Integrating a Common Visualization Service into a Metagrid.

R. Watson[1], S. Maad[1], and B. Coghlan[1]

Trinity College Dublin, Dublin, Ireland,
watsonr@cs.tcd.ie,
WWW home page: http://www.cs.tcd.ie/∼watsonr

Abstract. Existing architectures and tools for visualization on the Grid (Virtual Reality:VR, Augmented Reality:AR) include: CrossGrid [1], glogin [2], GVK [3], Migrating Desktop [4] and Gvid [5]. Whereas these architectures were useful for some applications they cannot be considered as complete. Our paper proposes a complete architecture that provides a more generic solution for applications involving visualization and simulation. The paper presents the architecture and scenarios for interoperability with existing grids.

1 Introduction

There are numerous visualization applications that may benefit from the Grid's computational power but are not doing so because of missing supporting grid infrastructure. Here a basic architecture is proposed to serve these applications and enhance their performance. A generic architecture that can be used by all visualization applications is required. The visualization pipeline can be conceived in several ways [6]. In this paper the visualization pipeline is considered as a sequence of four tasks, computation, interaction, rendering and display. The Grid can handle the computation and delegate the data produced from the Grid to a dedicated visualization engine which can then stream the images to a display. For it to be a common facility for arbitrary Grids, the visualization engine should be a part of a metagrid infrastructure.

This paper is divided into six sections including this one. The second section overviews existing grid visualization solutions and their limitations. The third section states the objectives of our architecture while the fourth and fifth sections describe the visualization architecture and it's interoperability with the Grid. The paper concludes with a summary and some comments about further improvements.

2 Literature Overview

In this section a quick overview of the technologies, architectures and tools used so far in grid visualization will be discussed and analysed. The limitations of those solutions are highlighted and the degree of success of these solutions in various applications is examined.

2.1 Technologies

This work derives from involvement in the EU CrossGrid project [1] which was oriented towards compute and data-intensive applications that involve the interaction of a user in the processing loop. Such applications require a response from the Grid to an action by a human agent in different time scales. Tools developed within the CrossGrid project are glogin, Gvid, GVK, and the Migrating Desktop. These are briefly described here.

glogin: This tool provides a tunnel into the Grid and therefore facilitates interaction with the Grid's resources.

Gvid: Allows rendering to be done on Grid resources, with transmission of resulting video over the Grid.

GVK: With GVK the user is able to control the execution of a grid application by installing a bi-directional interactive link between the scientific application and the visualization tool. The link itself is established using the glogin tool

Migrating Desktop: The Migrating Desktop is a framework for a graphical user interface for application management, grid and job monitoring, data and metadata management. This graphical environment is used as an advanced client for accessing grid resources in CrossGrid.

2.2 Applications

The above CrossGrid visualization tools have been applied in various domains including health and environment. These are briefly overviewed below.

Blood Flow Simulation: A Grid-based prototype system for pretreatment planning in vascular interventional and surgical procedures through real-time interactive simulation of vascular structure and flow. The system consists of a distributed real-time simulation environment, with which a user interacts in Virtual Reality (VR). A 3D model of a patient's arteries, derived using medical imaging techniques, serves as input to the environment for blood flow calculations.

Flood Crisis System: Grid-enabled simulations of three physical systems pertinent to flood crisis management: meteorology, hydrology and hydraulics. The main component of the system is a highly automated early warning system, based on hydro-meteorological (snowmelt) rainfall-runoff simulations.

2.3 Limitations

In each of the examples above there is a problem in that the application developer's task is to write application-specific plugins that can communicate with the appropriate web services. Is there away around this problem? For every application a plugin is required. This plugin can be very specific to the type of simulation being run within the visualization application. For widely used software, like VTK, it may be possible to have a range of plugins available, but it would be better if no extra software was needed on the application side.

Other interactive visualization applications have used tools such glogin to tunnel into the Grid and run the applications on the Grid through this pseudo terminal. This indeed is running an interactive application on the Grid but there are two problems with this. Firstly, when using common middleware such as Globus [7], LCG2 [8] or EGEE [9] the application has to be installed on the "gatekeeper" that the user uses glogin to connect to. Secondly this application is running on the gatekeeper and not a general compute node.

3 Objectives

In this section the objectives of the architecture will be described. Fundamental issues addressed are:

1. *Providing a complete architecture.*
2. *Covering most of the visualization applications.*
3. *Imposing minimum re-engineering costs.*
4. *Interoperating with all Grids.*
5. *Use of the Grid for the computation.*

What is needed is a complete architecture that will deal with visualization applications that are compute intensive, allowing for interaction and performance increases. Not restricting the architecture to a particular Grid infrastructure is also an important objective, and so we target a metagrid rather than a specific Grid. We intend that the visualization engine should be part of a metagrid infrastructure. This visualization engine is described in the next section.

4 Visualization Architecture

Visualization is generically broken down into four parts,

- Computation
- Interaction
- Rendering
- Display

While the Grid succeeded so far in offering computational power, it hasn't provided a universal solution for interaction, rendering or display. So in our architecture the main focus is on these three aspects. Existing solutions have tackled one or more of these issues to some extent, but not all three together. For interaction, glogin is a very adequate solution that gives the user a direct login connection into compute nodes. Rendering, on the other hand, traditionally depends on the local workstation at which the user is at. In some cases the visualization application is run on the gatekeeper and glogin is used to view its output. This is not ideal as the graphical capabilities of the gatekeeper are unknown, and more importantly that is not the job of the gatekeeper.

In our architecture we suggest that a dedicated visualization engine be put in place to deal with jobs that require more power in rendering their output. The head node of the visualization engine is added to a gatekeeper queue. By way of example we have constructed a Visualization Engine comprising of the following components:

– 9-node VRengine
– WorkerNode Software
– Scalable Coherent Interconnect (SCI)
– Chromium
– AccessGrid

Chromium [10] is a system for manipulating streams of graphics API commands on clusters of workstations. Chromium's stream filters can be arranged to create sort-first and sort-last parallel graphics architectures that, in many cases, support the same applications while using only commodity graphics accelerators. In addition, these stream filters can be extended programmatically, allowing the user to customize the stream transformations performed by nodes in a cluster.

We have incorporated a high-speed interconnect, SCI [11], configured in a 3-d torus. At present the only protocol that is supported by Chromium is TCP/IP, so the Dolphin SuperSockets [12] is used to provide a fast and transparent way for TCP/UDP/IP to use SCI as the transport medium. The major benefits are a high bandwidth and much lower socket latency than network technologies like Gigabit Ethernet, Infiniband and Myrinet.

Finally, for the display it is important that streams of rendered images be conveyed from the visualization engine to the (remote) user via widely used protocols that have good supporting display software. We have chosen to adopt the AccessGrid [13] streaming protocols for this purpose. This has the distinct advantage that the visualization output stream can be incorporated into AccessGrid video conferences.

5 Interoperability with Grid

Various job submission will be explored in this section that involve visualization.

Scenario 1: In this scenario the user submits a job to a resource broker. The simulation is then sent to one or more compute nodes by the resource broker.

Scenario 2: Involves the creation of a wrapper around the resource broker which would split the job into two parts. The first computation part would be sent to the compute nodes and the second part would be sent to the visualization engine.

Scenario 3: Involves a modification of the job description with added information about the visualization pipeline.

6 Comments

In this paper we have presented the integration of a visualization service into a metagrid and have presented three scenarios of job submission where this visualization engine could be used.

References

1. The EU CrossGrid Project, http://www.eu-crossgrid.org
2. H. Rosmanith, D. Kranzlmüller, *glogin - A Multifunctional, Interactive Tunnel into the Grid*, In Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, Pittsburgh, PA, USA, November 2004.
3. D. Kranzlmüller, P. Heinzlreiter, H. Rosmanith, J. Volkert, *Grid-Enabled Visualization with GVK*, Lecture Notes in Computer Science. Vol. 2970, 2004, pp.139–146
4. M. Kupczyk, R. Lichwala, N. Meyer, B. Palak, M. Plociennik, and P. Wolniewicz, *Roaming Access and Migrating Desktop*, Proceedings of The 2nd Cracow Grid Workshop, Cracow, Poland, December 2002, pp. 148-154
5. P. Heinzlreiter, *Interactive Result Visualization on the Grid*, Presentation at Grid Computing for ComplexProblems Bratislava, Slovakia, 29. November 2005.
6. R.B. Haber and D.A. McNabb, *Visualization Idioms: A Conceptual Model for Scientific Visualization Systems*, in G.M. Nielson, B. Shriver, and L.J. Rosenblum, (Eds.): Visualization in Scientific Computing, IEEE Computer Society, Los Alamitos, NM, USA, pp. 7493 (1990).
7. The Globus Project, http://www.globus.org/
8. LHC Computing Grid Project, http://lcg.web.cern.ch/LCG/
9. Enabling Grids for E-science in Europe, http://www.eu-egee.org/
10. G. Humphreys, M. Houston, Y. Ng, R. Frank, S. Ahern, P. Kirchner and J.T. Klosowski, *Chromium: A Stream Processing Framework for Interactive Graphics on Clusters*, ACM SIGGRAPH, July 2002.
11. Scalable Coherent Interface (SCI), ANSI/IEEE Standard, 1596-1992, August 1993.
12. F. Seifert and H. Kohmann, *SCI SOCKETS - A Fast Socket Implementation over SCI*, http://www.dolphinics.com/pdf/whitepapers/sci-socket.pdf
13. AccessGrid, http://www.accessgrid.org/