EG UK Theory and Practice of Computer Graphics (2010)
John Collomosse, Ian Grimstead (Editors)

Remote Scientific Visualization for Large Datasets

M. Turner, G. Leaver and J. Perrin

Research Computing, University of Manchester, UK

Abstract

Remote scientific visualization, where rendering services are provided by larger scale systems than are available on the desktop, are becoming increasingly important as dataset sizes increase beyond the capabilities of desktop workstations. Uptake of such services relies on access to suitable visualization applications and the ability to view the resulting visualization in a convenient form. We apply five rules from the e-Science community to meet these goals with the porting of a commercial visualization package to a large scale system and the integration of this code with the Access Grid. Example use cases from Materials Science are considered.

Categories and Subject Descriptors (according to ACM CCS): I.3.2 [Computer Graphics]: Distributed/network graphics

1. Introduction

Two issues are becoming overwhelmingly important within certain scientific research fields that need to be addressed. The first issue is that data is being produced by scientific instrumentation and *in silica* experiments, which are increasing in size far faster than system capacity can cope. The resulting datasets are overwhelming current top-of the range graphics cards and even high-end visualization workstations. Data volumes that need to be understood are routinely in the order of 100's of gigabytes to terabytes depending on acquisition fidelity and whether time-series data is captured.

The second issue is that of data location, as it is becoming rare for the researcher to be co-located with their large datasets. Computational and scientific instruments are now often scattered across the globe. The problem of data transmission then becomes another bottleneck, with at times the most efficient form of data recovery being the transportation physically of terabytes using usb hard-drives.

This paper attempts to discuss the implementation of an informed solution using non-GPU parallel visualization software, networked computer graphics techniques and optional video conferencing transmission. Over the last decade there has been the introduction of a new philosophy based on e-Science. This has been applied to the data creation, storage, transmission and manipulation process and recently through, what is now termed, as the fourth paradigm [HTT09]. This offers a few guidelines that we believe can and should be applied to the data visualization process.

The following section describes the key GPU/CPU limitations, and lists some prior work as well as possible solutions. Section 2 then describes some of the guidelines proposed from the UK e-Science program, which, although not specific for remote visualization, we believe could be considered beneficial for any large research based software development project that is considering to provide a visualization service element. A new visualization system, in the form of software ported to a Cray XT4 supercomputer, is described in detail in section 3. Finally section 4 looks at initial results and considers future work.

1.1. Project Summary

The visualization of large datasets has become a bottleneck in applications where validation of results, or data acquisition from scientific equipment and in silica experiments, is required at an early stage. Such validation would allow correctness of methods (such as the set up of a physical experiment) to be determined prior to further computational or imaging-machine resources being spent. Datasets are being produced by such experiments that are too large to be visualized on modern GPU technology due to limitations on memory available even in modern workstations.

One solution is to distribute rendering to a cluster of work-

DOI: 10.2312/LocalChapterEvents/TPCG/TPCG10/159-166

stations with a final compositing step used to form a complete image from the partial images produced on the cluster [MCEF94]. In this case the bottleneck is moved to that of transferring the dataset from the acquisition or simulation hardware to the visualization system.

In the case of simulation code running on supercomputers it is possible to move the visualization code to the supercomputer [MRG*08] or to add visualization capabilities to the simulation code [MZM*08]. In this latter case each process taking part in the simulation performs as a final step the rendering of its domain of data using raycasting. The final image is formed by a gather and composite of each partial image. This technique has the advantage of removing any dependency on third-party visualization software and possibly the need to write data to disk (apart from the final image), but is restricted to a particular visualization technique and lacks the ability to perform explorative visualization after the simulation has finished. In both cases rendering is often performed in software on the CPU in the absence of any GPU hardware on such systems. While this results in long render times, often purely in a batch process rather than as an interactive application, it does allow the large core counts and distributed memory of such systems to be utilised in order to render large datasets.

1.2. A Critical Problem and workflow for Materials Science Users

To clarify, specific examples are presented that considers the volume datasets acquired from a new range of X-ray imaging technologies. These include those incorporated within the Diamond Light Source, the third generation synchrotron light source facility run by STFC (currently with a focus on the I12 JEEP, Joint Engineering, Environmental and Processing beamline), and local facilities available at the University of Manchester (Henry Moseley X-ray Imaging Facility sited in the Materials Science Centre). Some of the most advanced facilities are now designed to generate volume datasets for a variety of engineering purposes in the order of 128Gb per time-step ($4K^3 \times 16$ bits). Figure 1 shows images representing the data workflow from a biological sample that is physically scanned on two X-Ray CT (Computer Tomography) machines each creating unique sets of 2D radiographs. These can be reconstructed into a 3D voxel volume that in turn can be segmented and visualized in qualitative and quantitative ways. Current generation of these facilities, in popular use, create volume sizes of 128^3 to 512^3 voxels on a regular basis. The next generation of experimental machines currently being deployed are producing results up to $2K^3$ and are able to produce data scans over multiple time steps. The next generation of machines being constructed, including systems at the Diamond Light Source, are specified up to $4K^3$ with 16 bits per voxel, thus requiring 128GB per scan per time step. Larger scale systems are available, but often have a longer capture time.

1.3. Prior development work: Remote Visualization

There have been numerous attempts to build visualization clusters both local to certain facilities as well as remote to allow researchers to analyse their data. Remote visualization to be useful requires an efficient parallel distribution of the data coupled with a parallel image compositing system, all preferably running at interactive rates. Since 1999 Research Computing Services, at the University of Manchester, has developed various parallel GPU and CPU solutions (http://www.rcs.manchester.ac.uk/). The NGS (National Grid Service, http://www.ngs.ac.uk) also has a visualization service at a national level [NSF08].

Figures 2 and 3 demonstrate the practicality of a small scalable system that can be used as a visualization service. It should be noted that even small systems can encounter data transmission issues. Timing results for the file reader module on a shared-memory multiple processor system reading a 512^3 volume data set across multiple processing elements are shown in figure 4. The yellow line on the graph shows the linear speed-up for isosurface calculation alone, which is as expected. Unfortunately, the red line indicates that the performance for file reading very quickly becomes saturated as there is only a single file service and this then dominates any speed advantage. The combined performance is indicated by the blue line.

To summarise there is a growing need for future visualization services, but also an indication that the data size may be an overriding issue when specifying a new system. Before considering a possible solution and discussing the port of software to enable this, we next consider some guideline rules introduced to aid large scale computational research projects.

2. Jim's Laws possible application to Remote Scientific Visualization

Jim Gray formulated several informal rules or laws that are designed to codify how to approach data engineering challenges related to large-scale scientific datasets. They are specifically designed from the experiences from managing projects within the UK e-Science program and are summarised in a later article by Szaley and Blakely as follows [SB09]:

Data Intensive Scientific computing is becoming increasingly data intensive and moving away from computationally intensive in terms of resource requirements. This follows from cases where we can collate huge quantities of data to even those computational tasks which then can produce huge quantities of data. It is proposed that data management is becoming a more complex problem than computation management.

"scale-out" architecture The solution is in always having a system that can scale preferably in a linear manner. This

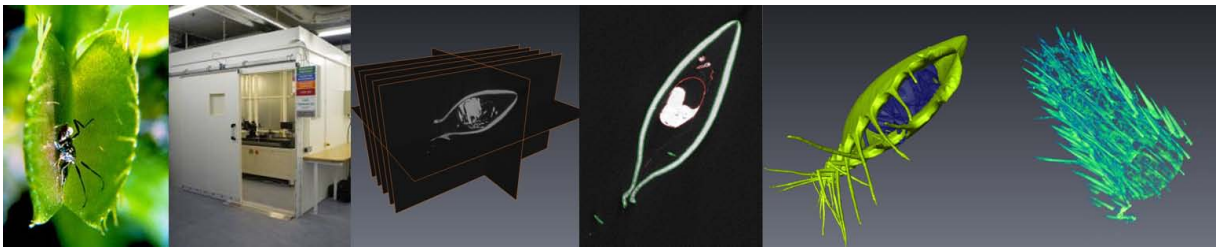

Figure 1: *An illustrative workflow from source to scanner and then data transfer to a remote high-end visualization workstation is shown. In this case the sample scanned is a fly trapped, and being dissolved, within a Venus fly-trap, which has been scanned using both an Xradia MicroXCT and a Xradia nanoXCT machines. The reconstructed volume used custom multi-threaded software and was then subsequently visualized on a specialist large memory footprint workstation.*

automatically removes a future hurdle which would impose limits to the data or computational size that can be exploited. A linear scale may not be possible, and could be considered quite rare, but keeping the computation and order of system complexity polynomial is considered essential for managing any large system that plans on expanding. The rule indicates that all systems should inherently be designed so they can expand.

Computations to the data Bring the computational problem to the data, rather than transferring the data to the computation processor. It is perceived that for large problems the size of the program is always smaller than the data and even if the computational processors are slower at the data location this is still likely to be a faster option than moving the data. It also removes or simplifies an issue of data confidentiality as the data remains at an approved location.

Ask "20 queries" Start the design with asking 20 questions from the researchers. The rule of *twenty* comes from the fact that a small number of questions, will not be enough to extract a broad picture, whereas a large number will dilute the focus; therefore 20 has been taken as a sensible rule-of-thumb.

Working solutions Always go from a working solution to a working solution at every iteration of development. This is especially important when creating applications that need to scale-up.

Considering these points and the up and coming user needs we embarked upon porting a full visualization system where the data will reside, that can scale-out and act in the same memory space. The following section describes some of the implementation details involved as well as the parallelism choices made.

3. Visualization Software

In order to provide large dataset visualization facilities this project has taken the approach of porting an existing visualization code to HECToR, the UK national supercomputer service (http://www.hector.ac.uk/). This is a Cray XT4 system providing 22,656 cores (as 5664 quad-cores) and 43.5TB of distributed memory. No GPU hardware is available and so rendering is performed in software using the MesaGL [Mes10] software implementation of OpenGL. Typically parallel codes are written using MPI [MPI09] to distribute computation across the system. The HECToR system imposes the restriction that each quad-core chip has access to at most 8GB of memory. Hence if four MPI processes are running on one of the quad-core chips each MPI process has access to only 2GB of memory. If a process requires more memory the utilization of the cores can be reduced so that, for example, a single MPI process runs on a quad-core chip allowing that single process access to the entire 8GB of memory. However this results in an increase in the number of nodes required by the MPI job.

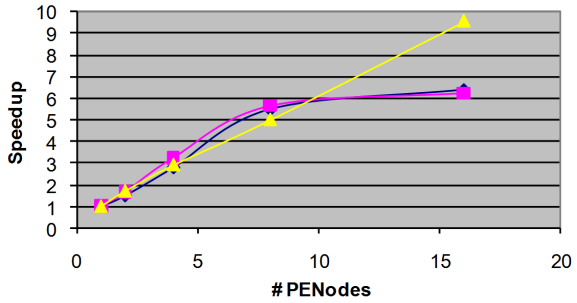

Figure 4: *The yellow line shows the performance for isosurface calculation alone, and indicates a linear speedup as expected. As also expected in this shared memory system using a single file server, the red line that shows the performance for file reading saturates after about eight nodes. This drop in performance dominates very quickly any speed advantage (as shown in the combined blue line).*

The existing visualization code comprises a number of components. The main application is AVS/Express, a com-

Figure 2: *Illustrating the DDR system on a shared memory multiple GPU system employing a non-parallel file reader, and then parallel volume cropping and isosurface modules to form a visualization network using 20 DDR MPI processes. Usually the 20 sub-images are not displayed but, in this case, all images are shown as well as the final composited image that is visible at the top right. The network shown on the left is the visual programming interface for AVS/Express, which shows how relatively simple a parallel rendering application can be constructed by a non-technical user.*

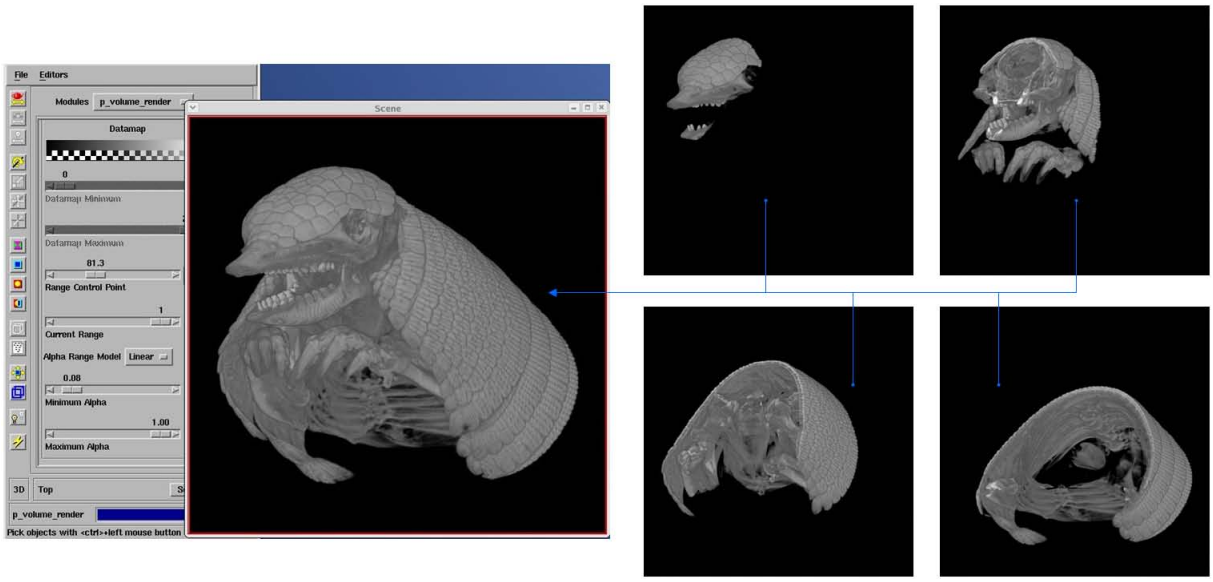

Figure 3: *Multiple screen shot of a five node visualization system that is in use as a local service for the University of Manchester. This is installed on a HP Scalable Visualization Array cluster composed of five x86 64 Xeon processors (currently 10 cores). This renders at interactive rates datasets up to 8-16Gb. The window shown on the left is the only one the user sees, which in this case is currently displaying a composited image from the raycasting volume rendering processes running in parallel on four different cluster nodes (right). The transmission of these four ParaComp framelets is over a dedicated InfiniBand network, that achieves effective real-time performance of about 20 fps.*

mercial visualization application developed by AVS Inc. (http://www.avs.com). This is a visual-programming application where the user connects modules together to form a visualization pipeline [HM90], the final result of executing the pipeline with an input dataset being a rendered image.

The Distributed Data Renderer (DDR) version of this product, developed from the Parallel Edition (http://www.avs.com/software/soft_t/parallel_edition.html) is able to render data on distributed compute nodes where no GPU hardware is available. In essence, data is distributed to a number of compute processes (via MPI) which perform a

mapping of the data to geometry by executing some chosen visualization algorithm (e.g., isosurfacing). Geometry is then passed to a companion MPI process which renders its small sub-volume of the overall dataset. Each rendering process produces a rendered image of its sub-volume. These images are then composited together using a parallel compositing library, forming a final complete image of the dataset for the user to view. This type of rendering is referred to as sort-last rendering [MCEF94] and allows much larger datasets to be rendered than could be handled by a single GPU, or indeed a single CPU. Similar work has been carried out on the HPCx national service [Bet09].

The flexibility of this type of system is that special consideration can be given to certain stages of the visualization pipeline. In particular parallel file reading modules can make use of the parallel filesystem on HECToR. This is essential for the scalability of the proposed system as initial tests have shown that dataset I/O is the limiting factor rather than the actual visualization techniques. Rendering modules may also make use of multiple cores available to an individual MPI process, allowing a mixed-mode implementation using MPI and OpenMP [CMDK00].

AVS/Express allows specific visualization techniques to be composed as an application (or *network* in its terminology). Such applications can be used as batch rendering jobs where the user simply supplies the dataset to be rendered and the number of processes to use for rendering. Alternatively the application can be used interactively allowing the user to manipulate visualization parameters and see the resulting changes to the visualization.

The HECToR system, like many supercomputer systems, distinguishes between login nodes and compute nodes. The key differences from our perspective are that interactive applications that rely on an X11 user interface can only be executed on the login nodes and that MPI jobs can only be executed on the backend compute nodes. In both cases jobs must be submitted to the system via the batch queueing mechanism. These distinctions mean that the AVS/Express user interface (referred to as simply *express* from now on) must be run on the login node but the parallel compute and rendering processes must be run on the compute nodes. Prior to this project express had to be run as part of the MPI job, it being assigned MPI rank 0. Express could then communicate with the compute and rendering process via MPI. As the user interacted with the visualization network, for example changing an isosurface level, these changes would be propagated to the compute processes.

To remove the need for express to be part of the MPI job, an MPI impostor library has been developed. The express application is linked against this version of MPI rather than the hardware vendor's MPI library. The impostor library does not make real MPI function calls and so removes the need for express to be launched by the MPI job launcher (*aprun* on the Cray). Instead it packages together a token representing an MPI function that express has called (e.g., MPI_Send()) and any arguments that the function requires (e.g., the MPI communicator and tag) and passes them to a new rank 0 MPI process running on one of the backend compute nodes (recall rank 0 was previously the main express user interface executable). This communication is performed via a socket opened between express the MPI rank 0 process. The new rank 0 process receives tokens (and function arguments) from express and calls the vendor's corresponding version of that MPI function (e.g., it receives a token for MPI_Send() and so calls the Cray MPI_Send() function with the supplied arguments). Any results from the real MPI function call are sent back to express via the socket. Hence express on the login node is unaware that its MPI function calls are actually being made by a *proxy* on one of the compute nodes. The standard compute and rendering MPI processes running on the compute nodes will communicate with the rank 0 proxy. Those processes are not aware they are communicating with a proxy, they simply assume rank 0 is the express process. The advantage of this method is that AVS modules written for the parallel version do not need to be re-written to remove MPI functions.

The existing AVS/Express DDR product uses the open source *paracomp* compositing library [Par10]. This library has several useful features including the ability to composite non-fullscreen *framelet* images which increases the scalability of the compositor by reducing the number of pixels sent over the network between the processes taking part in compositing. It also supports various networking technologies including Ethernet and InfiniBand. It does not, however, provide an MPI communication layer which makes it unsuitable for use the the Cray system. This is because it requires, in advance, the hostnames of all the nodes taking part in the compositing operation. The batch scheduling system will decide on which physical nodes a job will run and so the hostnames are not known in advance. To overcome this limitation an implementation of 2-3 Swap Composting [YWM08] has been developed. This uses MPI to send rendered images between the compositing processes in a scalable manner.

3.1. Video Dissemination System

There has always been a need for distributing remote visualization [HK07] to a wider audience. The current preference used is the Access Grid video conferencing system as it is an open source multi-video and audio streaming system, which over the last few year is now integrated and used within hundreds of educational establishments around the world. The system is more than a basic video conferencing system allowing sharing and transmission of different data types. This allows for the creation of third party servers including a project called ViCoVRE (Video Conversion for Virtual Research Environment), which allows any computer generated stream of images to be converted on the fly into the

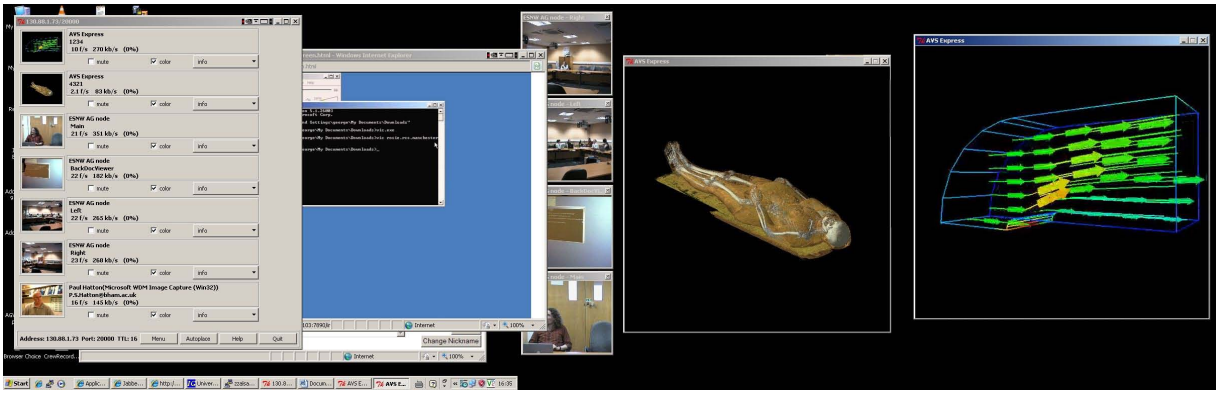

Figure 5: *Video dissemination of remote visualization streams using the ViCoVRE web-server. Two different visualization machines were connected to the ViCoVRE server that automatically converts the image data into two high quality video feeds. These were directed into a Virtual Venue that is then viewable from any Access Grid node in the world.*

Access Grid system. This also allows streams to be recorded and annotated as required.

Figure 5 shows multiple transmission of two remote scientific visualization streams played through a single ViCoVRE web-server and then both output streams being transmitted to the same Access Grid Virtual Venue. The two streams show respectfully, a rotating volume visualization of The Visible Human Project ⓡ, female dataset, and an animated streamline simulation showing air flow over a blunt-fin geometrical shape. These represent two standard visualization techniques that are used to aid in the visual analysis process.

4. Results and Analysis

This visualization application is available to all users of HECToR. It integrates with the Cray XT4's parallel file system that enables the efficient transfer of data from local file store to processors prior to computation.

A large run has involved 256 MPI processes, split evenly between 127 compute processes and 127 rendering processes with as described in the installation 1 proxy process and 1 rank-placement helper process. This setup distributes a large 351Gb dataset to be volume rendered using raycasting, composited and then transmitted to the researcher at between 3–5 fps.

There are some issues that need to be considered:

- The dataset needs to be locally accessible to the Cray XT4. This means if it has not been created on the Cray XT4 directly it has to be first transferred to the system. Even with very fast networks, the speed of light, means transmitting large datasets across the globe is always going to a problem. If the data is not local to a specific supercomputer, the solution may be to port the visualization code to a nearer supercomputer.
- Cray has developed an efficient parallel file system enabling all 127 processors to read sections of the same file. Even this required a few minutes to read the code, synchronise the MPI processes and then read in the 127 slices from the 351 GB raw dataset before interactive visualization could occur.
- Interaction remote visualization was limited to 3–4 fps which is acceptable, but users do require practice. Due to the fact the code is software rendering rather than hardware rendering true interactive rates are unlikely to be realised but this has not been considered an essential requirement. Although real-time rates were not achievable, network analysis indicates the main latency is at the final image transmission stage from the Cray XT4 in Edinburgh, to the researchers in Manchester.
- The Cray XT4 uses a batch system for running jobs and it may not schedule the interactive job at a convenient time for the user. The Cray XT4 was not specifically designed for interactive use, but within the code there is a batch mode that runs a pre-defined visualization network that can be used instead. The advantage of interactivity is that data exploration can occur and decisions made while running the job.

Although there are disadvantages to running this system on HECToR we should remind ourselves of the clear advantages available:

- This may be the only time that the researcher can explore the complete dataset in near real time without downsampling or considering intelligent pre-processing stages. The data does not now need to be moved from the supercomputer it was created from.
- Verification and more precise visual quantitative analysis

can take place on the complete dataset rather than on a subset of the data.
- Complete dataset high-resolution imagery is producible for dissemination purposes. This is limited as the maximum image resolution using MesaGL on the current installation is $4K^2$.

Scientific Visualization is not in itself a large e-Science project, but is often a component within one of these projects. We have shown here most of the software development elements, which have required us to go through similar processes when it comes to data management. With respect to the five informal rules the key ones of data management and maintaining local computation have been considered, but there are still local key data movements that go against the rules. The system works and scales to a level that is required by the current users, but has not yet been fully tested beyond this level.

Currently remote visualization users do not appear to be requiring the next level of peta-scale computing. There are no systems being proposed for the petabyte community, without some form of prior data mining operation for example, and there are no systems being considered for the 100,000+ CPU display architectures as screen projection systems are still limited even in multi-projector systems to less than 100 million pixels, a comparable scale to the human visual system. This may indicate that there is an upper bound on a scientific visualization system, but there is a growing move to add HPC computation within the visualization network and therefore closer integration is inevitable and future demands will grow.

Acknowledgments

For key funding we acknowledge NAG and EPSRC for providing support through their dCSE scheme which assisted in software development on the HECToR, Cray XT4, platform; which also included support and assistance from AVS Inc. The Access Grid video transmission system used the VRE (Virtual Research Environment) ViCoVRE server (enabling on the fly Video Conversion), which included financial support through JISC and was integrated into the academic Access Grid video conferencing system with assistance from the ja.net funded AGSC (Access Grid Support Centre: http://www.ja.net/services/video/agsc/AGSCHome/). For dissemination and guidance we need to thank vizNET a national UK network designed specifically to promote visualization across the UK (http://www.viznet.ac.uk/)

Key to the project and for future evaluation are our test user group, who were mostly based or connected with the Henry Moseley X-Ray Imaging Facility, within the Materials Science Centre, at the University of Manchester (http://www.materials.manchester.ac.uk/research/facilities/-moseley/), and especially the pioneer users; Prof Phil Withers, and Drs Paul Mummery, Phil Manning and Bill Sellers.

References

[Bet09] BETHUNE I.: *Parallel Visualization on HPCx*. Tech. rep., STFC, 2009. http://www.hpcx.ac.uk/research/hpc/technical_reports/HPCxTR0803.pdf. 5

[CMDK00] CHANDRA R., MENON R., DAGUM L., KOHR D.: *Parallel Programming with OpenMP*. Morgan Kaufmann, 2000. 5

[HK07] HOLMES I. R., KALAWSKY R. S.: Delivering effective and usable interactive 3d visualization on lightweight mobile devices. In *TP.CG'07* (Bangor, Wales, UK, June 2007), Shriver B., Neilson G., Rosenblum L., (Eds.), Eurographics Association. 5

[HM90] HABER R., MCNABB D.: Visualization idioms: A conceptual model for scientific visualization systems. In *Visualization In Scientific Computing* (1990), Shriver B., Neilson G., Rosenblum L., (Eds.), IEEE Computer Science Press, pp. 74–93. 4

[HTT09] HEY T., TANSLEY S., TOLLE K. (Eds.): *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, Redmond, Washington, 2009. 1, 8

[MCEF94] MOLNAR S., COX M., ELLSWORTH D., FUCHS H.: A sorting classification of parallel rendering. *IEEE Computer Graphics and Algorithms* (July 1994), 23–32. 2, 5

[Mes10] MESAGL: Open source software OpenGL implementation. http://www.mesa3d.org, 2010. 3

[MPI09] Message Passing Interface Forum, MPI: A message-passing interface standard, version 2.2. High Performance Computing Center Stuttgart (HLRS), 2009. 3

[MRG*08] MORELAND K., ROGERS D., GREENFIELD J., GEVECI B., MARION P., NEUNDORF A., ESCHENBERG K.: Large scale visualization on the Cray XT3 using ParaView. In *Cray User Group* (May 2008), Shriver B., Neilson G., Rosenblum L., (Eds.). http://www.cs.unm.edu/~kmorel/documents/PVCrayXT3/PVCrayXT3.pdf. 2

[MZM*08] MANOS S., ZASADA S., MAZZEO M. D., HAINES R., PINNING R., DOCTORS G., BROOKE J., VOEVODSKI K., BREW S., COVENEY P. V.: GENIUS – Grid Enabled Neurosurgical Imaging Using Simulation. In *SC08* (Austin, TX, USA, May 2008), SuperComputing. Analytics Challenge, Professional Entry. 2

[NSF08] NAGELLA S., SASTRY L., FOWLER R.: *Remote Rendering on Visualization Cluster using VirtualGL and Chromium*. Tech. rep., Science and Technologies Facilities Council, vizNET-WP4-16-STFC-UsingVizCluster, October 2008. http://www.viznet.ac.uk/files/VIZNET-WP4-16-STFC-UsingVizClusters-web.pdf, http://sct.esc.rl.ac.uk/viscluster/index.html. 2

[Par10] PARACOMP: Open source HP compositor library. http://paracomp.sourceforge.net, 2010. 5

[SB09] SZALAY A. S., BLAKELEY J. A.: Gray's laws: database-centric computing in science. In Hey et al. [HTT09], pp. 5–12. 2

[YWM08] YU H., WANG C., MA K.-L.: Massively parallel volume rendering using 2-3 swap image compositing. In *SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing* (Piscataway, NJ, USA, 2008), IEEE Press, pp. 1–11. 5