SOCA (2008) 2:187–201
DOI 10.1007/s11761-008-0031-6

ORIGINAL RESEARCH PAPER

Service-oriented visualization applied to medical data analysis

Elena Zudilova-Seinstra · Ning Yang · Lilit Axner · Adianto Wibisono · Dmitry Vasunin

Received: 17 August 2007 / Revised: 13 June 2008 / Accepted: 29 August 2008 / Published online: 23 September 2008
© The Author(s) 2008. This article is published with open access at Springerlink.com

Abstract With the era of Grid computing, data driven experiments and simulations have become very advanced and complicated. To allow specialists from various domains to deal with large datasets, aside from developing efficient extraction techniques, it is necessary to have available computational facilities to visualize and interact with the results of an extraction process. Having this in mind, we developed an Interactive Visualization Framework, which supports a service-oriented architecture. This framework allows, on one hand visualization experts to construct visualizations to view and interact with large datasets, and on the other hand end-users (e.g., medical specialists) to explore these visualizations irrespective of their geographical location and available computing resources. The image-based analysis of vascular disorders served as a case study for this project. The paper presents main research findings and reports on the current implementation status.

Keywords Service-oriented visualization · Grid/Web services · Image-based medical analysis · Workflow management

E. Zudilova-Seinstra (✉) · N. Yang · L. Axner
Section Computational Science, Informatics Institute,
University of Amsterdam, Amsterdam, The Netherlands
e-mail: E.V.Zudilova-Seinstra@uva.nl

N. Yang
e-mail: N.Yang@uva.nl

L. Axner
e-mail: L.Abrahamyan@uva.nl

A. Wibisono · D. Vasunin
Systems and Networking Engineering Group, Informatics Institute,
University of Amsterdam, Amsterdam, The Netherlands
e-mail: A.Wibisono@uva.nl

D. Vasunin
e-mail: D.Vasunin@uva.nl

1 Introduction

With the fast development of information technologies the amount of data at one's disposal is enormous. Data is produced at different scales, arrives from various sources (sensors, instruments, simulations, databases, direct manipulations, etc.) and has different structures. The data explosion has led to very large detailed datasets and the amount of details in these datasets continues growing at explosive rates. Sheer sizes, large-scale multi-regional and multi-institution collaborations have resulted in the distribution of the data around the world. Both our work and everyday life become more and more dependent on distributed computing.

Nowadays the data processing requirements are highly variable, both in the type of resources required and the processing demands made upon these systems. For medical applications remote access to the patient data, medical knowledge bases and special medical instruments are of utmost importance [19]. Processing, visualization and integration of large medical datasets are the major cornerstones of the modern Healthcare.

The Grid technology was introduced recently to enlarge concepts of distributed processing, visualization and integration of information from various sources. The advent of Grid computing paradigm and the linkage to Web services provide fresh challenges and opportunities for data delivery, information management and the close coupling of simulation and visualization [18]. Connectivity between distant locations, interoperability between different kinds of systems and resources and high levels of computational performance are some of the most promising characteristics of the Grid.

Interactivity is among important requisites of Grid computing [10,15,17]. However, the question of how to provide end-users with intuitive means to view and interact with large datasets and related computations remains open. Even though the importance of user-centred design has been already stressed in literature [7,11], many on-going e-Science projects still do not consider that their tools and applications will also be used by non-computer experts. Medical specialists, for instance, are not willing or able to spend substantial time on programming and learning how to use a new visualization system [28].

To address this research concern, we designed and implemented an Interactive Visualisation Framework (IVF) suitable for the work of both visualization experts and end-users. IVF is aimed to assist in building distributed visualizations and their execution on the Grid. It allows experts to develop visualization routines for end-users while end-users are able to intuitively explore their data by running these routines irrespective of the geographical location and available computing resources.

IVF supports a service-oriented visualization model. This model is based on the traditional visualization pipeline of Harber and McNabb [9] and suggests that each intermediate visualization sub-process is potentially transformable into an independent Grid/Web service. Thanks to the granularity of a service-oriented architecture users can compose and steer distributed visualization pipelines running on the Grid. Furthermore, intermediate visualization stages can be assigned to heterogeneous computing resources considering their complexity.

To demonstrate that a proposed architecture is suitable for medical visualization, we applied the IVF framework to the image-based analysis of vascular disorders. We focused on the visualization of both experimental and simulated medical data. The iso-surface extraction technique was applied to visualize the patient's vascular condition. Streamline-based visualization was chosen to represent results of the blood flow simulation. The visualization part has been performed using the Kitware Visualization Toolkit (VTK) [35]. A collection of VTK-based Grid/Web services has been developed in this project as part of the IVF framework. To allow users to interactively compose distributed visualization pipelines, IVF has been integrated with the VLAM-G environment, which is a Grid-based portal for the remote experiment control and scientific workflow management [25].

The paper presents main findings of this research project and is organised as follows. We provide an overview of related work and discuss the motivation of the project in Sect. 2. We then introduce a service-oriented approach to data visualization and discuss a possible medical data analysis scenario in Sect. 3. We describe the architectural design of the IVF framework in Sect. 4 and present a visualization case study in Sect. 5. In Sect. 6, the process of building visualization services from the VTK pipelines is explained in detail. We report on the current implementation status of the framework in Sect. 7. Results of performance measurements can be found in Sect. 8. We finalize with conclusions and discussion in Sect. 9.

2 Background

Vascular diseases are the leading cause of death and disability all over the world, especially in developed countries. According to the Canberra Fact Report,[1] "For a 40 year old, the risk of having a vascular disease at some point in time in their future life is 1 in 2 for men and 1 in 3 for women". The total burden of vascular diseases is expected to increase over the coming decades due to the growing number of elderly people, among whom these diseases are most common.

In general, vascular diseases fall into two main categories [2]: aneurisms and stenosis. An aneurysmal disease is a balloon like swelling of the artery. Stenosis is a narrowing of the artery. To redirect the blood flow or to repair the weakened artery, a vascular reconstruction procedure can be applied. A criterion for the success of a vascular procedure is the normalization of the blood flow in the affected area. The procedure may include balloon angioplasty, adding shunts or bypasses in the case of aneurysms or applying thrombolysis techniques, bypasses or stent placement for a stenosis.

The best treatment is not always obvious because of the complexity of the human vascular system and because of other diseases that a patient may have. Modelling of the blood flow in human vascular structures helps to obtain an extensive knowledge about its behaviour and to develop solutions for the treatment of vascular disorders [28]. Datasets generated via numerical modelling/simulation can be very difficult to interpret. This challenge can be tackled by extracting relevant information available in numerical datasets and creating efficient visualizations of such data.

Visualization is an important component of today's medical systems for training, diagnosis and treatment planning [19]. In the past ten years, it has evolved from a tool used by a few people into an indispensable approach towards improving understanding of extremely large medical datasets (both experimental and simulated) currently being produced.

Although a powerful graphics workstation equipped with a single display system can be sufficient for medical visualization in many cases, restriction to local hardware and software resources makes it difficult for researchers to easily share their visualizations. Also, sizes of numerical datasets and the amount of details in these datasets may vary. Processing and visualization of large and/or complicated data often require powerful computational resources, which are not ultimately available at the user site, especially if the users

[1] Fact report: Heart, stroke and vascular diseases: AIHW Cat. No. CVD27. Canberra: AIHW and National Heart Foundation of Australia, Cardiovascular Disease Series No. 22, 2004.

are clinicians. So aside from developing efficient extraction techniques, it is important to have facilities available (with adequate permissions) to visualize results of the extraction process irrespective of local resources.

Grid computing provides excellent opportunities in this respect [7,8] and hence there have already been several attempts made to build Grid enabled visualization frameworks. For instance, the ongoing RealityGrid [18] project uses visualization as part of a bespoke application to visualize the output of simulations running on the Grid. In the gViz project [26], an extension to the NAG IRIS Explorer [33] was developed, which allows individual visualization modules to be prepared and executed using remote Grid resources. The Grid Visualization Kernel (GVK) developed as part of the CrossGrid project is a middleware extension built on top of the Globus Toolkit. GVK allows remote interactive visualization of both experimental data and acquired simulation results [14,20].

Unfortunately, existing frameworks have drawbacks caused by the visualization models they support. None of traditional visualization models fully comply with needs and requirements of distributed computing.

Models such as visualization cycle model [23], visualization pipeline model [9,24], data-pull models [21] and reference visualization models [5,13] are based on the major linkage of visualization sub-processes. In the Grid environment, this linkage often results in under utilisation of certain machines for time longer than actually visualization runs [4,11].

Moreover, to allow interactive user steering over the intermediate visualization stages, external monitoring tools need to be developed. When frequently evoked, these tools can significantly slowdown the visualization process [15,26].

Another concern is related to the necessity of having special visualization hardware on the user site. Currently visualization front-ends are simply excluded from modelling. As a result, users having visualization equipment that differs from the specified configuration are simply unable to explore all visualization and interaction features provided by Grid enabled frameworks [29].

3 Applying a service-oriented approach to data visualization

To provide better user experience and to allow motivated allocation of Grid resources, we developed a service-oriented visualization model. This model originates from the traditional visualization pipeline model introduced by Harber and McNabb [9] and is based on the principle that any visualization sub-process can be transformed into an independent Grid/Web service [3] (see Fig. 1). In a proposed architecture, each visualization service performs a specific role and can be executed on a different logical machine at a different geographical location.

For instance, the reading service takes care of the data delivery. It delivers original raw data to the visualization site or to a specific data repository. The reading service can be implemented as an interface to a database or to a simple file or it can be also deployed as a complex service bringing together data from multiple sources. When processing of multiple data sources is required, conversion can be also included into a reading service.

The filtering service takes a data stream from the reading service and reduces its size by keeping the content required for this particular visualization only. This often implies that certain mathematical calculations have to be performed (e.g., interpolation).

The mapping service is responsible for mapping data to visual representations. At this visualization stage the data is converted into geometry to be rendered. An example is the iso-surface extraction, which calculates a boundary between two regions and generates a surface at that boundary.

The rendering service takes a geometry generated by the mapping service and renders this geometry as the final image so that the user can view and interact with it.

Thanks to the granularity of a proposed architecture, services can be assigned to different computing resources based on their complexity. For instance, the rendering service can be provided with a more significant computational power than the mapping service responsible only for the conversion of the filtered data into geometrical primitives.

In general, a service-oriented visualization model can be represented as it is shown in Fig. 2, where each transformation sub-process is represented as an independent service. However, it would be also possible to combine several visualization stages. For instance, to reduce data transfer between components of a distributed visualization pipeline it might be useful to combine mapping and rendering or reading and filtering stages within one Grid/Web service.

Interactive steering plays an important role in service-oriented visualization. To allow the interactive configuration of and control over a complete pipeline, we replaced the traditional forward data flow between services with a bi-directional one. Due to this bi-directionality, visualization parameters and the data about viewing conditions can be passed from one service to another also in a bottom-up manner. The last is vital for the generation of stereo images for autostereoscopic systems [4,16]. To allow the flexible configuration of the visualization front-end, available display systems can be viewed as data resources.

Figure 3 shows visualization services in a scenario and the image-based medical data analysis path using these services, where the numbered lines are steps of the analysis path. The path begins when the user (clinician) passes medical data (experimental and/or simulated) to the client application (1),

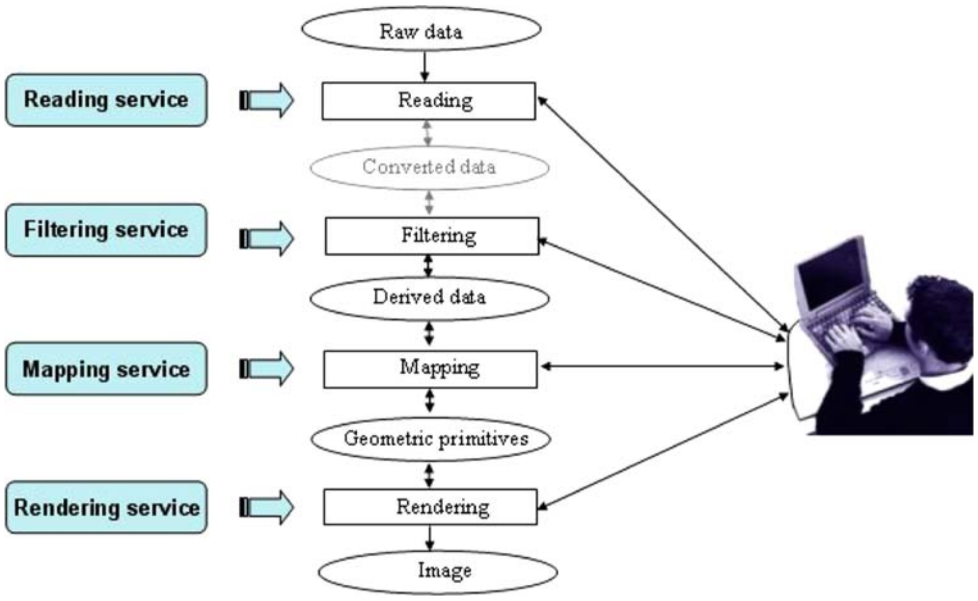

Fig. 1 From a visualization pipeline to a service-oriented model [29]

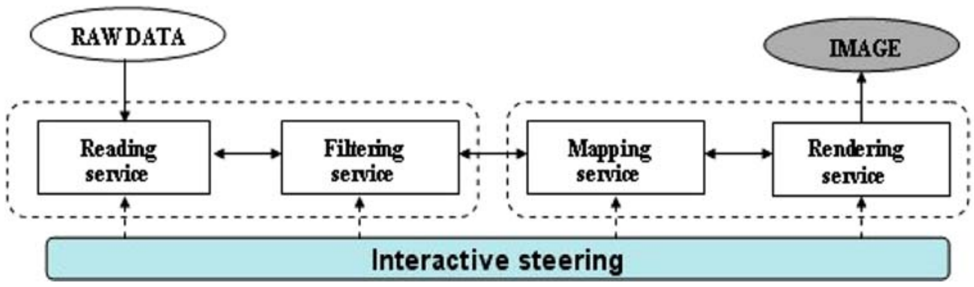

Fig. 2 A generic service-oriented visualization model

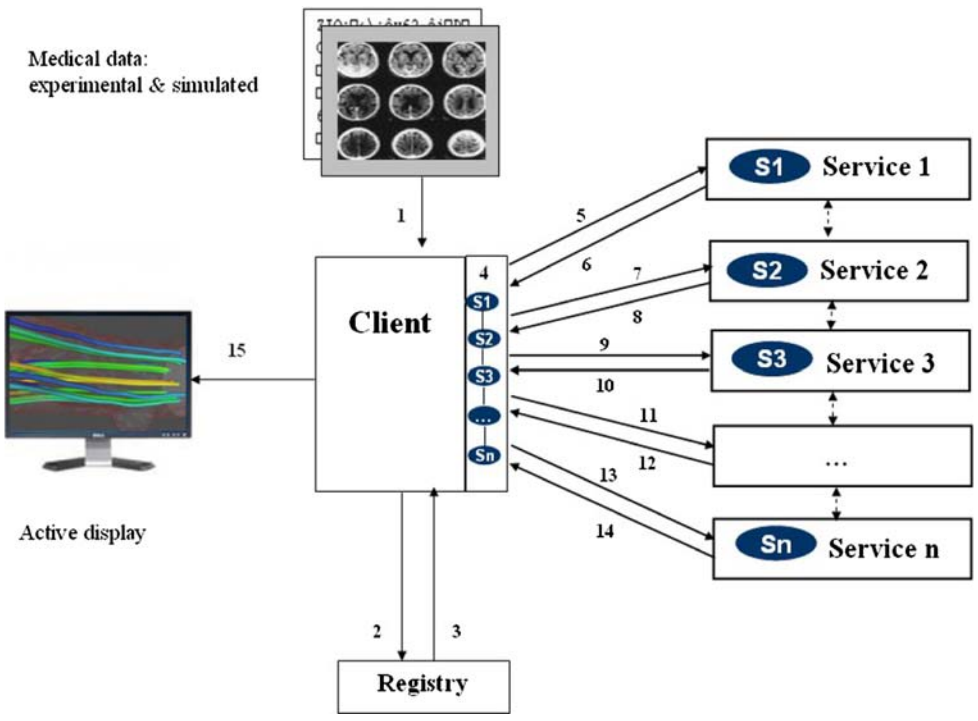

Fig. 3 Generic medical data analysis scenario

which quires a service registry for the location of service providers (2 and 3). Then using GUI client tools, the user composes a service-oriented visualization pipeline from Grid/Web services $S_1, S_2, \ldots, S_n$ provided by the service providers (4). The client then invokes services specified in a pipeline (5–14). The result of an upstream service is transmitted as the input of the next downstream service. Finally, the result of the last queried service is passed by the client to the image viewer running on the active display at the specified location and the user explores the final image to plan the future treatment of the patient (15).

As can be seen from the scenario above, service-oriented visualization allows end-users (e.g., medical specialists) to explore data visualisations irrespective of their geographical location and available computing resources. However, the major advent of a proposed architecture is its suitability for the work of visualization experts as well. Experts can develop visualization pipelines for end-users without a necessity of having local resources dedicated to each visualization task. Interactive steering makes it possible for experts to control every stage of a visualization pipeline (its input and output) and hence to find the optimal solution to a visualization problem much faster. Moreover, resource sharing provides nice opportunities for collaborative work, including collaboration between experts and end-users.

4 Architectural elements

An Interactive Visualization Framework (IVF) developed in this project supports a service-oriented visualization model and is aimed at assisting users in building distributed visualization pipelines and their execution on the Grid. IVF consists of the following main components: the client, visualization services and the central data repository.

The client plays a central role in the whole process (Figs. 3 and 4). The composition of a visualization pipeline takes place on the client and the communication between services goes through the client as well [29]. The client is able to find, locate and obtain necessary services and to broadcast events between services. The client transmits result of an upstream service as the input of the next downstream service. Essentially, the client is a thin client because all of the main processes are done by the services that it utilizes.

Services are responsible for transformational sub-process from the visualization pipeline shown in Fig. 2. Each service contains a visualization core and tools to interact with the client (handler in Fig. 4) and to access the data repository (GridFTP client in Fig. 4). Visualization services act independently without any knowledge about possible existence of other services. The service invocation is performed in a client-server manner as it is shown in Fig. 4.

Steering user interfaces to services are not in existence for the overall duration of the visualization process but can be invoked any time. A user may adjust visualization parameters (i.e., sample rate, region of interest, etc.) stop, pause or continue the execution of a running visualization service.

The central data repository (GridFTP server in Fig. 4) is responsible for hosting data from the client and intermediate data generated by visualization services. A centralized repository allows better data management and synchronisation compared to multiple repositories for each service [18].

A fragment of the IVF data flow is shown in Fig. 5. The client invokes a service by sending a message, which specifies the location of the data to be processed. The invoked service then takes this indicated data, processes it and stores results in a repository. After that, the service sends a message to the client specifying location of the new derived data. A service invoked next by the client takes this data from a repository, generates new results and stores them again in a repository. The process continues until the final image is rendered and displayed to the user.

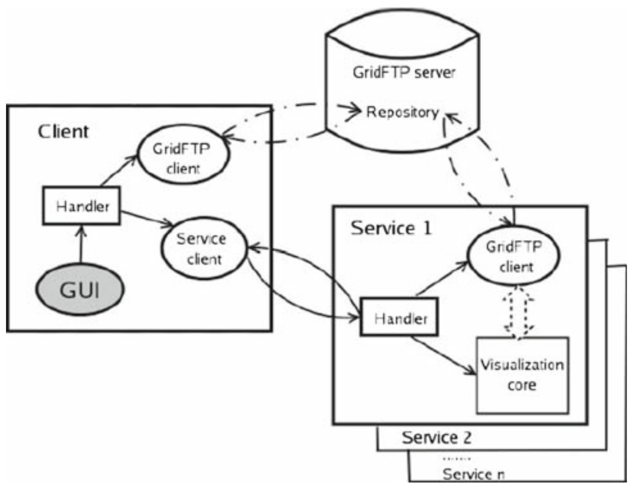

Fig. 4 Architectural design

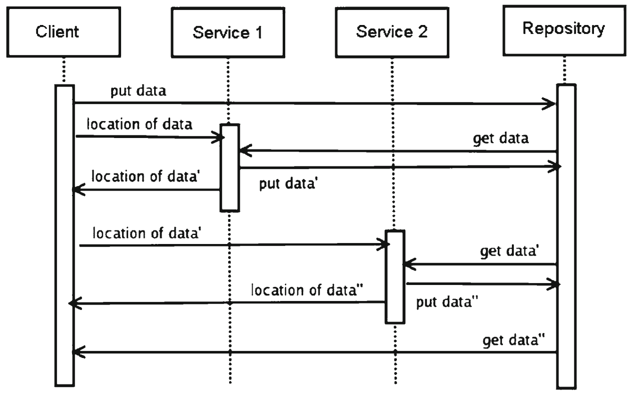

Fig. 5 An example of the data flow in an Interactive Visualization Framework

5 A visualization case study

The image-based analysis of vascular disorders served as a case study for this project. We focused on the visualization of both experimental and simulated medical datasets.

The experimental data about the patient's vascular condition can be obtained via medical imaging modalities [28] such as Computed Tomography (CT), Magnetic Resonance Imaging (MRI), Positron Emission Tomography (PET), etc. The choice of an imaging modality is usually determined by the structure or anomaly that needs to be observed given that some modalities are better suited for certain applications than others.

The volumetric data produced by imaging modalities can vary from relatively small (e.g., $64 \times 64 \times 32$ voxels) to very large (e.g., $2048 \times 2048 \times 1024$ voxels). In most cases, the data consists of a single scalar value per pixel but it is also possible to obtain three samples per pixel e.g., if the data is stored in the DICOM format [30]. Datasets produced by medical imaging modalities cannot be analyzed in their original numerical representations. A conversion to a visual representation needs to take place.

In this project, the iso-surface extraction technique has been applied to visualize the MRI data of the patient's vascular condition. It allows the representation of human vascular structures as surfaces (see Fig. 6a).

Surfaces are generally constructed using polygons as primitives. One of the most popular algorithms for the iso-surface extraction is the marching cubes algorithm [12]. In a 3D scalar field, cells are defined as cubes with eight vertices. The algorithm assumes a linear variation in each direction within the cell. The idea is to march through the domain cell by cell and determine whether the surface passes through the cell. According to the set of vertices of the cell that are inside or outside the surface, a lookup table with the possible combinations of how the surface passes through the cell is applied to determine the polygons to be used.

Visualization of the patient data can be enriched with the user interaction techniques. For instance, as it is shown in Fig. 6b, it is possible to interactively select a region of interest for the further thorough inspection, which may include simulation of specific blood flow parameters.

To model the microscopic properties of the blood flow while exploring it macroscopic properties, the lattice-Boltzmann method (LBM) is often applied. The main idea behind LBM is mapping the average motion of the fluid/blood particles on the lattice [2]. The LBM solver developed at the Section Computational Science of the University of Amsterdam allows to simulate the time-harmonic blood flow in the human vascular system and extracts the flow specific parameters (velocity, pressure and sheer stress) before and after a surgical intervention in order to predict its possible outcome.

The simulated flow data consists of vectors and scalars mapped to properties of visual objects such as location, orientation, size, colour and transparency. There exist several methods to visualize the flow data e.g., vector field visualization, animated particles, glyph visualization, streamline-based visualization, etc. [19].

In this project, streamline-based flow visualization has been applied to represent the simulated blood flow in a selected region of interest (Fig. 6c, d). Streamline-based visualization represents a set of paths traced out by massless particles as they move within a flow [17]. Particles follow the motion of the flow originating from one or more seed points that can be set manually by a user or pre-configured based on the known flow parameters. As the particle propagates, it leaves a trace called a streamline.

Streamlines are computed based on a numerical integration method [22], where the integration step together with the speed of flow determines the detail of the particle trace. Smaller steps result in more detailed streamlines. However, the achieved accuracy may cost additional computations and be very time consuming. On the other hand, when a big

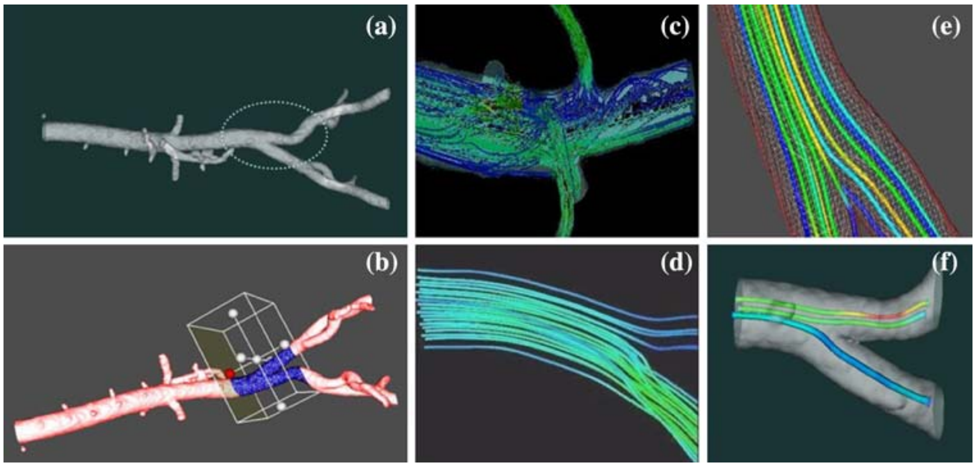

Fig. 6 Iso-surface extraction (*left*); streamline-based flow visualizations (*middle*); combined visualizations (*right*)

integration step is applied, this may cause the neglection of the areas of interest.

We have chosen the streamline-based visualization technique as it is one of the few flow visualization techniques where user interaction plays significant role [6]. The user can interactively change the number of streamlines, adjust their location and distribution radius and apply different colour palettes.

Using streamlines, the user (e.g., medical specialist) can investigate how the flow curves and whether it diverges or converges at specific points. To make it easier for users to analyze the behaviour of the blood flow, streamlines can be displayed in combination with the patient data where streamlines are positioned inside the arterial structures as it is shown in Fig. 6e, f. Also, to assist in analysis, a tube with rectangular shape can be wrapped around the path to improve the visibility of a streamline.

6 Visualization services

In this project, the actual visualization part has been performed using the Kitware Visualization Toolkit (VTK) 4.4.2 [35].

The VTK iso-surface extraction pipeline can be constructed as it is shown in Fig. 7, left. Because the patient specific data of vascular geometry is provided as a structured points VTK file, the iso-surface extraction pipeline starts from the reading sub-process handled by *vtkStructuredPointsReader* class. The purpose of *vtkStructuredPointsReader* is to ingest the initial data, create a derived data object and pass this object to the next pipeline module. *vtkContourFilter* class is the second module of the iso-surface extraction pipeline. This module generates iso-surfaces or iso-lines from the input dataset. Further on, the pipeline contains *vtkPolyDataMapper* class that maps polygonal data (passed by *vtkContourFilter*) to the graphics primitives, which are later processed by rendering software and hardware. *vtkRenderer* class is the last VTK module of the pipeline. This module is responsible for rendering graphics primitives into the final image (3D arterial structures).

The LBM solver simulates flow data as a structured points VTK file. Consequently, the streamline-based flow visualization pipeline (Fig. 7, right) starts from the reading sub-process handled by *vtkStructuredPointsReader*class. The derived data object is then passed to *vtkStreamLine* filter, which generates streamlines for an arbitrary dataset. By adjusting the integration step parameter, a time increment for the generation of streamline seed points can be changed. In order to yield an interesting result, the position of each streamline has to be properly defined. *vtkPointSource* class serves this purpose. *vtkPointSource* generates random points in a sphere space with the specified center, radius and number of seed points. *vtkTubeFilter* is an additional filter applied to generate tubes wrapped around streamlines. Next, the pipeline contains *vtkPolyDataMapper* and *vtkRenderer*. These two VTK classes serve similar purposes as explained for the iso-surface extraction pipeline. *vtkLookupTable* class is used to apply the colour palette to streamtubes. In this project, we mainly experimented with the blood flow velocity. In Figs. 6, 8 and 9, red colour corresponds to the highest velocity values, blue colour—to the lowest ones.

As stated earlier in this paper, any visualization sub-process or group of sub-processes can be transformed into an independent Grid/Web service. To perform this transformation, main conceptual blocks need to be defined for each visualization pipeline shown in Fig 7. These blocks can then be transformed into visualization services.

The VTK iso-surface extraction pipeline can be divided into two major blocks (see Fig. 7, left): Surface-reader

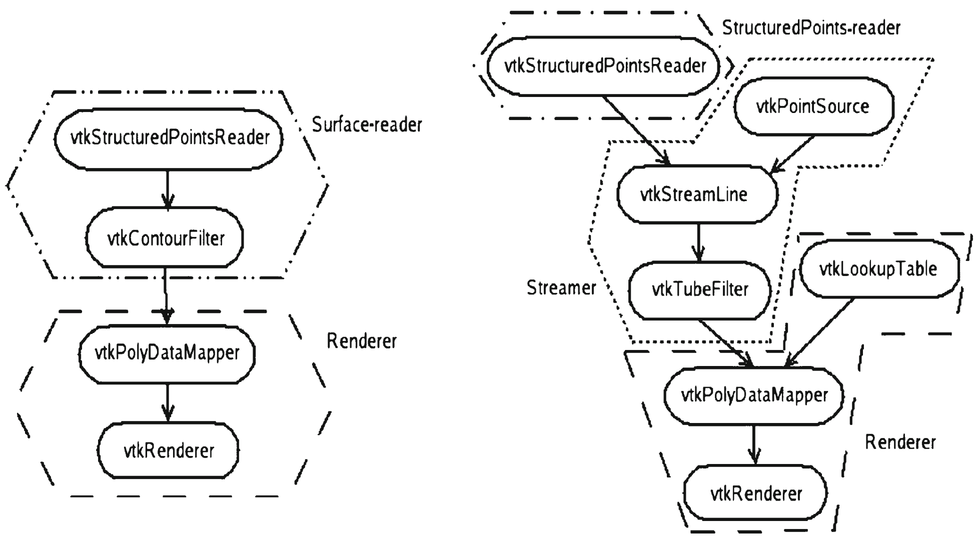

Fig. 7 From traditional pipelines to visualization services (*left*) iso-surface extraction; (*right*) streamline-based flow visualization

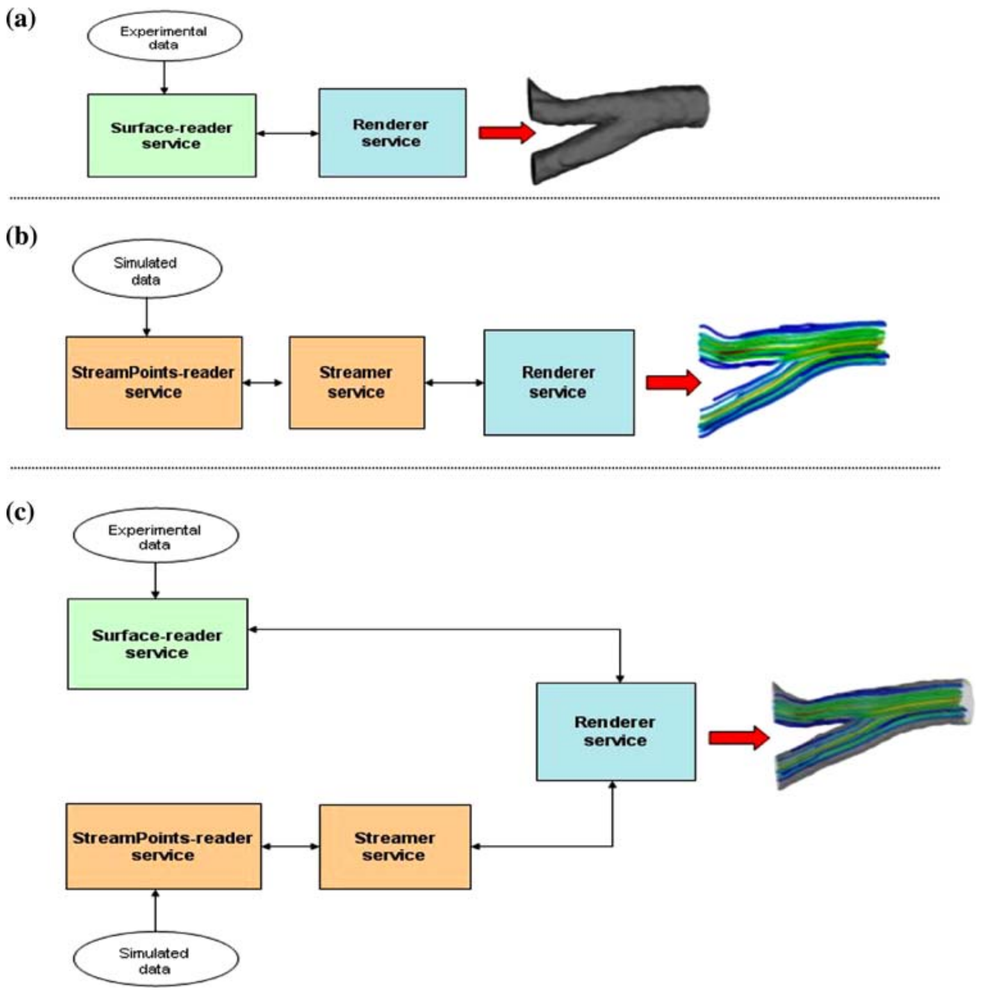

Fig. 8 Service-oriented visualization pipelines: **a** iso-surface extraction pipeline; **b** streamline-based visualization pipeline; **c** combined visualization pipeline

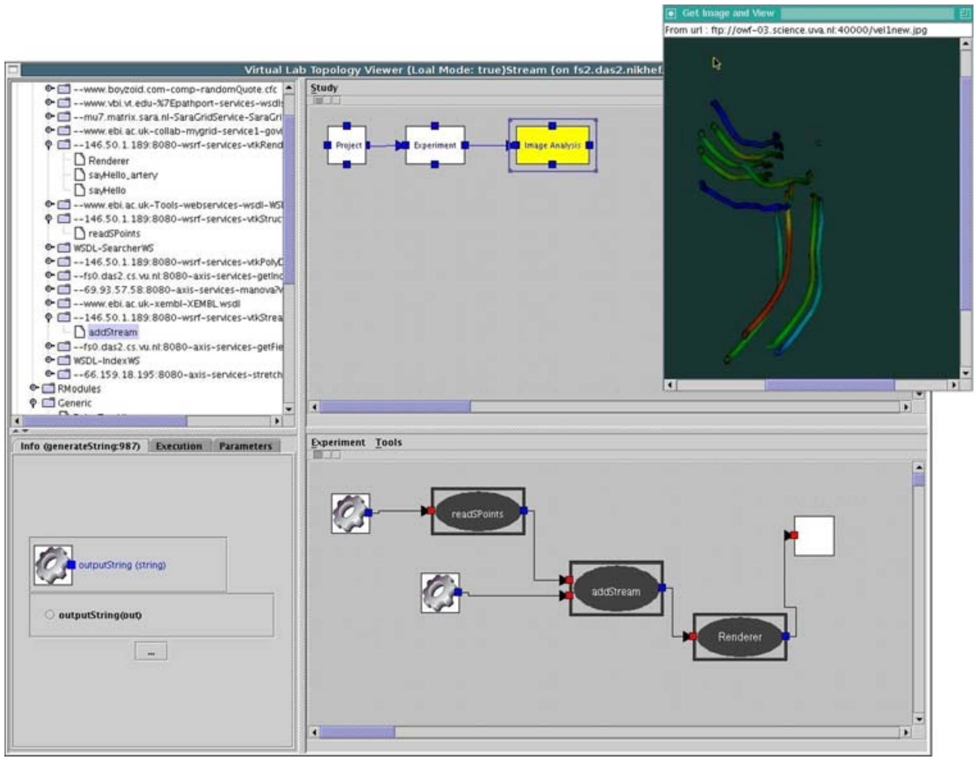
Virtual Lab Topology Viewer (Loal Mode: true)Stream (on fs2.das2.nikhef
--www.boyzoid.com-comp-randomQuote.cfc
--www.vbi.vt.edu-%7Epathport-services-wsdl
--mu7.matrix.sara.nl-SaraGridService-SaraGri
--www.ebi.ac.uk-collab-mygrid-service1-gov
--146.50.1.189:8080-wsrf-services-vtkRend
Renderer
sayHello_artery
sayHello
--www.ebi.ac.uk-Tools-webservices-wsdl-WS
--146.50.1.189:8080-wsrf-services-vtkStruc
readSPoints
WSDL-SearcherWS
--146.50.1.189:8080-wsrf-services-vtkPoly
--fs0.das2.cs.vu.nl:8080-axis-services-getInc
--69.93.57.58:8080-axis-services-manova
--www.ebi.ac.uk-xembl-XEMBL.wsdl
--146.50.1.189:8080-wsrf-services-vtkStrea
addStream
--fs0.das2.cs.vu.nl:8080-axis-services-getFie
WSDL-IndexWS
--66.159.18.195:8080-axis-services-stretch
RModules
Generic
Study
Project
Experiment
Image Analysis
Info (generateString:987)
Execution
Parameters
outputString (string)
outputString(out)
...
Experiment
Tools
readSPoints
addStream
Renderer
Get Image and View
From url : ftp://owf-03.science.uva.nl:40000/vel1new.jpg

Fig. 9 Composition of the distributed streamline-based visualization pipeline and the final result

and Renderer. Surface-reader consists of *vtkStructured PointsReader* and *vtkContourFilter* classes. This block is responsible for reading a raw dataset and for the extraction of iso-surfaces from this dataset. Renderer contains *vtkPolyDataMapper* and *vtkRenderer* classes. This block is responsible for mapping graphics primitives into an image and for rendering the final image. To save derived polygonal data as an intermediate dataset, we wrapped Surface-reader with *vtkPolyDataWriter* class and enriched Renderer with *vtkPolyDataReader*class to be able to read from the intermediate dataset generated by Surface-reader. These allow visualization blocks to be executed independently (on different hosts) and perform the same functionality as if they are merged.

In a similar way, the VTK streamline-based flow visualization pipeline can be split into three major blocks (see Fig. 7, right): StructuredPoints-reader, Streamer and Renderer. The first block contains *vtkStructuredPointsReader* class. The second one consists of *vtkStreamLine*, *vtkTubeFilter* and *vtkPointSource*. The last one includes *vtkPolyDataMapper* and *vtkRenderer* classes. It has the same name and serves the same functionality as the Renderer block from the iso-surface extraction pipeline. Additional VTK reader and writer objects were embedded into each block to serve as interfaces between the pipeline and intermediate datasets.

In total, four visualization services have been developed in this project as part of the IVF framework:

(1) The Surface-reader service is responsible for reading and filtering sub-processes[2] in the iso-surface extraction pipeline;
(2) The StructuredPoints-reader service is responsible for the reading sub-process in the streamline-based flow visualization pipeline;
(3) The Streamer service is responsible for the filtering sub-process in the streamline-based flow visualization pipeline;
(4) The Renderer service is responsible for mapping and rendering sub-processes[3] (applicable to both pipelines).

We kept block names from Fig. 8 to represent services. The aforementioned Grid/Web visualization services can be used for the construction of the following distributed pipelines: the iso-surface extraction pipeline (Fig. 8a), the streamline-based flow visualization pipeline (Fig. 8b) and the combined visualization pipeline (Fig. 8c).

[2] Reading and filtering were combined to avoid additional data transfer, which for the iso-surface extraction is more time-consuming than when both sub-processes run on the same logical machine.

[3] Mapping and rendering were combined due to their tight coupling in the VTK implementation.

7 Implementation

We now discuss our implementation and reasons behind the selection of various tools.

To deploy the IVF framework, we choose the Globus Toolkit 4 (GT4). The Globus Toolkit [31] has been developed to support the development of service-oriented distributed computing applications and infrastructures. GT4 represents a significant advance compared to the GT3 implementation of Web services functionality. It includes a Java WS core component that provides Grid/Web services container as a hosting environment for GT4 standard services and customized services of users.

In this project, the Web Services Resource Framework (WSRF) has been implemented via the GT4 Java WS Core. The Simple Object Access Protocol (SOAP) [7] is used as a communication protocol. Data transfer is performed via GridFTP [32]. We have decided to apply GridFTP instead of the traditional HTTP protocol because GridFTP is more efficient for the transfer of large datasets.

We choose to implement and advertise Grid/Web visualization services through Web Service Definition Language (WSDL) documents. When registered with a Universal Description Discovery and Integration (UDDI) server, a WSDL document enables remote users to find publicly available visualization resources and to connect automatically. Steering interfaces to services and the IVF client have been implemented in Java. We selected Java as this can run cross-platform without modification, and the majority of the Grid tools are implemented in Java. A GUI to the client is developed using Java Swing package.

The implementation also involved the use of two external libraries such as VTK and Mesa 3D graphics library. The use of VTK 4.4.2 highlighted some limitations of this system. For instance, VTK needs to load data from a local file. This problem was overcome by copying the remote output to a local file. Unfortunately, this solution requires the remote process to have completed before the service using VTK can start.

To allow users to construct interactively service-oriented visualization pipelines in an intuitive and flexible manner, we integrated the IVF framework with the VLAM-G (Grid-based Virtual Laboratory AMsterdam [27]) at a later stage of the project. VLAM-G is a Grid enabled problem-solving environment developed within the Virtual Laboratory for e-Science project [34]. VLAM-G allows scientific programmers to develop scientific portals where computing resource access details are hidden from end-users (scientists). These portals provide access to the user interfaces to various scientific studies e.g., information gathering, data processing, visualization, interpretation and documentation.

VLAM-G is developed under the assumption that all available computing resources have the Grid middleware installed

(Globus [31]) and have inbound and outbound communications that allow to create the data streams between the workflow processes located on geographically distributed Grid resources. VLAM-G provides generic tools for managing data and software resources for supporting scientific workflows.

Workflow technologies are emerging as the dominating approach to coordinate groups of distributed services to achieve a shared task or goal [27]. In the VLAM-G workflow, modules are the basic construction blocks. Each module is responsible for the performance of a specific computational task and consists of the processing and service parts. The processing part is the application code, which implements the program logic. The service part provides facilities for transferring the data between modules and supports the run time control interface over these modules.

VLAM-G consists of seven principle components (see Fig. 10 [27]): the graphical user interface (GUI), Process Flow Template (PFT) Assistant, Collaboration System, Session Manager, Module Repository, Run Time System (RTS) and Virtual laboratory Information Manager for Cooperation (VIMCO).

The GUI is the only part of VLAM-G that the user interacts directly with. Using the VLAM-G GUI, the user can create his/her own experiment (Fig. 9) by connecting output of one module with the input(s) of other module(s). For this, a module can be simply dragged from a list of available modules and dropped into the composition window to form a workflow topology. At run time, this workflow topology will be executed on the underlying Grid infrastructure (Fig. 11). The connection validity for each pair of modules is checked automatically prior to the execution. The PFT Assistant adds to the process by suggesting the user appropriate modules or experimental topologies.

Each module in the VLAM-G framework is provided with a description file in the Module Description Format [25],

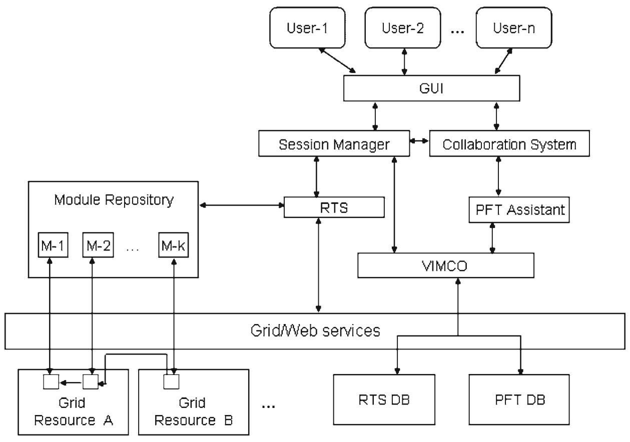

Fig. 10 VLAM-G architecture adapted from [27]

which contains information about module resource requirements also called as quality of service requirements (QoS). Furthermore, VLAM-G can import existing Grid/Web services as modules if access to their WSDL files is provided.

When the experiment is ready to be executed it is passed in XML form via the Session Manager to RTS. The Session Manager makes it possible to the user to work with multiple studies (execute one study while preparing another one) and provides a single access point to VIMCO (preparation) and RTS (execution).

VIMCO provides access to the study and experiment descriptions that are stored in application specific databases. RTS takes care of scheduling, instantiating and monitoring computational modules from the experiment defined. It makes extensive use of Globus services to perform these tasks. The Collaboration System offers audio and video communication, an electronic whiteboard and access to other collaborative facilities enabling cooperation between scientists participating in the experiment.

To allow the IVF visualization services to be controlled from VLAM-G, we modified all WSDL files. In particular, the conversion from the document-literal encoding to the Remote Procedure Call literal encoding has been performed to allow the automatic generation of client stubs, which are required by the VLAM-G RTS Manager to be able to access the IVF visualization services. The RTS Manager is a GT4-based service that uses standard security and notification interfaces provided by GT4. The notification mechanism allows the user to monitor progress of the pipeline execution.

Two service-oriented visualization pipelines composed using VLAM-G tools are shown in Figs. 9 and 11 along with the results of their successful execution. More detailed information about VLAM-G can be found in our earlier publications [1,25,27].

8 Experiments and results

We mainly experimented with the distributed streamline-based flow visualization pipeline. The heterogeneous computing resources have been used for performing experiments. Available hardware configurations included:

(1) Intel Pentium 4 CPU 2.80 GHz, 512 MB RAM, graphics card: 64 MB NVIDIA GeForce4 MX 440 AGP 8x
(2) Intel Xeon CPU 3.0 GHz, 4128 MB RAM, graphics card: 256 MB NVIDIA Quadro FX 3000 (we experimented with both on-screen and off-screen rendering)

In total, three experiments have been conducted. The aim of the first experiment was to verify whether Grid/Web services responsible for different visualization sub-processes had different computational demands.

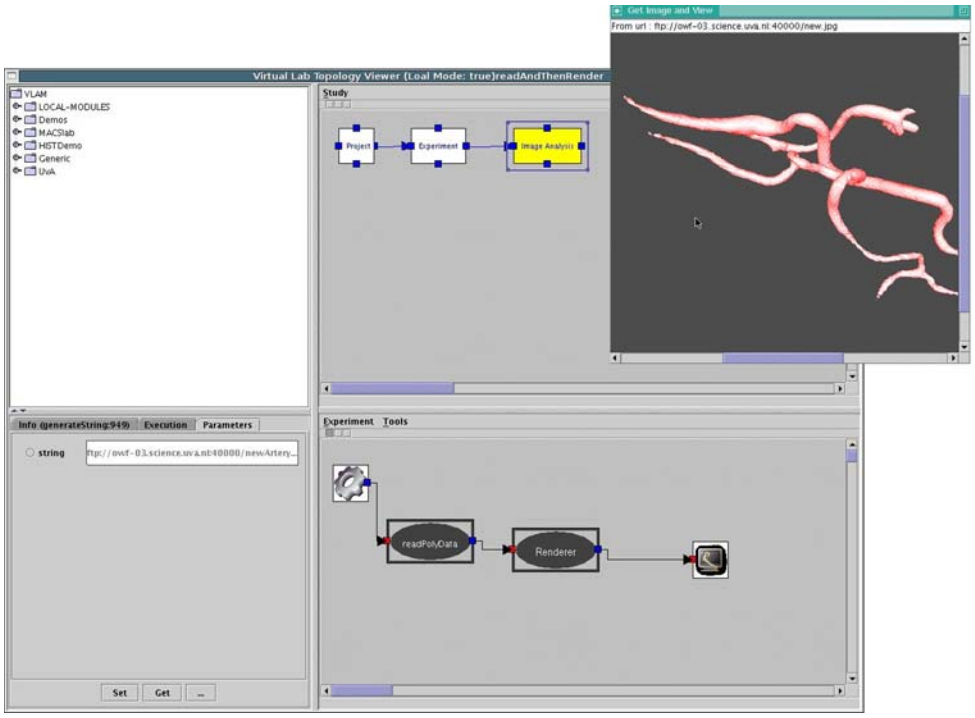

Fig. 11 Execution of the distributed iso-surface extraction pipeline and the final result

Execution time[4] of three visualization services (Stream Points-reader, Streamer and Renderer) was measured and compared depending on the size of an original raw dataset: 1 MB (81.874 data points), 30 MB (1.314.648 data points) and 70 MB (2.495.004 data points). Each dataset contained results of the LBM simulation performed for a specific region of interest.

Overall, the first hypothesis was well supported by the experimental data. As it is shown in Fig. 12a, the Stream Points-reader service requires the least processing power compared to the Streamer and Renderer services. Still, the Renderer service remains the most time consuming compared to the Streamer service. Consequently, the Renderer service should be assigned to the most powerful computing resource.

With the increase in the data size, the need in heterogeneous resources becomes more evident. It can be seen in Fig. 12 that the deviation in execution time measured for two different hardware configurations becomes essential when relatively big datasets ($\geq$ 30 MB) are processed.

In the described experimental setting, both configurations were equally well suited to perform light-weight filtering and mapping-rendering tasks. Hence, there is no need to assign the Renderer service to a more powerful computing resource if the size of an original raw dataset is $\sim$1 MB (Fig. 12c). The same concerns the Streamer service (Fig. 12b).

[4] Execution time is time taken by a Grid/Web service to process its sequence of activities.

As for heavy-weight tasks, when the size of a simulated dataset is more than 30 MB, for both the Streamer and Renderer services the difference in execution time becomes noticeable (Fig. 12b, c). However, the Renderer service still remains the most time consuming and the deviation gets even stronger when a 70 MB dataset has to be processed. Thus, in this last situation, it would be beneficial to have the Renderer service assigned to a more powerful resource.

In the second experiment, we investigated the impact of the visualization parameter "number of streamlines" on the choice of a computing resource for the Renderer service to be assigned to. Our initial hypothesis was that this parameter would have a significant influence. However, it was unclear to us prior to the experiment whether this impact would vary depending on the size of a raw dataset.

Time required to map and render different number of streamlines is shown in Fig. 13. We compared mapping-rendering time of the same VTK structured points datasets of three different sizes as in the first experiment. For better comparison, different integration steps have been applied to each dataset: 1 - to a 1 MB dataset, 100 - to a 30 MB dataset and 200 - to a 70 MB dataset.

According to Fig. 13, a more powerful computer (with the graphics card NVIDIA Quadro FX 3000) can perform mapping-rendering faster. The difference in performance becomes visible when the number of streamlines reaches the range of 40–50 for relatively big datasets ($\geq$ 30 MB) and 60–70 for small datasets ($\sim$1 MB).

Furthermore, mapping-rendering time is much lower when off-screen rendering on a computer with a more powerful

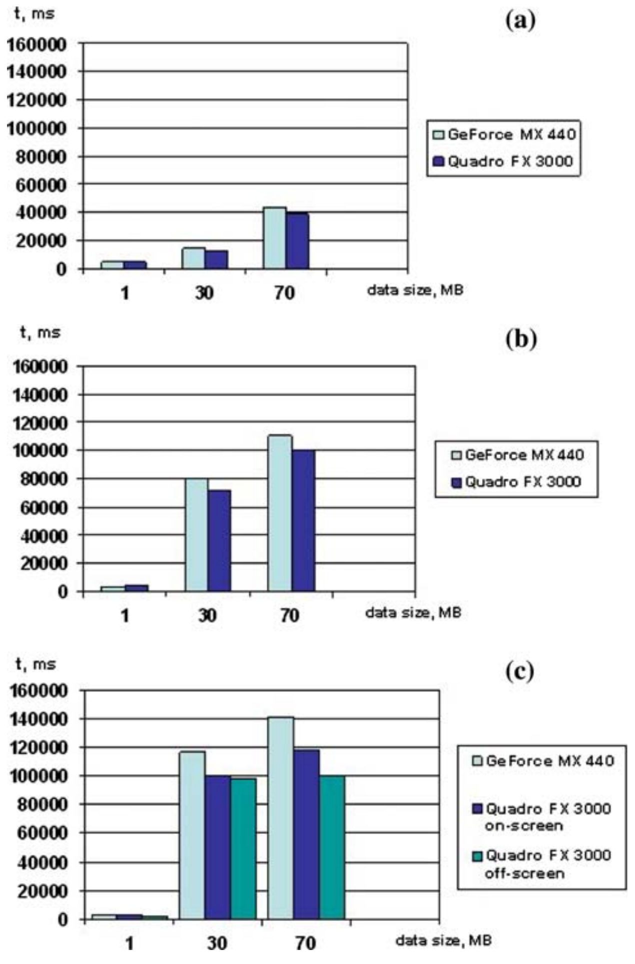

Fig. 12 Execution time of visualization services (integration step—150): **a** SreamPoints-reader; **b** Streamer; **c** Renderer

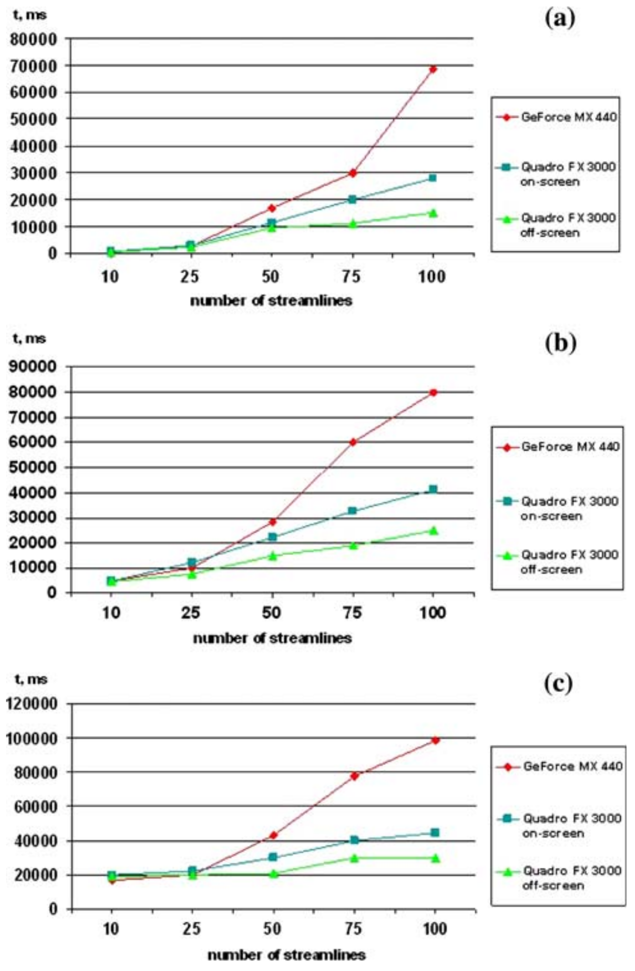

Fig. 13 Mapping-rendering time required to generate different number of streamlines: **a** 1 MB dataset, integration step 1; **b** 30 MB dataset, integration step 100; **c** 70 MB dataset, integration step 200

graphics card is applied compared to on-screen rendering performed on the same machine. The reason lies in the implementation of the VTK library. During on-screen rendering data has to be moved from the main RAM to the graphics card RAM, while during off-screen rendering main RAM is used directly via the Mesa API [35]. As can bee seen in Fig. 13, the time difference becomes significant when the number of streamlines reaches 75 (irrespective to the size of an original dataset).

Overall, the results of the second experiment do not contradict with our hypothesis. Mapping-rendering time is strongly affected by the number of streamlines to be visualized. In the described experimental setting, a more powerful graphics station becomes beneficial when the number of streamlines reaches 50. However, performance gained also depends on the size of an original structured points dataset. Hence, when assigning the Renderer service to a dedicated computing resource, it is important to take both criteria into account. In this particular implementation, off-screen rendering is preferable over on-screen rendering.

In the final third experiment, we measured round-trip time of the execution of the distributed streamline-based flow visualization pipeline (Figs. 14 and 15) and of the execution of the distributed iso-surface extraction pipeline (Fig. 16). We considered round-trip time as the elapse time from the moment the IVF client starts the transfer of a raw dataset to the moment the client receives the final image. We used homogeneous computing resources of the configuration 1 to perform this experiment.

Total round-trip time has been broken down to the following components:

- data reading and writing, including intermediate data objects (“R/W”);
- data transfer via the GridFTP protocol (“GridFTP”);
- SOAP-messaging (“SOAP”);
- other related issues e.g., loading of the VTK and Mesa libraries (“Others”)

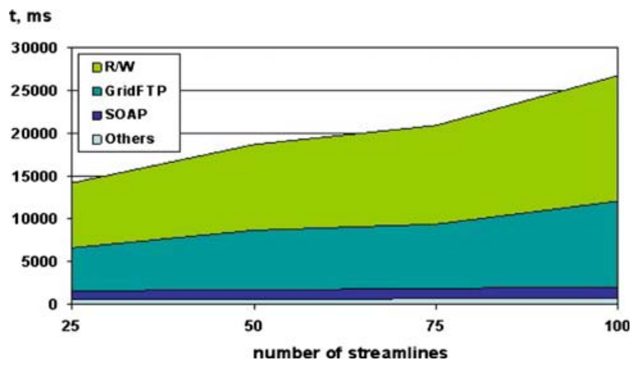

Fig. 14 Breakdown of round-trip execution time of the distributed streamline-based flow visualization pipeline for varying number of streamlines (70 MB dataset, integration step 200)

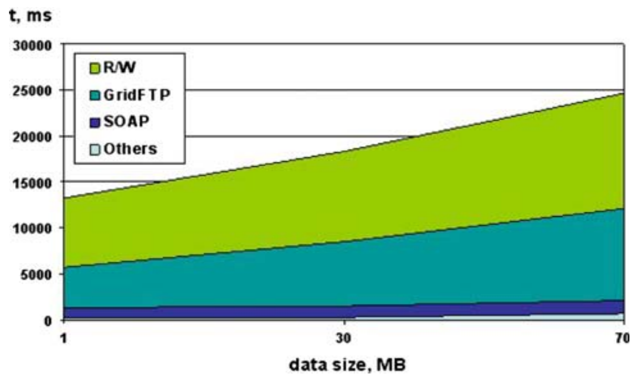

Fig. 15 Breakdown of round-trip execution time of the distributed streamline-based flow visualization pipeline for varying dataset size (integration step 150)

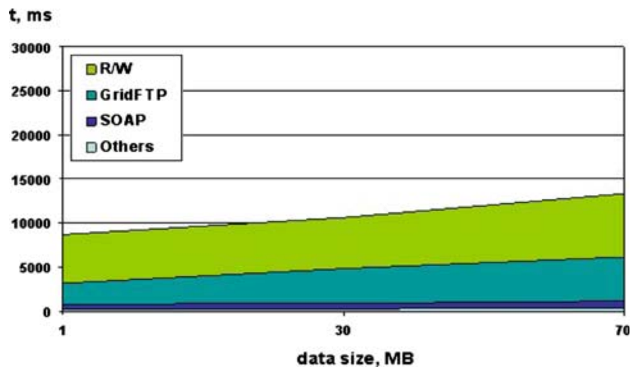

Fig. 16 Breakdown of round-trip execution time of the distributed iso-surface extraction visualization pipeline for varying dataset size

As can be seen from the charts, the most essential contributions to round-trip time are from the "R/W" and "GridFTP" components, which grow with the increase in the number of streamlines to be visualized (Fig. 14) and with the increase in the size of an original dataset (Figs. 15 and 16).

These results can be explained as follows. In the distributed streamline-based flow visualization pipeline (Fig. 14), the StreamPoints-reader and Streamer services generate intermediate data objects. Sizes of these intermediate objects vary depending on such parameters as the integration step and number of streamlines. Sometimes sizes of intermediate datasets can exceed the size of an original raw data by an order of magnitude. Consequently, the time required for reading/writing increases with big numbers of streamlines.

On the other hand, the sizes and number of generated intermediate objects as well as the size of the final image are strongly dependent on a selected region of interest (raw dataset). Hence, with the growth of original data, the time required for reading/writing increases as well. As can be seen in Figs. 15 and 16, this applies to both distributed pipelines.

As it was explained in Sect. 4, intermediate data objects are stored in a centralized data repository and retrieved from this repository upon request of a visualization service. This is achieved through the GridFTP protocol. The time spent on the data transfer via GridFTP depends on the size of the data to be transferred. Consequently, the contribution to round-trip time is strongly affected by both the size of an original dataset and by the visualization parameters applied (i.e., number of streamlines).

In the distributed iso-surface extraction pipeline however the contributions of reading/writing and the GridFTP transfer are much less compared to the distributed streamline-based pipeline. This happens because within the distributed streamline-based pipeline intermediate data objects are generated twice (by the StreamPoints-reader service and by the Streamer service) whilst within the iso-surface pipeline it is done only once (by the Surface-reader service).

As for the SOAP-messaging, its contribution to round-trip time remains constant. SOAP messages contain only control data (i.e., reference information and visualization parameters) sent over the network. These data do not change with the increase in the number of streamlines or with the growth of a raw data.

9 Discussion and conclusions

Traditional visualization models do not fully exploit features and benefits of distributed computing oriented mostly toward service-oriented architectures such as WSRF, Open Grid Services Architecture (OGSA), Globus [7,8,31], etc. To facilitate user-friendly interactivity and motivated resource allocation on the Grid, new visualization models need to be developed and applied.

Having this in mind, we designed and implemented an Interactive Visualization Framework (IVF) and demonstrated that a distributed service-oriented architecture across the Grid is applicable to the area of medical data visualization.

The IVF framework supports a service-oriented visualization model. This model originates from the traditional visualization pipeline of Harber and McNabb [9] and suggests that intermediate transformation visualization sub-processes included into this pipeline can be deployed as independent Grid/Web services. A service-oriented architecture allows interactive user steering over distributed visualizations running on the Grid and provides possibilities for assigning heterogeneous computing resources to different visualization sub-processes.

IVF is suitable for the work of both visualization experts and end-users (e.g., medical specialist). It allows experts to build visualization pipelines for complex data analysis avoiding a need of having local resources dedicated to each visualization task. Moreover, steering of each intermediate stage of a pipeline enables experts to experiment with different visualization schemes for finding an optimal solution to the visualization problem. As for end-users, they are able to explore visualizations irrespective of their geographical location and available computing resources.

The image-based analysis of vascular disorders served as a case study for this research project. We experimented with two medical datasets: the experimental data of a patient's vascular condition and the simulated blood flow data, which can be used to predict the outcome of a vascular reconstruction procedure.

IVF provides a collection of independent Grid/Web services that can be used to visualize these data. Each service is responsible for a specific visualization sub-process. To allow interactive composition of distributed pipelines from available visualization services, IVF has been integrated with the VLAM-G environment for scientific workflow management [25]. Using the GUI provided by VLAM-G, the user can browse and select among available services and check the connection validity between selected services. Three distributed visualization pipelines can be constructed from the Grid/Web services developed in this project: the iso-surface extraction pipeline, the streamline-based flow visualization pipeline and the combined visualization pipeline.

Using IVF, medical specialists are no longer restricted by hardware and software limitations and can perform image-based analysis on very large datasets. IVF allows the high-quality interactive visualizations to be delivered to the physically remote users, whose local resources would be otherwise overwhelmed.

Spreading visualization sub-processes over the Grid as independent services assigned to heterogeneous computing resources considering their complexity leads to the better efficiency in the execution of visualization tasks. Moreover, the motivated resource allocation allows to avoid the under utilization of certain machines for the time longer than actually visualization runs, which is normally the case when all visualization modules are linked together (in traditional visualization models).

Performance measurements indicate that different computing resources should be assigned to reading, filtering and mapping-rendering services. In particular, the two last services require more powerful machines and faster graphics cards. Experimental data shows that when rendering combined with mapping is assigned to a powerful graphics station, much better performance can be achieved. In addition, visualization parameters (e.g., the number of streamlines) and the size of an original raw dataset have significant impact on computational needs related to each stage of the visualization pipeline.

We do not argue that our approach provides the most efficient resource allocation scheme. But it is definitely a valuable solution when a distributed environment contains heterogeneous computer systems.

Currently IVF supports only visualization services developed in this project. However, our ultimate goal is to allow functionalities from different vendors to be brought together and combined within a single pipeline. This will allow any individual component in a pipeline to be swapped by an alternative one, which is faster or cheaper or simply more reliable. Another open question is how to organize multiple steering in such a way that each user may hold the information about his/her visualization pipeline separately.

Next to this, we would like to experiment with different visualization front-ends. The developed infrastructure allows diverse equipment to be utilized, including stereoscopic and auto-stereoscopic immersive systems. Therefore, with regard to the display diversity, the next step of this project will be to investigate possibilities to generate visualizations in an adaptive manner based on display parameters and the viewing situation (for auto-stereoscopic systems [4,16]).

Acknowledgments This research is partially funded by the NWO/VIEW "A Multi-modal Visualization Environment for Interactive Analysis of Medical Data" project, the BSIK "Virtual Laboratory for e-Science" project and the NWO/Token 2000 "Distributed Interactive Medical Exploratory for 3D Images" project.

Open Access This article is distributed under the terms of the Creative Commons Attribution Noncommercial License which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

References

1. Afsarmanesh H et al. (2003) VLAM-G, A grid-based virtual laboratory. Int J Future Gener Comput Syst 19:1167–1176
2. Axner L, Hoekstra AG, Sloot PMA (2007) Simulating time harmonic flows with the lattice Boltzmann method. Phys Rev E 75, 3, 036709, pp 1–7
3. Charters SM, Holliman NS, Munro M (2004) Visualisation on the grid: a web service approach. In: Proceedings of the UK eScience third All-Hands Meeting, Nottingham Conference Center, 31 August–3 September 2004
4. Charters SM, Holliman NS, Munro M (2003) Visualisation in e-Demand: a grid service architecture for stereoscopic visualisation. In: Proceedings of the. UK eScience Second All-Hands Meeting, Nottingham Conference Center, 2–4 September 2003

5. Chi EH (2000) A taxonomy of visualisation techniques using the data state reference model. In: Proceedings of the IEEE symposium on information visualisation, pp 69–75
6. Fayyad U, Grinstein GG, Wierse A (2002) Information visualisation in data mining and knowledge discovery. Morgan Kaufmann, San Francisco
7. Foster I, Kesselman C (1998) The grid: blueprint for a new computing infrastructure. Morgan Kaufmann, San Francisco
8. Foster I, Kesselman C, Nick J, Tuecke S (2002) The Physiology of the Grid: an open grid services architecture for distributed systems integration. Open Grid Service Infrastructure WG, Global Grid Forum
9. Haber RB, McNabb DA (1990) Visualisation idioms: a conceptual model for scientific visualisation systems. Visualisation in scientific computing. IEEE Press, Los Alamitos
10. Heinzlreiter P, Wasserbauer A, Baumgartner H, Kranzlmueller D, Kurka G, Volkert J (2002) Interactive virtual reality volume visualisation on the grid. In: Proceedings of DAPSYS 2002, Linz, Austria, pp 90–97
11. Houstis E, Gallopolous S, Rice JR, Bramley R (2000) Enabling technologies for computational science: frameworks, middleware, and environments. Kluwer, Boston
12. Kaufman A (1994) Trends in visualisation and volume graphics, scientific visualisation advances and challenges. IEEE Computer Society Press, Melbourne
13. Keim DA (2001) Visual exploration of large data sets. Commun ACM 44(8):38–44. doi:10.1145/381641.381656
14. Kranzlmueller D, Kurka G, Heinzlreiter P, Volkert J (2002) Optimizations in the grid visualisation kernel. In: Proceedings of PDIVM 2002, Workshop on parallel and distributed computing in image. Processing, Video Processing and Multimedia, IPDPS 2002 Symposium, Ft. Lauderdale, FL
15. Montagnat J et al (2004) Medical images simulation, storage, and processing on the European datagrid testbed. J Grid Comput 2:387–400. doi:10.1007/s10723-004-5744-y
16. Jones GR, Lee D, Holliman NS, Ezra D (2001) Controlling perceived depth in stereoscopic images. Stereoscopic displays and virtual reality systems VIII. San Jose, California, SPIE, vol 4297A
17. Raskin J (2000) The humane interface: new directions for designing interactive systems. Addison-Wesley, Reading
18. Stanton J, Newhouse S, Darlington J (2002) Implementing a scientific visualisation capability within a grid enabled component framework. In: The 8th international Euro-Par conference. Lecture notes in artificial intelligence, vol 2400, pp 185–193
19. Sloot PMA et al. (2003) Grid-based interactive visualisation of medical images. In: Norager S (ed) Proceedings of the first European HealthGrid conference. Commission of the European communities, Information Society Directorate-General, Brussels, Belgium, pp 57–66
20. Tirado-Ramos A et al. (2004) Integration of blood flow visualisation on the grid: the FlowFish/GVK approach. In: Proceedings of the 2nd European AcrossGrids conference, Nicosia, Cyprus, LNCS 3165, pp 77–79
21. Tory M, Möller T (2004) Rethinking visualisation: a high-level taxonomy. In: IEEE symposium on information visualisation, pp 151–158
22. Toro EF (1997) Riemann solvers and numerical methods for fluid dynamics and a practical introduction. Springer, Berlin
23. Upson C et al (1989) The application visualisation system: a computational environment for scientific visualisation. IEEE Comput Graph Appl 9(4):30–44. doi:10.1109/38.31462
24. VIPAR—Visualisation In PARallel. http://www.man.ac.uk/MVC/research/vipar/ (2005)
25. Wibisono A, Korkhov V, Vasunin D, Guevara-Masis V, Belloum A, de Laat C et al (2007) WS-VLAM: towards a scalable workflow system on the Grid. In: Proceeding of the 16th IEEE international symposium on high performance distributed computing, 27–29 June 2007. Monterey Bay, California
26. Wood J, Brodlie K, Walton J (2003) gViz—visualisation and steering for the grid. In: Proceedings of the UK All Hands Meeting 2003, Nottingham
27. Zhao Z, Belloum ASZ, Wibisono A, Terpstra F, de Boer PT, Sloot PMA et al (2005) Scientific workflow management: between generality and applicability. In: Proceedings of the international workshop on Grid and peer-to-peer based workflows in conjunction with the 5th international conference on quality software, pp 357–364. IEEE Computer Society Press, Melbourne, 19–21 September 2005
28. Zudilova EV, Sloot PMA (2005) Bringing combined interaction to a problem solving environment for vascular reconstruction. Int J Future Gener Comput Syst 21(7):1167–1176. doi:10.1016/j.future.2004.04.004
29. Zudilova-Seinstra EV, Yang N (2005) Towards service-based interactive visualization. In: Proceedings of international symposium on ambient intelligence and life, 21 and 22 July 2005. San Sebastian, Spain, pp 15–25
30. DICOM (Digital Imaging and Communication in Medicine). http://medical.nema.org
31. Globus Alliance. http://www.globus.org
32. GridFTP protocol. http://dev.globus.org/wiki/GridFTP
33. Explorer IRIS. http://www.nag.co.uk/Welcome_IEC.html
34. Virtual Laboratory for e-Science. http://www.vl-e.nl
35. The Kitware Visualisation Toolkit. http://www.vtk.org