

PGPGrid: Wide Area Rendering Environment

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

Project partners: Pepper's Ghost Productions Ltd. (PGP) [1]; 3D-Matic Labs, The University of Glasgow (3D-Matic) [2]; EPCC, The University of Edinburgh [3]

Abstract

PGPGrid aims to apply Grid technologies to the production of computer-generated animation. This will involve compute-intensive modelling and rendering using Grids in a virtual organisation setting. The project is implementing a wide-area rendering environment (WARE) to exploit remote rendering farms – this involves the design and implementation of a remote rendering system (RRS) based on Java and Web Services. Here we present the high level designs of the WARE and RRS and the experience gained from the implementation of a prototype based on these designs.

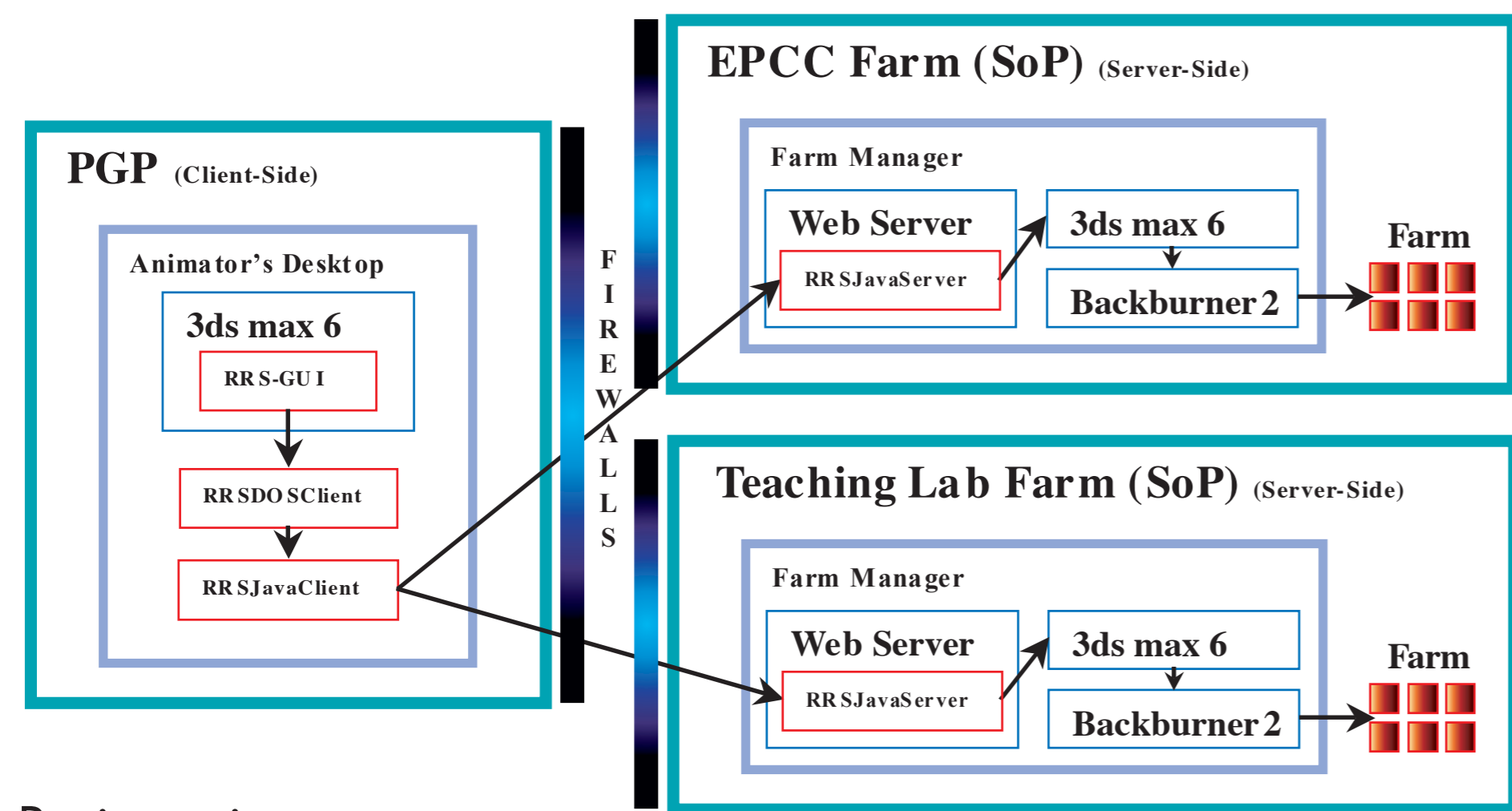


Figure 1: The deployment of the RRS client and server applications at the PGP domain (left) and the remote rendering farm domains (right), respectively. The farm manager machines act as the gateways to the remote rendering farms.

Project aims

The PGPGrid virtual organisation (VO) aims to determine:

- whether the formation of VOs is feasible for future computer-generated animation (CGA);
- whether the motion capture technology from 3D-Matic [4] can be used as part of the animation process;
- whether it is possible to undertake the animation processes of modelling and rendering within a Grid environment;
- whether CGA processes may be controlled remotely when run on such a Grid.

Wide Area Rendering Environment (WARE)

The design and implementation of PGPGrid's WARE will apply the ideas behind Grids for:

- the on-demand use of distributed, idle resources (in rendering farms) for compute-intensive rendering work;
- the use of a Grid interface for submission of rendering jobs to a set of rendering farms;
- the ease of use of a Grid interface from the animator's point of view;
- the transport of data files between the animator's domain and the rendering farms.

3D Studio Max Software

The 3D Studio Max software suite [5] will be used for rendering. It will be integrated into the WARE. Existing 3D Studio Max features are used for this integration.

Remote rendering system (RRS)

The WARE will be enabled by a remote rendering system (RRS) application layer encompassing the PGP and remote rendering farm domains. The RRS will consist of client-side and server-side applications (Figure 1). The client-side application will be used by animators to initiate and monitor rendering jobs, which will be submitted to the remote rendering farms via the server-side application (Figure 2).

Web Services and Java

The WARE and, therefore, the RRS, will be based on a Web Services infrastructure, with the aim of using GridFTP [6] for the transfer of associated data files. Java will be used for the development of the RRS.

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 communicates with 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, 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 the server will access the files required, which are stored locally at the farm (6).

The WARE components

The WARE components are shown in Figure 2.

Client-Side

- The RRS-GUI is integrated into 3D Studio Max and invokes the RRS client through RRSDOSClient, passing it arguments appropriate to the action required by the animator.
- The RRSDOSClient will invoke the RRSJavaClient with those arguments.
- The RRSJavaClient then calls the RRSJavaServer Web Service at each of the remote rendering farms.

Server-Side

- The RRSJavaServer Web Service calls the command-line interface of 3D Studio Max for job submission.
- This Web Service also performs custom job monitoring functions outwith the 3D Studio Max application's command-line capabilities.

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 perform this task automatically.

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. The use of GridFTP within the WARE architecture is being further investigated, with a prototype of the full system being implemented in August 2004.

The current, proof of principle prototype has already 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 using 3D Studio Max.

References

- [1] Pepper's Ghost Productions Ltd: www.peppersghost.com/
- [2] 3D-Matic Laboratory of The University of Glasgow: www.faraday.gla.ac.uk/
- [3] EPCC, The University of Edinburgh: www.epcc.ed.ac.uk/
- [4] 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.
- [5] 3D Studio Max (3ds max) software from Discreet: www.discreet.com
- [6] Globus Alliance website on GridFTP: www.globus.org/datagrid/gridftp.html

