Visualisation on the Grid: A Web Service Approach

Stuart M. Charters, Nicolas S. Holliman and Malcolm Munro
Visualisation Research Group,
e-Science Research Institute,
University of Durham,
South Road,
Durham,
DH1 3LE, UK
S.M.Charters@durham.ac.uk N.S.Holliman@durham.ac.uk Malcolm.Munro@durham.ac.uk

Abstract

The visualisation strand of the e-Demand project is working to distribute the visualisation pipeline across the Grid, allowing visualisations to be composed as needed. The traditional visualisation pipeline concept is maintained by providing an ability to divide a visualisation into its component parts and then built upon by allowing each of these parts to be deployed on an appropriate grid resource. The implementation has undergone several iterations, using releases of the Globus toolkit and using standard web service technology. A client that allows visualisation services to be composed has been developed. Case studies from a range of domains across the sciences demonstrate reuse of services and multiple path visualisation pipelines. An initial analysis of performance using the scientific case studies as a basis for experimentation is discussed. The issues encountered during the differing implementations of the architecture and those still outstanding are described. Future work is outlined including support for interactivity, collaboration and steering.

1 Introduction

The visualisation strand of the e-Demand project is working to design an architecture for distributed stereoscopic visualisation and to provide a prototype implementation based on the web service concept that works using non-proprietary mechanisms and protocols. The stereoscopic nature of the visualisations mean that special services have to be implemented as part of the development. The design of the architecture has been influenced by several scientific case studies.

Experiences gained during the development of the architecture initially using grid services and then using web services are outlined, highlighting the areas that caused difficulty in moving between implementations.

2 Architecture

The objectives of developing an architecture to allow visualisation on the grid are to provide desktop visualisation using distributed resources that can be combined as needed to solve different visualisation problems. The architecture operates using standard protocols and software and operate across institutional boundaries without special provisions being made. The architecture should also be extensible and allow collaboration to be supported.

The design of the architecture maintains a similar conceptual view to the traditional Haber and McNabb [7] visualisation pipeline as shown in Figure 1 but encapsulates each stage as a separate service. These services can be run on different machines in differing geographical locations, allowing the best resource for the task to be used and for this to aid in providing the best performance at each stage. In addition to separating each stage into a service the architecture provides a provision for a reverse flow of information allowing control of the pipeline by each stage and the user this is implemented through the use of the steering service.

Figure 1. Haber-McNabb Visualization Pipeline
The use of services in the architecture opens the way to a greater choice in building pipelines for visualisation. The encapsulation of each stage of the pipeline allows functionality from different vendors to be brought together and combined in a single pipeline. This choice element of services also allows individual components in a pipeline to swapped for alternative ones; one that uses a different algorithm, one that is faster or cheaper or simply more reliable.

Each stage of the pipeline has a specific role to perform and these are outlined below.

**Data Services** in the pipeline are responsible for delivering the raw data to the visualisation. These services can be an interface to a simple data file or complex services bringing together data from multiple sources and in a variety of formats.

**Filter Services** take data and converts the format of that data, if required, and reduces the size of a data stream by filtering off the content that is not required for the visualisation. Mathematical calculations and transformations of the data can also be done at this stage, for example interpolation.

**Mapping Services** are responsible for mapping data onto a visual representation. A metaphor, either abstract or real world, is used at this stage and the data is converted into geometry which can then be rendered. An example Mapping Service is an Isosurfacing Service, which calculates the boundary between two regions and generates a surface at that boundary.

**Render Services** take the geometry of a Mapping Service and renders it as an image which can be passed to the Presentation Service.

**Presentation Services** receives from a Render Service an image which can be displayed to the user who may then interact with the image presented.

The Render and Presentation stages of the pipeline may be implemented as a single service for devices where rendering can take place locally. The separation is maintained in the architecture for the purposes of flexibility and to allow remote rendering systems to be implemented.

Whilst services have a nominal stage in the pipeline, no requirement on the ordering of services is enforced and as such a pipeline may not make use of all types of service.

There are two issues concerning co-ordination, that of data co-ordination and of pipeline composition. Data co-ordination between services is achieved through the mechanism of grid level streaming. Pipeline co-ordination is achieved through the external pipeline composition tool.

### 3 Case Studies

Case studies to drive the architecture design have been drawn from a range of disciplines across the field of science, including X-Ray Crystallography, Cosmology and Astronomy. These case studies demonstrate the operation of services composed together and the use by multiple pipelines of some services.

Three case studies are outlined below each of which highlights particular features that would be desirable in the architecture. In each case study the problem, or a sub set of the problem has been visualised using existing visualisation tools such as IRIS Explorer and ParaView based on VTK [9] to demonstrate the type of output that is required and to determine the pipeline construction for each study.

#### 3.1 X-Ray Crystallography

The X-Ray Crystallography case study is taken from the chemistry domain and uses experimental data obtained by cooling crystals to low temperatures and exposing them to X-Rays. These X-Rays are then detected and the data processed to produce an output that describes the molecular composition of the crystal and the electron field density around each atom in the molecule. The case study demonstrates a branching pipeline showin in Figure 3, where two forms of data generated by an experiment are used to produce different parts of a visualisation, these parts are then combined to form the final representation.

![Figure 3. X-Ray Crystallography Pipeline](image)
The output produced by the pipeline is similar to the stereo pair shown in Figure 4.

![Figure 4. Stereo Pair Showing Field Density](image)

This case study highlights several important considerations for the architecture; the need to analyse different types of data, either from the same data source or different ones; and to merge these results to form the final visualisation. This means that the architecture must be able to support pipelines with multiple branches where these branches are merged back together. The issue of synchronization in the pipeline is also raised by the need to support these branching pipelines co-ordinating the results so that the visualisation is correct and valid. The output in Figure 4 shows a stereo pair produced for this visualisation. This highlights another area that the architecture should support, that of multiple display device types. Auto-stereoscopic displays have become increasingly available for the desktop. The use of these displays for visualisation can aid the scientist greatly in understanding their data and therefore the architecture should be able to support these different types of display device.

### 3.2 Cosmology Simulation

The cosmology case study is of a 10 billion particle simulation of the predicted distribution of dark matter in the universe and how it evolves over time [6]. The simulation outputs 512 files that are approximately 600MB in size each for every timestep. This results in approximately 300Gb per time step. The complete simulation has some 15TB of data in total.

The files are constructed using a space-filling curve which allows regions of space to be extracted easily from each file. Each segment of the file is an unstructured grid and contains a velocity vector for each particle. This case study highlights the need for scalable systems, the data set is massively parallel, well structured and contains a very large amount of data.

The pipeline as shown in Figure 5 demonstrates how the outputs shown in Figure 6 can be achieved.

The issues for consideration in the pipeline design raised by this case study include dealing with very large data sets, and how this impacts on service design. Services must be able to scale to cope with very large data sets. This case study also raises the issue of level of detail management. The size of the data set means that it is unlikely that the whole data set would be viewed on the individual particle level, more that a region of interest would be highlighted and that region would then be examined in more detail. This raises issues on how the region of interest is communicated to services in the pipeline.

### 3.3 Astronomy

This case study demonstrates reuse in the visualisation pipeline and the use of different rendering mechanisms. The data for this case study comes from observing a region of space for 10 hours with the SAURON integral field spectrograph on the William Herschel Telescope in La Palma [1]. The data shows three galaxies, two of which are star forming galaxies and the third a heavily dust-enshrouded galaxy. The goal of the observation and the visualisation is to understand the formation of galaxies in the very early universe.

Two different visualisations of this data have been pro-
duced, the first a volume rendering of the data shown in Figure 8 and produced by the pipeline shown in Figure 7. The second is a polygon isosurface demonstrating the ability to reuse the isosurface module seen in the X-Ray Crystallography case study. The pipeline is shown in Figure 9 and the output can be seen in Figure 10.

**Figure 7. Observing the Universe - Volume Rendered**

This case study exposes the reuse available in a visualisation pipeline. The isosurface module in the study could be the same module as used in the X-Ray crystallography case study. Equally however an isosurface module from a different vendor could be used. This demonstrates the choice available to users of a service based visualisation architecture.

The two different types of output, volume rendering and polygon rendering highlight the need for multiple types of some classes of services to cater for different users.

4 Implementation

The implementation of the architecture has been through several iterations, from the Globus Toolkit 3 Alpha Release 2 reported previously [4] to Globus Toolkit 3.2 and then to a Web Service based implementation following WS-GAF [8] and WS-Resource [5].

Each re-factoring of the implementation brought with it different challenges. The initial implementation using the Alpha 2 release of the Globus Toolkit suffered from building upon code that was not at production standard and not all functionality was implemented. The Globus Toolkit changed dramatically between Alpha Release 2 and the 3.2 production release, the packaging of APIs had changed substantially. These changes in the packaging of the Globus
APIs meant that each service in the pipeline had to be re-implemented. For this re-implementation the functional part of each service could be transferred to the new service but the way it interacts with the service had to be modified.

The migration to web services from the Globus Toolkit was a move to a more mature development platform with the existence of more tools and a wider community knowledge base to draw upon for solving implementation issues. Again the functional part of each service could be brought across from previous implementations but the interaction mechanisms had to change more significantly due to conceptual differences between grid services and web services.

The opportunity to re-implement the architecture was useful in many respects as it allowed design decisions to be revisited and for the benefit of previous experience to influence the approach taken to each implementation.

During the implementation many workarounds had to be found to solve problems, one such workaround was the use of local files, in the conceptual design a service would have an open communication channel to the next service, it would write data at the producing service end and be consumed at the client service end, similar to a socket connection. At present no implementation to support this type of protocol is in existence, it is not possible to use a socket connection as each service may have to communicate through firewalls which would, in many instances, block such connections. The solution was for the service to write a file locally and return to the next service a URL at which this could be accessed. The Java COG Kit [10] HTTP Streaming implementation was then used in client services to read the file across the network. The HTTP protocol is not the most efficient for file transfer but the approach provided a solution to the problem. An issue was discovered however. As the URL of the output file is returned before the output is completed, the client service may not read all of the data contained within the file if the rate of data output is slower than the rate the data is being read at. A blocking I/O mechanism would help to solve this problem but no work around has yet been found.

The implementation of the architecture involved the use of some external libraries and applications. One of these external libraries was the Visualisation Toolkit (VTK) from Kitware [9], the use of this external library highlighted some of the problems with using existing systems. The first issue was the need for VTK to load data from a local file, this was overcome by copying the remote output to a local file. This is not an ideal solution however as it requires the remote process to have completed before the service using VTK can begin. The second issue was binding to VTK in a web service The VTK toolkit is written in C++ with Java bindings. The use of this type of system proved problematic in a web service container and caused the web service to fail. As an interim solution the web service acted as a front end to a standalone daemon running the VTK service that could be called by an internal socket connection.

![Figure 11. Composition Client](image)

A client as shown in Figure 11 to allow the composition of visualisation pipelines is under development. Each of the services is registered with a UDDI server. This allows the client to discover them and for the user to compose them together into a pipeline. The composition client is based on traditional drag and drop concepts where registered services can be easily identified and composed together. The composition client allows the user to select which resources they require for their visualisation.

5 Preliminary Evaluation

Preliminary evaluation has been carried out at an informal level as the implementation of the architecture is at the prototype stage. A full evaluation to consider performance through the use of network communications and the improvements gained due to the use of different resources for different tasks will be carried out when the implementation is completed. The development of the composition tool has not been made significantly easier or harder by the transition from grid services to web services. The case studies are being used to formally evaluate the usability of the architecture.

The scalability of the architecture is an important area for grid systems The vision being that they can scale to cope with arbitrarily large problems automatically using extra resources as needed to complete the task. The scalability concerns for the visualisation architecture centre around those of data set size and data set complexity.

The issues around data set size relate to whether a data set can be loaded by services that need to process it. This issue often relates to how the original data was generated and
the format it is stored in. If a file can be designed so that it can be broken into smaller chunks with each of these chunks processed in parallel then the architecture should have little trouble scaling to accommodate even the largest problem. Issues may arise at later stages of the visualisation pipeline when the output of processing stages have to be composited back together but it is felt that good design of visualisation pipelines with data reduction techniques being used to generate high level views that problems can be avoided.

A larger issue with scalability is that of problem complexity where a relatively small data set increases dramatically due to the operations being performed on it. This could be due to the nature of the problem or due to the use of inappropriate visualisation techniques. These issues of problem complexity depend heavily on the data being analysed and therefore each individual problem should be examined to discover the best approach to solving the problem, from precomputing extra data to reducing data size or choosing a different visualisation technique.

The implementation of individual services can also impact on the scalability of the architecture and each service needs to be designed with scalability in mind. This service design can include creating multi-threaded services, outputting data in a parallel manner and being self contained so that it can be migrated to a new resource if required.

Performance of the prototype architecture is dependent on the individual services used in each pipeline and their implementation. However the use of web services adds an overhead due to the time to initialise the service and the communication delay incurred when invoking the service. This typically adds a time in the order of milliseconds to each call. As each service moves data across the network this adds a performance overhead to each stage of the process which is dependent on the quantity of the data to be moved, the structure of the data and the end to end network performance. Intelligent data management techniques for moving data between sources could greatly reduce the amount of data transmission and retransmission, it can also manage the data transfers in such a manner that data is delivered as needed, ‘just in time data delivery’, which would reduce the perceived impact of transferring data. The main impact of data delivery across the network is when the pipeline is constructed and initialised as there will be a delay before a visual output is achieved. However this delay is also noticed with other visualisation applications where large amounts of data are being loaded. Resource management however is not a goal of this project and other grid middleware is expected to be used to solve these resource management issues.

The design of the architecture will allow services to be used in multiple pipelines. This can be seen in the X-Ray crystallography and cosmological observation case studies where the isosurface surface can be reused. In all of the pipelines the same rendering and presentation services can be used. This reuse can reduce the time it takes to develop a bespoke visualisation and the time that it takes to try different visualisation techniques on the same data.

6 Implementation Issues

The development of a prototype architecture brought several issues for consideration these are discussed below.

As happens with all new technology the standards take time to evolve and settle and this is still the case with web services. As people begin to use them more extensively deficiencies in the existing standards are highlighted and this often leads to several development efforts to design a new mechanism for achieving the task. Until these mechanisms have been agreed by the standards body. There are often multiple methods of achieving a goal. These methods often do not inter operate. In the web services arena the initial standards are well established and have entered a state where incremental changes and developments can be made without fundamental shifts and without rendering previous implementations obsolete.

Other areas of web service technology such as security and context management have not yet reached that state and different vendors have provided different solutions to these problems. When building an architecture that is to be deployed in a heterogeneous environment on resources that the user may have limited control over, decisions about which features to utilise can have a large impact. The possible solutions include, providing your own implementation that does not rely on anything else or to wait for a standard to be reached. The first solution whilst solving the problem does require considerable development effort and would also require everyone who wished to interoperate with your services to make use of your implementation. This could severely restrict the take up of the architecture. The second solution whilst appearing to ignore the issue in many cases is a viable option as it eases the development burden and allows effort to be concentrated on core functionality and demonstrating concepts.

Whilst the visualisation pipeline as shown in Figure 1 has been used by many prior visualisation systems these systems were usually closed source and self contained and for that reason they each chose their own mechanism for data exchange between modules that suited the problems they were solving and the hardware architectures that the system was designed for. In an architecture where the underlying hardware is not known and there is a requirement for interoperability with other services designed by third parties data interchange formats and standards become of paramount importance. Few such standards exist as yet and an interim solution of providing many format conversion filters is envisaged.
One issue that impacted across all developments is the transfer of data between services. To efficiently visualise very large data sets it is felt that all services in the pipeline should be able to operate on data when it is ready rather than operate in a batch mode. Standard methods of communication for transfer of large data sets via a streaming mechanism that can be easily integrated with a service do not exist yet and so work arounds (currently HTTP Streaming File Transfer) have been used this is only partially acceptable because of protocol efficiency and the lack of blocking I/O support.

Stability of the architecture is an important issue. For an architecture to gain acceptance and to be used it must be usable by visualisation experts and scientists on a regular basis with confidence. In an architecture that is built on web services the stability of many different components affects the perceived scalability of the architecture, from the failure of an individual service, the failure of the web service container and even the failure of remote hardware. It is important therefore that the architecture can survive these error conditions and at the very least exit gracefully with an appropriate error message for the user. Whilst the architecture has been at the development stage this area of stability has not received much attention but as later revisions are made it will become more important. The other area of stability that impacts the architecture is the rate at which the underlying technology progresses and also the rate at which the architecture changes. If the underlying web service technology is changing rapidly and significantly and in such a way that services need to be re developed to work then the architecture is not going to be effective as too much time will be devoted to simply keeping up with the latest release. Similarly if the functionality of the architecture and the services provided are changing in such a way that pipelines composed with earlier versions no longer perform as expected then users will not be able to trust the visualisations they are developing.

The limitations of web services have themselves an impact on the functionality of the architecture. As web services run within a container environment this places limitations such as memory usage, security and communication time outs upon the services. These limitations mean that in many cases the best approach is to use a web service as a front end to a system that is deployed. This approach is also useful where external libraries are going to be used to provide functionality. The use of web services as a front end only also allows the internal configuration to be tuned to the environment in which it is deployed, avoiding many of the overheads and performance issues that the use of pervasive protocols such as HTTP bring.

For services to be used in the architecture they must be available for use, to be available they must be easily discovered by users. Discovery of services relies on those services being well described and registered in a directory. To assist in describing services an ontology of visualisation would allow the same terms to be used to describe the same services developed by different people. It is then hoped that directories of visualisation services would grow organically by people making their services available publicly. Work on a visualisation ontology [2] is already underway and it is hoped that this will help to drive the description of services.

There is a significant conceptual change between grid services and web services, that of state and state ownership. Grid services allows individual instances of a service to be created and for users to interact with those services. The web service approach means that one instance of a service is deployed and it is used anonymously. Progress is being made to develop standards to allow state information to be maintained within a web service such as WS-Context[3].

## 7 Conclusion and Future Work

The current implementation allows a forward information flow within the visualisation pipeline. Later implementations will allow the use of context information to assist with steering of services and allow re-execution of parts of the pipeline, this will allow support for interactive visualisation to be added. The use of visualisation in a collaborative manner remains an important goal for the system and can be investigated more fully in due course.

Knowledge of problem to be visualised is an area than impacts greatly on the visualisation process and it is for those problems that are well understood that visualisation pipelines that are most effective can be designed and built. Whilst visualisations of most data sets can be performed the resulting visualisations are not necessarily tailored to the objectives of the scientist. Time invested in understanding the problem and the data before building visualisation pipelines can have a huge impact on the outcomes of the visualisation process.

The architecture and the prototype implementation provide a method of achieving the project aim of distributing the visualisation pipeline across the Grid. Issues such as streaming data transfer and bidirectional communication still remain but the stable platform of web services provides a solid basis for overcoming these issues. The case studies have shown that the visualisation pipeline can be used in a range of different scientific domains and the evaluation has highlighted the areas of strength in the approach. The composition client provides an interface to allow the creation of visualisation pipelines easily by non-expert users.

It is believed that the distributed architecture for visualisation across the grid is widely applicable to the problem of visualisation but that current prototypes highlight the need for further work before such an approach is practical for use by visualisation and scientists alike. A number of areas for
future development remain but a step forward towards flexible, accessible, usable and powerful visualisation services on the grid has been made.

8 Acknowledgments

Thanks to Professor J. Howard, Professor C. Frenk, Dr. A. Jenkins and Dr. S. Morris for providing data for the case studies.

This work is part of the EPSRC/DTI funded e-Demand: A Demand-Led Service-Based Architecture for Dependable e-Science Applications Project (Grant No.: THBB/C008/00112C) which is supported by Industry partners Sharp, Sun and Sparkle Technology.

References