSDOSClient), passing it arguments appropriate to the action required by the animator. This will, in turn, invoke a Java RRS client application (RRSJavaClient) with those arguments, which, in turn, calls the RRS server application through the RRS server Web Service (RRSJavaServer) hosted at each of the remote rendering farms.

The use of the executable Windows batch file allows any environment set-up to be made at the client-side before the RRSJavaClient application is launched. This will allow easy client-side environment customisation for the RRS client, while allowing for a standard RRS-GUI and RRSJavaClient application for all client-side hosts regardless of environment requirements.

Server-Side

For rendering job submission, the RRSJavaServer Web Service calls the command-line interface of the 3D Studio Max application with the set of arguments that are sent to it by the RRSJavaClient application. The RRSJavaServer Web Service will also perform custom job monitoring functions that are outwith the 3D Studio Max application's command-line capabilities.

The sequence of events leading to the rendering of CGA frames on a remote farm are illustrated in Figure 2.

In the current design of the WARE, animators must select specific remote rendering farms for the submission of rendering jobs. Future versions of the design may include a Grid scheduling element to alleviate this need.

The WARE Prototype

As of July 2004, the prototype of the PGPGrid WARE consists of the components shown in Figure 2, except the file transfer components, including the use of a GridFTP server and client. It is intended that the GridFTP client will form part of the RRSJavaClient application (RRSJavaClient) and will be enabled via the Java CoG Kit [13]. As such, the current prototype requires the manual transfer of the necessary files for rendering, to and from the rendering farm. The use of GridFTP within the WARE architecture is being further investigated, with a planned prototype of the full system in August 2004.

This proof of principle prototype has shown that the designs of the WARE and RRS can be implemented and are suitable for a custom Grid environment required for the exploitation of remote rendering farms.

Acknowledgements

The PGPGrid project is jointly funded by the UK's Department of Trade and Industry's (DTI) e-Science Grid Core Programme (GCP), the UK's Engineering and Physical Sciences Research Council (EPSRC) and Pepper's Ghost Productions Ltd.

References

- [1] P. Cockshott *et al.*, Pepper's Ghost Productions Grid (PGPGrid) Project Proposal, National e-Science Centre Project Proposal, 2002.
- [2] The PGPGrid project web site:
<http://www.epcc.ed.ac.uk/pgpgrid/>
- [3] The Pepper's Ghost Productions Ltd. web site:
<http://www.peppersghost.com/>
- [4] The 3D-Matic Laboratory of The University of Glasgow web site:
<http://www.faraday.gla.ac.uk/>
- [5] The EPCC of The University of Edinburgh web site:
<http://www.epcc.ed.ac.uk/>
- [6] W. P. Cockshott, S. Hoff, J-C. Nebel. An Experimental 3D Digital TV Studio, IEE Proceedings – Vision, Image & Signal Processing, Institute of Electrical Engineers, 2003.
- [7] The 3D Studio Max (3ds max) software from Discreet:
<http://www.discreet.com>
- [8] The Globus Alliance web site on GridFTP:
<http://www.globus.org/datagrid/gridftp.html>
- [9] The Organization for the Advancement of Structured Information Standards (OASIS) web site on WSRF:
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf
- [10] The Globus Alliance web site on WSRF:
<http://www.globus.org/wsrf/>
- [11] The Organization for the Advancement of Structured Information Standards (OASIS) web site:
<http://www.oasis-open.org/home/index.php>
- [12] The Globus Alliance web site:
<http://www.globus.org/default.asp>
- [13] The Globus Alliance web site on the Java CoG Kit:
<http://www-unix.globus.org/cog/java/>